XY-pic Complete Sources with TeXnical Commentary

Kristoffer H. Rose ⟨Kristoffer.Rose@ens-lyon.fr⟩[1]
Ross Moore ⟨ross@mpce.mq.edu.au⟩[2]

XY-pic version 3.6 ⟨1998/03/06⟩

[1]Ecole Normale Supérieure de Lyon (LIP), 46 Allée d'Italie, F–69364 Lyon, France.
[2]MPCE (Mathematics dept.), Macquarie University, North Ryde, Sydney, Australia NSW 2109.

Printed October 12, 1998, from `xysource.man` version 3.3.

This report is typeset from the XY-pic sources, kernel version 3.6 released March 6, 1998. It includes all of the text in the XY-pic Reference Manual [16].

XY-pic related files can be retrieved by anonymous ftp from Internet host `ftp.ens-lyon.fr` in the directory `/pub/users/LIP/krisrose/TeX` in files with prefix `xy` as well as from CTAN hosts in `macros/generic/diagrams/xypic`. News about XY-pic can be obtained from the authors and through the 'XY-pic World Wide Web home page' with URL: `http://www.ens-lyon.fr/~krisrose/Xy-pic.html`.

Preface

In this report we present the TEX and METAFONT sources of the XY-pic package with TEXnical commentary. It assumes that you are familiar with the use of XY-pic as described in the User's Guide [14] as well as with the fundamentals of TEX [6] and METAFONT [7].

Overview. Chapter 1 explains the TEX sources of the XY-picture *kernel* drawing language. Chapter 2 explains the sources of the standard *extensions* providing extended graphic capabilities, and chapter 3 the sources of standard *features* providing specialised notation for particular diagram types. Chapter 5 exposes the METAFONT sources of the standard fonts provided in the distribution.

The appendices contain additional information: Appendix A contains answers to all the exercises of the main text. Appendix B summarises the backwards compatibility with XY-pic version 2. Appendix C contains the GNU General Public License detailing the conditions of use of XY-pic, and appendix D various support files contained in the distribution. Finally a bibliography and the index.

The report includes most of the information in the reference manual [**?**].

License. Most of the XY-pic package is copyright by the individual authors. We have chosen to use the "GNU Copyleft" to make the package available without restriction to as many users as possible and ensure that it is attributed to its authors without risk of persecution. The disclaimer can be found on the inside cover page; the full text of the license is reproduced as appendix C.

Contents

1 Kernel: xy.doc **1**
1.1 The XY-pic implementation . . . 1
1.1.1 Loading XY-pic . . . 1
1.1.2 Logo, version, and messages . . . 13
1.1.3 Fonts . . . 14
1.1.4 Allocations . . . 15
1.1.5 Utility macros . . . 17
1.2 Picture basics . . . 19
1.2.1 Positions . . . 20
1.2.2 Objects . . . 20
1.2.3 Connections . . . 20
1.2.4 Decorations . . . 21
1.2.5 The XY-pic state . . . 21
1.3 Positions . . . 22
1.4 Objects . . . 45
1.5 Decorations . . . 64
1.6 Kernel object library . . . 70
1.6.1 Directionals . . . 70
1.6.2 Circle segments . . . 85
1.6.3 Text . . . 89
1.7 XY-pic options . . . 90
1.7.1 Loading . . . 90
1.7.2 Option file format . . . 91
1.7.3 Driver options . . . 92
1.8 Algorithms . . . 97
1.8.1 Directions . . . 97
1.8.2 Edges . . . 103
1.8.3 Connections . . . 111

2 Extensions **119**
2.1 Curve and Spline extension . . . 119
2.1.1 Curved connections . . . 120
2.1.2 Circles and Ellipses . . . 188
2.1.3 Quadratic Splines . . . 193
2.2 Frame and Bracket extension . . . 196
2.2.1 Frames . . . 198
2.2.2 Brackets . . . 206
2.2.3 Filled regions . . . 209
2.2.4 Framing as object modifier . . . 211

2.2.5 Using curves for frames 212
2.3 More Tips extension 214
2.3.1 End & log 217
2.4 Line styles extension 217
2.5 Rotate and Scale extension 227
2.6 Colour extension 234
2.7 Pattern and Tile extension 240
2.8 Import graphics extension 247
2.9 Movie Storyboard extension 250
2.10 PostScript backend 253
2.10.1 Choosing the DVI-driver 254
2.10.2 Why use POSTSCRIPT[1] 258
2.10.3 Hooking into XY-pic 261
2.10.4 Kernel improvements 262
2.11 TPIC backend 268
2.12 em-TeX backend 275
2.13 Necula's extensions 279
2.13.1 Expansion 280
2.13.2 Polygon shapes 280

3 Features **291**
3.1 All features 291
3.2 Dummy option 292
3.3 Arrow and Path feature 294
3.3.1 Paths 295
3.3.2 Arrows 307
3.4 Two-cell feature 316
3.4.1 Typesetting 2-cells in Diagrams 317
3.4.2 Standard Options 323
3.4.3 Nudging 325
3.4.4 Extra Options 326
3.4.5 After Parsing... 330
3.4.6 2-cells in general XY-pictures 331
3.5 Matrix feature 341
3.5.1 XY-matrices 342
3.5.2 New coordinate formats 352
3.5.3 Spacing and rotation 354
3.5.4 Entries 357
3.6 Graph feature 360
3.7 Polygon feature 369
3.8 Lattice and web feature 389
3.9 Circle, Ellipse, Arc feature 394
3.9.1 Full Circles 396
3.9.2 Ellipses 399
3.9.3 Drawing arcs 400
3.9.4 Circular and Elliptical Arcs 404
3.10 Knots and Links feature 418
3.11 Smart Path option 445

[1] POSTSCRIPT is a registered Trademark of Adobe, Inc. [1].

4 Drivers **461**
4.1 Support for Specific Drivers . . . 461
4.2 dvidrv driver . . . 461
4.3 DVIPS driver . . . 462
4.4 DVITOPS driver . . . 467
4.5 OzTeX driver . . . 472
4.6 OzTeX v1.7 driver . . . 477
4.7 Textures driver . . . 481
4.8 Textures v1.6 driver . . . 485
4.9 XDVI driver . . . 489
4.10 CMacTeX driver . . . 494
4.11 Extra features with POSTSCRIPT support . . . 495
4.11.1 xyps-ps.doc . . . 495
4.11.2 Installation . . . 497
4.11.3 Extensions . . . 504
4.11.4 xyps-c.doc . . . 506
4.11.5 Colour . . . 507
4.11.6 xyps-f.doc . . . 509
4.11.7 Frames . . . 509
4.11.8 xyps-l.doc . . . 513
4.11.9 Line-styles . . . 513
4.11.10 xyps-r.doc . . . 515
4.11.11 Rotations and scaling . . . 516
4.11.12 xyps-t.doc . . . 518
4.11.13 Patterns and tiles . . . 519

5 Standard Fonts **523**
5.1 Semidirectional font . . . 523
5.1.1 xyd2.mf . . . 523
5.1.2 xydash10 . . . 524
5.2 Directional font . . . 527
5.2.1 xyd.mf . . . 527
5.2.2 xyatip10 . . . 528
5.2.3 xybtip10 . . . 530
5.2.4 xybsql10 . . . 531
5.3 Special fonts . . . 533
5.3.1 xycirc10 . . . 533
5.4 Optional fonts . . . 537
5.4.1 xycmat10 . . . 537
5.4.2 xycmbt10 . . . 539
5.4.3 xyeuat10 . . . 541
5.4.4 xyeubt10 . . . 542

A Answers to all exercises **545**

B Backwards Compatibility **553**
B.1 Version 2 Compatibility . . . 553
B.1.1 Unsupported incompatibilities . . . 554
B.1.2 Obsolete kernel features . . . 555
B.1.3 Obsolete extensions & features . . . 560

B.1.4 Obsolete loading . . . 563
B.1.5 Compiling v2-diagrams . . . 565
B.2 Obsolete fonts . . . 567
B.2.1 xyline10 . . . 567
B.2.2 xyqc10 . . . 570
B.2.3 xymisc10 . . . 572

C GNU General Public License **577**

D Distribution support files **583**
D.1 Hype . . . 583
D.1.1 CATALOG . . . 583
D.1.2 README . . . 583
D.2 Installation instructions . . . 585
D.2.1 TRAILER . . . 585
D.2.2 INSTALL . . . 587
D.3 Generation . . . 594
D.3.1 MAKE . . . 594
D.3.2 Makefile . . . 598
Bibliography . . . 628

Index **631**

Chapter 1

Kernel: xy.doc

After giving an overview of the XY-pic environment in §1.1, this chapter document the basic concepts of XY-picture construction in §1.2, including the maintained 'graphic state'. The following sections give the precise syntax rules of the main XY-pic constructions: the position language in §1.3, the object constructions in §1.4, and the picture 'decorations' in §1.5. §1.6 presents the kernel repertoire of objects for use in pictures; §1.7 documents the interface to XY-pic options like the standard 'feature' and 'extension' options.

Section §1.8 documents the more complicated algorithms used to compute directions, edges, and connections.

1.1 The XY-pic implementation

This section briefly discusses the various aspects of the present XY-pic kernel implementation of which the user should be aware.

1.1.1 Loading XY-pic

XY-pic is careful to set up its own environment in order to function with a large variety of formats. For most formats a single line with the command

```
\input xy
```

in the preamble of a document file should load the kernel (see 'integration with standard formats' below for variations possible with certain formats, in particular LaTeX [10]).

The rest of this section describes things you must consider if you need to use XY-pic together with other macro packages, style options, or formats. The less your environment deviates from plain TeX the easier it should be.

File header: Here is what actually happens in the header of `xy.doc`. It contains the copyright message, protection against loading the file more than once, and then bootstrap code to handle category codes and the `DOCMODE` format—we explain each separately below:

```
%% $Id: xy.doc,v 3.6 1998/03/06 01:28:05 krisrose Exp krisrose $
%%
%% Basic Xy-pictures: Xy-pic bootstrap and kernel macros.
%% Copyright (c) 1991-1998  Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
```

```
%% Copyright (c) 1991-1998  Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined\else\message{not reloaded}\endinput\fi
\let\xyloaded=\relax

% NOTE: Apart from the actual macros (as also found in xy.tex), this file
% contains both the Xy-pic kernel reference manual and TeXnical documentation.
% See xyrefer.man and xysource.man for how to typeset this information.

\message{Bootstrap\string'ing\string:}

{\catcode96 12\catcode`\#6\catcode`\.12\catcode`\:12\catcode`\'12\catcode`\@11
\ifx\xywarnifdefined\undefined\else \immediate\write16{}%
 \immediate\write16{Xy-pic Warning: \string\xywarnifdefined\space redefined.}%
 \immediate\write16{}\fi
\gdef\xywarnifdefined#1{\ifx#1\undefined\else \immediate\write16{}%
 \immediate\write16{Xy-pic Warning: `\string#1' redefined.}%
 \immediate\write16{}\fi}
\xywarnifdefined\xydef@ \gdef\xydef@#1{\xywarnifdefined#1\gdef#1}
\xywarnifdefined\xylet@ \gdef\xylet@#1{\xywarnifdefined#1\global\let#1}
\xywarnifdefined\xynew@
 \gdef\xynew@#1#2{\xywarnifdefined#2\csname new#1\endcsname#2}}

\message{catcodes\string,}

\xywarnifdefined\xyuncatcodes
\xywarnifdefined\xyreuncatcodes \def\xyreuncatcodes{\edef\xyuncatcodes{%
 \catcode92 0 \catcode123 1 \catcode125 2 \catcode37 14
 \catcode 9 \the\catcode 9 \catcode10 \the\catcode10 \catcode12 \the\catcode12
 \catcode35 \the\catcode35 \catcode36 \the\catcode36 \catcode38 \the\catcode38
 \catcode43 \the\catcode43 \catcode45 \the\catcode45 \catcode46 \the\catcode46
 \catcode47 \the\catcode47
 \catcode60 \the\catcode60 \catcode61 \the\catcode61 \catcode62 \the\catcode62
 \catcode64 \the\catcode64 \catcode96 \the\catcode96
 \newlinechar \the\newlinechar \endlinechar \the\endlinechar }}
\xyreuncatcodes

\xywarnifdefined\xycatcodes \def\xycatcodes{%
 \catcode 9 10
```

```
 \catcode 35  6 \catcode 36 3 \catcode 38 4
 \catcode 43 12 \catcode 45 12 \catcode 46 12 \catcode 47 12
 \catcode 60 12 \catcode 61 12 \catcode 62 12
 \catcode 64 11 \catcode 96 12 }

\xycatcodes

\message{docmode,}

{\catcode`\|0 \xywarnifdefined|DOCMODE
\gdef|DOCMODE#1{\ifx(#1\relax \xycatcodes \expandafter\ignorespaces
 \else \skipspecials@ \expandafter\docm@\fi}%

\xywarnifdefined\skipspecials@
\gdef\skipspecials@{%
 \catcode`\\12 \catcode`\{12 \catcode`\}12 \catcode`\#12 \catcode`\%12
 \catcode`\^^L12 \endlinechar`\^^J }%

\catcode`\/=12 \lccode`\/`\\%
\lccode`\D`\D \lccode`\O`\O \lccode`\C`\C \lccode`\M`\M \lccode`\E`\E
\lowercase{%
\xywarnifdefined\docm@ \gdef\docm@{\docm@i}%
\xywarnifdefined\docm@i \gdef\docm@i#1^^J{\docm@ii#1/DOCMODE\docm@iii}%
\xywarnifdefined\docm@ii
 \gdef\docm@ii#1/DOCMODE{\def\next@{#1}\futurelet\next\docm@iii}%
\xywarnifdefined\docm@iii \gdef\docm@iii#1\docm@iii{%
 \ifx\next\docm@iii \let\next\next@ \docecho@ \let\next@\docm@
 \else\ifx\next@\empty \let\next@\docfinish@
 \else \edef\next@{\noexpand\docm@iv\next@/DOCMODE#1\noexpand\docm@iv}%
 \fi\fi \next@}%
\xywarnifdefined\docm@iv
 \gdef\docm@iv#1/DOCMODE\docm@iv{\def\next{#1}\docecho@ \docm@}}%

\xywarnifdefined\docecho@ \global\let\docecho@\relax
\xywarnifdefined\docfinish@ \gdef\docfinish@{\xyuncatcodes|DOCMODE\next}}
```

Privacy: XY-pic will warn about control sequences it redefines—thus you can be sure that there are no conflicts between XY-pic-defined control sequences, those of your format, and other macros, provided you load XY-pic last and get no warning messages like

`Xy-pic Warning: `...' redefined.`

In general the XY-pic kernel will check all control sequences it redefines *except* that (1) generic temporaries like `\next` are not checked, (2) predefined font identifiers (see §1.1.3) are assumed intentionally preloaded, and (3) some of the more exotic control sequence names used internally (like `@{-}`) are only checked to be different from `\relax`.

This is handled by `\xywarnifdefined`—after we have ensured that it is unique itself[1]. `\xydef @`, `\xylet @`, and `\xynew@{`⟨type⟩`}` are abbreviations used to this end throughout XY-pic instead of `\let`, `\def`, and the `\new`⟨type⟩ commands.

[1]This may seem paranoid but in fact many inconvieniences in the TEX world stem from the fact that somebody copied somebody elses definition of, say, `\xywarnifdefined`, modified it, and then used it in something that somehow got distributed! The 'flat name space' problem remains TEX largest problem as a programming language in this TEX hackers opinion.

Next some auxilliaries: `\xydefcsname@` is similar to `\xydef@` except that it builds the control sequence with `\csname` … `\endcsname` which means that it is `\relax` when undefined—there is thus no way to prevent redefinition of control sequences bound to `\relax` ☹.

```
\xydef@\xydefcsname@#1{\DN@{#1}\DNii@##1{%
  \ifx ##1\relax\else \xywarning@{\string`\string##1\string' redefined}\fi
  \def##1}%
 \expandafter\nextii@\csname\codeof\next@\endcsname}
```

`\xyletcsnamecsname@` is to `\let` one weird control sequence be the same as another using several `\expandafter`s:

```
\xydef@\xyletcsnamecsname@#1#2{\def\1{#1}\def\2{#2}\DN@##1##2{%
  \ifx ##1\relax\else \xywarning@{`\string##1\string' redefined}\fi
  \let##1=##2}%
 \expandafter\expandafter\expandafter\next@
  \expandafter\csname\expandafter\codeof\expandafter\1\expandafter\endcsname
   \csname\codeof\2\endcsname}
```

Finally `\codeof`: a useful hack used to allow any characters in control sequences: `\codeof`⟨cs⟩ expands to the characters of the control sequence ⟨cs⟩ as a string of 'other' characters, *i.e.*, all of category 12 and with a ␣$_{12}$ after every control sequence. The ⟨cs⟩ must be a macro or it blows up.

```
\xywarnifdefined\codeof
\xywarnifdefined\codeof@
{\catcode`\:=12 % to ensure that all of :-> and <-: are other :-)
 \gdef\codeof#1{\expandafter\codeof@\meaning#1<-:}
 \gdef\codeof@#1:->#2<-:{#2}}
```

Category codes: The situation is complicated by the flexibility of TeX's input format. The culprit is the 'category code' concept of TeX (*cf.* [6, p.37]): when loaded Xy-pic requires the characters `␣\{}%` (the first is a space) to have their standard meaning and all other printable characters to have the *same category as when Xy-pic will be used*—in particular this means that (1) you should surround the loading of Xy-pic with `\makeatother` … `\makeatletter` when loading it from within a LaTeX package, and that (2) Xy-pic should be loaded after files that change category codes like the `german.sty` that makes `"` active. Some styles require that you reset the catcodes for every diagram, *e.g.*, with `french.sty` you should use the command `\english` before every `\xymatrix`.

We define `\xyuncatcodes` to restore the current catcodes, and `\xycatcodes` to install our own.

Here is an exact list of the category codes which Xy-pic requires (all standard in plain TeX):

character(s)	`\`	`{`	`}`	CR	TAB SP	`A`–`Z` `a`–`z`	`0`–`9`	`%`
category code	0	1	2	5	10	11	12	14

Furthermore none of the remaining printable ASCII characters

`!"#$&'()*,/:;?@[]^_`|~`

may be of category 0, 1, 2, 9, 14, or 15, because all should be tokens allowed in the replacement text of a `\def`—this also means that they may not be active characters defined to be "`\outer`"!

All other catcodes needed are established using `\xycatcodes` defined above—this is the reason the macros must be loaded at a time where the category codes are stable (otherwise it will make them stable!).

Internally XY-pic enforces the following category codes:

character	#	$	&	'	+	-	.	<	=	>	@	`
ASCII code	35	36	38	39	43	45	46	60	61	62	64	96
category code	6	3	4	12	12	12	12	12	12	12	11	12
abbreviation	HASH	DOLL	AND	RQ	PLUS	DASH	DOT	LT	EQ	GT	AT	LQ

with special control sequences named `\add`⟨abbreviation⟩`@` that take an argument and expand to it followed by the original character token, *i.e.*, many tests throughout the program look like `\addDOT@ \ifx \next` ...

```
\xywarnifdefined\addAT@
\xywarnifdefined\addHASH@
\xywarnifdefined\addDOLL@
\xywarnifdefined\addAND@
\xywarnifdefined\addRQ@
\xywarnifdefined\addPLUS@
\xywarnifdefined\addDASH@
\xywarnifdefined\addDOT@
\xywarnifdefined\addLT@
\xywarnifdefined\addEQ@
\xywarnifdefined\addGT@
\xywarnifdefined\addLQ@

\xydef@\xymakeADD@#1#2 #3 {\ifnum\catcode#3=6 \def#1##1{##1#2#2}%
 \else \def#1##1{##1#2}\fi}

\xydef@\xyrecat@{\xymakeADD@\addAT@}
\xydef@\xyrecat{\xyrecat@}
\xydef@\xyresetcatcodes{\def\xyrecat{\xyrecat@}\xyreuncatcodes
 \input xyrecat \relax \xyuncatcodes}

\xyuncatcodes \xyresetcatcodes \xycatcodes

\xydef@\testAND@{{\aftergroup\ifx
  \addAND@\ifx\next\addFalse@\else\addTrue@\fi}}
\xydef@\addFalse@{\aftergroup\iffalse}
\xydef@\addTrue@{\aftergroup\iftrue}
```

However, it is possible to 'repair' the problem in case any of the characters ``#$&'+-.<=>` `` change category code:

`\xyresetcatcodes`

will load the file `xyrecat.tex` (version 3.3) to do it.

```
%% $Id: xyrecat.doc,v 3.3 1996/12/19 03:31:56 krisrose Exp $
%%
%% Xy-pic ``Reset Category Codes'' utility.
%% Copyright (c) 1994-1996      Kristoffer H. Rose      <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996      Kristoffer H. Rose      <krisrose@brics.dk>
```

```
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
```

Header: The command is defined in `xy.doc`: the purpose of this file is to be reloadable such that the characters can be reread by TEX with fresh catcodes.

The code redefines the special `\add...@` control sequences used for parsing of just those characters, assuming `\next` is defined correctly before loading it:

```
%\xyreuncatcodes
\xyuncatcodes
```

```
\xyrecat @ 64 \catcode 64 11
```

```
\xymakeADD@\addHASH@    # 35
\xymakeADD@\addDOLL@    $ 36
\xymakeADD@\addAND@     & 38
\xymakeADD@\addRQ@      ' 39
\xymakeADD@\addPLUS@    + 43
\xymakeADD@\addDASH@    - 45
\xymakeADD@\addDOT@     . 46
\xymakeADD@\addLT@      < 60
\xymakeADD@\addEQ@      = 61
\xymakeADD@\addGT@      > 62
\xymakeADD@\addLQ@      ` 96
```

```
\endinput
```

```
% $Log: xyrecat.doc,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.1  1995/09/05 20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
```

```
% NEW file to go in version 3!
```

The last block of the XY-pic header bootstraps the "`DOCMODE` format" used in `.doc` variants of XY-pic macro files in order to keep documentation and macros together in a literal programming style (this is redundant in the `xy.tex` macro file where all instances of `DOCMODE` have been eliminated (see chapter §D.3.2 for how this is accomplished) but it is included anyway since users may load options still in `DOCMODE` format). The details of `DOCMODE` are described in `xydoc.sty`, a special LaTeX package used to typeset XY-pic documentation; please read it if you intend to write XY-pic options.

Integration with standard formats This is handled by the `xyidioms.tex` file and the integration as a LaTeX [10] package by `xy.sty`.

We input `xyidioms.tex` from the kernel:

```
\input xyidioms
```

xyidioms.doc: This included file provides some common idioms whose definition depends on the used format such that XY-pic can use predefined dimension registers etc. and yet still be independent of the format under which it is used. The current version (3.4) handles plain TeX (version 2 and 3 [6]), $\mathcal{A}\mathcal{M}\mathcal{S}$-TeX (version 2.0 and 2.1 [17]), LaTeX (version 2.09 [9] and 2ε [10]), $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX (version 1.0, 1.1 [2], and 1.2), and eplain (version 2.6 [3])[2].

```
%% $Id: xyidioms.doc,v 3.4 1997/05/18 01:14:25 krisrose Exp $
%%
%% Format-dependent idioms for Xy-pic.
%% Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyidiomsloaded\empty \message{not reloaded}\endinput \fi
\let\xyidiomsloaded=\empty
```

Header: This is a separate file so that we can abort the definitions easily using `\endinput`.

[2] The 'v2' feature introduces some name conflicts, in order to maintain compatibility with earlier versions of XY-pic.

Scratch registers: All the formats currently treated allow the use of the following plain TEX scratch register conventions (*cf.* [6, p.346])

- `\count@` is available as a local scratch counter register.
- `\dimen@`, `\dimen@i`, `\dimen@ii`, `\dimen3` to `\dimen9`, and `\dimen255` are available as scratch dimension registers.
- `\skip@`, `\skip1` to `\skip9`, and `\skip255` are available as scratch skip registers.
- `\toks@` and `\toks1` to `\toks9` are available as scratch token lists.
- `\boxz@` and `\box1` to `\box9` are available as scratch box registers.

with the constraint that `\global` should never (always) be used when assigning to number 0, 2, 4, 6, 8, and 255 (1, 3, 5, 7, and 9).

XY-specific scratch registers: We first give new names to dimension registers 4, 6, and 8, since none of the formats do that: `\A@`, `\B@`, and `\R@` are specific to XY-pic and will be referred to as A, B, and R in comments.

```
\xywarnifdefined\A@ \dimendef\A@=4
\xywarnifdefined\B@ \dimendef\B@=6
\xywarnifdefined\R@ \dimendef\R@=8
```

Some shared scratch registers are defined last.

Idioms: First check that `\undefined` is indeed undefined. . . or rather: the same as an extremely unlikely control sequence that we are making up:

```
\ifx\undefined\AveryUNLIKELYc@ntr@lSEQUENCE@@\else
\errmessage{Xy-pic Error: \string\undefined\space defined.}\fi
```

These idioms are so common they are just conditionally defined:

```
\ifx\undefined\literal@ \def\literal@#1{#1}\fi
\ifx\undefined\eat@     \def\eat@#1{}\fi

\xydef@\xyFN@{\futurelet\next} % Now private because of AMS-LaTeX change.
\ifx\undefined\DN@      \def\DN@{\def\next@}\fi
\ifx\undefined\DNii@    \def\DNii@{\def\nextii@}\fi

\ifx\undefined\setboxz@h\def\setboxz@h{\setbox\z@\hbox}\fi
\ifx\undefined\wdz@     \def\wdz@{\wd\z@}\fi
\ifx\undefined\boxz@    \def\boxz@{\box\z@}\fi

\ifx\undefined\W@       \def\W@{\immediate\write16 }\fi

\ifx\undefined\space@   \def\space@.{\futurelet\space@\relax}\space@. \fi

\ifx\undefined\notempty \def\notempty#1{T\if @#1@F\else T\fi}\fi
```

Delayed setup: This is used when it is advantageous to delay loading of something until after the preamble, *i.e.*, after all options, *etc.*, have been loaded. This is particularly true for nested `\xywithoption` uses that will otherwise be executed repeatedly. Currently properly supported with LATEX 2$_\varepsilon$ and the `amsppt` style of $\mathcal{A}_\mathcal{M}\mathcal{S}$-TEX.

```
\xydef@\xysetup@dummy#1{\xyuncatcodes#1}
```

```
\xywarnifdefined\xysetup@@
```

```
\ifx\AtEndDocument\undefined
 \expandafter\ifx\csname amsppt.sty\endcsname\relax
  \let\xysetup@@=\xysetup@dummy
 \else
  \def\xysetup@@#1{%
   \expandafter\def\expandafter\topmatter\expandafter{\topmatter
    #1\xyuncatcodes}}\fi
\else
 \def\xysetup@@#1{\AtBeginDocument{#1\xyuncatcodes}}
\fi
```

```
\ifx\xysetup@@\xysetup@dummy\else
 \xysetup@@{\let\xysetup@@=\xysetup@dummy \xyuncatcodes}\fi
```

Similarly, the following provides a place to insert ‘trailing messages’ if the format supports it; otherwise it just throws away the argument (useful for repeating the most important warnings).

```
\xywarnifdefined\xyclosedown@@
```

```
\ifx\AtEndDocument\undefined \let\xyclosedown@@=\eat@
\else \def\xyclosedown@@#1{\AtEndDocument{#1}}\fi
```

This may be useful to hook into specialised mechanisms for including `\special` material on the page.

```
\xylet@\xy@begindvi=\literal@
```

For example, LaTeX 2_ε (post Dec ’94) defines an `\AtBeginDvi` macro to allow zero-sized material to be placed at the beginning of each shipped-out page. Among other uses, this allows such material not to be discarded, when on a page containing only floats. This is used by XY-pic in connection with the PostScript header dictionary `xyps-ps.doc`.

```
\ifx\undefined\AtBeginDvi\else
 \AtBeginDocument{\let\xy@begindvi=\AtBeginDvi}
 \AtBeginDvi{\global\let\xy@begindvi=\literal@}\fi
```

Shared scratch registers: Everything else in this file is also done by AMS-TeX so we exit here when using that format after renaming their `\toks@@` to `\toks@ii`...

```
\ifx\amstexloaded@\relax
\xylet@\toks@ii=\toks@@ \endinput \fi
```

Thus the job of allocating scratch registers is simple except for counters where we only have one so we use LaTeX scratch counters when available and otherwise allocate two new ones:

```
\ifx\@tempcnta\undefined
\xynew@{count}\count@@
\xynew@{count}\count@@@
\else
\xylet@\count@@=\@tempcnta
\xylet@\count@@@=\@tempcntb
\fi
```

Finally the second scratch token register not available with LaTeX.

```
\ifx\undefined\toks@ii  \toksdef\toks@ii=2 \fi
```

End & log: That's all.

```
\endinput

% $Log: xyidioms.doc,v $
% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.11  1994/07/05  10:37:32  kris
% Third 3beta release [bug fixes].
% Experimental graph feature included (for ECCT-94 presentation).
%
% Revision 2.9  1994/06/09  15:02:49  kris
% Release 3beta.
%
% Revision 2.8  1994/04/08  04:30:00  kris
% Second (bug fix) 3alpha release.
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% Now contains scratch register allocations.
% Cleaned of AMS-TeX special macros.
% Renamed from xyamstex.doc to xyidioms.doc for version 2.7.
%
% Revision 2.4  1992/01/22  02:15:10  kris
% \\ works with AMS-LaTeX: don't use \Let@ [Werner Struckmann/Darrel Hankerson]
% Made redefinition of \text conditional.
%
% Revision 2.3  1992/01/10  21:43:09  kris
% Fixed AMS-LaTeX name clash problems.
%
% Revision 2.2  1992/01/09  04:05:40  kris
```

```
% Patched to fix problem with \text and support AMS-LaTeX.
%
% Revision 2.1  1992/01/02  14:55:27  kris
% Release version.
%
% Revision 1.3  1991/12/17  04:51:16  kris
% Version distributed with ‘final draft’ on Usenet.
```

xy.sty: If you use LaTeX then this file makes it possible to load XY-pic as a ‘package’ using the LaTeX 2_ε [10] `\usepackage` command:

`\usepackage [`⟨option⟩`,...] {xy}`

---

where the ⟨option⟩s will be interpreted as if passed to `\xyoption` (*cf.* §1.7).

The only exceptions to this are the options having the same names as those driver package options of chapter **??**, which appear in *cf.* [4, table 11.2, p.317] or the LaTeX 2_ε `graphics` bundle. These will automatically invoke any backend extension required to best emulate the LaTeX 2_ε behaviour. (This means that, *e.g.*, `[dvips]` and `[textures]` can be used as options to the `\documentclass` command, with the normal effect.)

The file also works as a LaTeX 2.09 [9] ‘style option’ although you will then have to load options with the `\xyoption` mechanism described in §1.7.

Here is the raw source of `xy.sty`.

```
%% $Id: xy.sty,v 3.3 1996/12/19 03:31:56 krisrose Exp $
%%
%% ‘‘Xy-pic as LaTeX 2.09 style option and LaTeX 2e package’’.
%% Copyright (c) 1993-1996      Kristoffer H. Rose      <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996      Kristoffer H. Rose      <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.

% Load Xy-pic with catcode of @ set as it will be in user text:
\count255=\the\catcode‘\@ {\catcode‘\@=11 \xdef\xystycatcode{\the\count255}}
\makeatletter

% REPAIR bug of the 2e version of AMS-LaTeX where \makeatother is broken...
```

```
\ifx\@ifpackageloaded\undefined \makeatother
\else \@ifpackageloaded{amstex}{\catcode`\@=\active}{\makeatother}\fi

\input xy

% That is all unless this is called as a LaTeX2e `native mode' package.  Is
% there an official way to check this?  Anyway, this seems to work:
\makeatletter
\ifx\if@compatibility\undefined \catcode`\@=\xystycatcode \endinput \fi

% Ensure that we are *really* running LaTeX2e...!
\NeedsTeXFormat{LaTeX2e}

% This file provides the current version of Xy-pic!
\edef\next{%
 \noexpand\ProvidesPackage{xy}[\xydate\space Xy-pic version \xyversion]}\next

% Recognise some standard LaTeX2e class options (cf. `LaTeX Companion' p.317):
\DeclareOption{cmactex}{\xyoption{cmactex}}
\DeclareOption{dvips}{\xyoption{dvips}\xyoption{ps}}
\DeclareOption{dvitops}{\xyoption{dvitops}\xyoption{ps}}
\DeclareOption{emtex}{\xyoption{emtex}}
\DeclareOption{ln}{\xywarning@{`ln' DVI driver not yet fully supported}}
\DeclareOption{oztex}{\xyoption{oztex}}
\DeclareOption{textures}{\xyoption{textures}}
\DeclareOption{xdvi}{\xyoption{xdvi}}

% This is special:
\DeclareOption{colour}{\xyoption{color}} % :-)

% This (obsolete) cmtip style option should be activated after loading:
\DeclareOption{cmtip}{\xyoption{cmtip}\UseComputerModernTips}

% Pass the desired tip size if defined
\DeclareOption{10pt}{\xywithoption{tips}{\def\tipsize@@{10}}}
\DeclareOption{11pt}{\xywithoption{tips}{\def\tipsize@@{11}}}
\DeclareOption{12pt}{\xywithoption{tips}{\def\tipsize@@{12}}}

% Other options are just loaded.
\DeclareOption*{\edef\next{\noexpand\xyoption{\CurrentOption}}\next}

% Reinstall catcode of @...
\catcode`\@=\xystycatcode

% ... process the options, and that is all.
\ProcessOptions

\endinput

% $Log: xy.sty,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
```

```
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.9  1994/06/09  14:59:19  kris
% Release 3beta.
%
% Revision 2.8  1994/04/08  04:30:00  kris
% Second (bug fix) 3alpha release.
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% New for version 2.7.
```

1.1.2 Logo, version, and messages

Loading XY-pic prints a banner containing the version and author of the kernel; small progress messages are printed when each major division of the kernel has been loaded. Any options loaded will announce themself in a similar fashion.

Of these, `\stripRCS` is a very useful hack for extracting the first component of an RCS `$...$` keyword value.

```
\xydef@\stripRCS$#1${\stripRCS@#1: @@ @@@}
\xydef@\stripRCS@#1: #2@ #3@@@{%
 \ifx @#2\string?\else\ifx :#2\else\stripRCS@@#2\fi\fi}
\xydef@\stripRCS@@#1 #2: @{#1}
```

```
\edef\next{\stripRCS$Revision: 3.6 $}
\xylet@\xyversion=\next
```

```
\edef\next{\stripRCS$Date: 1998/03/06 01:28:05 $}
\xylet@\xydate=\next
```

```
\xydef@\Xygreet@{%
 \W@{}%
 \W@{ Xy-pic version \xyversion\space<\xydate>}%
 \W@{ Copyright (c) 1991-1998 by Kristoffer H. Rose <krisrose@ens-lyon.fr>}%
 \W@{ Xy-pic is free software: see the User\string's Guide for details.}%
 \W@{}}
\Xygreet@
\expandafter\everyjob\expandafter{\the\everyjob\Xygreet@}
```

```
\message{Loading kernel:}
```

If you refer to XY-pic in your written text (please do ☺) then you can use the command `\Xy-pic` to typeset the "XY-pic" logo. The version of the kernel is typeset by `\xyversion` and the release date by `\xydate` (as found in the banner). By the way, the XY-pic *name*[3] originates from the fact that the first

[3] No description of a TEX program is complete without an explanation of its name.

version was little more than support for (x, y) coordinates in a configurable coordinate system where the main idea was that *all* operations could be specified in a manner independent of the orientation of the coordinates. This property has been maintained except that now the package allows explicit absolute orientation as well.

```
\xydef@\Xy{\leavevmode
 \hbox{\kern-.1em X\kern-.3em\lower.4ex\hbox{Y\kern-.15em}}}
```

Messages that start with "`Xy-pic Warning`" are indications that something needs your attention; an "`Xy-pic Error`" will stop TEX because Xy-pic does not know how to proceed.

We use the input line number if available and rudimentary help in the form of a reference to the manual if no specific help string is given. `\newlinechar` is set locally to `^^J` while writing such that messages of several lines can be written.

```
\message{messages;}
```

```
\xywarnifdefined\thelineno@
\ifx\inputlineno\undefined \edef\thelineno@{\string?}
\else \def\thelineno@{\the\inputlineno}\fi
\xydef@\xytracelineno@{ \string[\jobname:\thelineno@\string]}
```

```
\xydef@\xywarning@#1{{\newlinechar=10 %
  \W@{}\W@{Xy-pic Warning: #1\xytracelineno@.}\W@{}}}
```

```
\xydef@\xyerror@#1#2{\if\inxy@\xy@{ERROR #1}{}\fi
 {\def\2{#2}\newlinechar=10 %
  \ifx\2\empty \errhelp{See the Xy-pic manual for further information.}%
  \else \errhelp{#2}\fi
  \errmessage{Xy-pic error: #1}}}
```

Finally one that I hope will never get expanded ☹

```
\xydef@\xybug@#1{{\newlinechar=10 %
 \errhelp{This is a bug in Xy-pic and should not happen!^^J%
If it did then please send a bug report with the offending Xy-pic code^^J%
to the author of Xy-pic, krisrose@ens-lyon.fr.}%
 \errmessage{Xy-pic BUG: #1 -- notify krisrose@ens-lyon.fr.}}}
```

1.1.3 Fonts

The Xy-pic kernel implementation makes its drawings using five specially designed fonts:

Font	Characters	Default
`\xydashfont`	dashes	`xydash10`
`\xyatipfont`	arrow tips, upper half	`xyatip10`
`\xybtipfont`	arrow tips, lower half	`xybtip10`
`\xybsqlfont`	quarter circles for hooks and squiggles	`xybsql10`
`\xycircfont`	1/8 circle segments	`xycirc10`

The first four contain variations of characters in a large number of directions, the last contains 1/8 circle segments.

Note: The default fonts are not part of the Xy-pic kernel *specification*: they just set a standard for what drawing capabilities should at least be required by an Xy-pic implementation. Implementations exploiting capabilitites of particular output devices are in use. Hence the fonts are only loaded by XY-pic if the control sequence names are undefined—this is used to preload them at different sizes or prevent them from being loaded at all.

To be more precise, XY-pic requires `\xydashfont` to be a *semidirectional* font as METAFONT will generate with the driver file `xyd2.mf`—this is very important because the *italic corrections* of the characters in this particular font are used to approximate trigonometric computations, so if you replace `\xydashfont` be sure to replace it with another semidirectional font! Similarly, the three fonts `\xyatipfont`, `\xybtipfont`, and `\xybsqlfont` should be *directional* as METAFONT will generate with the driver file `xyd.mf`.

Finally, `\xycircfont` should contain 1/8 circle segments of various radii as described in `xycirc10.mf`.

The following code loads the fonts *unless it was already loaded* and defines some associated dimensions for `\xydashfont` and `\xybsqlfont`: for each of these f we define f_ℓ as the length of a unit in the current direction (used when juxtaposing for connections), f_h as the height of the unit (used for several parallel connections), and f_w as the 'line width' of the unit (to know how to interface to rules).

```
\message{fonts;}

\xydef@\xyfont@#1{\ifx#1\undefined \DN@{\global\font#1}\expandafter\next@
 \else \xywarning@{Using previously loaded \string#1\space font}\fi}

\xyfont@\xydashfont=xydash10
\xydef@\xydashl@{\fontdimen6\xydashfont}
\xydef@\xydashh@{\fontdimen5\xydashfont}
\xydef@\xydashw@{\fontdimen8\xydashfont}

\xyfont@\xyatipfont=xyatip10
\xyfont@\xybtipfont=xybtip10

\xyfont@\xybsqlfont=xybsql10
\xydef@\xybsqll@{\fontdimen6\xybsqlfont}
\xydef@\xybsqlh@{\fontdimen5\xybsqlfont}
\xydef@\xybsqlw@{\fontdimen8\xybsqlfont}

\xyfont@\xycircfont=xycirc10
```

1.1.4 Allocations

One final thing that you must be aware of is that XY-pic allocates a significant number of dimension registers and some counters, token registers, and box registers, in order to represent the state and do computations. The current kernel allocates 4 counters, 28 dimensions, 2 box registers,4 token registers, 1 read channel, and 1 write channel (when running under LaTeX; some other formats use slightly more because standard generic temporaries are used). Options may allocate further registers (currently loading *everything* loads 6 dimen-, 3 toks-, 1 box-, and 9 count-registers in addition to the kernel ones).

```
\message{allocations:}
```

See §1.1.1 for scratch register allocations.

Picture state: These realise the picture state as described in §1.2.5: c, p, the *base*, and the picture size:

```
\message{state,}

\xynew@{dimen}\X@c
\xynew@{dimen}\Y@c
```

```
\xynew@{dimen}\U@c
\xynew@{dimen}\D@c
\xynew@{dimen}\L@c
\xynew@{dimen}\R@c
\xynew@{toks}\Edge@c
```

```
\xynew@{dimen}\X@p
\xynew@{dimen}\Y@p
\xynew@{dimen}\U@p
\xynew@{dimen}\D@p
\xynew@{dimen}\L@p
\xynew@{dimen}\R@p
\xynew@{toks}\Edge@p
```

```
\xynew@{dimen}\X@origin \X@origin=\z@
\xynew@{dimen}\Y@origin \X@origin=\z@
\xynew@{dimen}\X@xbase  \X@xbase=1mm
\xynew@{dimen}\Y@xbase  \Y@xbase=\z@
\xynew@{dimen}\X@ybase  \X@ybase=\z@
\xynew@{dimen}\Y@ybase  \Y@ybase=1mm
```

```
\xynew@{dimen}\X@min
\xynew@{dimen}\Y@min
\xynew@{dimen}\X@max
\xynew@{dimen}\Y@max
```

Drop and connect: `\lastobjectbox@` stores the most recently dropped object.

```
\xynew@{box}\lastobjectbox@
```

`\zerodotbox@` is of zero size with a 'dot' in the form of a rule the width and height as the line width of the line font; `\zz @` is 'almost-zero-check'.

```
\xynew@{box}\zerodotbox@
\setbox\zerodotbox@=\hbox{\dimen@=.5\xydashw@
 \kern-\dimen@ \vrule width\xydashw@ height\dimen@ depth\dimen@}
\wd\zerodotbox@=\z@ \ht\zerodotbox@=\z@ \dp\zerodotbox@=\z@
```

```
\xynew@{dimen}\almostz@ \almostz@=50sp
\xydef@\zz@#1{\ifdim#1<\z@-\fi#1<\almostz@\relax}
```

```
\xynew@{if}\iftmp@
```

Direction state: The direction state is rather complicated and described in detail in §1.8.1.

```
\message{direction,}
```

```
\xynew@{dimen}\d@X
\xynew@{dimen}\d@Y
\xydef@\sd@X{}
\xydef@\sd@Y{}
```

```
\xynew@{count}\K@ \K@=1024
\xynew@{count}\KK@ \KK@=32
```

```
\xynew@{count}\Direction
```

```
\xynew@{dimen}\K@dXdY
\xynew@{dimen}\K@dYdX

\xydef@\cosDirection{}
\xydef@\sinDirection{}

\xywarnifdefined\DirectionChar
\xywarnifdefined\SemiDirectionChar
```

Miscellaneous: Finally some generic allocations used in the following:

```
\xynew@{read}\xyread@                    % for ‘safe input’
\xynew@{write}\xywrite@                  % for ‘saving’ to .xyc file

\xynew@{count}\csp@                      % for ‘control stack pointer’
\xynew@{dimen}\quotPTK@                  % for ‘fractions’
```

The required temporaries are defined by `xyidioms.tex`.

1.1.5 Utility macros

Finally we define some utility macros.

```
\message{utility macros;}
```

Simple queue: Just appending to the `\toks @` list.

```
\xydef@\addtotoks@#1{\toks@=\expandafter{\the\toks@#1}}
```

Safe input: Check that file is available before input. Tries the alternate extension `.doc` in case the `.tex` file is not there. The second is the lowest level function used when the desired files cannot be preloaded and do not exist in `.doc` form.

```
\xydef@\xyinputorelse@#1#2{%
 \expandafter\let\expandafter\next@\csname#1loaded\endcsname
 \ifx\next@\empty \else
  \DN@{\xyinputorelse@@{#1}{\xyinputorelse@@{#1.doc}{#2}}}%
 \fi \next@}

\xydef@\xyinputorelse@@#1#2{\openin\xyread@=#1 %
 \ifeof\xyread@ \DN@{#2}\else \DN@{\closein\xyread@\input#1 }\fi \next@}
```

Continuation stack: This is used to ‘enter’ a new context and ‘leave’ to the previous context. It works as a stack defining a control sequence for each level, thus using a counter as the stack pointer. Defines the following

`\csp@`	‘Continuation Stack Pointer’
`\enter@{`⟨code⟩`}`	Enter new block with ⟨code⟩ expanded as continuation
`\nter@{`⟨code⟩`}`	Enter new block with⟨code⟩ as continuation
`\dontleave@`	Execute continuation without leaving block
`\unenter@`	Leave block without executing its continuation
`\leave@`	Leave block (execute its continuation)

So `\enter@{}\leave@` is a noop and `\leave@` is the same as `\dontleave@\unenter@`.

```
\global\csp@=\z@
```

```
\xydef@\enter@#1{\global\advance\csp@\@ne
 \expandafter\xdef\csname cs@\number\csp@\endcsname{#1}\ignorespaces}
```

```
\xydef@\nter@#1{\global\advance\csp@\@ne
 \expandafter\gdef\csname cs@\number\csp@\endcsname{#1}\ignorespaces}
```

```
\xydef@\dontleave@{\csname cs@\number\csp@\endcsname}
```

```
\xydef@\unenter@{\global\advance\csp@\m@ne}
```

```
\xydef@\leave@{\expandafter\unenter@\csname cs@\number\csp@\endcsname}
```

Fractions: Below we often use a factor on the form of a quotient A/B. Here is a hack to get it; it is not very precise but suffices for our needs.

`\quotient@` ⟨cs⟩ {A} {B}	Defines ⟨cs⟩ to expand (immediately) to the factor corresponding to A/B; A, B must be dimensions where $\lvert A\rvert <$ `\maxdimen`/`KK` and $\lvert B\rvert >$ `KK`
`\quotient@@` ⟨cs⟩ {A} {B}	Same, but uses 8`KK` for `KK`.

Notes: (1) If `\c` is a count register, then `{1\c}` is a legal dimension. (2) Really computes

$$((A \times |\mathtt{KK}|)/(B/\,|\mathtt{KK}|)) \times (\mathtt{1pt}/\,|\mathtt{K}|)$$

and then defines ⟨cs⟩ to expand to the resulting `pt` value. This means that results are only reasonable for $|A| \ll$ `\maxdimen`/`KK` and $|B| \gg$ `KK`.

```
\quotPTK@=\p@ \divide\quotPTK@\K@
\xylet@\quotsign@@=\empty
\xywarnifdefined\removePT@
{\catcode`p=12 \catcode`t=12 \gdef\removePT@#1pt{#1}}
```

```
\xydef@\quotient@#1#2#3{\A@=#2\relax \B@=#3\relax
 \ifdim\A@<\z@\def\quotsign@@{-}\else\def\quotsign@@{+}\fi
 \ifdim\quotsign@@\A@<15pt \multiply\A@\K@
 \else\ifdim\quotsign@@\A@<511pt \multiply\A@\KK@
   \advance\B@\ifdim\B@<\z@-\fi 16sp \divide\B@\KK@
  \else
   \advance\B@\ifdim\B@<\z@-\fi 512sp \divide\B@\K@
 \fi\fi
 \ifdim\ifdim\B@<\z@-\fi\B@<\quotPTK@ \xywarning@{division overflow}%
 \else \advance\A@.5\B@ \divide\A@\B@ \fi
 \multiply\A@\quotPTK@ \edef#1{\expandafter\removePT@\the\A@}}
```

```
\xydef@\quotient@@#1#2#3{\A@=#2\relax \B@=#3\relax \multiply\A@\KK@
 \advance\B@\ifdim\B@<\z@-\fi 128sp \divide\B@\KK@ \divide\B@ 8 %
 \ifdim\B@=\z@\else \advance\A@.5\B@ \divide\A@\B@ \fi
 \B@=.125\quotPTK@ \multiply\A@\B@ \edef#1{\expandafter\removePT@\the\A@}}
```

Loops: XY-pic uses its own `\loop@` to avoid interference with plain `\loop`.

```
\xydef@\loop@#1\repeat@{\def\body@{#1}\iterate@}\xylet@\repeat@=\fi
\xydef@\iterate@{\body@\expandafter\iterate@\else\fi}
```

Execution: All execution of Xy-commands should be 'indirect', *i.e.*, execute

`\xy@{`⟨source⟩`}{`⟨internal commands⟩`}`

where the ⟨internal commands⟩ directly do the desired operation(s). This is used for tracing and can be used to separate parsing and execution by changing `\xy@`; `\oxy@` is kept stable such that `\let\xy@=\oxy@` will reestablish a sane state; it should only be changed with `\change@oxy@` to make this easy to verify.

```
\xydef@\xyinitial@#1#2{\DN@{#1}%
 \xyerror@{command used out of context: \codeof\next@}{}}
```

```
\xylet@\xy@=\xyinitial@
\xylet@\oxy@=\xy@
```

```
\xydef@\change@oxy@#1{\let\oxy@=#1\relax
% \W@{*OXY@:=\meaning\oxy@}%
}
```

This is also used to check whether an Xy-picture is already active; use as `\if\inxy@`...`\else`...`\fi`:

```
\xydef@\inxy@{T\ifx\xy@\xyinitial@ F\else T\fi}
```

The final execution command is a trick used to put bits of the user's input inside the `\next@` scratch macro *with the user's catcodes intact*: `\xy@@ix@{`...`}` is the same as `\xy@@{\global\toks9={`...`}}` except for the category codes used for the ....

```
\xydef@\xyxy@@ix@{\begingroup
 \xyuncatcodes\afterassignment\endgroup\global\toks9=}
```

This to save some tokens – maybe not worth it:

```
\xydef@\xy@@{\xy@{}}
```

Finally this to establish a sane state – only use within a group!

```
\xydef@\plainxy@{\let\xy@=\xyxy@ \change@oxy@\xy@ \let\xy@@ix@=\xyxy@@ix@}
```

To Do: Clean up all uses of these such that ⟨source⟩ is always that and only that. Define a method for 'inner' *aka* 'implied' ⟨source⟩ that doesn't really count in that it is a consequence of some other source...

1.2 Picture basics

The basic concepts involved when constructing Xy-pictures are positions and objects, and how they combine to form the state used by the graphic engine.

The general structure of an Xy-picture is as follows:

`\xy` ⟨pos⟩ ⟨decor⟩ `\endxy`

---

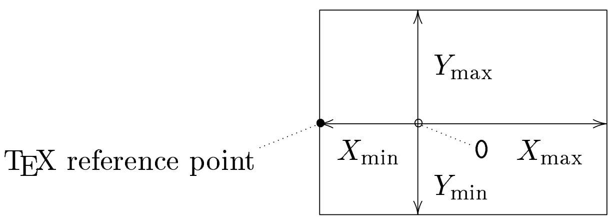
builds a box with an Xy-picture (LaTeX users may substitute `\begin{xy}` ... `\end{xy}` if they prefer).

⟨pos⟩ and ⟨decor⟩ are components of the special 'graphic language' which Xy-pictures are specified in. We explain the language components in general terms in this § and in more depth in the following §§.

The code for the `\xy`...`\endxy` command is presented last in this section.

1.2.1 Positions

All *positions* may be written `<`X`,`Y`>` where X is the TEX dimension distance *right* and Y the distance *up* from the *zero position* `0` of the XY-picture (`0` has coordinates `<0mm,0mm>`, of course). The zero position of the XY-picture determines the box produced by the `\xy`...`\endxy` command together with the four parameters $X_{\min}$, $X_{\max}$, $Y_{\min}$, and $Y_{\max}$ set such that all the objects in the picture are 'contained' in the following rectangle:

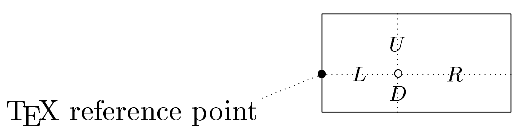


where the distances follow the "up and right > 0" principle, *e.g.*, the indicated TEX reference point has coordinates `<`$X_{\min}$`,0pt>` within the XY-picture. The zero position does not have to be contained in the picture, but $X_{\min} \le X_{\max} \wedge Y_{\min} \le Y_{\max}$ always holds. The possible positions are described in detail in §1.3.

When an XY-picture is entered in *math mode* then the reference point becomes the "vcenter" instead, *i.e.*, we use the point `<`$X_{\min}$`,-\the\fontdimen22>` as reference point.

1.2.2 Objects

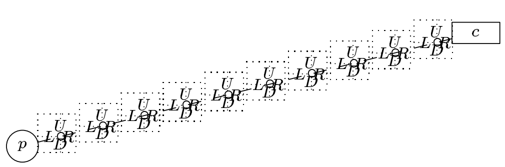
The simplest form of putting things into the picture is to 'drop' an *object* at a position. An object is like a TEX box except that it has a general *Edge* around its reference point—in particular this has the *extents* (*i.e.*, it is always contained within) the dimensions L, R, U, and D away from the reference point in each of the four directions left, right, up, and down. Objects are encoded in TEX boxes using the convention that the TEX reference point of an object is at its left edge, thus shifted `<`$-L$`,0pt>` from the center—so a TEX box may be said to be a rectangular object with L = `0pt`. Here is an example:



The object shown has a rectangle edge but others are available even though the kernel only supports rectangle and circle edges. It is also possible to use entire XY-pictures as objects with a rectangle edge, `0` as the reference point, $L = -X_{\min}$, $R = X_{\max}$, $D = -Y_{\min}$, and $U = Y_{\max}$. The commands for objects are described in §1.4.

1.2.3 Connections

Besides having the ability to be dropped at a position in a picture, all objects may be used to *connect* the two current objects of the state, *i.e.*, p and c. For most objects this is done by 'filling' the straight line between the centers with as many copies as will fit between the objects:



The ways the various objects connect are described along with the objects.

1.2.4 Decorations

When the `\xy` command reaches something that can not be interpreted as a continuation of the position being read, then it is expected to be a *decoration*, *i.e.*, in a restricted set of TEX commands which add to pictures. Most such commands are provided by the various *user options* (*cf.* §1.7)—only a few are provided within the kernel to facilitate programming of such options (and user macros) as described in §1.5.

1.2.5 The XY-pic state

Finally we summarise the user-accessible parts of the XY-picture state of two positions together with the last object associated with each: the *previous*, *p*, is the position $<X_p, Y_p>$ with the object L_p, R_p, D_p, U_p, $Edge_p$, and the *current*, *c*, is the position $<X_c, Y_c>$ with the object L_c, R_c, D_c, U_c, $Edge_c$.

Furthermore, XY-pic has a configurable *cartesian coordinate system* described by an *origin* position $<X_{origin}, Y_{origin}>$ and two *base vectors* $<X_{xbase}, Y_{xbase}>$ and $<X_{ybase}, Y_{ybase}>$ accessed by the usual notation using parentheses:

$$\begin{aligned}(x, y) \;=\; < \; & X_{origin} + x \times X_{xbase} + y \times X_{ybase} \;\;, \\ & Y_{origin} + x \times Y_{xbase} + y \times Y_{ybase} \;\; >\end{aligned}$$

This is explained in full when we show how to set the base in note 1.3d of §1.3.

Finally typesetting a connection will setup a "placement state" for referring to positions on the connection that is accessed through a special `?` position construction; this is also discussed in detail in §1.3.

The XY-pic *state* consists of all these parameters together. They are initialised to zero except for $X_{xbase} = Y_{ybase} =$ `1mm`.

The edges are are available to the programmer as token lists; see §1.8.2 for details.

Procedure: `\xy` ... `\endxy` builds an object from an XY-pic ⟨pos⟩ ⟨decor⟩ sequence as follows: (1.1114) `\xy` starts the `\hbox` to contain the XY-picture, (1.1115) starts an inner box to be resized appropriately later, sets `\xy@` to just execute immediately, and makes a fresh scope for global internal names, and (1.1117) initialises the XY-pic state (setting the size to a ridiculously large negative value), and finally passes control to the ⟨pos⟩ parser.

Some care is taken to 'lift' the diagram a bit to 'vcenter' it when in math mode, and `\mathsurround` is set to zero to make changes into math mode safe.

```
1107 \message{pictures: \string\xy,}

1109 \xydef@\xy{\ifmmode\expandafter\xymath@\else\expandafter\xynomath@\fi}
1110 \xydef@\xymath@{\hbox\bgroup \dimen@=\the\fontdimen22\textfont\tw@ \xyinside@}
1111 \xydef@\xynomath@{\hbox\bgroup \dimen@=\z@ \xyinside@}

1113 \xydef@\xyinside@{%
1114  \saveXyStyle@ \aftergroup\xycheck@end
1115  \setboxz@h\bgroup
1116   \plainxy@
1117   \X@c=\z@ \Y@c=\z@ \czeroEdge@
1118   \X@p=\z@ \Y@p=\z@ \U@p=\z@ \D@p=\z@ \L@p=\z@ \R@p=\z@ \Edge@p={\zeroEdge}%
1119   \X@min=\hsize \X@max=-\hsize \Y@min=\hsize \Y@max=-\hsize
1120   \mathsurround=\z@
1121   \expandafter\POS\everyxy@@}

1123 \xydef@\czeroEdge@{\U@c=\z@ \D@c=\U@c \L@c=\U@c \R@c=\U@c \Edge@c={\zeroEdge}}
```

```
1125 \xydef@\xyxy@#1#2{#2}

1127 \xywarnifdefined\everyxy
1128 \expandafter\def\addEQ@\everyxy#1{\def\everyxy@@{#1}\ignorespaces}

1130 \xylet@\everyxy@@=\empty
```

When finished `\endxy` does a `\relax` to disable any parser still active and (1.1144) resets the size of the generated box to zero if no (unhidden) objects were inserted, and (1.1148) defines a command to end both the temporary and the 'proper' box and set its size correctly—this uses `\edef` to expand the required dimensions used within the temporary box before leaving the two groups (namely the temporary box and the 'proper' box).

```
1142 \xydef@\endxy{\if\inxy@\else\xyerror@{Unexpected \string\endxy}{}\fi
1143  \relax
1144   \dimen@=\Y@max \advance\dimen@-\Y@min
1145   \ifdim\dimen@<\z@ \dimen@=\z@ \Y@min=\z@ \Y@max=\z@ \fi
1146   \dimen@=\X@max \advance\dimen@-\X@min
1147   \ifdim\dimen@<\z@ \dimen@=\z@ \X@min=\z@ \X@max=\z@ \fi
1148   \edef\tmp@{\egroup
1149     \setboxz@h{\kern-\the\X@min\boxz@}%
1150     \ht\z@=\the\Y@max \dp\z@=-\the\Y@min \wdz@=\the\dimen@
1151     \noexpand\maybeunraise@ \raise\dimen@\boxz@
1152     \noexpand\recoverXyStyle@ \egroup \noexpand\xy@end
1153     \U@c=\the\Y@max \D@c=-\the\Y@min \L@c=-\the\X@min \R@c=\the\X@max}\tmp@}

1155 \xydef@\maybeunraise@{\if\inxy@\else \dimen@ii=\dp\z@
1156   \ifdim\dimen@ii<\z@ \advance\dimen@\dimen@ii \fi\fi}
```

If an `\xy` is not properly closed by an `\endxy` then the error message is produced. This happens if (a) too many `\xy`s or (b) too many `\endxy`s or (c) if there is the correct number of each but the grouping becomes unbalanced due to a misplaced `}` or `\egroup`.

```
1165 \xydef@\xycheck@end{\xyFN@\xycheck@end@}
1166 \xydef@\xycheck@end@{\ifx\next\xy@end\DN@\xy@end{}\else\DN@{\xy@end}\fi\next@}
1167 \xydef@\xy@end{%
1168  \xyerror@{An \string\xy\space environment is not closed correctly.}%
1169   {I expected \string\endxy.  You probably have an umatched {} grouping.}}
```

It is possible to insert an 'initial' piece of ⟨pos⟩ ⟨decor⟩ at the start of every XY-picture with the declaration

---

`\everyxy={` ⟨text⟩ `}`

This will act as if the ⟨text⟩ was typed literally right after each `\xy` command, parsing the actual contents as if it follows this – thus it is recommended that ⟨text⟩ has the form ⟨pos⟩`,` such that users can continue with ⟨pos⟩ ⟨decor⟩.

1.3 Positions

```
1195 \message{positions,}
```

A ⟨pos⟩ition is a way of specifying locations as well as dropping objects at them and decorating them—in fact any aspect of the XY-pic state can be changed by a ⟨pos⟩ but most will just change the coordinates and/or shape of c.

All possible positions are shown in figure 1.1 with explanatory notes below.

Syntax			Action
⟨pos⟩	⟶	⟨coord⟩	$c \leftarrow$ ⟨coord⟩
	$\mid$	⟨pos⟩ `+` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩ + ⟨coord⟩[1.3a]
	$\mid$	⟨pos⟩ `-` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩ − ⟨coord⟩[1.3a]
	$\mid$	⟨pos⟩ `!` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩ then skew[1.3b] c by ⟨coord⟩
	$\mid$	⟨pos⟩ `.` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩ but also covering[1.3c] ⟨coord⟩
	$\mid$	⟨pos⟩ `,` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩ then $c \leftarrow$ ⟨coord⟩
	$\mid$	⟨pos⟩ `;` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩, swap p and c, $c \leftarrow$ ⟨coord⟩
	$\mid$	⟨pos⟩ `:` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩, set base[1.3d], $c \leftarrow$ ⟨coord⟩
	$\mid$	⟨pos⟩ `::` ⟨coord⟩	$c \leftarrow$ ⟨pos⟩, $\mathit{ybase} \leftarrow c - \mathit{origin}$, $c \leftarrow$ ⟨coord⟩
	$\mid$	⟨pos⟩ `*` ⟨object⟩	$c \leftarrow$ ⟨pos⟩, drop[1.3f] ⟨object⟩
	$\mid$	⟨pos⟩ `**` ⟨object⟩	$c \leftarrow$ ⟨pos⟩, connect[1.3g] using ⟨object⟩
	$\mid$	⟨pos⟩ `?` ⟨place⟩	$c \leftarrow$ ⟨pos⟩, $c \leftarrow$ ⟨place⟩[1.3h]
	$\mid$	⟨pos⟩ `@` ⟨stacking⟩	$c \leftarrow$ ⟨pos⟩, do ⟨stacking⟩[1.3o]
	$\mid$	⟨pos⟩ `=` ⟨saving⟩	$c \leftarrow$ ⟨pos⟩, do ⟨saving⟩[1.3p]
⟨coord⟩	⟶	⟨vector⟩	⟨pos⟩ is ⟨vector⟩ with zero size
	$\mid$	⟨empty⟩ $\mid$ `c`	reuse last c (do nothing)
	$\mid$	`p`	p
	$\mid$	`x` $\mid$ `y`	axis intersection[1.3k] with $\overline{pc}$
	$\mid$	`s`⟨digit⟩ $\mid$ `s{`⟨number⟩`}`	stack[1.3o] position ⟨digit⟩ or ⟨number⟩ below the top
	$\mid$	`"`⟨id⟩`"`	restore what was saved[1.3p] as ⟨id⟩ earlier
	$\mid$	`{` ⟨pos⟩ ⟨decor⟩ `}`	the c resulting from interpreting the group[1.3l]
⟨vector⟩	⟶	`0`	zero
	$\mid$	`<` ⟨dimen⟩ `,` ⟨dimen⟩ `>`	absolute
	$\mid$	`<` ⟨dimen⟩ `>`	absolute with equal dimensions
	$\mid$	`(` ⟨factor⟩ `,` ⟨factor⟩ `)`	in current base[1.3d]
	$\mid$	`a (` ⟨number⟩ `)`	angle in current base[1.3e]
	$\mid$	⟨corner⟩	from reference point to ⟨corner⟩ of c
	$\mid$	⟨corner⟩ `(` ⟨factor⟩ `)`	The ⟨corner⟩ multiplied with ⟨factor⟩
	$\mid$	`/` ⟨direction⟩ ⟨dimen⟩ `/`	vector ⟨dimen⟩ in ⟨direction⟩[1.3m]
⟨corner⟩	⟶	`L` $\mid$ `R` $\mid$ `D` $\mid$ `U`	offset[1.3n] to left, right, down, up side
	$\mid$	`CL` $\mid$ `CR` $\mid$ `CD` $\mid$ `CU` $\mid$ `C`	offset[1.3n] to center of side, true center
	$\mid$	`LD` $\mid$ `RD` $\mid$ `LU` $\mid$ `RU`	offset[1.3n] to actual left/down, … corner
	$\mid$	`E` $\mid$ `P`	offset[1.3n] to nearest/proportional edge point to p
⟨place⟩	⟶	`<` ⟨place⟩ $\mid$ `>` ⟨place⟩	shave[1.3h] `(0)`/`(1)` to edge of p/c, $f \leftarrow$ `0`/`1`
	$\mid$	`(` ⟨factor⟩ `)` ⟨place⟩	$f \leftarrow$ ⟨factor⟩
	$\mid$	⟨slide⟩	pick place[1.3h] and apply ⟨slide⟩
	$\mid$	`! {`⟨pos⟩`}` ⟨slide⟩	intercept[1.3j] with line setup by ⟨pos⟩ and apply ⟨slide⟩
⟨slide⟩	⟶	`/` ⟨dimen⟩ `/`	slide[1.3i] ⟨dimen⟩ further along connection
	$\mid$	⟨empty⟩	no slide

Figure 1.1: ⟨pos⟩itions.

Exercise 1.1: Which of the positions `0`, `<0pt,0pt>`, `<0pt>`, `(0,0)`, and `/0pt/` is different from the others?

Parsing: First the `\POS` and `\afterPOS` ⟨decor⟩ations, and similar `\afterCOORD` and `\afterVECTORorEMPTY` ones. They handle parsing of ⟨pos⟩, ⟨coord⟩, and ⟨vector⟩; parsing of ⟨corner⟩ and ⟨place⟩ is presented along with note 1.3n and 1.3h explaining them.

```
\xydef@\POS{\afterPOS{}}
```

```
\xydef@\afterPOS#1{%
 \DN@##1{\def\afterPOS@{\def\afterPOS@{##1}#1}}%
 \expandafter\next@\expandafter{\afterPOS@}%
 \afterCOORD{\xyFN@\POS@}}
```

```
\xylet@\afterPOS@=\empty
```

```
\xydef@\afterCOORD#1{%
 \DN@##1{\def\afterCOORD@{\def\afterCOORD@{##1}#1}}%
 \expandafter\next@\expandafter{\afterCOORD@}%
 \afterVECTORorEMPTY{\xy@@\czeroEdge@ \afterCOORD@}{\xyFN@\COORD@}}
```

```
\xylet@\afterCOORD@=\empty
```

```
\xydef@\afterVECTORorEMPTY#1#2{%
 \DN@##1{\def\afterVECTOR@{\def\afterVECTOR@{##1}%
  \ifVECTORempty@\DN@{#2}\else\DN@{#1}\fi \next@}}%
 \expandafter\next@\expandafter{\afterVECTOR@}%
 \xyFN@\VECTOR@}
```

```
\xynew@{if}\ifVECTORempty@
\xylet@\afterVECTOR@=\empty
```

The `\afterVECTORorEMPTY` command is special in that it takes two arguments: the 'continuation' if a ⟨vector⟩ was found and the continuation if ⟨empty⟩ was found (this is not applicable to the other two since ⟨empty⟩ is a legal ⟨coord⟩ and thus also a legal ⟨pos⟩).

Next we proceed with the actual parsing primitives: `\COORD@`, `\POS@`, and `\VECTOR@`. These are bound to `\xyCOORD@`, `\xyPOS@`, and `\xyVECTOR@` in order to be extendable, *e.g.*, the `matrix` option extends ⟨coord⟩ to support the `[`*row*`,` *column*`]` format by redefining `\COORD@` to first test for this new format and then call `\xyCOORD@`.

The parsing commands above are set up such that they all first call the `\VECTOR@` command. ⟨coord⟩ and ⟨pos⟩ parsing then proceeds with calling the `\COORD@` if there was no ⟨vector⟩. ⟨pos⟩ parsing then calls `\POS@` to continue the ⟨pos⟩ (in both cases).

First ⟨vector⟩s:

```
\xydef@\xyVECTOR@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\VECTOR@}%gobble spaces
 \else \ifcat A\noexpand\next \let\next@=\VECTOR@letter
 \else \let\next@=\VECTOR@other \fi\fi \next@}
```

```
\xylet@\VECTOR@=\xyVECTOR@
```

All letters used for ⟨vector⟩s are uppercase ⟨corner⟩s except for `a` used for angles (where the main code is in note 1.3e); this is also where we introduce the auxiliary `\notrelaxorelse@` that takes two control sequences and expands the first unless it is relax in which case it expands the second:

```
\def\notrelaxorelse@#1#2{\ifx#1\relax \expandafter#2\else\expandafter#1\fi}
```

```
\xydef@\VECTOR@letter{%
```

```
\ifx a\next \expandafter\VECTOR@a \else \expandafter\CORNER@ \fi}
```

```
\xydef@\VECTOR@a a(#1){\xy@{a(#1)}{\vfromcartesianangle@{#1}}%
\VECTORempty@false \afterVECTOR@}
```

The ⟨corner⟩ trick is to do nothing when there is nothing and initialise both X and Y in all other cases.

```
\xydef@\CORNER@{%
\xy@{}{\A@=-.5\L@c \advance\A@.5\R@c \B@=-.5\D@c \advance\B@.5\U@c
 \let\nextii@=\zeroit@}%
\VECTORempty@true\CORNER@i}
```

```
\xydef@\zeroit@#1{#1=\z@}
```

```
\xydef@\CORNER@i{%
\ifx D\next      \DN@ D{\xy@{D}{\Y@c=-\D@c \nextii@\X@c \B@=\Y@c}\CORNER@ii}%
\else\ifx U\next \DN@ U{\xy@{U}{\Y@c= \U@c \nextii@\X@c \B@=\Y@c}\CORNER@ii}%
\else\ifx L\next \DN@ L{\xy@{L}{\X@c=-\L@c \nextii@\Y@c \A@=\X@c}\CORNER@ii}%
\else\ifx R\next \DN@ R{\xy@{R}{\X@c= \R@c \nextii@\Y@c \A@=\X@c}\CORNER@ii}%
\else\ifx C\next \DN@ C{\xy@{C}{\X@c= \A@ \Y@c= \B@}\CORNER@ii}%
\else\ifx E\next \DN@ E{\xy@{E}{%
   \A@=\X@c \B@=\Y@c \the\Edge@c\z@ \advance\X@c-\A@ \advance\Y@c-\B@}%
  \CORNER@ii}%
\else\ifx P\next \DN@ P{\xy@{P}{%
   \A@=\X@c \B@=\Y@c \the\Edge@c\thr@@ \advance\X@c-\A@ \advance\Y@c-\B@}%
  \CORNER@ii}%
\else\ifx (\next %)
  \DN@(##1){\xy@{(##1)}{\X@c=##1\X@c \Y@c=##1\Y@c}\afterVECTOR@}%
\else \let\next@=\afterVECTOR@
\fi\fi\fi\fi\fi\fi\fi\fi \next@}
```

```
\xydef@\CORNER@ii{\xy@@{\let\nextii@=\eat@}%
\VECTORempty@false \xyFN@\CORNER@i}
```

`\CORNER@i` recognises the (⟨factor⟩) also; this does no harm as it was never called if the first character was a `(`.

The remaining ⟨vector⟩ forms just set X and Y.

```
\xydef@\VECTOR@other{%
\addLT@\ifx \next
 \addGT@{\addLT@\DN@##1}{\xy@{<##1>}{\vfromabsolute@{##1}}%
  \VECTORempty@false\afterVECTOR@}%
\else\ifx (\next %)
 \DN@({\xyFN@\VECTOR@other@open}%)
\else\ifx /\next %/
 \DN@/##1/{\xy@@ix@{{##1}}%
  \xy@{/##1/}{\expandafter\vfromslide@\the\toks9 }%
  \VECTORempty@false\afterVECTOR@}%
\else\ifx 0\next
 \DN@ 0{\xy@{0}{\X@c=\z@ \Y@c=\z@}\VECTORempty@false\afterVECTOR@}%
\else
 \DN@{\VECTORempty@true\afterVECTOR@}%
\fi\fi\fi\fi \next@}
```

Hack: escape out in case of the special `(*` introducer…picked up by the ⟨coord⟩ parser, then.

```
\xydef@\VECTOR@other@open{%
 \ifx *\next \DN@{\VECTORempty@true \xyFN@\afterVECTOR@(}%)
 \else
  \DN@##1){\xy@{(##1)}{\vfromcartesian@{##1}}\VECTORempty@false\afterVECTOR@}%
 \fi \next@}
```

Next ⟨coord⟩inates that are not ⟨vector⟩s:

```
\xydef@\xyCOORD@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\COORD@}%gobble spaces
 \else \ifcat A\noexpand\next \let\next@=\xyCOORD@letter
 \else \let\next@=\xyCOORD@other \fi\fi \next@}

\xylet@\COORD@=\xyCOORD@

\xydef@\xyCOORD@letter{%
 \ifx c\next
  \DN@ c{\xy@{c}{}\afterCOORD@}%
 \else\ifx p\next
  \DN@ p{\xy@{p}\cfromp@ \afterCOORD@}%
 \else\ifx x\next
  \DN@ x{\xy@{x}{\R@c=\X@xbase \U@c=\Y@xbase \intersect@}\afterCOORD@}%
 \else\ifx y\next
  \DN@ y{\xy@{y}{\R@c=\X@ybase \U@c=\Y@ybase \intersect@}\afterCOORD@}%
 \else\ifx s\next
  \DN@ s##1{\xy@{s{##1}}{\cfroms@{##1}}\afterCOORD@}%
 \else \let\next@=\afterCOORD@ \fi\fi\fi\fi\fi \next@}

\xydef@\xyCOORD@other{%
 \ifx "\next %"
  \DN@"##1"{\xy@{"##1"}{\cfromid@{##1}}\afterCOORD@}%
 \else\ifx \bgroup\next
  \DN@##1{\xy@{{##1}}{\enter@{\pfromthep@\basefromthebase@}}%
   \silencexy@ \POS##1\relax \unsilencexy@ \xy@@\leave@ \afterCOORD@}%
 \else\ifx (\next %)
  \DN@({\xyFN@\xyCOORD@other@open}%)
 \else \let\next@=\afterCOORD@ \fi\fi\fi \next@}

\xynew@{if}\ifsilentxy@

\xydef@\silencexy@{%
 \ifsilentxy@ \nter@{}%
 \else \nter@{\silentxy@false \let\xy@=\unsilent@@xy@}
  \silentxy@true \let\unsilent@@xy@=\xy@ \def\xy@##1##2{\unsilent@@xy@{}{##2}}%
 \fi}

\xydef@\unsilencexy@{\leave@}

\xydef@\xyCOORD@other@open{%
 \ifx *\next
  \DN@*##1*){\xy@{(*}{\enter@{\pfromthep@\basefromthebase@}}%
   \POS##1\relax \xy@{*)}\leave@ \afterCOORD@}%
 \else \DN@{\xyFN@\afterCOORD@(}%)
 \fi \next@}
```

Finally ⟨pos⟩ parsing after ⟨coord⟩ (possibly ⟨vector⟩) is interpreted:

```
\xydef@\xyPOS@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\POS@}%gobble spaces
 \else\addPLUS@\ifx \next
  \addPLUS@\DN@{\xy@+{\enter@\cplusthec@}%
    \afterCOORD{\xy@@\leave@ \xyFN@\POS@}}%
 \else\addDASH@\ifx \next
  \addDASH@\DN@{\xy@-{\enter@\cplusthec@}%
    \afterCOORD{\xy@@{\X@c=-\X@c \Y@c=-\Y@c\leave@}\xyFN@\POS@}}%
 \else\ifx !\next
  \DN@ !{\xy@!{\enter@\cskewthec@}\afterCOORD{\xy@@\leave@ \xyFN@\POS@}}%
 \else\addDOT@\ifx \next
  \addDOT@\DN@{\xy@.{\enter@\cmergethec@}%
   \afterCOORD{\xy@@\leave@ \xyFN@\POS@}}%
 \else\ifx ,\next
  \DN@ ,{\xy@,{\comma@@}\afterCOORD{\xyFN@\POS@}}%
 \else\ifx ;\next
  \DN@ ;{\xy@;{\swap@}\afterCOORD{\xyFN@\POS@}}%
 \else\ifx :\next
  \DN@ :{\xyFN@\POS@colon}%
 \else\addEQ@\ifx \next
  \addEQ@\DN@{\xyFN@\saveid@}%
 \else\ifx *\next
  \DN@ *{\xyFN@\POS@star}%
 \else \ifx ?\next
  \DN@?{\xy@?{}\afterPLACE{\xyFN@\POS@}}%
 \else \addAT@\ifx \next
  \addAT@\DN@{\xyFN@\STACK@}%
 \else
  \let\next@=\afterPOS@
 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}
```

```
\xylet@\comma@@=\relax
```

```
\xylet@\POS@=\xyPOS@
```

`\comma@@` is a hook used to change the operation of `,`, *e.g.*, when reading a stack setup where it means ‘push’.

The final functions serve only to distinguish between the single character `:`/`*` and dual character `::`/`**` operators:

```
\xydef@\POS@colon{\DNii@{\afterCOORD{\xyFN@\POS@}}%
 \ifx :\next \xy@{::}{\setbase@@\X@c\Y@c}\DN@:{\nextii@}%
 \else \xy@:{\setbase@\X@p\Y@p\X@c\Y@c}\let\next@=\nextii@ \fi
 \next@}
```

```
\xydef@\POS@star{%
 \ifx *\next
  \DN@*##1##{\nextii@{##1}}%
  \DNii@##1##2{\xy@@ix@{{##1}{##2}}%
   \xy@{**##1{##2}}{\expandafter\connect@\the\toks9 }\xyFN@\POS@}%
 \else
```

```
  \DN@##1##{\nextii@{##1}}%
  \DNii@##1##2{\xy@@ix@{{##1}{##2}}%
   \xy@{*##1{##2}}{\expandafter\drop@\the\toks9 }\xyFN@\POS@}%
 \fi
 \next@}
```

Simple actions: Next follow the simplest actions; the complicated ones are explained along with their notes below.

```
\xydef@\cfromp@{\X@c=\X@p \Y@c=\Y@p \U@c=\U@p \D@c=\D@p \L@c=\L@p \R@c=\R@p
 \Edge@c=\expandafter{\the\Edge@p}}
```

```
\xydef@\pfromc@{\X@p=\X@c \Y@p=\Y@c \U@p=\U@c \D@p=\D@c \L@p=\L@c \R@p=\R@c
 \Edge@p=\expandafter{\the\Edge@c}}
```

```
\xydef@\swapdimen@#1#2{\dimen@=#1\relax #1=#2\relax #2=\dimen@}
```

```
\xynew@{toks}\swaptoks@@
```

```
\xydef@\swap@{\swapdimen@\X@c\X@p \swapdimen@\Y@c\Y@p
 \swapdimen@\U@c\U@p\swapdimen@\D@c\D@p \swapdimen@\L@c\L@p\swapdimen@\R@c\R@p
 \swaptoks@@=\Edge@c \Edge@c=\Edge@p \Edge@p=\swaptoks@@}
```

Next the parsing of coordinate pairs in `<>`:

```
\xydef@\vfromabsolute@#1{\vfromabsolute@@#1,@}
```

```
\xydef@\vfromabsolute@@#1,#2@{\X@c=#1\relax
 \DN@{#2}\ifx\next@\empty \Y@c=\X@c
 \else \DN@##1,{\Y@c=##1}\next@#2\relax \fi
% \advance\X@c 1sp \advance\Y@c 1sp %HACK
}
```

The next group of commands are used to store on the control stack with the `\enter@` command, so they *expand* to something useful:

```
\xydef@\cfromthec@{\X@c=\the\X@c \Y@c=\the\Y@c
 \U@c=\the\U@c \D@c=\the\D@c \L@c=\the\L@c \R@c=\the\R@c
 \Edge@c={\expandafter\noexpand\the\Edge@c}}
```

```
\xydef@\cfromthep@{\X@c=\the\X@p \Y@c=\the\Y@p
 \U@c=\the\U@p \D@c=\the\D@p \L@c=\the\L@p \R@c=\the\R@p
 \Edge@c={\expandafter\noexpand\the\Edge@p}}
```

```
\xydef@\pfromthep@{\X@p=\the\X@p \Y@p=\the\Y@p
 \U@p=\the\U@p \D@p=\the\D@p \L@p=\the\L@p \R@p=\the\R@p
 \Edge@p={\expandafter\noexpand\the\Edge@p}}
```

```
\xydef@\pfromthec@{\X@p=\the\X@c \Y@p=\the\Y@c
 \U@p=\the\U@c \D@p=\the\D@c \L@p=\the\L@c \R@p=\the\R@c
 \Edge@p={\expandafter\noexpand\the\Edge@c}}
```

Notes

1.3a. When doing arithmetic with `+` and `-` then the resulting current object inherits the size of the ⟨coord⟩, *i.e.*, the right argument—this will be zero if the ⟨coord⟩ is a ⟨vector⟩.

Exercise 1.2: How do you set c to an object the same size as the saved object `"ob"` but moved `<`X`,`Y`>`?

```
\xydef@\cplusthec@{\advance\X@c\the\X@c \advance\Y@c\the\Y@c}
```

1.3b. *Skewing* using `!` just means that the reference point of c is moved with as little change to the shape of the object as possible, *i.e.*, the edge of c will remain in the same location except that it will grow larger to avoid moving the reference point outside c.

Exercise 1.3: What does the ⟨pos⟩ ... `!R-L` do? **Bug:** The result of `!` is always a rectangle currently.

Procedure: `!` moves the center of c by a temporarily read c' and then readjusts the extents:

$$\begin{array}{rcl l l}
D_c & := & Y' + Y_c - \min(Y' - D', Y' + Y_c) & = & \max(Y_c + D', 0) \\
U_c & := & \max(Y' + U', Y' + Y_c) - (Y' + Y_c) & = & \max(U' - Y_c, 0) \\
Y_c & := & Y' + Y_c & & \\
L_c & := & X' + X_c - \min(X' - L', X' + X_c) & = & \max(X_c + L', 0) \\
R_c & := & \max(X' + R', X' + X_c) - (X' + X_c) & = & \max(R' - X_c, 0) \\
X_c & := & X' + X_c & &
\end{array}$$

```
\xydef@\cskewthec@{%
 \noexpand\cskew@{\the\Y@c}{\the\X@c}{\the\D@c}{\the\U@c}{\the\L@c}{\the\R@c}}
\xydef@\cskew@#1#2#3#4#5#6{%
 \D@c=#3\advance\D@c \Y@c \ifdim\D@c<\z@ \D@c=\z@ \fi
 \U@c=#4\advance\U@c-\Y@c \ifdim\U@c<\z@ \U@c=\z@ \fi
 \advance\Y@c#1%
 \L@c=#5\advance\L@c \X@c \ifdim\L@c<\z@ \L@c=\z@ \fi
 \R@c=#6\advance\R@c-\X@c \ifdim\R@c<\z@ \R@c=\z@ \fi
 \advance\X@c#2%
 \Edge@c={\rectangleEdge}}
```

1.3c. A ⟨pos⟩ *covers* another if it is a rectangle with size sufficiently large that the other is "underneath". The `.` operation "extends" a ⟨pos⟩ to cover an additional one—the reference point of c is not moved but the shape is changed to a rectangle such that the entire p object is covered.

Bug: non-rectangular objects are first "translated" into a rectangle by using a diagonal through the object as the diagonal of the rectangle.

Procedure: `.` takes a temporary object c' and adjusts the extents of c such that it is covered.

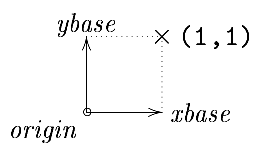
$$\begin{array}{rcl l l}
L_c & := & X' - \min(X' - L_c, X - L) & = & \max(L_c, A + L) \\
R_c & := & \max(X' + R_c, X + R) - X' & = & \max(R_c, -A + R) \\
D_c & := & Y' - \min(Y' - D_c, Y - D) & = & \max(D_c, B + D) \\
U_c & := & \max(Y' + U_c, Y + U) - Y' & = & \max(U_c, -B + U)
\end{array}$$

with `<`A`,`B`>` = `<`$X'-X$`,`$Y'-Y$`>`. First method 2 of the object is used to convert it into a rectangle.

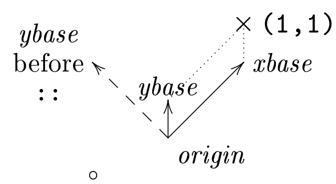
```
\xydef@\cmergethec@{%
 \noexpand\cmerge@{\the\Y@c}{\the\X@c}{\the\D@c}{\the\U@c}{\the\L@c}{\the\R@c}}
```

```
\xydef@\cmerge@#1#2#3#4#5#6{\the\Edge@c4%
 \A@=#2\advance\A@-\X@c \B@=#1\advance\B@-\Y@c
 \dimen@=#5\advance\L@c \A@ \ifdim\L@c<\dimen@ \L@c=\dimen@ \fi
 \dimen@=#6\advance\R@c-\A@ \ifdim\R@c<\dimen@ \R@c=\dimen@ \fi
 \dimen@=#3\advance\D@c \B@ \ifdim\D@c<\dimen@ \D@c=\dimen@ \fi
 \dimen@=#4\advance\U@c-\B@ \ifdim\U@c<\dimen@ \U@c=\dimen@ \fi
 \advance\X@c\A@ \advance\Y@c\B@}
```

1.3d. The operations `:` and `::` set the *base* used for ⟨coord⟩inates having the form (x,y). The `:` operation will set $\texttt{<}X_{origin}\texttt{,}\ Y_{origin}\texttt{>}$ to p, $\texttt{<}X_{xbase}\texttt{,}\ Y_{xbase}\texttt{>}$ to $c - origin$, and $\texttt{<}X_{ybase}\texttt{,}\ Y_{ybase}\texttt{>}$ to $\texttt{<}{-Y_{xbase}}\texttt{,}\ X_{xbase}\texttt{>}$ (this ensures that it is a usual square coordinate system). The `::` operation may then be used afterwards to make nonsqare bases by just setting *ybase* to $c - origin$. Here are two examples: firstly `0;<1cm,0cm>:` sets the coordinate system



while `<1cm,.5cm>;<2cm,1.5cm>:<1cm,1cm>::` defines



where in each case the $\circ$ is at `0`, the base vectors have been drawn and the $\times$ is at `(1,1)`.

When working with cartesian coordinates these three special ⟨factor⟩s are particularly useful:

`\halfroottwo`	$0.70710678 \approx \frac{1}{2}\sqrt{2}$
`\partroottwo`	$0.29289322 \approx 1 - \frac{1}{2}\sqrt{2}$
`\halfrootthree`	$0.86602540 \approx \frac{1}{2}\sqrt{3}$

More can be defined using `\def` (or `\newcommand` in LaTeX).

```
\xydef@\halfroottwo{.70710678}
\xydef@\partroottwo{.29289322}
\xydef@\halfrootthree{.8660254}
```

Procedure: The code chosen by the parsing is very simple; the only tricky bit is to ensure that `\basefromthebase@` always expands to set the current base.

```
\xydef@\vfromcartesian@#1{\vfromcartesian@@#1@}

\xydef@\vfromcartesian@@#1,#2@{%
 \X@c=\X@origin \advance\X@c#1\X@xbase \advance\X@c#2\X@ybase
 \Y@c=\Y@origin \advance\Y@c#1\Y@xbase \advance\Y@c#2\Y@ybase}

\xydef@\setbase@#1#2#3#4{%
 \X@origin=#1\relax \Y@origin=#2\relax
 \X@xbase=#3\relax \advance\X@xbase-\X@origin
```

```
1872 \Y@xbase=#4\relax \advance\Y@xbase-\Y@origin
1873 \X@ybase=-\Y@xbase \Y@ybase=\X@xbase}
```

```
1875 \xydef@\setbase@@#1#2{%
1876 \X@ybase=#1\relax \advance\X@ybase-\X@origin
1877 \Y@ybase=#2\relax \advance\Y@ybase-\Y@origin}
```

```
1879 \xydef@\basefromthebase@{\X@origin=\the\X@origin \Y@origin=\the\Y@origin
1880   \X@xbase=\the\X@xbase \Y@xbase=\the\Y@xbase
1881   \X@ybase=\the\X@ybase \Y@ybase=\the\Y@ybase}
```

1.3e. An *angle* α in XY-pic is the same as the coordinate pair `(` $\cos\alpha$`,` $\sin\alpha$`)` where α must be an integer interpreted as a number of degrees. Thus the ⟨vector⟩ `a(0)` is the same as `(1,0)` and `a(90)` as `(0,1)`, etc.

The translation involves several steps: (1.1956) Normalise the argument to be within $[0^\circ : 360^\circ[$. (1.1959) Flip angle around x-axis and then y-axis to ensure it is in the first quadrant, *i.e.*, within $[0^\circ : 90^\circ[$. (1.1964) Flip around diagonal to ensure angle within $[0^\circ : 45^\circ[$. (1.1967) Find values $\phi \le \alpha < \psi$ from the table in figure 1.2 (using recursive table lookup – at most 3 tests needed). (1.2024) build vector (x, y) interpolated between the sin/cos values for ϕ and ψ using the formula

$$(\cos\phi + k(\cos\psi - \cos\phi), \sin\phi + k(\sin\psi - \sin\phi)) \quad , \quad \text{where } k = \frac{\alpha - \phi}{\psi - \phi}$$

(1.2035) build the chosen vector.

```
1955 \xydef@\vfromcartesianangle@#1{\enter@\basefromthebase@ \R@=#1\p@
1956 \B@=360\p@
1957 \loop@ \ifdim\R@<\z@ \advance\R@\B@ \repeat@
1958 \loop@ \ifdim\R@>\B@ \advance\R@-\B@ \repeat@
1959 \ifdim\R@<.5\B@\else \R@=-\R@ \advance\R@\B@
1960  \X@ybase=-\X@ybase \Y@ybase=-\Y@ybase \fi
1961 \B@=180\p@
1962 \ifdim\R@<.5\B@\else \R@=-\R@ \advance\R@\B@
1963  \X@xbase=-\X@xbase \Y@xbase=-\Y@xbase \fi
1964 \B@=90\p@
1965 \ifdim\R@<.5\B@ \let\nextiii@=\literal@
1966 \else \R@=-\R@ \advance\R@\B@ \def\nextiii@##1,##2@{##2,##1@}\fi
1967 \dimen@=\z@ \DN@{1,0@}%
1968 \dimen@ii=45\p@ \DNii@{.70710678,.70710678@}%
1969 \chooseangleinterval@
1970 {\chooseangleinterval@
1971  {\chooseangleinterval@
1972   {\chooseangleinterval@
1973    {\chooseangleinterval@
1974     {}%
1975     {4.090909}{.99677570,.08023846@}%
1976     {}}%
1977    {6}{.99452190,.10452846@}%
1978    {\chooseangleinterval@
1979     {}%
1980     {8.181818}{.98982144,.14231484@}%
1981     {}}}%
```

Vectors for angles in $[0° : 45°]$: contains all angles required to typeset fractions up to $\frac{n}{12} \times 2\pi$, $\frac{1}{16} \times 2\pi$, and $\frac{1}{24} \times 2\pi$ exactly, and two extra low ones to ensure that all gaps are less than $5°$ and the precision of all sine/cosines better than $\frac{1}{1000}$.

α	$(\cos\alpha, \sin\alpha)$	fractions of 2π	flipped fractions of 2π
0	(1, 0)	$\frac{0}{n}$	$\frac{1}{2}, \frac{1}{4}, \frac{2}{4}, \frac{2}{8}, \frac{3}{4}, \frac{3}{6}, \frac{3}{12}, \frac{4}{8}, \frac{5}{10}, \frac{6}{8}, \frac{6}{12}, \frac{9}{12}$
4.090909	(.99677570, .08023846)	–	–
6	(.99452190, .10452846)	–	–
8.181818	(.98982144, .14231484)	$\frac{3}{11}$	$\frac{8}{11}$
10	(.98480775, .17364818)	$\frac{7}{9}$	$\frac{2}{9}$
12.857143	(.97492791, .22252093)	$\frac{2}{7}$	$\frac{5}{7}$
15	(.96592583, .25881905)	$\frac{1}{24}$	
16.363636	(.95949297, .28173256)	$\frac{6}{11}$	$\frac{5}{11}$
18	(.95105652, .30901699)	$\frac{3}{10}, \frac{4}{5}, \frac{8}{10}$	$\frac{1}{5}, \frac{2}{10}, \frac{7}{10}$
20	(.93969262, .34202014)	$\frac{5}{9}$	$\frac{4}{9}$
22.5	(.92387953, .38268343)	$\frac{1}{16}$	
24.545455	(.90963200, .41541501)	$\frac{9}{11}$	$\frac{2}{11}$
25.714286	(.90096887, .43388374)	$\frac{4}{7}$	$\frac{3}{7}$
30	(.86602540, .5)	$\frac{1}{3}, \frac{1}{12}, \frac{2}{6}, \frac{3}{9}, \frac{4}{12}, \frac{5}{6}, \frac{7}{12}, \frac{10}{12}$	$\frac{1}{6}, \frac{2}{3}, \frac{2}{12}, \frac{4}{6}, \frac{5}{12}, \frac{6}{9}, \frac{8}{12}, \frac{11}{12}$
32.727273	(.84125353, .54064082)	$\frac{1}{11}$	$\frac{10}{11}$
36	(.80901699, .58778525)	$\frac{1}{10}, \frac{3}{5}, \frac{6}{10}$	$\frac{2}{5}, \frac{4}{10}, \frac{9}{10}$
38.571429	(.78183148, .62348980)	$\frac{6}{7}$	$\frac{1}{7}$
40	(.76604444, .64278761)	$\frac{1}{9}$	$\frac{8}{9}$
40.909091	(.75574957, .65486073)	$\frac{4}{11}$	$\frac{7}{11}$
45	(.70710678, .70710678)	$\frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}$	–

Figure 1.2: Computing angle vectors

```
  {10}{.98480775,.17364818@}%
  {\chooseangleinterval@
   {}%
   {12.857143}{.97492791,.22252093@}%
   {}}}%
 {15}{.96592583,.25881905@}%
 {\chooseangleinterval@
  {\chooseangleinterval@
   {}%
   {16.363636}{.95949297,.28173256@}%
   {}}%
  {18}{.95105652,.30901699@}%
  {\chooseangleinterval@
   {}%
   {20}{.93969262,.34202014@}%
   {}}}}%
{22.5}{.92387953,.38268343@}%
{\chooseangleinterval@
 {\chooseangleinterval@
  {\chooseangleinterval@
   {}%
   {24.545455}{.90963200,.41541501@}%
   {}}%
  {25.714286}{.90096887,.43388374@}%
  {}}%
 {30}{.86602540,.5@}%
 {\chooseangleinterval@
  {\chooseangleinterval@
   {}%
   {32.727273}{.84125353,.54064082@}%
   {}}%
  {36}{.80901699,.58778525@}%
  {\chooseangleinterval@
   {\chooseangleinterval@
    {}%
    {38.571429}{.78183148,.62348980@}%
    {}}%
   {40.909091}{.75574957,.65486073@}%
   {\chooseangleinterval@
    {}%
    {40}{.76604444,.64278761@}%
    {}}}}}%
\A@=\R@ \advance\A@-\dimen@
\ifdim\ifdim\A@<\z@-\fi\A@<.01\p@ \edef\next@{\expandafter\nextiii@\next@}%
\else \B@=\dimen@ii \advance\B@-\R@
 \ifdim\A@<\B@ \dimen@=\toradians@\A@
  \edef\next@{\next@ \expandafter\removePT@\the\dimen@ @}%
 \else \dimen@=-\toradians@\B@
  \edef\next@{\nextii@ \expandafter\removePT@\the\dimen@ @}%
```

```
  \fi
  \expandafter\interpolatepoint@\next@
  \edef\next@{\expandafter\nextiii@\next@}%
 \fi
 \expandafter\vfromcartesian@@\next@
 \leave@}
```

```
\xydef@\chooseangleinterval@#1#2#3#4{%
 \B@=#2\p@ \def\next{#3}%
 \ifdim\R@<\B@ \dimen@ii=\B@ \let\nextii@=\next #1%
 \else \dimen@=\B@ \let\next@=\next \ifdim\B@<\R@ #4\fi\fi}
```

```
\xydef@\interpolateinterval@#1,#2@#3,#4@{%
 \A@=#1\p@ \dimen@=#3\p@ \advance\dimen@-\A@ \advance\A@\next\dimen@
 \B@=#2\p@ \dimen@=#4\p@ \advance\dimen@-\B@ \advance\B@\next\dimen@
 \edef\next@{\expandafter\removePT@\the\A@,\expandafter\removePT@\the\B@ @}}
```

```
\xydef@\toradians@{0.01745329}
```

```
\xydef@\interpolatepoint@#1,#2@#3@{%
 \A@=#1\p@ \dimen@ii=#3\A@ \dimen@ii=-.5\dimen@ii \advance\A@#3\dimen@ii
  \dimen@=-#2\p@ \advance\A@#3\dimen@
 \B@=#2\p@ \dimen@ii=#3\B@ \dimen@ii=-.5\dimen@ii \advance\B@#3\dimen@ii
  \dimen@=#1\p@ \advance\B@#3\dimen@
 \edef\next@{\expandafter\removePT@\the\A@,\expandafter\removePT@\the\B@ @}}
```

Figure 1.3 shows all directions.

1.3f. To *drop* an ⟨object⟩ at c with `*` means to actually physically typeset it in the picture with reference position at c—how this is done depends on the ⟨object⟩ in question and is described in detail in §1.4. The intuition with a drop is that it typesets something at `<`X_c`,`Y_c`>` and sets the edge of c accordingly.

Procedure: (??) sets up the direction to allow for directionals and builds the requested ⟨object⟩ in the (global) `\lastobjectbox@` box, (1.2106) adjust the picture size unless it is a hidden object, setting `\dimen@` $= X_c - L_c$, and (1.2112) drop the object in the picture at the right point by setting `box0` and using the `\Drop@@` method.

```
2104 \xydef@\drop@#1#2{%*[drop1]
2105  \global\setbox\lastobjectbox@=\object#1{#2}%
2106  \ifHidden@ \dimen@=\X@c \advance\dimen@-\L@c \else
2107   \dimen@=\Y@c \advance\dimen@ \U@c \ifdim\Y@max<\dimen@ \Y@max=\dimen@ \fi
2108   \dimen@=\Y@c \advance\dimen@-\D@c \ifdim\dimen@<\Y@min \Y@min=\dimen@ \fi
2109   \dimen@=\X@c \advance\dimen@ \R@c \ifdim\X@max<\dimen@ \X@max=\dimen@ \fi
2110   \dimen@=\X@c \advance\dimen@-\L@c \ifdim\dimen@<\X@min \X@min=\dimen@ \fi
2111  \fi
2112  \ifInvisible@\else
2113   \setboxz@h{\kern\dimen@ \raise\Y@c\box\lastobjectbox@}%
2114   \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ {\Drop@@}\fi}
```

Note: All typesetting into a picture should use or emulate `\drop@`!

1.3g. The *connect* operation `**` will first compute a number of internal parameters describing the direction from p to c and then typesets a connection filled with copies of the ⟨object⟩ as illustrated

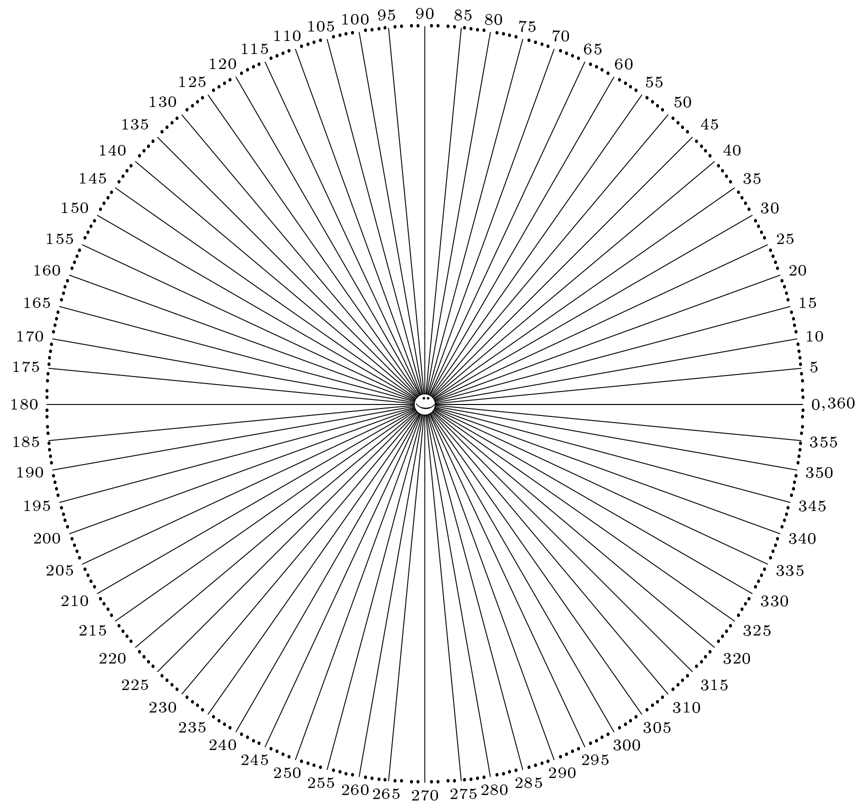


Figure 1.3: All directions.

in §1.2.3. The exact details of the connection depend on the actual ⟨object⟩ and are described in general in §1.4. The intuition with a connection is that it typesets something connecting p and c and sets the `?` ⟨pos⟩ operator up accordingly.

Procedure: Set up the direction to allow for directional objects, then save c, build the ⟨object⟩ in `\lastobjectbox@`, restore c, and perform the `\Connect@@` method to connect using `\lastobjectbox@`.

```
\xydef@\connect@#1#2{\setupDirection@ \enter@{\cfromthec@}%
 \global\setbox\lastobjectbox@=\object#1{#2}\leave@
 \Connect@@}
```

```
\xydef@\preconnect@#1#2{\setupDirection@ \enter@{\cfromthec@}%
 \global\setbox\lastobjectbox@=\object#1{#2}\leave@ \connectStore@
 \ifInvisible@ \Connect@@ \else \Invisible@true\Connect@@\Invisible@false \fi}
```

The `\preconnect` command is for internal use by arrow ... it makes use of the following:

```
\xynew@{box}\connectobjectbox@@
\xylet@\connectDrop@@=\empty
\xylet@\connectpreXY@style@=\empty
\xylet@\connectpostXY@style@=\empty
```

```
\xylet@\connectRest@@=\empty
```

```
\xydef@\connectStore@{%
 \global\setbox\connectobjectbox@@=\copy\lastobjectbox@
```

```
\let\connectDrop@@=\Drop@@
\let\connectpreXY@style@ =\preXY@style@
\let\connectpostXY@style@=\postXY@style@
\xdef\connectRest@@{%
 \def\noexpand\Upness@{\Upness@}%
 \def\noexpand\Leftness@{\Leftness@}%
 \ifInvisible@ \noexpand\Invisible@true \else \noexpand\Invisible@false \fi
 \ifHidden@ \noexpand\Hidden@true \else \noexpand\Hidden@false \fi}}
```

```
\xydef@\connectRestore@{%
\ifvbox\connectobjectbox@@
 \setbox\lastobjectbox@=\copy\voidb@x
\else
 \global\setbox\lastobjectbox@=\copy\connectobjectbox@@
\fi
\let\Drop@@=\connectDrop@@
\global\let\preXY@style@ =\connectpreXY@style@
\global\let\postXY@style@=\connectpostXY@style@
\connectRest@@}
```

See note 'define ⟨shape⟩' for the use of `\preXY@style@` and `\postXY@style@` to apply special ⟨style⟩s, specified via ⟨object-modifier⟩s; being saved here by `\connectStore@` as `\connectpreXY@style@` and `\connectpostXY@style@` to be reset by `\connectRestore@`.

1.3h. Using `?` will "pick a place" along the most recent connection typeset with `**`. What exactly this means is determined by the object that was used for the connection and by the modifiers described in general terms here.

The "shave" modifiers in a ⟨place⟩, `<` and `>`, change the default ⟨factor⟩, f, and how it is used, by 'moving' the positions that correspond to `(0)` and `(1)` (respectively): These are initially set equal to p and c, but shaving will move them to the point on the edge of p and c where the connection "leaves/enters" them, and change the default f as indicated. When one end has already been shaved thus then subsequent shaves will correspond to sliding the appropriate position(s) a TEX `\jot` (usually equal to `3pt`) further towards the other end of the connection (and past it). Finally the *pick* action will pick the position located the fraction f of the way from `(0)` to `(1)` where $f =$ `0.5` if it was not set (by `<`, `>`, or explicitly).

All this is probably best illustrated with some examples: each ⊗ in figure 1.4 is typeset by a sequence of the form p`;` c `**@{.}` `?`⟨place⟩ `*{\oplus}` where we indicate the `?`⟨place⟩ in each case. (We also give examples of ⟨slide⟩s.)

Procedure: The code for parsing ⟨place⟩ is the following. To get first `<>` to move to edge and the remaining to move a `\jot` we have both initial and continuing versions for each, the idea being that the second and following go to the edge of a small temporary object with radius `\jot`.

Note: This parser tests the *new parsing principle* that `\xy@` should always be called as `\xy@{` *source* `}{` *target* `}`!

```
\xydef@\afterPLACE#1{%
\DN@##1{\def\afterPLACE@{\xy@@\leave@ \def\afterPLACE@{##1}#1}}%
\expandafter\next@\expandafter{\afterPLACE@}%
\def\PLACEf@{{.5}}%
\xy@@{\enter@{\pfromthep@}%
```

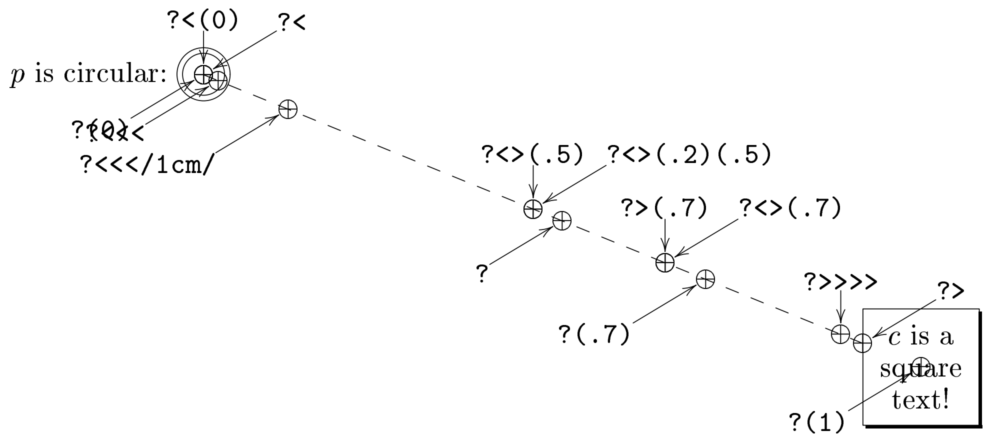


Figure 1.4: Example ⟨place⟩s

```
 \Creset@@ \def\PLACEf@{{.5}}%
  \let\PLACEedgep@@=\PLACEedgep@ \let\PLACEedgec@@=\PLACEedgec@}%
 \xyFN@\PLACE@}

\xydef@\PLACEf@{}

\xydef@\PLACEedgep@@{}
\xydef@\PLACEedgec@@{}

\xydef@\PLACEedgep@{\Cshavep@@ \def\PLACEedgep@@{\Cslidep@@\jot}}
\xydef@\PLACEedgec@{\Cshavec@@ \def\PLACEedgec@@{\Cslidec@@{-\jot}}}

\xylet@\afterPLACE@=\empty

\xydef@\PLACE@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PLACE@}%gobble spaces
 \else\addLT@\ifx \next
  \addLT@\DN@{\addLT@\xy@{\def\PLACEf@{{0}}\PLACEedgep@@}\xyFN@\PLACE@}%
 \else\addGT@\ifx \next
  \addGT@\DN@{\addGT@\xy@{\def\PLACEf@{{1}}\PLACEedgec@@}\xyFN@\PLACE@}%
 \else\ifx (\next %)
  \DN@(##1){\def\PLACEf@{{##1}}\xy@{(##1)}{\def\PLACEf@{{##1}}}\xyFN@\PLACE@}%
 \else\ifx !\next
  \DN@!{\xyFN@\PLACE@intercept}%
 \else
  \DN@{\xy@@{\expandafter\Calong@@\PLACEf@ \czeroEdge@}\PLACE@@}%
 \fi\fi\fi\fi\fi \next@}

\xydef@\PLACE@intercept{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PLACE@intercept}%gobble spaces
 \else\ifx \bgroup\next
  \DN@##1{\xy@{!{##1}}{}\PLACE@intercept@{##1}}%
 \else\ifx (\next %)
  \DN@(*##1*){\xy@{!(*##1*)}{}\PLACE@intercept@{##1}}%
 \else \DN@{\xyerror@{{<pos>} expected after ! in <place>}{}}%
 \fi\fi\fi \next@}

\xydef@\PLACE@intercept@#1{%
```

```
\xy@@{\enter@{\pfromthep@\basefromthebase@}\begingroup}%
\xy@@ix@{#1}\xy@@{\plainxy@ \expandafter\POS\the\toks9\relax
 \edef\next@{\endgroup
  \X@c =\the\X@c \Y@c=\the\Y@c \X@p=\the\X@p \Y@p=\the\Y@p}%
 \next@ \Cintercept@@ \leave@}%
\PLACE@@}
```

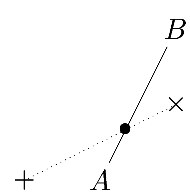
```
\xydef@\PLACE@@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PLACE@@}%gobble spaces
 \else\ifx /\next \DN@/##1/{\xy@{/##1/}{\Cslidec@@{##1}}\afterPLACE@}%
 \else \let\next@=\afterPLACE@
 \fi\fi \next@}
```

1.3i. A ⟨slide⟩ will move the position a dimension further along the connection at the picked position. For straight connections (the only ones kernel XY-pic provides) this is the same as adding a vector in the tangent direction, *i.e.*, `?`...`/`A`/` is the same as `?`...`+/`A`/`.

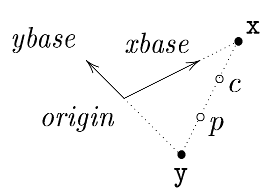
1.3j. This special ⟨place⟩ finds the point where the last connection intercepts with the line from p to c as setup by the ⟨pos⟩, thus usually this will have the form `!{`⟨coord⟩`;`⟨coord⟩`}`[4], for example, **Bug:** Only works for straight arrows at present.

```
\xy <1cm,0cm>:
 (0,0)*=0{+}="+" ;
 (2,1)*=0{\times}="*" **@{.} ,
 (1,0)*+{A} ; (2,2)*+{B} **@{-}
 ?!{"+";"*"} *{\bullet}
\endxy
```

will typeset



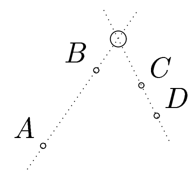
1.3k. The positions denoted by the *axis intersection* ⟨coord⟩inates `x` and `y` are the points where the line through p and c intersects with each axis. The following figure illustrates this:



Exercise 1.4: Given predefined points A, B, C, and D (stored as objects `"A"`, `"B"`, `"C"`, and `"D"`), write a ⟨coord⟩ specification that will return the point where the lines $\overline{AB}$ and $\overline{CD}$ cross

[4] The braces can be replaced by `(*`...`*)` *once*, *i.e.*, there can be no other braces nested inside it.

(the point marked with a large circle here):



Procedure: We solve the following equation in a, b:

$$\mathit{origin} + a \times \texttt{<}R_c\texttt{,}U_c\texttt{>} = c - b \times (c - p)$$

and then set

$$\texttt{<}X_c\texttt{,}Y_c\texttt{>} := \texttt{<}X_c\texttt{,}Y_c\texttt{>} - b \times (c - p) \quad \text{with zero size} \quad D_c, U_c, L_c, R_c := 0, 0, 0, 0.$$

The code uses $c = (X_c, Y_c, D_c, U_c, L_c, R_c)$ and A, B as temporaries and computes:

$$\begin{aligned}
\texttt{<}dX\texttt{,}dY\texttt{>} &:= \texttt{<}X_c\texttt{,}Y_c\texttt{>} - \texttt{<}X_p\texttt{,}Y_p\texttt{>} \\
\texttt{<}A\texttt{,}B\texttt{>} &:= \texttt{<}X_c\texttt{,}Y_c\texttt{>} - \texttt{<}X_{origin}\texttt{,}Y_{origin}\texttt{>} \\
\texttt{<}D_c\texttt{,}L_c\texttt{>} &:= \texttt{<}\begin{vmatrix} R & dX \\ U & dY \end{vmatrix}\texttt{,}\begin{vmatrix} R & A \\ U & B \end{vmatrix}\texttt{>} \\
\texttt{<}X_c\texttt{,}Y_c\texttt{>} &:= \texttt{<}X_c\texttt{,}Y_c\texttt{>} - (L_c/D_c) \times \texttt{<}dX\texttt{,}dY\texttt{>}
\end{aligned}$$

where we really do $D := (R/\texttt{pt})dY - (U/\texttt{pt})dX$ and similarly for L.

```
\xydef@\intersect@{
 \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
 \A@=\X@c \advance\A@-\X@origin \B@=\Y@c \advance\B@-\Y@origin
 \edef\next@{\expandafter\removePT@\the\R@c}%
 \edef\nextii@{\expandafter\removePT@\the\U@c}%
 \D@c=\next@\d@Y \advance\D@c-\nextii@\d@X \divide\D@c\KK@
 \L@c=\next@\B@ \advance\L@c-\nextii@\A@ \divide\L@c\KK@
 \ifdim\D@c=\z@\zeroDivide@\else \quotient@\next@\L@c\D@c \fi
 \advance\X@c-\next@\d@X \advance\Y@c-\next@\d@Y
 \czeroEdge@}
```

When there is no intersection point a wrong answer is returned, accompanied by a warning message. This behaviour can be altered by assigning a different value to the hook: `\zeroDivide@`. This macro must store a number in `\next@`.

```
\xydef@\zeroDivide@@{\zeroDivide@message{\intersect@}{treated as 0}\DN@{0}}
\xydef@\zeroDivide@message#1#2{\xywarning@{division by 0 in \string#1, #2}}
\xylet@\zeroDivide@=\zeroDivide@@
```

```
\xydef@\zeroDivideLimit@@{\ifdim\L@c=\z@ \DN@{0}%
 \else\ifdim\L@c<\z@\DN@{-\zeroDivide@Limit}%
 \else\DN@{\zeroDivide@Limit}\fi\fi
 \zeroDivide@message{\intersect@}{replaced by \zeroDivide@Limit}}
\xydef@\zeroDivideLimit@#1{\edef\zeroDivide@Limit{#1}%
 \let\zeroDivide@=\zeroDivideLimit@@}
\xylet@\zeroDivideLimit=\zeroDivideLimit@
```

By specifying `\zeroDivideLimit{`$\langle$num$\rangle$`}` the user can locally establish that $\frac{x}{0} = \operatorname{sgn}(x)\langle\text{num}\rangle$, whenever such a division by zero would otherwise occur in an intersection calculation.

1.3l. A ⟨pos⟩ ⟨decor⟩ *grouped* in `{}`-braces[5] is interpreted in a local scope in the sense that any p and *base* built within it are forgotten afterwards, leaving only the c as the result of the ⟨coord⟩. **Note:** Only p and *base* are restored – it is not a TeX group.

Exercise 1.5: What effect is achieved by using the ⟨coord⟩inate "`{;}`"?

The code is inside `\POS@`.

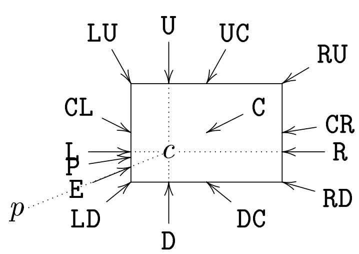
1.3m. The vector `/`Z`/`, where Z is a ⟨dimen⟩sion, is the same as the vector `<`$Z\cos\alpha$`,`$Z\sin\alpha$`>` where α is the angle of the last direction set by a connection (*i.e.*, with `**`) or subsequent placement (`?`) position.

```
\xydef@\vfromslide@#1{\enter@\DirectionfromtheDirection@ \begingroup
 \plainxy@\afterDIRECTIONorEMPTY\vfromslide@i\vfromslide@i#1@}
```

```
\xydef@\vfromslide@i#1@{\DN@{#1}%
 \edef\next{\endgroup
  \ifx\next@\empty \dimen@=.5pc \else \dimen@=#1\relax \fi
  \X@c=\cosDirection\dimen@ \Y@c=\sinDirection\dimen@}\next
 \leave@}
```

It is possible to give a ⟨direction⟩ as described in the next section (figure 1.5, note 1.4l in particular) that will then be used to set the value of α. It is also possible to omit the ⟨dimen⟩ in which case it is set to a default value of `.5pc`.

1.3n. A ⟨corner⟩ is an offset from the current `<`X_c`,`Y_c`>` position to a specific position on the edge of the c object (the two-letter ones may be given in any combination):



The 'edge point' `E` lies on the edge along the line from p to the centre of the object, in contrast to the 'proportional' point `P` which is also a point on the edge but computed in such a way that the object looks as much 'away from p' as possible.

Finally, a following `(`f`)` suffix will multiply the offset vector by the ⟨factor⟩ f.

Exercise 1.6: What is the difference between the ⟨pos⟩itions `c?<` and `c+E`?

Exercise 1.7: What does this typeset?

```
\xy *=<3cm,1cm>\txt{Box}*\frm{-}
 !U!R(.5) *\frm{..}*{\bullet} \endxy
```

Hint: `\frm` is defined by the frame extension and just typesets a frame of the kind indicated by the argument.

Bug: Currently only the single-letter corners (`L`, `R`, `D`, `U`, `C`, `E`, and `P`) will work for any shape—the others silently assume that the shape is rectangular.

[5] One can use `(*...*)` instead also here.

1.3o. The *stack* is a special construction useful for storing a sequence of ⟨pos⟩itions that are accessible using the special ⟨coord⟩inates `s`n, where n is either a single digit or a positive integer in `{}`s: `s0` is always the 'top' element of the stack and if the stack has depth d then the 'bottom' element of the stack has number `s{`$d-1$`}`. The stack is said to be 'empty' when the depth is 0 and then it is an error to access any of the `s`n or 'pop' which means remove the top element, shifting what is in `s1` to `s0`, `s2` to `s1`, etc. Similarly, 'push c' means to shift `s0` to `s1`, etc., and then insert the c as the new `s0`.

The stack is manipulated as follows:

`@`⟨stacking⟩	Action
`@+`⟨coord⟩	push ⟨coord⟩
`@-`⟨coord⟩	$c \leftarrow$ ⟨coord⟩ then pop
`@=`⟨coord⟩	load stack with ⟨coord⟩
`@@`⟨coord⟩	do ⟨coord⟩ for $c \leftarrow$ stack
`@i`	initialise
`@(`	enter new frame
`@)`	leave current frame

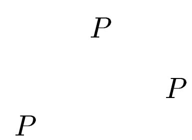
To 'load stack', means to load the entire stack with the positions set by ⟨coord⟩ within which `,` means 'push c'.

To 'do ⟨coord⟩ for all stack elements' means to set c to each element of the stack in turn, from the bottom and up, and for each interpret the ⟨coord⟩. Thus the first interpretation has c set to the bottom element of the stack and the last has c set to `s0`. If the stack is empty, the ⟨coord⟩ is not interpreted at all.

These two operations can be combined to repeat a particular ⟨coord⟩ for several points, like this:

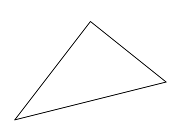
```
\xy
 @={(0,-10),(10,3),(20,-5)} @@{*{P}}
\endxy
```

will typeset



Finally, the stack can be forcibly cleared using `@i`, however, this is rarely needed because of `@(`, which saves the stack as it is, and then clears it, such when it has been used (and is empty), and `@)` is issued, then it is restored as it was at the time of the `@(`.

Exercise 1.8: How would you change the example above to connect the points as shown below?



First the stack top and bottom, both initially -1:

```
\xydef@\s@bot{-1}
\xydef@\s@top{-1}
```

Next the function to set $c \leftarrow \mathtt{s}n$, *i.e.*, test that $bot < n + bot < top$ and then run the associated stack element.

```
\xydef@\cfroms@#1{\tests@{#1}\runs@\outofranges@}

\xydef@\tests@#1#2#3{\DN@{#3}%
 \count@=\s@top \advance\count@-#1\relax
 \ifnum\count@>\s@bot\relax \ifnum\count@>\s@top\else\DN@{#2}\fi\fi
 \next@}

\xydef@\runs@{\csname S@\the\count@\endcsname}

\xydef@\outofranges@{\count@=\s@top \advance\count@-\s@bot
 \xyerror@{stack index out of range (should be 0..\the\count@)}{}}
```

Finally the actual code to do the stack operations: it depends on the ‘code’ passed after `@`; spaces are not allowed:

```
\xydef@\STACK@{%
 \addPLUS@\ifx\next
  \addPLUS@\DN@{\xy@{@+}{}\afterCOORD{\xy@@\spushc@ \xyFN@\POS@}}%
 \else\addDASH@\ifx\next
  \addDASH@\DN@{\xy@{@-}{}\afterCOORD{\xy@@\spop@ \xyFN@\POS@}}%
 \else \ifx i\next \DN@ i{\xy@{@i}\sinit@ \xyFN@\POS@}%
 \else \ifx (\next \DN@ ({\xy@{@(}\senter@ \xyFN@\POS@}%
 \else \ifx )\next \DN@ ){\xy@{@)}\sleave@ \xyFN@\POS@}%
 \else\addEQ@\ifx\next \addEQ@\DN@{\STACK@load}%
 \else\addAT@\ifx\next \addAT@\DN@{\xy@{@@}{}\smap@}%
 \else \DN@##1{\xyerror@{illegal stack command ##1}{}\afterCOORD{\xyFN@\POS@}}%
 \fi\fi\fi\fi\fi\fi\fi \next@}

\xydef@\STACK@load{\xy@{@=}{%
  \if\sempty@\else \xywarning@{loading on top of non-empty stack}\sinit@ \fi
  \let\comma@@=\spushc@}%
 \afterCOORD{\xy@@{\spushc@ \let\comma@@=\relax}\xyFN@\POS@}}

\xydef@\spushc@{%
 \count@=\s@top \advance\count@\@ne \edef\s@top{\the\count@}%
 \expandafter\edef\csname S@\s@top\endcsname{\cfromthec@}}

\xydef@\spushid@#1{\DNii@{#1}\edef\nextii@{\codeof\nextii@}%
 \expandafter\let\expandafter\next@\csname Q@\nextii@\endcsname
 \ifx\next@\relax \xyerror@{<pos> \string"\nextii@\string" not defined}{}%
 \else
  \count@=\s@top \advance\count@\@ne \edef\s@top{\the\count@}%
  \DNii@##1{\expandafter\def\csname S@\s@top\endcsname{##1}}%
  \expandafter\nextii@\expandafter{\next@}%
 \fi}

\xydef@\idfroms@#1#2{%
 \tests@{#2}{\DN@{\idfromxy@{#1}}%
  \expandafter\expandafter\expandafter\next@
   \expandafter\expandafter\expandafter{\csname S@\the\count@\endcsname}%
  }\outofranges@}

\xydef@\spop@{\count@=\s@top
```

```
 \ifnum\count@>\s@bot \advance\count@\m@ne \edef\s@top{\the\count@}%
 \else \xyerror@{nothing to pop from stack}{}\fi}
```

```
\xydef@\sinit@{\edef\s@top{\s@bot}}
```

```
\xydef@\senter@{%
 \count@=\s@top \advance\count@\@ne
 \expandafter\edef\csname S@\the\count@\endcsname{\s@bot}%
 \edef\s@bot{\the\count@}\edef\s@top{\the\count@}}
```

```
\xydef@\sleave@{%
 \ifnum\s@bot=\s@top\else \xywarning@{leaving non-empty stack}\sinit@ \fi
 \ifnum\s@bot>\m@ne \edef\s@bot{\csname S@\s@top\endcsname}%
  \count@=\s@top \advance\count@\m@ne \edef\s@top{\the\count@}%
 \edef\sbot{\the\count@}\fi}
```

```
\xydef@\sempty@{\ifnum\s@top=\s@bot TT\else TF\fi}
```

`\smap@` maps a ⟨coord⟩ over a stack:

```
\xydef@\xytotoks@#1#2{\addtotoks@{#2}}
\xydef@\xytotoks@@toksix@#1{\addtotoks@{\toks9={#1}}}
```

```
\xydef@\smap@{%
 \begingroup \toks@={}\let\xy@=\xytotoks@ \change@oxy@\xy@
  \let\xy@@ix@=\xytotoks@@toksix@
  \afterCOORD{\expandafter\endgroup
   \expandafter\smapxy@@\expandafter{\the\toks@}\xyFN@\POS@}}
```

```
\xydef@\smapxy@@#1{\xy@@{\edef\smapp@@{\s@bot}\smapxy@i{#1}}}
```

```
\xylet@\smapp@@=\empty
```

```
\xydef@\smapxy@i#1{%
 \ifnum\smapp@@<\s@top
  \count@=\smapp@@ \advance\count@\@ne \edef\smapp@@{\the\count@}%
  \DN@{\csname S@\smapp@@\endcsname #1\relax \smapxy@i{#1}}%
 \else \let\next@=\relax
 \fi \next@}
```

1.3p. It is possible to define new ⟨coord⟩inates on the form `"`⟨id⟩`"` by *saving* the current c using the ...`="`⟨id⟩`"` ⟨pos⟩ition form. Subsequent uses of `"`⟨id⟩`"` will then reestablish the c at the time of the saving.

Using a `"`⟨id⟩`"` that was never defined is an error, however, saving into a name that was previously defined just replaces the definition without warning, *i.e.*, `"`⟨id⟩`"` always refers to the last thing saved with that ⟨id⟩.

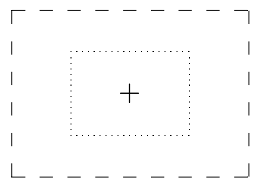
However, many other things can be 'saved': in general `@`⟨saving⟩ has either of the forms

`@:"`⟨id⟩`"`	`"`⟨id⟩`"` restores current *base*
`@`⟨coord⟩`"`⟨id⟩`"`	`"`⟨id⟩`"` reinterprets ⟨coord⟩
`@@"`⟨id⟩`"`	`@="`⟨id⟩`"` reloads this stack

The first form defines `"`⟨id⟩`"` to be a macro that restores the current *base*.

The second does not depend on the state at the time of definition at all; it is a macro definition. You can pass parameters to such a macro by letting it use coordinates named `"1"`, `"2"`, etc., and then use `="1"`, `="2"`, etc., just before every use of it to set the actual values of these. **Note:** it is not possible to use a ⟨coord⟩ of the form `"`⟨id⟩`"` directly: write it as `{"`⟨id⟩`"}`.

Exercise 1.9: Write a macro `"dbl"` to double the size of the current c object, *e.g.*, changing it from the dotted to the dashed outline in this figure:



The final form defines a special kind of macro that should only be used after the `@=` stack operation: the entire current stack is saved such that the stack operation `@="`⟨id⟩`"` will reload it.

Note: There is no distinction between the 'name spaces' of ⟨id⟩s used for saved coordinates and other things.

This parser distinguishes between the cases:

```
\xydef@\saveid@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\saveid@}%gobble spaces
 \else \ifx "\next\DN@"##1"{\xy@{="##1"}{\idfromc@{##1}}\xyFN@\POS@}%
 \else \ifx :\next\DN@:##1"##2"{\xy@{=:"##2"}{\idfrombase@{##2}}\xyFN@\POS@}%
 \else\addAT@\ifx\next
  \addAT@\DN@"##1"{\xy@{=@"##1"}{\idfromstack@{##1}}\xyFN@\POS@}%
 \else \ifx s\next
  \DN@ s##1"##2"{\xy@{=s##1"##2"}{\idfroms@{##2}{##1}}\xyFN@\POS@}%
 \else\addEQ@\ifx\next \let\saveid@COORD@@=\saveid@COORDii
  \addEQ@\DN@{\xyFN@\saveid@COORD}%
 \else \let\saveid@COORD@@=\saveid@COORDi \let\next@=\saveid@COORD
 \fi\fi\fi\fi\fi\fi \next@}
```

```
\xylet@\saveid@COORD@@=\relax
```

Here is the code for saving/restoring a position and a base.

```
\xydef@\idfromc@#1{\DN@{#1}%
 \expandafter\edef\csname Q@\codeof\next@\endcsname{\cfromthec@}}
```

```
\xydef@\idfrombase@#1{\DN@{#1}%
 \expandafter\edef\csname Q@\codeof\next@\endcsname{\basefromthebase@}}
```

```
\xydef@\idfromstack@#1{%
 \toks@={\if\sempty@\else
   \xywarning@{loading on top of non-empty stack}\sinit@ \fi}%
 \count@=\s@bot \advance\count@\@ne
 \ifnum\count@>\s@top\else
  \loop@
   \expandafter\let\expandafter\next@\csname S@\the\count@\endcsname
   \expandafter\addtotoks@\expandafter{\next@}%
  \ifnum\count@<\s@top \advance\count@\@ne \addtotoks@{\spushc@}\repeat@
 \fi
 \DN@{#1}\edef\next@##1{%
```

```
  \def\expandafter\noexpand\csname Q@\codeof\next@\endcsname{##1}}%
 \expandafter\next@\expandafter{\the\toks@}}
```

```
\xydef@\saveid@COORD{%
 \begingroup \toks@={}\let\xy@=\xytotoks@ \change@oxy@\xy@
  \let\xy@@ix@=\xytotoks@@toksix@
  \afterCOORD{\expandafter\saveid@COORDi\expandafter{\the\toks@}}}
```

```
\xydef@\saveid@COORDi#1#2"#3"{\endgroup \xy@@{\idfromxy@{#3}{#1}}\xyFN@\POS@}
\xydef@\saveid@COORDii#1#2"#3"{\endgroup \xy@@{\idfromcxy@{#3}{#1}}\xyFN@\POS@}
```

```
\xydef@\idfromxy@#1#2{\DN@{#1}%
 \expandafter\def\csname Q@\codeof\next@\endcsname{#2}}
```

```
\xydef@\idfromcxy@#1#2{\DN@{#1}%
 \expandafter\edef\csname Q@\codeof\next@\endcsname{\cfromthec@#2}}
```

```
\xydef@\cfromid@#1{\DNii@{#1}\edef\nextii@{\codeof\nextii@}%
 \expandafter\let\expandafter\next@\csname Q@\nextii@\endcsname
 \ifx\next@\relax \xyerror@{<pos> \string"\nextii@\string" not defined}{}%
 \else \expandafter\next@\fi}
```

1.4 Objects

```
\message{objects,}
```

Objects are the entities that are manipulated with the `*` and `**` ⟨pos⟩ operations above to actually get some output in XY-pictures. As for ⟨pos⟩itions the operations are interpreted strictly from left to right, however, the actual object is built *before* all the ⟨modifier⟩s take effect. The syntax of objects is given in figure 1.5 with references to the notes below. **Remark:** It is *never* allowed to include braces `{}` inside ⟨modifier⟩s! In case you wish to do something that requires `{`...`}` then check in this manual whether you can use `(*`...`*)` instead. If not then you will have to use a different construction!

We first discuss the parser and then summarise the required methods. The entry point to use of objects is `\object` described in note 1.4c. This should always be used because it initialises the token list and redefines `\xy@` to be `\addtotoks@` such that we can use ⟨pos⟩ parser routines within the ⟨object⟩!

Parsing: The ⟨object⟩ parser `\OBJECT@` will first parse the ⟨modifier⟩s, storing the action of each in sequence on the `\toks@` token list. When there are no more modifiers we insert `\objectbox` if we have reached the `{` otherwise we just assume that the rest of the ⟨object⟩ is some kind of box construction.

Note: The ⟨modifier⟩ actions doing shifts are implemented by having an independent vector for the shift: $\mathtt{<}R_p\mathtt{,}U_p\mathtt{>}$ always contains the vector from the current to the original TeX reference point; furthermore the initial L_c is saved as L_p such that we can retrieve the original (XY-pic) reference point again. Modifying the p values is safe because all actual changes are done in a local scope after the entire ⟨object⟩ is parsed and we have built the object box (in `\OBJECT@@`).

The ⟨modifier⟩s changing the direction are executed while parsing to make sure that the direction used when building the ⟨object⟩ is right, and restored in the right sequence while evaluating the ⟨modifier⟩s afterwards.

```
\xydef@\OBJECT@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\OBJECT@}%gobble spaces
 \else\ifcat A\noexpand\next \let\next@=\OBJECT@letter
 \else \let\next@=\OBJECT@other \fi\fi \next@}
```

<table>
<tr><th>Syntax</th><th></th><th></th><th>Action</th></tr>
<tr><td>⟨object⟩</td><td>⟶</td><td>⟨modifier⟩ ⟨object⟩</td><td>apply ⟨modifier⟩ to ⟨object⟩</td></tr>
<tr><td></td><td>|</td><td>⟨objectbox⟩</td><td>build ⟨objectbox⟩ then apply its ⟨modifier⟩s</td></tr>
<tr><td>⟨objectbox⟩</td><td>⟶</td><td>{ ⟨text⟩ }</td><td>build default[1.4a] object</td></tr>
<tr><td></td><td>|</td><td>⟨library object⟩ | @⟨dir⟩</td><td>use ⟨library object⟩ or ⟨dir⟩ectional (see §1.6)</td></tr>
<tr><td></td><td>|</td><td>⟨TEX box⟩ { ⟨text⟩ }</td><td>build box[1.4b] object with ⟨text⟩ using the given ⟨TEX box⟩ command, e.g., \hbox</td></tr>
<tr><td></td><td>|</td><td>\object ⟨object⟩</td><td>wrap up the ⟨object⟩ as a finished object box[1.4c]</td></tr>
<tr><td></td><td>|</td><td>\composite { ⟨composite⟩ }</td><td>build composite object box[1.4d]</td></tr>
<tr><td></td><td>|</td><td>\xybox { ⟨pos⟩ ⟨decor⟩ }</td><td>package entire XY-picture as object[1.4e]</td></tr>
<tr><td>⟨modifier⟩</td><td>⟶</td><td>! ⟨vector⟩</td><td>⟨object⟩ has its reference point shifted[1.4f] by ⟨vector⟩</td></tr>
<tr><td></td><td>|</td><td>!</td><td>⟨object⟩ has the original reference point reinstated</td></tr>
<tr><td></td><td>|</td><td>⟨add op⟩ ⟨size⟩</td><td>change ⟨object⟩ size[1.4g]</td></tr>
<tr><td></td><td>|</td><td>h | i</td><td>⟨object⟩ is hidden[1.4h], invisible[1.4i]</td></tr>
<tr><td></td><td>|</td><td>[⟨shape⟩]</td><td>⟨object⟩ is given the specified ⟨shape⟩[1.4j]</td></tr>
<tr><td></td><td>|</td><td>[= ⟨shape⟩]</td><td>define ⟨shape⟩[1.4k] to reestablish current object style</td></tr>
<tr><td></td><td>|</td><td>⟨direction⟩</td><td>set current direction for this ⟨object⟩</td></tr>
<tr><td>⟨add op⟩</td><td>⟶</td><td>+ | - | = | += | -=</td><td>grow, shrink, set, grow to, shrink to</td></tr>
<tr><td>⟨size⟩</td><td>⟶</td><td>⟨empty⟩</td><td>default size[1.4g]</td></tr>
<tr><td></td><td>|</td><td>⟨vector⟩</td><td>size as sides of rectangle covering the ⟨vector⟩</td></tr>
<tr><td>⟨direction⟩</td><td>⟶</td><td>⟨diag⟩</td><td>⟨diag⟩onal direction[1.4l]</td></tr>
<tr><td></td><td>|</td><td>v ⟨vector⟩</td><td>direction[1.4l] of ⟨vector⟩</td></tr>
<tr><td></td><td>|</td><td>q{ ⟨pos⟩ ⟨decor⟩ }</td><td>direction[1.4l] from p to c after ⟨pos⟩ ⟨decor⟩</td></tr>
<tr><td></td><td>|</td><td>⟨direction⟩ : ⟨vector⟩</td><td>vector relative to ⟨direction⟩[1.4m]</td></tr>
<tr><td></td><td>|</td><td>⟨direction⟩ _ | ⟨direction⟩ ^</td><td>90° clockwise/anticlockwise to ⟨direction⟩[1.4m]</td></tr>
<tr><td>⟨diag⟩</td><td>⟶</td><td>⟨empty⟩</td><td>last used direction (not necessarily diagonal[1.4l])</td></tr>
<tr><td></td><td>|</td><td>l | r | d | u</td><td>left, right, down, up diagonal[1.4l]</td></tr>
<tr><td></td><td>|</td><td>ld | rd | lu | ru</td><td>left/down, ... diagonal[1.4l]</td></tr>
<tr><td>⟨composite⟩</td><td>⟶</td><td>⟨object⟩</td><td>first object is required</td></tr>
<tr><td></td><td>|</td><td>⟨composite⟩ * ⟨object⟩</td><td>add ⟨object⟩ to composite object box[1.4d]</td></tr>
</table>

Figure 1.5: ⟨object⟩s.

```
3079 \xydef@\OBJECT@letter{%
3080  \ifx i\next \DN@ i{\addtotoks@\Invisible@true \xyFN@\OBJECT@}%
3081  \else\ifx h\next\DN@ h{\addtotoks@\Hidden@true \xyFN@\OBJECT@}%
3082  \else\ifx o\next\DN@ o{\xywarning@{Obsolete o modifier used}\OBJECT@shape{o}}%
3083  \else\ifx x\next\DN@ x{\xywarning@{Obsolete x modifier used}\OBJECT@shape{}}%
3084  \else\ifx @\next\DN@ @##1##{%
3085 %
3086 \xywarning@{Impossible @ (letter) should not be here!!}
3087 %
3088   \OBJECT@@{\dir##1}}%
3089  \else \let\next@=\OBJECT@direction
3090  \fi\fi\fi\fi\fi \next@}
```

```
3092 \xydef@\OBJECT@other{%
3093  \ifx !\next \DN@!{\OBJECT@shift}%
3094  \else\addPLUS@\ifx  \next \DN@{\OBJECT@change}%
3095  \else\addDASH@\ifx  \next \DN@{\OBJECT@change}%
3096  \else\addEQ@\ifx  \next \DN@{\OBJECT@set}%
3097  \else\ifx [\next %]
3098   \DN@[##1]{\xy@{[##1]}{\OBJECT@shape{##1}}}%
3099  \else\ifx ^\next \let\next@=\OBJECT@direction
3100  \else\ifx _\next \let\next@=\OBJECT@direction
3101  \else\ifx :\next \let\next@=\OBJECT@direction
3102  \else\ifx ?\next
3103   \DN@ ?{\xywarning@{\string? modifier used}\xyFN@\OBJECT@direction}%
3104  \else\ifx (\next %)
3105   \let\next@=\OBJECT@direction
3106  \else\addAT@\ifx\next \addAT@\DN@##1##{\OBJECT@@{\dir##1}}%
3107  \else \DN@##1##{\OBJECT@@{##1}}%
3108  \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}
```

`\OBJECT@@` is where we actually build the box by first l.3126 saving the previous edge in case it is needed, then l.3127 setting the defaults (including temporarily resetting `\xy@` to execute in case any `\xy@`s are used internally), l.3132 building the object box (which might change them) using `\objectbox` if no other command specified and setting the D, U, L, R dimensions ad required using the `\Leftness@` and `\Upness@` methods, and finally l.3136 setting up the R_p, U_p dimensions (as discussed above) and applying the ⟨modifier⟩s stored in `\toks@` and dumping the modified box.

```
3123 \xydef@\prevEdge@@{\zeroEdge}
```

```
3125 \xydef@\OBJECT@@#1#2{%
3126  \expandafter\def\expandafter\prevEdge@@\expandafter{\the\Edge@c}
3127  \expandafter\Edge@c\expandafter{\objectEdge}
3128  \Invisible@false\Hidden@false
3129  \def\Leftness@{.5}\def\Upness@{.5}%
3130  \gdef\preXY@style@{}\gdef\postXY@style@{}%
3131  \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@\relax}%
3132  \DN@{#1}\ifx\next@\empty \DNii@{#2}
3133   \ifx\nextii@\empty \DN@{\hbox\bgroup\no@}\else \let\next@=\objectbox \fi\fi
3134  \setbox\z@=\next@{#2}\L@c=\Leftness@\wdz@ \R@c=\wdz@ \advance\R@c-\L@c
3135  \D@c=\dp\z@ \advance\D@c\ht\z@ \U@c=\Upness@\D@c \advance\D@c-\U@c
3136  \R@p=\z@ \L@p=\L@c \U@p=\U@c \advance\U@p-\ht\z@ \D@p=-\U@p
```

```
 \the\toks@\toks@={}\setboxz@h{\kern\R@p \raise\U@p\boxz@}%
 \checkZeroEdge@
 \dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@ \ht\z@=\U@c \dp\z@=\D@c \boxz@
 \OBJECT@x}
```

```
\xydef@\adjustLR@{%
 \ifdim\zz@\wdz@ \L@c=\z@ \R@c=\z@ \dimen@=\Leftness@\p@
  \ifdim\dimen@<\z@ \L@c=\dimen@ \R@c=-\L@c
  \else\ifdim\dimen@>\p@ \L@c=\dimen@ \advance\L@c-\p@ \R@c=-\L@c \fi\fi
 \else \L@c=\Leftness@\wdz@ \R@c=\wdz@ \advance\R@c-\L@c \fi}
```

```
\xydef@\adjustUD@{\dimen@=\ht\z@ \advance\dimen@\dp\z@
 \ifdim\zz@\dimen@ \U@c=\z@ \D@c=\z@ \dimen@=\Upness@\p@
  \ifdim\dimen@<\z@ \U@c=\dimen@ \D@c=-\L@c
  \else\ifdim\dimen@>\p@ \U@c=\dimen@ \advance\U@c-\p@ \D@c=-\L@c \fi\fi
 \else \D@c=\dimen@ \U@c=\Upness@\dimen@ \advance\D@c-\U@c \fi}
```

```
\def\checkZeroEdge@{%
 \DN@{}%
 \ifdim\zz@\L@c \ifdim\zz@\R@c \ifdim\zz@\U@c \ifdim\zz@\D@c
   \DN@{\Edge@c={\zeroEdge}}\fi\fi\fi\fi
 \next@}
```

As an optimisation `\OBJECT@@` sets the edge type of all zero-sized objects to `\zeroEdge`.

`\OBJECT@x` cleans up the object by ensuring that it defines all the required methods: Essentially it terminates the box with the sequence "`}` `\def\Drop@@{`. . . `}` `\def\Connect@@{`. . . `}` `\D@c=`. . . `\U@c=`. . . `\L@c=`. . . `\R@c=`. . . `\Invisible@`. . . `\Hidden@`. . . `\def\Leftness@{`. . . `}` `\def\Upness@{`. . . `}`" where each . . . is set to the method defined within the object creation environment (started with `\hbox{` in `\OBJECT@@` or possibly elsewhere). We use rather heavy expansion hacking with `\toks@` to create the sequence so please look the other way. . . ☺

```
\xydef@\OBJECT@x{\toks@={\egroup\def\Drop@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\Drop@@}\def\Connect@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\Connect@@}}%
 \edef\tmp@{\D@c=\the\D@c \U@c=\the\U@c \L@c=\the\L@c \R@c=\the\R@c
  \Edge@c={\expandafter\noexpand\the\Edge@c}%
  \ifInvisible@\noexpand\Invisible@true\else\noexpand\Invisible@false\fi
  \ifHidden@\noexpand\Hidden@true\else\noexpand\Hidden@false\fi
  \def\noexpand\Leftness@{\Leftness@}\def\noexpand\Upness@{\Upness@}}%
 \expandafter\addtotoks@\expandafter{\tmp@}\the\toks@}
```

Methods: In addition to the "current object properties" for c (*cf.* 1.2.5) the following methods are set up by all objects:

`\Invisible@true` or `\Invisible@false`	whether object is invisible[1.4i]
`\Hidden@true` or `\Hidden@false`	whether object is hidden[1.4h]
`\def\Leftness@{`⟨factor⟩`}`	the desired $L/(L+R)$
`\def\Upness@{`⟨factor⟩`}`	the desired $U/(D+U)$
`\def\Drop@@{`. . . `}`	code that outputs the object, assuming `\boxz@` is of zero size and has the object displaced `<`X`,`Y`>` inside—usually just `\def`'d to `\styledboxz@`

`\def\Connect@@{...}`	code that builds a connection from p to c, assuming `\lastobjectbox@` contains the object

It is important to `\def` and not `\let` the last four methods since the TEXnique used in `\OBJECT@x` (and elsewhere) of passing them to surrounding scopes depends on it. The `\Connect@@` method should in turn setup several submethods as described in detail in 1.8.3.

Suitable defaults are set up by `\OBJECT@@` above which is why any box generating command can be used to construct objects as explained in note 1.4a below.

Here are the declarations:

```
\xynew@{if}\ifInvisible@
\xynew@{if}\ifHidden@

\xydef@\Leftness@{}
\xydef@\Upness@{}

\xydef@\Drop@@{\styledboxz@}
\xydef@\Connect@@{}
```

Notes

1.4a. An ⟨object⟩ is built using `\objectbox {`⟨text⟩`}`. `\objectbox` is initially defined as

```
\def\objectbox#1{%
 \hbox{$\objectstyle{#1}$}}
\let\objectstyle=\displaystyle
```

but may be redefined by options or the user. The ⟨text⟩ should thus be in the mode required by the `\objectbox` command—with the default `\objectbox` shown above it should be in math mode.

Actually it is

```
\xydef@\objectbox#1{\hbox{$\m@th\objectstyle{#1}$}}

\xylet@\objectstyle=\textstyle
```

1.4b. An ⟨object⟩ built from a TEX box with dimensions $w \times (h + d)$ will have $L_c = R_c = w/2$, $U_c = D_c = (h + d)/2$, thus initially be equipped with the adjustment `!C` (see note 1.4f). In particular: in order to get the reference point on the (center of) the base line of the original ⟨TEX box⟩ then you should use the ⟨modifier⟩ `!`; to get the reference point identical to the TEX reference point use the modifier `!!L`.

TEXnical remark: Any macro that expands to something that starts with a ⟨box⟩ may be used as a ⟨TEX box⟩ here.

This is done by the parsing above.

1.4c. Takes an object and constructs it, building a box; it is then processed according to the preceeding modifiers. This form makes it possible to use any ⟨object⟩ as a TEX box (even outside of XY-pictures) because a finished object is always also a box.

This macro is the main entry point to the ⟨object⟩ parser. It furthermore initialises the modifier list and the previous object edge.

```
%\xydef@\object{\hbox\bgroup\object@}
\xydef@\object{\hbox\bgroup\resetStyle@\object@}

\xydef@\object@{%
```

```
 \edef\next@{={\DirectionfromtheDirection@}}\expandafter\toks@\next@
% \resetStyle@\plainxy@ \xyFN@\OBJECT@}
 \plainxy@ \xyFN@\OBJECT@}
```

The initial value of `\toks@` ⟨modifier⟩ list is explained in note 1.4l below.

1.4d. Several ⟨object⟩s can be combined into a single object using the special command `\composite` with a list of the desired objects separated with `*`s as the argument. The resulting box (and object) is the least rectangle enclosing all the included objects.

First we collect all the objects smash on top of each other in box0 while we maintain the maximal extents in $(DULR)_p$. Then we reset box0 to contain the same but with the right spacing around.

```
\xydef@\composite#1#{\hbox\bgroup\composite@{#1}}

\xydef@\composite@#1#2{%
 \DN@{#1}\ifx\next@\empty\else\xywarning@{no variants of
        \string\composite\space allowed}\fi
 \global\setbox9=\hbox\bgroup
  \D@p=-\maxdimen \U@p=-\maxdimen \L@p=-\maxdimen \R@p=-\maxdimen
  \xyFN@\composite@i#2@}

\xydef@\composite@i{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\composite@i}%gobble spaces
 \else\ifx *\next \DN@ *{\xyFN@\composite@i}%
 \else\ifx @\next \DN@ @{\composite@x}%
  \xyerror@{<composite> object expected}{}\czeroEdge@
 \else \DN@{\composite@ii}\fi\fi\fi \next@}

\xydef@\composite@ii#1#{\composite@iii{#1}}

\xydef@\composite@iii#1#2{%
 \setbox\z@=\object#1{#2}%
 \ifInvisible@ \setboxz@h{}\else
  \setboxz@h{\kern-\L@c \boxz@}\ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ {\Drop@@}\fi
 \ifHidden@\else
  \ifdim\U@p<\U@c \U@p=\U@c \fi \ifdim\D@p<\D@c \D@p=\D@c \fi
  \ifdim\R@p<\R@c \R@p=\R@c \fi \ifdim\L@p<\L@c \L@p=\L@c \fi
 \fi
 \xyFN@\composite@iv}

\xydef@\composite@iv{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\composite@iv}%gobble spaces
 \else \ifx @\next \DN@ @{\composite@x}%
 \else \let\next@=\composite@i \fi\fi \next@}

\xydef@\composite@x{%
 \edef\tmp@{\egroup
  \D@c=\the\D@p \U@c=\the\U@p \L@c=\the\L@p \R@c=\the\R@p}\tmp@
 \setboxz@h{\kern\L@c\box9}\ht\z@=\U@c \dp\z@=\D@c
 \dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@
 \Edge@c={\rectangleEdge}\computeLeftUpness@ \boxz@
 \OBJECT@x}

\xydef@\computeLeftUpness@{%
 \dimen@=\L@c \advance\dimen@\R@c
```

```
\ifdim\zz@\dimen@ \def\Connect@@{\straight@{\dottedSpread@\jot}}%
 \ifdim\zz@\L@c\else
  \DN@{\zeroEdge}\expandafter\DNii@\expandafter{\the\Edge@c}%
  \ifx\next@\nextii@\Edge@c={\rectangleEdge}\fi\fi
\else \quotient@\Leftness@\L@c\dimen@ \fi
\dimen@=\U@c \advance\dimen@\D@c
\ifdim\zz@\dimen@ \def\Connect@@{\straight@{\dottedSpread@\jot}}%
 \ifdim\zz@\U@c\else
  \DN@{\zeroEdge}\expandafter\DNii@\expandafter{\the\Edge@c}%
  \ifx\next@\nextii@\Edge@c={\rectangleEdge}\fi\fi
\else \quotient@\Upness@\U@c\dimen@ \fi}
```

1.4e. Take an entire Xy-picture and wrap it up as a box as described in §1.2.1. Makes nesting of Xy-pictures possible: the inner picture will have its own zero point which will be its reference point *in* the outer picture when it is placed there.

This is simple exploiting the fact that `\endxy` actually sets up the extents of the 'object':

```
\xydef@\xybox#1{\xy#1\endxy \Edge@c={\rectangleEdge}\computeLeftUpness@}
```

1.4f. An object is *shifted* a ⟨vector⟩ by moving the point inside it which will be used as the reference point. This effectively pushes the object the same amount in the opposite direction.

Shifting uses the special value of R_p and U_p used while evalutaing the ⟨modifier⟩s. The fact that shifts like `!C` refer to the initial object's size means that we should parse the ⟨vector⟩ such that its actions happen at modification time... hence `\xytotoks@` is used to delay execution.

```
\xydef@\OBJECT@shift{%
 \let\xy@=\xytotoks@ \afterVECTORorEMPTY
  {\OBJECT@shift@}%
  {\addtotoks@{\X@c=-\L@c \advance\X@c\R@p \advance\X@c\L@p \Y@c=\U@p}%
   \OBJECT@shift@}}

\xydef@\OBJECT@shift@{%
 \addtotoks@{\advance\U@p-\Y@c
  \advance\L@c\X@c \advance\R@c-\X@c \advance\D@c\Y@c \advance\U@c-\Y@c
  \computeLeftUpness@}%
 \let\xy@=\oxy@ \xyFN@\OBJECT@}
```

Exercise 1.10: What is the difference between the ⟨pos⟩itions `0*{a}!DR` and `0*!DR{a}`?

1.4g. A ⟨size⟩ is a pair `<`W`,`H`>` of the width and height of a rectangle. When given as a ⟨vector⟩ these are just the vector coordinates, *i.e.*, the ⟨vector⟩ starts in the lower left corner and ends in the upper right corner. The possible ⟨add op⟩erations that can be performed are described in the following table.

⟨add op⟩	description
`+`	grow
`-`	shrink
`=`	set to
`+=`	grow to at least
`-=`	shrink to at most

In each case the ⟨vector⟩ may be omitted which invokes the "default size" for the particular ⟨add op⟩:

⟨add op⟩	default
+	+<2 × *objectmargin*>
-	-<2 × *objectmargin*>
=	=<*objectwidth* , *objectheight*>
+=	+=< $\max(L_c + R_c, D_c + U_c)$>
-=	-=< $\min(L_c + R_c, D_c + U_c)$>

The defaults for the first three are set with the commands

`\objectmargin` ⟨add op⟩ {⟨dimen⟩}
`\objectwidth` ⟨add op⟩ {⟨dimen⟩}
`\objectheight` ⟨add op⟩ {⟨dimen⟩}

---

where ⟨add op⟩ is interpreted in the same way as above.

```
\xylet@\objectmargin@=\jot
\xylet@\objectwidth@=\z@
\xylet@\objectheight@=\z@
```

```
\xydef@\objectmargin{\afterADDOP{\Addop@@\objectmargin@}}
\xydef@\objectwidth{\afterADDOP{\Addop@@\objectwidth@}}
\xydef@\objectheight{\afterADDOP{\Addop@@\objectheight@}}
```

The defaults for `+=`/`-=` are such that the resulting object will be the smallest containing/largest contained square.

Exercise 1.11: How are the objects typeset by the ⟨pos⟩itions "`*+UR{\sum}`" and "`*+DL{\sum}`" enlarged?

Bug: Currently changing the size of a circular object is buggy—it is changed as if it is a rectangle and then the change to the R parameter affects the circle. This should be fixed probably by a generalisation of the `o` shape to be ovals or ellipses with horizontal/vertical axes.

The three cases distinguished by the parsing above are handled similarly: they insert the parsed/default vector into X, Y in the modifications and then perform the operation at that time using the `\xytotoks@` trick described in note 1.4f:

```
\xydef@\OBJECT@change{%
 \afterADDOP{%
  \addEQ@\ifx \next
   \addtotoks@{\X@c=\D@c \advance\X@c\U@c \Y@c=\L@c \advance\Y@c\R@c}%
  \else
   \addtotoks@{\X@c=\objectmargin@ \advance\X@c\X@c \Y@c=\X@c}%
  \fi
  \let\xy@=\xytotoks@
  \afterVECTORorEMPTY\OBJECT@change@\OBJECT@change@}}
```

```
\xydef@\OBJECT@set{%
 \afterADDOP{%
  \let\xy@=\xytotoks@ \afterVECTORorEMPTY\OBJECT@change@
   {\addtotoks@{\X@c=\objectwidth@ \Y@c=\objectheight@}\OBJECT@change@}}}
```

The real work is done by the following command: a hack using expansion tricks to make use of the `\Addop@@` known now on the values in X, Y at modification time.

```
\xydef@\OBJECT@change@{%
 \addtotoks@{\advance\R@c\L@c \advance\R@p-\L@c \let\tmp@=\R@c}%
 \expandafter\addtotoks@\expandafter{\Addop@@\tmp@\X@c\R@c=\tmp@
  \L@c=\Leftness@\R@c \advance\R@p\L@c \advance\R@c-\L@c}%
 \addtotoks@{\advance\D@c\U@c \let\tmp@=\D@c}%
 \expandafter\addtotoks@\expandafter{\Addop@@\tmp@\Y@c\D@c=\tmp@
  \U@c=\Upness@\D@c \advance\D@c-\U@c}%
 \let\xy@=\oxy@ \xyFN@\OBJECT@}
```

It is clearly crucial that `\Addop@@` expands to its action immediately! Also note that enlarging changes the initial box offset in the horizontal direction only, *i.e.*, R_p.

Finally the code to interpret an ⟨add op⟩ used above: This simply parses it and creates a macro `\Addop@@` that takes a control sequence and a parameter ⟨dimen⟩ as arguments, and expands directly to commands that perform the ⟨add op⟩ of the ⟨dimen⟩ on the control sequence:

⟨add op⟩	effect of `\Addop@@` cs D
`+`	$cs \leftarrow cs + D$
`-`	$cs \leftarrow cs - D$
`=`	$cs \leftarrow D$
`+=`	$cs \leftarrow \begin{cases} D & \text{if } D \le cs \\ cs & \text{if } D > cs \end{cases}$
`-=`	$cs \leftarrow \begin{cases} D & \text{if } D \ge cs \\ cs & \text{if } D < cs \end{cases}$

Furthermore `\afterADDOP` leaves the `\next` token set to `=` in the last three cases only (this is used to determine the right default value in the size changes above).

The `\afterADDOP` macro is relatively simple because ⟨add op⟩s don't nest:

```
\xydef@\afterADDOP#1{\def\afterADDOP@{#1}\xyFN@\ADDOP@}

\xylet@\afterADDOP@=\empty

\xydef@\ADDOP@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\ADDOP@}%gobble spaces
 \else\addPLUS@\ifx \next \addPLUS@\DN@{\xyFN@\ADDOP@plus}%
 \else\addDASH@\ifx \next \addDASH@\DN@{\xyFN@\ADDOP@minus}%
 \else\addEQ@\ifx \next
  \addEQ@\DN@{\def\Addop@@{\Addop@0+=}\afterADDOP@}%
 \else
  \DN@{\def\Addop@@{\Addop@0+=}\afterADDOP@}%
 \fi\fi\fi\fi \next@}

\xydef@\ADDOP@plus{%
 \addEQ@\ifx \next
  \addEQ@\DN@{\def\Addop@@{\Addop@0+<}\afterADDOP@}%
 \else
  \DN@{\def\Addop@@{\Addop@1+=}\afterADDOP@}%
 \fi \next@}

\xydef@\ADDOP@minus{%
```

```
\addEQ@\ifx \next
 \addEQ@\DN@{\def\Addop@@{\Addop@0+>}\afterADDOP@}%
\else
 \DN@{\def\Addop@@{\Addop@1-=}\afterADDOP@}%
\fi \next@}
```

The work is done by the general `\Addop@` `{`f`}` `{`$\pm$`}` `{`$\bowtie$`}` `{`cs`}` D that defines

$$cs \equiv \begin{cases} f \times cs \pm D & \text{if } \neg((f \times cs \pm D) \bowtie D) \\ cs & \text{otherwise} \end{cases}$$

and also leaves the dimension in `\dimen@`.

```
\xydef@\Addop@#1#2#3#4#5{%
 \dimen@=#4\relax \edef#4{\the\dimen@}%
 \dimen@=#1\dimen@ \advance\dimen@#2#5\relax %\advance\dimen@ 1sp %HACK
 \ifdim\dimen@#3#4\else \edef#4{\the\dimen@}\fi
 \ifx\xy@\xyinitial@\else \DN@##1{\xy@@{\edef#4{##1}\checkZeroEdge@}}%
  \expandafter\next@\expandafter{#4}\fi}
```

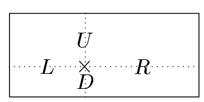
1.4h. A *hidden* object will be typeset but hidden from XY-pic in that it won't affect the size of the entire picture as discussed in §1.2.1.

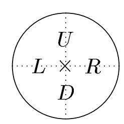
This is handled by the `\ifHidden@` conditional allocated with the methods.

1.4i. An *invisible* object will be treated completely normal except that it won't be typeset, *i.e.*, XY-pic will behave as if it was.

This is handled by the `\ifInvisible@` conditional allocated with the methods.

1.4j. Setting the *shape* of an object forces the shape of its edge to be as indicated. The kernel provides three shapes that change the edge, namely `[.]`, `[]`, and `[o]`, corresponding to the outlines

× , 

, and 

where the × denotes the point of the reference position in the object (the first is a point). Extensions can provide more shapes, however, all shapes set the extent dimensions L, R, D, and U.

The default shape for objects is `[]` and for plain coordinates it is `[.]`.

```
\xydef@\objectEdge{\rectangleEdge}
```

Furthermore the ⟨shape⟩s `[r]`, `[l]`, `[u]`, and `[d]`, are defined for convenience to adjust the object to the indicated side by setting the reference point such that the reference point is the same distance from the opposite of the indicated edge and the two neighbour edges but never closer to the indicated side than the opposite edge, *e.g.*, the object `[r]\hbox{Wide text}` has reference point at the × in Wide text but the object `[d]\hbox{Wide text}` has reference point at the × in Wide text. Finally, `[c]` puts the reference point at the center.

```
\xydefcsname@{shape [r]}{\advance\U@p\D@c \centerobject@
 \ifdim\L@c>\D@c
  \advance\R@c\L@c \L@c=.5\U@c \advance\L@c.5\D@c \advance\R@c-\L@c \fi
 \advance\U@p-\D@c}
```

```
\xydefcsname@{shape [l]}{\advance\U@p\D@c \centerobject@
 \ifdim\L@c>\D@c
  \advance\L@c\R@c \R@c=.5\U@c \advance\R@c.5\D@c \advance\L@c-\R@c \fi
 \advance\U@p-\D@c}
\xydefcsname@{shape [u]}{\advance\U@p\D@c \centerobject@
 \ifdim\D@c>\L@c
  \advance\U@c\D@c \D@c=.5\L@c \advance\D@c.5\R@c \advance\U@c-\D@c \fi
 \advance\U@p-\D@c}
\xydefcsname@{shape [d]}{\advance\U@p\D@c \centerobject@
 \ifdim\D@c>\L@c
  \advance\D@c\U@c \U@c=.5\L@c \advance\U@c.5\R@c \advance\D@c-\U@c \fi
 \advance\U@p-\D@c}
\xydefcsname@{shape [c]}{\advance\U@p\D@c \centerobject@ \advance\U@p-\D@c}
```

```
\xydef@\centerobject@{%
 \advance\D@c\U@c \D@c=.5\D@c \U@c=\D@c \advance\L@c\R@c \L@c=.5\L@c \R@c=\L@c}
```

Note: Extensions can add new ⟨shape⟩ object ⟨modifier⟩s which are then called ⟨style⟩s. These will always be either of the form `[`⟨keyword⟩`]` or `[`⟨character⟩ ⟨argument⟩`]`. Some of these ⟨style⟩s do other things than set the edge of the object.

A "simple shape" is just a control sequence `\shape [`⟨shape⟩`]` setting the appropriate edge. When such a `[`⟨shape⟩`]` modifier is encountered then we expand this control sequence onto the modifier queue, unless a control sequence `\style [`⟨shape⟩`]` exists: then that is expected to do it (and whatever else is required).

```
\xydef@\OBJECT@shape#1{\DN@{shape [#1]}%
 \expandafter\let\expandafter\nextii@\csname\codeof\next@\endcsname
 \ifx\nextii@\relax\DN@{style [#1]}%
  \expandafter\let\expandafter\nextii@\csname\codeof\next@\endcsname
  \ifx\nextii@\relax \DN@{\OBJECT@shapei[#1]}%
  \else\DN@{\nextii@\xyFN@\OBJECT@}\fi
 \else \expandafter\addtotoks@\expandafter{\nextii@}%
  \DN@{\xyFN@\OBJECT@}%
 \fi \next@}
```

```
\xydefcsname@{shape []}{\the\Edge@c5\relax}%
\xydefcsname@{shape [Outer]}{\the\Edge@c5\relax}%
\xydefcsname@{shape [Inner]}{\the\Edge@c4\relax}%
\xydefcsname@{shape [o]}{\Edge@c={\circleEdge}\the\Edge@c5\relax
 \Edge@c={\circleEdge}\def\prevEdge@@{\circleEdge}}
\xydefcsname@{shape [.]}{\czeroEdge@}
```

Add more simple shapes by defining more commands like these and proceed with coding the `\`...`Edge` command as described in §1.8.2.

Alternatively it is a "complex shape" of which none are defined in the kernel but some options like more variation... It is characterised by its first token and the rest of the contents of the `[]`s is the argument (remember: no `{}[]` characters!); it is searched for as control sequence `\[`⟨shape⟩`...]` overridden in the same way by `\*stylechar@`⟨shape⟩`@`.

```
\def\OBJECT@shapei[#1#2]{\DN@{shape [#1...]}%
 \expandafter\let\expandafter\next\csname\codeof\next@\endcsname
 \ifx\next\relax\DN@{*stylechar@#1@}%
```

```
  \expandafter\let\expandafter\next\csname\codeof\next@\endcsname
  \ifx\next\relax\DNii@{shape [#1#2]}%
   \xywarning@{illegal [<shape>] ignored: \codeof\nextii@\space not defined}%
   \DN@{\xyFN@\OBJECT@}%
  \else% Delete this: \expandafter\addtotoks@\expandafter{\next{#2}}%
   \DN@{\next{#2}\xyFN@\OBJECT@}%
  \fi
 \else
\expandafter\addtotoks@\expandafter{\next{#2}}\DN@{\next{#2}\xyFN@\OBJECT@}%
 \fi \next@}
```

Bug: The above is messy!

1.4k. While typesetting an object, some of the properties are considered part of the 'current object style'. Initially this means nothing but some of the ⟨style⟩s defined by extensions have this status, *e.g.*, colours `[red]`, `[blue]` say, using the `xycolor` extension, or varying the width of lines using `xyline`. Such styles are processed *left-to-right*; for example,

`*[red][green][=NEW][blue]{A}`

will typeset a blue `A` and define `[NEW]` to set the colour to green (all provided that `xycolor` has been loaded, of course).

The method of requesting a special effect is via the ⟨shape⟩ modifiers. However the code required for implementation is quite different. In particular it must allow for the different ways in which the ⟨driver⟩ may implement the `\special`s required to activate, and turn off, the effect. Another difficulty is that the ⟨style⟩ applies also to nested sub-parts of any ⟨object⟩. This means that the code cannot be wholly delayed until after the ⟨object⟩ has been constructed.

Usually some value has to be stored, so as to be accessible to sub-parts, though the `\special` commands cannot be fully constructed until the ⟨object⟩ is completely known and ready to be `\Drop@@` or `\Connect@@`'ed.

Finally, the overhead in using `\special` commands can be quite high with some `dvi`-drivers. Thus it is not best that every ⟨style⟩ request generate a `\special` command; indeed *two* `\special` commands, since each style-change also requires a command to revert to the previous style, after the change is no longer applicable.

Thus, as with ⟨shape⟩s, all the style-change requests are stored until the ⟨object⟩ is ready to be `\Drop@@`'ed. This also helps with getting the sequencing correct when later requests compound on, or override, earlier requests.

When the ⟨style⟩ requests are processed, after the ⟨object⟩ has been constructed, information of the form ⟨control-word⟩`{`⟨data⟩`}` is added to a global list called `\preXY@style@`. Similar information is added to another global list `\postXY@style@`, for the purpose of reverting the style parameters, within the ⟨driver⟩-file, to their previous values prior to the current ⟨object⟩.

Both lists are initially ⟨empty⟩ at the start of an ⟨object⟩. The contents of these lists are processed when the macros `\xypre@Style@@` and `\xypost@Style@@` are encountered in the expansion of `\styledboxz@`.

```
\xydef@\preXY@style@{}
\xydef@\postXY@style@{}

\xydef@\resetStyle@{\gdef\preXY@style@{}\gdef\postXY@style@{}}
```

The ⟨data⟩ contains the code needed to implement the requested style-change, with the particular ⟨driver⟩ being used.

The ⟨control-word⟩ is a macro whose expansion could be fixed, or could depend upon what other style-change requests are made for this ⟨object⟩. Its purpose is to decide whether or not the ⟨data⟩ is actually required. If so, it is added to the token-list `\styletoks@` for later use.

Whether the ⟨control-word⟩`{`⟨data⟩`}` tokens are prepended or appended to the global lists depends upon the nature of the style-change requested. For example, with `*[red][green]`⟨object⟩ the information `\`⟨set-color⟩`{`⟨green⟩`}` could be appended to `\preXY@style@`, with `\`⟨revert-to-color⟩`{`⟨red⟩`}` prepended to `\postXY@style@`. Then the dvi-output would set the color to ⟨red⟩ then ⟨green⟩ and restore it first to ⟨red⟩ then ⟨black⟩.

To Do: A more sophisticated implementation could instead prepend `\`⟨set-color⟩`{`⟨green⟩`}` to `\preXY@style@` and append `\`⟨revert-to-color⟩`{`⟨red⟩`}` to `\postXY@style@`. Now the first `\`⟨set-color⟩ encountered adds its ⟨data⟩ to `\styletoks@` then rebinds `\`⟨set-color⟩ to `\eat@`, to ignore subsequent occurrences. Similarly `\`⟨revert-to-color⟩ is rebound so that only the is restoration to ⟨black⟩ is retained.

This latter strategy is preferred when the overhead for a `\special` command is high, within the ⟨driver⟩'s output. Indeed the PostScript back-end is even more efficient. Since PostScript is itself a programming language, the ⟨data⟩ can be just the PostScript code. A single `\special` command is sufficient to accommodate *all* the style-changes for an individual ⟨object⟩. This is achieved by rebinding `\xydoprestyles@@` and `\xydopoststyles@@` to include the ⟨data⟩ within an appropriately prepared `\special` command.

```
\xynew@{toks}{\styletoks@}
\xydef@\addtostyletoks@#1{%
 \expandafter\styletoks@\expandafter{\the\styletoks@#1}}
```

The ⟨data⟩ to be used is collected in the token-list `\styletoks@`. If this is ⟨empty⟩ for an ⟨object⟩ then the style processing is skipped completely. For safety, in case multiple occurences of `\xypre@Style@` have somehow slipped into the token stream, these and `\xypost@Style@@` are bound to `\relax` until the ⟨object⟩ is finished. Indeed `\xypost@Style@@` is always bound to `\relax` unless `\styletoks@` is non-empty.

```
\xydef@\styledboxz@{%
%
%\W@{STYLED BOX:}%
%\W@{: PRE={\meaning\xypre@Style@@}}%
%
 \xypre@Style@@
%
%{\expandafter\DN@\expandafter{\the\styletoks@}\W@{: STYLTOKS={\codeof\next@}}}%
%
 \boxz@
%
%\W@{: POST={\meaning\xypost@Style@@}}%
%
 \xypost@Style@@}
\xydef@\xypre@Style@{\styletoks@={}\saveXyStyle@\preXY@style@
 \expandafter\DN@\expandafter{\the\styletoks@}%
 \ifx\next@\empty \DN@{\let\xypre@Style@@=\relax
  \let\xypost@Style@@=\relax \xypre@skipStyle@}%
```

```
 \else
  \let\xypre@Style@@=\relax \let\xypost@Style@@=\xypost@Style@
  \DN@{\expandafter\xydoprestyles@\expandafter{\the\styletoks@}}%
 \fi \next@ }
```

```
\xydef@\xypost@Style@{\styletoks@={}\postXY@style@
 \expandafter\xydopoststyles@\expandafter{\the\styletoks@}%
 \let\xypost@Style@@=\relax \let\xypre@Style@@=\xypre@Style@ }
```

```
\xydef@\xypre@skipStyle@#1\xypost@Style@@{#1\relax
 \let\xypre@Style@@=\xypre@Style@ }
```

```
\xydef@\xydoprestyles@@{\literal@}
\xydef@\xydopoststyles@@{\literal@}
\xylet@\xydoprestyles@=\xydoprestyles@@
\xylet@\xydopoststyles@=\xydopoststyles@@
```

Some extensions use the following macros to add style ⟨data⟩ to the global macros. Here code is added sequentially to `\preXY@style@` so that it is acted upon in the order of occurrence (FIFO) of the ⟨shape⟩ modifiers. Code is added to `\postXY@style@` in reverse order (LIFO), so that each addition to `\preXY@style@` can be closed off, if necessary, in correctly nested sequence.

```
\xydef@\modXYstyle@{%
 \ifx\xy@style@\empty\resetStyle@\fi \checkXyStyle@
 \expandafter\expandafter\expandafter\DN@
  \expandafter\expandafter\expandafter{\preXYstyle@@}%
 \ifx\next@\empty\else\DN@{\modXYstyle@@}\fi \next@ }
```

```
\xydef@\modXYstyle@@{%
 \DN@##1{\expandafter\def\expandafter\tmp@\expandafter{##1}}%
 \expandafter\next@\expandafter{\preStyle@@}%
 \DN@##1{\expandafter\gdef\expandafter\preXY@style@\expandafter{%
   \preXY@style@ ##1}}\expandafter\next@\expandafter{\tmp@}%
 \DN@##1{\expandafter\def\expandafter\tmp@\expandafter{##1}}%
  \expandafter\next@\expandafter{\postStyle@@}%
  \DN@##1{\expandafter\gdef\expandafter\postXY@style@\expandafter{%
   \tmp@ ##1}}\expandafter\next@\expandafter{\postXY@style@}}
```

```
\xydef@\xy@style@{}
\xydef@\checkXyStyle@{\ifx\xy@style@\empty\resetStyle@\fi
 \let\xy@style@=\relax}
```

The `\preStyle@` and `\postStyle@` temporarily hold style ⟨data⟩. The macro `\preXYstyle@@` is used to check whether there is any ⟨data⟩ to process. Normally this is just an alias for `\preStyle@@`, but some back-ends may use the two control-names differently.

```
\xydef@\preStyle@@{}
\xydef@\postStyle@@{}
\xydef@\preXYstyle@@{\relax}
```

```
\xydef@\loadxystyle@{%
 \let\xypre@Style@@=\xypre@Style@ \def\preXYstyle@@{\preStyle@@}}
```

```
\loadxystyle@
```

Initially styles are activated but can be suppressed using `\Unloadstyle@`.

```
\xydef@\Unloadstyle@{%
 \let\xypre@Style@@=\relax \let\xypost@Style@@=\relax
 \def\preXYstyle@@{\relax}}
```

Global macros are used, so that the same styles can be reused by successive objects without having to re-interpret ⟨shape⟩ modifiers, as described next.

Saving styles: Once specified for an ⟨object⟩, the collection of ⟨style⟩s can be assigned a name, using `[=`⟨word⟩`]`. Then `[`⟨word⟩`]` becomes a new ⟨style⟩, suitable for use with the same or other ⟨objects⟩s. Use a single ⟨word⟩ built from ordinary letters. If `[`⟨word⟩`]` already had meaning the new definition will still be imposed, but the following type of warning will be issued:

`Xy-pic Warning: Redefining style [`⟨word⟩`]`

The latter warning will appear if the definition occurs within an `\xymatrix`. This is perfectly normal, being a consequence of the way that the matrix code is handled. Similarly the message may appear several times if the style definition is made within an `\ar`.

The following illustrates how to avoid these messages by defining the style without typesetting anything.

```
\setbox0=\hbox{%
 \xy\drop[OrangeRed][=A]{}\endxy}
```

Note 1: The current colour is regarded as part of the style for this purpose.

Note 2: Such namings are global in scope. They are intended to allow a consistent style to be easily maintained between various pictures and diagrams within the same document.

```
\xydefcsname@{shape [=...]}#1{\checkXyStyle@ \addtotoks@{\xynamestyle@{#1}}}

\xydef@\xynamestyle@#1{\checkXyStyle@
 \expandafter\DNii@\expandafter{\csname shape [#1]\endcsname}%
 \expandafter\ifx\nextii@\relax %%\xywarning@{Defining new style [#1]}%
 \else\xywarning@{Redefining style [#1]}\fi
 \expandafter\xynamestyle@@\csname shape [#1]\endcsname }

\xydef@\xynamestyle@@#1{%
 \expandafter\def\expandafter\tmp@\expandafter{\preXY@style@}%
 \DN@##1{\def\tmp@{\checkXyStyle@\gdef\preXY@style@{##1}}}%
 \expandafter\next@\expandafter{\preXY@style@}%
 \DN@##1{\expandafter\gdef\expandafter#1\expandafter{\tmp@
  \gdef\postXY@style@{##1}}}%
 \expandafter\next@\expandafter{\postXY@style@}\DN@{}}
```

If the same ⟨style⟩ is intended for several ⟨object⟩s occurring in succession, the `[|*]` ⟨modifier⟩ can be used on the later ⟨object⟩s. This only works when `[|*]` precedes any other ⟨style⟩ modifiers; it is local in scope, recovering the last ⟨style⟩s used at the same level of TEX grouping.

```
\xydef@\prevXypreStyle@@{}
\xydef@\prevXypostStyle@@{}

\xydef@\saveXyStyle@{\expandafter
 \def\expandafter\prevXypreStyle@@\expandafter{\preXY@style@}%
 \expandafter\def\expandafter\prevXypostStyle@@\expandafter{%
  \postXY@style@}}
```

```
\xydef@\recoverXyStyle@{\expandafter
 \gdef\expandafter\preXY@style@\expandafter{\prevXypreStyle@@}%
 \expandafter\gdef\expandafter\postXY@style@\expandafter{%
  \prevXypostStyle@@}}
```

Defining new effects Allow new effects to be declared and a default action provided. The name is constructed from `#1` and the action to be taken is passed as `#2`. If `#3` is non-empty then overwrite any existing definition for a control sequence of the same name.

```
\xydef@\newxystyle#1#2#3{%
 \DN@{#3}\ifx\next@\empty
  \xydefcsname@{shape [#1]}{\csname xyshape@#1@\endcsname}%
 \else \expandafter\def\csname shape [#1]\endcsname{%
  \csname xyshape@#1@\endcsname}\fi
 \DN@{#2}\ifx\next@\empty
  \expandafter\def\csname xyshape@#1@\endcsname{%
   \xyundefinedStyle@{#1}{}@@}%
  \else \expandafter\def\csname xyshape@#1@\endcsname{#2}\fi}
```

```
\xydef@\xyundefinedStyle@#1#2@@{%
 \xywarning@{style #1 not defined, nothing to apply}}
```

Support for a new ⟨style⟩ takes the following form, in which `\thestyle@` expands to the current value of the ⟨style⟩ information to be maintained. If this is kept in a ⟨count⟩ or ⟨dimen⟩ register, named `\xystyle@` say, then replace `\thestyle@` by `\the\style@`. (However due to the large number of ⟨dimen⟩s already allocated, it is preferable to keep a text form of the value in `\thestyle@`.)

```
\xyshape@stylechange@{\modifystyle@{<how>}}
\xydef@\modifystyle@#1{\checkXyStyle@
 \edef\prevstyle@{\thestyle@}\modifystyle@@{#1}%
 \let\prestyle@=\prestyle@@ \let\poststyle@=\poststyle@@
 \applystyle@\prestyle@{\thestyle@}\poststyle@{\prevstyle@}}
\xydef@\resetstyle@@#1{\def\thestyle@{#1}\outputstyle@{#1}}
\xydef@\outputstyle@#1{\transformstyle@{#1}%
 \expandafter\stylespecial@\expandafter{\next@}}
\xylet@\transformstyle@@=\DN@
\xylet@\transformstyle@=\transformstyle@@
```

In the above the macros `\modifystyle@@` and `\stylespecial@` are adapted to the particular ⟨style⟩ information, see below. Typically ⟨how⟩ is actual TEX code and `\modifystyle@@` is simply `\literal@` to execute this code. Typesetting is done by `\stylespecial@#1` which is typically of the form `\special{`⟨key⟩ `#1}`.

The `\transformstyle@` is required in case a dvi-⟨driver⟩ needs information in a form that is different to how XY-pic maintains it in `\thestyle@`. A ⟨driver⟩ file should define `\`⟨driver⟩`@style@@` to perform the transformation and place its result in `\next@`. Install this macro using `\let\transformstyle@=\`⟨driver⟩`@style@@` within the ⟨driver⟩ installation macro.

Next we discuss `\applystyle@`; which uses `\prestyle@`, `\prestyle@@`, `\poststyle@` and `\poststyle@@`. There are two strategies here, affecting how much information is placed into the dvi-file, indicated by the alternative expansions given above. The appropriateness for a given style must be decided by the author of the style-option.

Suppose several modifications are made to a particular style parameter. We must build a list of commands `\prestyle@{`⟨value⟩`}`, to be executed later, which will recover the required value. Also we build a list of commands `\poststyle@{`⟨value⟩`}` to reset to the previous value.

In most cases it is only the result of the modifications that need be placed into the dvi-file. Since we do not know how many, if any, more modifications follow we could store each new result *after* the previous, so that its effect will override the previous value. This is building a FIFO list for the `\prestyle@` commands, with a FILO list for the `\poststyle@` commands:

```
\xydef@\applyFIFOstyle@#1#2#3#4{\bgroup
  \styletoks@={\egroup\gdef\preXY@style@}%
  \expandafter\toks@\expandafter{\preXY@style@}%
  \expandafter\addtotoks@\expandafter{\expandafter#1\expandafter{#2}}%
  \expandafter\addtostyletoks@\expandafter{\expandafter{\the\toks@}%
   \gdef\postXY@style@}%
  \expandafter\toks@\expandafter{\expandafter#3\expandafter{#4}}%
  \expandafter\addtotoks@\expandafter{\postXY@style@}%
  \expandafter\addtostyletoks@\expandafter{\expandafter{\the\toks@}}%
 \the\styletoks@ }
```

Use this via: `\applyLIFOstyle@\prestyle@{\thestyle@}\poststyle@{\prevstyle@}`.
With this method each modification places two entries into the dvi-file; there is no need for separate `\prestyle@` and `\prestyle@@` macros.

Perhaps a more efficient strategy is to build a LIFO list for the `\prestyle@` commands. This way the required value is encountered first. This first instance should then cancel all subsequent instances of `\prestyle@`. This is where the need comes for a `\prestyle@@`. Accompanying this is a FIFO list for the `\poststyle@` commands, with a corresponding `\poststyle@@` to kill subsequent instances:

```
\xydef@\applyLIFOstyle@#1#2#3#4{\bgroup
  \styletoks@={\egroup\gdef\preXY@style@}%
  \expandafter\toks@\expandafter{\expandafter#1\expandafter{#2}}%
  \expandafter\addtotoks@\expandafter{\preXY@style@}%
  \expandafter\addtostyletoks@\expandafter{\expandafter{\the\toks@}%
   \gdef\postXY@style@}%
  \expandafter\toks@\expandafter{\postXY@style@}%
  \expandafter\addtotoks@\expandafter{\expandafter#3\expandafter{#4}}%
  \expandafter\addtostyletoks@\expandafter{\expandafter{\the\toks@}}%
 \the\styletoks@ }
```

Use this via: `\applyLIFOstyle@\prestyle@{\thestyle@}\poststyle@{\prevstyle@}`.
The advantage here is that only two pieces of data need be output to the dvi-file, no matter how many modifications are requested.

1.4l. Setting the current direction is simply pretending for the typesetting of the object (and the following ⟨modifier⟩s) that some connection set it – the ⟨empty⟩ case just inherits the previous direction.

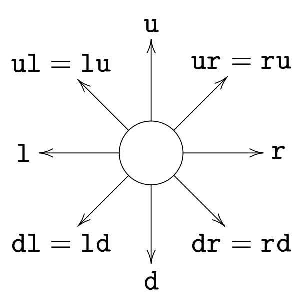
The code just calls the general ⟨direction⟩ parser below:

```
\xydef@\OBJECT@direction{\afterDIRECTIONorEMPTY{%
  \edef\next@{{\DirectionfromtheDirection@}}\expandafter\addtotoks@\next@
  \xyFN@\OBJECT@}%
 {\xyFN@\OBJECT@}}
```

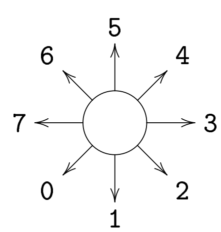
Here is the ⟨direction⟩ parser: first the parts parsing the ⟨diag⟩ part then the parts parsing the ⟨trailer⟩ part:

```
\xydef@\afterDIRECTIONorEMPTY#1#2{%
 \DN@##1{\def\afterDIRECTION@{\def\afterDIRECTION@{##1}%
  \ifDIRECTIONempty@\DN@{#2}\else\DN@{#1}\fi \next@}}%
 \expandafter\next@\expandafter{\afterDIRECTION@}%
 \xyFN@\DIRECTION@}

\xylet@\afterDIRECTION@=\empty
\xynew@{if}\ifDIRECTIONempty@

\xydef@\DIRECTION@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\DIRECTION@}%gobble spaces
 \else\ifx v\next \DN@ v{\DIRECTION@v}%
 \else\ifx \bgroup\next \let\next@=\DIRECTION@group
 \else\ifx (\next \DN@({\xyFN@\DIRECTION@open}%))
 \else
  \DN@{\count@=8 %
   \afterDIAG{\ifnum\count@=8 \DN@{\DIRECTIONempty@true \xyFN@\DIRECTION@i}%
    \else \DN@{\xy@@{\dimen@=\xydashl@}\Directionfromdiag@}\fi \next@}}%
 \fi\fi\fi\fi \next@}

\xydef@\DIRECTION@open{%
 \ifx *\next \DN@*##1*){\DIRECTION@group{##1}}%
 \else \DN@{\xyerror@{(* <pos> *) expected}{} \xyFN@\DIRECTION@i}%
 \fi \next@}
```

Diagonal directions. It is particularly easy to set ⟨diag⟩onal directions:



More to the point, ⟨diag⟩onals are stored internally as



Expanding `\afterDIAG{`⟨stuff⟩`}`⟨diag⟩ will result in `\count@` being set to the ⟨diag⟩ code (not changed in case the ⟨diag⟩ is ⟨empty⟩) before expanding ⟨stuff⟩.

```
\def\afterDIAG#1{\def\afterDIAG@{#1}\xyFN@\DIAG@}

\xydef@\DIAG@{%
```

```
\ifx d\next \DN@ d{\count@=1 \xyFN@\DIAG@@}%
\else\ifx r\next \DN@ r{\count@=3 \xyFN@\DIAG@@}%
\else\ifx u\next \DN@ u{\count@=5 \xyFN@\DIAG@@}%
\else\ifx l\next \DN@ l{\count@=7 \xyFN@\DIAG@@}%
\else \let\next@=\afterDIAG@
\fi\fi\fi\fi \next@}
```

```
\xydef@\DIAG@@{\ifcase\count@ \or
%\count@=1           3                   5                   7
 \DIAG@@@ l0r2\or\or \DIAG@@@ d2u4\or\or \DIAG@@@ r4l6\or\or \DIAG@@@ u6d0%
 \else\xybug@{impossible <diag> number}\fi
 \next@}
```

```
\xydef@\DIAG@@@#1#2#3#4{%
 \ifx #1\next \count@=#2\DN@#1{\afterDIAG@}%
 \else \ifx #3\next \count@=#4\DN@#3{\afterDIAG@}%
 \else \let\next@=\afterDIAG@ \fi\fi}
```

The action in case of a ⟨diag⟩ is simply to pick the right direction setup routine according to the encoding, getting the ⟨diag⟩ from `\count@` and the length of the d vector from `\dimen@`:

```
\xydef@\Directionfromdiag@{\ifcase\count@
     \xy@@{\dlDirection@\dimen@}%
 \or \xy@@{\dDirection@\dimen@}%
 \or \xy@@{\drDirection@\dimen@}%
 \or \xy@@{\rDirection@\dimen@}%
 \or \xy@@{\urDirection@\dimen@}%
 \or \xy@@{\uDirection@\dimen@}%
 \or \xy@@{\ulDirection@\dimen@}%
 \or \xy@@{\lDirection@\dimen@}%
 \or % 8 is legal and means change nothing
 \else\xybug@{impossible <diag>}\fi
 \DIRECTIONempty@false \xyFN@\DIRECTION@i}
```

Vector directions. Alternatively `v`⟨vector⟩ sets the direction as if the connection from `0` to the ⟨vector⟩ had been typeset except that the *origin* is assumed zero such that directions `v(`x`,`y`)` mean the natural thing, *i.e.*, is the direction of the connection from `(0,0)` to `(`x`,`y`)`.

The action for a `v` reads a ⟨vector⟩ and sets the direction accordingly using some expansion hackery to propagate it out. The *origin* is cleared locally to make `v(`x`,`y`)` behave as it should.

```
\xydef@\DIRECTION@v{%
 \xy@{v}{\enter@{\cfromthec@ \X@origin=\the\X@origin \Y@origin=\the\Y@origin
   \X@p=\the\X@p \Y@p=\the\Y@p}%
  \X@origin=\z@ \Y@origin=\z@}%
 \afterVECTORorEMPTY
  {\xy@@{\X@p=\z@ \Y@p=\z@ \setupDirection@ \leave@}%
   \DIRECTIONempty@false \xyFN@\DIRECTION@i}%
  {\xy@@\leave@ \xyerror@{<vector> expected after v}{}%
   \DIRECTIONempty@false \xyFN@\DIRECTION@i}}
```

Complex direction. In case the direction is not as simple, you can construct `{` ⟨pos⟩ ⟨decor⟩ `}` that sets up p and c such that $\overline{pc}$ has the desired direction. **Note:** that you must use the `(*`...`*)` form if this is to appear in an object ⟨modifier⟩!

The code for this is rather like the one for ⟨vector⟩s.

```
\xydef@\DIRECTION@group#1{%
 \xy@@{\begingroup}\xy@@ix@{#1}\xy@@{\plainxy@\expandafter\POS\the\toks9\relax
  \setupDirection@\edef\next@{\endgroup \DirectionfromtheDirection@}\next@}%
 \DIRECTIONempty@false \xyFN@\DIRECTION@i}
```

Exercise 1.12: What effect is achieved by using ⟨modifier⟩s `v/1pc/` and `v/-1pc/`?

1.4m. Once the initial direction is established as either the last one or an absolute one then the remainder of the ⟨direction⟩ is interpreted.

Adding a single `^` or `_` denotes the result of rotating the default direction a right angle in the positive and negative direction, *i.e.*, anti-/clockwise, respectively. **Note:** Do *not* use `^^` but only `__` to reverse the direction!

A trailing `:`⟨vector⟩ is like `v`⟨vector⟩ but uses the ⟨direction⟩ to set up a standard square base such that `:(0,1)` and `:(0,-1)` mean the same as `:a(90)` and `:a(-90)` and as `^` and `_`, respectively.

```
\xydef@\DIRECTION@i{%
 \ifx ^\next \DN@ ^{\xy@^{\aboveDirection@\xydashl@}%
   \DIRECTIONempty@false \xyFN@\DIRECTION@i}%
 \else\ifx _\next \DN@ _{\xy@_{\belowDirection@\xydashl@}%
   \DIRECTIONempty@false \xyFN@\DIRECTION@i}%
 \else\ifx :\next \DN@ :{%
  \xy@{:}{\enter@{\cfromthec@ \basefromthebase@ \X@p=\the\X@p \Y@p=\the\Y@p}%
   \X@origin=\z@ \Y@origin=\z@
   \X@xbase=\cosDirection\xydashl@ \Y@xbase=\sinDirection\xydashl@
   \X@ybase=-\Y@xbase \Y@ybase=\X@xbase}%
  \afterVECTORorEMPTY
   {\xy@@{\X@p=\z@ \Y@p=\z@ \setupDirection@ \leave@}%
    \DIRECTIONempty@false \xyFN@\DIRECTION@i}%
   {\xy@@\leave@ \xyerror@{<vector> expected after :}{}%
    \DIRECTIONempty@false \xyFN@\DIRECTION@i}}%
 \else
  \let\next@=\afterDIRECTION@
 \fi\fi\fi \next@}
```

Exercise 1.13: What effect is achieved by using ⟨modifier⟩s `v/1pc/:(1,0)` and `v/-1pc/__`?

1.5 Decorations

```
\message{decorations;}
```

⟨Decor⟩ations are actual TeX macros that decorate the current picture in manners that depend on the state. They are allowed *after* the ⟨pos⟩ition either of the outer `\xy`...`\endxy` or inside `{`...`}`. The possibilities are given in figure 1.6 with notes below.

<table>
<tr><th colspan="3">Syntax</th><th>Action</th></tr>
<tr><td>⟨decor⟩</td><td>⟶</td><td>⟨command⟩ ⟨decor⟩</td><td>either there is a command…</td></tr>
<tr><td></td><td>|</td><td>⟨empty⟩</td><td>…or there isn't.</td></tr>
<tr><td>⟨command⟩</td><td>⟶</td><td>\save ⟨pos⟩</td><td>save state[1.5a], then do ⟨pos⟩</td></tr>
<tr><td></td><td>|</td><td>\restore</td><td>restore state[1.5a] saved by matcing \save</td></tr>
<tr><td></td><td>|</td><td>\POS ⟨pos⟩</td><td>interpret ⟨pos⟩</td></tr>
<tr><td></td><td>|</td><td>\afterPOS { ⟨decor⟩ } ⟨pos⟩</td><td>interpret ⟨pos⟩ and then perform ⟨decor⟩</td></tr>
<tr><td></td><td>|</td><td>\drop ⟨object⟩</td><td>drop ⟨object⟩ as the ⟨pos⟩ * operation</td></tr>
<tr><td></td><td>|</td><td>\connect ⟨object⟩</td><td>connect with ⟨object⟩ as the ⟨pos⟩ ** operation</td></tr>
<tr><td></td><td>|</td><td>\relax</td><td>do nothing</td></tr>
<tr><td></td><td>|</td><td>⟨TeX commands⟩</td><td>any TeX commands[1.5b] and user-defined macros that neither generates output (watch out for stray spaces!), nor changes the grouping, may be used</td></tr>
<tr><td></td><td>|</td><td>\xyverbose | \xytracing | \xyquiet</td><td>tracing[1.5c] commands</td></tr>
<tr><td></td><td>|</td><td>\xyignore {⟨pos⟩ ⟨decor⟩}</td><td>ignore[1.5d] XY-code</td></tr>
<tr><td></td><td>|</td><td>\xycompile {⟨pos⟩ ⟨decor⟩}</td><td>compile[1.5e] to file ⟨prefix⟩⟨no⟩.xyc</td></tr>
<tr><td></td><td>|</td><td>\xycompileto{⟨name⟩}{⟨pos⟩⟨decor⟩}</td><td>compile[1.5e] to file ⟨name⟩.xyc</td></tr>
</table>

Figure 1.6: ⟨decor⟩ations.

Most options add to the available ⟨decor⟩, in particular the `v2` option loads many more since XY-pic versions prior to 2.7 provided most features as ⟨decor⟩.

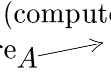
Simple decorations: `\POS` and `\afterPOS` have already been defined; the following are just simple applications of previously defined commands:

```
\xydef@\drop#1#{\DN@##1{\xy@@ix@{{#1}{##1}}%
 \xy@{\drop#1{##1}}{\expandafter\drop@\the\toks9 }\ignorespaces}\next@}

\xydef@\connect#1#{\DN@##1{\xy@@ix@{{#1}{##1}}%
 \xy@{\connect#1{##1}}{\expandafter\connect@\the\toks9 }\ignorespaces}\next@}

\xydef@\preconnect#1#{\DN@##1{\xy@@ix@{{#1}{##1}}%
 \xy@{\connect#1{##1}}{\expandafter\preconnect@\the\toks9 }\ignorespaces}\next@}
```

Notes

1.5a. Saving and restoring allows 'excursions' where lots of things are added to the picture without affecting the resulting XY-pic state, *i.e.*, c, p, and *base*, and without requiring matching `{}`s. The independence of `{}` is particularly useful in conjunction with the `\afterPOS` command, for example, the definition

```
\def\ToPOS{\save\afterPOS{%
  \POS**{}?>*@2{>}**@{-}\restore};p,}
```

will cause the code `\ToPOS`⟨pos⟩ to construct a double-shafted arrow from the current object to the ⟨pos⟩ (computed relative to it) such that `\xy *{A} \ToPOS +<10mm,2mm>\endxy` will typeset the picture .

Note: Saving this way in fact uses the same state as the `{}` 'grouping', so the code p_1 `,` `{`p_2`\save}`, ... `{\restore}` will have $c = p_1$ both at the ... and at the end!

```
\xydef@\save{\relax\saveC}
\xydef@\saveC{\xy@\save\save@ \POS}
```

```
\xydef@\save@{\enter@{\cfromthec@ \pfromthep@ \basefromthebase@}}
```

```
\xydef@\restore{\xy@\restore\leave@ \ignorespaces}
```

1.5b. One very tempting kind of TEX commands to perform as ⟨decor⟩ is arithmetic operations on the XY-pic state. This will work in simple XY-pictures as described here but be warned: *it is not portable* because all XY-pic execution is indirect, and this is used by several options in nontrivial ways. Check the TEX-nical documentation [15] for details about this!

Macros that expand to ⟨decor⟩ will always do the same, though.

TEX hackers like the author may enjoy changing the XY-pic state directly using ⟨decor⟩ of the form `\xy@{`⟨id⟩`}{`⟨code⟩`}`...

1.5c. `\xyecho` will turn on echoing of all interpreted XY-pic ⟨pos⟩ characters. **Bug:** Not completely implemented yet. `\xyverbose` will switch on a tracing of all XY-pic commands executed, with line numbers. `\xytracing` traces even more: the entire XY-pic state is printed after each modification. `\xyquiet` restores default quiet operation.

The trick is to replace the `\xy@` command such that it calls the 'normal' one between writing out a trace message and the state.

```
\xydef@\xyecho{%
 \xy@\xyecho{\let\xy@=\xyecho@ \message{\string\xyecho}}\POS}
```

```
\xydef@\xyecho@#1#2{{\def\1{#1}\ifx\1\empty\else\message{\codeof\1}\fi}%
 \oxy@{#1}{#2}}
```

```
\xydef@\xyverbose{%
 \xy@\xyverbose{\let\xy@=\xyverbose@
 \W@{Xy: \string\xyverbose\xytracelineno@}}\POS}
```

```
\xydef@\xyverbose@#1#2{%
 {\def\1{#1}\ifx\1\empty\else\W@{Xy: \codeof\1\xytracelineno@}\fi}%
 \oxy@{#1}{#2}}
```

```
\xydef@\xytracing{%
 \xy@\xytracing{\let\xy@=\xytracing@
 \W@{Xy TRACE: \string\xytracing\xytracelineno@}\xystatus@:}\POS}
```

```
\xydef@\xytracing@#1#2{{\def\1{#1}\def\2{#2}%
 \W@{Xy TRACE: \codeof\1 {\codeof\2}\xytracelineno@}}\oxy@{#1}{#2}\xystatus@:}
```

```
\xydef@\xystatus@#1{\xystatus@cp{#1}%
 \W@{#1 d=<\the\d@X,\the\d@Y>
        Direction=\the\Direction=\string(\cosDirection,\sinDirection\string)}%
 \W@{#1 base=<\the\X@origin,\the\Y@origin>+%
        <\the\X@xbase,\the\Y@xbase>x+<\the\X@ybase,\the\Y@ybase>y}%
 \W@{#1 min/max=<\the\X@min,\the\Y@min>/<\the\X@max,\the\Y@max>
        S=\the\csp@}}
```

```
\xydef@\xystatus@cp#1{%
 \W@{#1 c=<\the\X@c,\the\Y@c> \expandafter\string\the\Edge@c
```

```
        \string[\the\L@c+\the\R@c,\the\D@c+\the\U@c\string]
        \ifInvisible@ I\fi\ifHidden@ H\fi}%
 \W@{#1 p=<\the\X@p,\the\Y@p> \expandafter\string\the\Edge@p
        \string[\the\L@p+\the\R@p,\the\D@p+\the\U@p\string]}}
```

```
\xydef@\xystatus@c#1{%
 \W@{#1<\the\X@c,\the\Y@c> \expandafter\string\the\Edge@c
        \string[\the\L@c+\the\R@c,\the\D@c+\the\U@c\string]}}
```

```
\xydef@\xyquiet{\xy@\xyquiet{\let\xy@=\oxy@}}
```

1.5d. Ignoring means that the ⟨pos⟩ ⟨decor⟩ is still parsed the usual way but nothing is typeset and the Xy-pic state is not changed.

We ignore in a group to ensure that nothing done inside 'leaks' to the outside.

```
\xydef@\xyignore#1{\xy@\xyignore{\xyignore@{#1}}\ignorespaces}
```

```
\xydef@\xyignore@#1{{\let\xy@=\xyeat@ \change@oxy@\xy@ \POS#1\relax}}
```

```
\xydef@\xyeat@#1#2{}
```

1.5e. It is possible to save an intermediate form of commands that generate parts of an Xy-picture to a file such that subsequent typesetting of those parts is significantly faster: this is called *compiling*. The produced file contains code to check that the compiled code still corresponds to the same ⟨pos⟩⟨decor⟩ as well as efficient Xy-code to redo it; if the ⟨pos⟩⟨decor⟩ has changed then the compilation is redone.

There are two ways to use this. The direct is to invent a ⟨name⟩ for each diagram and then embrace it in `\xycompileto{`⟨name⟩`}|{`...`}` – this dumps the compiled code into the file ⟨name⟩`.xyc`.

When many diagrams are compiled then it is easier to add `\xycompile{`...`}` around the ⟨pos⟩⟨decor⟩ to be compiled. This will assign file names numbered consecutively with a ⟨prefix⟩ which is initially the expansion of `\jobname-` but may be set with

---

`\CompilePrefix{`⟨prefix⟩`}`

This has the disadvantage, however, that if additional compiled Xy-pictures are inserted then all subsequent pictures will have to be recompiled. One particular situation is provided, though: when used within constructions that typeset their contents more than once (such as most AMS-LaTeX equation constructs) then the declaration

---

`\CompileFixPoint{`⟨id⟩`}`

---

can be used inside the environment to fix the counter to have the same value at every passage.

It is done by just writing all `\xy@`-commands to the file. The file establishes the correct input mode through use of the appropriate commands itself.

```
\xydef@\xycompile@@{\jobname-}
\xydef@\xycompileno@@{0}
```

```
\xydef@\CompilePrefix#1{%
 \def\xycompile@@{#1}\xdef\xycompile@@{\codeof\xycompile@@}%
 \xdef\xycompileno@@{0}}
```

```
\xydef@\xycompile{%
 \count@=\xycompileno@@ \advance\count@\@ne
```

```
 \xdef\xycompileno@@{\ifnum10>\count@ 0\fi \the\count@}%
 \edef\next{\noexpand\xycompileto{\xycompile@@\xycompileno@@}}\next}
```

```
\xylet@\compilename@@=\empty
\xylet@\xyrecompile@@=\relax
```

```
\xydef@\xycompileto#1#2{%
 \if\inxy@ \DN@{\xy@@{\nter@{}}}%
 \else \DN@{\xy \xy@@{\nter@{\endxy}}}\fi \next@
 \ifxysaving@ \xyerror@{Compilations can not be nested}{}\fi
 \DN@{#1}\edef\compilename@@{\codeof\next@}\DNii@{#2}%
 \def\xyrecompile@@{recompiling TRUNCATED}%
 \expandafter\xyinputorelse@@\expandafter{\compilename@@.xyc}%
 {\def\xyrecompile@@{compiling to}}%
 \ifx\xyrecompile@@\relax \else \expandafter\xyrecompile@ \fi
 \xy@@\leave@ \ignorespaces}
```

Recompilation is done by just writing all `\xy@`-commands to the file. The file establishes the correct input mode and terminates itself; after it has been finished it is simply reread to actually get the drawing done in the document.

```
\xydef@\xyrecompile@{%
 \message{(\xyrecompile@@\space\string`\compilename@@.xyc\string'}%
 \DN@{\immediate\openout\xywrite@=}\expandafter\next@\compilename@@.xyc
 \immediate\write\xywrite@{%
  \string\xycompiled{\compilename@@}%
   {\the\year/\the\month/\the\day\string:\the\time\xytracelineno@}%
   {Xy-pic \xyversion}\xycomment@}%
 \immediate\write\xywrite@{{\codeof\nextii@}\relax}%
 {\xysaving@ \expandafter\POS\nextii@ \relax}%
 \immediate\write\xywrite@{\string\xyendcompiled}%
 \immediate\closeout\xywrite@ \message{done)}%
 \expandafter\input\compilename@@.xyc
 \outlines@@\writeoutline@\writeoutline@}
```

```
\xydef@\xysaving@{\let\xy@=\xysave@ \change@oxy@\xy@
 \let\xy@@ix@=\xysave@@toksix@ \xysaving@true}
```

```
\xynew@{if}\ifxysaving@ \xysaving@false
```

```
\xydef@\xysave@#1#2{{\DN@{{#1}{#2}}%
  \immediate\write\xywrite@{\string\xy@\codeof\next@\xycomment@}}}
```

```
\xydef@\xysave@@toksix@#1{{\DN@{{#1}}%
  \immediate\write\xywrite@{\string\xy@@ix@\codeof\next@\relax}}}
```

```
\xywarnifdefined\xycomment@
{\catcode`\%=12 \catcode`\(=1 \catcode`\)=2 \gdef\xycomment@(%)}
```

Hack1: The `\ifxysaving@` can never be locally switched off! Anyway it is used to allow a gross hack avoiding building a queue in the matrix option that will generate too long lines!!

Hack2: `\xysave@@toksix@` is not doing the catcode jive because it can never be invoked while loading a file (knock, knock ☺).

The initial command in all `.xyc` files check that this is the right file and that neither the version of Xy-pic nor the user's code has changed:

```
\xydef@\xycompiled#1#2#3#4{\DN@{#1}\edef\next@{\codeof\next@}%
 \ifx\next@\compilename@@\else
  \xywarning@{This file does not contain the result of
        \string\xycompileto{\compilename@@}{...}^^J%
        but of \string\xycompileto{\next@}}\fi
 \edef\next{Xy-pic \xyversion}\DN@{#3}\ifx\next\next@
  \DN@{#4}\ifx\next@\nextii@ \xycatcodes \outlines@@\relax\readoutline@
  \else \def\xyrecompile@@{recompiling to}\endinput \fi
 \else \def\xyrecompile@@{Xy-pic version change - recompiling}\endinput \fi}

\xydef@\xyendcompiled{\let\xyrecompile@@=\relax \xyuncatcodes\endinput}
```

Finally the fix-point thing.

```
\xydef@\CompileFixPoint#1{%
 \expandafter\ifx\csname#1FIX@@\endcsname\relax
  \expandafter\xdef\csname#1FIX@@\endcsname{\xycompileno@@}%
 \else
  \xdef\xycompileno@@{\csname#1FIX@@\endcsname}\fi}
```

Finally, when many 'administrative typesetting runs' are needed, *e.g.*, readjusting LaTeX cross references and such, then it may be an advantage to not typeset any XY-pictures at all during the intermediate runs. This is supported by the following declarations which for each compilation creates a special file with the extension `.xyd` containing just the size of the picture:

```
\MakeOutlines
\OnlyOutlines
\ShowOutlines
\NoOutlines
```

The first does no more. The second uses the file to typesets a dotted frame of the appropriate size instead of the picture (unless the picture has changed and is recompiled, then it is typeset as always and the `.xyd` file is recreated for subsequent runs). The third shows the outlines as dotted rectangles. The last switches outline processing completely off.

The implementation is all handled by `\outlines@@` which is called at strategic places in the compilation macros above. It just expands to nothing when the state is 'No', the first argument when it is 'Make', and the second when it is 'Only'.

```

\xydef@\outlines@no#1#2{}
\xydef@\outlines@make#1#2{#1}
\xydef@\outlines@only#1#2{#2}

\xylet@\outlines@@=\outlines@no
\xylet@\writeoutline@@=\relax
\xylet@\readoutline@@=\relax

\xydef@\MakeOutlines{\let\outlines@@=\outlines@make}

\xydef@\OnlyOutlines{\xywarning@{Compiled pictures only shown as outlines}%
 \let\outlines@@=\outlines@only}

\xydef@\ShowOutlines{\let\readoutline@@=\typesetoutline@}

\xydef@\NoOutlines{\let\outlines@@=\outlines@no}
```

```
\xydef@\writeoutline@{%
 \DN@{\immediate\openout\xywrite@=}\expandafter\next@\compilename@@.xyd
 \immediate\write\xywrite@{\xycomment@\space\compilename@@.xyd}
 \immediate\write\xywrite@{%
  \string\X@min=\the\X@min \string\X@max=\the\X@max
  \string\Y@min=\the\Y@min \string\Y@max=\the\Y@max \string\relax}%
 \writeoutline@@
 \immediate\closeout\xywrite@}
```

```
\xynew@{if}\ifoutlineread@
```

```
\xydef@\readoutline@{\outlineread@true
 \expandafter\xyinputorelse@@\expandafter{\compilename@@.xyd}%
  {\def\xyrecompile@@{outline missing - recompiling to}\outlineread@false}%
 \ifoutlineread@ \readoutline@@
  \writeoutline@\expandafter\xyendcompiled\fi}
```

```
\xydef@\typesetoutline@{%
 \xy@@{\czeroEdge@ \U@p=\z@ \D@p=\z@ \L@p=\z@ \R@p=\z@ \Edge@p={\zeroEdge}%
  \X@p=\X@min \Y@p=\Y@min \X@c=\X@min \Y@c=\Y@max \connect@\dir{.}%
  \X@p=\X@c \Y@p=\Y@c \X@c=\X@max \Y@c=\Y@max \connect@\dir{.}%
  \X@p=\X@c \Y@p=\Y@c \X@c=\X@max \Y@c=\Y@min \connect@\dir{.}%
  \X@p=\X@c \Y@p=\Y@c \X@c=\X@min \Y@c=\Y@min \connect@\dir{.}}}
```

1.6 Kernel object library

```
\message{kernel objects:}
```

In this section we present the *library objects* provided with the kernel language—several options add more library objects. They fall into three types: Most of the kernel objects (including all those usually used with `**` to build connections) are *directionals*, described in §1.6.1. The remaining kernel library objects are *circles* of §1.6.2 and *text* of §1.6.3.

1.6.1 Directionals

```
\message{directionals,}
```

The kernel provides a selection of *directionals*: objects that depend on the current direction. They all take the form

---

`\dir`⟨dir⟩

---

to typeset a particular ⟨dir⟩ectional object. All have the structure

---

⟨dir⟩ ⟶ ⟨variant⟩`{`⟨main⟩`}`

---

with ⟨variant⟩ being ⟨empty⟩ or one of the characters `^_23` and ⟨main⟩ some mnemonic code.

We will classify the directionals primarily intended for building connections as *connectors* and those primarily intended for placement at connection ends or as markers as *tips*.

Figure 1.7 shows all the ⟨dir⟩ectionals defined by the kernel with notes below; each ⟨main⟩ type has a line showing the available ⟨variant⟩s. Notice that only some variants exist for each ⟨dir⟩—when a nonexisting variant of a ⟨dir⟩ is requested then the ⟨empty⟩ variant is used silently. Each is shown

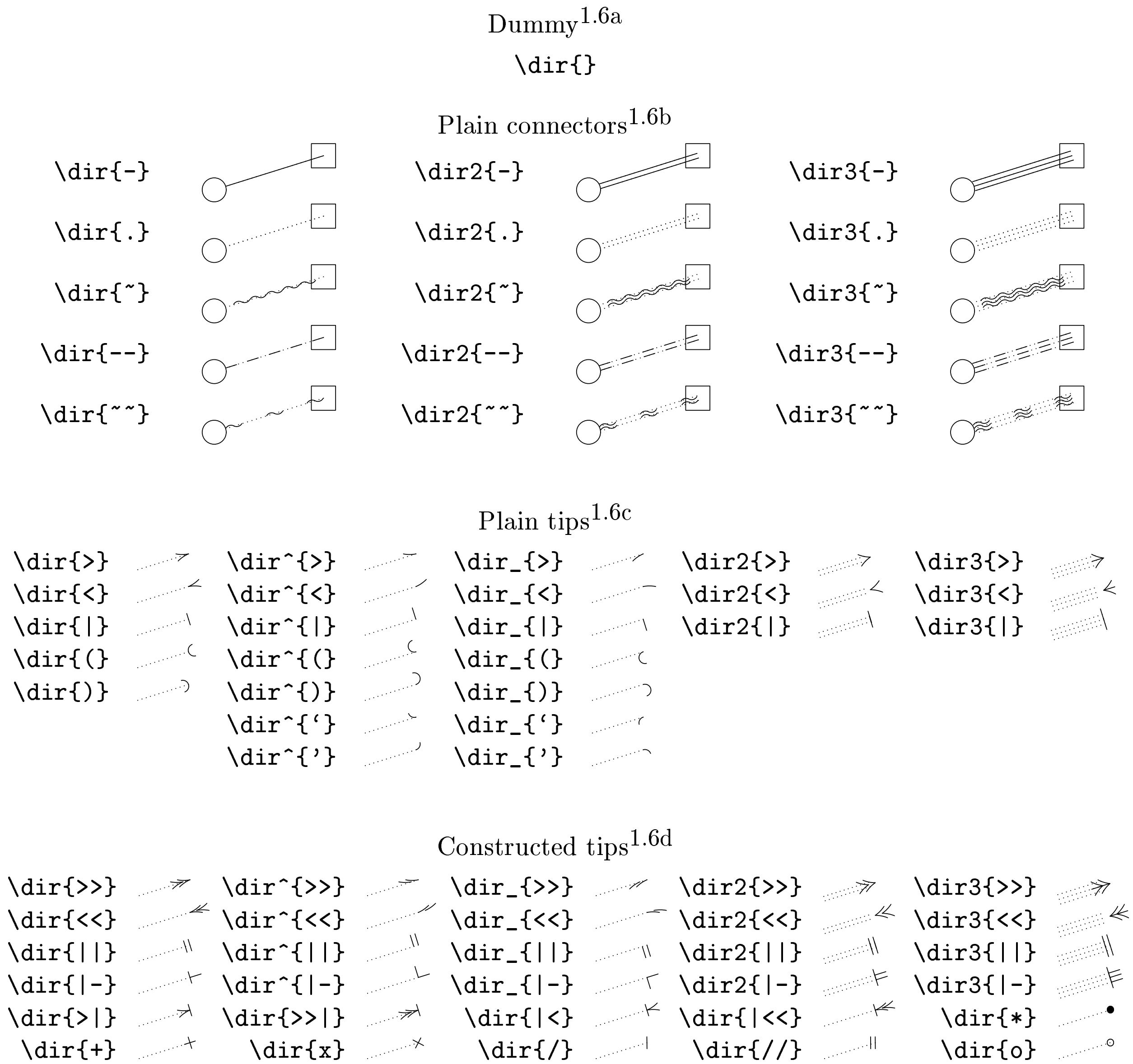


Figure 1.7: Kernel library ⟨dir⟩ectionals

in either of the two forms available in each direction as applicable: connecting a ◯ to a □ (typeset by `**\dir`⟨dir⟩) and as a tip at the end of a dotted connection of the same variant (*i.e.*, typeset by the ⟨pos⟩ `**\dir`⟨variant⟩`{.} ?> *\dir`⟨dir⟩).

As a special case an entire ⟨object⟩ is allowed as a ⟨dir⟩ by starting it with a `*`: `\dir*` is equivalent to `\object`.

Setup: `\dir` starts an ⟨object⟩ and passes control to a 'finisher' named `\dir`⟨variant⟩`{`⟨main⟩`}` otherwise to the one corresponding to an ⟨empty⟩ ⟨variant⟩. The kernel ones described here have in common that they make use of the generic `\straight@` defined in §1.8.3.

```
\xydef@\dir{\hbox\bgroup\xyFN@\dir@i}

\xydef@\dir@i{\ifx *\next \DN@*{\resetStyle@\object@}%
  \else \let\next@=\dir@ii \fi \next@}

\xydef@\dir@ii#1#{\dir@{#1}}

\xydef@\dir@#1#2{\DN@{dir#1{#2}}%
 \expandafter\let\expandafter\next\csname\codeof\next@\endcsname
 \ifx\next\relax \DN@{dir{#2}}%
  \expandafter\let\expandafter\next\csname\codeof\next@\endcsname
  \ifx\next\relax \DN@{\dir#1{#2}}%
   \xyerror@{illegal <dir>: \codeof\next@\space not defined}{}%
   \let\next=\no@ \fi\fi \next}
```

Notes

1.6a. You may use `\dir{}` for a "dummy" directional object (in fact this is used automatically by `**{}`). This is useful for a uniform treatment of connections, *e.g.*, making the `?` ⟨pos⟩ able to find a point on the straight line from p to c without actually typesetting anything.

Uses an empty droppping, the `\no@@` connection. All the variants are defined for optimisation reasons and it is also named `\dir{ }` to allow spurious spaces:

```
\xydefcsname@{dir{}}{\no@}

\xyletcsnamecsname@{dir0{}}{dir{}}
\xyletcsnamecsname@{dir1{}}{dir{}}
\xyletcsnamecsname@{dir^{}}{dir{}}
\xyletcsnamecsname@{dir_{}}{dir{}}
\xyletcsnamecsname@{dir2{}}{dir{}}
\xyletcsnamecsname@{dir3{}}{dir{}}

\xyletcsnamecsname@{dir{ }}{dir{}}

\xydef@\no@{\egroup \czeroEdge@ %\Invisible@false \Hidden@false %%%%HACK?
 \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\setbox\z@=\copy\voidb@x}\def\Connect@@{\no@@}}
```

1.6b. The *plain connectors* group contains basic directionals that lend themself to simple connections.

The bulk of the code is in fact in the description of these. First each of the three types—lines, dots, and squiggles—then the code for doubling and tripling.

Lines: A single `\dir{-}` object is a dash in the current direction: build box with character C of the semidirectional `\xydashfont`; use the characters natural width w and construct a height/depth from $d = |\sin(\texttt{\Direction})|$ `em` (where `1em` is the dash length, *cf.* `xydash10.mf`) as follows:

C	L	R	D	U	*flip if*
$0\ldots 30$	0	w	0	d	$dY < 0$
$31\ldots 63$	0	w	d	0	$dY > 0$
$64\ldots 95$	0	w	d	0	$dX < 0$
$96\ldots 127$	0	w	0	d	$dX < 0$

where *flip* means shift the box opposite vertically and horizontally, *i.e.*, $(L, R, D, U) := (R, L, U, D)$, and then lower the box $D - U$.

Procedure: (1.5132) Compute d, (1.5133) set D, U, and flip condition, (1.5137) build box to get w, L, R, (1.5139) dump box that is flipped if condition holds, and (1.5148) finally setup the required parameters properly.

```
5125 \xydefcsname@{dir1{-}}{\line@}
5126 \xydefcsname@{dir2{-}}{\line@ \double@\xydashh@}
5127 \xydefcsname@{dir3{-}}{\line@ \triple@\xydashh@}
5128 \xyletcsnamecsname@{dir0{-}}{dir{}}
5129 \xyletcsnamecsname@{dir{-}}{dir1{-}}
5130 \xyletcsnamecsname@{dir{=}}{dir2{-}}

5132 \xydef@\line@{\dimen@=\sd@Y\sinDirection\xydashl@
5133   \ifnum\SemiDirectionChar<31 \D@c=\z@ \U@c=\dimen@ \DN@{\d@Y<\z@}%
5134   \else\ifnum\SemiDirectionChar<64 \D@c=\dimen@ \U@c=\z@ \DN@{\z@<\d@Y}%
5135   \else\ifnum\SemiDirectionChar<96 \D@c=\dimen@ \U@c=\z@ \DN@{\d@X<\z@}%
5136   \else \D@c=\z@ \U@c=\dimen@ \DN@{\d@X<\z@}\fi\fi\fi
5137   \setboxz@h{\line@@}\ht\z@=\U@c \dp\z@=\D@c
5138   \L@c=\z@ \R@c=\wdz@
5139   \ifdim\next@ \dimen@=\R@c \R@c=\L@c \L@c=\dimen@
5140    \dimen@=\U@c \U@c=\D@c \D@c=\dimen@ \advance\dimen@-\U@c
5141    \lower\dimen@\boxz@
5142   \else \boxz@ \fi
5143 %
5144 %\ifnum\SemiDirectionChar=31\relax
5145 % \W@{LINE@(\the\SemiDirectionChar): WH=<\the\L@c+\the\R@c,\the\D@c+\the\U@c>;
5146 %       d=<\the\d@X,\the\d@Y>}\fi
5147 %
5148   \edef\tmp@{\egroup \U@c=\the\U@c \D@c=\the\D@c \L@c=\the\L@c \R@c=\the\R@c}%
5149  \tmp@
5150  \Edge@c={\rectangleEdge}%%%%%\Invisible@false\Hidden@false
5151  \ifdim\z@<\U@c \def\Upness@{1}\else \def\Upness@{0}\fi
5152  \ifdim\z@<\L@c \def\Leftness@{1}\else \def\Leftness@{0}\fi
5153  \def\Drop@@{\styledboxz@}\def\Connect@@{\solid@}}

5155 \xydef@\line@@{{\xydashfont\SemiDirectionChar\/}}
```

Bug: `\line@` should allow the size of the object to be changed after typesetting—this should make `\Connect@@` do dashing. Hm.

As mentioned above a dash will 'Connect' to make lines by using rules when strictly horizontal or vertical. This is controlled by enabling or disabling the test `\ifjusthvtest@` discussed below.

```
5166 \xydef@\solid@{%
5167 %
```

```
%\W@{SOLID}\xystatus@:%
%
 \checkXy@rules@\solidhrule@\solidvrule@{\straight@\solidSpread@}}
```

```
\xydef@\checkXy@rules@#1#2#3{%
 \dimen@=\Y@c \advance\dimen@-\Y@p
 \ifjusthvtest@.05pt>\ifdim\dimen@<\z@-\fi\dimen@ \DN@{#1}%
 \else \dimen@=\X@c \advance\dimen@-\X@p
  \ifjusthvtest@.05pt>\ifdim\dimen@<\z@-\fi\dimen@ \DN@{#2}%
  \else \DN@{#3}\fi\fi \next@}
```

Finally, we give the algorithm for ‘spreading’ the dashes along a solid line: just add an extra dash so they always overlap (see §1.8.3 for a proper defintion of the requirements to spreading).

```
\xydef@\solidSpread@{\ifnum\z@<\count@@ \advance\count@@\@ne \fi}
```

By default XY-pic will typeset horizontal and vertical `\dir{-}` connections using TEX rules. Unfortunately rules is the feature of the DVI format most commonly handled wrong by DVI drivers. Therefore XY-pic provides the ⟨decor⟩ations

---

```
\NoRules
\UseRules
```

---

that will switch the use of such off and on.

They simply redefine the conditional used to select typesetting with rules in `\solid@` above:

```
\xylet@\ifjusthvtest@=\ifdim
```

```
\xydef@\NoRules{\let\ifjusthvtest@=\iffalse}
\xydef@\UseRules{\let\ifjusthvtest@=\ifdim}
```

The actual typesetting essentially means calling `\drop@` to box with a rule of the appropriate length and with line width set to that of `\xydashfont` (as stored in `\xydashw@`).

```
\xydef@\solidvrule@{\no@@ \solidvrule@typeset \ifHidden@\else\noinsert@\fi
 \solidvrule@pre}
```

```
\xydef@\solidvrule@pre{%
%
%\W@{VRULE SETUP}%
%
 \def\Cbreak@@{%
%
%\W@{VRULE Cbreak}%
%
  \connectRestore@ \swap@\lastbreak@@\swap@
  \solidvrule@typeset \edef\lastbreak@@{\cfromthec@}\Creset@@}%
 \def\Clast@@{%
%
%\W@{VRULE Clast}%
%\let\Clast@@=\undefined
%
  \connectRestore@ \Creset@@\swap@\lastbreak@@\swap@
  \solidvrule@typeset \Creset@@ \edef\lastbreak@@{\cfromthep@}}%
```

```
}

\xydef@\solidvrule@typeset{%
%
%\W@{VRULE TYPESET}\xystatus@cp:%
%
 \ifInvisible@ \let\next@=\relax \else \let\next@=\solidvrule@typeset@ \fi
 \checkoverlap@@ \next@}

\xydef@\solidvrule@typeset@{{%
%
 \advance\X@c\X@p \X@c=.5\X@c \X@p=\X@c
 \advance\X@c-.5\xydashw@ \advance\X@p.5\xydashw@
%
 \ifdim\Y@c<\Y@p \advance\Y@c\U@c \advance\Y@p-\D@p
 \else \swapdimen@\Y@c\Y@p \advance\Y@c\U@p \advance\Y@p-\D@c \fi
%
 \Droprule@}}

\xydef@\solidhrule@{\no@@ \solidhrule@typeset \ifHidden@\else\noinsert@\fi
 \solidhrule@pre}

\xydef@\solidhrule@pre{%
%
%\W@{HRULE SETUP}%
%
 \def\Cbreak@@{%
%
%\W@{HRULE Cbreak}%
%
  \connectRestore@ \swap@\lastbreak@@\swap@
  \solidhrule@typeset \edef\lastbreak@@{\cfromthec@}\Creset@@}%
 \def\Clast@@{%
%
%\W@{HRULE Clast}%
%\let\Clast@@=\undefined
%
  \connectRestore@ \Creset@@\swap@\lastbreak@@\swap@
  \solidhrule@typeset \Creset@@ \edef\lastbreak@@{\cfromthep@}}%
}

\xydef@\solidhrule@typeset{%
%
%\W@{HRULE TYPESET}\xystatus@cp:%
%
 \ifInvisible@ \let\next@=\relax \else \let\next@=\solidhrule@typeset@ \fi
 \checkoverlap@@ \next@}

\xydef@\solidhrule@typeset@{{%
 \ifdim\X@c<\X@p \advance\X@c\R@c \advance\X@p-\L@p
 \else \swapdimen@\X@c\X@p \advance\X@c\R@p \advance\X@p-\L@c \fi
 \advance\Y@c\Y@p \Y@c=.5\Y@c \Y@p=\Y@c
 \advance\Y@c-.5\xydashw@ \advance\Y@p.5\xydashw@
```

```
 \Droprule@}}
```

`\Droprule@` puts the actual ink on the page: with lower left corner at c and upper right corner at p.

```
\xydef@\Droprule@{\advance\X@p-\X@c
%
%\W@{: kern\the\X@c\space width\the\X@p\space depth-\the\Y@c\space
%       height\the\Y@p}%
%
 \setboxz@h{\kern\X@c \vrule width\X@p depth-\Y@c height\Y@p}%
 \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ \Drop@@}
```

Dots: `\dir{.}` creates a very boring dot when used as an ⟨object⟩, but interesting dotted lines when used to connect. `\zerodot` should expand to a zero-sized box with a dot (initialised to use `\zerodotbox@`); the ⟨object⟩ is built using `\pointlike@` ⟨text⟩ ⟨spread-dimen⟩ that we will use again later.

```
\xydef@\zerodot{\copy\zerodotbox@}

\xydefcsname@{dir1{.}}{\point@}
\xydefcsname@{dir2{.}}{\point@ \double@\xydashh@}
\xydefcsname@{dir3{.}}{\point@ \triple@\xydashh@}
\xyletcsnamecsname@{dir0{.}}{dir{}}
\xyletcsnamecsname@{dir{.}}{dir1{.}}
\xyletcsnamecsname@{dir{:}}{dir2{.}}

\xydef@\point@{\pointlike@\zerodot\p@}

\xydef@\pointlike@#1#2{%
 \setboxz@h{#1}\wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@\egroup
% \Invisible@false \Hidden@false
 \def\Leftness@{.5}\def\Upness@{.5}\ctipEdge@
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@{\dottedSpread@{#2}}}}
```

This is reflected by the rather complicated spreading routine: 'Dotting' is the art of putting zero-sized objects together with equal distance independent of the chosen direction. So we must recompute the number of segments N (likely to be very big or -1) with trigonometry; using *radius* for the individual dots this becomes

$$
\begin{gathered}
A := |\cos \texttt{\Direction}| \times \mathit{radius} \\
B := |\sin \texttt{\Direction}| \times \mathit{radius} \\
\mathit{if}\ \ A + B > |\texttt{\d@X}| + |\texttt{\d@Y}|\ \ \mathit{then}\ N = 0\ \mathit{and\ exit} \\
\texttt{<}A\texttt{,}B\texttt{>} := 2 \times \texttt{<}A\texttt{,}B\texttt{>} \\
\mathit{Filler} := \langle \text{box with the original filler centered...} \rangle \\
\texttt{<\d@X,\d@Y>} := \texttt{<\d@X,\d@Y>} + \texttt{<\sd@X} \times A\texttt{,\sd@Y} \times B\texttt{>} \\
\texttt{<}X\texttt{,}Y\texttt{>} := \texttt{<}X\texttt{,}Y\texttt{>} + \texttt{<\sd@X} \times A/2\texttt{,\sd@Y} \times B/2\texttt{>} \\
N := \lfloor \mathit{if}\ \ |\texttt{\d@X}| > |\texttt{\d@Y}|\ \ \mathit{then}\, |\texttt{\d@X}|\,/A \mathit{else}\, |\texttt{\d@Y}|\,/B \rfloor + 1
\end{gathered}
$$

as realised below:

```
\xydef@\dottedSpread@#1{%\setupDirection@
 \dimen@=#1\relax \DN@{\A@=2\A@ \B@=2\B@ \dottedSpread@i}%
 \A@=\sd@X\cosDirection\dimen@ \B@=\sd@Y\sinDirection\dimen@
```

```
 \dimen@=\A@ \advance\dimen@\B@
 \dimen@ii=\sd@X\d@X \advance\dimen@ii\sd@Y\d@Y
 \ifdim\dimen@>\dimen@ii \DN@{\count@@=\z@}\fi
 \next@}
```

```
\xydef@\dottedSpread@i{%\setupDirection@
 \global\setbox8=\hbox to\A@{\hss
  \kern.5\A@\box\lastobjectbox@\kern.5\A@\hss}\dp8=.5\B@ \ht8=.5\B@
 \advance\d@X\sd@X\A@ \advance\d@Y\sd@Y\B@
 \advance\X@c\sd@X.5\A@ \advance\Y@c\sd@Y.5\B@
 \ifdim\sd@Y\d@Y<\sd@X\d@X \dimen@=\sd@X\d@X
  \ifdim\zz@\A@\else \divide\dimen@\A@ \fi \count@@=\dimen@
 \else \dimen@=\sd@Y\d@Y
  \ifdim\zz@\B@\else \divide\dimen@\B@ \fi \count@@=\dimen@
 \fi \advance\count@@\@ne}
```

A particular aspect of this is that we have to enlarge and recenter the actual box used for the typesetting (`\lastobjectbox@`).

Squiggles: These are just a lot of box maneuvering using the directional characters of `\xybsqlfont` (see `xybsql10.mf` for details):

```
\xydefcsname@{dir1{~}}{\squiggle@}
\xydefcsname@{dir2{~}}{\squiggle@ \double@\xybsqlh@}
\xydefcsname@{dir3{~}}{\squiggle@ \triple@\xybsqlh@}
\xyletcsnamecsname@{dir0{~}}{dir{}}
\xyletcsnamecsname@{dir{~}}{dir1{~}}
```

```
\xydef@\squiggle@{\xybsqlfont
  \dimen@=\sd@X\cosDirection\xybsqll@ \advance\dimen@.1\p@
  \dimen@ii=\sd@Y\sinDirection\xybsqll@
  \kern\dimen@\squiggle@@
 \edef\tmp@{\egroup \U@c=\the\dimen@ii \L@c=\the\dimen@}\tmp@
 \wdz@=2\L@c \R@c=\L@c \ht\z@=\U@c  \D@c=\U@c \dp\z@=\U@c
 \Edge@c={\rectangleEdge}%
% \Invisible@false \Hidden@false
 \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@\squiggledSpread@}}
```

```
\xydef@\squiggle@@{\DirectionChar \count@=\DirectionChar
 \advance\count@-64 \ifnum\count@<\z@ \advance\count@128 \fi \char\count@}
```

The interesting bit is that they spread by not spreading, *i.e.*, by centering between the endpoints—this means

$$\begin{aligned} X &:= X - d/2, dX := dX - d \quad \text{where } d = \texttt{\textbackslash sd@X}(|dX| - N \times A + \texttt{.1pt}) \\ Y &:= Y - d/2, dY := dY - d \quad \text{where } d = \texttt{\textbackslash sd@Y}(|dY| - N \times B + \texttt{.1pt}) \end{aligned}$$

```
\xydef@\squiggledSpread@{%
 \dimen@=\d@X \advance\dimen@-\sd@X\count@@\A@ \advance\dimen@\sd@X.3\p@
 \advance\X@c-.5\dimen@ \advance\d@X-\dimen@
 \dimen@=\d@Y \advance\dimen@-\sd@Y\count@@\B@ \advance\dimen@\sd@Y.3\p@
 \advance\Y@c-.5\dimen@ \advance\d@Y-\dimen@}
```

Double and triple directionals: As can be seen by the last two columns, these (and most of the other connectors) also exist in double and triple versions with a 2 or a 3 prepended to the name. For convenience `\dir{=}` and `\dir{:}` are synonyms for `\dir2{-}` and `\dir2{.}`, respectively; similarly `\dir{==}` is a synonym for `\dir2{--}`.

This is very simple, really: `\double@` and `\triple@` do the work by redefining the `\Drop@@` method to do its job twice and thrice. **To Do:** should probably extend any already defined `\Drop@@` method?

```
%\xydef@\double@#1{\edef\Drop@@{\dimen@=#1\relax
%  \dimen@=.5\dimen@ \A@=-\sinDirection\dimen@ \B@=\cosDirection\dimen@
%  \setbox2=\hbox{\kern\A@\raise\B@\copy\z@}\dp2=\z@ \ht2=\z@ \wd2=\z@ \box2 %
%  \setbox2=\hbox{\kern-\A@\raise-\B@\boxz@}\dp2=\z@ \ht2=\z@ \wd2=\z@ \box2 }}

%\xydef@\triple@#1{\edef\Drop@@{\dimen@=#1\relax
%  \A@=-\sinDirection\dimen@ \B@=\cosDirection\dimen@
%  \setbox2=\hbox{\kern\A@\raise\B@\copy\z@}\dp2=\z@ \ht2=\z@ \wd2=\z@ \box2 %
%  \setbox2=\hbox{\kern-\A@\raise-\B@\copy\z@}\dp2=\z@ \ht2=\z@ \wd2=\z@ \box2
%  \dp\z@=\z@ \ht\z@=\z@ \wdz@=\z@ \boxz@}}

\xydef@\double@#1{\edef\Drop@@{\dimen@=#1\relax
  \dimen@=.5\dimen@ \A@=-\sinDirection\dimen@ \B@=\cosDirection\dimen@
  \setboxz@h{\setbox2=\hbox{\kern\A@\raise\B@\copy\z@}%
   \dp2=\z@ \ht2=\z@ \wd2=\z@ \box2 %
   \setbox2=\hbox{\kern-\A@\raise-\B@ \noexpand\boxz@}%
   \dp2=\z@ \ht2=\z@ \wd2=\z@ \box2 }%
  \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \noexpand\styledboxz@}}

\xydef@\triple@#1{\edef\Drop@@{\dimen@=#1\relax
  \A@=-\sinDirection\dimen@ \B@=\cosDirection\dimen@
  \setboxz@h{\setbox2=\hbox{\kern\A@\raise\B@\copy\z@}%
   \dp2=\z@ \ht2=\z@ \wd2=\z@ \box2 %
   \setbox2=\hbox{\kern-\A@\raise-\B@\copy\z@}%
   \dp2=\z@ \ht2=\z@ \wd2=\z@ \box2 %
   \dp\z@=\z@ \ht\z@=\z@ \wdz@=\z@ \noexpand\boxz@}%
  \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \noexpand\styledboxz@}}
```

Dashing directionals: First traditional dashing:

```
\xydefcsname@{dir1{--}}{\dash@}
\xydefcsname@{dir2{--}}{\dash@ \double@\xydashh@}
\xydefcsname@{dir3{--}}{\dash@ \triple@\xydashh@}
\xyletcsnamecsname@{dir0{--}}{dir{}}
\xyletcsnamecsname@{dir{--}}{dir1{--}}
\xyletcsnamecsname@{dir{==}}{dir2{--}}

\xydef@\dash@{\line@ \wdz@=2\wdz@ \ht\z@=2\ht\z@ \dp\z@=2\dp\z@
 \multiply\D@c\tw@ \multiply\U@c\tw@ \multiply\L@c\tw@ \multiply\R@c\tw@
 \def\Connect@@{\straight@\dashedSpread@}}
```

Since the dashes should reach the endpoints we do this:

$$\begin{aligned}
&\textit{if } N > 0 \textit{ then } N := N + 1\\
&dX := dX + d/2 \textit{ where } d = \texttt{\textbackslash sd@X}A\\
&dY := dY + d/2 \textit{ where } d = \texttt{\textbackslash sd@Y}B\\
&\textit{if } dX > 0 \textit{ then } X := X + A/2\\
&Y := Y + \texttt{\textbackslash sd@Y}A/2
\end{aligned}$$

```
\xydef@\dashedSpread@{\ifnum\z@<\count@@ \advance\count@@\@ne \fi
 \advance\d@X\sd@X.5\A@ \advance\d@Y\sd@Y.5\B@
 \ifdim\z@<\d@X \advance\X@c.5\A@ \fi \advance\Y@c\sd@Y.5\B@}
```

Dashed dashing of squiggled lines are simpler since squiggles are symmetric:

```
\xydefcsname@{dir1{~~}}{\dashsquiggle@}
\xydefcsname@{dir2{~~}}{\dashsquiggle@ \double@\xybsqlh@}
\xydefcsname@{dir3{~~}}{\dashsquiggle@ \triple@\xybsqlh@}
\xyletcsnamecsname@{dir0{~~}}{dir{}}
\xyletcsnamecsname@{dir{~~}}{dir1{~~}}
```

```
\xydef@\dashsquiggle@{\squiggle@
 \multiply\D@c\tw@ \multiply\U@c\tw@ \multiply\L@c\tw@ \multiply\R@c\tw@
 \dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@ \ht\z@=\U@c \dp\z@=\D@c
 \def\Connect@@{\straight@\dashsquiggledSpread@}}
```

The spreading of squiggles is similarly simpler: we just shave 1/4 squiggle size of each end of the conection in order to eliminate the blank space at both ends:

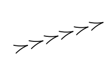
```
\xydef@\dashsquiggledSpread@{\ifnum\z@<\count@@ \advance\count@@\@ne \fi
 \advance\X@c.5\A@ \advance\d@X.5\A@ \advance\Y@c.25\B@ \advance\d@Y.5\B@}
```

Finally "dashed dotting" synonyms:

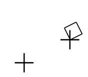
```
\xyletcsnamecsname@{dir1{..}}{dir{.}}
\xyletcsnamecsname@{dir2{..}}{dir2{.}}
\xyletcsnamecsname@{dir3{..}}{dir3{.}}
\xyletcsnamecsname@{dir{..}}{dir1{.}}
\xyletcsnamecsname@{dir{::}}{dir2{.}}
```

1.6c. The group of *plain tips* contains basic objects that are useful as markers and arrowheads making connections, so each is shown at the end of a dotted connection of the appropriate kind.

They may also be used as connectors and will build dotted connections. *e.g.*, `**@{>}` typesets



Exercise 1.14: Typeset the following two +s and a tilted square:



Hint: the dash created by `\dir{-}` has the length `5pt` (here).

```
\xylet@\ctipEdge@=\czeroEdge@
```

To Do: Change tips to have a tiny size of 2sp which may be taken as an indication that it is a tip (this can be used by some features, *e.g.*, 'arrow').

Arrow heads: The ones intended for single connections are just characters from `\xyatipfont` and `\xybtipfont`.

```
\xydefcsname@{dir1{>}}{\tip@}
\xydefcsname@{dir^{>}}{\atip@}
\xydefcsname@{dir_{>}}{\btip@}
\xyletcsnamecsname@{dir0{>}}{dir{}}
\xyletcsnamecsname@{dir{>}}{dir1{>}}

\xydefcsname@{dir1{<}}{\reverseDirection@\tip@}
\xydefcsname@{dir^{<}}{\reverseDirection@\btip@}
\xydefcsname@{dir_{<}}{\reverseDirection@\atip@}
\xyletcsnamecsname@{dir0{<}}{dir{}}
\xyletcsnamecsname@{dir{<}}{dir1{<}}

\xydef@\tip@{\tip@x\tip@@}
\xydef@\atip@{\tip@x\atip@@}
\xydef@\btip@{\tip@x\btip@@}

\xydef@\tip@x#1{#1\egroup
 \ctipEdge@ %\Invisible@false \Hidden@false
 \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@{\dottedSpread@\jot}}}

\xydef@\tip@@{\atip@@\btip@@}
\xydef@\atip@@{\xyatipfont\DirectionChar}
\xydef@\btip@@{\xybtipfont\DirectionChar}
```

Double and triple tips are realised by taking the two halfs and 'wringing them apart'; as the naming indicates they are meant to be put at the end of 2- and 3-connections. This is currently done the slightly hacky (but efficient) way of adding directly to `\DirectionChar`; maybe this should be using `\vDirection@`?

```
\xydefcsname@{dir2{>}}{\Tip@}
\xydefcsname@{dir2{<}}{\reverseDirection@\Tip@}

\xydef@\Tip@{\kern2.5pt \vrule height2.5pt depth2.5pt width\z@
 \Tip@@ \kern2.5pt \egroup
 \U@c=2.5pt \D@c=2.5pt \L@c=2.5pt \R@c=2.5pt \Edge@c={\circleEdge}%
% \Invisible@false \Hidden@false
 \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@{\dottedSpread@\jot}}}

\xydef@\Tip@@{\count@=\DirectionChar
 \advance\count@-4 \ifnum\count@<\z@ \advance\count@128 \fi
 \xyatipfont\char\count@
 \advance\count@ 8 \ifnum127<\count@ \advance\count@-128 \fi
 \xybtipfont\char\count@}

\xydefcsname@{dir3{>}}{\Ttip@}
\xydefcsname@{dir3{<}}{\composite@{}{:(-1,0)\dir3{>}}}

\xydef@\Ttip@{\kern3.2pt \vrule height3.2pt depth3.2pt width\z@
 \Ttip@@ \kern3.2pt \egroup
 \U@c=3.2pt \D@c=3.2pt \L@c=3.2pt \R@c=3.2pt \Edge@c={\circleEdge}%
% \Invisible@false \Hidden@false
```

```
 \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@{\dottedSpread@\jot}}}
```

```
\xydef@\Ttip@@{%
 \setboxz@h\bgroup\reverseDirection@\line@ \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@
 \kern-\L@c \boxz@ \kern\L@c
 {\vDirection@(1,-.31)\xydashl@ \xyatipfont\char\DirectionChar}%
 {\vDirection@(1,+.31)\xydashl@ \xybtipfont\char\DirectionChar}}
```

Stopper: `\dir{|}` makes a 'stopper' using just the appropriate `\xydashfont` character rotated 90° and centered; the `^` and `_` variants are just shifted appropriately and two are used to make the 2 and 3 variants longer.

```
\xydefcsname@{dir1{|}}{\stopper@}
\xydefcsname@{dir^{|}}{\aboveDirection@\xydashl@
 \shiftdir@\line@\z@ \pointlike@{}\xydashh@}
\xydefcsname@{dir_{|}}{\belowDirection@\xydashl@
 \shiftdir@\line@\z@ \pointlike@{}\xydashh@}
\xydefcsname@{dir2{|}}{\stopper@ \double@\xydashh@}
\xydefcsname@{dir3{|}}{\stopper@ \double@{2\xydashh@}}
```

```
\xyletcsnamecsname@{dir0{|}}{dir{}}
\xyletcsnamecsname@{dir{|}}{dir1{|}}
```

```
\xydef@\stopper@{\tip@x\stopper@@}
```

```
\xydef@\stopper@@{\setboxz@h{\count@=\SemiDirectionChar \advance\count@64 %
  \ifnum127<\count@ \advance\count@-128 \fi \xydashfont\char\count@\/}%
 \setboxz@h{\kern-.5\wdz@ \dimen@=\sd@Y\cosDirection\xydashl@
  \ifnum\SemiDirectionChar=95 \dimen@=\sd@X\sd@Y\dimen@ \fi
  \raise.5\dimen@\boxz@}%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
```

Hooks: These are halfcircles opening towards or opposite `\Direction` and fastened by their center or either endpoint. Build by lots of box manipulation with the `\xybsqlfont` quarter circles. . . ☺

```
\xydefcsname@{dir1{(}}{\hook@}
\xydefcsname@{dir^{(}}{\ahook@}
\xydefcsname@{dir_{(}}{\bhook@}
\xyletcsnamecsname@{dir0{(}}{dir{}}
\xyletcsnamecsname@{dir{(}}{dir1{(}}
```

```
\xydefcsname@{dir1{)}}{\reverseDirection@\hook@}
\xydefcsname@{dir^{)}}{\reverseDirection@\bhook@}
\xydefcsname@{dir_{)}}{\reverseDirection@\ahook@}
\xyletcsnamecsname@{dir0{)}}{dir{}}
\xyletcsnamecsname@{dir{)}}{dir1{)}}
```

```
\xydef@\hook@{\tip@x\hook@@}
\xydef@\hook@@{\setboxz@h{\xybsqlfont
  \vDirection@(1,-1){.707107\xybsqll@}%
  \hbox{\DirectionChar
```

```
   \kern-\d@Y\raise\d@X\hbox{\count@=\DirectionChar \advance\count@-32 %
    \ifnum\count@<\z@ \advance\count@128 \fi \char\count@}}}%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
```

```
\xydef@\ahook@{\tip@x\ahook@@}
\xydef@\ahook@@{\setboxz@h{\xybsqlfont
  \vDirection@(1,-1){.707107\xybsqll@}\kern-\d@X
  \lower\d@Y\hbox{\DirectionChar
   \kern-\d@Y\raise\d@X\hbox{\count@=\DirectionChar \advance\count@-32 %
    \ifnum\count@<\z@ \advance\count@128 \fi \char\count@}}}%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
```

```
\xydef@\bhook@{\tip@x\bhook@@}
\xydef@\bhook@@{\setboxz@h{\xybsqlfont
  \vDirection@(-1,-1){.707107\xybsqll@}\DirectionChar
  \kern\d@X\raise\d@Y\hbox{\count@=\DirectionChar \advance\count@-96 %
   \ifnum\count@<\z@ \advance\count@128 \fi \char\count@}}%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
```

Quarter turns: These are quarter circles fastened by their start or end point in `\Direction`. Build by box manipulation of the `\xybsqlfont` quarter circles. The intention is that the `‘’` directionals are half the corresponding `()` directional.

```
\xydefcsname@{dir^{’}}{\reverseDirection@\bturn@}
\xydefcsname@{dir_{’}}{\reverseDirection@\aturn@}
```

```
\xydefcsname@{dir^{‘}}{\aturn@}
\xydefcsname@{dir_{‘}}{\bturn@}
```

```
\xydef@\aturn@{\tip@x\aturn@@}
\xydef@\aturn@@{\setboxz@h{\xybsqlfont
  \vDirection@(1,-1){.707107\xybsqll@}\kern-\d@X
  \lower\d@Y\hbox{\DirectionChar}}%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
```

```
\xydef@\bturn@{\tip@x\bturn@@}
\xydef@\bturn@@{\setboxz@h{\xybsqlfont
  \vDirection@(-1,-1){.707107\xybsqll@}\DirectionChar}%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
```

1.6d. These tips are combinations of the plain tips provided for convenience (and optimised for efficiency). New ones can be constructed using `\composite` and by declarations of the form

---

`\newdir` ⟨dir⟩ `{`⟨composite⟩`}`

---

which defines `\dir`⟨dir⟩ as the ⟨composite⟩ (see note 1.4d for the details).

`\newdir` is simple:

```
\xydef@\newdir#1#{\newdir@{#1}}
```

```
\xydef@\newdir@#1#2#3{\xydefcsname@{dir#1{#2}}{\composite@{}{#3}}}
```

Then the somewhat more efficient `\shiftdir@` used internally for moving a tip in the current direction—it does so by making a local hbox within which the argument tip is constructed and subsequently shifted and made of zero size. Use as

`\shiftdir@`⟨tip@⟩⟨dimen⟩⟨tip@⟩

where ⟨tip@⟩ means a tip command without the leading `\hbox{`.

```
\xydef@\shiftdir@#1#2{%
 \setbox\z@=\hbox\bgroup#1\relax
 \setboxz@h{\dimen@ii=#2\relax
  \dimen@=-\cosDirection\dimen@ii \advance\dimen@-\L@c
  \kern\dimen@ \lower\sinDirection\dimen@ii\boxz@}%
 \wdz@\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
```

Then the tips, with the `\tipjot@` hook allowing changing the spacing of tips used for single lines.

```
\xylet@\tipjot@=\jot

\xydefcsname@{dir1{>>}}{\shiftdir@\tip@\tipjot@ \tip@}
\xydefcsname@{dir^{>>}}{\shiftdir@\atip@\tipjot@ \atip@}
\xydefcsname@{dir_{>>}}{\shiftdir@\btip@\tipjot@ \btip@}
\xydefcsname@{dir2{>>}}{\composite@{}{h!/\tipjot@/\dir2{>}*\dir2{>}}}
\xydefcsname@{dir3{>>}}{\composite@{}{h!/\tipjot@/\dir3{>}*\dir3{>}}}
\xyletcsnamecsname@{dir0{>>}}{dir{}}
\xyletcsnamecsname@{dir{>>}}{dir1{>>}}

\xydefcsname@{dir1{<<}}{\reverseDirection@ \shiftdir@\tip@\tipjot@ \tip@}
\xydefcsname@{dir^{<<}}{\reverseDirection@ \shiftdir@\btip@\tipjot@ \btip@}
\xydefcsname@{dir_{<<}}{\reverseDirection@ \shiftdir@\atip@\tipjot@ \atip@}
\xydefcsname@{dir2{<<}}{\composite@{}{h!/-\tipjot@/\dir2{<}*\dir2{<}}}
\xydefcsname@{dir3{<<}}{\composite@{}{h!/-\tipjot@/\dir3{<}*\dir3{<}}}
\xyletcsnamecsname@{dir0{<<}}{dir{}}
\xyletcsnamecsname@{dir{<<}}{dir1{<<}}

\xydefcsname@{dir1{||}}{\shiftdir@\stopper@\xydashh@ \shiftdir@\stopper@\z@
 \pointlike@{}\jot}
\xydefcsname@{dir^{||}}{\shiftdir@{\aboveDirection@\xydashl@\line@}\xydashh@
 \shiftdir@{\aboveDirection@\xydashl@\line@}\z@ \pointlike@{}\jot}
\xydefcsname@{dir_{||}}{\shiftdir@{\belowDirection@\xydashl@\line@}\xydashh@
 \shiftdir@{\belowDirection@\xydashl@\line@}\z@ \pointlike@{}\jot}
\xydefcsname@{dir2{||}}{\shiftdir@\stopper@\xydashh@ \shiftdir@\stopper@\z@
 \pointlike@{}\jot \double@\xydashh@}
\xydefcsname@{dir3{||}}{\shiftdir@\stopper@\xydashh@ \shiftdir@\stopper@\z@
 \pointlike@{}\jot \double@{2\xydashh@}}
\xyletcsnamecsname@{dir0{||}}{dir{}}
\xyletcsnamecsname@{dir{||}}{dir1{||}}

\xydefcsname@{dir1{>|}}{\shiftdir@\stopper@\z@ \tip@}
\xydefcsname@{dir2{>|}}{\composite@{}{\dir2{>}*\dir2{|}}}
\xydefcsname@{dir3{>|}}{\composite@{}{\dir3{>}*\dir3{|}}}
\xyletcsnamecsname@{dir0{>|}}{dir{}}
\xyletcsnamecsname@{dir{>|}}{dir1{>|}}

\xydefcsname@{dir1{>>|}}{\shiftdir@\stopper@\z@ \shiftdir@\tip@\tipjot@ \tip@}
\xydefcsname@{dir2{>>|}}{\composite@{}{h!/\tipjot@/\dir2{>}*\dir2{>}*\dir2{|}}}
\xydefcsname@{dir3{>>|}}{\composite@{}{h!/\tipjot@/\dir3{>}*\dir3{>}*\dir3{|}}}
```

```
\xyletcsnamecsname@{dir0{>>|}}{dir{}}
\xyletcsnamecsname@{dir{>>|}}{dir1{>>|}}

\xydefcsname@{dir1{|<}}{\reverseDirection@ \shiftdir@\stopper@\z@ \tip@}
\xydefcsname@{dir2{|<}}{\reverseDirection@ \shiftdir@\stopper@\z@ \Tip@}
\xydefcsname@{dir3{|<}}{\reverseDirection@ \shiftdir@\stopper@\z@ \Ttip@}
\xyletcsnamecsname@{dir0{|<}}{dir{}}
\xyletcsnamecsname@{dir{|<}}{dir1{|<}}

\xydefcsname@{dir1{|<<}}{\reverseDirection@
 \shiftdir@\stopper@\z@ \shiftdir@\tip@\tipjot@ \tip@}
\xydefcsname@{dir2{|<<}}{%
 \composite@{}{h!/-\tipjot@/\dir2{<}*\dir2{<}*\dir2{|}}}
\xydefcsname@{dir3{|<<}}{%
 \composite@{}{h!/-\tipjot@/\dir3{<}*\dir3{<}*\dir3{|}}}
\xyletcsnamecsname@{dir0{|<<}}{dir{}}
\xyletcsnamecsname@{dir{|<<}}{dir1{|<<}}

\xydefcsname@{dir1{|-}}{\shiftdir@\stopper@\z@
 \shiftdir@\line@\z@ \pointlike@{}\jot}
\xydefcsname@{dir^{|-}}{\shiftdir@{\aboveDirection@\xydashl@ \line@}\z@
 \shiftdir@\line@\z@ \pointlike@{}\jot}
\xydefcsname@{dir_{|-}}{\shiftdir@{\belowDirection@\xydashl@ \line@}\z@
 \shiftdir@\line@\z@ \pointlike@{}\jot}
\xydefcsname@{dir2{|-}}{\shiftdir@\stopper@\z@
 \shiftdir@\line@\z@ \pointlike@{}\jot \double@\xydashh@}
\xydefcsname@{dir3{|-}}{\shiftdir@\stopper@\z@
 \shiftdir@\line@\z@ \pointlike@{}\jot \triple@\xydashh@}
\xyletcsnamecsname@{dir0{|-}}{dir{}}
\xyletcsnamecsname@{dir{|-}}{dir1{|-}}
\xyletcsnamecsname@{dir{|=}}{dir2{|-}}

\xydefcsname@{dir{+}}{%
 \DN@##1{\composite@{}{##10\dir{|}*!C##10\dir{-}}}\addEQ@\next@}
\xyletcsnamecsname@{dir0{+}}{dir{}}

\xydefcsname@{dir{x}}{\vDirection@(1,1)\jot
 \DN@##1{\composite@{}{##10\dir{|}*!C##10\dir{-}}}\addEQ@\next@}
\xyletcsnamecsname@{dir0{x}}{dir{}}

\xydefcsname@{dir{/}}{\vDirection@(1,-.3)\jot \stopper@}
\xyletcsnamecsname@{dir0{/}}{dir{}}

\xydefcsname@{dir{//}}{\vDirection@(1,-.3)\jot
 \shiftdir@\stopper@\xydashh@ \stopper@}
\xyletcsnamecsname@{dir0{//}}{dir{}}

\xydefcsname@{dir{*}}{\solidpoint@}
\xydef@\solidpoint@{%
 \pointlike@{\kern-1.8pt\lower1.8pt\hbox{$\scriptstyle\bullet$}}\jot}
\xyletcsnamecsname@{dir0{*}}{dir{}}

\xydefcsname@{dir{o}}{\hollowpoint@}
\xydef@\hollowpoint@{%
 \pointlike@{\kern-1.8pt\lower1.8pt\hbox{$\scriptstyle\circ$}}\jot}
```

```
\xyletcsnamecsname@{dir0{o}}{dir{}}
```

Syntax			Action
`\cir` ⟨radius⟩ `{` ⟨cir⟩ `}`			⟨cir⟩cle segment with ⟨radius⟩
⟨radius⟩	$\longrightarrow$	⟨empty⟩	use R_c as the radius
	$\vert$	⟨vector⟩	use X of the ⟨vector⟩ as radius
⟨cir⟩	$\longrightarrow$	⟨empty⟩	full circle of ⟨radius⟩
	$\vert$	⟨diag⟩ ⟨orient⟩ ⟨diag⟩	partial circle from first ⟨diag⟩onal through to the second ⟨diag⟩onal in the ⟨orient⟩ation
⟨orient⟩	$\longrightarrow$	`^`	anticlockwise
	$\vert$	`_`	clockwise

Figure 1.8: ⟨cir⟩cles.

1.6.2 Circle segments

```
\message{circles,}
```

Circle ⟨object⟩s are round and typeset a segment of the circle centered at the reference point. The syntax of circles is described in figure 1.8 with explanations below.

The `\cir` command is the hub: it parses the optional ⟨radius⟩ (to `\R@`, default from R_c) and `{`⟨cir⟩`}`, bailing out with a `\zerodot` if the radius is to small:

```
\xydef@\cir#1#{\hbox\bgroup
 \afterVECTORorEMPTY{\xy@@{\R@=\X@c}\cir@}{\xy@@{\R@=\R@c}\cir@}#1@}

\xydef@\cir@#1@#2{%
 \DN@{#1}\ifx\next@\empty\else \xyerror@{illegal circle <radius>: must be
        <vector> or <empty>}{}\fi
 \afterCIRorDIAG{\xyFN@\cir@cir}{\xyFN@\cir@diag}#2@}
```

The code to actually typeset the ⟨cir⟩ just parsed starts by checking that the ⟨cir⟩ was immediately followed by the `@` we put there in `\cir`:

```
\xydef@\cir@cir{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\cir@cir}%gobble spaces
 \else \ifx @\next \DN@ @{\cir@i}%
 \else \xyerror@{illegal <cir>: must have form <diag><orient><diag> or
        <empty>}{}%
 \fi\fi \next@}
```

Similarly when an ⟨empty⟩ one was given—the parser will recognise this as a ⟨diag⟩ but we hack that here:

```
\xydef@\cir@diag{%
 \DN@{\xyerror@{illegal <cir>: must have form <diag><orient><diag> or
        <empty>}{}}%
 \ifx @\next \ifnum\count@=8 %
   \DN@ @{\def\CIRin@@{0}\def\CIRorient@@{\CIRfull@}\def\CIRout@@{7}\cir@i}%
```

```
\fi\fi \next@}
```

...and then use the constructed methods to build it:

```
\xydef@\cir@i{%
 \ifnum\CIRin@@=8 \xyerror@{incomplete <cir> specification}{%
The <cir> you specified as <diag><orient><diag> is not sufficiently specific.}%
  \def\CIRin@@{0}\fi
 \ifdim\R@<.5\p@ \R@=\z@ \zerodot
 \else \CIRorient@@ \cirbuild@ \fi
 \L@c=\R@ \R@c=\R@ \D@c=\R@ \U@c=\R@ \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@\relax}%
 \Edge@c={\circleEdge}\OBJECT@x}
```

Parsing: The `\afterCIRorDIAG` parser handles the parsing: it either

- parses the ⟨cir⟩ and sets *in*, *orient*, and *out*, and passes control to the first argument continuation, or
- parses the single ⟨diag⟩ specified, store it in `\count@` (as `8` if an ⟨empty⟩ one given), and pass control to the second continuation argument,

where the ⟨diag⟩ internal representation number of note 1.4l of is used. An ⟨empty⟩ circle is treated as an ⟨empty⟩ diagonal; specifying an ⟨empty⟩ first ⟨diag⟩ of a ⟨cir⟩ is equivalent to using the value of the *in* method at call time.

The parser is very simple, setting methods stored in the usual `@@`-terminated control sequences (**To Do:** Rename all non-method control sequences that end in `@@`... to use `@`⟨romannumeral⟩ suffixes...):

```
\xydef@\CIRin@@{3}
\xydef@\CIRout@@{3}
\xylet@\CIRorient@@=\empty
\xydef@\afterCIRorDIAG#1#2{\def\afterCIR@{#1}\def\afterCIRDIAG@{#2}\xyFN@\CIR@}
\xylet@\afterCIR@=\empty
\xylet@\afterCIRDIAG@=\empty
\xydef@\CIR@{\count@=8 \afterDIAG{\edef\CIRin@@{\the\count@}\xyFN@\CIR@@}}
\xydef@\CIR@@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\CIR@@}%gobble spaces
 \else\ifx ^\next
  \DN@ ^{\def\CIRorient@@{\CIRacw@}%
   \afterDIAG{\edef\CIRout@@{\the\count@}\afterCIR@}}%
 \else\ifx _\next
  \DN@_{\def\CIRorient@@{\CIRcw@}%
   \afterDIAG{\edef\CIRout@@{\the\count@}\afterCIR@}}%
 \else
  \DN@{\def\CIRorient@@{\relax}\afterCIRDIAG@}%
 \fi\fi\fi \next@}
```

The default is to generate a *full circle* with the specified radius, *e.g.*,

`\xy*\cir<4pt>{}\endxy`	typesets	"○"
`\xy*{M}*\cir{}\endxy`	—	"M"

All the other circle segments are subsets of this and have the shape that the full circle outlines.

Finally we present the *orient* methods. They use these 'internal methods' to actually draw the circles

```
\xylet@\CIRtest@@=\relax
\xydef@\CIRlo@@{0}
\xydef@\CIRhi@@{0}
```

Below we call them *lo*, *hi*, and *test*; the first two are coded as described in note 1.4l and the last takes two arguments: a dimension and something to do if the test succeeds. `\count@@` and `\count@` should be set to *in* and *out* internally as well in case `\cirbuild@` and friends below should be used.

The dummy *orient* used for simple circles is the simplest:

```
\xydef@\CIRfull@{\def\CIRtest@@##1##2{##2}}
```

The kernel `\cirbuild@` builds the actual ⟨object⟩ using characters from the `\xycircfont` assumed coded like `xycirc10.mf`: `\cirrestrict@@` choses a group and adjusts the radius `\R@` to fit it exactly. The group is multiplied by 8 to get the group character offset [`\count@`].

```
\xydef@\cirbuild@{\cirrestrict@@ \multiply\count@8 %
 \circhar@0\circhar@7\kern\dimen@
 \circhar@1\circhar@6\kern\dimen@
 \circhar@2\circhar@5\kern\dimen@
 \circhar@3\circhar@4\kern\dimen@}
```

```
\xydef@\circhar@#1{%
 \setboxz@h{\circhar@@{#1}}\dimen@=\wdz@ \wdz@=\z@ \ht\z@=\R@ \dp\z@=\R@
 \CIRtest@@#1{\styledboxz@}\setbox\z@=\copy\voidb@x}
```

```
\xydef@\circhar@@#1{{\xycircfont \advance\count@#1\relax \char\count@}}
```

`\cirrestrict@@` computes the group g [`\count@`] of circle segments to use from the radius r [`\R@`] using the formula (the reverse of the one in `xycirc10.mf`)

$$g = \begin{cases} \left\lfloor \frac{r}{1\mathrm{pt}} \right\rfloor - 1 & \text{if } 1\mathrm{pt} \le r < 8\mathrm{pt} \\ \left\lfloor \frac{r}{2\mathrm{pt}} \right\rfloor + 3 & \text{if } 8\mathrm{pt} \le r < 16\mathrm{pt} \\ \left\lfloor \frac{r}{4\mathrm{pt}} \right\rfloor + 7 & \text{if } 16\mathrm{pt} \le r < 32\mathrm{pt} \\ 15 & \text{if } 32\mathrm{pt} \le r \end{cases}$$

(where we know from `\cir@i` that $r \ge \frac{1}{2}\mathrm{pt}$), and then adjusts the radius to be exactly the one chosen through the use of group g using the formula in `xycirc10.mf`...this is necessary because of the restriction on `tfm` files that they can only have 15 different nonzero heights and depths. Subsequent calls to `\cirrestrict@@` should compute the same values.

```
\xydef@\cirrestrict@@{\begingroup \dimen@=\R@
 \setboxz@h{\xycircfont\char\z@\char\@ne}\A@=\wdz@
 \ifdim\R@<8\A@ \count@=\dimen@ \divide\count@\A@ \advance\count@\m@ne
 \else\ifdim\R@<16\A@ \count@=\dimen@
  \dimen@=2\A@ \divide\count@\dimen@ \advance\count@3 %
 \else\ifdim\R@<32\A@ \count@=\dimen@
  \dimen@=4\A@ \divide\count@\dimen@ \advance\count@7 %
 \else \count@=15 \fi\fi\fi
 \R@=\A@
 \ifnum\count@<8 \multiply\R@\count@ \advance\R@\A@
 \else\ifnum\count@<12 \multiply\R@\count@ \multiply\R@\tw@ \advance\R@-6\A@
 \else\ifnum\count@<16 \multiply\R@\count@ \multiply\R@ 4 \advance\R@-28\A@
```

```
\else \multiply\R@ 32 \fi\fi\fi
\edef\@tmp{\endgroup \R@=\the\R@ \count@=\the\count@}\@tmp}
```

Partial circle segments with ⟨orient⟩ation are the part of the full circle that starts with a tangent vector in the direction of the first ⟨diag⟩onal (see note 1.4l) and ends with a tangent vector in the direction of the other ⟨diag⟩onal after a clockwise (for `_`) or anticlockwise (for `^`) turn, *e.g.*,

`\xy*\cir<4pt>{l^r}\endxy`	typesets	“ ”
`\xy*\cir<4pt>{l_r}\endxy`	—	“ ”
`\xy*\cir<4pt>{dl^u}\endxy`	—	“ ”
`\xy*\cir<4pt>{dl_u}\endxy`	—	“ ”
`\xy*+{M}*\cir{dr_ur}\endxy`	—	“M”

If the same ⟨diag⟩ is given twice then nothing is typeset, *e.g.*,

`\xy*\cir<4pt>{u^u}\endxy` typesets “ ”

Special care is taken to setup the ⟨diag⟩onal defaults:

- After `^` the default is the diagonal 90° anticlockwise from the one before the `^`.
- After `_` the default is the diagonal 90° clockwise from the one before the `_`.

The ⟨diag⟩ before `^` or `_` is required for `\cir` ⟨objects⟩.

Exercise 1.15: Typeset the following shaded circle with radius `5pt`:

These two macros implement the defaults and setup of *lo* and *hi* for anticlockwise and clockwise segments. Here is what they set:

⟨cir⟩	*lo*	*hi*	*test*(s)
d_1^$d_2, d_1 \le d_2$	$d_1 -_8 1$	$d_2 -_8 1$	$lo \le s \wedge s < hi$
d_1^$d_2, d_1 > d_2$	$d_2 -_8 1$	$d_1 -_8 1$	$s < lo \vee hi \le s$
d_1_$d_2, d_1 < d_2$	$d_1 +_8 3$	$d_2 +_8 3$	$s < lo \vee hi \le s$
d_1_$d_2, d_1 \ge d_2$	$d_2 +_8 3$	$d_1 +_8 3$	$lo \le s \wedge s < hi$

where $+_8$ and $-_8$ are $+$ and $-$ modulo 8; d_1 and d_2 are in `\count@@` and `\count@`, respectively.

```
\xydef@\CIRacw@{\count@@=\CIRin@@ \count@=\CIRout@@
 \ifnum\count@=8 \count@=\count@@
  \ifnum\count@<6 \advance\count@\tw@ \else \advance\count@-6 \fi \fi
 \ifnum\count@@<\@ne \advance\count@@7 \else \advance\count@@\m@ne \fi
 \ifnum\count@<\@ne \advance\count@7 \else \advance\count@\m@ne \fi
 \ifnum\count@@>\count@ \let\CIRtest@@=\CIRtest@outside
  \edef\CIRlo@@{\the\count@}\edef\CIRhi@@{\the\count@@}%
 \else \let\CIRtest@@=\CIRtest@inside
  \edef\CIRlo@@{\the\count@@}\edef\CIRhi@@{\the\count@}%
 \fi}
```

```
\xydef@\CIRcw@{\count@@=\CIRin@@ \count@=\CIRout@@
 \ifnum\count@=8 \count@=\count@@
```

```
  \ifnum\count@>\@ne \advance\count@-\tw@ \else \advance\count@6 \fi \fi
 \ifnum\count@@<5 \advance\count@@\thr@@ \else \advance\count@@-5 \fi
 \ifnum\count@<5 \advance\count@\thr@@ \else \advance\count@-5 \fi
 \ifnum\count@@<\count@ \let\CIRtest@@=\CIRtest@outside
  \edef\CIRlo@@{\the\count@@}\edef\CIRhi@@{\the\count@}%
 \else \let\CIRtest@@=\CIRtest@inside
  \edef\CIRlo@@{\the\count@}\edef\CIRhi@@{\the\count@@}%
 \fi}
```

```
\xydef@\CIRtest@inside#1#2{\let\next@=\relax
 \ifnum\CIRlo@@>#1\else \ifnum#1<\CIRhi@@\DN@{#2}\fi\fi \next@}
```

```
\xydef@\CIRtest@outside#1#2{\let\next@=\relax
 \ifnum\CIRlo@@>#1\DN@{#2}\else \ifnum#1<\CIRhi@@\else\DN@{#2}\fi\fi \next@}
```

1.6.3 Text

```
\message{text;}
```

Text in pictures is supported through the ⟨object⟩ construction

---

`\txt` ⟨width⟩ ⟨style⟩ `{`⟨text⟩`}`

---

that builds an object containing ⟨text⟩ typeset to ⟨width⟩ using ⟨style⟩; in ⟨text⟩ `\\` can be used as an explicit line break; all lines will be centered. ⟨style⟩ should either be a font command or some other stuff to do for each line of the ⟨text⟩ and ⟨width⟩ should be either `<`⟨dimen⟩`>` or ⟨empty⟩.

The code just parses the ⟨width⟩ defaulting it to `<\maxdimen>` which is recognised as 'free form'.

```
\xydef@\txt{\hbox\bgroup \xyFN@\txt@}
```

```
\xydef@\txt@{%
 \addLT@\ifx\next \addGT@{\addLT@\DN@##1}{\A@=##1\txt@i}%
 \else \DN@{\A@=\maxdimen \txt@i}\fi \next@}
```

```
\xydef@\txt@i#1#{%
 \setboxz@h{#1\mathstrut}\dimen@=\ht\z@ \advance\dimen@\dp\z@
 \baselineskip=1.1\dimen@ \lineskip=.2\dimen@ \lineskiplimit=\lineskip
 \def\txtline@@##1{\txtline@{#1}{##1}}\object@\txt@ii}
```

```
\xylet@\txtline@@=\eat@
\xydef@\txtline@#1#2{\relax\setboxz@h{#1{\ignorespaces #2\unskip}}%
 \ifdim\A@<\wdz@ \setboxz@h{\hsize=\A@
   \leftskip=0pt plus4em \rightskip=\leftskip
   \parfillskip=0pt \parindent=0pt %
   \spaceskip=.3333em \xspaceskip=.5em %
   \pretolerance=9999 \tolerance=9999 %
   \hyphenpenalty=9999 \exhyphenpenalty=9999 %
   \vbox{#1\noindent\ignorespaces #2\unskip}}%
 \else\ifdim\A@<\maxdimen\setboxz@h to\A@{\hfil\boxz@\hfil}\fi\fi
 \boxz@}
```

```
\xydef@\txt@ii#1{\vbox{%
  \let\\=\cr
  \tabskip=\z@skip \halign{\relax\hfil\txtline@@{##}\hfil\cr\leavevmode#1\crcr}}}
```

1.7 XY-pic options

Note: LaTeX 2_ε users should also consult the paragraph on "xy.sty" in §1.1.1.

1.7.1 Loading

XY-pic is provided with a growing number of options supporting specialised drawing tasks as well as exotic output devices with special graphic features. These should all be loaded using this uniform interface in order to ensure that the XY-pic environment is properly set up while reading the option.

`\xyoption {` ⟨option⟩ `}`
`\xyrequire {` ⟨option⟩ `}`

`\xyoption` will cause the loading of an XY-pic option file which can have one of several names. These are tried in sequence: `xy`⟨option⟩`.tex`, `xy`⟨option⟩`.doc`, `xy`⟨short⟩`.tex`, and `xy`⟨short⟩`.doc`, where ⟨short⟩ is ⟨option⟩ truncated to 6 (six) characters to conform with the TWG-TDS [18].

`\xyrequire` is the same except it is ignored if an option with the same name is already present (thus does not check the version etc.).

```
\message{options;}

\xylet@\xyoption@@=\relax

\xydef@\xyoption#1{\DN@{#1}\edef\next@{\codeof\next@}%
 \csname xyeveryrequest@\next@ @\endcsname
 \xyinputorelse@{xy#1}{\xyoption@truncated#1@@@@@@@@@,#1@}%
 \def\xyoption@@{#1}\edef\xyoption@@{\codeof\xyoption@@}
 \expandafter\let\expandafter\next@\csname xy\xyoption@@ version\endcsname
 \expandafter\let\csname xy\xyoption@@ loaded\endcsname=\next@
 \runxywith@ \ignorespaces}

\xydef@\xyoption@truncated#1#2#3#4#5#6#7@#8,#9@{%
 \DN@{#7}\ifx\next@\empty \DN@##1##2{##2}\else\let\next@=\xyinputorelse@ \fi
 \next@{xy#1#2#3#4#5#6}%
  {\DN@{#1#2#3#4#5#6}\edef\next@{\codeof\next@}%
   \xyerror@{No `\next@' option}{%
Your \xyoption{#9} request could not be granted: the required^^J%
file `xy#9.tex' could not be located. Please make sure that it is^^J%
properly installed before continuing.}}}

\xydef@\xyrequire#1{\DN@{#1}\edef\next@{\codeof\next@}%
 \csname xyeveryrequest@\next@ @\endcsname
 \expandafter\let\expandafter\next@\csname xy\codeof\next@ loaded\endcsname
 \ifx \next@\relax \DN@{\xyoption{#1}}\else \DN@{\ignorespaces}\fi \next@}
```

Sometimes some declarations of an option or header file or whatever only makes sense after some particular other option is loaded. In that case the code should be wrapped in the special command

`\xywithoption {` ⟨option⟩ `} {` ⟨code⟩ `}`

which indicates that if the ⟨option⟩ is already loaded then ⟨code⟩ should be executed now, otherwise it should be saved and if ⟨option⟩ ever gets loaded then ⟨code⟩ should be executed afterwards. **Note:** The ⟨code⟩ should allow more than one execution; it is saved with the catcodes at the time of the `\xywithoption` command.

```
\xylet@\xywith@@=\empty
```

```
\xydef@\runxywith@{\let\xywithdo@@=\xywithtest@ \xywith@@}
```

```
\xydef@\xywithoption#1#2{\DN@{#1}\edef\next@{\codeof\next@}%
 \expandafter\let\expandafter\nextii@\csname xy\next@ loaded\endcsname
 \ifx \nextii@\relax
  \expandafter\def\expandafter\xywith@@\expandafter{\xywith@@
   \xywithdo@@{#1}{#2}}%
 \else \expandafter\xywithrun@\expandafter{\next@}{#2}\fi}
```

```
\xydef@\xywithtest@#1#2{\DN@{#1}\edef\next@{\codeof\next@}%
 \ifx\next@\xyoption@@ \expandafter\xywithrun@\expandafter{\next@}{#2}\fi}
```

```
\xydef@\xywithrun@#1#2{\csname xyeverywithoption@#1@\endcsname #2}
```

Finally, it is possible to declare ⟨code⟩ as some commands to be executed before every actual execution of `\xywithoption{`⟨option⟩`}{...}`, and similarly ⟨code⟩ to be executed before every `\xyoption{`⟨option⟩`}` and `\xyrequire{`⟨option⟩`}` (collectively called 'requests')::

`\xyeverywithoption {` ⟨option⟩ `} {` ⟨code⟩ `}`
`\xyeveryrequest {` ⟨option⟩ `} {` ⟨code⟩ `}`

This is most often used by an option to activate some hook every time it is requested itself.

These last two use the usual expansion trickery to define or append to the control sequences named `\xyeveryrequest@`⟨option⟩`@` and `\xyeverywithoption@`⟨option⟩`@` used in the definitions above.

```
\xydef@\xyevery@#1#2#3{\DN@{#2}\edef\next@{\codeof\next@}%
 \expandafter\ifx\csname xyevery#1@\next@ @\endcsname\relax
  \expandafter\let\csname xyevery#1@\next@ @\endcsname=\empty \fi
 \DNii@##1{\expandafter\def\expandafter##1\expandafter{##1#3}}%
 \expandafter\nextii@\csname xyevery#1@\next@ @\endcsname}
```

```
\xydef@\xyeveryrequest{\xyevery@{request}}
\xydef@\xyeverywithoption{\xyevery@{withoption}}
```

1.7.2 Option file format

Option files must have the following structure:

> `%%` ⟨identification⟩
> `%%` ⟨copyright, *etc.*⟩
>
> `\ifx\xyloaded\undefined \input xy \fi`
>
> `\xyprovide{`⟨option⟩`}{`⟨name⟩`}{`⟨version⟩`}%`
> `{`⟨author⟩`}{`⟨email⟩`}{`⟨address⟩`}`
>
> ⟨body of the option⟩
>
> `\xyendinput`

The 6 arguments to `\xyprovide` should contain the following:

⟨option⟩ Option load name as used in the `\xyoption` command. This should be safe and distinguishable for any operating system and is thus limited to characters chosen among the lowercase letters (`a`–`z`), digits (`0`–`9`), and dash (`-`), and all options should be uniquely identifiable by the first 6 (six) characters only.

⟨name⟩ Descriptive name for the option.

⟨version⟩ Identification of the version of the option.

⟨author⟩ The name(s) of the author(s).

⟨email⟩ The electronic mail address(es) of the author(s) *or* the affiliation if no email is available.

⟨address⟩ The postal address(es) of the author(s).

This information is used not only to print a nice banner but also to (1) silently skip loading if the same version was preloaded and (2) print an error message if a different version was preloaded.

The `\xyprovide` command checks that the option is not already loaded and that the loaded version is the same as the preloaded one by checking the existence and contents of the macro `\xy`⟨option⟩`loaded`. Finally it calls `\xycatcodes` such that the option internals are loaded in 'TEX programming mode'. `\xyendinput` undoes this.

```
\xydef@\xyprovide#1#2#3#4#5#6{%
 \def\xyoption@@{#1}\edef\xyoption@@{\codeof\xyoption@@}\edef\next@{#3}%
 \message{ Xy-pic option: #2 v.\next@}%
 \expandafter\let\expandafter\nextii@\csname xy\xyoption@@ loaded\endcsname
 \ifx \next@\nextii@ \message{not reloaded}\endinput
 \else
  \ifx \nextii@\relax\else \xyerror@{Option `\xyoption@@' version mismatch}{%
You previously loaded, or the format has preloaded, a different version^^J%
of this option.  Just continue to try to load this version instead (and^^J%
be prepared for a lot of warnings about redefinitions).}%
  \fi
  \expandafter\let\csname xy\xyoption@@ version\endcsname=\next@
  \expandafter\let\expandafter\xyenddocmode@\csname DOCMODE\endcsname
  \expandafter\let\csname DOCMODE\endcsname\xyprovidedocmode@
  \xycatcodes
 \fi \ignorespaces}
```

```
\xydef@\xyendinput{\expandafter\let\csname DOCMODE\endcsname=\xyenddocmode@
 \message{loaded}\xyuncatcodes\endinput}
```

The above is futher complicated because an option may loaded in its `DOCMODE` form (ending in `.doc`). So we make sure that the mode that skips documentation, defined in the `xy.tex` file header, is active now.

```
\expandafter\xylet@\expandafter\xyprovidedocmode@\csname DOCMODE\endcsname
\xylet@\xyenddocmode@=\relax
```

The 'dummy' option described in §3.2 is a minimal option using the above features. It uses the special `DOCMODE` format to include its own documentation for this document (like all official XY-pic options) but this is not a requirement.

1.7.3 Driver options

The ⟨driver⟩ options described in chapter **??** require special attention because each driver can support several extension options, and it is sometimes desirable to change ⟨driver⟩ or even mix the support provided by several.[6]

[6]The kernel support described here is based on the (now defunct) `xydriver` include file by Ross Moore.

A ⟨driver⟩ option is loaded as other options with `\xyoption{`⟨driver⟩`}` (or through LaTeX 2_ε class or package options as described in §1.1.1). The special thing about a ⟨driver⟩ is that loading it simply declares the name of it, establishes what extensions it will support, and selects it temporarily. Thus the special capabilities of the driver will only be exploited in the produced DVI file if some of these extensions are also loaded and if the driver is still selected when output is produced. Generally, the order in which the options are loaded is immaterial. (Known exceptions affect only internal processing and are not visible to the user in terms of language and expected output.) In particular one driver can be preloaded in a format and a different one used for a particular document.

The following declarations control this:

`\UseSingleDriver`	forces one driver only
`\MultipleDrivers`	allows multiple drivers
`\xyReloadDrivers`	resets driver information

The first command restores the default behaviour: that ony one ⟨driver⟩ is allowed, *i.e.*, each loading of a ⟨driver⟩ option cancels the previous. The second allows consecutive loading of drivers such that when loading a ⟨driver⟩ only the extensions actually supported are selected, leaving other extensions supported by previously selected drivers untouched. Beware that this can be used to create DVI files that cannot be processed by any actual DVI driver program!

The last command is sometimes required to reset the XY-pic ⟨driver⟩ information to a sane state, for example, after having applied one of the other two in the middle of a document, or when using simple formats like plain TeX that do not have a clearly distinguished preamble.

Driver database. The main structure, the list of 'loaded' and 'selected' ⟨driver⟩s (ordered by age), is implemented as traditional TeX '`\do`-lists' (applications of `\do{`⟨driver⟩`}` to each element).

```
\xydef@\xydriversloaded@@{\do{unload}}
\xylet@\xydriversselected@@=\empty
```

This is used to provide entry points into a secondary structure that lists for each ⟨driver⟩ the extensions supported by that driver. As should be apparent from the above, only the `unload` ⟨driver⟩ is present initially: it represents the implementation technique used for the extension when no ⟨driver⟩ is selected. Every extension must add an entry which reestablishes the default implementation of the extension:

```
\xylet@\xydriver@unload@support@@=\empty
```

All ⟨driver⟩s will have a similar structure named `\xydriver@`⟨driver⟩`@support@@` which contains the extensions supported by that particular ⟨driver⟩: the 'signature' of each driver as `\doii`-list with two arguments (hence the `ii` ☺): the extension and the command to install the ⟨driver⟩-extension combination.

The two selection principles differ significantly. When only a single ⟨driver⟩ is selectable then all extensions need to be unloaded *before* the selection. When multiple ⟨driver⟩s are allowed then only the added support needs to be executed.

```
\xydef@\selectdriver@#1{\DN@{#1}\edef\next@{\codeof\next@}%
 \expandafter\selectdriver@@\expandafter{\next@}}

\xydef@\selectdriver@single#1{\xysetup@@{\def\xydriversselected@@{\do{#1}}%
 \xyLoadDrivers@}\let\selectdriver@@=\changedriver@single}

\xydef@\changedriver@single#1{\xysetup@@{\def\xydriversselected@@{\do{#1}}%
 \xyReloadDrivers@}}

\xydef@\selectdriver@multiple#1{\expandafter\xysetup@@\expandafter{%
```

```
\expandafter\def\expandafter\xydriversselected@@\expandafter{%
 \xydriversselected@@ \do{#1}}%
\let\do=\activatedriver@ \xydriversselected@@}}

\xydef@\UseSingleDriver{\let\selectdriver@@=\selectdriver@single}
\xydef@\MultipleDrivers{\let\selectdriver@@=\selectdriver@multiple}

\xydef@\xyLoadDrivers@{\let\do=\activatedriver@ \xydriversselected@@}
\xydef@\xyReloadDrivers@{\activatedriver@{unload}\xyLoadDrivers@}
\xylet@\xyReloadDrivers=\xyReloadDrivers@

\xydef@\activatedriver@#1{%
 \let\doii=\activatedriversupport@ \csname xydriver@#1@support@@\endcsname}

\xydef@\activatedriversupport@#1#2{%
 \expandafter\ifx\csname xy#1loaded\endcsname\relax\DN@{}%
 \else\DN@{#2}\fi \next@}

\UseSingleDriver
```

Finally we define the following to get complete independence of load sequence: at every loading of an extension option known to be supported, we run through the list of presently selected drivers to activate any support for it, using the following:

```
\xydef@\xyselectoptionsupport@{\let\do=\activatedriver@ \xydriversselected@@}
```

As the above suggests it sometimes makes sense to load ⟨driver⟩s in the actual textual part of a document, however, it is recommended that only drivers *also* loaded in the preamble are reloaded later, and that `\xyReloadDrivers` is used when there is doubt about the state of affairs. In case of confusion the special command `\xyShowDrivers` will list all the presently supported and selected driver-extension pairs to the TEX log.

```
\xydef@\xyShowDrivers{{\W@{Loaded:}\let\do=\doShow \xydriversloaded@@
  \let\next@=\empty
  \def\do##1{\ifx\next@\empty \DN@{##1}%
    \else \expandafter\DN@\expandafter{\next@, ##1}\fi}%
  \xydriversselected@@\W@{Selected: \next@.}}}

\def\doShow#1{\let\next@=\empty
 \def\doii##1##2{\ifx\next@\empty \DN@{##1}%
  \else \expandafter\DN@\expandafter{\next@, ##1}\fi}%
 \csname xydriver@#1@support@@\endcsname
 \W@{ <driver> #1 supports \next@.}}
```

Driver internals. A ⟨driver⟩ option files must have the following structure:

```
%% ⟨identification⟩
%% ⟨copyright, etc.⟩
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{⟨option⟩}{⟨name⟩}{⟨version⟩}%
        {⟨author⟩}{⟨email⟩}{⟨address⟩}
\newdriver{%
\xyaddsupport{⟨extension⟩}⟨cs⟩
⋮
}
```

⟨body of the ⟨driver⟩ option⟩

`\xyendinput`

There should be an `\xyaddsupport` command for each supported ⟨extension⟩ which specicifies the ⟨cs⟩ (control sequence) with which the indicated support is *activated*.

The `\newdriver` command first establishes the 'support' list for the ⟨driver⟩ (running the argument), then adds the ⟨driver⟩ to the 'loaded' list, and finally selects it now and for every subsequent request of this ⟨driver⟩ option.

It is important that the ⟨cs⟩ that *activates* each driver/extension combination only does *rebinding of hooks*. In order to see what hooks are available for an extension, look for declarations of the form

---

`\xyaddunsupport {`⟨extension⟩`}` ⟨cs⟩

---

and see what rebindings that ⟨cs⟩ does: those are the hokks. *because* they are the ones that will be safely reestablished! This should be adherede to because both activation and deactivation ⟨cs⟩s may be executed many times.

The procedure is as follows: (1.6809) add the driver and clear its support control sequence unless it is already defined, (1.6815) execute the support commands, and (1.6816) setup to select it after loading.

```
6808 \xydef@\newdriver#1{%
6809  \def\nextiii@##1{
6810   \expandafter\def\expandafter\xydriversloaded@@
6811    \expandafter{\xydriversloaded@@\do{##1}}%
6812    \expandafter\let\csname xydriver@\xyoption@@ @support@@\endcsname=\empty}%
6813  \def\do##1{\DNii@{##1}\ifx\xyoption@@\nextii@ \let\nextiii@=\eat@ \fi}%
6814  \xydriversloaded@@ \expandafter\nextiii@\expandafter{\xyoption@@}%
6815  #1\relax
6816  \DN@##1{\xywithoption{##1}{
6817    \selectdriver@{##1}\xyeveryrequest{##1}{\selectdriver@{##1}}}}%
6818  \expandafter\next@\expandafter{\xyoption@@}\ignorespaces}
```

The support commands merely add to the support `\do`-lists which in turn (1.6845) adds `\doii{`⟨extension⟩`}{...}` to the `\xydriver@`⟨driver⟩`@support@@` control sequence and adds a check for when to apply it every time that ⟨extension⟩ is requested!

```
6828 \xydef@\xyaddsupport{\expandafter\xyadddriversupport@\expandafter{\xyoption@@}}
6829 \xydef@\xyaddunsupport{\xyadddriversupport@{unload}}
```

```
6831 \xydef@\xyadddriversupport@#1#2#3{%
6832  \DNii@{#1}\edef\nextii@{\codeof\nextii@}%
6833  \def\nextiii@{#2}\edef\nextiii@{\codeof\nextiii@}%
6834 %
6835 % CHECK that the <driver> is loaded!
6836 \def\next{\xybug@{<driver> \nextii@\space not loaded}}%
6837 \def\do##1{\DN@{##1}\edef\next@{\codeof\next@}%
6838  \ifx\next@\nextii@ \let\next=\relax \fi}\xydriversloaded@@ \next
6839 %
6840 % CHECK that the specified <extension> is not already supported!
6841 \def\next{\xybug@{<driver> \nextii@\space already supports \nextiii@}}%
6842 \def\doii##1##2{\DN@{##1}\edef\next@{\codeof\next@}%
6843  \ifx\next@\nextiii@ \let\next=\relax \fi}\xydriversloaded@@ \next
6844 %
```

```
\DN@##1##2{\ifx##1\relax \let##1=\empty \fi
 \expandafter\def\expandafter##1\expandafter{##1\doii{##2}{#3}}}%
\expandafter\expandafter\expandafter\next@
 \expandafter\expandafter\csname xydriver@\nextii@ @support@@\endcsname
  \expandafter{\nextiii@}%
\DN@##1{\expandafter\xyeverywithoption\expandafter{\nextiii@}{%
 \xytestforsupport@{##1}}}%
\expandafter\next@\expandafter{\nextii@}\xydriversloaded@@
\ignorespaces}
```

```
\xydef@\xytestforsupport@#1{\def\do##1{\DN@{##1}\DNii@{#1}%
 \ifx\next@\nextii@ \expandafter\xyselectoptionsupport@ \fi}}
```

Driver related messages. Most extensions will print a warning when a capability is used which is not supported by the presently loaded ⟨driver⟩. Such messages are only printed once, however, (for some formats they are repeated at the end). Similarly, when the support of an extension that exploits a particular ⟨driver⟩ is used a warning message will be issued that the DVI file is not portable.

This is implemented by defining a control sequence for each such warning and repeat it at the end. Warnings should be issued using one of

---

`\xyununsupportwarning@` {⟨extension⟩} {⟨message⟩}
`\xyunsupportwarning@` {⟨driver⟩} {⟨extension⟩}

```
\xynew@{if}\ifunsupportwarnings@
```

```
\xydef@\xyunsupportwarning@#1#2{{%
 \DN@{#1}\edef\next@{\codeof\next@}
 \expandafter\ifx\csname xywarn@unload@\next@ @\endcsname\relax
  \expandafter\gdef\csname xywarn@unload@\next@ @\endcsname{}%
  \ifunsupportwarnings@ \xyclosedown@@\xyunsupportwarnings@@ \fi
  \global\unsupportwarnings@true
  \expandafter\gdef\expandafter\xyunsupportwarnings@@\expandafter{%
   \xyunsupportwarnings@@ \W@{ #2.}}%
  \W@{}%
  \W@{Xy-pic Warning: `\next@' reproduction is NOT EXACT\xytracelineno@:}%
  \W@{ #2.}%
  \W@{}%
 \fi}}
```

```
\xydef@\xyunsupportwarnings@@{\W@{}%
 \W@{Xy-pic Warning: The produced DVI file is NOT EXACT:}}
```

```
\xynew@{if}\ifsupportwarnings@
```

```
\xydef@\xysupportwarning@#1#2{{%
 \DN@{#1}\edef\next@{\codeof\next@}\DNii@{#2}\edef\nextii@{\codeof\nextii@}%
 \expandafter\ifx\csname xywarn@\next@ @\nextii@ @\endcsname\relax
  \expandafter\gdef\csname xywarn@\next@ @\nextii@ @\endcsname{}%
  \ifsupportwarnings@ \xyclosedown@@\xysupportwarnings@@ \fi
  \global\supportwarnings@true
  \expandafter\gdef\expandafter\xysupportwarnings@@\expandafter{%
   \xysupportwarnings@@ \driverextensioncomplain@{#1}{#2}}%
```

```
  \W@{}%
  \W@{Xy-pic Warning: The produced DVI file is NOT PORTABLE\xytracelineno@:}%
  \driverextensioncomplain@{#1}{#2}%
  \W@{}%
 \fi}}
```

```
\xydef@\xysupportwarnings@@{%
 \W@{Xy-pic Warning: The produced DVI file is NOT PORTABLE:}}
```

```
\xydef@\driverextensioncomplain@#1#2{%
 \DN@{#1}\edef\next@{\codeof\next@}\DNii@{#2}\edef\nextii@{\codeof\nextii@}%
 \W@{ It contains <driver> `\next@' \string\special s for the `\nextii@'
        extension.}}
```

Backwards compatibility...

```
\xydef@\dvimessage@#1#2{\xysupportwarning@{#1}{#2}}
```

```
\xynew@{if}\ifxydrivernoloads@
\xydef@\xydrivernoload@#1{\xyerror@{Could not load xy#1.tex}{}%
 \xydrivernoloads@true}
```

```
\xydef@\xydriverfail@#1{\xyerror@{Sorry, #1 not supported.}{}}
```

```
\xydef@\xyunload@#1{\xywarning@{Unloading #1.}}
```

1.8 Algorithms

This section presents the more complicated algorithms used in XY-pic.

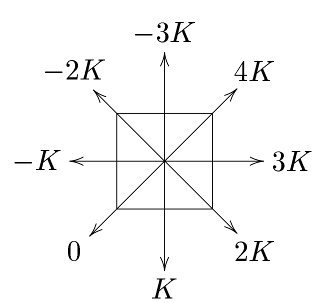
1.8.1 Directions

```
\message{algorithms: directions,}
```

The *direction state* is described by the following parameters:

Direction	'angle' of the direction on $]-4K \dots 4K]$ unit square
dX,*dY*	the vector $c - p$
sdX,*sdY*	sign of dX and dY
`\K@dXdY`,`\K@dYdX`	quotients $K\frac{dX}{dY}$ and $K\frac{dY}{dX}$ (as dimensions in sp)
DirectionChar,*SemiDirectionChar*	`\chardefs` for directional and semidirectional fonts
cosDirection,*sinDirection*	factors in the range $]-1 \dots 1]$ corresponding to cos and sin of Direction

where the "$]-4K \dots 4K]$ unit square" has the following angles:



where the intermediate K angle in each interval correspond to equidistant points on the unit square. Thus only for $n \in \{-3, 2, 1, 0, 1, 2, 3, 4\}$ the angle of direction $n \times K$ is exactly $n \times 45^\circ - 135^\circ$ (0° being the direction straight right).

As usual `\DirectionfromtheDirection@` expands to code setting the current direction.

```
6997 \xydef@\DirectionfromtheDirection@{\noexpand\Direction=\the\Direction
6998  \noexpand\d@X=\the\d@X \noexpand\d@Y=\the\d@Y
6999  \def\noexpand\sd@X{\sd@X}\def\noexpand\sd@Y{\sd@Y}%
7000  \noexpand\K@dXdY=\the\K@dXdY \noexpand\K@dYdX=\the\K@dYdX
7001  \chardef\noexpand\DirectionChar=\the\DirectionChar
7002  \chardef\noexpand\SemiDirectionChar=\the\SemiDirectionChar
7003  \def\noexpand\cosDirection{\cosDirection}%
7004  \def\noexpand\sinDirection{\sinDirection}%
7005  \noexpand\resetupDirection@}
```

The actual direction computation is done using `\setupDirection@`.

Procedure: Is really not so complicated. [1.7024] dX and dY are computed from $c - p$ and we skip if the current setting is based on these (this is stored in the internal `\Directiontest@@` method); [1.7027] if the direction is one of the principal ones then proceed with an optimised special case for those; otherwise proceed with the generic code.

```
7021 \xydef@\Directiontest@@#1#2{#2}

7023 \xydef@\setupDirection@{%
7024  \d@X=\X@c\advance\d@X-\X@p \d@Y=\Y@c\advance\d@Y-\Y@p
7025  \Directiontest@@\relax\setupDirection@i}

7027 \xydef@\setupDirection@i{\DN@{\setupDirection@ii}
7028  \ifdim\d@X=\d@Y
7029   \ifdim\zz@\d@Y \DN@{}%
7030   \else\ifdim\d@X<\z@ \DN@{\dlDirection@{-1.4142\d@X}}%
7031   \else \DN@{\urDirection@{1.4142\d@X}}\fi\fi
7032  \else\ifdim\d@X<\d@Y
7033   \ifdim\zz@\d@X \DN@{\uDirection@\d@Y}%
7034   \else\ifdim\zz@\d@Y \DN@{\lDirection@{-\d@X}}%
7035   \else\ifdim-\d@X=\d@Y \DN@{\ulDirection@{-1.4142\d@X}}\fi\fi\fi
7036  \else
7037   \ifdim\zz@\d@X \DN@{\dDirection@{-\d@Y}}%
7038   \else\ifdim\zz@\d@Y \DN@{\rDirection@\d@X}%
7039   \else\ifdim\d@X=-\d@Y \DN@{\drDirection@{1.4142\d@X}}\fi\fi\fi
7040  \fi\fi \next@}
```

The procedures for the special ⟨diag⟩onal cases are summarised in this table:

⟨diag⟩onal	*Direction*	cos(*Direction*), sin(*Direction*)	*sign*(dX, dY)	*Char*	*Semi*
`dl`	0	$-\sqrt{\frac{1}{2}},\ -\sqrt{\frac{1}{2}}$	$-,-$	127	127
`d`	K	$0,\ -1$	$+,-$	15	31
`dr`	$2K$	$\sqrt{\frac{1}{2}},\ -\sqrt{\frac{1}{2}}$	$+,-$	31	63
`r`	$3K$	$1,\ 0$	$+,+$	47	95
`ur`	$4K$	$\sqrt{\frac{1}{2}},\ \sqrt{\frac{1}{2}}$	$+,+$	63	127
`u`	$-3K$	$0,\ 1$	$+,+$	79	31
`ul`	$-2K$	$-\sqrt{\frac{1}{2}},\ \sqrt{\frac{1}{2}}$	$-,+$	95	63
`l`	$-K$	$-1,\ 0$	$-,+$	111	95

In each case the argument is used as the unit circle, *i.e.*, non-zero of dX and dY, and $\texttt{<}K\frac{dX}{dY}\texttt{,}K\frac{dY}{dX}\texttt{>} := KK \times \texttt{<}dX\texttt{,}dY\texttt{>}\ldots$

```
\xydef@\dlDirection@{\Direction=\z@
 \def\cosDirection{-.7071}\def\sinDirection{-.7071}\def\sd@X{-}\def\sd@Y{-}%
 \chardef\DirectionChar=127\relax\chardef\SemiDirectionChar=127\relax
 \K@dXdY=1\K@ \K@dYdX=1\K@ \fixedDirection@}
```

```
\xydef@\dDirection@{\Direction=\K@
 \def\cosDirection{0}\def\sinDirection{-1}\def\sd@Y{-}%\def\sd@X{+}
 \edef\sd@X{\ifdim\X@c<\X@p-\else+\fi}%
 \chardef\DirectionChar=15\relax\chardef\SemiDirectionChar=31\relax
 \K@dXdY=\z@ \K@dYdX=\KK@\K@ \fixedDirection@}
```

```
\xydef@\drDirection@{\dimen@ii=2\K@ \Direction=\dimen@ii
 \def\cosDirection{+.7071}\def\sinDirection{-.7071}\def\sd@X{+}\def\sd@Y{-}%
 \chardef\DirectionChar=31\relax\chardef\SemiDirectionChar=63\relax
 \K@dXdY=-1\K@ \K@dYdX=-1\K@ \fixedDirection@}
```

```
\xydef@\rDirection@{\dimen@ii=3\K@ \Direction=\dimen@ii
 \def\cosDirection{+1}\def\sinDirection{0}\def\sd@X{+}%\def\sd@Y{+}%
 \edef\sd@Y{\ifdim\Y@c<\Y@p-\else+\fi}%
 \chardef\DirectionChar=47\relax\chardef\SemiDirectionChar=95\relax
 \K@dXdY=\KK@\K@ \K@dYdX=\z@ \fixedDirection@}
```

```
\xydef@\urDirection@{\dimen@ii=4\K@ \Direction=\dimen@ii
 \def\cosDirection{+.7071}\def\sinDirection{+.7071}\def\sd@X{+}\def\sd@Y{+}%
 \chardef\DirectionChar=63\relax\chardef\SemiDirectionChar=127\relax
 \K@dXdY=1\K@ \K@dYdX=1\K@ \fixedDirection@}
```

```
\xydef@\uDirection@{\dimen@ii=-3\K@ \Direction=\dimen@ii
 \def\cosDirection{0}\def\sinDirection{+1}\def\sd@Y{+}%\def\sd@X{+}
 \edef\sd@X{\ifdim\X@c<\X@p-\else+\fi}%
 \chardef\DirectionChar=79\relax\chardef\SemiDirectionChar=31\relax
 \K@dXdY=\z@ \K@dYdX=\KK@\K@ \fixedDirection@}
```

```
\xydef@\ulDirection@{\dimen@ii=-2\K@ \Direction=\dimen@ii
 \def\cosDirection{-.7071}\def\sinDirection{+.7071}\def\sd@X{-}\def\sd@Y{+}%
 \chardef\DirectionChar=95\relax\chardef\SemiDirectionChar=63\relax
 \K@dXdY=-1\K@ \K@dYdX=-1\K@ \fixedDirection@}
```

```
\xydef@\lDirection@{\Direction=-\K@
```

```
7107 \def\cosDirection{-1}\def\sinDirection{0}\def\sd@X{-}%\def\sd@Y{+}%
7108 \edef\sd@Y{\ifdim\Y@c<\Y@p-\else+\fi}%
7109 \chardef\DirectionChar=111\relax\chardef\SemiDirectionChar=95\relax
7110 \K@dXdY=\KK@\K@ \K@dYdX=\z@ \fixedDirection@}

7112 \xydef@\fixedDirection@#1{\dimen@ii=#1\relax
7113 \d@X=\cosDirection\dimen@ii \d@Y=\sinDirection\dimen@ii
7114 \resetupDirection@}
```

Here is the procedure for the generic code.

1.7171 Make sign variables and slopes: $sdX := sign(dX)$, $sdY := sign(dY)$, $K\frac{dX}{dY} := sdX \times sdY \times \lfloor |KK \times dX| \,/\, |dY/KK| \rfloor$, and $K\frac{dY}{dX} := sdX \times sdY \times \lfloor |KK \times dY| \,/\, |dX/KK| \rfloor$, where the somewhat exotic computation method is used to ensure that the 'native' floor function provided by TeX `\divide` can be used (it only acts predictably for positive numbers), that overflow is avoided even for large dX, dY, and that it is reasonable to use the convention of `\quotient@` that division by zero is like multiplying with one... Also takes care not to multiply too big dimensions with each other.

1.7188 If $-K \le K\frac{dX}{dY} \le K$ then the direction is mostly up or down: if $dY < 0$ [down, 1.7189]: $Direction := -K\frac{dX}{dY} + 1K$; else [up, 1.7191]: $Direction := -K\frac{dX}{dY} - 3K$.

1.7194 If $-K < K\frac{dY}{dX} < K$ then direction is mostly left or right: if $dX < 0$ [left, 1.7195]: $Direction := K\frac{dY}{dX} - K$; else [right, 1.7197]: $Direction := K\frac{dY}{dX} + 3K$.

1.7222 Compute character codes for direction and semidirection fonts. [1.7223]: $DirectionChar := (8K + Direction + K/32)$ div $(K/16) - 1$; while $DirectionChar > 127 : DirectionChar - := 128$. [1.7226]: $SemiDirectionChar := (8K + Direction + K/64)$ div $(K/32) - 1$; while $SemiDirectionChar > 127 : SemiDirectionChar - := 128$. In both cases the $8K$ are added to ensure that TeX will round down. **Hack:** The `16`, `\KK@`, and `64` in these lines are really $K/64$, $K/32$, and $K/16$...

1.7229 Build *cosDirection* and *sinDirection* from appropriate characters in the `\xydashfont`. [1.7229]: $cosDirection := wd($`\xydashfont`$SemiDirectionChar)$. [1.7231] $C := SemiDirectionChar - 64$, if $C < 0$: $C := C + 128$, $sinDirection := wd($`\xydashfont`$C)$.

1.7207 Register this dX,dY for next time.

```
7170 \xydef@\setupDirection@ii{%
7171 \ifdim\d@X<\z@ \def\sd@X{-}\else \def\sd@X{+}\fi
7172 \ifdim\d@Y<\z@ \def\sd@Y{-}\else \def\sd@Y{+}\fi
7173 \K@dXdY=\sd@X\d@X \ifdim\K@dXdY<500pt \multiply\K@dXdY\KK@ \fi
7174  \dimen@=\sd@Y\d@Y
7175 % \advance\dimen@.5\KK@ \divide\dimen@\KK@
7176  \advance\dimen@\ifdim\dimen@<\z@-\fi.5\KK@ \divide\dimen@\KK@
7177 \ifdim\dimen@=\z@\else %\count@=\dimen@ \divide\count@\tw@
7178  \advance\K@dXdY by.5\dimen@\relax \divide\K@dXdY\dimen@
7179 \fi \K@dXdY=\sd@X\sd@Y\K@dXdY
7180 \K@dYdX=\sd@Y\d@Y \ifdim\K@dYdX<500pt \multiply\K@dYdX\KK@ \fi
7181  \dimen@=\sd@X\d@X
7182 % \advance\dimen@.5\KK@ \divide\dimen@\KK@
7183  \advance\dimen@\ifdim\dimen@<\z@-\fi.5\KK@ \divide\dimen@\KK@
7184 \ifdim\dimen@=\z@\else %\count@=\dimen@ \divide\count@\tw@
7185  \advance\K@dYdX by.5\dimen@\relax \divide\K@dYdX\dimen@
```

```
 \fi \K@dYdX=\sd@X\sd@Y\K@dYdX
 \Direction=\maxdimen
 \ifnum\K@dXdY<-\K@ \else \ifnum\K@<\K@dXdY \else
   \ifdim \d@Y<\z@
    \Direction=\K@ \advance\Direction-\K@dXdY
   \else
    \Direction=\K@ \multiply\Direction-\thr@@ \advance\Direction-\K@dXdY
 \fi\fi\fi
 \ifnum-\K@<\K@dYdX \ifnum\K@dYdX<\K@
   \ifdim \d@X<\z@
    \Direction=-\K@ \advance\Direction\K@dYdX
   \else
    \Direction=\K@ \multiply\Direction\thr@@ \advance\Direction\K@dYdX
 \fi\fi\fi
 \ifnum\Direction=\maxdimen
  \Direction=\K@dYdX \advance\Direction-\K@dXdY
  \advance\Direction\ifnum\Direction<\z@-\fi\@ne \divide\Direction\tw@
  \ifnum\K@dXdY<\z@ \advance\Direction\K@ \advance\Direction\K@
  \else \advance\Direction-\K@ \advance\Direction-\K@ \fi
 \fi
 \imposeDirection@i
 \resetupDirection@
 }

\xydef@\imposeDirection@{\count@@=\Direction
 \loop@\ifnum\count@@>4096 \advance\count@@-8192 \repeat@
 \loop@\ifnum\count@@<-4096 \advance\count@@8192 \repeat@
 \def\sd@X{+}\ifnum\count@@<\K@ \relax
 \ifnum\count@@>-3072 \def\sd@X{-}\fi\fi
 \def\sd@Y{+}\ifnum\count@@<3072 \relax
  \ifnum\count@@>-\K@ \relax\def\sd@Y{-}\fi\fi
 \imposeDirection@i
 \d@X=\cosDirection\xydashl@ \d@Y=\sinDirection\xydashl@
 \resetupDirection@}

\xydef@\imposeDirection@i{%
 \count@@=\K@ \multiply\count@@ by8 \advance\count@@\Direction
 \count@=\count@@ \advance\count@\KK@ \divide\count@64 \advance\count@\m@ne
 \loop@\ifnum127<\count@ \advance\count@-128 \repeat@
 \chardef\DirectionChar\count@
 \advance\count@@16 \divide\count@@\KK@ \advance\count@@\m@ne
 \loop@\ifnum127<\count@@ \advance\count@@-128 \repeat@
 \chardef\SemiDirectionChar\count@@
 \setbox8=\hbox{\xydashfont\SemiDirectionChar\/}%
 \quotient@@\cosDirection{\sd@X\wd8}\xydashl@
 \setbox8=\hbox{\xydashfont\count@=\SemiDirectionChar\advance\count@-64
  \ifnum\count@<\z@ \advance\count@128 \fi \char\count@\/}%
 \quotient@@\sinDirection{\sd@Y\wd8}\xydashl@}

\xydef@\getxyDirection@#1{\xy@@\bgroup\xy@@ix@{#1}%
 \xy@@{\expandafter\POS\the\toks9\relax\setupDirection@
```

```
  \edef\next@{\egroup \Direction=\the\Direction}\next@ \imposeDirection@ }}
```

Finally some special cases used by the ⟨direction⟩s and directional library objects. All manipulate the Direction dependent parameters and then call `\resetupDirection@`: `\reverseDirection` reverses it; `\above`- and `\belowDirection@` are for `^` and `_`, and `\vDirection@`(x,y){L} is for `:`(x,y), *i.e.*, computes a new direction as the vector

$$<X - x \times \cos\alpha \times L - y \times (-\sin\alpha) \times L, Y - x \times \sin\alpha \times L - y \times \cos\alpha \times L>$$

where α is the previous direction angle.

```
\xydef@\reverseDirection@{%
 \d@X=-\d@X \d@Y=-\d@Y
 \ifdim\d@X<\z@ \def\sd@X{-}\else \def\sd@X{+}\fi
 \ifdim\d@Y<\z@ \def\sd@Y{-}\else \def\sd@Y{+}\fi
 \dimen@=4\K@ \ifnum\Direction<\z@ \advance\Direction\dimen@
  \else \advance\Direction-\dimen@ \fi
 \count@=\DirectionChar \ifnum\count@<64 \advance\count@64 %
  \else \advance\count@-64 \fi \chardef\DirectionChar=\count@
 \edef\cosDirection{\if-\cosDirection\else-\cosDirection\fi}%
 \edef\sinDirection{\if-\sinDirection\else-\sinDirection\fi}%
 \resetupDirection@}

\xydef@\aboveDirection@#1{%
 \dimen@=\d@X \d@X=-\d@Y \d@Y=\dimen@
 \dimen@=\K@dXdY \K@dXdY=-\K@dYdX \K@dYdX=-\dimen@
 \ifdim\d@X<\z@ \def\sd@X{-}\else \def\sd@X{+}\fi
 \ifdim\d@Y<\z@ \def\sd@Y{-}\else \def\sd@Y{+}\fi
 \dimen@=2\K@ \ifdim 1\Direction<\dimen@\else \dimen@=-6\K@ \fi
  \advance\Direction\dimen@
 \count@=\DirectionChar \ifnum\count@<96 \advance\count@32 %
  \else \advance\count@-96 \fi \chardef\DirectionChar=\count@
 \count@=\SemiDirectionChar \ifnum\count@<64 \advance\count@64 %
  \else \advance\count@-64 \fi \chardef\SemiDirectionChar=\count@
 \let\tmp@=\cosDirection
 \edef\cosDirection{\if-\sinDirection\else-\sinDirection\fi}%
 \let\sinDirection=\tmp@
 \dimen@=#1\relax \d@X=\cosDirection\dimen@ \d@Y=\sinDirection\dimen@
 \resetupDirection@}

\xydef@\belowDirection@#1{%
 \dimen@=\d@X \d@X=\d@Y \d@Y=-\dimen@
 \dimen@=\K@dXdY \K@dXdY=-\K@dYdX \K@dYdX=-\dimen@
 \ifdim\d@X<\z@ \def\sd@X{-}\else \def\sd@X{+}\fi
 \ifdim\d@Y<\z@ \def\sd@Y{-}\else \def\sd@Y{+}\fi
 \dimen@=-2\K@\ifdim 1\Direction<\dimen@\dimen@=6\K@\fi
  \advance\Direction\dimen@
 \count@=\DirectionChar \ifnum\count@<32 \advance\count@96 %
  \else \advance\count@-32 \fi \chardef\DirectionChar=\count@
 \count@=\SemiDirectionChar \ifnum\count@<64 \advance\count@64 %
  \else \advance\count@-64 \fi \chardef\SemiDirectionChar=\count@
 \let\tmp@=\sinDirection
 \edef\sinDirection{\if-\cosDirection\else-\cosDirection\fi}%
```

```
\let\cosDirection=\tmp@
\dimen@=#1\relax \d@X=\cosDirection\dimen@ \d@Y=\sinDirection\dimen@
\resetupDirection@}
```

```
\xydef@\vDirection@(#1,#2)#3{\dimen@ii=#3\relax
\dimen@=#1\dimen@ii \dimen@ii=#2\dimen@ii
\d@X=\cosDirection\dimen@ \advance\d@X-\sinDirection\dimen@ii
\d@Y=\sinDirection\dimen@ \advance\d@Y \cosDirection\dimen@ii
\X@p=\X@c \advance\X@p-\d@X \Y@p=\Y@c \advance\Y@p-\d@Y
\setupDirection@\ignorespaces}
```

The above all make use of the following; use them also when the direction state is known to be correct: `\resetDirection@` should be called when p and/or c are moved along the line $\vec{pc}$, `\resetupDirerection` when the entire direction state is changed in a consistent manner.

```
\xydef@\resetDirection@{%
\d@X=\X@c\advance\d@X-\X@p \d@Y=\Y@c\advance\d@Y-\Y@p
\let\next@=\resetupDirection@
\ifdim\sd@X\d@X<\z@ \let\next@=\setupDirection@i \fi
\ifdim\sd@Y\d@Y<\z@ \let\next@=\setupDirection@i \fi
\next@}
```

```
\xydef@\resetupDirection@{%
\edef\Directiontest@@##1##2{\noexpand\DN@{##2}%
 \noexpand\ifdim\noexpand\d@X=\the\d@X\relax
  \noexpand\ifdim\noexpand\d@Y=\the\d@Y\relax \noexpand\DN@{##1}%
 \noexpand\fi\noexpand\fi \noexpand\next@}}
```

```
\xydef@\unsetupDirection@{\def\Directiontest@@##1##2{##2}}
```

Finally the initial direction: up!

```
\uDirection@\xydashl@
```

1.8.2 Edges

An *Edge* is a token list describing the edge of an object. It must have the form `{`⟨expandable token⟩ ⟨unexpandable tokens⟩`}`. To find the edge of an object then first make it the current object and then do

`\the\Edge@c`⟨code⟩

where ⟨code⟩ determines what should be done:

0 c is changed to be equal to the point on the edge intersecting with the line segment from p (for `E` ⟨corner⟩, also this was the v2.6 behaviour).

  Note: This should not change any of A, B, or any component of the state except X_c and Y_c!

1 Test whether the center of p, *i.e.*, `<`X_p`,`Y_p`>`, is 'inside' the c object (or on the edge). Sets the test `\ifInside@` accordingly.

2 Set `\dimen@` to the distance from the center to the edge towards p (as set with code 0).

  Note: This is only positive in the direction towards p (thus negative-sized circles and rectangles make it negative).

3 c is changed to be equal to the point on the edge furthest in the direction towards p (for `P` ⟨corner⟩).

Note: This should not change any of A, B, or any component of the state except X_c and Y_c!

4 Replace c with rectangle with corners where the line from p intersects with the edge of c (thus this is the inner rectangle with corners as the current direction dictates).

5 Replace c with smallest rectangle that encloses the current object completely.

(if this reminds the reader of a 'dictionary' as used by object-oriented programming languages then they probably share this author's regret that TEX is not object-oriented :-)

```
\message{edges,}
```

```
\xynew@{if}\ifInside@
```

Points: The simplest shape is none at all – a point.

```
\xydef@\zeroEdge#1{%
 \ifcase#1\relax \or \Inside@false \or \dimen@=\z@
 \or \else \Edge@c={\rectangleEdge}\fi}
```

Circles and Ellipses: Next we define round things, with several special cases for the individual variants described below: centered circular, centered elliptical, and general elliptical. The only 'intelligent' choice is for the 'Under' method where the object is forced to be a centered variant before applying either the circular or elliptical version.

```
\xydef@\circleEdge#1{\ifcase#1\expandafter\circleEdge@
 \or \expandafter\circleUnder@ \or \dimen@=\R@c
 \or \expandafter\circleProp@ \or \expandafter\circleInner@
 \else \expandafter\circleOuter@ \fi}
```

```
\xydef@\circleEdge@{\DN@{\ellipseEdge@}%
 \ifdim\R@c=\L@c\relax \ifdim\U@c=\D@c\relax
  \ifdim\R@c=\U@c\DN@{\circlecentredEdge@}%
  \else\DN@{\ellipsecentredEdge@}\fi
 \fi\fi \next@}
```

```
\xydef@\circleProp@{\DN@{\reverseDirection@\ellipseEdge@}%
 \ifdim\R@c=\L@c\relax \ifdim\U@c=\D@c\relax
  \ifdim\R@c=\U@c\DN@{\reverseDirection@\circlecentredEdge@}%
  \else\DN@{\reverseDirection@\ellipsecentredEdge@}\fi
 \fi\fi \next@}
```

```
\xydef@\circleUnder@{\Inside@false
 \ifdim\X@p=\X@c \relax \ifdim\Y@p=\Y@c \Inside@true \fi \fi
 \ifInside@ \else \expandafter \circleCentre@ \fi}
```

```
\xydef@\circleCentre@{{%
 \ifdim\L@c=\R@c \relax\else
  \dimen@=\R@c\advance\dimen@-\L@c \divide\dimen@\tw@
  \advance\X@c\dimen@ \advance\R@c-\dimen@ \fi
 \d@X=\X@c \advance\d@X-\X@p \d@X=\ifdim\d@X<\z@-\fi\d@X
 \ifdim\U@c=\D@c\relax \else
  \dimen@=\U@c\advance\dimen@-\D@c \divide\dimen@\tw@
  \advance\Y@c\dimen@ \advance\U@c-\dimen@ \fi
 \d@Y=\Y@c \advance\d@Y-\Y@p \d@Y=\ifdim\d@Y<\z@-\fi\d@Y
```

```
\DN@{}\ifdim\d@X>\R@c \relax \else \ifdim\d@Y>\U@c \relax
 \else \ifdim\U@c=\R@c \DN@{\circlecentredUnder@}%
 \else \DN@{\ellipsecentredUnder@}\fi
\fi\fi \next@}}
```

```
\xydef@\circleInner@{\DN@{\ellipseInner@}%
\ifdim\R@c=\L@c\relax \ifdim\U@c=\D@c\relax
 \ifdim\R@c=\U@c\DN@{\circlecentredInner@}%
 \else\DN@{\ellipsecentredInner@}\fi
\fi\fi \next@}
```

```
\xydef@\circleOuter@{\DN@{\ellipseOuter@}%
\ifdim\R@c=\L@c\relax \ifdim\U@c=\D@c\relax
 \ifdim\R@c=\U@c\DN@{\circlecentredOuter@}%
 \else\DN@{\ellipsecentredOuter@}\fi
\fi\fi \next@}
```

True circles, centered: Code 0 moves $<X_c, Y_c>$ to the point $<X_c - R_c \times \cos\alpha, Y_c - R_c \times \sin\alpha>$ where α is the current direction angle, code 1 tests whether the p center is located between those two points., code 2 just returns the radius, code 3 is as code 0 and code 4 is the only nontrivial one, replacing with the inner symmetric rectangle with corner at the point of code 0.

```
\xydef@\circlecentredEdge@{%
\dimen@=-\cosDirection\R@c \advance\X@c\dimen@
\dimen@=-\sinDirection\R@c \advance\Y@c\dimen@}
```

```
\xydef@\circlecentredUnder@{%
\loop\ifdim\R@c>100\p@ \circlescale@ \repeat
\edef\tmp@{\expandafter\removePT@\the\R@c}\dimen@=\tmp@\R@c
\edef\tmp@{\expandafter\removePT@\the\d@X}\advance\dimen@-\tmp@\d@X
\edef\tmp@{\expandafter\removePT@\the\d@Y}\advance\dimen@-\tmp@\d@Y
\ifdim\dimen@>\z@ \aftergroup\Inside@true \fi}
```

```
%%\xydef@\circlescale@{\divide\R@c\KK@ \divide\d@X\KK@ \divide\d@Y\KK@ }
\xydef@\circlescale@{\divide\R@c\KK@ \divide\d@X\KK@ \divide\d@Y\KK@ }
```

```
\xydef@\circlecentredInner@{%
\L@c=\sd@X\cosDirection\R@c \D@c=\sd@Y\sinDirection\R@c
\R@c=\L@c \U@c=\D@c \Edge@c={\rectangleEdge}}
```

```
\xydef@\circlecentredOuter@{%
\L@c=\R@c \D@c=\R@c \U@c=\D@c \Edge@c={\rectangleEdge}}
```

Ellipsis, centered: When c is at the centre of an elliptical object, first rescale the axes to make the object circular. Rescale dX and dY appropriately, reset the direction and move to the edge as previously. Then scale back to the original shape, adjusting X_c and Y_c appropriately.

```
\xydef@\ellipsecentredEdge@{\bgroup \X@p=\X@c \Y@p=\Y@c
\ifdim\U@c>\R@c
 \X@c=\cosDirection\U@c \Y@c=\sinDirection\U@c
 \quotient@\tmp@\U@c\R@c \X@c=\tmp@\X@c \R@c=\U@c
\else
 \X@c=\cosDirection\R@c \Y@c=\sinDirection\R@c
 \quotient@\tmp@\R@c\U@c \Y@c=\tmp@\Y@c
```

```
\fi
\advance\X@c\X@p \advance\Y@c\Y@p
\setupDirection@ \X@c=\X@p \Y@c=\Y@p \circlecentredEdge@
\d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
\ifdim\U@c>\L@c \quotient@\tmp@\L@c\U@c \d@X=\tmp@\d@X
\else \quotient@\tmp@\U@c\R@c \d@Y =\tmp@\d@Y \fi
\X@c=\X@p \advance\X@c\d@X \Y@c=\Y@p \advance\Y@c\d@Y
\edef\next@{\egroup \X@c=\the\X@c \Y@c=\the\Y@c}\next@ }%
```

```
\xydef@\ellipsecentredUnder@{%
\ifdim\R@c>64\p@ \circlescale@ \divide\U@c\KK@
\else \ifdim\U@c>64\p@ \circlescale@ \divide\U@c\KK@ \fi\fi
\edef\tmp@{\expandafter\removePT@\the\R@c}\d@Y=\tmp@\d@Y
\edef\tmp@{\expandafter\removePT@\the\U@c}\d@X=\tmp@\d@X
\R@c=\tmp@\R@c \circlecentredUnder@ }
```

For elliptical objects the Outer-rectangle retains all the extents, so just change the edge-type.

```
\xydef@\ellipsecentredOuter@{\Edge@c={\rectangleEdge}}
```

For elliptical objects the Inner-rectangle needs to be calculated for each direction. This is done by first finding the edge, using `\ellipseEdge@` or `\ellipsecentredEdge@` when appropriate.

```
\xydef@\ellipsecentredInner@{%
\bgroup \X@p=\X@c \Y@p=\Y@c \ellipsecentredEdge@
% \advance\X@c-\X@p \L@c=\ifdim\X@c<\z@-\fi\L@c
  \advance\X@c-\X@p \L@c=\ifdim\X@c<\z@-\fi\X@c
% \advance\Y@c-\Y@p \D@c=\ifdim\Y@c<\z@-\fi\D@c
  \advance\Y@c-\Y@p \D@c=\ifdim\Y@c<\z@-\fi\Y@c
\edef\next@{\egroup \L@c=\the\L@c \D@c=\the\D@c}\next@
\R@c=\L@c \U@c=\D@c \Edge@c={\rectangleEdge}}
```

Non-centered variants: The code for the more general `\ellipseEdge@` uses a Newton iteration to solve a quadratic equation.

First locate the actual centre of the ellipse and the lengths of the major axis a and minor axis b. Let (x, y) denote the displacement of the required edge point from the centre of the ellipse, so that $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. Now let (dx, dy) be the displacement from the centre to the reference point; then $x = dx + s\cos\theta$ and $y = dy + s\sin\theta$ where s is the distance to (x, y) in the required direction θ. We must find the value of s, presumed to satisfy $s \geq 0$.

If $a < b$ we search for a root of

$$f\left(\frac{s}{a}\right) = \left(\tilde{dx} + \tfrac{s}{a}\cos\theta\right)^2 + \left(\tilde{dy} + \tfrac{s}{a}\tfrac{a}{b}\sin\theta\right)^2$$
$$f'\left(\frac{s}{a}\right) = 2\left(\tilde{dx} + \tfrac{s}{a}\cos\theta\right)\cos\theta + 2\left(\tilde{dy} + \tfrac{s}{a}\tfrac{a}{b}\sin\theta\right)\tfrac{a}{b}\sin\theta$$

where $\tilde{dx} = dx/a$ and $\tilde{dy} = dy/b$. By choosing a starting value $s_0 = 2a + 2b$ the Newton-iteration converges rapidly to the correct root, provided $b > 0$. If $a > b$ then we solve a similar quadratic in $\frac{s}{b}$.

To Do: The case of $a = 0$ or $b = 0$, so that the ellipse degenerates to a line segment, should be trapped earlier. Such an object would be more easily handled as a degenerate case of a rectangle.

To Do: In the case when the reference point lies outside the ellipse the Newton iteration is not guaranteed to converge. A simple test — s growing rather than decreasing in size — detects possible converge problems and exits the iteration. The following message is issued ...

```
Xy-pic warning: poor convergence.
```

Often, particularly when the reference point is on or near the edge of the ellipse, an acceptable approximation to the edge has been found anyway.

```
\xydef@\ellipseEdge@{\bgroup
 \A@=\R@c \B@=\U@c
 \ifdim\R@c=\L@c \d@X=\z@
 \else \d@X=.5\R@c \advance\d@X-.5\L@c
  \advance\A@\L@c \divide\A@\tw@ \fi
 \ifdim\U@c=\D@c \d@Y=\z@
 \else \d@Y=.5\U@c \advance\d@Y-.5\D@c
  \advance\B@\D@c \divide\B@\tw@ \fi
 \bgroup
  \L@c=\A@ \U@c=\B@
  \R@p=\U@c \advance\R@p\L@c \multiply\R@p\tw@
  \ifdim\B@<\A@  \quotient@\tmp@\U@c\L@c \R@c=\tmp@\p@ \D@c=\p@
     \quotient@\tmp@\R@p\L@c \R@p=\tmp@\p@
  \else
   \ifdim\A@<\B@ \quotient@\tmp@\L@c\U@c \D@c=\tmp@\p@ \R@c=\p@
    \quotient@\tmp@\R@p\U@c \R@p=\tmp@\p@
  \else
   \R@c=\p@ \D@c=\p@ \quotient@\tmp@\R@p\U@c \R@p=\tmp@\p@
  \fi\fi
  \quotient@\sd@X\d@X\L@c \d@X=\sd@X\p@
  \quotient@\sd@Y\d@Y\U@c \d@Y=\sd@Y\p@
 \loop
  \bgroup \U@p=-\p@ \D@p=\z@
   \ifdim\R@c<\p@
    \edef\tmp@{\expandafter\removePT@\the\R@c}\dimen@=\tmp@\R@p
    \advance\d@X\cosDirection\dimen@
   \else \advance\d@X\cosDirection\R@p \fi
   \edef\sd@X{\expandafter\removePT@\the\d@X}%
   \advance\U@p\sd@X\d@X
   \ifdim\R@c<\p@
    \edef\tmp@{\expandafter\removePT@\the\R@c}\dimen@=\tmp@\d@X
    \advance\D@p\cosDirection\dimen@
   \else \advance\D@p\cosDirection\d@X \fi
%
   \ifdim\D@c<\p@
    \edef\tmp@{\expandafter\removePT@\the\D@c}\dimen@=\tmp@\R@p
    \advance\d@Y\sinDirection\dimen@
   \else \advance\d@Y\sinDirection\R@p \fi
   \edef\sd@Y{\expandafter\removePT@\the\d@Y}%
   \advance\U@p\sd@Y\d@Y
   \ifdim\D@c<\p@
    \edef\tmp@{\expandafter\removePT@\the\D@c}\dimen@=\tmp@\d@Y
    \advance\D@p\sinDirection\dimen@
   \else \advance\D@p\sinDirection\d@Y \fi
```

```
   \multiply\D@p\tw@
   \dimen@=\ifdim\U@p<\z@-\fi\U@p
    \ifdim\dimen@<.01\p@ \U@p=\z@
    \else
     \quotient@\tmp@\U@p\D@p \U@p=\tmp@\p@
     \ifdim\U@p<\z@\xywarning@{poor convergence}\U@p=\z@
     \else \advance\R@p-\U@p \U@p=\ifdim\U@p<\z@-\fi\U@p
    \fi \fi
    \edef\next@{\egroup \R@p=\the\R@p \U@p=\the\U@p \D@p=\the\D@p}\next@
   \ifdim\U@p>\z@ \repeat
  \edef\next@{\egroup \dimen@=\the\R@p}\next@
  \edef\tmp@{\expandafter\removePT@\the\dimen@}%
  \ifdim\B@<\A@ \dimen@=\tmp@\B@ \else \dimen@=\tmp@\A@ \fi
  \dimen@=-\dimen@
  \advance\X@c\cosDirection\dimen@
  \advance\Y@c\sinDirection\dimen@
 \edef\next@{\egroup \X@c=\the\X@c \Y@c=\the\Y@c}\next@ }%
```

The ‘Under’ procedure is identical to the centered version.

```
\xydef@\ellipseOuter@{\Edge@c={\rectangleEdge}}
```

```
\xydef@\ellipseInner@{%
 \bgroup \X@p=\X@c \Y@p=\Y@c \ellipseEdge@
 \d@X=\X@c\advance\d@X-\X@p
 \ifdim\d@X>\z@ \R@p=\d@X \L@p=\R@p
  \ifdim\L@c=\R@c\else\advance\L@p\L@c \advance\L@p-\R@c \fi
 \else \L@p=-\d@X \R@p=\L@p
  \ifdim\L@c=\R@c \else\advance\R@p\R@c \advance\R@p-\L@c \fi
 \fi
 \d@Y=\Y@c\advance\d@Y-\Y@p
 \ifdim\d@Y>\z@ \U@p=\d@Y \D@p=\U@p
  \ifdim\D@c=\U@c\else\advance\D@p\D@c \advance\D@p-\U@c \fi
%\else \D@p=-\d@X \R@p=\D@p
 \else \D@p=-\d@Y \U@p=\D@p                     % <<<<<<========
  \ifdim\D@c=\U@c\else\advance\U@p\U@c \advance\U@p-\D@c \fi
 \fi
 \edef\next@{\egroup
  \L@c=\the\L@p \D@c=\the\D@p \R@c=\the\R@p \U@c=\the\U@p}%
 \next@ \Edge@c={\rectangleEdge}}
```

Rectangles:

Rectangles intersection is slightly more complicated and handled separately for the horizontal and vertical case.

```
\xydef@\rectangleEdge#1{\ifcase#1\expandafter\rectangleEdge@
 \or \expandafter\rectangleUnder@ \or \expandafter\rectangleDist@
 \or \expandafter\rectangleProp@ %%%\or \expandafter\rectangleInner@ BUG!
 \else \relax \fi}
```

Bug: no ‘Inner’ for rectangles!

```
% \rectangleEdge@
```

```
%       Sets <X,Y> to the intersection of a line from <X-dX,Y-dY> to <X,Y>
%       and the rectangle from <X-L,Y-D> to <X+R,Y+U>:
%
%       %1a     dY<0, dX<0:     X := X + min{R, U*|dX/dY|},
%                               Y := Y + min{U, R*|dY/dX|};
%         b     dY<0, dX=0:     Y := Y + U;
%         c     dY<0, dX>0:     X := X - min{L, U*|dX/dY|},
%                               Y := Y + min{U, L*|dY/dX|};
%
%       %2a     dY=0, dX<0:     X := X + R;
%         b     dY=0, dX=0:     ;
%         c     dY=0, dX>0:     X := X - L;
%
%       %3a     dY>0, dX<0:     X := X + min{R, D*|dX/dY|},
%                               Y := Y - min{D, R*|dY/dX|};
%         b     dY>0, dX=0:     Y := Y - D;
%         c     dY>0, dX>0:     X := X - min{L, D*|dX/dY|},
%                               Y := Y - min{D, L*|dY/dX|};
%
%       %4 \resetupDirection@ to register that even though dX,dY changed all
%          Direction parameters are is still valid!
%
%       NOTE: d=0 really means |d| < .05pt.
%
\xydef@\rectangleEdge@{%
 \ifdim\d@Y<-.05\p@ \rectangleEdge@i                                  %1
 \else\ifdim\d@Y<.05\p@ \rectangleEdge@ii                             %2
 \else \rectangleEdge@iii\fi\fi
 \resetupDirection@}

\xydef@\rectangleEdge@i{%
 \ifdim\d@X<-.05\p@
  \settomin@\X@c+\R@c\U@c\d@X\d@Y \settomin@\Y@c+\U@c\R@c\d@Y\d@X%1a
 \else\ifdim\d@X<.05\p@ \advance\Y@c\U@c                              %1b
 \else \settomin@\X@c-\L@c\U@c\d@X\d@Y \settomin@\Y@c+\U@c\L@c\d@Y\d@X   %1c
 \fi\fi}

\xydef@\rectangleEdge@ii{%
 \ifdim\d@X<-.05\p@ \advance\X@c\R@c                                  %2a
 \else\ifdim\d@X<.05\p@                                               %2b
 \else \advance\X@c-\L@c                                              %2c
 \fi\fi}

\xydef@\rectangleEdge@iii{%
 \ifdim\d@X<-.05\p@
  \settomin@\X@c+\R@c\D@c\d@X\d@Y \settomin@\Y@c-\D@c\R@c\d@Y\d@X%3a
 \else\ifdim\d@X<.05\p@ \advance\Y@c-\D@c                             %3b
 \else \settomin@\X@c-\L@c\D@c\d@X\d@Y \settomin@\Y@c-\D@c\L@c\d@Y\d@X   %3c
 \fi\fi}

\xydef@\settomin@#1#2#3#4#5#6{%
 % Perform d := #2 min{#3, #4*|#5/#6|}; #1 := #1+d ...
```

```
7759 \edef\nextii@{\A@=\the\A@ \B@=\the\B@}\quotient@\next@{#5}{#6}\nextii@
7760 \dimen@=\sd@X\sd@Y\next@#4\relax
7761 \ifdim#3<\dimen@ \dimen@=#3\fi \advance#1#2\dimen@}
```

Checking that p is under is simpler: Normally it is not but if $c = p$ or ($-L_c < X_p - X_c < R_c$ and $-D_c < Y_p - Y_c < U_c$) then it is.

```
7769 \xydef@\rectangleUnder@{\Inside@false
7770  \ifdim\X@p=\X@c \ifdim\Y@p=\Y@c \Inside@true \fi\fi
7771  \ifInside@ \else
7772   \dimen@=\X@p \advance\dimen@-\X@c
7773   \ifdim \dimen@>-\L@c \relax \ifdim\dimen@<\R@c
7774    \dimen@=\Y@p \advance\dimen@-\Y@c
7775    \ifdim \dimen@>-\D@c \relax \ifdim\dimen@<\U@c
7776     \Inside@true
7777  \fi\fi\fi\fi\fi }
```

Calculating the width is like computing the edge point, just simpler: (1.7787) Set A to the horizontal and B to the vertical distance applicable; if either is zero (or close) use the other unmodified . (1.7796) If both nonzero use $\min(A/\cos\alpha, B/\sin\alpha)$ where α is the current angle.

```
7787 \xydef@\rectangleDist@{\let\next@=\rectangleDist@i
7788  \ifdim\d@X<-.05\p@ \dimen@=\R@c
7789  \else\ifdim\d@X<.05\p@ \dimen@=\z@ \DN@{\dimen@=\dimen@ii}%
7790  \else \dimen@=\L@c \fi\fi
7791  \ifdim\d@Y<-.05\p@ \dimen@ii=\U@c
7792  \else\ifdim\d@Y<.05\p@ \DN@{}%
7793  \else \dimen@ii=\D@c \fi\fi
7794  \next@}

7796 \xydef@\rectangleDist@i{
7797  \begingroup \quotient@\next\p@{\sd@X\cosDirection\p@}%
7798   \edef\next{\endgroup \dimen@=\next\dimen@}\next
7799  \begingroup \quotient@\next\p@{\sd@Y\sinDirection\p@}%
7800   \edef\next{\endgroup \dimen@ii=\the\dimen@ii}\next
7801  \ifdim\dimen@ii<\dimen@ \dimen@=\dimen@ii \fi}
```

Setting c to the 'proportional edge point' is straight out of v2.6's `\setlabel@@` macro…

```
7808 % First rotate to opposite direction.

7810 % Then compute Leftf,Upf [\next@,\nextii@]:
7811 %               0,(D-2K)/2K     for  2K < D <= 4K       [right]
7812 %        (2K-D)/2K,0            for   0 < D <= 2K       [down]
7813 %               1,-D/2K         for -2K < D <= 0        [left]
7814 %        (D+4K)/2K,1            for -4K < D <= -2K      [up]

7816 % Finally set <X,Y> to
7817 %        < X - L + Leftf*(R+L) , Y + U - Upf*(D+U) >

7819 \xydef@\rectanglePropT@{%
7820  \enter@{\A@=\the\A@ \B@=\the\B@ \DirectionfromtheDirection@}%
7821  \reverseDirection@
7822  \dimen@=1\Direction \count@=\K@ \multiply\count@\tw@
7823  \ifnum \Direction>\count@
7824   \DN@{0}%
```

```
 \advance\dimen@-2\K@ \quotient@\nextii@{\dimen@}{2\K@}%
\else\ifnum \Direction>\z@
 \dimen@=-\dimen@ \advance\dimen@2\K@ \quotient@\next@{\dimen@}{2\K@}%
 \DNii@{0}%
\else\ifnum \Direction>-\count@
 \DN@{1}%
 \quotient@\nextii@{-\dimen@}{2\K@}%
\else
 \advance\dimen@4\K@ \quotient@\next@{\dimen@}{2\K@}%
 \DNii@{1}%
\fi\fi\fi
\advance\X@c-\L@c \dimen@=\L@c \advance\dimen@\R@c
 \ifdim\dimen@=\z@ \advance\X@c 2\L@c \else \advance\X@c\next@\dimen@ \fi
\advance\Y@c+\U@c \dimen@=\D@c \advance\dimen@\U@c
 \ifdim\dimen@=\z@ \advance\Y@c-2\U@c \advance\Y@c\Upness@\U@c
 \else \advance\Y@c-\nextii@\dimen@ \fi
\leave@}
```

1.8.3 Connections

```
\message{connections;}
```

Connections describe how a particular ⟨object⟩ may be used to connect p to c. The following parameters and methods are defined by any connection; they should be used in the indicated sequence:

`\Invisible@@`, `\Hidden@@`, and `\Drop@@`: as for ⟨object⟩.

`\Creset@@`: (Re)set the connection parameters to allow use of the following to move to a point on the connection (this is what the interpretation of ⟨pos⟩ `?` does first). All effects obtained by the following methods are undone: `\Creset@@` reestablishes the state to what it was just after the `\Connect@@` every time.

`\Cshavep@@` or `\Cshavec@@`: Change p or c to a zero-sized object at the start or finish of the connection (the first `<` and `>` ⟨place⟩ components).

`\Calong@@{`⟨factor⟩`}`: Move c to point the ⟨factor⟩ along the connection and set the direction parameters as a tangent to it in this point (the `(`⟨factor⟩`)` ⟨place⟩ component).

`\Cslidep@@{`⟨dimen⟩`}` or `\Cslidec@@{`⟨dimen⟩`}`: Move p or c the ⟨dimen⟩sion further in the current direction. Can be used both before and after the `\Calong@@` method (these handle subsequent `<` and `>` as well as the `/`⟨slide⟩`/` ⟨place⟩ component).

`\Cintercept@@`: Set c to a zerosized object at the point where the connection crosses the straight line from p to c (the `!{...}` ⟨place⟩ component).

`\Cbreak@@` and `\Clast@@`: These support typesetting of the connection in several 'subsegments' as used by the arrow option. The idea is the following, using the internal *lastbreak* and *nextbreak* positions:

1. `\Connect@@` sets *lastbreak* $:= p$ and *nextbreak* $:= c$
2. The commands `\Cslidec@@`, `\Cslidep@@`, and `\Cintercept@@`, set *nextbreak* logically to the position that is obtained by the manipulation.

3. `\Cbreak@@` retypesets the connection except only the piece from *lastbreak* to the current c object, which *must* be at on top of *nextbreak*, is typeset. Afterwords it sets $lastbreak :=$ $nextbreak$ and $nextbreak := c$ (where c is the original c of the connection).
4. The command `\Clast@@` typesets from *lastbreak* and the remainder of the connection, and resets $lastbreak := p$ and $nextbreak := c$.

Notice that it is not necessary for *nextbreak* to actually exist, in fact in the kernel it is just an alias for c at the time `\Cbreak@@` is called.

These are the defaults for basic XY-pictures that only has straight connections hence the macros all starting with `\no`... are rather simple – in particular they just merge with zerosized objects at the locations cutoff locations (this was inherited from the broken `arrow` code ☺).

```
\xydef@\Creset@@{}
\xydef@\Cshavep@@{\noCshavep@@}
\xydef@\Cshavec@@{\noCshavec@@}
\xydef@\Cslidep@@{\noCslidep@@}
\xydef@\Cslidec@@{\noCslidec@@}
\xydef@\Calong@@{\noCalong@@}
\xydef@\Cintercept@@{\noCintercept@@}
\xydef@\Cbreak@@{\noCbreak@@}
\xydef@\Clast@@{\noClast@@}

\xylet@\lastbreak@@=\empty

\xydef@\noCshavep@@{\setupDirection@
 \enter@{\cfromthec@ \DirectionfromtheDirection@}%
 \reverseDirection@ \cfromp@ \the\Edge@c\z@
 \pfromc@ \leave@ \resetDirection@}

\xydef@\noCshavec@@{\setupDirection@ \the\Edge@c\z@ \resetDirection@}

\xydef@\noCslidep@@#1{\dimen@=#1\relax
 \advance\X@p\cosDirection\dimen@ \advance\Y@p\sinDirection\dimen@
 \resetDirection@}

\xydef@\noCslidec@@#1{\dimen@=#1\relax
 \advance\X@c\cosDirection\dimen@ \advance\Y@c\sinDirection\dimen@
 \resetDirection@}

\xydef@\noCalong@@#1{%
 \d@X=#1\d@X \d@Y=#1\d@Y \X@c=\X@p \Y@c=\Y@p \advance\X@c\d@X \advance\Y@c\d@Y
 \resetupDirection@}
```

Next the intercept method that just uses `\intersect@`.

```
\xydef@\noCintercept@@{\enter@{\pfromthep@}%
 \begingroup\Creset@@ \edef\tmp@{\endgroup
  \X@origin=\the\X@p \Y@origin=\the\Y@p \R@c=\the\d@X \U@c=\the\d@Y}\tmp@
 \loop@\dimen@=\ifdim\R@c<\z@-\fi\R@c \advance\dimen@\ifdim\U@c<\z@-\fi\U@c
  \ifdim\dimen@>10\p@ \advance\R@c \ifdim\R@c<\z@-\fi 16sp \divide\R@c\KK@
   \advance\U@c \ifdim\U@c<\z@-\fi 16sp \divide\U@c\KK@ \repeat@
 \intersect@ \leave@}
```

And finally the dummy 'break' method which of course does nothing except the required shifting.

```
\xydef@\noCbreak@@{\edef\lastbreak@@{\cfromthec@}}
\xydef@\noClast@@{\Creset@@ \edef\lastbreak@@{\cfromthep@}}
```

The kernel only defines one kind of connection which is used by default: 'filling' many copies of the connect object on the straight line from p to c.

First the 'dummy' connection type `no` used for ⟨object⟩s where it doesn't really make sense to 'connect', and used to provide defaults for easy operations:

```
7986 \xydef@\no@@{\setupDirection@ \edef\lastbreak@@{\cfromthep@}%
7987  \enter@{\cfromthec@\pfromthep@\DirectionfromtheDirection@}%
7988   \edef\Creset@@{\cfromthec@ \pfromthep@ \DirectionfromtheDirection@}%
7989   \edef\next@{{\sd@X}{\sd@Y}}\expandafter\straightcheckoverlap@\next@
7990   \setupDirection@ \the\Edge@c\z@
7991   \swap@\reverseDirection@\the\Edge@c\z@\swap@\reverseDirection@
7992   \resetDirection@
7993   \edef\Cshavep@@{\DirectionfromtheDirection@
7994    \pfromthep@\noexpand\resetDirection@}%
7995   \edef\Cshavec@@{\DirectionfromtheDirection@
7996    \cfromthec@\noexpand\resetDirection@}%
7997  \leave@
7998  \def\Cslidep@@{\noCslidep@@}\def\Cslidec@@{\noCslidec@@}%
7999  \def\Calong@@{\noCalong@@}\def\Cintercept@@{\noCintercept@@}%
8000  \def\Cbreak@@{\noCbreak@@}\def\Clast@@{\noClast@@}\noinsert@}
```

```
8002 \xydef@\noinsert@{\ifHidden@\else
8003   \ifdim\Y@c>\Y@max \Y@max=\Y@c \fi  \ifdim\Y@p>\Y@max \Y@max=\Y@p \fi
8004   \ifdim\Y@c<\Y@min \Y@min=\Y@c \fi  \ifdim\Y@p<\Y@min \Y@min=\Y@p \fi
8005   \ifdim\X@c>\X@max \X@max=\X@c \fi  \ifdim\X@p>\X@max \X@max=\X@p \fi
8006   \ifdim\X@c<\X@min \X@min=\X@c \fi  \ifdim\X@p<\X@min \X@min=\X@p \fi
8007  \fi}
```

Next 'straight' connections that use the following special parameters to customises how to repeat a 'filler' object as many times as needed to draw the connection:

`\lastobjectbox@`: box to fill with; assumed to be 'trimmed' to have only the size needed for the filler; size is $w \times (h + d)$).

The method is called as `\straight@{`⟨spread⟩`}` where ⟨spread⟩ is a macro to expand *after* the number of fillers N (in `\count@@`) is known and with $A = w$ and $B = d + h$ but *before* any filling is done. May change N as well as dX, dY, X, Y, w, d, h in order to affect the filling.

Procedure: [1.8046] setup direction parameters, clear the break state, and define the `\Creset@@` macro to reestablish this initial state; [1.8049] build macro to discover if the edges of the objects overlap (it edges were removed); [??] move both p, c to the edges of the objects…and define the `\Cshave*@@` macros accordingly; [1.8058] choose either to ignore the connection if requested or there is overlap between the objects of choose either vbox or hbox version (see below)…; and [1.8063] build `\Calong@@`, `\Cbreak@@`, and `\Clast@@` macros, and reset pc.

Note: Assumes that the direction is not tampered with between Creset is defined and used…

```
8043 \xydef@\Spread@@{}
8044 \xydef@\checkoverlap@@{}
```

```
8046 \xydef@\straight@#1{\setupDirection@ \def\Spread@@{#1}%
8047  \edef\Creset@@{\cfromthec@ \pfromthep@  \DirectionfromtheDirection@}%
8048  \edef\lastbreak@@{\cfromthep@}%
8049  \edef\next@{{\sd@X}{\sd@Y}}\expandafter\straightcheckoverlap@\next@
```

```
 \setupDirection@ \the\Edge@c\z@
 \swap@\reverseDirection@\the\Edge@c\z@\swap@\reverseDirection@
 \resetDirection@
 \edef\Cshavep@@{\DirectionfromtheDirection@
  \pfromthep@\noexpand\resetDirection@}%
 \edef\Cshavec@@{\DirectionfromtheDirection@
  \cfromthec@\noexpand\resetDirection@}%
 \noinsert@
 \straight@typeset
 \def\Cbreak@@{\straightCbreak@}\def\Clast@@{\straightClast@}%
% \DN@{\solidSpread@}\ifx\next@\Spread@@
%  \checkXy@rules@\solidhrule@pre\solidvrule@pre\relax
% \fi
 \def\Cslidep@@{\noCslidep@@}\def\Cslidec@@{\noCslidec@@}
 \def\Calong@@{\noCalong@@}\def\Cintercept@@{\noCintercept@@}%
 \Creset@@}
```

```
\xydef@\straightcheckoverlap@#1#2{\def\checkoverlap@@{%
  \ifdim#1\X@p>#1\X@c \let\next@=\relax \fi
  \ifdim#2\Y@p>#2\Y@c \let\next@=\relax \fi}}
```

```
\xydef@\straight@typeset{%
 \ifInvisible@ \let\next@=\relax
 \else\ifdim 1\Direction<-2\K@ \DN@{\straightv@}%
 \else\ifdim 1\Direction<\z@   \DN@{\straighth@}%
 \else\ifdim 1\Direction<2\K@  \DN@{\straightv@}%
 \else                         \DN@{\straighth@}%
 \fi\fi\fi\fi \checkoverlap@@ \next@}
```

The methods `\straightCbreak@` and `\straightClast@` implement the break method parametrised on the 'typeset' primitive which is usually `\straight@typeset` which is a simplified version of `\straight@` that does not initialise. This relies on the connection being straight: the `Cbreak@@` method executes $p, c := p\,.\,\mathit{lastbreak}, c\,.\,\mathit{nextbreak}$ and `Clast@@` executes $p, c := p\,.\,\mathit{lastbreak}, c$, where in both cases the `[Inner]` modifier is applied just before the merging to make sure the edge is respected by the merged object.

```
\xylet@\nextbreak@@=\empty
```

```
\xydef@\straightCbreak@{\connectRestore@
 \edef\nextbreak@@{\cfromthec@}%
 \Creset@@ \swap@ \the\Edge@c4\enter@{\cmergethec@}\lastbreak@@ \leave@
 \swap@ \the\Edge@c4\enter@{\cmergethec@}\nextbreak@@ \leave@
 \setupDirection@
 \the\Edge@c\z@
 \swap@ \reverseDirection@ \the\Edge@c\z@ \swap@ \reverseDirection@
 \resetDirection@ \noinsert@ \straight@typeset
 \Creset@@ \let\lastbreak@@=\nextbreak@@}
```

```
\xydef@\straightClast@{\connectRestore@
 \Creset@@ \swap@ \the\Edge@c4\enter@{\cmergethec@}\lastbreak@@ \leave@
 \swap@
 \setupDirection@
 \the\Edge@c\z@
```

```
8108 \swap@\reverseDirection@ \the\Edge@c\z@ \swap@\reverseDirection@
8109 \resetDirection@ \noinsert@ \straight@typeset
8110 \Creset@@ \edef\lastbreak@@{\cfromthep@}}
```

Now for the typesetting – to summarise: these are parametrised on direction parameters and use X, Y: endpoint of connection ('Direction end'). dX, dY: connection distance ('Direction vector'). `\Leftness@` of ⟨object⟩. `\lastobjectbox@` to fill with; assumed to be 'trimmed'. `\Spread@@` to expand to modify the default spreading used. `\Drop@@` to expand to actually typeset the finished connection when it is in box0! **Note:** Must make box0 void to avoid 'box leaks'.

Two variants exist: one for mostly horisontal and one for mostly vertical filling. We detail the 'mostly horisontal' one:

[??] start box; [1.8151] compute initial adjustment (X right and Y up) and $A := w$, $B := h + d$, $N = \lfloor(\rfloor\,|dX|/A)$, and run user supplied Spread; [1.8155] adjust first filler position horizontally: if $dX > 0$ then $X := X - w$; [1.8156] adjust first filler vertically: $Y := Y - sdX \times L \times (dY/dX) \times w$ where $L = \text{if} dX > 0 \text{then} (1 - Leftness) \text{else} Leftness$; [1.8160] recompute the move dimensions (A right and B up): $A := -sdX(|dX| - w)/(N-1)$, $B := sdY\,|sdX(w) \times dY/dX - dY|/(N-1)$, and set the filler box to have $w := A$; [1.8166] output first filler adjusted X, Y and the following $N - 1$ each A further right and raised B more than the previous; and finally end object with usual bravour!

```
8149 \xydef@\straighth@{\setbox\z@=\hbox{%
8150   \setbox8=\copy\lastobjectbox@
8151   \A@=\wd8\relax \B@=\dp8\relax \advance\B@\ht8\relax
8152   \ifdim \A@=\z@ \count@@=\m@ne
8153   \else \dimen@=\sd@X\d@X \divide\dimen@\A@ \count@@=\dimen@ \fi
8154   \Spread@@
8155   \ifdim\d@X>\z@ \advance\X@c-\wd8\relax\fi
8156   \dimen@=-\sd@X\wd8\relax
8157   \multiply\dimen@\K@dYdX \divide\dimen@\K@
8158   \ifdim\d@X>\z@ \advance\Y@c\dimen@ \advance\Y@c-\Leftness@\dimen@
8159   \else \advance\Y@c\Leftness@\dimen@ \fi
8160   \dimen@=\wd8\relax \A@=\sd@X\d@X \advance\A@-\dimen@
8161   \B@=\sd@X\dimen@ \multiply\B@\K@dYdX \divide\B@\K@
8162   \advance\B@-\d@Y \B@=\sd@Y\B@
8163   \count@=\count@@ \advance\count@\m@ne
8164   \ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi
8165   \A@=-\sd@X\A@ \B@=\sd@Y\B@ \wd8=\A@
8166   \kern\X@c \count@=\z@
8167   \loop@\ifnum\count@<\count@@ \advance\count@\@ne
8168    \raise\Y@c\copy8\relax \advance\Y@c\B@ \repeat@}%
8169 \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ {\Drop@@}}
```

The mosly vertical is analogous. **Bug:** `\dimen@` holds the unadjusted $h+d$ throughout – somehow it works anyway!

```
8176 \xydef@\straightv@{\setbox\z@=\vtop{%
8177   \setbox8=\copy\lastobjectbox@
8178   \A@=\wd8\relax \B@=\dp8\relax \advance\B@\ht8\relax
8179   \ifdim \B@=\z@ \count@@=\m@ne
8180   \else \dimen@=\sd@Y\d@Y \divide\dimen@\B@ \count@@=\dimen@ \fi
8181   \Spread@@
8182   \dimen@=\dp8\relax \advance\dimen@\ht8\relax
8183   \B@=\sd@Y\d@Y \advance\B@-\dimen@
```

```
 \A@=\sd@Y\dimen@ \multiply\A@\K@dXdY \divide\A@\K@ \advance\A@-\d@X
 \A@=\sd@X\A@ \count@=\count@@ \advance\count@\m@ne
 \ifnum\z@<\count@ \divide\B@\count@ \divide\A@\count@ \fi
 \B@=\sd@Y\B@ \A@=\sd@X\A@ \ht8=\B@ \dp8=\z@
 \ifdim\d@Y<\z@
  \advance\Y@c\dimen@ \advance\Y@c\Upness@\B@
 \else
  \advance\dimen@\Upness@\B@ \advance\Y@c-\dimen@ \advance\Y@c\B@
 \fi
 \advance\Y@c\B@
 \ifdim\d@X<\z@ \else \advance\X@c-\wd8\relax \fi
 \null \kern-\Y@c \count@=\z@
 \loop@\ifnum\count@<\count@@ \advance\count@\@ne
  \nointerlineskip \moveright\X@c\copy8\relax \advance\X@c\A@
 \repeat@}%
\ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ {\Drop@@}}
```

End & log

XY-pic ends with a message saying so.

```
\message{ Xy-pic loaded}\xyuncatcodes \endinput
```

XY-pic is maintained using the RCS "Revision Control System" by Walther F. Tichy. The following is the revision history since the first release to Usenet.

```
% $Log: xy.doc,v $
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.5  1997/05/28 13:05:01  krisrose
% Fixed missing breaks bug.
%
% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.2  1995/09/19  18:22:27  kris
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.14  1995/07/05  22:20:09  kris
% Buglets...
%
```

```
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:55:12  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.11  1994/07/05  10:37:32  kris
% Third 3beta release [bug fixes].
% Experimental graph feature included (for ECCT-94 presentation).
%
% Revision 2.10  1994/06/15  12:55:07  kris
% Second 3beta release: bug fixes.
%
% Revision 2.9  1994/06/09  14:59:19  kris
% Release 3beta.
%
% Revision 2.8  1994/04/11  09:31:09  kris
% Second (bug fix) 3alpha release [corrected].
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% MAJOR REWRITE and REORGANISATION:
% File xypic.doc split into separate files: xy.doc for ‘kernel’ and other
% files with the ‘extensions’ and ‘features’.
% Documented in special DOCMODE LaTeX-based format supported by xydoc.sty.
%
% Revision 2.6.1.1  1992/07/01  07:08:24        kris
% Send to EuroTeX ’92...
%
% Revision 2.6  1992/06/24  01:23:34  kris
% Added hooks using font xyqc10.
% Added new POSitions: * and !.
% Added triple lines \Ssolid and \Ddashed.
%
% Revision 2.5  1992/02/24  03:30:54  kris
% Fixed bugs in \Direction calculation logic...
% Added (FACTOR) to \rotate to allow arbitrary rotation.
% ‘ intermediate points now accept an optional /RADIUS argument.
% Added \Tip with wide tip.
% [See ChangeLog for further details].
%
% Revision 2.4  1992/01/22  02:15:10  kris
% Fixed bugs [with thanks]:
% No spurious arrow heads with LaTeX: \pit now undefined [Werner Struckmann]
% \Solid works: sets \Density [Dave Bowen]
```

```
% Short diagonal lines work...major rewrite of \connectv@ [Eric Domenjoud]
%
% Revision 2.3  1992/01/13  23:28:12  kris
% Swapped definitions of \ddtoX and \uutoX [found by Nico Verwer].
% Diagonal lines were wrong [Eric Domenjoud].
%
% Revision 2.2  1992/01/09  04:05:40  kris
% Still problems with rules in frames and horizontal/vertical \solids. Grrr.
%
% Revision 2.1  1992/01/02  14:54:07  kris
% Release version.
%
% Revision 1.40  1991/12/17 04:53:23 kris
% Version distributed as 'final draft' on Usenet.
```

Chapter 2

Extensions

This chapter describes the options that provide extensions to the kernel, *i.e.*, support facilities that are not obtainably using the kernel. Most such features are only approximated, relying on a driver for the implementation.

2.1 Curve and Spline extension

Vers. 3.6 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This option provides XY-pic with the ability to typeset spline curves by constructing curved connections using arbitrary directional objects and by encircling objects similarly. *Warning*: Using curves can be quite a strain on TEX's memory; you should therefore limit the length and number of curves used on a single page. Memory use is less when combined with a backend capable of producing its own curves; *e.g.*, the POSTSCRIPT backend).

Header:

```
%% $Id: xycurve.doc,v 3.6 1998/03/06 01:28:05 krisrose Exp krisrose $
%%
%% Xy-pic ``Curves and Splines'' extension.
%% Copyright (c) 1993-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
```

```
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{curve}{Curve and Spline extension}{\stripRCS$Revision: 3.6 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

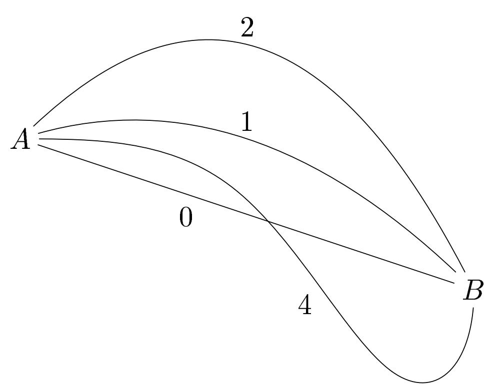
2.1.1 Curved connections

```
\message{curve,}
```

Simple ways to specify curves in XY-pic are as follows:

`**\crv{`⟨poslist⟩`}`	curved connection
`**\crvs{`⟨dir⟩`}`	get ⟨poslist⟩ from the stack
`\curve{`⟨poslist⟩`}`	as a ⟨decor⟩ation

in which ⟨poslist⟩ is a list of valid ⟨pos⟩itions. The decoration form `\curve` is just an abbreviation for `\connect\crv`. As usual, the current p and c are used as the start and finish of the connection, respectively. Within ⟨poslist⟩ the ⟨pos⟩itions are separated by `&`. A full description of the syntax for `\crv` is given in figure 2.1.



We need a counter to track the number of control points, (i.e. the number of ⟨pos⟩s in ⟨poslist⟩) and provide a macro to read it. Also a token list which will store the vital information for later use, to allow curved connections to work properly.

```
\xynew@{count}\crv@cnt@
\xydef@\xynumctrlpts@{\the\crv@cnt@}
\xynew@{toks}\crvpts@
```

If ⟨poslist⟩ is empty a straight connection is computed. When the length of ⟨poslist⟩ is one or two then the curve is uniquely determined as a single-segment Bézier quadratic or cubic spline. The tangents at p and c are along the lines connecting with the adjacent control point. With three or more ⟨pos⟩itions a cubic B-spline construction is used. Bézier cubic segments are calculated from the given control points.

```
\xydef@\setcurve@{%
  \edef\xysplineparams@{%
   \expandafter\noexpand\csname params@\endcsname}%
  \edef\xysplineedges@{%
```

```
   \expandafter\noexpand\csname edges@\endcsname}%
  \edef\xycrvcnt@{%
   \expandafter\noexpand\csname crvcnt@\endcsname}%
  \expandafter\edef\xycrvcnt@{\number\crv@cnt@}%
 \ifcase\crv@cnt@
  \expandafter\xycvxhull@\or
  \expandafter\xyquadbezier@\or
  \expandafter\xycubicbezier@\or
  \expandafter\xybspline@iii\or
  \expandafter\xybspline@iv\else
  \expandafter\xybspline@
 \fi}
```

The previous picture was typeset using:

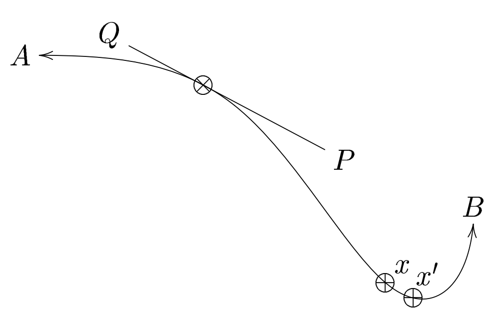
```
\xy (0,20)*+{A};(60,0)*+{B}
**\crv{}
**\crv{(30,30)}
**\crv{(20,40)&(40,40)}
**\crv{(10,20)&(30,20)&(50,-20)&(60,-10)}
\endxy
```

except for the labels, which denote the number of entries in the ⟨poslist⟩. (Extending this code to include the labels is set below as an exercise).

The `?`-operator of §1.3 (note 1.3h) is used to find arbitrary ⟨place⟩s along a curve in the usual way.

Exercise 2.1: Extend the code given for the curves in the previous picture so as to add the labels giving the number of control points.

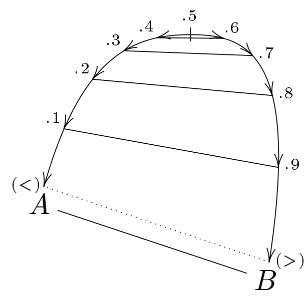
Using `?` will set the current direction to be tangential at that ⟨place⟩, and one can ⟨slide⟩ specified distances along the curve from a found ⟨place⟩ using the `?`... `/`⟨dimen⟩`/` notation:



Exercise 2.2: Suggest code to produce something like the above picture; the spline curve is the same as in the previous picture. *Hints*: The line is 140pt long and touches 0.28 of the way from A to B and the x is 0.65 of the way from A to B.

The positions in ⟨poslist⟩ specify *control points* which determine the initial and final directions of the curve—leaving p and arriving at c—and how the curve behaves in between, using standard spline constructions. In general, control points need not lie upon the actual curve.

A natural spline parameter varies in the interval $[0, 1]$ monotonically along the curve from p to c. This is used to specify ⟨place⟩s along the curve, however there is no easy relation to arc-length. Generally the parameter varies more rapidly where the curvature is greatest. The following diagram illustrates this effect for a cubic spline of two segments (3 control points).



Exercise 2.3: Write code to produce a picture such as the one above. (*Hint*: Save the locations of places along the curve for later use with straight connections.)

To have the same ⟨pos⟩ occuring as a multiple control point simply use a delimiter, which leaves the ⟨pos⟩ unchanged. Thus `\curve{`⟨pos⟩`&}` uses a cubic spline, whereas `\curve{`⟨pos⟩`}` is quadratic.

Repeating the same control point three times in succession results in straight segments to that control point. Using the default styles this is an expensive way to get straight lines, but it allows for extra effects with other styles.

Curve Objects: At present the syntax is supported for using a `\curve` object only as a decoration, constructed from a `\crv` object used as a connection.

```
\xydef@\curve{\connect\crv}
```

The more general `\crv` object currently works as a connection. In future this object-type will be extended to allow closed curves as the boundaries of objects.

```
\xydef@\crv#1#{\hbox \bgroup \crvresetbreaks@ \initshape@
 \def\pure@crv{\crv@saveStyles@}\crv@{#1}}
```

```
\xydef@\crv@#1#2{%
 \DN@{#1}\ifx\empty\next@
  \DN@{\def\afterCURVEaction@{}\parsecurve@\xycurve@@}%
 \else\DN@{\parseCURVE@#1\parsecurve@\curveSTYLE@}\fi
 \next@ #2\@endcurve\endcrv@ }
```

```
\xydef@\curveSTYLE@{\xycurve@@}%  default style
\xydef@\afterCURVEaction@{}%  default after-curve action
\xydef@\@endcurve{}%
\xydef@\endcrv@@{\endcrv@\POS}%
\xydef@\pure@crv{\relax}
```

The `\@endcurve` inserted here ensures that subsequent parsing with `\checkendcurve@` (see below) will terminate cleanly.

The `\crvs` object has restricted functionality with choice of styles and is always used for curves only. It has a single braced argument which can be used to set curve styles. Its main use is for re-typesetting different portions of the same curve, for example the subsegments of a curved arrow/path segment.

This is done by first examinining `\bstartPLACE@`. If `\relax` or undefined, then the curve is set using `\@crv@` to decide the style. If `\bstartPLACE@` is `\empty` then the curve is fully processed as a connection but nothing is actually typeset; this is used by curved arrows where the curve is first set as "invisible" before breaks and labels are processed. When `\bstartPLACE@` contains a number, normally within the range 0 to 1, this is interpreted as a ⟨place⟩ along a curved connection that has already been established. No typesetting may occur before this ⟨place⟩.

In this case the control point information is discarded since the curve can be recovered using `\splinereset@`. The current p and c are no longer the endpoints of the curve but are $\langle$pos$\rangle$itions along the curve between which the typesetting should occur. The value of `\bstartPLACE@` is typically a $\langle$place$\rangle$ along the curve which is within the $\langle$pos$\rangle$ at p. It is used to help locate the edge of this $\langle$pos$\rangle$ where typesetting should commence. Similarly `\bendPLACE@` is typically a $\langle$place$\rangle$ along the curve which is within the $\langle$pos$\rangle$ at c.

```
\xywarnifdefined\crv@normaltemplate
\xywarnifdefined\crv@othertemplate
\xywarnifdefined\crv@specialtemplate@@
{\xyuncatcodes \catcode`\@=11 \catcode`\#=6
 \gdef\crv@normaltemplate#1{{}{~**#1\xy@@crvaddstack@}}
 \gdef\crv@othertemplate#1{{}{~*=<2\jot>{}~**#1\xy@@crvaddstack@}}
 \gdef\crv@specialtemplate@@#1{{}{~**\dir{#1}\xy@@crvaddstack@}}
}

\xydef@\crvs#1#{\hbox \bgroup \def\pure@crv{\relax}\crvs@{#1}}%
\xydef@\crvi#1#{\hbox \bgroup \def\pure@crv{\relax}\invisbreaks@ \crvs@{#1}}%

\xydef@\crvs@#1{\DN@{\initshape@ \@crv@{#1}}%
 \def\crv@defaultshape{-}%
 \ifx\bstartPLACE@\relax \else
 \ifx\bstartPLACE@\empty \Invisible@true \else
\splinetrace@{bstartPLACE@=\bstartPLACE@, bendPLACE@=\bendPLACE@}%
  \DN@{\let\xy@@crvaddstack@=\xy@@samecurve@
   \let\saveshape@=\savesame@
   \let\savectrlptsnum@=\relax
   \let\startxycurve@=\recovercurve@
   \crv@cnt@=\xycrvptsnum@\relax \@crv@{#1}}%
 \fi \fi \next@ }

\xydef@\xy@@samecurve@{\xyFN@\checkendcurve@}
\xydef@\savesame@{\egroup }
\xydef@\recovercurve@{%
 \crv@cnt@=\xycrvptsnum@ \splinereset@ \recovercv@end
 \ifx\bendPLACE@\relax \def\bendPLACE@{1}\fi
 \ifdim\zz@\R@c \ifdim\zz@\L@c \ifdim\zz@\D@c \ifdim\zz@\U@c
     \Edge@c={\zeroEdge}\fi\fi\fi\fi
 \edef\cv@end{\cfromthec@}\edef\cv@start{\cfromthep@}}

\xydef@\recovercv@end{\count@=\ptsnum@\relax\advance\count@\@ne
 \edef\tmp@{\the\count@}\csname cv@\tmp@\endcsname}%

\xydef@\crvSTYLE@@{}
\expandafter\xydef@\expandafter\crv@defaultshape\expandafter{\addDASH@{}}
\xydef@\crv@noobject{\Invisible@true}

\xydef@\@crv@#1#2{\DN@{#1#2}%
 \ifx\next@\empty \edef\next@{\crv@defaultshape}%
  \ifx\bstartPLACE@\empty \xdef\crvSTYLE@@{{\crv@defaultshape}}\fi
 \else
  \ifx\bstartPLACE@\empty \gdef\crvSTYLE@@{#1{#2}}\fi
 \fi
```

```
\ifx\next@\empty \crv@noobject \DN@{\crv@{}{\xy@@crvaddstack@}}%
\else\def\tmp@{-}\ifx\next@\tmp@ \DN@{\crv@{}{\xy@@crvaddstack@}}%
\else\def\tmp@{=}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir{=}}}%
\else\def\tmp@{2-}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir{2.}}}%
\else\def\tmp@{3-}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir{3.}}}%
\else\def\tmp@{--}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@specialtemplate@{--}}%
\else\def\tmp@{==}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir2{--}}}%
\else\def\tmp@{2--}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir2{--}}}%
\else\def\tmp@{3--}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir3{--}}}%
\else\def\tmp@{.}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@specialtemplate@{.}}%
\else\def\tmp@{:}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir{:}}}%
\else\def\tmp@{2.}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@normaltemplate{\dir{:}}}%
\else\def\tmp@{..}\ifx\next@\tmp@
 \DN@{\expandafter\crv@\crv@specialtemplate@{.}}%
\else
 \DN@{\expandafter\crv@\crv@othertemplate{\dir#1{#2}}}%
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}
```

```
\xylet@\crv@specialtemplate@=\crv@specialtemplate@@
\xydef@\crv@special#1{}
```

Bug: this should use a `\Step@@` method to get the spacing for dotting; this will eliminate the need for templates. . .

Parsing: Two separate parsers are required: one for `\curve...`, the other for the contents of `{...}`.

Set `\curveSTYLE@` to be the default of `\xycurve@@` then examine the following tokens to see if this must be changed.

Set `\curveSTYLE@` to be the default of `\xycurve@@` then examine the following tokens to see if this must be changed.

Procedure: If the first token of `#1` (from `\crv@`) is `~` then a letter should follow, determining how to set the style. If instead it is an active control sequence then issue a warning message, but let it do its thing anyway. However `\parsecurve@` is a normal thing to encounter, so no warning is required. Currently a single letter without the `~` will be recognised, but two-letter combinations definitely need the `~`.

```
\xydef@\parseCURVE@{\def\curveSTYLE@{\xycurve@@}\def\afterCURVEaction@{}%
 \xyFN@\parseCURVE@@}
\xydef@\parseCURVE@@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\parseCURVE@@}% gobble spaces
 \else\ifx ~\next \DN@ ~{\xy@~{}\xyFN@\setcurveSTYLE@@}%
 \else\ifx\parsecurve@\next\DN@{}%
```

Syntax			Action
`\curve`⟨modifier⟩`{`⟨curve-object⟩⟨poslist⟩`}`			construct curved connection
⟨modifier⟩	⟶	⟨empty⟩	zero or more modifiers possible; default is `~C`
	|	`~`⟨curve-option⟩ ⟨modifier⟩	set ⟨curve-option⟩
⟨curve-option⟩	⟶	`p` | `P` | `l` | `L` | `c` | `C`	show only[2.1d] control points (`p`=points), joined by lines (`l`=lines), or curve only (`c`=curve)
	|	`pc` | `pC` | `Pc` | `PC`	show control points[2.1f] and curve[2.1e]
	|	`lc` | `lC` | `Lc` | `LC`	show lines joining[2.1g] control points and curve[2.1e]
	|	`cC`	plot curve twice, with and without specified formatting
⟨curve-object⟩	⟶	⟨empty⟩	use the appropriate default style
	|	`~*`⟨object⟩ ⟨curve-object⟩	specify the "drop" object[2.1a] and maybe more[2.1c]
	|	`~**`⟨object⟩ ⟨curve-object⟩	specify "connect" object[2.1b] and maybe more[2.1c]
⟨poslist⟩	⟶	⟨empty⟩ | ⟨pos⟩ ⟨delim⟩ ⟨poslist⟩	list of positions for control points
	|	`~@` | `~@` ⟨delim⟩ ⟨poslist⟩	add the current stack[2.1h] to the control points
⟨delim⟩	⟶	`&`	allowable delimiter

Figure 2.1: Syntax for curves.

```
\else\ifcat\active\noexpand\next\DN@{\xyactive@Warning}%
\else\ifcat a\noexpand\next\DN@##1{\xytilde@Warning\setcurveSTYLE@{##1}%
 \xyFN@\parseCURVE@@}%
\else\DN@{}\fi\fi\fi\fi\fi \next@ }%
```

Options are normally specified using `~`. Sometimes this can be safely omitted; if such a case is encountered then a warning message is issued advising to use `~`. This nicety may be removed in future versions.

```
\xydef@\xyactive@Warning{\xywarning@{expandable control sequences should
 not be here, results are unpredictable in diagrams.}}%     (2)
\xydef@\xytilde@Warning{Please use ~ to set curve style.}%  (3a)
```

Examine the next token to see if it determines a curve style. Currently only `p`, `l` and `c` are recognised, along with the uppercase variants `P`, `L` and `C`. When one of these letters is encountered, set `\curveSTYLE@` and keep examining tokens with `\setafterCURVEaction@`.

```
\xydef@\setcurveSTYLE@#1{\xyFN@\setcurveSTYLE@@#1}
\xydef@\setcurveSTYLE@@{%
 \ifx\next p\def\curveSTYLE@{\xyc@trlpts@@}%
   \DN@ p{\xyFN@\setafterCURVEaction@}%
 \else\ifx\next P\def\curveSTYLE@{\xyc@trlpts@}%
   \DN@ P{\xyFN@\setafterCURVEaction@}%
 \else\ifx\next l\def\curveSTYLE@{\xyc@vxhull@@}%
   \DN@ l{\xyFN@\setafterCURVEaction@}%
 \else\ifx\next L\def\curveSTYLE@{\xyc@vxhull@}%
   \DN@ L{\xyFN@\setafterCURVEaction@}%
 \else\ifx\next c\def\curveSTYLE@{\xycurve@@}%
```

```
   \DN@ c{\xyFN@\setafterCURVEaction@}%
  \else\ifx\next C\def\curveSTYLE@{\xycurve@@}%
   \DN@ C{\xyFN@\setafterCURVEaction@}%
  \else \DN@##1{\xywarning@{unknown curve style ##1}\xyFN@\parseCURVE@@}%
  \fi\fi\fi\fi\fi\fi \next@ }
```

`\setafterCURVEaction@` examines tokens to see if a 2-letter combination is being specified; e.g. `pc`, `lC`, etc. Alternatively another `~` can set a new specification.

```
\xydef@\setafterCURVEaction@{%
 \ifx\next~\DN@~{\xyFN@\setcurveSTYLE@@}%
 \else\ifcat a\noexpand\next\DN@{\setafterCURVEaction@@}%
 \else\DN@{\xyFN@\parseCURVE@@}%
 \fi\fi \next@}
```

Currently the only 2-letter specifications have either `c` or `C` as the 2nd letter.

```
\xydef@\setafterCURVEaction@@{%
 \ifx\next c\DN@ c{%
  \def\afterCURVEaction@{\noexpand\endcurve\noexpand\xy@curve@}%
   \xyFN@\parseCURVE@@ }%
 \else\ifx\next C\DN@ C{%
  \def\afterCURVEaction@{\noexpand\endcurve\noexpand\xy@curve@@}%
   \xyFN@\parseCURVE@@ }%
 \else \DN@##1{\xywarning@{unknown after-curve action ##1}%
  \xyFN@\parseCURVE@@ }%
 \fi\fi \next@ }
```

When `\parsecurve@` is encountered, this signifies the end of this part of the parsing. The only active control sequence that should be encountered legitimately here is `\@endcurve`, which signifies that the ⟨poslist⟩ is empty and default styles are required. Otherwise assume we the subsequent tokens are from `#2` of `\crv@`. Control passes to `\xycurve@@` to prepare for reading the ⟨crv-object⟩s and ⟨poslist⟩.

```
\xydef@\parsecurve@{\xyFN@\parsecurve@@}
\xydef@\parsecurve@@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\parsecurve@@}%
 \else\ifx\next\@endcurve\DN@\@endcurve{\checkafterCURVE@}%
 \else\ifcat\active\noexpand\next\DN@{\relax}%
 \else\DN@{\xycurve@@}\fi\fi\fi\next@}%
```

```
\xydef@\xycurve@{\begingroup\afterCURVE{\setcurve@\endgroup}\crvobjects@
 \startxycurve@\xycrvmods@}
\xydef@\xycurve@@{\afterCURVE{\setcurve@}\crvobjects@\startxycurve@
 \xycrvmods@}
\xydef@\xy@curve@{\xy@curve@@@{\splinedefaulttol@\crvobjects@}}
\xydef@\xy@curve@@{\xy@curve@@@{\resetcrvobjects@}}
\xydef@\xy@curve@@@#1{\ifx\cv@start\relax
  \DN@{\xywarning@{There is no curve to plot}}.
 \else\DN@{\begingroup\afterCURVE{\setcurve@\endgroup}#1\cv@end}%
 \fi \next@ }
\xydef@\crvobjects@{\def\xycrvdrop@{}\def\xycrvconn@{}}
\xydef@\resetcrvobjects@{\crvobjects@}
```

These were originally provided for sophisticated-user access. They are otherwise undocumented and may be removed.

```
613 \xylet@\savecurve=\xycurve@@
614 \xylet@\samecurve=\xy@curve@
```

`\xy@crvmods@` handles reading the ⟨curve-object⟩s, *i.e.* the "drop" object and the "connect" object.

Procedure: l.635 Get next token. l.639 Is it a ~? l.640 If not, exit. l.645 If so, is it followed by a *? l.647 If not, issue a message and go back to l.635. Otherwise, l.652 is there a second *? l.653 If not then read and store the "drop" object, l.661 else, read and store the "connect" object. When finished, in either case, l.659 look for further specifications. l.657 An empty "drop" object is not allowed, so this actually gives the default of `\zerodot`.

```
635 \xydef@\xycrvmods@{\xyFN@\xy@crvmods@}

637 \xydef@\xy@crvmods@{%
638  \ifx\space@\next\expandafter\DN@\space{\xyFN@\xy@crvmods@}%gobble spaces
639  \else\ifx\next ~\DN@ ~{\xyFN@\xy@crvmods@@}
640  \else\DN@{\xy@crvmods@@@@}\fi\fi \next@}
641 \xydef@\xy@crvmods@@@@{\xyFN@\checkendcurve@}%

643 \xydef@\xy@crvmods@@{%
644  \ifx\space@\next\expandafter\DN@\space{\xyFN@\xy@crvmods@@}%gobble spaces
645  \else\ifx\next *\DN@ *{\xyFN@\xy@@crvmods@}
646  \else\addAT@\ifx\next\addAT@\DN@{\xyFN@\xy@@crvaddstack}%
647  \else\DN@{\xywarning@{badly-formed curve option}\xyFN@\xy@crvmods@}
648  \fi\fi\fi \next@}%

650 \xydef@\xy@@crvmods@{%
651  \ifx\space@\next\expandafter\DN@\space{\xyFN@\xy@@crvmods@}%gobble spaces
652  \else\ifx\next *\DN@ *{\xyFN@\xy@@crvmods@@}
653  \else\DN@{\xy@crv@mods@@}\fi\fi \next@}

655 \xydef@\xy@crv@mods@@#1#{\DN@##1{\xy@crvmods@@@{#1}{##1}}\next@}%
656 \xydef@\xy@crvmods@@@#1#2{%
657  \DN@{#1#2}\ifx\next@\empty \def\xycrvdrop@{ #1{\zerodot}}
658  \else \def\xycrvdrop@{ #1{#2}}\fi
659  \xyFN@\xy@crvmods@ }

661 \xydef@\xy@@crvmods@@#1#{\DN@##1{\xy@@crvmods@@@{#1}{##1}}\next@}
662 \xydef@\xy@@crvmods@@@#1#2{\def\xycrvconn@{ #1{#2}}\xyFN@\xy@crvmods@ }%
```

Parsing of the ⟨poslist⟩ is quite simple, recognising few special tokens:

- `\endcurve` and `\endxy` terminate reading of control points.
- `&`, `\\` and `\relax` delimit ⟨pos⟩itions
- anything else is treated as a ⟨pos⟩, being handled by the `\POS@` parser.

```
680 \xydef@\checkendcurve@{%
681  \ifx\space@\next\expandafter\DN@\space{\xyFN@\checkendcurve@}%gobble spaces
682  \else\ifx\next\endcurve\DN@\endcurve{\clearsearchflag@\endxycurve@}%
683  \else\ifx\next\@endcurve\DN@\@endcurve{\checkafterCURVE@}%
684  \else\ifx\next\xy@@crvaddstack@\DN@{}%
685  \else\ifx\next\endxy\DN@{\clearsearchflag@\sloppyendcrv@}%
```

```
% \else\addAND@\ifx\next\addAND@\DN@{\clearsearchflag@\searchnextpt@}%
 \else\testAND@\iffalse \DN@##1{\clearsearchflag@\searchnextpt@}%
 \else\ifx ~\next\DN@ ~{\xyFN@\checkcrvtilde@}%
 \else\ifx\next\relax\DN@\relax{\clearsearchflag@\searchnextpt@}%
 \else\ifx\next\\\DN@\\{\clearsearchflag@\searchnextpt@}%
 \else\ifx\searchflag@\relax
   \DN@{\setsearchflag@\searchnextpt@}%
   \else\DN@##1{\flagwarn@{##1}}\fi
 \fi\fi\fi\fi\fi\fi\fi\fi\fi \next@ }
\xydef@\searchnextpt@{\afterPOS{\xy@@\addtocrv@ \xyFN@\checkendcurve@}}
```

There could be a problem in that an invalid token would cause an infinite loop, passing back and forth between `\checkendcurve@` and `\POS@`. This is avoided by setting a flag `\setsearchflag@` when the `\POS@` parser is called, not following a valid delimiter. Encountering a valid delimiter clears the flag. If `\checkendcurve@` is called with the flag set, a warning message is issued and the token is skipped; the flag remains set. Processing continues, but it is possible that the wrong number of positions will be read; presumably there is some kind of error that needs to be fixed anyway.

```
\xydef@\checkafterCURVE@{%
 \edef\tmp@{\noexpand\xyFN@\noexpand\checkendcurve@ %
  \afterCURVEaction@\noexpand\endcurve}\tmp@ }
```

```
\xylet@\searchflag@=\relax
\xydef@\setsearchflag@{\let\searchflag@=@}
\xydef@\clearsearchflag@{\let\searchflag@=\relax}
\xydef@\flagwarn@#1{%
 \ifx\next\@endcurve\DN@{\xyFN@\checkendcurve@\endcurve}%
 \else\setsearchflag@\DN@{\xyFN@\checkendcurve@}%
  \xywarning@{skipping invalid token \string#1\space in curve}%
 \fi \next@ }
```

It is possible for the final `\endcurve` to be omitted, but only when the next token is `\endxy`. A warning message is written to the log file, protesting against this sloppy TeX-ing.☺

```
\xydef@\sloppyendcrv@{%
\xywarning@{Please use \string\endcurve\space to end the curve.}\endxycurve@}
```

Creation of the `\crv` is completed when `\endcrv@` is encountered. At this point the usual methods `\Drop@@` and `\Connect@@` are defined. Extra information is retained, using `\saveshape@` and `\savecrvparams@` for the benefit of methods which treat the curve as a connection.

```
\xydef@\endcrv@{\saveshape@ %\def\Upness@{0}%
 \Edge@c={\splineEdge}\Invisible@false\Hidden@false
 \edef\Drop@@{\noexpand\ifInvisible@ \setbox\z@=\copy\voidb@x
  \noexpand\else
  \noexpand\wdz@=\z@
  \noexpand\ht\z@=\z@ \noexpand\dp\z@=\z@
  \noexpand\splineDrop@
 \noexpand\fi }%
%
 \edef\Connect@@{%
  \noexpand\setbox\z@=\noexpand\box\lastobjectbox@
  \noexpand\wdz@=\z@
  \noexpand\ht\z@=\z@ \noexpand\dp\z@=\z@
```

```
  \expandafter\noexpand\pure@crv
  \noexpand\ifInvisible@ \noexpand\crv@saveStyles@ \noexpand\else
  \noexpand\crv@getStyles@\noexpand\styledboxz@ \noexpand\fi
%
  \noexpand\ifHidden@\noexpand\else
  \noexpand\ifx\noexpand\COORD@\noexpand\xymatrixCOORD@ \noexpand\else
   \noexpand\X@min=\the\X@min \noexpand\X@max=\the\X@max
   \noexpand\Y@min=\the\Y@min \noexpand\Y@max=\the\Y@max
  \noexpand\fi \noexpand\fi
  \savecrvparams@ \noexpand\crvconnect@ %\noexpand\initbreaks@
 }%
 \wdz@=\z@
 \ht\z@=\z@ \dp\z@=\z@ }
```

The control names `\crvXY@pre@` and `\crvXY@post@` are used to transfer style information passed with `\preconnect@` to subsegments.

```
\xydef@\crvXY@pre@{}
\xydef@\crvXY@post@{}
```

```
\xydef@\crv@saveStyles@{\def\xy@style@{}%
 \expandafter\def\expandafter\crvXY@pre@\expandafter{\preXY@style@}%
 \expandafter\def\expandafter\crvXY@post@\expandafter{\postXY@style@}}
```

```
\xydef@\crv@getStyles@{\def\xy@style@{}%
 \gdef\preXY@style@{\crvXY@pre@}\gdef\postXY@style@{\crvXY@post@}}
```

The values of `\X@min`, `\X@max`, *etc.* are used to compute the extents `\L@c`, `\R@c`, *etc.* of the object. This information must be maintained after the group is closed, unless the “hidden” attribute is required or we are inside a matrix construction.

In calculating the size of the box containing the curve `\X@min`, `\X@max`, *etc.* are initialised to describe the minimum rectangle enclosing p and c. At the same time we save the current scope.

```
\xydef@\initshape@{\savescope@
 \ifdim\X@c>\X@p \X@min=\X@p \X@max=\X@c \else \X@max=\X@p \X@min=\X@c \fi
 \ifdim\Y@c>\Y@p \Y@min=\Y@p \Y@max=\Y@c \else \Y@max=\Y@p \Y@min=\Y@c \fi }
```

`\saveshape` is used to pass necessary information up one level of grouping.

```
\xydef@\saveshape@{\savectrlptsnum@
 \R@c=\X@max \advance\R@c-\X@c \L@c=\X@c \advance\L@c-\X@min
 \U@c=\Y@max \advance\U@c-\Y@c \D@c=\Y@c \advance\D@c-\Y@min
 \edef\tmp@{\egroup \U@c=\the\U@c \D@c=\the\D@c \L@c=\the\L@c \R@c=\the\R@c
 \noexpand\ifdim\noexpand\X@min>\the\X@min\noexpand\X@min=\the\X@min\noexpand\fi
 \noexpand\ifdim\noexpand\Y@min>\the\Y@min\noexpand\Y@min=\the\Y@min\noexpand\fi
 \noexpand\ifdim\noexpand\X@max<\the\X@max\noexpand\X@max=\the\X@max\noexpand\fi
 \noexpand\ifdim\noexpand\Y@max<\the\Y@max\noexpand\Y@max=\the\Y@max\noexpand\fi
 \savecrvparams@ \def\noexpand\pure@crv{\expandafter\noexpand\pure@crv}}%
 \tmp@ }
```

```
\xydef@\savecrvparams@{\crvpts@={\the\crvpts@}%
 \noexpand\def\noexpand\crvconnect@{\expandafter\noexpand\crvconnect@}}%
```

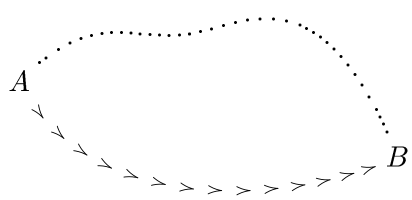
```
\xydef@\splineEdge{\rectangleEdge}
```

To Do: This can be improved. For example, trace along the spline until an appropriate point is found. There may be more than one such point, so extra criteria may be required.

Notes

2.1a. The “drop” object is set once, then “dropped” many times at appropriately spaced places along the curve. If directional, the direction from p to c is used. Default behaviour is to have tiny dots spaced sufficiently closely as to give the appearance of a smooth curve. Specifying a larger size for the “drop” object is a way of getting a dotted curve (see the example in the next note).

2.1b. The “connect” object is also dropped at each place along the curve. However, if non-empty, this object uses the tangent direction at each place. This allows a directional object to be specified, whose orientation will always match the tangent. To adjust the spacing of such objects, use an empty “drop” object of non-zero size as shown here:




```
\xy (0,0)*+{A}; (50,-10)*+{B}
**\crv{~*=<4pt>{.} (10,10)&(20,0)&(40,15)}
**\crv{~*=<8pt>{}~**!/-5pt/\dir{>}(10,-20)
 &(40,-15)} \endxy
```

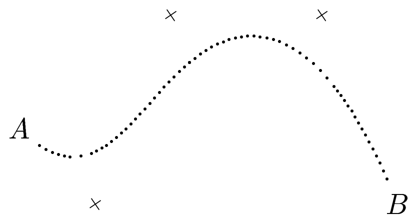
When there is no “connect” object then the tangent calculations are not carried out, resulting in a saving of time and memory; this is the default behaviour.

2.1c. The “drop” and “connect” objects can be specified as many times as desired. Only the last specification of each type will actually have any effect. (This makes it easy to experiment with different styles.)

2.1d. Complicated diagrams having several spline curves can take quite a long time to process and may use a lot of TEX’s memory. A convenient device, especially while developing a picture, is to show only the location of the control points or to join the control points with lines, as a stylized approximation to the spline curve. The ⟨curve-option⟩s `~p` and `~l` are provided for this purpose. Uppercase versions `~P` and `~L` do the same thing but use any ⟨curve-object⟩s that may be specified, whereas the lowercase versions use plain defaults: small cross for `~p`, straight line for `~l`. Similarly `~C` and `~c` set the spline curve using any specified ⟨curve-option⟩s or as a (default) plain curve.

2.1e. Use of `~p`, `~l`, etc. is extended to enable both the curve and the control points to be easily shown in the same picture. Mixing upper- and lower-case specifies whether the ⟨curve-option⟩s are to be applied to the spline curve or the (lines joining) control points. See the examples accompanying the next two notes.

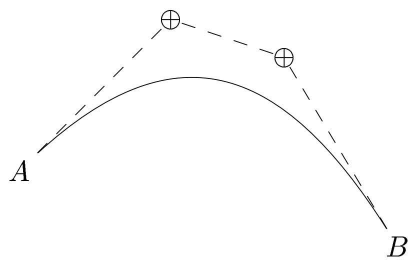
2.1f. By default the control points are marked with a small cross, specified by `*\dir{x}`. The “connect” object is ignored completely.



was typeset by . . .

```
\xy (0,0)*+{A};(50,-10)*+{B}
**\crv~pC{~*=<\jot>{.}(10,-10)&(20,15)
 &(40,15)} \endxy
```

2.1g. With lines connecting control points the default "drop" object is empty, while the "connect" object is `\dir{-}` for simple straight lines. If non-empty, the "drop" object is placed at each control point. The "connect" object may be used to specify a fancy line style.



was typeset by . . .

```
\xy (0,0)*+{A};(50,-10)*+{B}
**\crv~Lc{~**\dir{--}~*{\oplus}
 (20,20)&(35,15)} \endxy
```

2.1h. When a stack of ⟨pos⟩itions has been established using the `@i` and `@+` commands, these positions can be used and are appended to the ⟨poslist⟩.

```
\xydef@\xy@@crvaddstack{%
 \ifx\next ~\xywarning@{there should be a poslist here, \string~@ ignored}%
  \DN@{\xy@crvmods@}%
 \else\DN@{\xy@@crvaddstack@}\fi \next@}

\xydef@\xy@@crvaddstack@{\smapxy@@\addtocrv@ \xyFN@\checkendcurve@}

\xydef@\checkcrvtilde@{%
 \addAT@\ifx\next\addAT@\DN@{\xy@@crvaddstack@}%
 \else\xywarning@{invalid token after ~, ignored}%
  \DN@##1{\xyFN@\checkendcurve@}%
 \fi \next@ }
```

Spline Methods Each `\curve` has to set the sliding abilities. This is done by a call to `\crvconnect@@` which gives values to the methods `\Creset@@`, `\Cshavep@`, etc. These values depend on the style of the curve itself.

```
\xydef@\crvconnect@@{%
  \def\Creset@@{\crvreset@}%
  \def\Cshavep@@{\crvshavep@}%
```

```
  \def\Cshavec@@{\crvshavec@}%
  \def\Cslidep@@{\crvslidep@}%
  \def\Cslidec@@{\crvslidec@}%
  \def\Calong@@{\crvalong@}%
  \def\Cintercept@@{\crvintercept@}%
  \def\Cbreak@@{\crvbreak@}%
  \def\Clast@@{\crvlast@}%
}
```

When straight lines are required …

```
\xydef@\straitconnect@{\crvconnect@@
  \let\crvreset@=\straitreset@
  \let\crvshavep@=\straitshavep@
  \let\crvshavec@=\straitshavec@
  \let\crvslidep@=\straitslidep@
  \let\crvslidec@=\straitslidec@
  \let\crvalong@=\straitalong@
  \let\crvintercept@=\straitintercept@
  \let\crvbreak@=\straitbreak@
  \let\crvlast@=\straitlast@
}
```

These are the actual methods used for a sequence of straight segments.

```
\xydef@\straitreset@{\the\crvpts@
 \edef\xycrvptsnum@{\csname ptsnum@\endcsname}%
 \def\segmentnum@{0}}
\xydef@\straitslidep@{\setupDirection@ii \noCslidep@@ }
\xydef@\straitslidec@{\setupDirection@ii \noCslidec@@ }

\xydef@\straitshavep@{%
 \expandafter\ifnum\xycrvptsnum@>\z@
  \def\segmentnum@{0}%
  \csname cv@1\endcsname
  \resetDirection@ \setupDirection@ii
 \fi \noCshavep@@ }

\xydef@\straitshavec@{%
 \expandafter\ifnum\xycrvptsnum@>\z@
  \expandafter\count@@\xycrvptsnum@\relax
  \bgroup \csname cv@\number\count@@\endcsname
  \edef\tmp@{\egroup \X@p=\the\X@c \Y@p=\the\Y@c}\tmp@
  \edef\segmentnum@{\xycrvptsnum@}%
  \resetDirection@ \setupDirection@ii
 \fi \noCshavec@@ }

\xydef@\straitalong@#1{%
 \ifdim#1\p@<\z@
  \xywarning@{parameter value #1 is negative, using 0}%
  \cfromp@ \def\PLACEf@{{0}}%
 \else
  \dimen@=#1\p@ \count@=\dimen@ \divide\count@\p@
  \expandafter\count@@\xycrvptsnum@\relax \advance\count@@\@ne
```

```
  \advance\dimen@\segmentnum@\p@
  \advance\count@\segmentnum@\relax
   \def\segmentnum@{0}%
  \ifnum\count@@<\count@
   \xywarning@{parameter value #1 too large, using \the\count@@}%
   \count@=\count@@ \advance\count@\m@ne \edef\PLACEf@{{1}}%
  \else
   \advance\dimen@-\count@\p@
   \ifdim\zz@\dimen@\ifnum\count@>\z@
    \advance\count@\m@ne \advance\dimen@\p@ \fi\fi
   \edef\PLACEf@{{\expandafter\removePT@\the\dimen@}}%
  \fi
  \ifnum\count@=\z@\else \bgroup
   \csname cv@\number\count@\endcsname
   \edef\tmp@{\egroup \X@p=\the\X@c \Y@p=\the\Y@c}\tmp@
  \fi
  \advance\count@\@ne
  \ifnum\count@<\count@@
\splinetrace@{find place \the\dimen@\space along straight segment
 \the\count@}%
   \csname cv@\number\count@\endcsname
  \fi
  \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
 \fi \expandafter\noCalong@@\PLACEf@ \czeroEdge@ }
```

```
\xydef@\straitintercept@{\noCintercept@@}
\xydef@\straitbreak@{\straightCbreak@@}
\xydef@\straitlast@{\straightClast@@}
```

When curved segments are required . . .

```
\xydef@\splineconnect@{%
  \let\crvreset@=\splinereset@
  \let\crvshavep@=\splineshavep@
  \let\crvshavec@=\splineshavec@
  \let\crvslidep@=\splineslidep@
  \let\crvslidec@=\splineslidec@
  \let\crvalong@=\splinealong@
  \let\crvintercept@=\splineintercept@
  \let\crvbreak@=\splinebreak@
  \let\crvlast@=\splinelast@
}
```

These are the actual methods.

Reset the spline parameters:

`\xycrvptsnum@`: expands to a control sequence, whose name incorporates the correct scoping level, giving the number of control points.

`\the\crvpts@`: reads the control point locations and other information from a token list.

`\xysplineparams@`: set to a control sequence name, incorporating the correct scoping level, which will allow access to the Beźier control points for each spline segment, when required.

`\xysplineedges@`: set to a control sequence name, incorporating the correct scoping level, which gives the spline parameters of the edges of objects at p and c;

`\segmentnum@`: reset to 0.

`\splinecorrect@p` and `\splinecorrect@c`: reset to 0 until changed by a `\Cshavep@` or `\Cshavec@`.

`\splinelength@`: reset to 0 until changed by a `\Cshavep@`, `\Cshavec@` or `\Calong@`; this is needed by `\Cshavep@` and `\Cshavec@` to quickly locate where to slide from, especially when it is necessary to slide across more than a single spline segment.

`\splineplace@`: reset to .5.

```
\xydef@\splinereset@{\the\crvpts@
 \edef\xycrvptsnum@{\csname ptsnum@\endcsname}%
 \edef\xysplineparams@{%
  \expandafter\noexpand\csname params@\endcsname}%
 \edef\xysplineedges@{%
  \expandafter\noexpand\csname edges@\endcsname}%
 \def\segmentnum@{1}\def\splinecorrect@p{0}\def\splinecorrect@c{0}%
 \splinelength@=\z@ \def\splineplace@f{.5}%
}
```

Shaving to the appropriate edge. This is simply a matter of reading the stored edge information and storing the appropriate value in `\splinecorrect@p` or `\splinecorrect@c`. The difficult part is to construct the correct control sequence name and then to extract the correct part of the text in its expansion.

```
\xydef@\splineshavep@{\begingroup
  \edef\xysplineedges@{%
   \expandafter\noexpand\csname edges@\endcsname}%
 \expandafter\expandafter\expandafter\getsplineedges@safe\xysplineedges@ @@%
 \edef\tmp@{\noexpand\removePT@\the\dimen5}%
 \edef\tmp@{\endgroup
 \noexpand\def\noexpand\splinecorrect@p{\tmp@}}\tmp@
 \setupDirection@ii }
```

```
\xydef@\splineshavec@{\begingroup
 \edef\xysplineedges@{%
  \expandafter\noexpand\csname edges@\endcsname}%
 \expandafter\expandafter\expandafter\getsplineedges@safe\xysplineedges@ @@%
 \dimen@=-\dimen7\relax \advance\dimen@\p@
%% \getcrvsegsnum@ \advance\dimen@\count@\p@
 \edef\tmp@{\noexpand\removePT@\the\dimen@}%\W@{**shave**<\tmp@>***}%
 \edef\tmp@{\endgroup
  \noexpand\def\noexpand\splinecorrect@c{\tmp@}}\tmp@
 \setupDirection@ii }
```

```
\xydef@\splineslidep@#1{\enter@{\pfromthep@}%
 \begingroup \splinealong@@{0}{#1}+\splineslidep@@@
  \setupDirection@ii \leave@ }%
```

```
\xydef@\splineslidep@@@{%
 \d@X=\dimen5 \d@Y=\dimen7 \edef\tmp@{\postfind@}\tmp@ \dimen@=\splineval@
```

```
 \expandafter\count@\xycrvptsnum@\relax
 \ifnum\count@>\tw@ \advance\count@\m@ne
  \advance\dimen@\segmentnum@\p@ \advance\dimen@-\p@
 \fi
 \edef\tmp@{\endgroup \endgroup
  \noexpand\def\noexpand\xysplineval@{\the\splineval@}%
  \noexpand\def\noexpand\splinecorrect@p{\expandafter\removePT@\the\dimen@}%
\splinetrace@{slide = \the\splinelength@}%
  \X@c=\dimen@i \Y@c=\dimen3  % something may be wrong here!!
  \d@X=\the\d@X \d@Y=\the\d@Y %
  \noexpand\def\noexpand\segmentnum@{\segmentnum@}}\tmp@ }

\xydef@\splineslidec@#1{\enter@{\pfromthep@}%
 \begingroup
  \expandafter\splinealong@@\PLACEf@{#1}+\splineslidec@@@
  \setupDirection@ii \leave@ }

\xydef@\splineslidec@@@{%
 \d@X=\dimen5 \d@Y=\dimen7 \edef\tmp@{\postfind@}\tmp@ \dimen@=\splineval@
 \expandafter\count@\xycrvptsnum@\relax
 \ifnum\count@>\tw@ \advance\count@\m@ne
  \advance\dimen@\segmentnum@\p@ \advance\dimen@-\p@
  \dimen@=-\dimen@ \advance\dimen@\count@\p@
 \else \dimen@=-\dimen@ \advance\dimen@\p@ \fi %\W@{**slide**<\the\dimen@>***}%
%%  splinecorrect@c  is probably wrong here!!
 \edef\tmp@{\endgroup \endgroup
  \noexpand\def\noexpand\xysplineval@{\the\splineval@}%
  \noexpand\def\noexpand\splinecorrect@c{\expandafter\removePT@\the\dimen@}%
\splinetrace@{slide = \the\splinelength@; correct@c=\noexpand\splinecorrect@c}%
  \X@c=\dimen@i \Y@c=\dimen3  %
  \d@X=\the\d@X \d@Y=\the\d@Y %
  \noexpand\def\noexpand\segmentnum@{\segmentnum@}}\tmp@ }

\xydef@\checkslidemore@#1{%
 \dimen@ii=-#1\relax\ifdim\dimen@ii<\z@\multiply\dimen@ii\m@ne\fi
 \dimen@=-\splinelength@ \advance\dimen@\dimen@ii\relax
 \ifdim\dimen@<\z@\dimen@ii=-\dimen@\else\dimen@ii=\dimen@\fi
 \ifdim\dimen@ii<.2\p@\DN@{\splineslidec@@ \leave@}%%%% ?
 \else \splinetrace@{not found yet, \the\dimen@ii\space still to go}%
 \dimen@ii=#1\relax\ifdim\dimen@ii<\z@\multiply\dimen@\m@ne\fi
 \DN@{\expandafter\splineslidemore@\expandafter{\the\dimen@}}\fi
 \next@}%

\xydef@\splineslidemore@#1{%
 \expandafter\count@\segmentnum@\relax
 \ifdim#1<\z@\advance\count@\m@ne\dimen@=\p@
 \else\advance\count@\@ne\dimen@=\z@\fi
 \ifnum\count@<\@ne
   \expandafter\splinesegment@\expandafter{\segmentnum@}%
  \DN@{\xywarning@{cannot slide beyond start of curve}%
   \spline@find{\z@}{\z@}{}\splineslidec@@ \leave@ }%%%% ?
 \else \expandafter\count@@\xycrvptsnum@\relax\advance\count@@\m@ne
```

```
 \expandafter\ifnum\count@@<\count@\relax
  \expandafter\splinesegment@\expandafter{\segmentnum@}%
 \DN@{\xywarning@{cannot slide beyond end of curve}%
 \spline@find{\p@}{\z@}{}\splineslidec@@ \leave@ }%%%%% ?
\else
 \edef\segmentnum@{\the\count@}%
  \expandafter\splinesegment@\expandafter{\segmentnum@}%
\splinetrace@{sliding onto segment \segmentnum@}%
 \expandafter\spline@find\expandafter{\the\dimen@}{#1}{}%
 \DN@{\checkslidemore@{#1}}%
\fi\fi  \next@ }

\xydef@\splineslidec@@{%
\dimen@=\splineval@ \advance\dimen@\segmentnum@\p@\advance\dimen@-\p@
\edef\tmp@{\endgroup
 \noexpand\def\noexpand\xysplineval@{\the\dimen@}%
\splinetrace@{slide = \the\splinelength@}%
 \X@c=\the\dimen@i \Y@c=\the\dimen3 \d@X=\the\dimen5 \d@Y=\the\dimen7 \postfind@
 \noexpand\def\noexpand\segmentnum@{\segmentnum@}}\tmp@
 \setupDirection@ii}
```

The method `\splinealong@#1` finds places along the curve, when `#1` is given as a factor, normally between 0 and 1. The location of the edges of the objects at p and c are stored in `\splinecorrect@p` and `\splinecorrect@c`. These are the places found if the factor is either 0 or 1. If an edge is non-zero then it is actually possible to move to points inside that edge by specifying a factor less than 0 or greater than 1.

```
\xydef@\splinealong@#1{\def\postfind@{}\splinealong@@{#1}\z@+%
\splineslidec@@
\dimen@=\xysplineval@\relax
\count@=\xycrvptsnum@\relax \ifnum\count@>\@ne \advance\count@\m@ne\fi
\ifnum\count@>\@ne \dimen@nth\count@\dimen@
\fi \edef\bthisPLACE@{\expandafter\removePT@\the\dimen@}%
\splinetrace@{bthisPLACE@=\bthisPLACE@}}%
```

The parameter `#2` in `\splinealong@@#1#2#3` allows for a subsequent slide through a given ⟨dimen⟩ along the curve, from the place specifed by the factor in `#1`.

The information supplied in `#1` must be converted to the segment number and parameter value of the specified point. These will be calculated and stored temporarily in `\count@` and `dimen@` before the values are passed respectively to `\splinesegment@` which chooses the segment, and `\spline@find` to locate the point on this segment.

To do the calculation, first we correct for the parameter values of the edges of the objects at the end-points, which are stored in `\splinecorrect@p` and `\splinecorrect@c`. If the resulting number x is not an integer then `\count@` is set to $1 + \lfloor x \rfloor$ while `\dimen@` becomes $(x - \lfloor x \rfloor)$`\p@`. If x is an integer: $x = 0$ gives `\count@=1` and `\dimen@=\z@`, otherwise `\count@` $= x$ and `\dimen@=\p@`.

```
\xydef@\splinealong@@#1#2#3{%
\edef\tmp@{#1}\relax \dimen@ii\tmp@\p@
%% \getcrvsegsnum@
%% \dimen@=-\count@\p@
\dimen@=-\p@
\expandafter\advance\expandafter\dimen@\splinecorrect@p\p@
\expandafter\advance\expandafter\dimen@\splinecorrect@c\p@
```

```
 \edef\tmp@{#1}\relax \dimen@=-\tmp@\dimen@ \relax
 \expandafter\advance\expandafter\dimen@\splinecorrect@p\p@
 \getcrvsegsnum@ \multiply \dimen@ by\count@\relax
\splinetrace@{actual:\the\dimen@; f:\the\dimen@ii; p:\splinecorrect@p, c:\splinecorrect@c}%
 \count@=\dimen@ \divide\count@\p@ \DN@{\relax}%
 \def\tmp@{\relax}\ifx\next@\tmp@
  \expandafter\ifnum\xycrvptsnum@<\count@
   \xywarning@{parameter value #1 too large}%
   \DNii@{\relax}\DN@{\begingroup}%
  \else
   \edef\xysplineval@{\the\dimen@}%
   \advance\dimen@-\count@\p@
   \ifdim\dimen@=\z@
\splinetrace@{find knot point \the\count@}%
    \ifnum\count@=\z@ \count@=\@ne \else \dimen@=\p@ \fi
   \else \advance\count@\@ne
\splinetrace@{find point \the\dimen@\space on spline segment \the\count@}%
  \fi
  \edef\segmentnum@{\the\count@}%
  \edef\next@{\begingroup\noexpand\splinesegment@{\the\count@}}%
 \edef\nextii@{%
   \noexpand\spline@find{\the\dimen@}{#2}{}\noexpand\checksplinefind@{#2}}%
 \ifx>#3\relax
  \edef\nextii@{%
   \noexpand\spline@find{\the\dimen@}{#2}#3\noexpand\checksplinefind@{#2}}%
 \else\ifx-#3\relax \ifdim\dimen@>\z@\relax \ifdim\dimen@<\p@
  \dimen@=-\dimen@ \advance\dimen@\p@
  \edef\nextii@{\noexpand\splinereverse@
   \noexpand\spline@find{\the\dimen@}{#2}{}\noexpand\checksplinefind@{#2}}%
  \fi\fi
 \else\ifx<#3\relax \ifdim\dimen@>\z@\relax \ifdim\dimen@<\p@
  \dimen@=-\dimen@ \advance\dimen@\p@
  \edef\nextii@{\noexpand\splinereverse@
   \noexpand\spline@find{\the\dimen@}{#2}#3\noexpand\checksplinefind@{#2}}%
  \fi\fi
 \fi\fi\fi
 \fi\fi  \expandafter\next@\nextii@ }
```

This macro checks whether the required sliding has been carried out, or whether it is necessary to continue sliding along the next segment.

```
\xydef@\checksplinefind@#1{%
 \dimen@ii=#1\relax \dimen@=\splinelength@
 \ifdim\dimen@ii=\z@\DN@{}%
 \else
  \ifdim\dimen@ii<\z@\advance\dimen@\dimen@ii
  \else\advance\dimen@-\dimen@ii\fi
  \ifdim\dimen@<\z@\multiply\dimen@\m@ne\fi
 \ifdim\dimen@<.2\p@\DN@{}%
 \else
  \count@=\segmentnum@\relax
```

```
  \ifdim\dimen@ii<\z@\advance\dimen@ii\splinelength@
   \ifnum\count@>\@ne\advance\count@\m@ne
   \edef\next@{\noexpand\splinesegment@{\the\count@}%
    \noexpand\spline@find\p@{\the\dimen@ii}{}%
     \noexpand\checksplinefind@{\the\dimen@ii}}%
   \else
    \xywarning@{cannot slide \the\dimen@ii\space before start}%
    \DN@{\splinesegment@\@ne \spline@find\z@\z@{}}\fi
  \else\advance\dimen@ii-\splinelength@
   \count@@=\numcontrolpts
   \ifnum\count@<\count@@ \advance\count@\@ne
    \edef\next@{\noexpand\splinesegment@{\the\count@}%
     \noexpand\spline@find\z@{\the\dimen@ii}{}%
      \noexpand\checksplinefind@{\the\dimen@ii}}%
   \else
    \xywarning@{cannot slide \the\dimen@ii\space beyond end}%
    \DN@{\splinesegment@{\the\count@@}\spline@find\p@\z@{}}\fi
  \fi
\fi\fi \next@ }
```

Breaks Typesetting breaks in curves is quite involved. The `\bstartPLACE@` and `\bendPLACE@` are used to hold parameter values for the positions of the beginning and end of a segment, respectively, when known. The default values of `\relax` indicate that these values need to be found, usually from the start- and end-edges of the curve. The value of `\empty` for `\bstartPLACE@` is used initially with a new curved connection, to set up all its parameters — no typesetting takes place. This is used by curved arrows to set up the connection before places can be found for any breaks.

```
\xydef@\resetbreaks@{%
 \let\bstartPLACE@=\relax \let\bthisPLACE@=\empty
 \let\bendPLACE@=\relax \let\bprevPLACE@=\relax}
```

```
\xydef@\invisbreaks@{\resetbreaks@ \let\bstartPLACE@=\empty}
\xydef@\crvresetbreaks@{\resetbreaks@ \let\bstartPLACE@=\relax }
```

```
\xydef@\initbreaks@{\xy@@{\ifx\bstartPLACE@\relax
 \def\bstartPLACE@{0}\def\bendPLACE@{1}\else
 \ifx\bstartPLACE@\empty\def\bstartPLACE@{0}\def\bendPLACE@{1}%
 \fi\fi}}
```

```
\xydef@\lastbreaks@{\ifx\bprevPLACE@\relax \def\bstartPLACE@{0}%
 \else \expandafter \bprevPLACE@ \fi
 \let\bprevPLACE@\relax \let\bendPLACE@\relax}
```

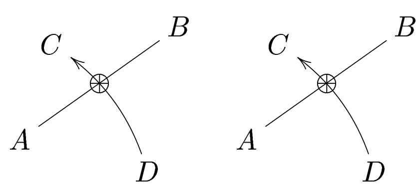
```
\xylet@\bstartPLACE@\relax
\xylet@\bendPLACE@\relax
\xylet@\bprevPLACE@\relax
\xydef@\bthisPLACE@{1}
```

These are called in the actual setting of curved sub-segments.

```
\xydef@\splinebreak@{%
\splinetrace@{***new break*** this:\bthisPLACE@ }%
 \ifx\bthisPLACE@\empty\def\bendPLACE@{1}%
```

```
 \else\edef\bendPLACE@{\bthisPLACE@}\let\bthisPLACE@\empty\fi
%\xystatus@{BR1}%
 \ifx\bprevPLACE@\relax\def\bstartPLACE@{0}\else\expandafter\bprevPLACE@\fi
 \edef\bprevPLACE@{\def\noexpand\bstartPLACE@{\bendPLACE@}\pfromthec@}%
\splinetrace@{splineBREAK: \bstartPLACE@ <> \bendPLACE@}%
%\xystatus@{BR}%
 \edef\cv@brend{\cfromthec@}%
 \edef\cv@brstart{\cfromthep@}%
 \expandafter\connect\expandafter\crvs\crvSTYLE@@{}% ? optimise
 \let\cv@brend=\relax
 \let\cv@brstart=\relax
\splinetrace@{**break done**}%
}

\xydef@\splinelast@{\splinereset@ \lastbreaks@ %\splinetracing
\splinetrace@{splineLAST}%
%\xystatus@{LA}%
 \edef\cv@brend{\cfromthec@}%
 \edef\cv@brstart{\cfromthep@}%
 \expandafter\connect\expandafter\crvs\crvSTYLE@@{}% ? optimise
 \let\cv@brend=\relax
 \let\cv@brstart=\relax
\splinetrace@{**last done**}%
 \resetbreaks@}
```

Intersection with a curved connection Just as the intersection of two lines (1.3j) can be found, so can the intersection of a straight line with a curved connection, or the intersection of a curve with a straight connection.




```
\xy*+{A}="A";p+/r5pc/+(0,15)*+{B}="B"
 ,p+<1pc,3pc>*+{C}="C"
 ,"A"+<4pc,-1pc>*+{D}="D",{\ar@/_/"C"}
 ,?!{"A";"B"**@{-}}*++{\oplus}
\endxy \quad \xy
 *+{A}="A";p+/r5pc/+(0,15)*+{B}="B",
 ,p+<1pc,3pc>*+{C}="C"
 ,"A"+<4pc,-1pc>*+{D}="D","A";"B"**@{-}
 ,?!{"D",{\ar@/_/"C"}}*++{\oplus}
\endxy
```

When the line separates the end-points of a curve an intersection can always be found. If there is more than one then that occurring earliest along the curve is the one found.

Presume the end-points of the spline segment are at (X_p, Y_p) and (X_c, Y_c) with (L_p, U_p) and (R_p, D_p) as two points on the line. Then the macro `\xysidesofinterval@` returns 3 pieces of information.

- `\next` expands to 2 characters, each from $\{+,-,0\}$ indicating which side of, or on, the line lies either end-point;
- `\this@` expands to either $+$ or $-$ indicating which end is closer to the line, returning $+$ if equi-distant;
- `\howclose@` contains a non-negative ⟨num⟩ indicating the distance to the nearest end-point.

```
\xydef@\splineNointercept@{\enter@{\pfromthep@}%
 \begingroup \splinereset@ \csname cv@0\endcsname \swap@
  \count@=\ptsnum@ \advance\count@\@ne \csname cv@\the\count@\endcsname
  \edef\tmp@{\endgroup \X@origin=\the\X@p \Y@origin=\the\Y@p
   \R@c=\the\d@X \U@c=\the\d@Y}\tmp@
 \loop@
  \dimen@=\ifdim\R@c<\z@-\fi\R@c
  \advance\dimen@ \ifdim\U@c<\z@-\fi\U@c
  \ifdim\dimen@>10\p@
   \advance\R@c \ifdim\R@c<\z@-\fi 16sp \divide\R@c\KK@
   \advance\U@c \ifdim\U@c<\z@-\fi 16sp \divide\U@c\KK@
 \repeat@
 \intersect@ \leave@}

\xydef@\splineintercept@{\Hidden@true\Invisible@true\splineintercept@i}

\xydef@\splineintercept@i{\enter@{\pfromthep@}\begingroup
 \R@p=\X@c \L@p=\X@p \d@X=\R@p \advance\d@X-\L@p
 \D@p=\Y@c \U@p=\Y@p \d@Y=\D@p \advance\d@Y-\U@p
 \dimen@nth\KK@\d@X \edef\sd@X{\expandafter\removePT@\the\d@X}%
 \dimen@nth\KK@\d@Y \edef\sd@Y{\expandafter\removePT@\the\d@Y}%
 \dimen@=\sd@Y\L@p \advance\dimen@-\sd@X\U@p
 \edef\sideOffset@{\the\dimen@}\def\closestseg@{1}%
 \let\crvconnect@@=\relax \crvconnect@
 \let\spline@end=\splinef@end
 \def\spline@start{\bgroup \def\prevnext@{+-}%
  \def\splinerec@{\let\splinerec@=\splineIrec@ \splineinterceptTest@}%
  \def\splinedecast@{\let\splinedecast@=\splinedecast@@
   \def\splineadvance@{\splineIadvance@@}\splinedecast@@}%
  \def\splineadvance@{\intercepthull@}%
% \def\splinepoint@{\splineIpt@@}%
  \global\let\splinepoint=\splineIpt@@
  \global\let\splinegoal@=\splinef@pt
  \global\let\splinefinish@=\splinefinish@@}%
 \begingroup \splineIsegment@
  \splinesetparams@ \splined@@ \def\postfind@{}\splineIplace@
 \setupDirection@ii \leave@ }

\xydef@\splineIplace@{%
 \d@X=\dimen5 \d@Y=\dimen7 \dimen@=\splineval@
 \expandafter\count@\xycrvptsnum@\relax
 \ifnum\count@>\tw@ \advance\count@\m@ne %\advance\count@\m@ne
  \advance\dimen@\segmentnum@\p@ \advance\dimen@-\p@
  \dimen@nth\count@\dimen@  %  \divide\dimen@\count@
```

```
 \fi \edef\tmp@{\endgroup \endgroup
  \def\noexpand\PLACEf@{{\expandafter\removePT@\the\dimen@}}%
  \X@c=\dimen@i \Y@c=\dimen3 %
  \d@X=\the\d@X \d@Y=\the\d@Y }\tmp@
\splinetrace@{PLACE = \PLACEf@:(\the\X@c, \the\Y@c)}%
 \expandafter\edef\expandafter\bthisPLACE@\PLACEf@
 }

\xydef@\splineIrec@{\splineinterceptTest@ }

\xydef@\squineIpt@{\edef\tmp@{\endgroup
 \def\noexpand\prevnext@{\next}\begingroup}\tmp@ }
\xydef@\squineIadvance@{\edef\tmp@{\endgroup
  \def\noexpand\prevnext@{\next}\def\noexpand\howclose@{\howclose@}}%
 \tmp@ \splineadvance@@\begingroup}

\xydef@\cubicIpt@{\edef\tmp@{\endgroup
 \def\noexpand\prevnext@{\next}\begingroup}\tmp@ }
\xydef@\cubicIadvance@{\edef\tmp@{\egroup
  \def\noexpand\prevnext@{\next}\def\noexpand\howclose@{\howclose@}}%
 \tmp@ \splineadvance@@\bgroup}

\xydef@\splineIsegment@{\DN@{\splinesegment@\@ne}%
 \ifx\splinesegment@\squinesegment@\else
 \ifx\splinesegment@\cubicsegment@\else\DN@{\findIsegment@}%
 \fi\fi\next@}

\xydef@\findIsegment@{\expandafter\count@\xycrvptsnum@ \count@@\z@
 \loop@ \advance\count@@\@ne \bgroup
  \expandafter\splinesegment@\expandafter{\the\count@@}%
  \xysidesofinterval@
  \findIsegment@i \repeat@
  \expandafter\dimen@\prevhowclose@ \relax
  \ifdim\dimen@=\z@ \edef\closestseg@{\the\count@@}\fi
  \expandafter\splinesegment@\expandafter{\closestseg@}}

\xydef@\findIsegment@i{%
 \expandafter\ifx\next \expandafter\findIsegment@ii
 \else\expandafter\findIsegment@x\fi \next@ }

\xydef@\findIsegment@ii{\expandafter\count@\xycrvptsnum@ \relax
 \advance\count@\m@ne \ifnum\count@>\segmentnum@\relax
 \expandafter\findIsegment@y\else\expandafter\findIsegment@x\fi}

\xydef@\findIsegment@x{%
 \edef\next@{\egroup\def\noexpand\next{\next}%
  \def\noexpand\prevhowclose@{\prevhowclose@}%
  \def\noexpand\closestseg@{\closestseg@}\noexpand\iffalse}}

\xydef@\findIsegment@y{%
 \edef\next@{\egroup\def\noexpand\next{\next}%
  \def\noexpand\prevhowclose@{\prevhowclose@}%
  \def\noexpand\closestseg@{\closestseg@}\noexpand\iftrue}}

\xydef@\xysidesofinterval@{\bgroup
  \L@p=\sd@X\Y@p \advance\L@p-\sd@Y\X@p \advance\L@p\sideOffset@
```

```
 \U@p=\sd@X\Y@c \advance\U@p-\sd@Y\X@c \advance\U@p\sideOffset@
\def\this@{}\ifdim\L@p>\almostz@ \ifdim\U@p>\almostz@ \def\next{++}\else
 \ifdim\U@p<-\almostz@ \def\next{+-}\else\def\next{+0}\def\this@{-}\fi\fi
\else\ifdim\L@p<-\almostz@ \ifdim\U@p>\almostz@ \def\next{-+}\else
 \ifdim\U@p<-\almostz@ \def\next{--}\else\def\next{-0}\def\this@{+}\fi\fi
\else \ifdim\U@p>\almostz@ \def\next{0+}\else
 \ifdim\U@p<-\almostz@ \def\next{0-}\else\def\next{00}\def\this@{+}\fi\fi
\fi\fi
\ifx\this@\empty
 \L@p=\ifdim\L@p<\z@-\fi\L@p \U@p=\ifdim\U@p<\z@-\fi\U@p
 \ifdim\L@p<\U@p \edef\howclose@{\the\L@p}\def\this@{-}%
 \else \edef\howclose@{\the\U@p}\def\this@{+}\fi
\else\def\howclose@{0.0pt}\fi
\splinetrace@{\next:\this@:\howclose@;\the\splineval@:%
 (\the\X@p,\the\Y@p);(\the\X@c,\the\Y@c)}%
\ifdim\howclose@<\prevhowclose@\relax
 \edef\tmp@{\egroup\def\noexpand\this@{\this@}\def\noexpand\next{\next}%
  \def\noexpand\howclose@{\howclose@}\def\noexpand\prevhowclose@{\howclose@}%
  \def\noexpand\closestseg@{\segmentnum@}}%
\splinetrace@{closest yet: \howclose@, in segment \segmentnum@}%
\else
 \edef\tmp@{\egroup\def\noexpand\this@{\this@}\def\noexpand\next{\next}%
  \def\noexpand\howclose@{\howclose@}}%
\fi \tmp@ }
```

```
\xydef@\splineinterceptTest@{%
\xysidesofinterval@ \DN@{\splineadvance@}%
\DNii@{0.0pt}\ifx\howclose@\nextii@ \DN@{\splineclosest@}%
\else\expandafter\ifx\next \expandafter\ifx\prevnext@
 \ifx\next\prevnext@\else\DN@{\X@c=\X@p \Y@c=\Y@p
  \splinefinishf@}\fi\fi
\else \DN@{\splinedecast@}\fi\fi
{\splinefar@false \splinenear@\X@p\X@c\Y@p\Y@c
 \ifsplinefar@ \aftergroup\next@
 \else\aftergroup\splineclosest@
\splinetrace@{too close: (\the\X@p,\the\Y@p)::(\the\X@c,\the\Y@c)}%
\fi}}
```

```
\xydef@\splineclosest@{%\global\advance\splineval@-.5\splinedepth@
\expandafter\ifx\this@-\relax
 \X@c=\X@p \Y@c=\Y@p \global\advance\splineval@-\splinedepth@
\fi \splinefinishf@ }%
```

When the line misses the convex hull of the curve's control points it is still possible to get the point of closest approach.

```
\xydef@\squinetestcvxhull@{\begingroup
\splinetrace@{hull: closest in segment \closestseg@}%
\expandafter\splinesegment@\ifx\undefined\closestseg@\@ne\else\closestseg@\fi
\X@c=\L@c \Y@c=\U@c \xysidesofinterval@
\expandafter\ifx\next \expandafter\splineoutsidehull@@
\else \expandafter\splineinsidehull@\fi}
```

```
\xydef@\cubictestcvxhull@{\begingroup
 \expandafter\splinesegment@\ifx\undefined\closestseg@\@ne\else\closestseg@\fi
 \X@c=\L@c \Y@c=\U@c \xysidesofinterval@
 \expandafter\ifx\next \expandafter\cubictestcvxhull@i
 \else \expandafter\splineinsidehull@\fi}
```

```
\xydef@\cubictestcvxhull@i{%\begingroup
 \X@c=\R@c \Y@c=\D@c \xysidesofinterval@
 \expandafter\ifx\next \expandafter\splineoutsidehull@@
 \else \expandafter\splineinsidehull@\fi}
```

```
\xydef@\xysidesofhull@{\bgroup
 \L@p=\sd@X\Y@p \advance\L@p-\sd@Y\X@p \advance\L@p\sideOffset@
 \U@p=\sd@X\Y@c \advance\U@p-\sd@Y\X@c \advance\U@p\sideOffset@
 \L@p=\ifdim\L@p<\z@-\fi\L@p \U@p=\ifdim\U@p<\z@-\fi\U@p
 \ifdim\L@p<\U@p \edef\howclose@{\the\L@p}\def\this@{-}%
 \else \edef\howclose@{\the\U@p}\def\this@{+}\fi
 \ifx\splinedecast@\squinedecast@ \D@c=\U@c \R@c=\L@c \fi
 \L@p=\sd@X\U@c \advance\L@p-\sd@Y\L@c \advance\L@p\sideOffset@
 \L@p=\ifdim\L@p<\z@-\fi\L@p
 \expandafter\if\this@-%
  \ifdim\L@p<\prevhowclose@ \def\next{-+}\else \def\next{--}\fi
 \else
  \ifdim\L@p<\howclose@ \def\next{-+}\else
   \U@p=\sd@X\D@c \advance\U@p-\sd@Y\R@c \advance\U@p\sideOffset@
   \U@p=\ifdim\U@p<\z@-\fi\U@p
   \ifdim\U@p<\howclose@ \def\next{+-}\else \def\next{++}\fi
 \fi\fi
 \edef\tmp@{\egroup\def\noexpand\this@{\this@}\def\noexpand\next{\next}%
  \def\noexpand\howclose@{\howclose@}}\tmp@ }
```

This is for straight connections intercepting with a curve. It requires a redefinition of `\PLACE@intercept@` to recognise that the intercept is with a curved connection.

```
\def\PLACE@intercept@#1{%
 \xy@@{\enter@{\pfromthep@\basefromthebase@}\begingroup}%
 \xy@@ix@{#1}\xy@@{\plainxy@ \def\Creset@@{}%
  \expandafter\POS\the\toks9\relax
  \DNii@{\crvreset@}\ifx\nextii@\Creset@@ \edef\next@{\endgroup
   \def\noexpand\xyinterceptpts@{\the\crvpts@}%
   \X@c =\the\X@c \Y@c=\the\Y@c \X@p=\the\X@p \Y@p=\the\Y@p
   \noexpand\CCintercept@@ }%
  \else \edef\next@{\endgroup
   \X@c =\the\X@c \Y@c=\the\Y@c \X@p=\the\X@p \Y@p=\the\Y@p
   \noexpand\Cintercept@@ }%
  \fi \next@ \leave@}%
 \PLACE@@}
```

```
\xydef@\CCintercept@@{\splineSintercept@}
\xydef@\splineSintercept@{\enter@{\pfromthep@}\begingroup
  \Creset@@ \edef\connect@pts{\cfromthec@ \pfromthep@}%
   \xyinterceptpts@ \splinereset@ \connect@pts \splineintercept@
```

```
  \edef\next@{\endgroup \cfromthec@ \DirectionfromtheDirection@ }%
 \next@ \let\xyinterceptpts@=\empty \leave@}
```

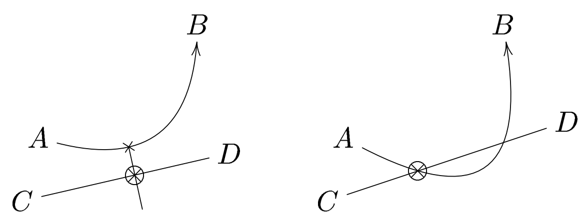
To Do: If both connections are curves, only the line joining the end-points of the first connection is used. Find a way to do better than this.

If the line does not separate the end-points then there may be no intersection with the curve. If there is one then either the line is tangential or necessarily there will also be at least one other intersection. A message

```
perhaps no curve intersection, or many.
```

is written to the log-file, but a search for an intersection will still be performed and a “sensible” place found on the curve. In the usual case of a single quadratic or cubic segment, the place nearest the line is found and the tangent direction is established.

The following examples show this, and show how to get the place on the line nearest to the curve.




```
\xy *+{A}="A";p+/r5pc/+(0,15)*+{B}="B",
 ,p-<.5pc,2pc>*+{C}="C","A"+<6pc,-.5pc>
 ,*+{D}="D","A",{\ar@/_25pt/"B"}
 ,?!{"C";"D"**@{-}}*\dir{x}="E"
 ,+/_2pc/="F";"E"**@{-},?!{"C";"D"}
 ,*{\otimes}\endxy\qquad\xy
 *+{A}="A";p+/r5pc/+(0,15)*+{B}="B",
 ,p-<.5pc,2pc>*+{C}="C"
 ,"A"+<7pc,.5pc>*+{D}="D","A"
 ,{\ar@/_40pt/"B"},?!{"C";"D"**@{-}}
 ,*{\otimes}\endxy
```

The warning message, that no intersection may exist, always occurs inside a grouping so it can safely switch itself off for deeper levels.

```
\xydef@\nointerceptwarning@{\let\nointerceptwarning@=\relax
 \xywarning@{perhaps no curve intersection, or many. }}

\xydef@\intercepthull@{\nointerceptwarning@
 \expandafter\def\expandafter\prevnext@\expandafter{\next}%
 \splinetestcvxhull@}
```

When the line meets the convex hull of the current control points, then a closer point exists inside the current portion of the curve. We must “decast” to find it, after storing the closest value known so far, within `\prevhowclose@`.

```
\xydef@\splinetestcvxhull@{\begingroup
 \X@c=\L@c \Y@c=\U@c \splinetrace@{hull:}\xysidesofinterval@
 \expandafter\ifx\next \expandafter\splineoutsidehull@@
 \else \expandafter\splineinsidehull@\fi}

\xydef@\splineinsidehull@{\edef\tmp@{\endgroup
```

```
 \def\noexpand\howcvxclose@{\howclose@}%
  \noexpand\interceptinsidehull@}\tmp@ }
```

```
\xydef@\interceptinsidehull@{\edef\prevhowclose@{\howclose@}%
 \splinedecast@@ }
```

If the line does not cross into the convex hull, then we use the distance from the control points to determine whether or not there can be a closer point. The value of `\prevhowclose@` indicates the best so far, while `\howcvxclose@` temporarily holds the distance to the closest control point. When this is nearer then there will be a closer point on the curve.

```
{\xdef\next{\the\maxdimen}}
\xylet@\prevhowclose@=\next
```

```
\xydef@\splineoutsidehull@{\edef\tmp@{\endgroup
  \def\noexpand\howcvxclose@{\howclose@}%
   \noexpand\interceptoutsidehull@}\tmp@}
\xylet@\splineoutsidehull@@=\splineoutsidehull@
```

```
%\xydef@\interceptoutsidehull@{\dimen@=\howcvxclose@ \relax
% \ifdim\dimen@<\howclose@ \relax
%  \ifdim\howclose@<\prevhowclose@\relax \edef\prevhowclose@{\howclose@}\fi
%   \DN@{\begingroup\splinedecast@@ \endgroup}%
% \else\ifdim\howclose@>\prevhowclose@ \relax \DN@{\splineHadvance@}%
% \else \DN@{\splineHadvance@ }\fi\fi \next@ }
\xydef@\interceptoutsidehull@{\dimen@=\howcvxclose@ \relax
% \global\splineval@=\z@
\splinetrace@{OUTSIDE HULL: \the\dimen@ : (\howclose@)(\prevhowclose@)}%
 \let\splinedecast@=\splinedecast@@
 \let\splinerec@\splineHnear@test
 \let\splineadvance@\splineIadvance@@
 \splinedecast@@ }
```

```
\xydef@\splineHnear@test{%
 \xysidesofhull@ \DN@{\splinedecast@}%
 \ifdim\howclose@<\prevhowclose@\edef\howclose@{\prevhowclose@}\fi
 \expandafter\if\this@-%
  \expandafter\ifx\next\DN@{\splineclosest@}\fi
 \else
  \expandafter\ifx\next\DN@{\splineadvance@}\fi
 \fi
 {\splinefar@false \splinenear@\X@p\X@c\Y@p\Y@c
  \ifsplinefar@ \aftergroup\next@
  \else\aftergroup\splineclosest@
\splinetrace@{too close: (\the\X@p,\the\Y@p)::(\the\X@c,\the\Y@c)}%
 \fi}}
```

For cubic segments it is necessary to look at both control-points off the curve and compare their distances; hence the use of `\prevhowcvxclose@`, initially set to `\maxdimen`.

```
\xydef@\cubicoutsidehull@{\edef\tmp@{\endgroup
  \def\noexpand\howcvxclose@{\howclose@}}\tmp@
  \edef\prevhowcvxclose@{\howcvxclose@}\begingroup
 \X@p=\R@c \Y@p=\D@c \splinetrace@{hullii:}\xysidesofinterval@
 \expandafter\ifx\next \expandafter\cubicoutsidehull@i
```

```
 \else \expandafter\cubicinsidehull@\fi}
```

```
\xydef@\cubicinsidehull@{\edef\tmp@{\endgroup
 \def\noexpand\howclose@{\howclose@}}\tmp@
 \cubicoutsidehull@ii }
```

```
\xydef@\cubicoutsidehull@i{\edef\tmp@{\endgroup
 \def\noexpand\howcvxclose@{\howclose@}}\tmp@
 \dimen@=\prevhowcvxclose@\relax
 \ifdim\dimen@<\howcvxclose@ \relax
  \edef\howcvxclose@{\the\dimen@}\expandafter\interceptoutsidehull@
 \else \expandafter\cubicoutsidehull@ii\fi }
```

```
\xydef@\cubicoutsidehull@ii{%
 \def\splineadvance@{\splineadvance@@}%
 \def\splinepoint\endgroup{\endgroup
%  \global\let\splinepoint=\splinepoint@
  \global\let\splinepoint=\relax
  \def\splineadvance@{\intercepthull@}}%
 \splinedecast@@ }%
```

This macro `\splineHadvance@` is used at the end of a `\splinedecast@` in order to pass information from out of a subsegment, to help decide whether there is any point in further searching.

```
\xydef@\splineHadvance@{\futurelet\next@\splineHadvance@i}
\xydef@\splineHadvance@i{\ifx\next@\endgroup
 \expandafter\splineHadvance@x\else\expandafter\splineclosest@\fi}
```

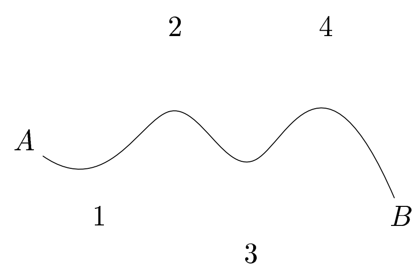
```
\xydef@\splineHadvance@x{\edef\tmp@{\endgroup
  \def\noexpand\prevhowclose@{\prevhowclose@}%
  \def\noexpand\prevnext@{\next}\def\noexpand\howcvxclose@{\howclose@}}%
 \tmp@ \dimen@=\prevhowclose@ \relax
 \ifdim\dimen@<\howclose@ \relax \DN@{\splineclosest@\begingroup}%
 \else\ifdim\howcvxclose@=\dimen@\DN@{\splineclosest@\begingroup}%
 \else\DN@{\splineadvance@@\begingroup}\fi\fi \next@ }
```

Miscellaneous features The command `\clearcurve` was originally provided to discard the curve information, and return to the graphics state before the curve was read, if this is ever found to be necessary. This will probably be removed.

```
\xydef@\xyclearcrv@{\cv@end\cv@restore\loop@
 \expandafter\let\csname cv@\number\crv@cnt@\endcsname\relax
 \ifnum\crv@cnt@>0\advance\crv@cnt@\m@ne\repeat@
 \let\cv@start=\relax \let\cv@end=\relax \let\cv@restore=\relax \POS}
\xylet@\clearcurve=\xyclearcrv@
```

To Do: Define an `\extendcurve` as a ⟨postcurve⟩ which allows the curve to be continued smoothly past the endpoint c.

A macro can be used within the modifier, to provide different labels at each control point. For example. . .




```
\newcount\cptlabel\global\cptlabel=0
\def\nextcptlabel{\global\advance\cptlabel 1 %
 \number\cptlabel}
\xy (0,0)*+{A}; (50,-10)*+{B}
**\crv~Pc{~*{\nextcptlabel}(10,-10)
 &(20,15)&(30,-15)&(40,15)} \endxy
```

Two control sequences are provided to access the control points. Firstly `\numcontrolpts` returns the number of them, while `\xycontrolpt`⟨num⟩ sets c to be the particular control point. Here ⟨num⟩ must be a single digit, grouped integer *e.g.* `{-15}` or count register containing a numerical value. If larger than `\numcontrolpts` then c becomes the end-point of the curve, while if zero it becomes the starting point. A negative value sets both p and c to be the starting point, whereas other values leave p unchanged.

```
\xylet@\numcontrolpts=\xynumctrlpts@
```

```
\xydef@\xycontrolpt@#1{{%
 \ifnum#1<\z@\aftergroup\cv@restore\aftergroup\cv@start
 \else
  \expandafter\count@\xycrvptsnum@\relax \advance\count@\@ne
  \ifnum#1>\count@ \ifx\cv@end\relax
   \expandafter\aftergroup\csname cv@\number#1\endcsname
   \else\aftergroup\cv@end\fi
  \else
   \expandafter\ifx\csname cv@\number#1\endcsname\relax
    \ifnum#1=\z@\aftergroup\cv@start\else\aftergroup\cv@end\fi
   \else\expandafter\aftergroup\csname cv@\number#1\endcsname
 \fi\fi\fi}}%
\xylet@\xycontrolpt=\xycontrolpt@
```

Storing control point locations: Upon starting to read the control points for a curve, first set the counter `\crv@cnt@` to be zero. Store the position of p and c and the current base in control sequences `\cv@start`, `\cv@end` and `\cv@restore` for later retrieval and to reinstate the current graphics state. `\cv@start` is a little more complicated in order to retain the existing value of p.

```
\xydef@\startxycurve@@{\crv@cnt@=\z@
 \edef\cv@start{\cfromthep@}%
 \ifdim\zz@\R@c \ifdim\zz@\L@c \ifdim\zz@\D@c \ifdim\zz@\U@c
     \Edge@c={\zeroEdge}\fi\fi\fi\fi
 \edef\cv@end{\cfromthec@}%
 \edef\cv@restore{\pfromthep@\basefromthebase@}}
\xylet@\startxycurve@=\startxycurve@@
```

```
\xylet@\cv@start=\relax
```

```
\xylet@\cv@end=\relax
\xylet@\cv@restore=\relax
\xydef@\readxycurve@{\startxycurve@\xycrvmods@}
\xylet@\readxycurve=\readxycurve@
```

The position of each control point is stored as the expansion of a control sequence whose name encodes its place within the ⟨poslist⟩. Expanding this control sequence sets c to be the appropriate position.

```
\xydef@\addtocrv@{\advance\crv@cnt@\@ne
 \edef\tmp@{%
   \expandafter\noexpand\csname cv@\number\crv@cnt@\endcsname}%
 \expandafter\edef\tmp@{\cfromthec@}%
 \edef\tmp@{\noexpand\addtocrvpts@{\noexpand\def
  \expandafter\noexpand\tmp@{\X@c=\the\X@c \Y@c=\the\Y@c }}}\tmp@
 \edef\xycrvptsnum@{\the\crv@cnt@}%
}
```

```
\xydef@\xycrvdrop@{ {\zerodot}}
\xydef@\xycrvconn@{}
```

```
\xydef@\xyc@trlpts@{\def\xycrvdrop@{ \dir{x}}\def\xycrvconn@{}%
 \afterCURVE{\savecrvobjects@\xyctrlpts@}\readxycurve@}
\xydef@\xyc@trlpts@@{%
 \afterCURVE{\savecrvobjects@\def\xycrvdrop@{ \dir{x}}%
  \def\xycrvconn@{}\xyctrlpts@}\readxycurve@}
\xydef@\xyc@vxhull@{\def\xycrvdrop@{ {}}%
 \edef\xycrvconn@{ \noexpand\dir{\addDASH@{}}}%
 \afterCURVE{\savecrvobjects@\xycvxhull@}\readxycurve@}
\xydef@\xyc@vxhull@@{%
 \afterCURVE{\savecrvobjects@\def\xycrvdrop@{ {}}%
  \edef\xycrvconn@{ \noexpand\dir{\addDASH@{}}}\xycvxhull@}\readxycurve@}
\xylet@\controlpts=\xyc@trlpts@
\xylet@\convexhull=\xyc@vxhull@
\xydef@\savecrvobjects@{\bgroup
  \expandafter\toks@\expandafter{\xycrvdrop@}%
  \expandafter\toks@ii\expandafter{\xycrvconn@}%
  \edef\tmp@{\egroup\noexpand\def\noexpand\resetcrvobjects@{%
   \noexpand\def\noexpand\xycrvdrop@{\the\toks@}%
   \noexpand\def\noexpand\xycrvconn@{\the\toks@ii}%
  }}\tmp@}
```

Alternate curve styles: There is the potential to do many other things here, by using `\afterCURVE`, once the list of control points has been read.

```
\xydef@\afterCURVE#1{\def\afterCURVE@{#1}}
\xydef@\endxycurve@{\cv@end\savectrlptsnum@
 \expandafter\def\csname params@\endcsname{\the\crvpts@}%
 \cv@restore\afterCURVE@\POS}
\xylet@\endcurve=\endxycurve@
```

In fact this is essentially how `\controlpts` and `\convexhull` were interfaced before extensions were added to change styles.

```
\def\controlpts{\afterCURVE{\xyctrlpts}%
 \readxycurve}
\def\convexhull{\afterCURVE{\xycvxhull}%
 \readxycurve}
```

The real work is done by `\xyctrlpts` and `\xycvxhull`. At the point when these macros are expanded the graphics state has the restored p and c to their original values, *i.e.* the end-points of the curve.

`\controlpts`: This macro visits in order each of the control points dropping the requisite object. The connection style is not used and the direction does not change. For a loop-counter we use `\count@@@`.

```
\xydef@\xyctrlpts@{\cv@end \cv@restore
 \def\crvconnect@{\straitconnect@}%
 \ifnum\crv@cnt@>\z@
  \count@@@=\@ne \DN@{\loop@\xycontrolpt@{\count@@@}\relax
  \expandafter\xycvxhulldrop@\xycrvdrop@
  \ifnum\crv@cnt@>\count@@@\advance\count@@@\@ne\repeat@
  \cv@end \cv@restore }%
 \else\DN@{\relax}\fi \next@ }
\xylet@\xyctrlpts=\xyctrlpts@
```

The token list `\crvpts@` stores the following information: current scope , location of p and c, number of control points `\crv@cnt@` and their locations. It does this by storing `\def`s for control sequence names, depending on the scope, which expand to this information. Thus by executing `\the\crvpts@` at the right level, the information is conveniently recovered via control sequences which do not conflict with anything else, and whose names can be reconstructed.

```
\xydef@\addtocrvpts@#1{\crvpts@=\expandafter{\the\crvpts@#1}}
```

```
\xydef@\savescope@{\crvpts@={}%
 \edef\tmp@{\expandafter\noexpand\csname cv@0\endcsname}%
 \edef\tmp@{\noexpand\addtocrvpts@{\noexpand\def
  \expandafter\noexpand\tmp@{\X@c=\the\X@p \Y@c=\the\Y@p}}}%
 \tmp@ }
```

```
\xydef@\savectrlptsnum@{%
 \advance\crv@cnt@\@ne \edef\tmp@{%
   \expandafter\noexpand\csname cv@\number\crv@cnt@\endcsname}%
  \edef\tmp@{\noexpand\addtocrvpts@{\noexpand\def
   \expandafter\noexpand\tmp@{\X@c=\the\X@c \Y@c=\the\Y@c}}}\tmp@
 \advance\crv@cnt@\m@ne
 \edef\tmp@{\expandafter\noexpand\csname ptsnum@\endcsname}%
 \edef\tmp@{\noexpand\addtocrvpts@{%
  \noexpand\def\expandafter\noexpand\tmp@{\number\crv@cnt@}}}%
 \tmp@ }
```

Convex Hull: This macro visits in order each of the control points dropping the requisite object. With the previous as p and the current one as c, a connection is set in the requisite style. For a loop-counter we use `\count@@@`.

Use a group `{\expandafter\POS\xycrvconn@}` else the `\loop@` gets upset. This does not affect the size of the XY-picture, since the dropped objects are not similarly shrouded.

Also used for the trivial case where there are no control points, hence a straight line between p and c is what is requested.

```
\xydef@\xycvxhull@{\cv@end \cv@restore
  \addtocrvpts@{\def\crvconnect@{\straitconnect@}}%
 \def\crvconnect@{\straitconnect@}%
 \ifnum\crv@cnt@>\z@
  \count@@@=\@ne\relax \DN@{\loop@\xycontrolpt@{\count@@@}\relax
   \expandafter\xycvxhulldrop@\xycrvdrop@
   {\expandafter\connect\xycrvconn@{}}\pfromc@
   \ifnum\crv@cnt@>\count@@@\relax \advance\count@@@\@ne\repeat@
  \cv@end }%
 \else
  \ifx\empty\xycrvconn@
   \edef\xycrvconn@{\noexpand!C\noexpand\dir{\addDASH@{}}}\fi
  \DN@{\relax}\fi \next@
 {{\expandafter\connect\xycrvconn@{}}}%
 \cv@end \cv@restore }
\xylet@\xycvxhull=\xycvxhull@
```

```
\xydef@\xycvxhulldrop@#1#{\drop@{#1}}
```

Spline constructions The control sequence `\xysplineparams@` provides a way to access a control sequence name that is unique to the current XY-pic scope. This control sequence will be used to store the parameters which determine the spline curve. Similarly `\xysplineedges@` will be used to store the parameter values and lengths (along the squine) to the locations where the spline crosses the edge of the objects at the ends p and c.

```
\xydef@\savespline@{%
\splinetrace@{savespline@:}%
% \edef\endspline@{\endgroup
 \edef\endspline@{\endgroup
 \savesplineparams@ \savesplinerefs@
 \X@min=\the\X@min\relax \X@max=\the\X@max\relax
 \Y@min=\the\Y@min\relax \Y@max=\the\Y@max\relax
  \ifInvisible@\noexpand\Invisible@true
  \else\noexpand\Invisible@false\fi
 }\endspline@
 \edef\tmp@{\noexpand\addtocrvpts@{\savesplineparams@}}\tmp@
 \xycontrolpt@\z@ \cv@end }
```

```
\xydef@\savesplineparams@{%
 \noexpand\def\expandafter\noexpand\xysplineedges@{\xysplineedges@}}
```

```
\xydef@\savesplinerefs@{%
 \noexpand\def\noexpand\xysplineparams@{%
  \expandafter\noexpand\xysplineparams@}%
 \noexpand\def\noexpand\xysplineedges@{%
  \expandafter\noexpand\xysplineedges@}%
}
```

One new box register is required, for the object used to actually set the curve, . . .

```
\xynew@{box}\splinebox@
```

... and 4 dimen registers ...

```
\xynew@{dimen}\splineval@
```

... which is for tracking the natural spline parametrization;

```
\xynew@{dimen}\splinedepth@
```

... which is for tracking he depth of the recursive algorithms;

```
\xynew@{dimen}\splinetol@
```

... which contains the tolerance for the spline, (this can probably be shifted to an ordinary macro);

```
\xynew@{dimen}\splinelength@
```

... which accumulates a measure of the length along a spline curve. (Currently this is not being used, as most things can be done satisfactorily using the spline parametrization only.)

Some new conditionals are also required. These are for testing whether successive places on the squine are sufficiently close together, and whether the objects should be placed or not.

```
\xynew@{if}\ifsplinefar@
\xynew@{if}\ifsplineplot@ % to plot points or not
```

Provide some tracing ability, for debugging.

```
\xydef@\splinetracing{\let\splinetrace@=\W@}
\xydef@\splineignore@#1{}
\xylet@\splinetrace@=\splineignore@
```

The parameters for the current section of the spline are stored indirectly in `\xysplineparams@`, while information concerning the places where the spline crosses the edges of objects at p and c is stored indirectly in `\xysplineedges`. A control sequence `\readsplineparams@` is used to extract this information. Each spline type must provide a `\getsplineparams@` which puts the information into easily usable form.

```
\xydef@\readsplineparams@{%
 \expandafter\expandafter\expandafter\getsplineparams@\xysplineparams@
 \expandafter\expandafter\expandafter\getsplineedges@safe\xysplineedges@ @@}

\xydef@\getsplineedges@safe#1@@{\DN@{#1}%
 \ifx\next@\empty \DN@{\getsplineedges@.0pt;0.0pt,0.0pt;0.0pt,}%
 \else \DN@{\getsplineedges@safe@#1@@}\fi \next@}

\xydef@\getsplineedges@safe@#1;#2,#3@@{\DN@{#3}%
\splinetrace@{getsplineedges@safe}%
 \ifx\next@\empty \DN@{\getsplineedges@#1;#2,0.0pt;0.0pt,}%
 \else \DN@{\getsplineedges@#1;#2,#3}\fi \next@}
```

The edge locations are stored as $x.y$ where $x + 1 =$ spline-segment number and y or $1 - y =$ parameter value of the start, resp. finish, in the segment $x + 1$.

```
\xydef@\getsplineedges@#1;#2,#3;#4,{%
 \global\dimen@i=#1\global\dimen5=#2\relax
 \dimen@=#3\multiply\dimen@\m@ne \advance\dimen@\splinelength@
 \global\dimen3=\dimen@
 \dimen@=#4\relax
 \ifdim\dimen@>\p@ \xywarning@{edge out-of-range: \the\dimen@}%
  \count@=\dimen@ \divide\count@\p@
```

```
  \advance\dimen@-\count@\p@ \multiply\dimen@\m@ne
  \advance\dimen@ \count@\p@ \advance\dimen@\p@
\splinetrace@{\the\dimen@}%
 \else
  \multiply\dimen@\m@ne \advance\dimen@\p@
 \fi
 \global\dimen7=\dimen@ }
```

Sometimes TEX will run short of memory when many curves are used without a backend with special support for curves. In that case the following commands, that obey normal TEX groupings, may be helpful:

`\SloppyCurves`
`\splinetolerance{`⟨dimen⟩`}`

---

allow adjustment of the tolerance used to typeset curves. The first sets tolerance to .8pt, after which `\splinetolerance{0pt}` resets to the original default of fine curves.

Some useful methods for accurate division by integers.

```
\xydef@\dimen@half{%
  \advance\dimen@ \ifdim\dimen@<\z@-\fi 1sp \divide\dimen@\tw@}

\xydef@\halve@dimen#1{\dimen@=#1\relax \dimen@half #1=\dimen@}

\xydef@\dimen@nth#1#2{\dimen@=#2\relax \dimen@ii=#1sp %
  \advance\dimen@ \ifdim\dimen@<\z@-\fi.5\dimen@ii
  \divide\dimen@ by#1\relax #2=\dimen@}
```

Before constructing the curve we first examine the "drop" object, setting it in an `\xybox`. The size of this object determines the spacing of objects placed along the curve, via the *tolerance* of the spline. This determines the maximum separation of places at which objects will be dropped; *i.e.* when two adjacent places would be separated by more than this amount, the spline algorithm constructs another place on the curve intermediate between these two.

Note: Alter the spline tolerance by adjusting the size of the "drop" object. In particular, a dotted line can be achieved by setting an empty object with non-zero size. If the size is zero then the default tolerance is used. Initially this is .4pt; it may be altered using `\splinetolerance`⟨dimen⟩, where ⟨dimen⟩ must be greater than zero, else the initial tolerance is reset.

```
\xydef@\splinetolerance@#1{\dimen@=#1\relax
 \ifdim\dimen@>\z@ \splinetol@=\dimen@
 \else\splinetol@=.4\p@\fi}

\xylet@\splinedefaulttol@=\relax

\xydef@\splinetolerance#1{\splinetolerance@{#1}%
 \edef\splinedefaulttol@{\splinetol@=\the\splinetol@}}

\splinetolerance\z@

\xydef@\SloppyCurves{\splinetolerance{.8\p@}}

\xydef@\xylowtolerance@{\splinedefaulttol@
 \ifdim\splinetol@<\p@ \divide\splinetol@\tw@ \fi}
```

Similarly examine the "connect" object. If none has been specified then there is no need to calculate the tangent direction at each place along the curve. This saves on both time and memory requirement.

Procedure: (1) If `\xycrvdrop@` is empty, define `\splinedrop@` to expand to `\copy\zerodotbox@`, thus creating a curve constructed from small dots. Otherwise (2a) set the `\splinebox@` with the specified object and (2b) use `\copy\splinebox@` as the expansion of `\splinedrop@`. (3) Set `\splinetol@` to be $\sqrt{2}\times$ maximum of width and height+depth of the box.

If `\xycrvconn@` is empty (4) then `\splineconn@` is set to `\relax`. (5) Otherwise it is necessary to reset the direction at each place along the curve before dropping the object specified by `\xycrvconn@`.

```
\xywarnifdefined\splinedrop@
\xydef@\setstdsplinedrop@{%
 \setbox\splinebox@=\hbox\bgroup
  \setboxz@h{\expandafter\object\xycrvdrop@{}}%
  \splinedefaulttol@ \dimen@=\wd\z@ \dimen@ii=\ht\z@ \advance\dimen@ii\dp\z@
  \ifdim\dimen@ii>\dimen@ \dimen@=\dimen@ii \fi \dimen@=1.4142\dimen@
  \ifdim\dimen@>\splinetol@ \splinetol@=\dimen@\else\dimen@=\splinetol@\fi
  \dimen@=.5\wd\z@
  \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \kern-\dimen@ \Drop@@ \kern\dimen@
  \edef\tmp@{\egroup \splinetol@=\the\splinetol@
   \ifInvisible@\noexpand\Invisible@true
   \else\noexpand\Invisible@false\fi}\tmp@
 \edef\splinedefaulttol@{\splinetol@=\the\splinetol@}%
 \def\splinedrop@{\copy\splinebox@\splineextra@@}}
```

```
\xywarnifdefined\splineconn@
\xydef@\stdsplineconn@{{\setsplinedir@ %  %(5)
  \setboxz@h\bgroup\expandafter\object\xycrvconn@{}%
   \edef\tmp@{\egroup\L@c=\the\L@c \R@c=\the\R@c \D@c=\the\D@c \U@c=\the\U@c
    \noexpand\def\noexpand\Drop@@}\expandafter\tmp@\expandafter{\Drop@@}%
  \setboxz@h{\kern-\L@c \boxz@}%
  \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \Drop@@}}
```

```
\xydef@\setcrvobjects@{\setstdsplinedrop@ \def\splineconn@{\stdsplineconn@}}
\xylet@\splineextra@@=\relax
```

```
\xydef@\splinenear@#1#2#3#4{{\d@X=#1\advance\d@X-#2%
 \ifdim\d@X<\z@ \d@X=-\d@X\fi
 \ifdim\splinetol@<\d@X \aftergroup\splinefar@true
 \else \d@Y=#3\advance\d@Y-#4\ifdim\d@Y<\z@ \d@Y=-\d@Y\fi
  \ifdim\splinetol@<\d@Y\aftergroup\splinefar@true
  \else \xydist@\d@X\d@Y
   \ifdim\splinetol@<\dimen@ \aftergroup\splinefar@true
   \else
    \ifdim\dimen@<.1\splinetol@\aftergroup\splinetooclose@\fi
 \fi\fi\fi}}
```

```
\xydef@\splineadvance@@{\global\advance\splineval@\splinedepth@}
```

This stops objects being placed too close together along the spline. It is necessary at the beginning of a segment with a non-zero edge. Otherwise, with a large `\splinetolerance@`, the accuracy of the place-finding mechanism results in being so deep in the tree that objects placed at every level on the way up would be much too close.

```
\xydef@\splinetooclose@{\aftergroup\aftergroup\aftergroup\splinetooclose@@}
\xydef@\splinetooclose@@{\def\splineplotpt@{\relax}}
```

Each place along the curve is tested according to a `\splinetest@`. Normally this is for visibility according to the extent of the objects at the end-points of the curve. When an edge is crossed then the `\splinecontinue` routine is called to decide how the behaviour may change e.g. whether the recursion should continue or exit to the top of the existing tree.

```
\xydef@\splinepoint@@{%
\splinetrace@{P@:(\the\X@c,\the\Y@c)::\the\splineval@;\the\splinedepth@}%
 {\splinetest@\ifsplineplot@\relax\else\aftergroup\splinecontinue\fi}}
```

```
\xydef@\splinepoint@{%
 {\splinetest@\ifsplineplot@\aftergroup\splinecontinue\fi}}
\xylet@\splinepoint=\splinepoint@
```

```
\xydef@\splinecontinue@{%
 \global\let\splinepoint=\splinepoint@@
 \global\let\splinecontinue=\splinecontinue@@}
\xydef@\splinecontinue@@{%
 \global\let\splinepoint=\splinepoint@
 \global\let\splinecontinue=\relax}
\xylet@\splinecontinue=\splinecontinue@
```

Points where segments join lie at the top of the recursive tree. They are tested directly against the objects at the endpoints, in order to decide whether or not they are visible.

```
\xydef@\splineknotpoint@{{\X@p=\X@c \Y@p=\Y@c \cv@start
 \the\Edge@c\@ne \ifInside@\else\aftergroup\splineknotpoint@@\fi}}%
\xydef@\splineknotpoint@@{{\X@p=\X@c \Y@p=\Y@c \cv@end
 \the\Edge@c\@ne \ifInside@\else \aftergroup\splinepoint\fi}}%
```

```
\xylet@\firstsplinepoint@=\splineknotpoint@
\xylet@\lastsplinepoint@= \splineknotpoint@
```

This routine is used to find where the spline crosses the edges of the objects at p and c. First (1) we set up the tests which will determine when the edge has been crossed. Then commence the scan. (2)

```
\xydef@\splined@@@{%
\splinetrace@{finding end types}%
\splinetrace@{bstartPLACE@:\bstartPLACE@; bendPLACE@:\bendPLACE@}%
 \DN@{\splinescan@}\ifx\bstartPLACE@\empty
   \ifx\bendPLACE@\relax\def\bendPLACE@{1}\fi
  \else\ifx\bstartPLACE@\relax
  \else\ifx\bstartPLACE@\undefined
  \else
   \DN@{\splinescanbreak@}\ifx\bendPLACE@\relax\def\bendPLACE@{1}\fi
   \DNii@{0}\ifx\bstartPLACE@\nextii@
    \DNii@{1}\ifx\bendPLACE@\nextii@\DN@{}%
%    \else\DNii@{0.0}\ifx\bendPLACE@\nextii@\DN@{}\fi
   \fi\fi  % use the saved values
  \fi\fi\fi \next@ }%  %(2)
```

```
\xydef@\splineStarttest@{{\X@p=\X@c \Y@p=\Y@c \cv@start \the\Edge@c\@ne
  \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}%
\xydef@\splineEndtest@{{\X@p=\X@c \Y@p=\Y@c \cv@end \the\Edge@c\@ne
  \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}%
```

```
\xydef@\splineBreakStarttest@{{\X@p=\X@c \Y@p=\Y@c \cv@brstart \the\Edge@c\@ne
```

```
  \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}%
```

```
\xydef@\splineBreakEndtest@{{\X@p=\X@c \Y@p=\Y@c \cv@brend \the\Edge@c\@ne
  \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}%
%  \ifInside@\aftergroup\splineplot@true\else\aftergroup\splineplot@false\fi}}%
```

```
\xylet@\splineStarttest=\splineStarttest@
\xylet@\splineEndtest=\splineEndtest@
```

```
\xydef@\splineSlowStarttest{{\X@p=\X@c \Y@p=\Y@c \cv@start \the\Edge@c\@ne
  \ifInside@\aftergroup\splineplot@maybe@@\else\aftergroup\splineplot@true\fi}}%
\xydef@\splineSlowEndtest{{\X@p=\X@c \Y@p=\Y@c \cv@end \the\Edge@c\@ne
  \ifInside@\aftergroup\splineplot@maybe@@\else\aftergroup\splineplot@true\fi}}%
\xydef@\splineplot@maybe{\splineplot@false}%
\xylet@\splineplot@maybe@@=\splineplot@maybe
```

(3) Start a new group and bind some macros appropriate to finding the starting edge. (4) start the spline algorithm, using a small tolerance. (5) Close the group, thus removing the bindings of (3). (6) Record the information, which has been stored globally. (7) Do the same for the ending edge. (8) Clear the global information. After this, set the curve as usual. (9)

```
\xydef@\splinescan@{\bgroup%
 \expandafter\def\xysplineedges@{\z@;\z@,\z@;\z@,}%
 \bgroup \cv@start
  \expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}%
  \ifx\next@\nextii@ \gdef\splineedges@{\z@;\z@,}%
  \else\aftergroup\splinestartScan@\fi
 \egroup \relax
 \bgroup \cv@end
  \expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}%
  \ifx\next@\nextii@
   \getcrvsegsnum@ \edef\segmentnum@{\the\count@}%
   \splineval@=\z@ \splineRecordEndValue@
  \else\aftergroup\splineendScan@\fi
 \egroup \egroup
 \expandafter\testsplineedges@\splineedges@;,@@%
 \expandafter\edef\xysplineedges@{\splineedges@}%  %(6)
 \global\let\splineedges@=\relax %  %(8)
\splinetrace@{edge params: \xysplineedges@}}%
```

```
\xydef@\testsplineedges@#1;#2,#3;#4,#5@@{%
 \DN@{#5}\ifx\next@\empty
  \DNii@{#4}\ifx\nextii@\empty
   \xynoedgespline@{end}\def\splineedges@{#1;#2,\z@;\z@,}\fi
  \DNii@{#2}\ifx\nextii@\empty
   \xynoedgespline@{start}\def\splineedges@{\z@;\z@,\z@;\z@,}\fi
 \fi }
\xydef@\xynoedgespline@#1{\xywarning@{#1 edge of spline not found}}%
```

```
\xydef@\splinestartScan@{\bgroup % %(3)
\splinetrace@{scan for start}%
 \global\let\splineadvance@=\splineadvance@@  %(3)
 \let\spline@start=\splinefindStart@   %(3)
 \let\spline@end=\spline@end@@   %(3)
```

```
 \let\splinefinish@=\splinefinish@@
 \let\splinewhich=\relax
 \global\splinelength@=\z@
% \let\splinerec@=\splineSrec@
 \def\splinerec@{\let\splinerec@=\splineSrec@ \splinedecast@@}%
 \edef\splineedges@{}%
 \expandafter\let\xysplineedges@\relax
% \bgroup
  \xylowtolerance@ \splined@@
  \ifx\splineedges@\empty\expandafter\splineslowScan@\fi
  \ifx\splineedges@\relax\expandafter\splinenostart@\fi
 \egroup }%  %(4)

\xydef@\splineslowScan@{%
 \let\splineStarttest=\splineSlowStarttest
 \let\splinescanStarttest=\relax
 \def\splinerec@{\let\splinerec@=\splineSrec@ \splinedecast@@}%
 \splined@@ }

\xydef@\splineslowEnd@{%
 \global\let\splineEndtest=\splineSlowEndtest
 \let\splinescanEndtest=\relax
 \def\splinerec@{\let\splinerec@=\splineErec@ \splinedecast@@}%
 \splined@@ }

\xydef@\splinenostart@{%
 \def\segmentnum@{1}\splinelength@=\z@ \splineval@=\z@
 \xynoedgespline@{start}\splineRecordValue@ }

\xydef@\splineendScan@{\bgroup % %(3)<-----------
 \bgroup %  %(7)
\splinetrace@{scan for end}%
 \global\let\splineadvance@=\splineadvance@@
 \let\spline@start=\splinefindEnd@
 \let\spline@end=\spline@end@@
 \let\splinefinish@=\splinefinish@@
 \let\splinewhich=\relax
 \global\splinelength@=\z@
% \let\splinerec@=\splineErec@
 \global\let\splineEndtest=\splineEndtest@
 \def\splinerec@{\let\splinerec@=\splineErec@ \splinedecast@@}%
 \expandafter\def\expandafter\prevedges@\expandafter{\splineedges@}%
 \xylowtolerance@ \splined@@
 \ifx\splineedges@\prevedges@\expandafter\splineslowScan@\fi
 \egroup \egroup}

\xydef@\splinescanbreak@{%
\splinetrace@{SCANBREAK}%
 \gdef\breakedges@{}%
 \DN@{0}\ifx\next@\bstartPLACE@
  \DN@{\bgroup \def\tmp@####1;####2,####3;####4,{%
    \global\splineval@=####2\relax}%
   \expandafter\expandafter\expandafter\tmp@\xysplineedges@
```

```
   \def\segmentnum@{1}\splineRecordBreakValue@ \egroup}%
 \else
  \bgroup \cv@start
   \expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}%
   \ifx\next@\nextii@ \aftergroup\splinezerostart
   \else\aftergroup\splinewidestart\fi
  \egroup
 \fi \next@
 \DN@{1}\ifx\next@\bendPLACE@
  \DN@{\bgroup \def\tmp@####1;####2,####3;####4,{%
    \global\splineval@=####4\relax}%
   \expandafter\expandafter\expandafter\tmp@\xysplineedges@
%%   \multiply\splineval@\m@ne \advance\splineval@\p@
%%   \def\segmentnum@{1}
  \getcrvsegsnum@ \edef\segmentnum@{\the\count@}%
  \global\multiply\splineval@\count@
  \splineRecordEndBreakValue@ \egroup}%
 \else
  \bgroup \cv@end
   \expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}%
   \ifx\next@\nextii@ \aftergroup\splinezeroend
   \else\aftergroup\splinewideend\fi
  \egroup
 \fi \next@
 \expandafter\def\expandafter
  \xybreakedges@\expandafter{\breakedges@}%  %(6)
 \global\let\breakedges@=\relax %  %(8)
\splinetrace@{break params: \xybreakedges@::\bstartPLACE@--\bendPLACE@}%
 \expandafter\let\xysplineedges@\xybreakedges@ }

\xydef@\getcrvsegsnum@{% sets \count@
  \count@=\ptsnum@\relax \ifnum\count@>\@ne\advance\count@\m@ne\fi}

\xydef@\splinezerostart{\DN@{\bgroup
\splinetrace@{splinezerostart:}%
  \dimen@=\bstartPLACE@\p@
  \global\splineval@=\dimen@
  \splineRecordBreakValue@ \egroup }}

\xydef@\splinezeroend{\DN@{\bgroup
\splinetrace@{splinezeroend:}%
  \dimen@=\bendPLACE@\p@
  \multiply\dimen@\m@ne \advance\dimen@\p@
  \def\segmentnum@{0}\global\splineval@=\dimen@
  \splineRecordEndBreakValue@ \egroup }}

\xydef@\splinewidestart{\DN@{\bgroup
 \splinetrace@{scan for start}%
  \let\splinefbcontinue@=\breakstartcontinue@
  \let\splinef@pt=\splinef@breakpt
  \let\splinef@end=\break@start
  \edef\tmp@{{\bstartPLACE@}}%
```

```
  \expandafter\splinealong@@\tmp@\z@>\endgroup\egroup }}
```

```
\xydef@\splinewideend{\DN@{\bgroup
 \splinetrace@{scan for wide end}%
  \let\splinefbcontinue@=\breakendcontinue@
  \let\splinef@pt=\splinef@breakpt
  \let\splinef@end=\break@end
  \edef\tmp@{{\bendPLACE@}}%
  \expandafter\splinealong@@\tmp@\z@<\endgroup\egroup }}
```

```
\xydef@\splineSrec@{{%
\splinetrace@{v=\the\splineval@ + d=\the\splinedepth@ :
 (\the\X@p,\the\Y@p);(\the\X@c,\the\Y@c):Srec }%
 \splineStarttest
 \ifsplineplot@
  \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@
   \global\advance\splineval@ .5\splinedepth@
  \else\aftergroup\splinedecast@@ \fi
 \else \aftergroup\splineadvance@ \fi}}
```

```
\xydef@\splinefindStart@@{\bgroup \bgroup
 \expandafter\expandafter\expandafter\getsplineparams@\xysplineparams@
 \global\let\splineadvance@=\splineadvance@@
 \global\let\splinepoint=\relax
 \let\splinegoal@=\splineRecordValue@ \global\dimen5=\z@
 {\X@c=\X@p \Y@c=\Y@p \splinescanStarttest }%% first test initial point.
}
```

```
\xydef@\splinefindStart@{%
 \expandafter\crv@cnt@\xycrvptsnum@\relax
 \ifnum\crv@cnt@>\tw@
  \crv@cnt@=\@ne \splineplot@false \splinesegment@{\crv@cnt@}%
  {\splineStarttest %% test end of segment.
  \ifsplineplot@\aftergroup\splineplot@true\fi}%
  \ifsplineplot@ \else \searchforStartsegment@ \fi
  \splinesegment@{\crv@cnt@}%
\splinetrace@{start is in segment \segmentnum@}%
 \else \splinesegment@\@ne \fi
 \splineplot@false  \splinefindStart@@
}
```

This finds the first spline segment whose endpoint is beyond the edge of the object.

```
\xydef@\searchforStartsegment@{\splineplot@true\loop@
   \advance\crv@cnt@\@ne
   \expandafter\count@\xycrvptsnum@\relax
   \ifnum\count@=\crv@cnt@\expandafter\splineplot@false
   \else
    \splinesegment@{\crv@cnt@}{\splineStarttest
    \ifsplineplot@\aftergroup\splineplot@false
    \else\aftergroup\splineplot@true\fi }%
   \fi \ifsplineplot@\repeat@
}
```

Finding the end is similar, but the spline is searched in reverse order.

```
\xydef@\splinefindEnd@@{\bgroup \bgroup
 \expandafter\expandafter\expandafter\getsplineparams@\xysplineparams@
\splinetrace@{params:\xysplineparams@}%
 \splinereverse@@
\splinetrace@{params:\xysplineparams@}%
 \global\let\splineadvance@=\splineadvance@@
 \global\let\splinepoint=\splinescanEndtest
 \let\splinegoal@=\splineRecordEndValue@ \global\dimen5=\z@
 {\X@c=\X@p \Y@c=\Y@p \splinescanEndtest }%% first test initial point.
}

\xydef@\splinereverse@@{\splinereverse@@@}
\xydef@\splinereverse@@@{%
 \def\postspline@{\d@X=-\d@X \d@Y=-\d@Y}%
 \dimen@ii=\X@c \X@c=\X@p \X@p=\dimen@ii
 \dimen@ii=\Y@c \Y@c=\Y@p \Y@p=\dimen@ii
 \global\dimen5=-\dimen5\relax
 \global\advance\dimen5\splinelength@ }
\xydef@\postspline@{}

\xydef@\splineErec@{{%
 \dimen@=\splineval@ \advance\dimen@\splinedepth@ %
\splinetrace@{ER (\the\X@p,\the\Y@p);(\the\X@c,\the\Y@c);;;
\the\dimen@,\the\splinedepth@}%
 \splineEndtest
 \ifsplineplot@
  \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@
%   \global\advance\splineval@ .5\splinedepth@
  \else\aftergroup\splinedecast@@ \fi
 \else \aftergroup\splineadvance@ \fi}}

\xydef@\splinefindEnd@{%
 \expandafter\crv@cnt@\xycrvptsnum@\relax
 \ifnum\crv@cnt@>\tw@
  \advance\crv@cnt@\m@ne
  \splineplot@false \splinesegment@{\crv@cnt@}%
  {\X@c=\X@p \Y@c=\Y@p \splineEndtest %% test end of segment.
  \ifsplineplot@\aftergroup\splineplot@true\fi }%
  \ifsplineplot@
  \else \searchforEndsegment@ \fi
  \splinesegment@{\crv@cnt@}%
\splinetrace@{end is in segment: \segmentnum@ }%
 \else \splinesegment@\@ne \fi
 \splineplot@false\splinefindEnd@@
}
\xydef@\searchforEndsegment@{\splineplot@true\loop@
   \advance\crv@cnt@\m@ne
   \ifnum\crv@cnt@=\z@\expandafter\splineplot@false
   \else
    \splinesegment@{\crv@cnt@}{\X@c=\X@p \Y@c=\Y@p \splineEndtest
```

```
    \ifsplineplot@\aftergroup\splineplot@false
    \else\aftergroup\splineplot@true\fi}%
   \fi \ifsplineplot@\repeat@
}
```

These are the tests, to determine when the edge-point has been found.

```
\xydef@\splinescanStarttest{%
\splinetrace@{SST (\the\X@c,\the\Y@c); \the\splinelength@, \the\splineval@}%
 {\splineplot@false\splineStarttest
   \ifsplineplot@\expandafter\splinefinish@\fi}}
```

```
\xydef@\splinescanEndtest{%
\splinetrace@{SET (\the\X@c,\the\Y@c); \the\splinelength@, \the\splineval@}%
 {\splineplot@false\splineEndtest
   \ifsplineplot@\expandafter\splinefinish@\fi}}
```

```
\xydef@\splineBrec@{{%
 \dimen@=\splineval@ \advance\dimen@\splinedepth@ %
\splinetrace@{BR (\the\X@p,\the\Y@p);(\the\X@c,\the\Y@c);;;
\the\dimen@,\the\splinedepth@}%
 \splineEndtest
 \ifsplineplot@ \aftergroup\splineadvance@ \else
  \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@
%   \global\advance\splineval@ .5\splinedepth@
  \else\aftergroup\splinedecast@@ \fi \fi}}
```

```
\xydef@\splineBSrec@{{%
\splinetrace@{SR (\the\X@p,\the\Y@p);(\the\X@c,\the\Y@c);;;
\the\dimen@,\the\splinedepth@:SR}%
 \splineStarttest
 \ifsplineplot@
  \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@
   \global\advance\splineval@ .5\splinedepth@
  \else \advance\splineval@\splinedepth@
   \splinetest@ \advance\splineval@-\splinedepth@
   \ifsplineplot@\aftergroup\splineadvance@
   \else\aftergroup\splinedecast@@\fi
  \fi
 \else
  \global\advance\splineval@\splinedepth@
  \aftergroup\splinefinish@ \fi }}
```

```
\xydef@\splineBErec@{{%
 \dimen@=\splineval@ \advance\dimen@\splinedepth@ %
\splinetrace@{BE (\the\X@p,\the\Y@p);(\the\X@c,\the\Y@c);;;
 \the\dimen@,\the\splinedepth@}%
  \splineEndtest
 \ifsplineplot@
  \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@
   \global\advance\splineval@ .5\splinedepth@
  \else \advance\splineval@\splinedepth@
   \splinetest@ \advance\splineval@-\splinedepth@
```

```
   \ifsplineplot@\aftergroup\splineadvance@
   \else\aftergroup\splinedecast@@\fi
  \fi
 \else
  \ifdim\splinedepth@<\p@ \global\advance\splineval@\splinedepth@ \fi
  \aftergroup\splinefinish@ \fi }}

\xydef@\splinef@breakpt#1{% gobbles |\splinecancel|
  \ifdim\splineval@>\z@
\splinetrace@{found: val=\the\splineval@;(\the\X@c,\the\Y@c)}%
  \else
\splinetrace@{found: val=\the\splineval@;(\the\X@p,\the\Y@p)}%
  \fi  \splinefbcontinue@ }

\xydef@\splinef@pt@@{%
  \ifdim\splineval@>\z@
\splinetrace@{found: val=\the\splineval@;c:(\the\X@c,\the\Y@c)}%
   \global\dimen@i=\X@c \global\dimen3=\Y@c
  \else
\splinetrace@{found: val=\the\splineval@;p:(\the\X@p,\the\Y@p)}%
  \fi
 \setsplinedir@ \global\dimen5=\d@X \global\dimen7=\d@Y
  \global\splineval@=\splineval@
  \aftergroup\splinefocus@ }

\xydef@\findbreakwarning@#1{%
 \xywarning@{#1 of break not found: bSTART=\bstartPLACE@, bEND=\bendPLACE@}%
 \DN@{#1}\def\DNii@{start}\ifx\next@\nextii@\splineRecordBreakValue@
 \else\splineRecordEndBreakValue@\fi\egroup}%

\xydef@\breakstartcontinue@{\bgroup \cv@start
%\xystatus@{SS}%
  \expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}%
  \ifx\next@\nextii@ \aftergroup\breakstartfound@
  \else \aftergroup\breakstartcontinue@i\fi \egroup }%

\xydef@\breakstartcontinue@i{%
\splinetrace@{move to start edge, from (\the\X@c,\the\Y@c): val=\the\splineval@ }%
% \global\let\splinetest@=\splineStarttest
 \global\let\splineEndtest=\splineBreakStarttest@
 \global\let\splinetest@=\splineBreakStarttest@
% \global\let\splinerec@=\splineSrec@
 \gdef\splinerec@{\global\let\splinerec@=\splineSrec@ \splinedecast@@}%
 \global\let\splinepoint=\relax
 \global\let\spline@end=\checkfoundSbreak@
% \global\let\splinegoal@=\splineRecordBreakValue@
 \global\let\splinegoal@=\splineRecordSBreakValue@
 \global\let\splinefinish@=\splinefinish@@  }

\xydef@\breakstartfound@{%
\splinetrace@{found start edge, at (\the\X@c,\the\Y@c): val=\the\splineval@ }%
 \global\let\splinetest@=\splineStarttest
 \gdef\splinerec@{\splinegoal@ \global\let\splinerec@=\splinefinish@ }%
```

```
 \global\let\splinepoint=\relax
 \global\let\spline@end=\relax %\checkfoundSbreak@
 \global\let\splinegoal@=\splineRecordBreakValue@
 \global\let\splinefinish@=\splinefinish@@
 }
\xydef@\checkfoundSbreak@{%
  \ifnum\xycrvptsnum@<\thr@@\DN@{\findbreakwarning@{start}}%
  \else\DN@{\searchBreakSsegment@}\fi \next@ }%
\xydef@\searchBreakSsegment@{%
 \crv@cnt@=\segmentnum@ \DN@{}\count@=\crv@cnt@
 \DNii@{}\ifnum\splineval@=\z@ \ifnum\crv@cnt@=\@ne
   \ifnum\xycrvptsnum@=\@ne\count@=\z@\DNii@{\egroup}\fi
  \else \DN@{\egroup}\fi
 \else \ifnum\splineval@=\m@ne
 \else \advance\crv@cnt@\@ne \fi \fi
 \ifnum\crv@cnt@=\xycrvptsnum@\relax
  \ifnum\xycrvptsnum@>\tw@ \DNii@{\egroup}\fi
  \DN@{\expandafter\splineRecordBreakValue@\nextii@}%
 \else
  \DN@{\splinesegment@{\crv@cnt@}\splinesetparams@
   \splined@@ \egroup}%
 \fi \next@ }
\xydef@\checkfoundEbreak@{%
  \ifnum\xycrvptsnum@<\thr@@\DN@{\findbreakwarning@{end}}%
  \else\DN@{\searchBreakEsegment@}\fi \next@ }%
\xydef@\searchBreakEsegment@{%
 \crv@cnt@=\segmentnum@ \DN@{}\count@=\crv@cnt@
 \DNii@{}\ifnum\splineval@=\z@ \ifnum\crv@cnt@=\@ne
   \ifnum\xycrvptsnum@=\@ne\count@=\z@\DNii@{\egroup}\fi
  \else\DN@{\egroup}\fi
 \else \advance\crv@cnt@\m@ne \fi
 \ifnum\count@=\z@
% \ifnum\crv@cnt@=\z@
  \ifnum\xycrvptsnum@>\tw@ \splineval@=\m@ne\p@
   \DNii@{\egroup}\fi
  \DN@{\expandafter\splineRecordEndBreakValue@\nextii@}\else
  \DN@{\splinesegment@{\crv@cnt@}\splinereverse@
   \readsplineparams@ \splined@@ \egroup}%
 \fi \next@ }
\xydef@\break@start{\egroup \egroup
\splinetrace@{scan for start}%
 \global\let\splineadvance@=\splineadvance@@  %(3)
 \let\spline@start=\splinefindBStart@   %(3)
 \let\spline@end=\egroup
 \let\splinefinish@=\splinefinish@@
 \let\splinewhich=\relax
 \global\let\splineadvance@=\splineadvance@@
 \global\let\splinepoint=\relax
```

```
 \let\splinegoal@=\splineRecordBreakValue@
 \global\dimen5=\z@  \global\splinelength@=\z@
 \let\splinerec@=\splineSrec@
 \def\breakedges@{}%
 \bgroup \bgroup \splined@@ }

\xydef@\splinefindBStart@{\bgroup
% \global\splineval@=\z@ \global\splinedepth@=.5\p@}
 \global\splineval@=\z@ \splinedepth@=.5\p@}

\xydef@\breakendcontinue@{\bgroup
 \edef\tmp@{\X@p = \the\X@c \Y@p=\the\Y@c}\cv@end\tmp@ % <--- irrelevant ?
 \expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}%
%\xystatus@{EE}%
 \ifx\next@\nextii@ \aftergroup\breakendfound@
 \else\aftergroup\breakendcontinue@i\fi \egroup }%

\xydef@\breakendcontinue@i{%
\splinetrace@{move to end edge, from (\the\X@c,\the\Y@c): val=\the\splineval@ }%
% \global\let\splinetest@=\splineEndtest
 \global\let\splineEndtest=\splineBreakEndtest@
 \global\let\splinetest@=\splineBreakEndtest@
 \gdef\splinerec@{\global\let\splinerec@=\splineErec@\splinedecast@@}%
 \global\let\splinepoint=\relax
 \global\let\spline@end=\checkfoundEbreak@
% \global\let\splinegoal@=\splineRecordEndBreakValue@
 \global\let\splinegoal@=\splineRecordEBreakValue@
 \global\let\splinefinish@=\splinefinish@@
 \splinerec@ }

\xydef@\splineRecordEBreakValue@{%\getcrvsegsnum@
\splinetrace@{\the\count@\space segments, \the\splineval@}%
%  \ifnum\segmentnum@>\z@ \dimen@=\segmentnum@\p@ \else \dimen@=\p@\fi
%  \advance\dimen@-\splineval@\relax
%  \ifnum\count@>\@ne \divide\dimen@\count@\fi
%  \global\splineval@=\dimen@
  \splineRecordEndBreakValue@ }

\xydef@\splineRecordSBreakValue@{\getcrvsegsnum@
  \dimen@=\splineval@
  \advance\dimen@\segmentnum@\p@ \advance\dimen@-\p@
  \dimen@nth\count@\dimen@
  \global\splineval@=\dimen@ \def\segmentnum@{1}%
  \splineRecordBreakValue@ }

\xydef@\breakendfound@{%
\splinetrace@{found end edge, at (\the\X@c,\the\Y@c): val=\the\splineval@ }%
 \global\let\splinetest@=\splineEndtest
 \gdef\splinerec@{\splinegoal@ \global\let\splinerec@=\splinefinish@}%
 \global\let\splinepoint=\relax
 \global\let\spline@end=\relax  %\checkfoundEbreak@
 \global\let\splinegoal@=\splineRecordEndBreakValue@
 \global\let\splinefinish@=\splinefinish@@  }
```

```
\xydef@\break@end{\egroup \egroup
 \global\let\splineadvance@=\splineadvance@@
 \let\spline@start=\splinefindBEnd@
 \let\spline@end=\egroup
 \let\splinefinish@=\splinefinish@@
 \let\splinewhich=\relax
 \global\let\splineadvance@=\splineadvance@@
 \global\let\splinegoal@=\splineRecordEndBreakValue@
 \global\splinelength@=\z@
 \let\splinerec@=\splineErec@
 \global\let\splineEndtest=\splineEndtest@
 \bgroup \splined@@ }
```

```
\xydef@\splinefindBEnd@{\bgroup\bgroup
 \splinereverse@ \readsplineparams@
 \global\splineval@=\z@ }
```

This appends the new information to that currently stored in the control sequence referenced by `\xysplineparams@`. It is temporarily stored globally in `\splineparams@`, to be later transferred to (the c.s. referenced by) `\xysplineparams@` when at the appropriate level of groupingl.

```
\xydef@\splineRecordValue@{%
 \ifx\unknown\segmentnum@\relax
 \else
  \getcrvsegsnum@
  \dimen@=\splineval@ \advance\dimen@ \segmentnum@\p@
  \advance\dimen@-\p@ \dimen@nth\count@\dimen@
  \global\splineval@=\dimen@
 \fi
 \xdef\splineedges@{\splineedges@\the\splinelength@;\the\splineval@,}%
\splinetrace@{found edge: \splineedges@}}
```

```
\xydef@\splineRecordEndValue@{%
 \ifx\unknown\segmentnum@\relax
 \else
  \getcrvsegsnum@
  \dimen@=\splineval@ \advance\dimen@ \count@\p@
  \advance\dimen@-\segmentnum@\p@ \dimen@nth\count@\dimen@
  \global\splineval@=\dimen@
 \fi
 \xdef\splineedges@{\splineedges@\the\splinelength@;\the\splineval@,}%
\splinetrace@{found edge: \splineedges@}}
```

```
\xydef@\splineRecordBreakValue@{%
 \ifx\unknown\segmentnum@\relax
 \else\expandafter\advance\expandafter\splineval@\segmentnum@\p@
  \count@=\segmentnum@\relax
  \ifnum\count@>0 \advance\splineval@-\p@ \fi\fi
% \ifx\breakedges@\relax\relax\else\ifx\breakedges@\empty\else
%  \ifdim\splineval@<\p@\ifdim\splineval@>\z@
%       \expandafter\testbreakedges@\breakedges@ @%
%  \fi\fi\fi\fi
 \xdef\breakedges@{\breakedges@\the\splinelength@;\the\splineval@,}%
```

```
 \global\let\spline@end=\egroup
\splinetrace@{found break edge: \breakedges@}}

\xydef@\splineRecordEndBreakValue@{%
 \ifx\unknown\segmentnum@\relax
 \else\count@=\segmentnum@\relax \ifnum\count@=\z@
  \else \getcrvsegsnum@
   \count@@=\count@ \advance\count@@-\segmentnum@\relax
   \dimen@=\splineval@ \advance\dimen@ \count@@\p@
   \dimen@nth\count@\dimen@
   \global\splineval@=\dimen@
 \fi\fi
% \multiply\splineval@\m@ne \advance\splineval@\p@ % <----------  !!!
 \ifdim\splineval@<\z@ \relax\splineval@=\z@ \fi
 \ifx\breakedges@\relax\relax\else\ifx\breakedges@\empty\else
  \ifdim\splineval@<\p@\relax \ifdim\splineval@>\z@
       \expandafter\testbreakedges@\breakedges@ @%
  \fi\fi\fi\fi
 \xdef\breakedges@{\breakedges@\the\splinelength@;\the\splineval@,}%
 \global\let\spline@end=\egroup
\splinetrace@{found break edge: \breakedges@}}

\xydef@\testbreakedges@#1;#2,#3@{\bgroup \dimen@=#2\relax
 \advance\dimen@\splineval@ \advance\dimen@-\p@
\splinetrace@{testbreak: #1;#2,#3@  \the\splineval@::\the\dimen@}%
 \DN@{\egroup}\ifdim\dimen@>\z@
  \edef\next@{\egroup\def\noexpand\breakedges@{#1;1pt,#3}}%
 \else\ifdim\zz@\dimen@
  \edef\next@{\egroup\def\noexpand\breakedges@{#1;1pt,#3}}%
 \fi \fi \next@ }

\xydef@\splineRecordBreakEValue{%
 \ifdim\splineval@>\z@
  \splineval@=-\splineval@ \advance\splineval@\p@ \fi
 \splineRecordBreakValue@ }%
```

Use the following distance approximation:

$$dist = \begin{cases} dX + .5dY/dXdY & \text{if } dY/dXdY < (\sqrt{2}-1)\times dX \\ \frac{3\sqrt{2}}{4}dX + \frac{\sqrt{2}}{4}dY/dXdY & \text{if } (\sqrt{2}-1)\times dX < dY/dXdY < dX \end{cases}$$

and similarly, interchanging dY and dX when $dY > dX$.

$\sqrt{2} \approx 1.41422$, $(\sqrt{2})/2 \approx 0.70711$, $(\sqrt{2})/4 \approx 0.353555$, $3(\sqrt{2})/4 \approx 1.060665$

```
\xydef@\xydist@#1#2{\ifdim#1>#2\relax
  \quotient@@\tmp@#2#1\expandafter\dimen@ii\tmp@#2\relax
  \ifdim\dimen@ii>.41422#1\relax
   \dimen@=1.060665#1\advance\dimen@ .353555\dimen@ii\relax
  \else \dimen@=#1\advance\dimen@.5\dimen@ii\fi
 \else
  \quotient@@\tmp@#1#2\expandafter\dimen@ii\tmp@#1\relax
  \ifdim\dimen@ii>.41422#2\relax
   \dimen@=1.060665#2\advance\dimen@ .353555\dimen@ii\relax
```

```
  \else \dimen@=#2\advance\dimen@.5\dimen@ii\fi
 \fi}
```

```
\xydef@\splineDadvance@{\splineadvance@@
  \expandafter\splineDadvance@@\splineDadvance@@@}
\xydef@\splineDadvance@@#1#2#3#4{%
 {\d@X=#1\advance\d@X-#2\ifdim\d@X<\z@ \d@X=-\d@X\fi
  \d@Y=#3\advance\d@Y-#4\ifdim\d@Y<\z@ \d@Y=-\d@Y\fi
 \xydist@\d@X\d@Y
 \global\advance\splinelength@\dimen@
\splinetrace@{d=(\the\d@X,\the\d@Y): v=\the\splineval@: l=\the\splinelength@}%
}}
```

```
\xydef@\spline@end@{\lastsplinepoint@ \egroup \egroup
 \ht\z@=\z@ \dp\z@=\z@ \wdz@=\z@ \boxz@}%
\gdef\splinetest@{\splineplot@true}
\xydef@\spline@end@@{\egroup \egroup}
```

...value for `\splinerec@` for the fast find...

```
\xydef@\splinefrec@{{%
 \advance\splineval@\splinedepth@
\splinetrace@{F:(\the\X@p,\the\Y@p);(\the\X@c,\the\Y@c);;
\the\splineval@;\the\splinedepth@}%
 \ifdim\dimen5=\splineval@ \aftergroup\splinefinishf@
 \else \dimen@=\dimen5\advance\dimen@-\splineval@
  \ifdim\dimen@<\z@ \dimen@=-\dimen@ \fi
  \ifdim\dimen@<.0001\p@ \aftergroup\splinefinishf@
  \else
   \ifdim\dimen5<\splineval@\aftergroup\splinedecast@@
   \else \aftergroup\splineadvance@
 \fi\fi\fi}}
```

```
\xydef@\splinefinishf@{\global\advance\splineval@\splinedepth@
 \splinefinish@}%
```

The macro `\spline@find#1#2` attempts to finds a specific point on a single spline segment. `#1` is the parameter value, in the range [0,1] on that segment; `#2` is a ⟨dimen⟩ denoting how much further to slide along the segment. It really only checks whether `#2` has a negative value before passing the information to `\spline@@find#1#2`, which does the actual search.

If `#2` is negative then the spline segment is searched in the reverse direction, starting at its endpoint. This is done by reversing the order of the control points, hence the parameter value found as x implies that we really want $1-x$ on the un-reversed segment. Similarly the resulting vaules for `\d@X` and `\d@Y` must be negated. The instructions to do this are loaded into a macro `\postfind@` which is expanded once the search has been completed. For an unreversed segment `\postfind@` expands to `{}`.

```
\xydef@\spline@find#1#2#3{%
\splinetrace@{spline@find #1 #2 #3}%
 \readsplineparams@
 \global\let\splinefinish@=\splinefinish@@
 \global\let\splinegoal@=\splinef@pt
 \global\let\splinepoint=\relax
 \dimen@=#1\relax\dimen@ii=#2\relax
```

```
 \ifdim\dimen@ii=\z@ \def\postfind@{}%
 \else\def\postfind@{}%
  \ifdim\dimen@ii<\z@ \splinereverse@
   \readsplineparams@
\splinetrace@{reverse orientation: (\the\X@p,\the\Y@p),(\the\X@c,\the\Y@c)}%
   \dimen@=#1\relax\dimen@ii=#2\relax
   \multiply\dimen@\m@ne \advance\dimen@\p@ \multiply\dimen@ii\m@ne
   \def\postfind@{\d@X=-\the\d@X \d@Y=-\the\d@Y \noexpand\reversesplineval@}%
 \fi\fi
 \expandafter\splinetrace@\expandafter{\xysplineparams@}%
  \edef\next@{\noexpand\spline@@find{\the\dimen@}{\the\dimen@ii}{#3}}%
 \next@ }
```

```
\xydef@\reversesplineval@{\splineval@=-\splineval@ \advance\splineval@\p@ }
```

```
\xydef@\spline@@find#1#2#3{%
\splinetrace@{spline@@find #1 #2 #3}%
  \let\splinerec@=\splinefrec@
  \global\let\splineadvance@=\splineadvance@@
  \dimen5=#1\relax
  \ifdim #2=\z@\relax
   \ifdim #1=\z@\relax \bgroup
    \let\splined@@=\spline@@knot
    \global\splineval@=\m@ne\p@
   \else\ifdim #1=\p@\relax \bgroup
     \let\splined@@=\spline@@knot
     \global\splineval@=\z@
   \else
    \global\splineval@=\z@
    \global\let\splineadvance@=\splineadvance@@
    \DN@{#3}\ifx\next@\empty \setsplinetest@\splineval@>{#1}{}%
      {{\ifdim\splineval@=\dimen5\aftergroup\splineplot@false\fi}}%
    \else\DNii@{>}\ifx\next@\nextii@
     \setsplineundertest@\splineval@>{#1}{}{}%
     \let\splinerec@=\splineBSrec@
    \else\DNii@{<}\ifx\next@\nextii@
     \setsplineundertest@\splineval@>{#1}{}{}%
     \let\splinerec@=\splineBErec@
    \else\xywarning@{unknown find-mode: #3}\fi\fi\fi
   \fi\fi
  \else
   \global\splineval@=\z@
   \def\splineslidetest@@{%
    \setsplinetest@\splinelength@<{#2}{}{}}%
   \let\splinefinish@=\splinefcontinue@
  \fi
  \let\spline@end=\splinef@end
  \let\splinepoint@=\relax
  \splinelength@=\z@
  \def\spline@start{\bgroup\xylowtolerance@}%
```

```
  \splined@@ }
\xydef@\spline@@knot{%
 \let\setsplinedir@=\setsplineknotdir@
 \splinesetparams@\spline@start
% \global\splinedepth@=\p@
 \splinedepth@=\p@
 \global\dimen@i=\X@p \global\dimen3=\Y@p
 \splinefinishf@ \spline@end \egroup }
\xydef@\splinefcontinue@{%
\splinetrace@{sliding... from (\the\X@c,\the\Y@c): val=\the\splineval@ }%
 \global\let\splinerec@=\splinerec@@
 \global\splinelength@=\z@
 \global\splineval@=\splineval@
 \global\let\splinepoint=\splinefindtest@
 \global\let\splinegoal@=\splinef@pt
 \global\let\splineadvance@=\splineDadvance@
 \global\let\splinefinish@=\splinefinish@@
 \splineslidetest@@
 }
\xydef@\splinefindtest@{%
\splinetrace@{SFT (\the\X@c,\the\Y@c); \the\splinelength@, \the\splineval@}%
 {\splineplot@false\splinetest@
  \ifsplineplot@\expandafter\splinefinishf@\fi}}
\def\setsplineundertest@#1#2#3#4#5{%
\splinetrace@{setsplineundertest@: #1,#2,#3,#4,#5,}%
 \DNii@{\gdef\splinetest@}%
 \DN@##1\next{\def\tmp@{{\splineplot@false
  \ifdim#1#2##1\relax #4\else\splineplot@true#5\fi }}}%
 \next@#3\relax\next
 \expandafter\nextii@\tmp@ }
\xydef@\splinef@end{\edef\tmp@{\egroup\splinelength@=\the\splinelength@}\tmp@}
\xydef@\splinef@pt{%
  \ifdim\splineval@>\z@
\splinetrace@{found: val=\the\splineval@;c:(\the\X@c,\the\Y@c)}%
   \global\dimen@i=\X@c \global\dimen3=\Y@c
  \else
\splinetrace@{found: val=\the\splineval@;p:(\the\X@p,\the\Y@p)}%
  \fi
  \setsplinedir@ \global\dimen5=\d@X \global\dimen7=\d@Y
  \global\splineval@=\splineval@
  \aftergroup\splinefocus@ }
\def\setsplinetest@#1#2#3#4#5{%
 \DNii@{\gdef\splinetest@}%
 \DN@##1\next{\def\tmp@{{\splineplot@false \ifdim#1#2##1\relax
  #4\else\splineplot@true#5\fi}}}%
 \next@#3\relax\next
 \expandafter\nextii@\tmp@ }%
```

```
\xydef@\splinefocus@{%
 \d@X=\dimen5\relax \d@Y=\dimen7\relax \setsplinedir@ }
```

```
\xydef@\splinesegment@#1{}
```

There are some extra hooks.

```
\xydef@\xyprecurve@{}
\xydef@\xypostcurve@{}
```

```
\xydef@\splineset@{%
\splinetrace@{splineset@:}%
 \ifx\bstartPLACE@\empty %\splinetrace@{invisible curve}%
  \DN@{\xyprecurve@
   \edef\tmp@{\egroup \X@max=\the\X@max \X@min=\the\X@min
    \Y@max=\the\Y@max \Y@min=\the\Y@min}\tmp@ \xypostcurve@ }%
 \else
  \DN@{\xyprecurve@ \xysplinespecialcases@
   \edef\tmp@{\egroup \X@max=\the\X@max \X@min=\the\X@min
    \Y@max=\the\Y@max \Y@min=\the\Y@min}\tmp@ \xypostcurve@ }%
 \fi
 \next@ }
```

For the actual setting, provide hooks which will allow alternative back-ends to be used in the special cases.

```
\xydef@\xysplinespecialcases@{%
 \ifx\empty\xycrvdrop@
  \ifx\empty\xycrvconn@ \DN@{\splinesolid@}%
  \else \DN@{ \dir{-}}\ifx\next@\xycrvconn@ \DN@{\splinesolid@}%
  \else \DN@{ \dir 2{-}}\ifx\next@\xycrvconn@ \DN@{\splinedoubled@}%
  \else \DN@{ \dir{=}}\ifx\next@\xycrvconn@ \DN@{\splineribboned@}%
  \else \DN@{ \dir {2.}}\ifx\next@\xycrvconn@ \DN@{\splinedoubled@}%
  \else \DN@{ \dir 3{-}}\ifx\next@\xycrvconn@ \DN@{\splinetrebled@}%
  \else \DN@{ \dir {3.}}\ifx\next@\xycrvconn@ \DN@{\splinetrebled@}%
  \else \DN@{ \dir{--}}\ifx\next@\xycrvconn@ \DN@{\splinedashed@}%
  \else \DN@{ \dir{.}}\ifx\next@\xycrvconn@ \DN@{\splinedotted@}%
  \else \DN@{ \dir{:}}\ifx\next@\xycrvconn@ \DN@{\splinedbldotted@}%
  \else \ifdim\splinetol@>\z@ \else \splinedefaulttol@ \fi
   \DN@{\splineset@@}\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
 \else
  \DN@{\splineset@@}%
 \fi \ifInvisible@\DN@{}\fi \next@ }
```

The special cases are handled just like the normal case except However the control sequence names provide a place for rebinding to accomodate alternative back-ends.

```
\xydef@\splinesolid@{\setbox\splinebox@=\zerodot \def\xycrvdrop@{ }%
 \def\xycrvconn@{}\splineset@@}
```

```
\xydef@\splinedoubled@{\splinedefaulttol@\def\xycrvdrop@{ }\def\splinedrop@{}%
 \def\xycrvconn@{!C\dir2{.}}\splineset@@}
```

```
\xydef@\splineribboned@{\splinedefaulttol@\def\xycrvdrop@{ }\def\splinedrop@{}%
 \def\xycrvconn@{!C\dir{:}}\splineset@@}
```

```
\xydef@\splinetrebled@{\splinedefaulttol@\def\xycrvdrop@{ }\def\splinedrop@{}%
 \def\xycrvconn@{!C\dir3{.}}\splineset@@}

\xydef@\splinedashed@{\splinetol@=1.5pc \def\xycrvdrop@{ }%+=<10pt>[o]{}}%
 \def\splinedrop@{}\edef\xycrvconn@{\noexpand!C\noexpand\dir{\addDASH@{}}}%
 \splineset@@}

\xydef@\splinedotted@{\setbox\splinebox@=\zerodot \def\xycrvdrop@{ }%
 \splinetol@=1.5\jot \def\xycrvconn@{}\splineset@@}

\xydef@\splinedbldotted@{\def\xycrvdrop@{ }\def\splinedrop@{}%
 \splinetol@=1.5\jot \def\xycrvconn@{!C\dir{:}}\splineset@@}
```

This establishes the test appropriate to actually setting the spline curve. Global definitions are used. This may not always be necessary!!

Bug: the `7.5pt` below should be the `\Step@@` method to be included.

```
\xydef@\splineset@@{%
 \readsplineparams@
 \ifx\xycrvdrop@\empty \splinetol@=7.5\p@
  \def\splinedefaulttol@{\splinetol@=7.5\p@}\def\splinedrop@{}%
 \else\edef\splinedefaulttol@{\splinetol@=\the\splinetol@}\fi
 \ifx\xycrvconn@\empty \def\splineconn@{}\fi
\splinetrace@{set the curve: >\the\dimen5, <\the\dimen7}%
 \global\let\splineadvance@=\splineadvance@@
 \let\splinerec@=\splineTrec@
 \def\splineStarttest{\splinetest@}%
 \global\let\splinepoint=\relax %6
 \let\spline@start=\spline@start@
 \let\spline@end=\spline@end@ %5
 \ifdim\dimen7=\p@
  \gdef\splinecontinue{\splinesetting@
    \global\let\splinetest@=\splineplot@true}%
 \else
  \ifdim\dimen7=\tw@\p@
%%   \ifdim\bendPLACE@\p@=\z@ \emptyspline@ \else
    \ifdim\bendPLACE@\p@=\p@ %\emptyspline@
   \fi %%\fi
 \else
  \xdef\splinecontinue{\noexpand\setsplinetest@
   \splineval@>{\the\dimen7}{\noexpand\splinebreakcancel@}{}%
   \noexpand\splinesetting@ }%
 \fi\fi
 \ifdim\dimen7>\z@ \DN@{\splinesetting@\splinecontinue}%
  \ifdim\zz@{\dimen5}%
  \else
   \edef\next@{\noexpand\setsplinetest@
     \splineval@<{\the\dimen5}{}%
    {\noexpand\expandafter\noexpand\splinecontinue}%
     \noexpand\splinesetting@ }%
  \fi
 \else\DN@{\dimen5=\p@}\fi  % ifdim\dimen7=\z@
```

```
 \next@
%
 \ifdim\dimen5=\p@\DN@{}\else\DN@{\splined@@}\fi \next@
 }
```

```
\xydef@\emptyspline@{\xywarning@{empty curve subsegment}%
\splinetrace@{bstartPLACE@=\bstartPLACE@, bendPLACE@=\bendPLACE@,
  empty segment}%
   \gdef\splinecontinue{\splinesetting@
    \global\let\splinetest@=\splineplot@false
    \global\dimen5=\z@ }}
```

`\splineTrec@` is the initial value for `\splinerec@` when a spline is being set. It descends the tree of places on the spline (1) until the required parameter value is found (2a) or is sufficiently close (2b). At this point call `\splinesetting@` and `\splinecontinue` to setup, and proceed with, the actual type-setting.

```
\xydef@\splineTrec@{{%
 \advance\splineval@\splinedepth@                        % (1)
 \ifdim\dimen5=\splineval@                               % (2a)
  \aftergroup\splinecontinue  % (3)
 \else \dimen@=\dimen5\advance\dimen@-\splineval@
  \ifdim\dimen@<\z@ \dimen@=-\dimen@ \fi
  \ifdim\dimen@<.001\p@  % possibly too high ?           % (2b)
   \aftergroup\splinecontinue % (3)
  \else
   \ifdim\dimen5<\splineval@\aftergroup\splinedecast@@    % (1)
   \else \aftergroup\splineadvance@                      % (1)
 \fi\fi\fi}}
```

Setting the curve uses the current `\splinerec@@` and sets `\splinepoint` to `\splineplotpt@` to cause objects to be typeset. This must be done via a `\gdef` rather than a `\global\let` since it is sometimes necessary to omit the object when it would be too close, see `\splinetooclose@`.

```
%\xydef@\splinesetting@{\xyFN@\splinesetting@@}
\xydef@\splinesetting@{\splinesetting@@}
```

```
\xydef@\splinesetting@@{%
\splinetrace@{splinesetting@@:}%
 \ifdim\splinedepth@=\z@ \splinedepth@=\p@ \fi
 \ifdim\splinedepth@=\p@
  \def\splinerec@{\global\let\splinerec@=\splinerec@@\splinedecast@@}%
 \else \global\let\splinerec@=\splinerec@@ \fi
 \gdef\splinepoint{\splineplotpt@}}
```

This handles, for each point of the curve, whether to place an ⟨object⟩ or horizontal glue.

```
\xydef@\splineplotpt@{{\global\advance\dimen@i-\X@c
 \splinetest@ \DN@{\splineglue@}\ifsplineplot@
%  \splineStarttest@ \ifsplineplot@
%   \splineEndtest@ \ifsplineplot@
   \DN@{\dosplineplotpt@}%
%  \fi\fi
 \fi \next@ \global\dimen@i=\X@c \global\dimen3=\Y@c }}
```

```
\xydef@\splineglue@{\hglue-\dimen@i
\splinetrace@{N:(\the\X@c,\the\Y@c);;\the\dimen@i;\the\dimen3%
   ,\the\splinelength@,\the\splineval@,\the\splinedepth@}}

\xydef@\dosplineplotpt@{\ifdim\dimen@i<\z@
   \raise\Y@c\hbox to-\dimen@i{\hfill\splinedrop@}%
  \else \kern-\dimen@i\raise\Y@c\hbox{\splinedrop@}\fi
   \ifx\splineconn@\relax\else\raise\Y@c\hbox{%
    \setboxz@h{\splineconn@}\Drop@@}\fi
\splinetrace@{P:(\the\X@c,\the\Y@c);;\the\dimen@i;\the\dimen3
   ,\the\splinelength@,\the\splineval@,\the\splinedepth@}}
```

These handle the “cleaning up” after a point on the curve has been located.

```
\xydef@\splinefinish@@{\postspline@ \splinegoal@ \splinecancel@ }

\xydef@\splinecancel@{%
 \global\let\splinerec@=\relax
 \global\let\splinepoint=\relax
 \global\let\splinecontinue=\relax
% \global\let\splinedecast@@=\relax
 \global\let\splinedecast@=\relax
 }

\xydef@\splinebreakcancel@{\bsplinecancel@ \splinecancel@}
\xydef@\bsplinecancel@{\gdef\lastbspline@{\leave@\leave@}%
 \global\let\middlebspline@=\lastbspline@ }

\xydef@\splinegoal@{\ifdim\splineval@>\z@
  \global\dimen@i=\X@c \global\dimen3=\Y@c \fi
 \global\splineval@=\splineval@ \global\splinelength@=\splinelength@
% \setsplineTangent@ }
 \splinesetdir@ }

\xylet@\splinefinish@\splinefinish@@
\xylet@\splinewhich\splineDwhich@

\xydef@\squinewhich@{%
 \ifdim\dimen5>\squinelength % \X@c=\L@c \Y@c=\D@c
 \d@X=\X@c \d@Y=\Y@c \advance\d@X-\L@c \advance\d@Y-\U@c
 \advance\squineval-\squinedepth
 \else \X@c=\X@p \Y@c=\Y@p \advance\squineval-\squinedepth
 \d@X=\L@c \d@Y=\U@c \advance\d@X-\X@p \advance\d@Y-\Y@p \fi
 \d@X=-\d@X \d@Y=-\d@Y %%% hack which works.
 \squinefinish}
\xydef@\squineDwhich@{%
 \ifdim\dimen5>\squinelength % \X@c=\L@c \Y@c=\D@c
 \d@X=\X@c \d@Y=\Y@c \advance\d@X-\L@c \advance\d@Y-\U@c
 \else \X@c=\X@p \Y@c=\Y@p \advance\squineval-\squinedepth
 \d@X=\L@c \d@Y=\U@c \advance\d@X-\X@p \advance\d@Y-\Y@p \fi
 \d@X=-\d@X \d@Y=-\d@Y %%% hack which works.
 \squinefinish}
```

Bézier quadratic splines — squines

```
\xydef@\xyquadbezier@{%
 \global\let\splinerec@=\relax
  \addtocrvpts@{\def\crvconnect@{\squineconnect@}}%
  \xyquadmethods@ \let\splinesegment@=\squinesegment@
  \begingroup\setcrvobjects@ \xysplineparams@ \splined@@@
  \bgroup \setupsquine@ \squined@ \splineset@ \savespline@ }

\xydef@\xyquadmethods@{%
  \def\crvconnect@{\squineconnect@}%
  \def\splinereverse@@{\splinereverse@@@}%
  \let\getsplineparams@=\getsquineparams@
  \let\spline@start@=\squine@start@
  \let\splinedecast@@=\squinedecast@
  \let\splinerec@@=\squinerec@
  \let\splined@@=\squined@@
  \let\splineIpt@@=\squineIpt@
  \let\splineIadvance@@=\squineIadvance@
  \let\splineinfo@=\squineinfo@
  \let\setsplinedir@=\setsquinedir@
  \let\splinetestcvxhull@=\squinetestcvxhull@
  \let\setsplineknotdir@=\setsquineknotdir@
  \let\splinereverse@=\squinereverse@
  \let\splineDadvance@@@=\squineDadvance@@@
  \let\splinesetparams@=\squinesetparams@
  \let\splineplot@maybe@@=\squineplot@maybe
}

\xydef@\squineconnect@{\splineconnect@ \crvconnect@@
  \let\splinesegment@=\squinesegment@ }%

\xydef@\squinesegment@#1{\xyquadmethods@ \def\segmentnum@{1}%
  \setupsquine@ \squined@ }
```

Use the registers \A@ and \B@ to store the coordinates of the single control point. The whole curve lies within the convex polygon with vertices at p, $(p+a)/2$, $(c+a)/2$ and c where a denotes the control point. Set \X@max, \X@min, \Y@max and \Y@min to be the extremes of the coordinates of these 4 points. There may not actually be any point on the curve achieving these extremes, but certainly we get pretty close.

```
\xydef@\setupsquine@{%
 \xycontrolpt@\z@ \X@p=\X@c \Y@p=\Y@c
 \xycontrolpt@\@ne \A@=\X@c \B@=\Y@c
 \xycontrolpt@\tw@
 \ifdim \A@>\X@max
  \dimen@=\X@p\advance\dimen@\A@ \dimen@half
   \ifdim \dimen@>\X@max \X@max=\dimen@ \fi
  \dimen@=\X@c\advance\dimen@\A@ \dimen@half
   \ifdim \dimen@>\X@max \X@max=\dimen@ \fi
 \else \ifdim \A@<\X@min
  \dimen@=\X@p\advance\dimen@\A@ \dimen@half
```

```
   \ifdim \dimen@<\X@min \X@min=\dimen@ \fi
  \dimen@=\X@c\advance\dimen@\A@ \dimen@half
   \ifdim \dimen@>\X@min \X@min=\dimen@ \fi
 \fi\fi
 \ifdim \B@>\Y@max
  \dimen@=\Y@p\advance\dimen@\B@ \dimen@half
   \ifdim \dimen@>\Y@max \Y@max=\dimen@ \fi
  \dimen@=\Y@c\advance\dimen@\B@ \dimen@half
   \ifdim \dimen@>\Y@max \Y@max=\dimen@ \fi
 \else \ifdim \B@<\Y@min
  \dimen@=\Y@p\advance\dimen@\B@ \dimen@half
   \ifdim \dimen@<\Y@min \Y@min=\dimen@ \fi
  \dimen@=\Y@c\advance\dimen@\B@ \dimen@half
   \ifdim \dimen@>\Y@min \Y@min=\dimen@ \fi
 \fi\fi }
```

The algorithm used for computing coordinates of points on quadratic Bézier splines is essentially that used by D. E. Knuth in `picmac.tex`[?] (aka `gpxmac.tex`). It is a recursive de Casteljau algorithm of the "divide and conquer" type. (In Knuth's macros these types of curves were given the name "squines". This explains some of the control sequence names used here.)

The differences from Knuth's algorithm are simply to allow more of the available information to be used at points along the spline. In particular the tangent direction can be calculated and tests can be performed to decide when to break out of the algorithm, rather than letting it run its full course.

```
\xydef@\squinedecast@{\halve@dimen\splinedepth@ %\divide\splinedepth@\tw@
 \R@c=\L@c \advance\L@c\X@p \halve@dimen\L@c    %\divide\L@c\tw@
 \advance\R@c\X@c \halve@dimen\R@c              %\divide\R@c\tw@
 \A@=\L@c \advance\A@\R@c \halve@dimen\A@       %\divide\A@\tw@
 \D@c=\U@c \advance\U@c\Y@p \halve@dimen\U@c    %\divide\U@c\tw@
 \advance\D@c\Y@c \halve@dimen\D@c              %\divide\D@c\tw@
 \B@=\U@c \advance\B@\D@c \halve@dimen\B@       %\divide\B@\tw@
 \begingroup \X@c=\A@ \Y@c=\B@ \splinerec@ \endgroup
 \begingroup \X@c=\A@ \Y@c=\B@ \splinepoint \endgroup
 \X@p=\A@ \L@c=\R@c \Y@p=\B@ \U@c=\D@c \splinerec@}
```

```
\xydef@\squinerec@{{\splinefar@false \splinenear@\X@p\X@c\Y@p\Y@c
 \ifsplinefar@ \aftergroup\squinedecast@
 \else \aftergroup\splineadvance@ \fi}}
```

```
\xydef@\squineDadvance@@@{\X@p\L@c\Y@p\D@c}
```

```
\xydef@\squineplot@maybe{{\X@p=\L@c \Y@p=\U@c \cv@start \the\Edge@c\@ne
  \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}
```

The tangent direction is computed from the displacement to the "recursive" control point.

```
\xydef@\setsquinedir@{%
 \d@X=\X@c \advance\d@X-\L@c \d@Y=\Y@c \advance\d@Y-\U@c
 \dimen@=\ifdim\d@X<\z@-\fi\d@X \ifdim\dimen@<.02\p@
  \dimen@=\ifdim\d@Y<\z@-\fi\d@Y \ifdim\dimen@<.02\p@
   \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p \fi\fi
 \ifdim\X@c=\X@p \ifdim\Y@c=\Y@p
  \d@X=\L@c \advance\d@X-\X@p \d@Y=\U@c \advance\d@Y-\Y@p \fi\fi
 \setupDirection@ii }
```

```
\xydef@\setsquineknotdir@{%
 \ifdim\zz@\splineval@ \expandafter \setsquinezerodir@
 \else\expandafter \setsquinedir@ \fi }
```

```
\xydef@\setsquinezerodir@{%
 \d@X=\L@c \advance\d@X-\X@p \d@Y=\U@c \advance\d@Y-\Y@p
 \ifdim\zz@\d@X \ifdim\zz@\d@Y
   \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
 \fi\fi \setupDirection@ii }
```

We need a way to access the information in `\splineparams`.

```
\xydef@\getsquineparams@#1,#2,#3,#4,#5,#6,#7,{%
 \splinelength@=#1\relax\X@p=#2\relax\Y@p=#3\relax
 \A@=#4\relax\B@=#5\relax\X@c=#6\relax\Y@c=#7\relax
}
\xydef@\squineinfo@{%
\expandafter\removePT@\the\X@c\space
\expandafter\removePT@\the\Y@c\space
\expandafter\removePT@\the\A@\space
\expandafter\removePT@\the\B@\space
\expandafter\removePT@\the\X@p\space
\expandafter\removePT@\the\Y@p\space
\expandafter\removePT@\the\dimen5\space\space
\expandafter\removePT@\the\dimen7\space
 }%
\xydef@\squinereverse@{\readsplineparams@
 \expandafter\edef\xysplineparams@{\the\splinelength@,%
  \the\X@c,\the\Y@c,\the\A@,\the\B@,\the\X@p,\the\Y@p,}}%
```

Start the picture by (1) setting a box, as usual. (2) Initialize global variables; these must be global since they are used to store information which must be preserved outside the grouping which is vital to the recursive nature of the algorithm. (3) Move horizontally to the starting point at p. (4) Begin by handling the starting point; *i.e.* decide whether it is to be plotted or not.

```
\xydef@\squine@start@{\setboxz@h\bgroup %   %(1)
 \global\splinelength@=\z@ \global\dimen@i=\z@ \global\dimen3=\z@ %  %(2)
 \kern\X@p  %  %(3)
 {\squinesetparams@ \X@c=\X@p \Y@c=\Y@p \firstsplinepoint@   %  %(4)
 }\bgroup }
```

```
\xydef@\squinesetparams@{%
 \global\dimen@i=\X@p \global\dimen3=\Y@p
 \L@c=\A@ \U@c=\B@ \R@c=\L@c \D@c=\U@c }%
```

```
\xydef@\squined@{%
 \expandafter\edef\xysplineparams@{%
  \the\z@,\the\X@p,\the\Y@p,\the\A@,\the\B@,\the\X@c,\the\Y@c,}}
```

```
\xydef@\squined@@{%
% \global\splinedepth@=\p@
 \splinedepth@=\p@
 \global\splineval@=\z@
 \global\dimen@i=\X@p \global\dimen3=\Y@p
 \spline@start
```

```
 \ifx\splinerec@\relax \let\splinerec@=\squinerec@\fi
 \L@c=\A@ \U@c=\B@ \R@c=\L@c \D@c=\U@c
 \splinerec@ \spline@end }
```

Bézier cubic splines

```
\xydef@\xycubicbezier@{%
 \addtocrvpts@{\def\crvconnect@{\cubicconnect@}}%
 \xycubicmethods@ \let\splinesegment@=\cubicsegment@
 \begingroup \setcrvobjects@ \xysplineparams@ \splined@@@
 \bgroup \setupcubic@ \cubiced@ \splineset@ \savespline@ }
```

```
\xydef@\xycubicmethods@{%
  \def\crvconnect@{\cubicconnect@}%
  \def\splinereverse@@{\cubicreverse@@}%
  \let\getsplineparams@=\getcubicparams@
  \let\spline@start@=\cubic@start@
  \let\splinedecast@@=\cubicdecast@
  \let\splinerec@@=\cubicrec@
  \let\splined@@=\cubiced@@
  \let\splineIpt@@=\cubicIpt@
  \let\splineIadvance@@=\cubicIadvance@
  \let\splineoutsidehull@@=\cubicoutsidehull@
  \let\splinetestcvxhull@=\cubictestcvxhull@
  \let\splineinfo@=\cubicinfo@
  \let\setsplinedir@=\setcubicdir@
  \let\setsplineknotdir@=\setcubicknotdir@
  \let\splinereverse@=\cubicreverse@
  \let\splineDadvance@@@=\cubicDadvance@@@
  \let\splinesetparams@=\cubicsetparams@
  \let\splineplot@maybe@@=\cubicplot@maybe
 }
```

```
\xydef@\cubicconnect@{\splineconnect@ \crvconnect@@
  \let\splinesegment@=\cubicsegment@ }%
```

```
\xydef@\cubicsegment@#1{\xycubicmethods@ \def\segmentnum@{1}%
 \setupcubic@ \cubiced@ }
```

Use the registers `\A@`, `\B@`, `\dimen3` and `\dimen5` to store coordinates of the two control points, denoted l and r say. The whole curve lies within the convex polygon p, $(p + l)/2$, $(c + r)/2$ and c where the vertices are not necessarily in this order. Set `\X@max`, `\X@min`, `\Y@max` and `\Y@min` to be the extremes of the coordinates of these 4 points. There may not actually be any point on the curve achieving these extremes, but certainly we get pretty close.

```
\xydef@\setupcubic@{%
 \xycontrolpt@\z@ \X@p=\X@c \Y@p=\Y@c
 \xycontrolpt@\@ne \A@=\X@c \B@=\Y@c
 \xycontrolpt@\tw@ \global\dimen3=\X@c \global\dimen5=\Y@c
 \xycontrolpt@\thr@@
 \adjustmaxmin@ }
```

```
\xydef@\adjustmaxmin@{%
```

```
 \ifdim \A@>\X@max \dimen@=\A@ \advance\dimen@\X@p \dimen@half
  \ifdim\dimen@>\X@max \X@max=\dimen@ \fi \else
 \ifdim \A@<\X@min \dimen@=\A@ \advance\dimen@\X@p \dimen@half
  \ifdim\dimen@<\X@min \X@min=\dimen@ \fi \fi\fi
 \ifdim \B@>\Y@max \dimen@=\B@ \advance\dimen@\Y@p \dimen@half
  \ifdim\dimen@>\Y@max \Y@max=\dimen@ \fi \else
 \ifdim \B@<\Y@min \dimen@=\B@ \advance\dimen@\Y@p \dimen@half
  \ifdim\dimen@<\Y@min \Y@min=\dimen@ \fi \fi\fi
%
 \ifdim \dimen3>\X@max \dimen@=\dimen3\advance\dimen@\X@c \dimen@half
  \ifdim\dimen@>\X@max \X@max=\dimen@ \fi \else
 \ifdim \dimen3<\X@min \dimen@=\dimen3\advance\dimen@\X@c \dimen@half
  \ifdim\dimen@<\X@min \X@min=\dimen@ \fi \fi\fi
 \ifdim \dimen5>\Y@max \dimen@=\dimen5\advance\dimen@\Y@c \dimen@half
  \ifdim\dimen@>\Y@max \Y@max=\dimen@ \fi \else
 \ifdim \dimen5<\Y@min \dimen@=\dimen5\advance\dimen@\Y@c \dimen@half
  \ifdim\dimen@<\Y@min \Y@min=\dimen@ \fi \fi\fi
%
 \dimen@=\A@ \advance\dimen@\dimen3 \dimen@half
 \ifdim\dimen@>\X@max \X@max=\dimen@
  \else \ifdim\dimen@<\X@min \X@min=\dimen@ \fi\fi
 \dimen@=\B@ \advance\dimen@\dimen5 \dimen@half
 \ifdim\dimen@>\Y@max \Y@max=\dimen@
  \else \ifdim\dimen@<\Y@min \Y@min=\dimen@ \fi\fi
}
```

The recursive algorithm for cubic Bézier splines is similar to the quadratic one. Now there are two "recursive" control points to be calculated upon each subdivision.

On the p-side:

$$\begin{aligned} p_p &= p \\ l_p &= (p+l)/2 \\ r_p &= (p+2l+r)/4 \\ c_p &= (p+3l+3r+c)/8 \end{aligned}$$

while on the c-side

$$\begin{aligned} p_c &= (p+3l+3r+c)/8 \\ l_c &= (l+2r+c)/4 \\ r_c &= (r+c)/2 \\ c_c &= c \end{aligned}$$

Notice that $c_p = p_c$ and that the tangents match there.

```
\xydef@\cubicdecast@{\halve@dimen\splinedepth@ %\divide\splinedepth@\tw@
 \A@=\L@c \advance\A@\R@c \halve@dimen\A@      %\divide\A@\tw@
 \advance\R@c\X@c \halve@dimen\R@c             %\divide\R@c\tw@
 \B@=\U@c \advance\B@\D@c \halve@dimen\B@      %\divide\B@\tw@
```

```
 \advance\D@c\Y@c \halve@dimen\D@c             %\divide\D@c\tw@
 \advance\L@c\X@p \halve@dimen\L@c             %\divide\L@c\tw@
 \advance\U@c\Y@p \halve@dimen\U@c             %\divide\U@c\tw@
 \begingroup
  \X@c=\R@c \advance\X@c\L@c \halve@dimen\X@c  %\divide\X@c\tw@
  \advance\X@c\A@ \halve@dimen\X@c             %\divide\X@c\tw@
  \Y@c=\D@c \advance\Y@c\U@c \halve@dimen\Y@c  %\divide\Y@c\tw@
  \advance\Y@c\B@ \halve@dimen\Y@c             %\divide\Y@c\tw@
  \R@c=\L@c \advance\R@c\A@ \halve@dimen\R@c   %\divide\R@c\tw@
  \D@c=\U@c \advance\D@c\B@ \halve@dimen\D@c   %\divide\D@c\tw@
  \bgroup \splinerec@
  \egroup \splinepoint \endgroup
 \X@p=\L@c \advance\X@p\R@c \halve@dimen\X@p   %\divide\X@p\tw@
 \advance\X@p\A@ \halve@dimen\X@p              %\divide\X@p\tw@
 \L@c=\R@c \advance\L@c\A@ \halve@dimen\L@c    %\divide\L@c\tw@
 \Y@p=\U@c \advance\Y@p\D@c \halve@dimen\Y@p   %\divide\Y@p\tw@
 \advance\Y@p\B@ \halve@dimen\Y@p              %\divide\Y@p\tw@
 \U@c=\D@c \advance\U@c\B@ \halve@dimen\U@c    %\divide\U@c\tw@
 \splinerec@ }
```

```
\xydef@\cubicrec@{{\splinefar@false \splinenear@\X@p\X@c\Y@p\Y@c
 \ifsplinefar@ \aftergroup\cubicdecast@
 \else \aftergroup\splineadvance@ \fi}}
```

```
\xydef@\cubicDadvance@@@{\X@p\X@c\Y@p\Y@c}
```

```
\xydef@\cubicplot@maybe{{\X@p=\L@c \Y@p=\U@c \cv@start \the\Edge@c\@ne
  \ifInside@\aftergroup\cubicplot@maybeii\else\aftergroup\splineplot@true\fi}}
\xydef@\cubicplot@maybeii{{\X@p=\R@c \Y@p=\D@c \cv@start \the\Edge@c\@ne
  \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}
```

```
\xydef@\cubiced@@{%
% \global\splinedepth@=\p@
 \splinedepth@=\p@
 \global\splineval@=\z@
 \spline@start
 \global\dimen@i=\X@p \global\dimen3=\Y@p
 \splinerec@ \spline@end }
```

The tangent direction is computed from the displacement to the "recursive" control point.

```
\xydef@\setcubicdir@{%
 \ifdim\splinedepth@<.001\p@\DN@{\cubiccoarsedir@}%
  \else\DN@{\cubicfinedir@}\fi \next@ }
```

```
\xydef@\cubicfinedir@{%
 \d@X=\X@c \advance\d@X-\R@c \d@Y=\Y@c \advance\d@Y-\D@c
 \ifdim\zz@\d@X \ifdim\zz@\d@Y
  \d@X=\X@c \advance\d@X-\L@c \d@Y=\Y@c \advance\d@Y-\U@c
  \ifdim\zz@\d@X \ifdim\zz@\d@Y
   \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
 \fi\fi \fi\fi
% \ifdim\X@c=\X@p \ifdim\Y@c=\Y@p
%  \d@X=\L@c \d@Y=\U@c \advance\d@X-\X@p \advance\d@Y-\Y@p
```

```
%  \ifdim\d@X=\z@ \ifdim\d@Y=\z@
%   \d@X=\R@c \d@Y=\D@c \advance\d@X-\X@p \advance\d@Y-\Y@p
%  \fi\fi
% \fi\fi
 \setupDirection@ii
\splinetrace@{dir:(\the\d@X,\the\d@Y),\the\Direction; depth:\the\splinedepth@}%
}

\xydef@\cubiccoarsedir@{%
 \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p \setupDirection@ii
 \global\dimen5=\d@X \global\dimen7=\d@Y
\splinetrace@{dir:(\the\d@X,\the\d@Y),\the\Direction; depth:\the\splinedepth@}%
}

\xydef@\setcubicknotdir@{%
 \ifdim\zz@\splineval@ \DN@{\setcubiczerodir@}%
 \else \DN@{\cubicfinedir@}\fi \next@ }

\xydef@\setcubiczerodir@{%
 \d@X=\L@c \advance\d@X-\X@p \d@Y=\U@c \advance\d@Y-\Y@p
 \ifdim\zz@\d@X \ifdim\zz@\d@Y
   \d@X=\R@c \advance\d@X-\X@p \d@Y=\D@c \advance\d@Y-\Y@p
   \ifdim\zz@\d@X \ifdim\zz@\d@Y
     \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
 \fi\fi \fi\fi \setupDirection@ii }
```

We need a way to access the information in `\splineparams`.

```
\xydef@\getcubicparams@#1,#2,#3,#4,#5,#6,#7,#8,#9,{%
 \splinelength@=#1\relax\X@p=#2\relax\Y@p=#3\relax\L@c=#4\relax
 \U@c=#5\relax\R@c=#6\relax\D@c=#7\relax\X@c=#8\relax\Y@c=#9\relax
}%
\xydef@\cubicinfo@{%
\expandafter\removePT@\the\X@p\space
\expandafter\removePT@\the\Y@p\space
\expandafter\removePT@\the\L@c\space
\expandafter\removePT@\the\U@c\space
\expandafter\removePT@\the\R@c\space
\expandafter\removePT@\the\D@c\space
\expandafter\removePT@\the\X@c\space
\expandafter\removePT@\the\Y@c\space
\expandafter\removePT@\the\dimen5\space\space
\expandafter\removePT@\the\dimen7\space
 }%
\xydef@\cubicreverse@{\readsplineparams@
 \expandafter\edef\xysplineparams@{\the\splinelength@,%
  \the\X@c,\the\Y@c,\the\R@c,\the\D@c,\the\L@c,\the\U@c,\the\X@p,\the\Y@p,}}%
\xydef@\cubicreverse@@{%
 \splinereverse@@@
 \dimen@ii=\L@c \L@c=\R@c \R@c=\dimen@ii
 \dimen@ii=\U@c \U@c=\D@c \D@c=\dimen@ii }
```

Start the picture by (1) setting a box, as usual. (2) Initialize global variables; these must be global since they are used to store information which must be preserved outside the grouping which is vital to the recursive nature of the algorithm. (3) Move horizontally to the starting point at p. (4) Begin by handling the starting point; *i.e.* decide whether it is to be plotted or not.

```
\xydef@\cubic@start@{\setboxz@h\bgroup %   %(1)
 \global\splinelength@=\z@ \global\dimen@i=\z@ \global\dimen3=\z@ %  %(2)
 \kern\X@p  %  %(3)
 {\cubicsetparams@ \X@c=\X@p \Y@c=\Y@p \R@c=\X@c \D@c=\Y@c
  \firstsplinepoint@ }\bgroup
} %  %(4)
```

```
\xydef@\cubicsetparams@{%
 \expandafter\expandafter\expandafter\getsplineparams@\xysplineparams@
 \global\dimen@i=\X@p \global\dimen3=\Y@p }
```

```
\xydef@\cubiced@{%
 \expandafter\edef\xysplineparams@{\the\z@,\the\X@p,\the\Y@p
  ,\the\A@,\the\B@,\the\dimen3,\the\dimen5,\the\X@c,\the\Y@c,}%
 }%
```

B-splines

The cases of 3 and 4 control points have some special simplifying features.

```
\xydef@\xybspline@iii{%
  \addtocrvpts@{\def\crvconnect@{\bsplineiiiconnect@}}%
  \def\crvconnect@{\bsplineiiiconnect@}\xybsplinemethods@
  \let\splinesegment@=\bsplinesegment@iii
  \begingroup\setcrvobjects@ \dobspline@ \savespline@ }
\xydef@\bsplineiiiconnect@{\splineconnect@ \crvconnect@@
  \let\splinesegment@=\bsplinesegment@iii }%
```

```
\xydef@\xybspline@iv{%
  \addtocrvpts@{\def\crvconnect@{\bsplineivconnect@}}%
  \def\crvconnect@{\bsplineivconnect@}\xybsplinemethods@
  \let\splinesegment@=\bsplinesegment@iv
  \begingroup\setcrvobjects@ \dobspline@ \savespline@ }
\xydef@\bsplineivconnect@{\splineconnect@ \crvconnect@@
  \let\splinesegment@=\bsplinesegment@iv }%
```

Now for the general case of ≥ 5 control points.

```
\xydef@\xybspline@{%
\splinetrace@{B-spline with \numcontrolpts\space control points.}%
  \addtocrvpts@{\def\crvconnect@{\bsplineconnect@}}%
  \def\crvconnect@{\bsplineconnect@}\xybsplinemethods@
  \let\splinesegment@=\bsplinesegment@
  \begingroup\setcrvobjects@
 \dobspline@ \savespline@ }
```

```
\xydef@\xybsplinemethods@{%
  \def\splinereverse@@{\cubicreverse@@}%
  \let\getsplineparams@=\getbsplineparams@
  \let\spline@start@=\cubic@start@
```

```
  \let\splinedecast@@=\cubicdecast@
  \let\splinerec@@=\cubicrec@
%  \let\splined@@=\cubiced@@
  \let\splineIpt@@=\cubicIpt@
  \let\splineIadvance@@=\cubicIadvance@
  \let\splined@@=\bsplined@@
  \let\splineinfo@=\cubicinfo@
  \let\setsplinedir@=\setcubicdir@
  \let\setsplineknotdir@=\setcubicknotdir@
  \let\splinereverse@=\cubicreverse@
  \let\splineDadvance@@@=\cubicDadvance@@@
  \let\splinesetparams@=\cubicsetparams@
  \global\let\lastbspline@=\lastbspline@@
  \global\let\middlebspline@=\middlebspline@@
}
\xydef@\bsplined@@{%
% \global\splinedepth@=\p@
 \splinedepth@=\p@
 \global\splineval@=\z@
 \spline@start
 \global\dimen@i=\X@p \global\dimen3=\Y@p
 \splinerec@ \spline@end }
```

```
\xydef@\bsplineconnect@{\splineconnect@ \crvconnect@@
  \let\splinesegment@=\bsplinesegment@ }%
```

```
\xydef@\getbsplineparams@{\getcubicparams@}
```

```
\xydef@\dobspline@{\xysplineparams@ \scanbspline@ \firstbspline@ }%
```

Registers `\dimen3` and `\dimen5` are used globally, since the standard local registers are already used.

```
\xydef@\firstbspline@{%
  \enter@{\pfromthep@ \cfromthec@}\enter@{\cfromthec@}\bgroup
  \def\segmentnum@{1}\xycontrolpt@\z@ \X@p=\X@c \Y@p=\Y@c
\splinetrace@{0: \the\X@p, \the\Y@p}%
  \xycontrolpt@\@ne \A@=\X@c \B@=\Y@c
\splinetrace@{1: \the\X@c, \the\Y@c}%
  \xycontrolpt@\tw@
\splinetrace@{2: \the\X@c, \the\Y@c}%
   \dimen@=\X@c \advance\dimen@\A@ \dimen@half
   \global\dimen3=\dimen@
   \dimen@=\Y@c \advance\dimen@\B@ \dimen@half
   \global\dimen5=\dimen@
  \xycontrolpt@{3}%
\splinetrace@{3: \the\X@c, \the\Y@c}%
  \expandafter\count@\xycrvcnt@\relax %%%  very important to |\relax|
  \ifnum\count@=3\relax
   \advance\X@c-\A@ \dimen@nth4\X@c    % \divide\X@c by4
   \advance\X@c\dimen3\relax
   \advance\Y@c-\B@ \dimen@nth4\Y@c    % \divide\Y@c by4
```

```
   \advance\Y@c\dimen5\relax
   \enter@{\X@p=\the\X@c \Y@p=\the\Y@c \crv@cnt@=\@ne\relax
    \noexpand\lastbspline@ }%
  \else
   \advance\X@c 7\dimen3 \advance\X@c-2\A@
    \dimen@nth6\X@c  % \divide\X@c by6\relax
   \advance\Y@c 7\dimen5 \advance\Y@c-2\B@
    \dimen@nth6\Y@c  % \divide\Y@c by6\relax
   \enter@{\X@p=\the\X@c \Y@p=\the\Y@c \crv@cnt@=\@ne\relax
    \noexpand\middlebspline@ }%
  \fi \adjustmaxmin@ \czeroEdge@
 \bsplined@ \splineset@ \leave@ }

\xydef@\lastbspline@@{\bgroup
  \advance\crv@cnt@\@ne \edef\segmentnum@{\the\crv@cnt@}%
\splinetrace@{<: \the\X@p, \the\Y@p}%
  \advance\crv@cnt@\@ne \xycontrolpt@{\crv@cnt@}%
  \global\dimen3=\X@c \global\dimen5=\Y@c
\splinetrace@{\the\crv@cnt@: \the\X@c, \the\Y@c}%
   \advance\crv@cnt@\m@ne \xycontrolpt@{\crv@cnt@}%
\splinetrace@{\the\crv@cnt@: \the\X@c, \the\Y@c}%
    \A@=\X@c \advance\A@\dimen3 \halve@dimen\A@  % \divide\A@\tw@
    \B@=\Y@c \advance\B@\dimen5 \halve@dimen\B@  % \divide\B@\tw@
  \adjustmaxmin@ \leave@  \relax \cv@end % <---------------
\splinetrace@{>: \the\X@c, \the\Y@c}%
  \bsplined@ \splineset@ \leave@ }%
\xylet@\lastbspline@=\lastbspline@@

\xydef@\middlebspline@@{%
  \advance\crv@cnt@\@ne \edef\segmentnum@{\the\crv@cnt@}%
\splinetrace@{<: \the\X@p, \the\Y@p}%
  \bgroup
   \xycontrolpt@{\crv@cnt@}\A@=2\X@c \B@=2\Y@c
\splinetrace@{\the\crv@cnt@: \the\X@c, \the\Y@c}%
   \advance\crv@cnt@\@ne \xycontrolpt@{\crv@cnt@}%
\splinetrace@{\the\crv@cnt@: \the\X@c, \the\Y@c}%
    \advance\A@\X@c \dimen@nth3\A@   % \divide\A@ by3
    \advance\B@\Y@c \dimen@nth3\B@   % \divide\B@ by3\relax
    \advance\X@c\A@ \halve@dimen\X@c % \divide\X@c\tw@
    \advance\Y@c\B@ \halve@dimen\Y@c % \divide\Y@c\tw@
    \global\dimen3=\X@c \global\dimen5=\Y@c
   \advance\crv@cnt@\@ne \xycontrolpt@{\crv@cnt@}%
\splinetrace@{\the\crv@cnt@: \the\X@c, \the\Y@c}%
  \expandafter\count@\xycrvcnt@\relax
  \ifnum\crv@cnt@<\count@\relax
   \advance\X@c 7\dimen3\advance\X@c-2\A@
    \dimen@nth6\X@c  %  \divide\X@c by6\relax
   \advance\Y@c 7\dimen5\advance\Y@c-2\B@
    \dimen@nth6\Y@c  %  \divide\Y@c by6\relax
   \enter@{\X@p=\the\X@c \Y@p=\the\Y@c \crv@cnt@=\segmentnum@\relax
    \noexpand\middlebspline@}%
```

```
  \else
   \advance\X@c-\A@ \dimen@nth4\X@c  % \divide\X@c by4
   \advance\X@c \dimen3\relax
   \advance\Y@c-\B@ \dimen@nth4\Y@c  % \divide\Y@c by4
   \advance\Y@c \dimen5\relax
   \enter@{\X@p=\the\X@c \Y@p=\the\Y@c \noexpand\lastbspline@}%
  \fi \adjustmaxmin@
   \bsplined@ \splineset@ \leave@ }
\xylet@\middlebspline@=\middlebspline@@

\xydef@\scanbspline@{\splined@@@}

\xydef@\bsplined@{\cubiced@
 \expandafter\ifx\xysplineedges@\relax\relax\DN@{}%
 \else\DN@{\adjustbsplineedges@}\fi \next@ }

\xydef@\adjust@forsegments@{\getcrvsegsnum@
 \ifnum\count@>\@ne
\splinetrace@{adjust for \the\count@\space segments:
 \the\dimen5, \the\dimen7}%
  \multiply\dimen5 by \count@\relax \dimen@=\segmentnum@\p@
  \ifnum\dimen5>\dimen@ \relax\dimen5=\p@
  \else \advance\dimen5-\segmentnum@\p@ \advance\dimen5 by\p@ \fi
  \ifnum\dimen5<\z@ \relax\dimen5=\z@\relax\fi
  \multiply\dimen7 by \count@\relax
  \dimen@=\segmentnum@\p@ %\advance\dimen@-\segmentnum@\p@
  \ifnum\dimen7<\dimen@ \advance\dimen@-\p@
   \ifnum\dimen7>\dimen@ \advance\dimen7-\dimen@
   \else \dimen7=\z@ \fi
  \else \dimen7=\p@ \fi
\splinetrace@{adjusted for \the\count@\space segments:
 \the\dimen5, \the\dimen7}%
 \fi}

\xydef@\adjustbsplineedges@{\bgroup % dimen5 and dimen7 are NOT global here
\splinetrace@{** adjusting edges **}%
  \readsplineparams@ \adjust@forsegments@
%   \count@@=\dimen7\divide\count@@\p@
%    \advance\dimen7-\count@@\p@ \advance\count@@\@ne
%   \count@=\dimen5\divide\count@\p@
%    \advance\dimen5-\count@\p@ \advance\count@\@ne
\splinetrace@{params:\xysplineparams@}%
\splinetrace@{segment \segmentnum@:<\the\dimen5,\the\dimen7>:\xysplineedges@}%
%   \expandafter\ifnum\segmentnum@=\count@
%   \else
%    \expandafter\ifnum\segmentnum@<\count@\dimen5=\p@
%    \else\dimen5=\z@ \fi\fi
%   \expandafter\ifnum\segmentnum@=\count@@
    \multiply\dimen7by\m@ne \advance\dimen7by\p@
%   \else
%    \expandafter\ifnum\segmentnum@>\count@@\dimen5=\p@\dimen7=\p@
```

```
%    \else\dimen7=\z@ \fi\fi
  \expandafter\edef\xysplineedges@{%
   \noexpand\z@;\the\dimen5,\noexpand\z@;\the\dimen7,}%
  \edef\tmp@{\egroup
   \noexpand\def\expandafter\noexpand\xysplineedges@{\xysplineedges@}}%
  \tmp@ }
```

These macros select the correct Bézier control points for each segment of the spline. This is needed for finding places on the constructed curves.

In general there are 5 types of segment: first, second, middle, penultimate, final. The conversion from B-spline to Bézier is slightly different for each type. For middle segments the Bézier control points are determined in the following way:

$$
\begin{aligned}
X^{(1)}_{B_i} &= \frac{1}{3}\left(2X^{(i)} + X^{(i+1)}\right) \\
X^{(2)}_{B_i} &= \frac{1}{3}\left(X^{(i)} + 2X^{(i+1)}\right) \\
X^{(0)}_{B_i} &= \frac{1}{6}\left(X^{(i-1)} + 4X^{(i)} + X^{(i+1)}\right) \\
&= \frac{1}{6}\left(X^{(i-1)} + 7X^{(1)}_{B_i} - 2X^{(2)}_{B_i}\right) \\
X^{(3)}_{B_i} &= \frac{1}{6}\left(X^{(i)} + 4X^{(i+1)} + X^{(i+2)}\right) \\
&= \frac{1}{6}\left(X^{(i+2)} + 7X^{(2)}_{B_i} - 2X^{(1)}_{B_i}\right)
\end{aligned}
$$

```
\xydef@\bsegment@@ii@iii{%
 \xycontrolpt@{\count@@}\global\dimen3=\X@c \global\dimen5=\Y@c
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
 \advance\count@@\@ne \xycontrolpt@{\count@@}\A@=\X@c \B@=\Y@c
 \advance\A@ by2\dimen3 \dimen@nth3\A@  % \divide\A@ by3\relax
 \advance\B@ by2\dimen5 \dimen@nth3\B@  % \divide\B@ by3\relax
 \dimen@=\dimen3 \advance\dimen@ by2\X@c
  \dimen@nth3\dimen@  %  \divide\dimen@\thr@@
 \global\dimen3=\dimen@
 \dimen@=\dimen5 \advance\dimen@ by2\Y@c
  \dimen@nth3\dimen@  %  \divide\dimen@\thr@@
 \global\dimen5=\dimen@
\splinetrace@{<: \the\dimen3, \the\dimen5}%
\splinetrace@{>: \the\A@, \the\B@}%
 \advance\count@@\m@ne }
```

```
\xydef@\bsegment@@i{\bgroup
 \advance\count@@\m@ne \xycontrolpt@{\count@@}%
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
 \edef\tmp@{\egroup \X@p=\the\X@c \Y@p=\the\Y@c}\tmp@
 \advance\X@p-2\dimen3\advance\X@p by7\A@
   \dimen@nth6\X@p  %  \divide\X@p by6\relax
 \advance\Y@p-2\dimen5\advance\Y@p by7\B@
   \dimen@nth6\Y@p  %  \divide\Y@p by6\relax
}
```

```
\xydef@\bsegment@@iv{%
 \advance\count@@\tw@ \xycontrolpt@{\count@@}%
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
 \advance\X@c by7\dimen3\advance\X@c-2\A@
   \dimen@nth6\X@c  %  \divide\X@c by6\relax
 \advance\Y@c by7\dimen5\advance\Y@c-2\B@
   \dimen@nth6\Y@c  %  \divide\Y@c by6\relax
\splinetrace@{>>: \the\X@c, \the\Y@c }%
}
```

For the second segment the expression for $X^{(0)}_{B_2}$ is different. Similarly the expression for $X^{(3)}_{B_{n-1}}$ is altered for the penultimate segment.

$$
\begin{aligned}
X^{(0)}_{B_2} &= \frac{1}{4}\left(X^{(1)} + 4X^{(1)}_{B_2} - X^{(2)}_{B_2}\right) \\
X^{(3)}_{B_{n-1}} &= \frac{1}{4}\left(X^{(3)} + 4X^{(2)}_{B_{n-1}} - X^{(1)}_{B_{n-1}}\right)
\end{aligned}
$$

```
\xydef@\bsegment@@ii@i{\bgroup
 \advance\count@@\m@ne \xycontrolpt@{\count@@}%
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
 \edef\tmp@{\egroup \X@p=\the\X@c \Y@p=\the\Y@c}\tmp@
 \advance\X@p-\dimen3\advance\X@p by4\A@
   \dimen@nth4\X@p  %  \divide\X@p by4\relax
 \advance\Y@p-\dimen5\advance\Y@p by4\B@
   \dimen@nth4\Y@p  %  \divide\Y@p by4\relax
\splinetrace@{<<: \the\X@p, \the\Y@p}%
}
\xydef@\bsegment@@y@iv{%
 \advance\count@@\tw@ \xycontrolpt@{\count@@}%
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
 \advance\X@c by4\dimen3\advance\X@c-\A@
   \dimen@nth4\X@c  %  \divide\X@c by4\relax
 \advance\Y@c by4\dimen5\advance\Y@c-\B@
   \dimen@nth4\Y@c  %  \divide\Y@c by4\relax
}
```

For the first segment the first three Bézier control points are always determined in the same way:

$$
\begin{array}{lll}
X^{(0)}_{B_1} = X^{(0)} \quad X^{(1)}_{B_1} = X^{(1)} & & X^{(2)}_{B_1} = \frac{1}{2}(X^{(1)}_{B_1} + X^{(2)}) \\
\text{3+ segments} & & X^{(3)}_{B_1} = \frac{1}{6}(7X^{(2)}_{B_1} - 2X^{(1)}_{B_1} + X^{(3)}) \\
\text{2 segments} & & X^{(3)}_{B_1} = \frac{1}{4}(4X^{(2)}_{B_1} - X^{(1)}_{B_1} + X^{(3)})
\end{array}
$$

```
\xydef@\bsegment@i{%
 \xycontrolpt@\z@ \X@p=\the\X@c \Y@p=\the\Y@c
% \ifx\cv@start\relax
%  \bgroup\csname cv@0\endcsname
```

```
%  \edef\tmp@{\egroup \X@p=\the\X@c \Y@p=\the\Y@c}\tmp@
% \else\xycontrolpt@\z@ \X@p=\the\X@c \Y@p=\the\Y@c\fi
\splinetrace@{0: \the\X@p \the\Y@p}%
  \xycontrolpt@\@ne
\splinetrace@{1: \the\X@c, \the\Y@c}\A@=\X@c \B@=\Y@c
  \xycontrolpt@\tw@
  \dimen@=\X@c \advance\dimen@\A@ \dimen@half
  \global\dimen3=\dimen@
\splinetrace@{2: \the\X@c, \the\Y@c}%
  \dimen@=\Y@c \advance\dimen@\B@ \dimen@half
  \global\dimen5=\dimen@ }
```

```
\xydef@\bsegment@i@iv{%
  \xycontrolpt@{3}%
\splinetrace@{3: \the\X@c, \the\Y@c}%
   \advance\X@c by7\dimen3\advance\X@c-2\A@
    \dimen@nth6\X@c  %  \divide\X@c by6\relax
   \advance\Y@c by7\dimen5\advance\Y@c-2\B@
    \dimen@nth6\Y@c  %  \divide\Y@c by6\relax
 }
```

The last segment is determined symmetrically from the final four control points:

$$X^{(3)}_{B_n} = X^{(n+2)} \quad X^{(2)}_{B_n} = X^{(n+1)} \quad X^{(1)}_{B_n} = \frac{1}{2}(X^{(2)}_{B_n} + X^{(n)})$$

$$\text{3+ segments} \qquad X^{(0)}_{B_n} = \frac{1}{6}(X^{(n-1)} - 2X^{(2)}_{B_n} + 7X^{(1)}_{B_n})$$

$$\text{2 segments} \qquad X^{(0)}_{B_n} = \frac{1}{4}(X^{(1)} - X^{(2)}_{B_n} + 4X^{(1)}_{B_n})$$

```
\xydef@\bsegment@z{%
 \expandafter\count@@\xycrvptsnum@ \advance\count@@\@ne
 \xycontrolpt@{\the\count@@}\relax
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
 \advance\count@@\m@ne
 \bgroup
  \xycontrolpt@{\the\count@@}%
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
  \global\dimen3=\X@c \global\dimen5=\Y@c
  \advance\count@@\m@ne \xycontrolpt@{\the\count@@}%
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
  \A@=\X@c \advance\A@\dimen3 \halve@dimen\A@   %  \divide\A@\tw@
  \B@=\Y@c \advance\B@\dimen5 \halve@dimen\B@   %  \divide\B@\tw@
  \edef\tmp@{\egroup \A@=\the\A@ \B@=\the\B@ \count@@=\the\count@@}%
 \tmp@ }
```

```
\xydef@\bsegment@z@i{%
 \expandafter\count@@\xycrvptsnum@\relax\advance\count@@-\tw@
 \bgroup
  \xycontrolpt@{\count@@}\X@p=\the\X@c \Y@p=\the\Y@c
\splinetrace@{\the\count@@: \the\X@c, \the\Y@c}%
  \advance\X@p by7\A@ \advance\X@p-2\dimen3 \dimen@nth6\X@p
```

```
  \advance\Y@p by7\B@ \advance\Y@p-2\dimen5 \dimen@nth6\Y@p
  \edef\tmp@{\egroup \X@p=\the\X@p \Y@p=\the\Y@p}%
 \tmp@ }
```

This is the switching-yard.

```
\xydef@\bsplinesegment@#1{\xybsplinemethods@
 \count@@=#1\relax
 \ifnum\count@@=\@ne       %  first segment
  \bsegment@i \bsegment@i@iv
 \else\ifnum\count@@=\tw@  %  second segment
  \bsegment@@ii@iii \bsegment@@ii@i \bsegment@@iv
% \else\expandafter\count@\xycrvcnt@\advance\count@\m@ne
 \else\expandafter\count@\xycrvptsnum@\advance\count@\m@ne
 \ifnum\count@@=\count@    %  last segment
  \bsegment@z \bsegment@z@i
 \else\advance\count@\m@ne
  \ifnum\count@@=\count@   %  penult. segment
   \bsegment@@ii@iii \bsegment@@i \bsegment@@y@iv
  \else                    %  middle segments
   \bsegment@@ii@iii \bsegment@@i \bsegment@@iv
 \fi\fi\fi\fi
 \expandafter\def\expandafter\segmentnum@\expandafter{\number#1}%
 \bsplined@ }
```

Here are the differences for 3 control points.

```
\xydef@\bsplinesegment@iii#1{\xybsplinemethods@
\splinetrace@{2 segments; \number#1}%
 \count@@=#1\relax
 \ifnum\count@@=\@ne \bsegment@i
   \xycontrolpt@{3}%
\splinetrace@{3: \the\X@c, \the\Y@c}%
   \advance\X@c 4\dimen3\advance\X@c-\A@
    \dimen@nth4\X@c   %  \divide\X@c by4\relax
   \advance\Y@c 4\dimen5\advance\Y@c-\B@
    \dimen@nth4\Y@c   %  \divide\Y@c by4\relax
 \else\ifnum\count@@=\tw@ \bsegment@z
   \bgroup\xycontrolpt@{\@ne}%
\splinetrace@{1: \the\X@c, \the\Y@c}%
   \edef\tmp@{\egroup \X@p=\the\X@c \Y@p=\the\Y@c}\tmp@
   \advance\X@p 4\A@ \advance\X@p-\dimen3 %
    \dimen@nth4\X@p   %  \divide\X@p by4\relax
   \advance\Y@p 4\B@ \advance\Y@p-\dimen5 %
    \dimen@nth4\Y@p   %  \divide\Y@p by4\relax
 \fi\fi
 \expandafter\def\expandafter\segmentnum@\expandafter{\number#1}%
 \bsplined@ }
```

Here are the differences for 4 control points.

```
\xydef@\bsplinesegment@iv#1{\xybsplinemethods@
\splinetrace@{3 segments; \number#1}%
```

```
\count@@=#1\relax
\ifnum\count@@=\@ne \bsegment@i \bsegment@i@iv
\else\ifnum\count@@=\tw@
 \bsegment@@ii@iii \bsegment@@ii@i \bsegment@@y@iv
\else\ifnum\count@@=3\relax
 \bsegment@z \bsegment@z@i %\xycontrolpt@{5}%
\fi\fi\fi
\expandafter\def\expandafter\segmentnum@\expandafter{\number#1}%
\bsplined@ }
```

2.1.2 Circles and Ellipses

Here we describe the means to a specify circles of arbitrary radius, drawn with arbitrary line styles. When large-sized objects are used they are regularly spaced around the circle. Similarly ellipses may be specified, but only those having major/minor axes aligned in the standard directions; spacing of objects is no longer regular, but is bunched toward the narrower ends.

Such a circle or ellipse is specified using…

`\xycircle`⟨vector⟩`{`⟨style⟩`}`

---

where the components of the ⟨vector⟩ determine the lengths of the axis for the ellipse; thus giving a circle when equal. The ⟨style⟩ can be any ⟨conn⟩, as in 3.2 that works with curved arrows—many do. Alternatively ⟨style⟩ can be any ⟨object⟩, which will be placed equally-spaced about the circle at a separation to snugly fit the ⟨object⟩s. If ⟨empty⟩ then a solid circle or ellipse is drawn.

To Do: Recognize special ⟨style⟩s; e.g. `.` for dotted with `\zerodot`, `=` for a doubled circle — alter the radius, draw two circles; `:` for double dotted (perhaps use `\doubled@`, `\tripled@`); `>` and `<` for chevrons; any others?

To use any of these special symbols as the ⟨object⟩ for ⟨style⟩ then enclose it within extra braces, *e.g.* `\xycircle<20pt>{{>}}`.

The curves are not truly circular or elliptical, but are approximations given by cubic Bézier segments. Hence the `xycurve` feature must be loaded.

```
\message{circles,}
```

The circles are constructed from four Bézier cubic curves, one for each quadrant of the circle or circle. To do this it is sufficient to establish the control points for each cubic segment. This is straightforward, using the following "magic number", given a square basis such that the desired circle is the unit circle, or rectangular basis for which the "unit circle" is the desired circle.

```
\xydef@\Circmagic@@{0.5517847}
\xylet@\Circmagic=\Circmagic@@
\xydef@\twoPi@{6.2831852}
\xydef@\fullPi@{3.1415926}
\xydef@\halfPi@{1.5707963}
```

This is the magic number, exactly given by $\frac{1}{12}(\sqrt{385}-13)$, that helps construct the Bézier cubic curve that best approximates a quadrant of a circle. It does so with remarkable accuracy, differing by at most .5% of the radius at any angle; the average deviation along the whole quadrant being less than .13%. The basic ⟨object⟩ defined here is `\circle`.

```
\xynew@{dimen}\L@
```

```
\xydef@\xycircle#1#{\hbox\bgroup\afterVECTORorEMPTY{%
 \xy@@{\R@=\X@c \L@=\Y@c}\xycircle@}{\xy@@{\R@=\R@c \L@=\L@c}\xycircle@}#1@}
```

```
\xydef@\xycircle@#1@#2{%
 \DN@{#1}\ifx\next@\empty\def\onlyQuad@{}\else \count@=#1\relax
  \ifnum\count@<5 \advance\count@-3\relax
   \ifnum\count@<\z@ \advance\count@ 4\relax\fi \edef\onlyQuad@{\the\count@}%
  \else\xyerror@{illegal circle <radius>: must be <vector> or <empty>}{}\fi
 \fi \xy@@{\def\circleSTYLE@{#2}}\def\circleSTYLE@{#2}\xycircle@i}

\xydef@\xycircle@i{\hbox{\vbox{\vskip\L@
 \hbox to2\R@{\hfill \buildcircle@ \hfill}\vskip\L@ }}%
 \L@c=\R@ \R@c=\R@ \D@c=\L@ \U@c=\L@ \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@\relax}\Edge@c={\circleEdge}%
 \OBJECT@x}

\xydef@\adjustMinMaxcirc@#1#2{%
 \dimen@=\X@c \advance\dimen@#1\relax\ifdim\dimen@>\X@max \X@max=\dimen@\fi
 \dimen@=\X@c \advance\dimen@-#1\relax\ifdim\dimen@<\X@min \X@min=\dimen@\fi
 \dimen@=\Y@c \advance\dimen@#2\relax\ifdim\dimen@>\Y@max \Y@max=\dimen@\fi
 \dimen@=\Y@c \advance\dimen@-#2\relax\ifdim\dimen@<\Y@min \Y@min=\dimen@\fi
}

%\xydef@\buildcircle@{\save@
%% \cubicCircleControls@@
% \ifx\circleSTYLE@\empty\DN@{\solidcircle@{}}%
% \else \expandafter\DN@\expandafter{\addDASH@{}}\relax
%  \ifx\next@\circleSTYLE@ \DN@{%
%   \expandafter\addDASH@\expandafter{\expandafter{\addDASH@{}}}}%
% \else\DN@{\circledobjects@}\fi\fi\next@ \leave@}

\xydef@\cubicCircleControls@@{\X@origin=\z@ \Y@origin=\z@
  \X@xbase=\R@ \Y@xbase=\z@ \X@ybase=\z@ \Y@ybase=\L@
  \vfromcartesian@@1,0@\czeroEdge@\idfromc@{0@c}%
  \vfromcartesian@@0,1@\czeroEdge@\idfromc@{1@c}%
  \vfromcartesian@@-1,0@\czeroEdge@\idfromc@{2@c}%
  \vfromcartesian@@0,-1@\czeroEdge@\idfromc@{3@c}%
  \vfromcartesian@@1,\Circmagic@@ @\czeroEdge@\idfromc@{1@m}%
  \vfromcartesian@@\Circmagic@@,1@\czeroEdge@\idfromc@{2@m}%
  \vfromcartesian@@-\Circmagic@@,1@\czeroEdge@\idfromc@{3@m}%
  \vfromcartesian@@-1,\Circmagic@@ @\czeroEdge@\idfromc@{4@m}%
  \vfromcartesian@@-1,-\Circmagic@@ @\czeroEdge@\idfromc@{5@m}%
  \vfromcartesian@@-\Circmagic@@,-1@\czeroEdge@\idfromc@{6@m}%
  \vfromcartesian@@\Circmagic@@,-1@\czeroEdge@\idfromc@{7@m}%
  \vfromcartesian@@1,-\Circmagic@@ @\czeroEdge@\idfromc@{8@m}%
 }

\xydef@\doCircleQuadrant@@#1#2{\save@ \ifcase#2\relax
 \doCircleQuadrant@@@{0@c}{1@m}{2@m}{1@c}{#1}\or
 \doCircleQuadrant@@@{1@c}{3@m}{4@m}{2@c}{#1}\or
 \doCircleQuadrant@@@{2@c}{5@m}{6@m}{3@c}{#1}\or
 \doCircleQuadrant@@@{3@c}{7@m}{8@m}{0@c}{#1}\or
 \doCircleQuadrant@@@{0@c}{1@m}{2@m}{1@c}{#1}\or
 \doCircleQuadrant@@@{1@c}{3@m}{4@m}{2@c}{#1}\or
 \doCircleQuadrant@@@{2@c}{5@m}{6@m}{3@c}{#1}\or
```

```
 \doCircleQuadrant@@@{3@c}{7@m}{8@m}{0@c}{#1}\fi \leave@ }
```

```
\xydef@\doCircleQuadrant@@@#1#2#3#4#5{%
 \cfromid@{#1}\pfromc@\senter@
 \cfromid@{#2}\spushc@\cfromid@{#3}\spushc@\cfromid@{#4}%
 \let\bstartPLACE@=\empty \connect@\crvs{#5}\relax\sinit@\sleave@
 \resetbreaks@}
```

The hook `\dosolidcircle@@` is provided so that back-ends may provide an alternative method to draw the circles/ellipses. Note that the token following `\solidcircle@` will be a group representing the ⟨style⟩ to be used.

```
\xydef@\dosolidcircle@#1{%
  \cubicCircleControls@@
  \doCircleQuadrant@@{#1}1 \relax\Clast@@
  \doCircleQuadrant@@{#1}2 \relax\Clast@@
  \doCircleQuadrant@@{#1}3 \relax\Clast@@
  \doCircleQuadrant@@{#1}4 \relax\Clast@@}%
```

```
\xydef@\solidcircle@{\dosolidcircle@@}
\xylet@\dosolidcircle@@=\dosolidcircle@
```

This places objects equally spaced around a circle, according to angular position.

Bug: objects are not equally spaced around ellipses constructed this way. To get regular spacing around ellipses, in terms or arc-length say, is much more difficult, both theoretically and practically.

To Do: the size of the resulting ⟨object⟩ does not take into account the size of the ⟨object⟩ dropped around the circle.

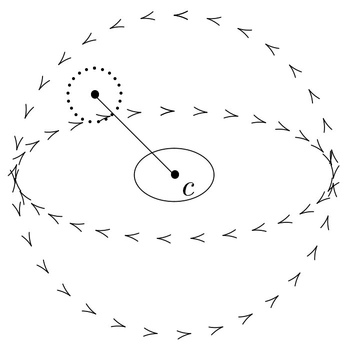
```
\xydef@\circledobjects@{\cubicCircleControls@@
 \setboxz@h{\expandafter\object\circleSTYLE@{}}%
 \ifdim\L@>\R@ \dimen@ii=\ht\z@ \advance\dimen@ii \dp\z@
  \ifdim\zz@\dimen@ii \dimen@ii=\wdz@\fi
 \else \dimen@ii=\wdz@
  \ifdim\zz@\dimen@ii \dimen@ii=\ht\z@ \advance\dimen@ii \dp\z@\fi
 \fi \ifdim\zz@\dimen@ii \dimen@ii=2\jot\fi
 \expandafter\dimen@\ifdim\L@>\R@ \twoPi@\L@\else \twoPi@\R@\fi
 \advance\dimen@.5\dimen@ii
 \count@@=\dimen@ \count@=\dimen@ii \divide\count@@\count@
 \edef\numobjects{\the\count@@}\dimen@=4\p@
 \dimen@nth\numobjects\dimen@   %  \divide\dimen@\numobjects\relax
 \edef\circleseparation@{\expandafter\removePT@\the\dimen@}%
 \edef\circleplace@{-\circleseparation@}\placeQuadrants@}%
```

```
\xydef@\placeQuadrants@{\DN@{\placeAllQuadrants@}%
 \ifx\onlyQuad@\undefined\else\ifx\onlyQuad@\empty\else
  \DN@{\expandafter\placeCircQuadrant@\expandafter{\onlyQuad@}}%
 \fi\fi \next@ }%
%\def\placeQuadrants@{\placeAllQuadrants@}
```

```
\xydef@\placeAllQuadrants@{%
 \placeCircQuadrant@{1}%
 \advance\dimen@-\p@\advance\dimen@-\circleseparation@\p@
 \edef\circleplace@{\expandafter\removePT@\the\dimen@}%
 \placeCircQuadrant@{2}%
 \advance\dimen@-\p@\advance\dimen@-\circleseparation@\p@
```

```
 \edef\circleplace@{\expandafter\removePT@\the\dimen@}%
 \placeCircQuadrant@{3}%
 \advance\dimen@-\p@\advance\dimen@-\circleseparation@\p@
 \edef\circleplace@{\expandafter\removePT@\the\dimen@}%
 \placeCircQuadrant@{4}}

\xydef@\placeCircQuadrant@#1{%
 \let\bstartPLACE@=\relax \doCircleQuadrant@@{}#1%
 \loop@
  \expandafter\dimen@\circleplace@\p@
  \advance\dimen@\circleseparation@\p@
  \edef\circleplace@{\expandafter\removePT@\the\dimen@}%
  \ifdim\dimen@<\p@ \placeCircObject@
 \repeat@ }

\xydef@\placeCircObject@{\begingroup
  \crvconnect@@ \Creset@@ \Invisible@false
  \expandafter\splinealong@\expandafter{\circleplace@}%
  \expandafter\drop\circleSTYLE@{}\endgroup }

\xydef@\buildcircle@{\save@
% \cubicCircleControls@@
 \DNii@{\circledobjects@}%
 \ifx\circleSTYLE@\empty\DNii@{\solidcircle@{}}%
 \else \expandafter\DN@\expandafter{\addDASH@{}}\relax
  \ifx\next@\circleSTYLE@ \DNii@{\solidcircle@{}}%
 \else \expandafter\DN@\expandafter{\addEQ@{}}%
  \ifx\next@\circleSTYLE@
   \expandafter\circletemplate@\expandafter{\addEQ@{}}{}%
 \else \expandafter\DN@\expandafter{\addDASH@ 2}%
  \ifx\next@\circleSTYLE@ \DNii@{\doublecircle@{}}%
 \else \expandafter\DN@\expandafter{\addDASH@ 3}%
  \ifx\next@\circleSTYLE@ \DNii@{\triplecircle@{}}%
 \else \expandafter\expandafter\expandafter\DN@\expandafter\expandafter
  \expandafter{\expandafter\addDASH@\addDASH@{}}%
  \ifx\next@\circleSTYLE@ \DNii@{\dashedcircle@{}}%
 \else \expandafter\expandafter\expandafter\DN@\expandafter\expandafter
  \expandafter{\expandafter\addEQ@\addEQ@{}}%
  \ifx\next@\circleSTYLE@ \expandafter\expandafter\expandafter
   \circletemplate@\expandafter\expandafter\expandafter{%
    \expandafter\addEQ@\addEQ@{}}2%
 \else \expandafter\expandafter\expandafter\DN@\expandafter\expandafter
  \expandafter{\expandafter\addEQ@\addEQ@2}%
  \ifx\next@\circleSTYLE@ \expandafter\expandafter\expandafter
   \circletemplate@\expandafter\expandafter\expandafter{%
    \expandafter\addEQ@\addEQ@{}}2%
 \else \expandafter\expandafter\expandafter\DN@\expandafter\expandafter
  \expandafter{\expandafter\addEQ@\addEQ@3}%
  \ifx\next@\circleSTYLE@ \expandafter\expandafter\expandafter
   \circletemplate@\expandafter\expandafter\expandafter{%
    \expandafter\addEQ@\addEQ@{}}3%
```

```
 \else \expandafter\DN@\expandafter{\addDOT@{}}\relax
  \ifx\next@\circleSTYLE@ \DNii@{\dottedcircle@{}}%
 \else \expandafter\DN@\expandafter{\addDOT@2}\relax
  \ifx\next@\circleSTYLE@ \DNii@{\dbldottedcircle@{}}%
 \else \expandafter\DN@\expandafter{\addDOT@3}\relax
  \ifx\next@\circleSTYLE@ \DNii@{\trpldottedcircle@{}}%
 \else \DN@{:}\relax
  \ifx\next@\circleSTYLE@ \DNii@{\dbldottedcircle@{}}%
% \else\DN@{\circledobjects@}%
 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \nextii@ \leave@}

\xydef@\circletemplate@#1#2{\def\circleSTYLE@{\dir#2{#1}}}
\xydef@\doublecircle@#1{%
 \advance\R@-\p@ \advance\L@-\p@ \solidcircle@{}%
 \advance\R@ 2\p@ \advance\L@ 2\p@ \solidcircle@{}}
\xydef@\triplecircle@#1{%
 \advance\R@-2\p@ \advance\L@-2\p@ \solidcircle@{}%
 \advance\R@ 2\p@ \advance\L@ 2\p@ \solidcircle@{}%
 \advance\R@ 2\p@ \advance\L@ 2\p@ \solidcircle@{}}

%\xydef@\dashedcircle@#1{\expandafter\expandafter\expandafter
%   \circletemplate@\expandafter\expandafter\expandafter{%
%    \expandafter\addDASH@\addDASH@{}}{}\circledobjects@ }%

\xydef@\dottedcircle@#1{%
 \expandafter\def\expandafter\circleSTYLE@\expandafter{%
%  \expandafter{\addDOT@{}}}%
 \zerodot@i}%
 \circledobjects@}

\xydef@\dashedcircle@#1{%
 \expandafter\def\expandafter\circleSTYLE@\expandafter{\zerodash@i}%
 \circledobjects@}

\xydef@\dbldottedcircle@#1{%
 \advance\R@-\p@ \advance\L@-\p@ \dottedcircle@{}%
 \advance\R@ 2\p@ \advance\L@ 2\p@ \dottedcircle@{}}
\xydef@\trpldottedcircle@{%
 \advance\R@-2\p@ \advance\L@-2\p@ \dottedcircle@{}%
 \advance\R@ 2\p@ \advance\L@ 2\p@ \dottedcircle@{}%
 \advance\R@ 2\p@ \advance\L@ 2\p@ \dottedcircle@{}}

{\xyuncatcodes \gdef\next{=<.75\jot>{\zerodot}}}
\xylet@\zerodot@i=\next

{\dimen0=2\xydashl@ \xyuncatcodes
\expandafter\gdef\expandafter\next\expandafter{\expandafter
 =\expandafter<\the\dimen0 >_\dir{|}}}
\xylet@\zerodash@i=\next
```




```
\xy 0;/r5pc/:*\dir{*}
 ;p+(.5,-.5)*\dir{*}="c"
,**\dir{-},*+!UL{c},"c"
,*\xycircle(1,.4){++\dir{<}}
,*\xycircle(1,1){++\dir{>}}
,*\xycircle<15pt,10pt>{}
;*\xycircle<10pt>{{.}}
\endxy
```

```
{\xyuncatcodes\gdef\next{--}}
\xylet@\dashcorntemp@@=\next
{\xyuncatcodes\gdef\next{.}}
\xylet@\dotcorntemp@@=\next

\xydef@\dotcorner@#1{%
  \expandafter\circleCorner@\expandafter{\dotcorntemp@@}{#1}}%
\xydef@\dashcorner@#1{%
  \expandafter\circleCorner@\expandafter{\dashcorntemp@@}{#1}}

\xydef@\circleCorner@#1#2{\setboxz@h\bgroup
 \edef\next{#2\relax\R@=\the\R@\relax\L@=\R@ }%
 \expandafter\xycircle@\next @{#1}%
 \ifcase#2\or \setboxz@h{\lower\R@\boxz@\kern\X@c}%
 \or \setboxz@h{\kern-\R@\lower\R@\boxz@\kern\R@\kern\X@c}%
 \or \setboxz@h{\kern-\R@\lower\R@\boxz@\kern\R@\kern\X@c}%
 \or \setboxz@h{\lower\R@\boxz@\kern\X@c}\fi
  \wdz@=\R@ \ht\z@=\R@ \dp\z@=\R@ \boxz@
 }
```

2.1.3 Quadratic Splines

Quadratic Bézier splines, as distinct from cubic Bézier splines, are constructed from parabolic arcs, using 'control points' to determine the tangents where successive arcs are joined.

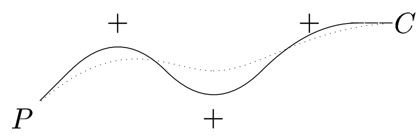
Various implementations of such curves exist. The one adopted here is consistent with the `xfig` drawing utility and TPIC implementations. These have the property of beginning and ending with straight segments, half the length to the corresponding adjacent control-point. Furthermore the mid-point between successive control-points lies on the spline, with the line joining the control-points being tangent there.

Such curves are specified, either as a ⟨decor⟩ or as an ⟨object⟩, using...

`\qspline{`⟨style⟩`}`

where the start and end of the curve are at p and c respectively. The control-points are taken from the current stack, see 1.3o. If this stack is empty then a straight line is constructed.

The following example compares the quadratic spline with the gentler curving B-spline having the same control points, using `\crvs`.




```
\xy /r1.5pc/:,+<5pc,3pc>*+{P};p
@(,+(2,2)*{+}@+, +(2,-2)*{+}@+
,+(2,2)*{+}@+, +(2,0)*+{C}="C"
,*\qspline{},"C",**\crvs{.}
,@i @)\endxy
```

If the current stack is empty, simply pass everything to `\crvs@`, as if the request originated from a `\crvs` or `\ar`. Otherwise we must build up the segments of the spline.

```
\xydef@\qspline{\hbox\bgroup\crvresetbreaks@ \xy@spline@}

\xydef@\xy@spline@#1#{\if\sempty@ \DN@{\crvs@{}\resetbreaks@}%
 \else\DN@{\xy@spline@i{#1}}\fi \next@ }%
```

The `#1` parameter to `\xy@spline@i` is to allow alternative implementations; currently it is ignored. When `#1` is indeed empty,

```
\xydef@\xy@spline@i#1#2{\def\splineSTYLE@{#2}%
 \DN@{\xy@spline@ii}\DNii@{#1}%
 \ifx\nextii@\empty\DN@{\xy@spline@ii@@}\fi \next@ }

\xydef@\xy@spline@ii{\save@ \cfromp@\idfromc@{@p}%
 \enter@\cplusthec@ \count@@=\s@top\advance\count@@\m@ne
 \cfroms@{\count@@}\leave@
 \halve@dimen\X@c  %  \divide\X@c\tw@
 \halve@dimen\Y@c  %  \divide\Y@c\tw@
 \czeroEdge@ \idfromc@{@c}%
 \expandafter\doSplineStraight@\expandafter{\splineSTYLE@}%
 \count@=\@ne \edef\next@{\the\count@}%
 \ifx\next@\s@top\DN@{\xy@spline@iv}\else\DN@{\xy@spline@iii}\fi
 \next@ }
\xylet@\xy@spline@ii@@=\xy@spline@ii

\xydef@\xy@spline@iii{\crv@cnt@=\s@top
 \loop \advance\crv@cnt@\m@ne \xy@spline@g \ifnum\crv@cnt@>\@ne\repeat
 \xy@spline@iv}

\xydef@\xy@spline@g{\cfromid@{@c}\idfromc@{@p}%
 \count@@=\crv@cnt@\cfroms@{\count@@}\idfromc@{@m}%
 \enter@\cplusthec@ \count@@=\crv@cnt@ \advance\count@@\m@ne
  \cfroms@{\count@@}\leave@
 \halve@dimen\X@c  %  \divide\X@c\tw@
 \halve@dimen\Y@c  %  \divide\Y@c\tw@
 \czeroEdge@\idfromc@{@c}%
 \expandafter\doSplineSegment@\expandafter{\splineSTYLE@}}
```

```
\xydef@\xy@spline@iv{\cfromid@{@c}\idfromc@{@p}%
 \cfroms@{\z@}\idfromc@{@m}\leave@ \save@
 \enter@\cplusthec@ \cfromid@{@m}\leave@
 \halve@dimen\X@c  %  \divide\X@c\tw@
 \halve@dimen\Y@c  %  \divide\Y@c\tw@
 \czeroEdge@\idfromc@{@c}%
 \expandafter\doSplineSegment@\expandafter{\splineSTYLE@}%
 \xy@spline@v }

\xydef@\xy@spline@v{\cfromid@{@c}\idfromc@{@p}%
 \leave@ \save@ \idfromc@{@c}%
 \expandafter\doSplineStraight@\expandafter{\splineSTYLE@}%
 \xy@spline@x }

\xydef@\xy@spline@x{\leave@
 \def\Drop@@{\splineDrop@ \resetbreaks@}%
 \def\Connect@@{\straight@\relax\splineConnect@ \resetbreaks@}%
 \OBJECT@x}

\xydef@\splineDrop@{\ifInvisible@\setboxz@h{}%
 \else\setboxz@h{\kern-\the\X@c\raise-\the\Y@c\hbox{\styledboxz@}}\fi
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@ }

\xydef@\splineConnect@{%
 \ifInvisible@\else\box\lastobjectbox@\fi }%
```

The initial and final points in a segment are stored as `@p` and `@c` respectively. Where needed, the control is in `@m`.

```
\xydef@\doSplineStraight@#1{\begingroup \sinit@
 \cfromid@{@p}\pfromc@\cfromid@{@c}%
 \connect@\crvs{#1}\relax\sleave@\endgroup }

\xydef@\doSplineSegment@#1{\begingroup \sinit@
 \cfromid@{@p}\pfromc@\senter@\cfromid@{@m}\spushc@\cfromid@{@c}%
 \connect@\crvs{#1}\relax\sinit@\sleave@\endgroup }
```

The end & Log

```
\xyendinput

% $Log: xycurve.doc,v $
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  09:01:45  ross
% major revisions for the new BREAK methods
% spline edge-finding is more robust
% spline-breaks fully implemented
% intersections of curve with straight connections implemented
% improved tracings
% adjusted methods for styles
```

```
%
% Revision 3.2  1995/09/19  18:20:20  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/05  07:58:43  ross
% Ready for v3 release?
%
% Revision 2.12  1994/10/25  03:01:14  ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
%
% Revision 2.12  1994/09/05  08:22:11  ross
% incorporates some speed-ups, extra documentation
%
% Revision 2.11  1994/07/05  09:27:52  ross
% Minor fixes; use 2.11 kernel stack code.
%
% Revision 2.9  1994/06/09  14:50:54  ross
% Release 3beta.
%
% Revision 2.8  1994/04/08  10:36:40  ross
% Second 3alpha release.
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  ross
% Last internal 3alpha and pre-2.7 release.
%
% NEW for version 2.7 inspired by Knuth's picmac.tex.
```

2.2 Frame and Bracket extension

Vers. 3.6 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

The `frame` extension provides a variety of ways to puts frames in XY-pictures.

Header:

```
%% $Id: xyframe.doc,v 3.6 1998/03/06 01:28:05 krisrose Exp krisrose $
%%
%% Xy-pic ``Frames and Brackets'' extension.
%% Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@brics.dk>
%%
```

```
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{frame}{Frame and Bracket extension}{\stripRCS$Revision: 3.6 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

The frames are XY-pic ⟨object⟩s on the form

`\frm{` ⟨frame⟩ `}`

---

to be used in ⟨pos⟩itions: Dropping a frame with `*`...`\frm{`⟨frame⟩`}` will frame the c object; connecting with `**`...`\frm{`...⟨frame⟩`}` will frame the result of $c.p$.

XY-pic interface The frames are integrated in XY-pic as ⟨object⟩s as follows. They generate a dummy object, then define the `\Drop@@` action to place the requested frame. The `\Connect@@` operation is defined to place the requested frame around the current *and* the previous object (using the `\cmerge@` command of the kernel).

```
\xydef@\frm{\hbox{}\dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@
 \ht\z@=\U@c \dp\z@=\D@c \dimen@=\U@c \advance\dimen@\D@c
% \ifdim\wdz@=\z@ \def\Leftness@{0}\else \quotient@\Leftness@\L@c\wdz@ \fi
% \ifdim\dimen@=\z@ \def\Upness@{0}\else \quotient@\Upness@\U@c\dimen@ \fi
 \ifdim\wdz@=\z@ \def\Leftness@{.5}\else \quotient@\Leftness@\L@c\wdz@ \fi
 \ifdim\dimen@=\z@ \def\Upness@{.5}\else \quotient@\Upness@\U@c\dimen@ \fi
 \expandafter\Edge@c\expandafter{\prevEdge@@}\let\frmradius@@=\z@
 \def\Drop@@{}%
 \def\Connect@@{\xy@@{\save@\cmerge@\Y@p\X@p\D@p\U@p\L@p\R@p\Drop@@}\restore}%
 \xyFN@\frm@i}
```

```
\xylet@\frmradius@@=\z@
```

```
\xydef@\frm@i{%
 \addLT@\ifx\next \addGT@{\addLT@\DN@##1}{\def\frmradius@@{##1}\xyFN@\frm@i}%
```

```
 \else \DN@##1##{\frm@{##1}}\fi \next@}
```

The main command is `\frm@` which looks up if a custom control sequence is available for the requested frame or whether the generic 'curve along the edge' code should be invoked.

```
\xydef@\frm@#1#2{\DNii@{frm#1{#2}}%
 \expandafter\let\expandafter\next@\csname\codeof\nextii@\endcsname
 \ifx\next@\relax \DNii@{frm{#2}}%
  \expandafter\let\expandafter\next@\csname\codeof\nextii@\endcsname
  \ifx\next@\relax \let\next@=\frm@generic \fi\fi \next@}

\xydef@\frm@generic{\xyerror@{No generic frames yet!}{}}
```

This extra command is used to define `\Drop@@` for each frame such that `\frmradius@@` is available.

```
\xydef@\frmDrop@#1{%
 \ifx\frmradius@@\z@ \addtoDrop@@{\let\frmradius@@=\z@ #1}%
 \else \expandafter\addtoDrop@@\expandafter{%
    \expandafter\def\expandafter\frmradius@@\expandafter{\frmradius@@}#1}\fi}

\xydef@\EdgefromtheEdge@{%
 \U@c=\the\U@c \D@c=\the\D@c \L@c=\the\L@c \R@c=\the\R@c
 \Edge@c={\expandafter\noexpand\the\Edge@c}}
\xydef@\prevEdgefromtheEdge@{%
 \U@c=\the\U@c \D@c=\the\D@c \L@c=\the\L@c \R@c=\the\R@c
 \noexpand\def\noexpand\prevEdge@@{\expandafter\noexpand\the\Edge@c}}

\xydef@\addtoDrop@@#1{%
 \expandafter\def\expandafter\Drop@@\expandafter{\Drop@@#1}}
\xydef@\addbeforeDrop@@#1{\DNii@{#1}%
 \expandafter\DN@\expandafter{\expandafter\nextii@\Drop@@}%
 \expandafter\expandafter\expandafter\def
  \expandafter\expandafter\expandafter\Drop@@
   \expandafter\expandafter\expandafter{\next@}}
```

Below we distinguish between 'ordinary' frames, 'brackets' and 'fills'; last we present how some frames can be added to other objects using object modifier ⟨shape⟩s.

2.2.1 Frames

Figure 2.2 shows the possible frames and the applicable ⟨modifier⟩s with reference to the notes below.

Notes

2.2a. The `\frm{}` frame is a dummy useful for not putting a frame on something, *e.g.*, in macros that take a ⟨frame⟩ argument.

```
\xydefcsname@{frm{}}{}
\xyletcsnamecsname@{frm[]{}}{frm{}}
\xyletcsnamecsname@{frm[o]{}}{frm{}}
```

2.2b. *Rectangular* frames include `\frm{.}`, `\frm{-}`, `\frm{=}`, `\frm{--}`, `\frm{==}`, and `\frm{o-}`. They all make rectangular frames that essentially trace the border of a rectangle-shaped object.

The ⟨frame⟩s `\frm{-}` and `\frm{=}` allow an optional *corner radius* that rounds the corners of the frame with quarter circles of the specified radius. This is not allowed for the other frames—

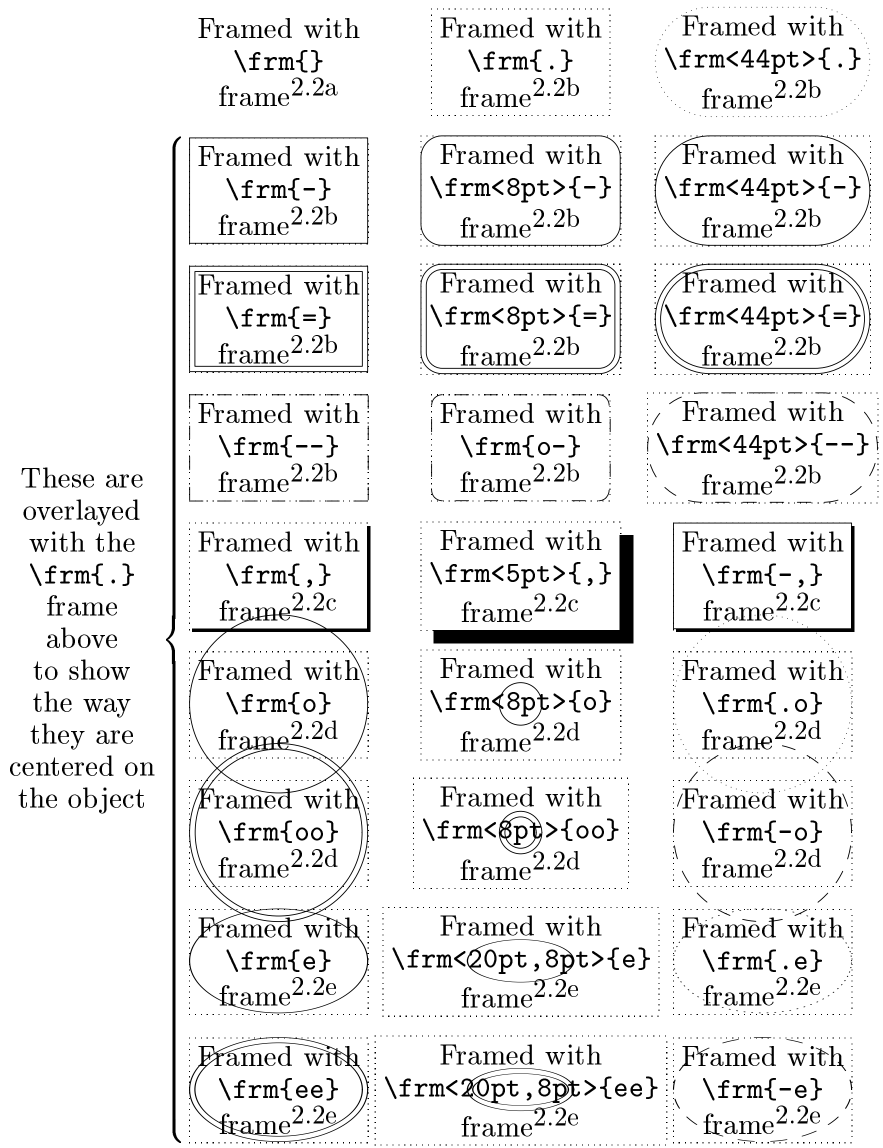


Figure 2.2: Plain ⟨frame⟩s.

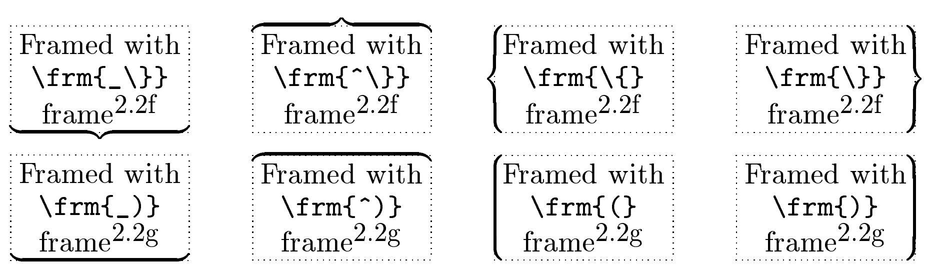
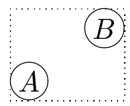


Figure 2.3: Bracket ⟨frame⟩s.

the `\frm{o-}` frame always gives rounded corners of the same size as the used dashes (when `\xydashfont` is the default one then these are `5pt` in radius).

Exercise 2.4: How do you think the author typeset the following?



The commands still hack away with rules and stuff...

`\frm{.}` just fills the edges of the object rectangle border with dots using dots as in dotted connections in the kernel.

```
\xydefcsname@{frm{.}}{\frmDrop@{\let\zerocorner=\framezerodot@@
  \let\framehfill=\frm@doth@@ \let\framevfill=\frm@dotv@@
  \framed@@{\frmradius@@}}}

\xyletcsnamecsname@{frm[]{.}}{frm{.}}

\xydef@\frm@doth@@{\setboxz@h{\kern5\B@\framezerodot@@\kern5\B@}\ht\z@=\B@
 \dp\z@=\B@\kern5\B@\xleaders\boxz@\hss\kern5\B@}
\xydef@\frm@dotv@@{\setboxz@h to2\B@{\hss\framezerodot@@\hss}\ht\z@=5\B@
 \dp\z@=5\B@\kern5\B@\xleaders\boxz@\vss\kern5\B@}

\xyletcsnamecsname@{frm{..}}{frm{.}}

\xylet@\framezerodot@@=\zerodot
```

The `\frm`⟨variant⟩`{-}` and `\frm`⟨variant⟩`{=}` set a single/double frame that just surrounds the current object; if a ⟨radius⟩ is given then it should be a `<`⟨dimen⟩`>` where the ⟨dimen⟩ will be used for the corner radius.

The commands for `\frm{-}` and `\frm{=}` are quite similar in that they just call `\framed@` one or two times, respectively, with the corner radius and L, R, D, U extents reflecting the size of the frame. `\framed@` then sets the horizontal and vertical fill commands to generate rules of the right length, then calls `\framed@@` where the real work is done.

```
\xydefcsname@{frm{-}}{\frmDrop@{\framed@\frmradius@@}}
\xyletcsnamecsname@{frm[]{-}}{frm{-}}

\xydef@\framed@{\let\zerocorner=\framezerodot@@
 \let\framehfill=\frm@solidh@@ \let\framevfill=\frm@solidv@@ \framed@@}

\xylet@\framesetthick@=\relax

\xydef@\frm@solidh@@{\framesetthick@\leaders\hrule height\B@ depth\B@\hfill}
\xydef@\frm@solidv@@{\framesetthick@\leaders\vrule width2\B@\vfill}

\xydefcsname@{frm{=}}{\frmDrop@{\framed@\frmradius@@
  {\advance\L@c-2\p@ \advance\R@c-2\p@ \advance\U@c-2\p@ \advance\D@c-2\p@
   \dimen@=\frmradius@@
   \ifdim\dimen@>2\p@ \advance\dimen@-2\p@ \else \dimen@=\z@\fi
  \framed@\dimen@}}}

\xyletcsnamecsname@{frm[]{=}}{frm{=}}
```

When the line extension is also loaded then we use the line thickness for frames:

```
\xydef@\framesetthick@line{\B@=.5\xylinethick@}
```

```
\xydef@\frm@thickc@@{\hbox{\framesetthick@line
 \vrule height.5\B@ depth.5\B@ width\B@}}
```

```
\xywithoption{line}{%
 \let\framezerodot@@\frm@thickc@@ \let\framesetthick@=\framesetthick@line}
```

\framed@@ is where we build a box with the sides of the frame displaced appropriately. **To DoGeneralise this to handle any directional!**

```
%       Procedure:
%       %0 setup hrulefill and vrulefill as apropriate...
%       %1 Lower R to be less than half of both (U+D) and (L+R).
%       %2 Start vbox at (X-L,Y-D) except center overshoot.
%       %3 Generate corner C04 at R; w[\A@] := -1/2 width(corner);
%          r[\B@] := 1/2 rulewidth; h[\dimen@ii] := U+D+2w;
%       %4 Row 1: hbox to W{kern w C04 hrulefill C03 kern w}.
%       %5 Row 2: hbox to W{kern-r vbox to h{vrulefill}
%                             hfill vbox to h{vrulefill} kern-r}.
%       %6 Row 3: hbox to W{kern w C01 hrulefill C02 kern w}.
```

```
\xydef@\framed@@#1{\setboxz@h{\R@=#1\relax
 \dimen@=\L@c \advance\dimen@\R@c \dimen@ii=\U@c \advance\dimen@ii\D@c %1
 \ifdim.5\dimen@<\R@ \R@=.5\dimen@ \fi
 \ifdim.5\dimen@ii<\R@ \R@=.5\dimen@ii \fi
 \A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@-\D@c                  %2
 \ifdim\R@<\p@\else \cirrestrict@@ \fi
 \dimen@=2\R@ \advance\dimen@-\L@c \advance\dimen@-\R@c
 \ifdim\dimen@>\z@ \advance\A@-.5\dimen@ \fi
 \dimen@=2\R@ \advance\dimen@-\U@c \advance\dimen@-\D@c
 \ifdim\dimen@>\z@ \advance\B@-.5\dimen@ \fi
 \kern\A@ \raise\B@\vbox to \dimen@ii{\framed@body@@}}%
\ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}
```

```
\xydef@\framed@body@{%
\advance\dimen@ii.22\p@ \advance\dimen@-.2\p@
\setboxz@h{\corn@@4\R@}\A@=-.5\wdz@ \B@=.5\xydashw@                   %3
\dimen@=\L@c \advance\dimen@\R@c \advance\dimen@-.2\p@
\setboxz@h to\dimen@{\kern\A@ \boxz@ \framehfill\corn@@3\R@\kern\A@}%4
\advance\dimen@ii-\dp\z@ \kern-\ht\z@ \nointerlineskip\boxz@
\dimen@=\L@c \advance\dimen@\R@c \advance\dimen@-.2\p@
\setboxz@h to\dimen@{\kern\A@\corn@@1\R@\framehfill\corn@@2\R@\kern\A@}%6
\advance\dimen@ii-\ht\z@
\dimen@=\L@c \advance\dimen@\R@c \advance\dimen@.2\p@
\nointerlineskip\hbox to\dimen@{%                                      %5
 \setbox\z@=\vbox to\dimen@ii{\framevfill}\kern-.5\wdz@\copy\z@
 \hss \copy\z@\kern-.5\wdz@ }%
\nointerlineskip\boxz@ \vss \kern\z@}
```

```
\xylet@\framed@body@@=\framed@body@
```

```
\xylet@\zerocorner=\framezerodot@@
```

```
\xydef@\solidcorner@#1#2{\hbox\bgroup \R@=#2\relax
```

```
 \ifdim\R@<\p@ \zerocorner
 \else \cirrestrict@@ \multiply\count@8 %
  \setboxz@h{\corn@x@@{#1}}\dimen@=\wdz@
  \ifcase#1\or \ht\z@=\z@ \kern\dimen@ \raise\dimen@\boxz@
  \or \ht\z@=\z@ \raise\dimen@\boxz@ \kern\dimen@
  \or \dp\z@=\z@ \lower\dimen@\boxz@ \kern\dimen@
  \or \dp\z@=\z@ \kern\dimen@ \lower\dimen@\boxz@ \fi
  \vrule height\dimen@ depth\dimen@ width\z@\fi
 \edef\tmp@{\egroup \U@c=\the\dimen@}\tmp@ \D@c=\U@c \L@c=\U@c \R@c=\U@c
 \Edge@c={\circleEdge}\ignorespaces}
```

```
\xydef@\corn@x@#1{\ifcase#1\or
 \circhar@@\z@ \circhar@@\@ne\or\circhar@@\tw@ \circhar@@3\or
 \circhar@@5\circhar@@4\or\circhar@@7\circhar@@6\fi}
```

```
\xylet@\corn@x@@=\corn@x@
```

```
\xydef@\corn@@{\DN@{\solidcorner@}%
  \ifx\framehfill\frm@doth@@ \DN@{\let\cirrestrict@@=\relax
   \let\corn@x@@=\dotcorner@@ \let\zerodot=\framezerodot@@
   \solidcorner@ }%
  \else\ifx\framehfill\frm@dashh@@ \DN@{\let\cirrestrict@@=\relax
   \let\corn@x@@=\dashcorner@@ \let\zerodot=\framezerodot@@
   \solidcorner@ }%
 \fi\fi \next@}
```

```
\xydef@\dotcorner@@#1{\zerocorner}
\xydef@\dashcorner@@#1{\zerocorner}
```

```
\xywithoption{curve}{%
 \let\dotcorner@@=\dotcorner@ \let\dashcorner@@=\dashcorner@}
```

The dashed frames 'dash' with the dash used for dashed lines.

```
\xydefcsname@{frm{--}}{%
  \frmDrop@{\let\zerocorner=\framezerodot@@
  \let\framehfill=\frm@dashh@@ \let\framevfill=\frm@dashv@@
  \framed@@\frmradius@@}}
```

```
\xyletcsnamecsname@{frm[]{--}}{frm{--}}
```

```
\xydef@\frm@dashh@@{\setboxz@h{\dashhfillchar}%
  \ht\z@=\B@ \dp\z@=\B@ \wdz@=2\xydashl@
  \copy\z@ \xleaders\copy\z@\hss \boxz@\kern-\xydashl@}
\xydef@\frm@dashv@@{\setboxz@h to2\B@{\hss\dashvfillchar\hss}\ht\z@=\z@
  \dp\z@=2\xydashl@ \nointerlineskip\copy\z@ \xleaders\copy\z@\vss
  \nointerlineskip\boxz@\kern-\xydashl@}%
```

```
\xydef@\dashhfillchar{{\rDirection@\xydashl@ \line@@}}
\xydef@\dashvfillchar{{\dDirection@\xydashl@ \line@@}}
```

Bug: Setting the corner-radius to less than 10pt has no effect when using `\frm{o-}`; instead a default size of `.8\xydashl@` is used.

```
\xydefcsname@{frm{o-}}{%
 \ifdim\frmradius@@<2\xydashl@\relax \dimen@=.8\xydashl@
  \expandafter\def\expandafter\frmradius@@\expandafter{\the\dimen@}\fi
```

```
 \frmDrop@{\let\zerocorner=\framezerodot@@
  \let\framehfill=\frm@rddashh@@ \let\framevfill=\frm@rddashv@@
  \framed@@\frmradius@@}}%
```

```
\xydef@\frm@rddashh@@{\setboxz@h{\kern\xydashl@\dashhfillchar}%
  \ht\z@=\B@ \dp\z@=\B@ \copy\z@ \xleaders\copy\z@\hss \boxz@ \kern\xydashl@}
\xydef@\frm@rddashv@@{\ifdim\dimen@ii<2\xydashl@\else
  \setbox\z@=\vbox to2\xydashl@{\hbox to2\B@{\hss\dashvfillchar\hss}\vss}%
  \kern\xydashl@ \copy\z@ \xleaders\copy\z@\vss
  {\ifdim\dimen@ii<2\xydashl@\else\aftergroup\boxz@\fi}\fi}
```

2.2c. The frame `\frm{,}` puts a shade, built from rules, into the picture beneath the (assumed rectangular) object, thereby giving the illusion of 'lifting' it; `\frm<`⟨dimen⟩`>{,}` makes this shade ⟨dimen⟩ deep.

`\frm{-,}` combines a `\frm{-}` with a `\frm{,}`.

A black rule is just that, a shade is two rules placed under and left of the rectangular object outline.

```
\xydefcsname@{frm{,}}{%
 \ifx\frmradius@@\z@ \shaded@{1.2\p@}\else \shaded@\frmradius@@ \fi}
```

```
\xydef@\shaded@#1{\frmDrop@{\R@=#1\relax
  {\advance\X@c\R@ \advance\Y@c-\D@c \U@c=\z@ \D@c=\R@ \blacked@}%
  {\advance\X@c\R@c \advance\Y@c-\R@ \L@c=\z@ \R@c=\R@ \blacked@}}}
```

```
\xydefcsname@{frm{-,}}{\addtoDrop@@{\drop\frm{-}\drop\frm{,}}}
```

```
\xydef@\blacked@{\setboxz@h{%
  \A@=\X@c \advance\A@-\L@c \dimen@=\L@c \advance\dimen@\R@c
  \B@=\Y@c \advance\B@-\D@c \dimen@ii=\Y@c \advance\dimen@ii\U@c
  \advance\A@-.5\xydashw@
 % \advance\B@-\xydashw@  \advance\dimen@ii-.5\xydashw@
  \kern\A@ \blacked@@ }%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}
```

```
\xydef@\bblacked@{\setboxz@h{%
  \dimen@=\L@c \advance\dimen@\R@c \B@=-\D@c \dimen@ii=\U@c
  \kern-\L@c \blacked@@ \kern-\dimen@ \kern\L@c}%
  \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}
```

```
\xydef@\blacked@@{\vrule width\dimen@ depth-\B@ height\dimen@ii}
```

The `\blacked` option is subsumed within a more general `\filled`; see below.

2.2d. Circles done with `\frm{o}` have radius as $(R + L)/2$ and with `\frm<`⟨dimen⟩`>{o}` have radius as the ⟨dimen⟩; `\frm{oo}` makes a double circle with the outermost circle being the same as that of `\frm{o}`.

Exercise 2.5: What is the difference between `*\cir{}` and `*\frm{o}`?

```
\xydefcsname@{frm{o}}{%
 \ifx\frmradius@@\z@
  \frmDrop@{\dimen@=.5\L@c \advance\dimen@.5\R@c
   \let\framehfill=\frm@solidh@@ \let\framevfill=\frm@solidv@@
```

```
   \circled@\dimen@}%
 \else
  \frmDrop@{\let\framehfill=\frm@solidh@@
   \let\framevfill=\frm@solidv@@ \circled@\frmradius@@}%
 \fi}
```

```
\xydefcsname@{frm{oo}}{%
 \ifx\frmradius@@\z@
  \frmDrop@{\dimen@=.5\L@c \advance\dimen@.5\R@c
   \let\framehfill=\frm@solidh@@ \let\framevfill=\frm@solidv@@
   {\circled@\dimen@}\advance\dimen@-2.1\p@ \circled@\dimen@}%
 \else
  \frmDrop@{\dimen@=\frmradius@@
   \let\framehfill=\frm@solidh@@ \let\framevfill=\frm@solidv@@
   {\circled@\dimen@}\advance\dimen@-2.1\p@ \circled@\dimen@}%
 \fi}
```

```
\xydef@\circled@#1{\setboxz@h\bgroup \R@=#1\relax
  \setboxz@h{\vbox{\circled@x@@}}%
  \A@=\X@c \advance\A@-.5\wd\z@
  \B@=\Y@c \advance\B@-.5\ht\z@ \advance\B@.5\dp\z@ \advance\B@.2\p@
  \edef\tmp@{\egroup\dimen@=\the\wdz@}%
  \kern\A@ \raise\B@\boxz@
 \tmp@ \divide\dimen@4 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}
```

```
\xydef@\circled@x@{\hbox{\corn@@4\R@ \corn@@3\R@}%
    \nointerlineskip\hbox{\corn@@1\R@ \corn@@2\R@}}
\xylet@\circled@x@@=\circled@x@
```

The code here is repetitive and could be improved. The only purpose of `\let\framehfill=\frm@dashh@@` etc. is to serve as a marker for the required style: solid/dashed/dotted.

```
\xydefcsname@{frm{.o}}{%
 \ifx\frmradius@@\z@ \frmDrop@{\let\framehfill=\frm@doth@@
  \dimen@=.5\L@c \advance\dimen@.5\R@c \circled@\dimen@}%
 \else
  \frmDrop@{\let\framehfill=\frm@doth@@ \circled@\frmradius@@}%
 \fi}
```

```
\xydefcsname@{frm{-o}}{%
 \ifx\frmradius@@\z@ \frmDrop@{\let\framehfill=\frm@dashh@@
   \dimen@=.5\L@c \advance\dimen@.5\R@c \circled@\dimen@}%
 \else
  \frmDrop@{\let\framehfill=\frm@dashh@@ \circled@\frmradius@@}%
 \fi}
```

2.2e. Ellipses specified using `\frm{e}` have axis lengths $(R + L)/2$ and $(U + D)/2$, while those with `\frm<`⟨dimen,dimen⟩`>{e}` use the given lengths for the axes. `\frm{ee}` makes a double ellipse with outermost ellipse being the same as that of `\frm{e}`.

Without special support to render the ellipses, either via a ⟨driver⟩ or using the `arc` feature, the ellipse will be drawn as a circle of radius approximately the average of the major and minor axes.

```
\xydefcsname@{frm{e}}{\ellipse@whichfrm@\ellipsesolidframed@}%
```

```
\xydefcsname@{frm{.e}}{\ellipse@whichfrm@\ellipsedotframed@}%
\xydefcsname@{frm{-e}}{\ellipse@whichfrm@\ellipsedashframed@}%

\xydefcsname@{frm{ee}}{\ellipse@whichdblfrm@\ellipsesolidframed@}%
\xydefcsname@{frm2{.e}}{\ellipse@whichdblfrm@\ellipsedotframed@}%
\xydefcsname@{frm2{-e}}{\ellipse@whichdblfrm@\ellipsedashframed@}%

\xyletcsnamecsname@{frm[o]{-}}{frm{e}}
\xyletcsnamecsname@{frm[o]{=}}{frm{ee}}
\xyletcsnamecsname@{frm[o]{.}}{frm{.e}}
\xyletcsnamecsname@{frm[o]{--}}{frm{-e}}

\xydef@\ellipse@whichfrm@#1{%
 \ifx\frmradius@@\z@
  \frmDrop@{#1\dimen@=.5\L@c \advance\dimen@.5\R@c
   \dimen@ii=.5\U@c \advance\dimen@ii.5\D@c
   \edef\next@{\noexpand\ellipsed@{\the\dimen@}{\the\dimen@ii}}\next@ }%
 \else
  \frmDrop@{#1\expandafter\ellipsed@i@\frmradius@@,,@@}%
 \fi}

\xydef@\ellipse@whichdblfrm@#1{%
 \ifx\frmradius@@\z@
  \frmDrop@{#1\dimen@=.5\L@c \advance\dimen@.5\R@c
   \dimen@ii=.5\U@c \advance\dimen@ii.5\D@c
   \edef\next@{\noexpand\ellipsed@e@{\the\dimen@}{\the\dimen@ii}}\next@ }%
 \else
  \frmDrop@{#1\expandafter\ellipsed@ei@\frmradius@@,,@@}%
 \fi}

\xydef@\ellipsed@i@#1,#2,#3@@{\DNii@{,}\DN@{#3}%
 \ifx\next@\nextii@
  \dimen@=#1\relax \dimen@ii=#2\relax
  \edef\next@{\noexpand\ellipsed@{\the\dimen@}{\the\dimen@ii}}%
 \else \ifx\next@\empty \DN@{#2}%
  \ifx\next@\empty \DN@{#1}%
    \dimen@=.5\L@c \advance\dimen@.5\R@c
    \dimen@ii=.5\U@c \advance\dimen@ii.5\D@c
    \edef\next@{\noexpand\ellipsed@{\the\dimen@}{\the\dimen@ii}}%
  \else \dimen@=#1\relax
   \edef\next@{\noexpand\ellipsed@{\the\dimen@}{\the\dimen@ii}}%
  \fi
 \else\xywarning@{ignoring extra tokens in elliptical frame: #3 }%
  \dimen@=#1\relax \dimen@ii=#2\relax
  \edef\next@{\noexpand\ellipsed@{\the\dimen@}{\the\dimen@ii}}%
 \fi\fi \next@ }

\xydef@\ellipsed@ei@#1,#2,#3@@{\DNii@{,}\DN@{#3}%
 \ifx\next@\nextii@
  \dimen@=#1\relax \dimen@ii=#2\relax
  \edef\next@{\noexpand\ellipsed@e@{\the\dimen@}{\the\dimen@ii}}%
 \else \ifx\next@\empty \DN@{#2}%
  \ifx\next@\empty \DN@{#1}%
```

```
    \dimen@=.5\L@c \advance\dimen@.5\R@c
    \dimen@ii=.5\U@c \advance\dimen@ii.5\D@c
   \edef\next@{\noexpand\ellipsed@e@{\the\dimen@}{\the\dimen@ii}}%
  \else
   \dimen@=#1\relax
   \edef\next@{\noexpand\ellipsed@e@{\the\dimen@}{\the\dimen@}}%
  \fi
 \else\xywarning@{ignoring extra tokens in elliptical frame: #3 }%
  \dimen@=#1\relax \dimen@ii=#2\relax
   \edef\next@{\noexpand\ellipsed@e@{\the\dimen@}{\the\dimen@ii}}%
 \fi\fi \next@ }

\xydef@\ellipsed@e@#1#2{%
  \dimen@=#1\relax \dimen@ii=#2\relax
  \edef\next@{\noexpand\ellipsed@{\the\dimen@}{\the\dimen@ii}}\next@
  \dimen@=#1\relax \dimen@ii=#2\relax
  \ifdim\dimen@>2\p@ \advance\dimen@-2\p@ \else \dimen@=\z@\fi
  \ifdim\dimen@ii>2\p@ \advance\dimen@ii-2\p@ \else \dimen@ii=\z@\fi
  \edef\next@{\noexpand\ellipsed@{\the\dimen@}{\the\dimen@ii}}\next@ }%

\xydef@\ellipsed@#1#2{\setboxz@h\bgroup \relax
  \setboxz@h{\vbox{\ellipsed@x@@{#1}{#2}}}%
  \A@=\X@c \advance\A@-.5\wd\z@
  \B@=\Y@c \advance\B@-.5\ht\z@ \advance\B@.5\dp\z@ \advance\B@.2\p@
  \edef\tmp@{\egroup\dimen@=\the\wdz@}%
  \kern\A@ \raise\B@\boxz@
 \tmp@ \divide\dimen@4 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}

\xydef@\ellipsed@i#1#2{{\R@c=#1\relax\L@c=\R@c
 \U@c=#2\relax\D@c=\U@c \ellipsed@{#1}{#2}}}

\xydef@\ellipsed@x@#1#2{\R@=#1\relax\advance\R@#2\relax
 \divide\R@\tw@ \circled@x@@ }
\xylet@\ellipsed@x@@=\ellipsed@x@

\xydef@\ellipsesolidframed@{%
 \let\framehfill=\frm@solidh@@ \let\framevfill=\frm@solidv@@}
\xydef@\ellipsedotframed@{%
 \let\framehfill=\frm@doth@@ \let\framevfill=\frm@dotv@@}
\xydef@\ellipsedashframed@{%
 \let\framehfill=\frm@dashh@@ \let\framevfill=\frm@dashv@@}
```

To Do: Allow ⟨frame variant⟩s like those used for directionals, *i.e.*, `\frm2{-}` should be the same as `\frm{=}`. Add `\frm{o,}` and more brackets.

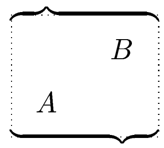
2.2.2 Brackets

The possible brackets are shown in figure 2.3 with notes below.

Notes

2.2f. *Braces* are just the standard plain TEX large braces inserted correctly in XY-pic pictures with the 'nib' aligned with the reference point of the object they brace.

Exercise 2.6: How do you think the author typeset the following?



They just use the pieces of plain TEX brace delimiters at `\Bigg` size and those of `\overbrace` and `\underbrace`.

```
\xydefcsname@{frm{\{}}{\addtoDrop@@{\lbraced}}
\xydefcsname@{frm{\}}}{\addtoDrop@@{\rbraced}}
```

```
\xydefcsname@{frm{^\}}}{\addtoDrop@@{\ubraced}}
\xydefcsname@{frm{_\}}}{\addtoDrop@@{\dbraced}}
```

The horizontal ones mimick the plain TEX ones quite closely:

```
\xydef@\dbraced{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@-\D@c
  \setboxz@h{$\m@th\bracelu$}\dimen@=2\wdz@ \advance\B@-.5\ht\z@
  \ifdim\R@c<\dimen@ \R@c=\dimen@ \fi
  \ifdim\L@c<\dimen@ \advance\A@-\dimen@ \advance\A@\L@c \L@c=\dimen@ \fi
  \advance\A@-.25pt %
  \kern\A@\raise\B@\hbox{%
   \hbox to\L@c{$\m@th\bracelu\leaders\vrule\hfil\bracerd$}%
   \kern.5pt %
   \hbox to\R@c{$\m@th\braceld\leaders\vrule\hfil\braceru$}}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

```
\xydef@\ubraced{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@\U@c
  \setboxz@h{$\m@th\bracelu$}\dimen@=2\wdz@
  \ifdim\R@c<\dimen@ \R@c=\dimen@ \fi
  \ifdim\L@c<\dimen@ \advance\A@-\dimen@ \advance\A@\L@c \L@c=\dimen@ \fi
  \advance\A@-.25pt %
  \kern\A@\raise\B@\hbox{%
   \hbox to\L@c{$\m@th\braceld\leaders\vrule\hfil\braceru$}%
   \kern.5pt %
   \hbox to\R@c{$\m@th\bracelu\leaders\vrule\hfil\bracerd$}}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

The inserted extra `.5pt` at `(1)` is the authours responsibility…

The vertical ones repeat the above for the vertical brace extension characters:

```
{\catcode`\"=12 %
 \global\mathchardef\braceur="338 %
 \global\mathchardef\braceul="339 %
 \global\mathchardef\bracedr="33A %
 \global\mathchardef\bracedl="33B %
 \global\mathchardef\bracecl="33C %
 \global\mathchardef\bracecr="33D %
 \global\mathchardef\bracec="33E }
```

```
\xydef@\lbraced{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@-\D@c
  \setboxz@h{$\m@th\bracecl$}\dimen@ii=\dp\z@ \advance\A@-.5\wdz@
  \setboxz@h{$\m@th\bracec$}\dimen@=\dp\z@
  \ifdim\U@c<.5\dimen@ \U@c=.5\dimen@ \fi
  \ifdim\D@c<.5\dimen@ \advance\B@-.5\dimen@ \advance\B@\D@c \D@c=.5\dimen@ \fi
  \advance\U@c.6\p@ \advance\D@c.6\p@  \advance\B@-.6\p@
  \kern\A@\raise\B@\vbox{\vbox to\U@c{%
     \nointerlineskip\hbox{$\m@th\braceur$}%
     \kern-.61\dimen@ \cleaders\copy\z@\vfil \kern-.4\dimen@
     \nointerlineskip\hbox{$\m@th\bracecl$}\kern-.5\dimen@ii}%
    \nointerlineskip\vbox to\D@c{\kern.5\dimen@ii
     \kern-.4\dimen@ \cleaders\copy\z@\vfil \kern-.61\dimen@
    \nointerlineskip\hbox{$\m@th\bracedr$}\kern\z@}}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

```
\xydef@\rbraced{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@\R@c \B@=\Y@c \advance\B@-\D@c
  \setboxz@h{$\m@th\bracecr$}\dimen@ii=\dp\z@ \advance\A@-.5\wdz@
  \setboxz@h{$\m@th\bracec$}\dimen@=\dp\z@
  \ifdim\U@c<.5\dimen@ \U@c=.5\dimen@ \fi
  \ifdim\D@c<.5\dimen@ \advance\B@-.5\dimen@ \advance\B@\D@c \D@c=.5\dimen@ \fi
  \advance\U@c.6\p@ \advance\D@c.6\p@  \advance\B@-.6\p@
  \kern\A@\raise\B@\vbox{\vbox to\U@c{%
     \nointerlineskip\hbox{$\m@th\braceul$}%
     \kern-.61\dimen@ \cleaders\copy\z@\vfil \kern-.4\dimen@
     \nointerlineskip\hbox{$\m@th\bracecr$}\kern-.5\dimen@ii}%
    \nointerlineskip\vbox to\D@c{\kern.5\dimen@ii
     \kern-.4\dimen@ \cleaders\copy\z@\vfil \kern-.61\dimen@
    \nointerlineskip\hbox{$\m@th\bracedl$}\kern\z@}}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

An *ad hoc* correction to compensate for the 'undershoot' of top/bottom segments is added at `(1)`.

2.2g. *Parenthesis* are like braces except they have no nib and thus do not depend on where the reference point of c is.

```
\xydefcsname@{frm{(}}{\addtoDrop@@{\lparenthesized}}
\xydefcsname@{frm{)}}{\addtoDrop@@{\rparenthesized}}
```

```
\xydefcsname@{frm{^)}}{\addtoDrop@@{\uparenthesized}}
\xydefcsname@{frm{_)}}{\addtoDrop@@{\dparenthesized}}
```

The horizontal ones repeat the braces only without a nib:

```
\xydef@\dparenthesized{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@-\D@c
  \setboxz@h{$\m@th\bracelu$}\dimen@=2\wdz@ \advance\B@-.5\ht\z@
  \ifdim\R@c<\dimen@ \R@c=\dimen@ \fi
  \ifdim\L@c<\dimen@ \advance\A@-\dimen@ \advance\A@\L@c \L@c=\dimen@ \fi
  \dimen@=\L@c \advance\dimen@\R@c
  \kern\A@\raise\B@\hbox to\dimen@{%
   $\m@th\bracelu\leaders\vrule\hfil\braceru$}}%
```

```
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

```
\xydef@\uparenthesized{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@\U@c
  \setboxz@h{$\m@th\bracelu$}\dimen@=2\wdz@
  \ifdim\R@c<\dimen@ \R@c=\dimen@ \fi
  \ifdim\L@c<\dimen@ \advance\A@-\dimen@ \advance\A@\L@c \L@c=\dimen@ \fi
  \dimen@=\L@c \advance\dimen@\R@c
  \kern\A@\raise\B@\hbox to\dimen@{%
   $\m@th\braceld\leaders\vrule\hfil\bracerd$}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

The vertical ones are very like braces including the ad hoc correction.

```
\xydef@\lparenthesized{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@-\D@c
  \setboxz@h{$\m@th\bracecr$}\dimen@ii=\dp\z@ \advance\A@-.5\wdz@
  \setboxz@h{$\m@th\bracec$}\dimen@=\dp\z@
  \ifdim\U@c<.5\dimen@ \U@c=.5\dimen@ \fi
  \ifdim\D@c<.5\dimen@ \advance\B@-.5\dimen@ \advance\B@\D@c \D@c=.5\dimen@ \fi
  \advance\U@c.6\p@ \advance\D@c.6\p@  \advance\B@-.6\p@
  \dimen@ii\U@c \advance\dimen@ii\D@c
  \kern\A@\raise\B@\vbox to \dimen@ii{%
     \nointerlineskip\hbox{$\m@th\braceur$}%
     \kern-.61\dimen@ \cleaders\copy\z@\vfil \kern-.61\dimen@
    \nointerlineskip\hbox{$\m@th\bracedr$}\kern\z@}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

```
\xydef@\rparenthesized{\xy@@{\setboxz@h{%
  \A@=\X@c \advance\A@\R@c \B@=\Y@c \advance\B@-\D@c
  \setboxz@h{$\m@th\bracecr$}\dimen@ii=\dp\z@ \advance\A@-.5\wdz@
  \setboxz@h{$\m@th\bracec$}\dimen@=\dp\z@
  \ifdim\U@c<.5\dimen@ \U@c=.5\dimen@ \fi
  \ifdim\D@c<.5\dimen@ \advance\B@-.5\dimen@ \advance\B@\D@c \D@c=.5\dimen@ \fi
  \advance\U@c.6\p@ \advance\D@c.6\p@  \advance\B@-.6\p@
  \dimen@ii\U@c \advance\dimen@ii\D@c
  \kern\A@\raise\B@\vbox to \dimen@ii{%
     \nointerlineskip\hbox{$\m@th\braceul$}%
     \kern-.61\dimen@ \cleaders\copy\z@\vfil \kern-.61\dimen@
    \nointerlineskip\hbox{$\m@th\bracedl$}\kern\z@}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}}
```

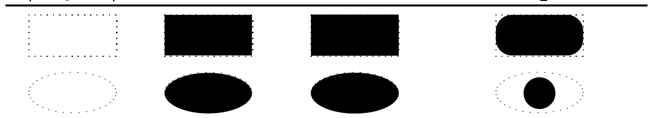
Bug: The brackets above require that the computer modern `cmex` font is loaded in TeX font position 3.

2.2.3 Filled regions

In addition to the above there is a special frame that "fills" the inside of the current object with ink: `\frm {*}` and `\frm {**}`; the latter is intended for *emphasizing* and thus "strokes" the outline, using the thinnest black line available on the printer or output device; furthermore it moits the actual filling in case this would obscure further text typeset on top. Some alteration to the shape is possible, using

`*\frm<dimen>{*}`. Hence rectangular, oval, circular and elliptical shapes can be specified for filling. The following examples illustrate this in each case:

⟨object⟩	`\frm{*}`	`\frm{**}`	`\frm<6pt>{*}`



However, filling non-rectangular shapes will result in a rectangle unless a driver is used that supports arbitrary filling. With some drivers the above fills will thus all be identical, as rectangular.

When the previous ⟨object⟩ has `\rectangleEdge` then the shape is either oval or rectangular, depending on whether a ⟨dimen⟩ has been specified, using the ⟨dimen⟩ as radius for the corners, but not exceeding half the shorter side-length.

When the current ⟨object⟩ has `\circleEdge` then the region is elliptical, using the extents `R+L` and `U+D` as the major/minor axes, or circular if a ⟨dimen⟩ is specified. The ⟨dimen⟩ will become the radius after reducing to a maximum of half the longest side of the enclosing rectangle. With `\zeroEdge` presume rectangular, or circular if a ⟨dimen⟩ is supplied.

The default for `\frm{**}` is to just make a plain frame.

```
\xydefcsname@{frm{*}}{%
 \expandafter\frmDrop@\expandafter{%
  \expandafter\def\expandafter\prevEdge@@\expandafter{\prevEdge@@}%
 \frame@fill@@\frmradius@@}}
```

```
\xydefcsname@{frm{**}}{%
 \expandafter\frmDrop@\expandafter{%
  \expandafter\def\expandafter\prevEdge@@\expandafter{\prevEdge@@}%
 \frame@emph@@\frmradius@@}}
```

```
\xydef@\frame@fill@#1{\filled@{#1}\empty}
\xydef@\frame@emph@#1{\framed@\frmradius@@}
```

```
\xylet@\frame@fill@@=\frame@fill@
\xylet@\frame@emph@@=\frame@emph@
```

```
\xydef@\filled@#1{\R@=#1\relax
 \expandafter\DNii@\expandafter{\prevEdge@@}%
 \DN@{\rectangleEdge}\ifx\next@\prevEdge@@
  \ifdim\R@=\z@ \DN@{\filled@Rectangle@}%
  \else \DN@{\filled@Oval@}\fi
 \else \DN@{\circleEdge}\ifx\next@\prevEdge@@
  \ifdim\R@=\z@ \DN@{\filled@Ellipse@}%
  \else \DN@{\restR@max\filled@Circle@}\fi
 \else
  \ifdim\R@=\z@ \DN@{\filled@Rectangle@}%
  \else \DN@{\filled@Circle@}\fi
 \fi\fi \next@}
```

```
\xydef@\restR@max{%
 \dimen@=\L@c\advance\dimen@\R@c
 \dimen@ii=\U@c \advance\dimen@ii\D@c
 \ifdim\dimen@ii>\dimen@ \dimen@=\dimen@ii\fi
 \ifdim\dimen@>\z@ \divide\dimen@\tw@
  \ifdim\R@>\dimen@\R@=\dimen@\fi\fi }
```

The default implementation uses only `\bblacked@`, in all cases. The distinction between `\frm{*}` and `\frm{**}` can be ignored.

```
\xylet@\filledRectangle@@=\bblacked@
\xylet@\filledCircle@@=\bblacked@
\xylet@\filledEllipse@@=\bblacked@
\xylet@\filledOval@@=\bblacked@
```

Alternative implementations may use these, in which the parameter `#1` is the contents of a `\vbox` filling to the height, depth and width of the region to be filled.

```
\xydef@\filled@Region@#1#2{\setboxz@h\bgroup
  \setboxz@h{\vbox{#1#2\empty}}%
  \A@=\X@c \advance\A@-.5\wd\z@
  \B@=\Y@c \advance\B@-.5\ht\z@ \advance\B@.5\dp\z@
  \edef\tmp@{\egroup\dimen@=\the\wdz@}%
  \kern\A@ \raise\B@\boxz@ \kern-\A@
 \tmp@ \divide\dimen@\tw@ \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \styledboxz@}

\xydef@\filled@Rectangle@{\filled@Region@\filledRectangle@@}
\xydef@\filled@Oval@{\filled@Region@\filledOval@@}
\xydef@\filled@Circle@{\filled@Region@\filledCircle@@}
\xydef@\filled@Ellipse@{\filled@Region@\filledEllipse@@}
```

2.2.4 Framing as object modifier

In addition, frames may be accessed using the special `[F`⟨frame⟩`]` object modifier ⟨shape⟩s that will add the desired ⟨frame⟩ to the current object. The frame appropriate to the edge of the object will be chosen (presently either rectangular or elliptical).

If shape modifiers need to be applied to the ⟨frame⟩ alone then they can be included using `:` as separator. Thus `[F-:red]` will make a red frame (provided the color extension is active, of course). Additionally the variant of frames using `<`⟨dimen⟩`>` can be accessed by specifying `[`...`:<`⟨dimen⟩`>]`.

```
\xydefcsname@{shape [F...]}#1{\xyFN@\Fshape@#1:@}

\xylet@\whichframe@@=\empty
\xylet@\whichoptions@@=\empty

\xydef@\Fshape@#1:{\def\whichframe@@{{#1}}%
 \DN@{{}}\ifx\whichframe@@\next@ \def\whichframe@@{{-}}\fi
 \expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\circleEdge}\ifx\next@\nextii@
  \DN@##1{\def\whichframe@@{[o]##1}}%
  \expandafter\next@\expandafter{\whichframe@@}\fi
 \let\whichoptions@@=\empty \xyFN@\Fshape@i}

\xydef@\Fshape@i{%
 \ifx @\next \let\next@=\Fshape@x
 \else\addLT@\ifx\next \let\next@=\Fshape@iii
 \else \let\next@=\Fshape@ii \fi\fi \next@}

\xydef@\Fshape@ii#1:{%
 \expandafter\def\expandafter\whichoptions@@\expandafter{\whichoptions@@[#1]}%
 \xyFN@\Fshape@i}

\xydef@\Fshape@iii#1:{\DN@##1{\def\whichframe@@{#1##1}}%
 \expandafter\next@\expandafter{\whichframe@@}\xyFN@\Fshape@i}
```

```
\xydef@\Fshape@x @{%
 \edef\next@##1{\noexpand\addbeforeDrop@@{{\EdgefromtheEdge@##1}}}%
 \DNii@##1##2{\next@{\saveframestyles@{\setbox\z@=\object##2\frm##1%
  \xypre@Style@@\Drop@@\xypost@Style@@\restoreframestyles@}}}%
 \expandafter\expandafter\expandafter\nextii@
  \expandafter\expandafter\expandafter{%
   \expandafter\whichframe@@\expandafter}\expandafter{\whichoptions@@}}
```

The following are used to ensure that the current styles, after typesetting, are those of the ⟨object⟩, rather than the ⟨frame⟩.

```
\xydef@\saveframestyles@{%
 \edef\afterframe@preStyle@{\preXY@style@}%
 \edef\afterframe@postStyle@{\postXY@style@}}

\xydef@\restoreframestyles@{%
 \xdef\preXY@style@{\afterframe@preStyle@}\def\afterframe@preStyle@{}%
 \xdef\postXY@style@{\afterframe@postStyle@}\def\afterframe@postStyle@{}}
```

Here are some simple examples using this feature.

text with background

bold white on black

```
\xy *+<1.5pt>[F**:white]++[F**:red]
\txt{text with background}
,+!D+/d1pc/,*++[F**:black][white]
\txt\bf{bold white on black}\endxy
```

Notice that when multiple frame-modifiers are used, the frames are actually placed in reverse order, so that earlier ones are printed on top of later ones.

To Do: The frame option is not quite complete yet: some new frames and several new brackets should be added.

2.2.5 Using curves for frames

If the curve option is loaded, then circular and elliptical frames of arbitrary radius can be constructed, by specifying `\UseCurvedFrames`. This can be negated by `\UseFontFrames`. Both of these commands obey normal TEX grouping. Furthermore, dotted and dashed frames now have a regular spacing of their constituent objects. The usual warnings about memory requirements for large numbers of curves apply here also.

Use the `\xycircle` setup in `xycurve.doc` to implement circular and elliptical frames.

```
\xydef@\xycircleframe@x@{\L@=\R@ \xycircleframe@x@i}%
\xydef@\xyellipseframe@x@#1#2{\R@=#1\relax \L@=#2\relax \xycircleframe@x@i}

\xydef@\xycircleframe@x@i{%
 \ifx\framehfill\frm@solidh@@\def\circleSTYLE@{}%
 \else\ifx\framehfill\frm@doth@@
  \expandafter\def\expandafter\circleSTYLE@\expandafter{\addDOT@{}}%
 \else\ifx\framehfill\frm@dashh@@
  \expandafter\expandafter\expandafter\def\expandafter\expandafter
   \expandafter\circleSTYLE@\expandafter\expandafter\expandafter
```

```
   {\expandafter\addDASH@\addDASH@{}}%
 \fi\fi\fi \xycircleframe@x@x }

\xydef@\xycircleframe@x@x{\hbox{\vbox{\vskip\L@
 \hbox to2\R@{\hfill \buildcircle@ \hfill}\vskip\L@ }}}

\xydef@\UseCurveFrames@{%
 \let\circled@x@@=\xycircleframe@x@
 \let\ellipsed@x@@=\xyellipseframe@x@ }
\def\UseCurveFrames{\xywithoption{curve}{\UseCurveframes@}}%

\xydef@\UseFontFrames@{%
 \let\circled@x@@=\circled@x@
 \let\ellipsed@x@@=\ellipsed@x@ }
\xylet@\UseFontFrames=\UseFontFrames@
```

Finally, the frame extension is added to the ⟨driver⟩ system:

```
\xydef@\UnloadFrames@{%
 \let\framed@body@@=\framed@body@
 \let\circled@x@@=\circled@x@
 \let\ellipsed@x@@=\ellipsed@x@
 \let\frame@fill@@=\frame@fill@
 \let\frame@emph@@=\frame@emph@
 \let\filledRectangle@@=\bblacked@
 \let\filledCircle@@=\bblacked@
 \let\filledEllipse@@=\bblacked@
 \let\filledOval@@=\bblacked@
}

\xyaddunsupport{frame}\UnloadFrames@
```

End & log

```
\xyendinput

% $Log: xyframe.doc,v $
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.2  1995/09/19  18:22:27  kris
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
```

```
%
% Revision 2.14  1995/07/05  22:11:03  kris
% Buglets...
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.10  1994/06/15  12:55:07  kris
% Second 3beta release: bug fixes.
%
% Revision 2.9  1994/06/09  14:59:19  kris
% Release 3beta.
%
% Revision 2.8  1994/04/08  04:30:00  kris
% Second (bug fix) 3alpha release.
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% NEW for version 2.7 based on frame code in xypic.doc 2.6.1.1.
```

2.3 More Tips extension

Vers. 3.3 by Kristoffer H. Rose ⟨kris@diku.dk⟩

This extension provides several additional styles of 'tips' for use (primarily) as arrow heads, and makes it possible to define customised tips. This is used to support tips that mimic the style of the Computer Modern fonts[1] by Knuth (see [8] and [6, appendix F]) and of the Euler math fonts distributed by the $\mathcal{AMS}$.

Header:

```
%% $Id: xytips.doc,v 3.3 1996/12/19 04:12:13 krisrose Exp $
%%
%% Xy-pic ‘‘More Tips’’ extension.
%% Copyright (c) 1992-1996  Kristoffer H. Rose  <kris@diku.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% Copyright (c) 1991-1996  Kristoffer H. Rose  <kris@diku.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
```

[1] This function was earlier supported by the cmtip extension which is still included in the distribution but is now obsolete.

```
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{tips}{More Tips extension}{\stripRCS$Revision: 3.3 $}%
 {Kristoffer H.~Rose}{kris@diku.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

First save the XY-commands that may need restoring.

```
\xylet@\tip@xy=\tip@
\xylet@\atip@xy=\atip@
\xylet@\btip@xy=\btip@

\xylet@\Tip@xy=\Tip@
\xylet@\Ttip@xy=\Ttip@

\xylet@\tipjot@xy=\tipjot@
```

Next declare the fonts (initially the original `cm` style).

```
\xyfont@\xy@@atfont=xycmat10
\xyfont@\xy@@btfont=xycmbt10
```

Font selection is done with the command

`\SelectTips {`⟨family⟩`} {`⟨size⟩`}`

```
\xydef@\tipfamily@@{cm}
\xydef@\tipsize@@{10}

\xydef@\SelectTips#1#2{%
 \DN@{#1}\ifx\next@\empty\else \def\tipfamily@@{#1}\fi
 \DN@{#2}\ifx\next@\empty\else \def\tipsize@@{#2}\fi
 \csname tipfamily \tipfamily@@\endcsname \ignorespaces}
```

where the ⟨family⟩ and ⟨size⟩ should be selected from the following table.

Family	10	11	12
`xy`	$\to \Rightarrow \Rrightarrow$	$\to \Rightarrow \Rrightarrow$	$\to \Rightarrow \Rrightarrow$
`cm`	$\to \Rightarrow \Rrightarrow$	$\to \Rightarrow \Rrightarrow$	$\to \Rightarrow \Rrightarrow$
`eu`	$\to \Rightarrow \Rrightarrow$	$\to \Rightarrow \Rrightarrow$	$\to \Rightarrow \Rrightarrow$

The table is really encoded in control sequences `\tipfamily` ⟨family⟩ that look very much alike since all the currently existing combinations are merely alternative fonts. This is likely to change, however, ... ☺

First cm, *i.e.*, computer modern: these are simplified by the fact that we can use the normal double arrows.

```
\xydefcsname@{tipfamily cm}{\loadtipfonts@
 \let\tip@=\tip@cm \let\atip@=\atip@cm \let\btip@=\btip@cm
 \let\Tip@=\Tip@xy \let\Ttip@=\Ttip@xy \def\tipjot@{.2em}}

\xydef@\loadtipfonts@{\edef\next##1##2{##1=xy\tipfamily@@##2\tipsize@@}%
 \next{\font\xy@@atfont}{at}
 \next{\font\xy@@btfont}{bt}
}

\xydef@\tip@cm{\activatetips@ \tip@x\tip@@}
\xydef@\atip@cm{\activatetips@ \tip@x\atip@@}
\xydef@\btip@cm{\activatetips@ \tip@x\btip@@}

\xydef@\activatetips@{\let\xyatipfont=\xy@@atfont \let\xybtipfont=\xy@@btfont}
```

Next Euler tips: the single tips merely reuse the cm code except for the sligtly smaller distance between double tips; the double arrows have to be recoded to mimic the steeper Euler appearance.

```
\xydefcsname@{tipfamily eu}{\loadtipfonts@
 \let\tip@=\tip@cm \let\atip@=\atip@cm \let\btip@=\btip@cm
 \let\Tip@=\Tip@eu \let\Ttip@=\Ttip@eu \def\tipjot@{.15em}}

\xydef@\Tip@eu{\kern1.5pt \vrule height1.5pt depth1.5pt width\z@
 \Tip@@eu \kern1.5pt \egroup
 \U@c=1.5pt \D@c=1.5pt \L@c=1.5pt \R@c=1.5pt \Edge@c={\circleEdge}%
 \Invisible@false \Hidden@false \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@{\dottedSpread@\jot}}}

\xydef@\Tip@@eu{\count@=\DirectionChar
 \advance\count@-10 \ifnum\count@<\z@ \advance\count@128 \fi
 \xyatipfont\char\count@
 \advance\count@ 20 \ifnum127<\count@ \advance\count@-128 \fi
 \xybtipfont\char\count@}

\xydef@\Ttip@eu{\kern2.4pt \vrule height2.4pt depth2.4pt width\z@
 \Ttip@@eu \kern2.4pt \egroup
 \U@c=2.4pt \D@c=2.4pt \L@c=2.4pt \R@c=2.4pt \Edge@c={\circleEdge}%
 \Invisible@false \Hidden@false
 \def\Leftness@{.5}\def\Upness@{.5}%
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@{\dottedSpread@\jot}}}

\xydef@\Ttip@@eu{%
 \setboxz@h\bgroup\reverseDirection@\line@ \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@
 \kern-\L@c \boxz@ \kern\L@c \Tip@@eu}
```

Once a selection is made, the following commands are available:

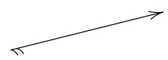
`\UseTips`	activate selected tips
`\NoTips`	deactivate

```
\xydef@\UseTips{\csname tipfamily \tipfamily@@\endcsname}

\xydef@\NoTips{\let\tip@=\tip@xy \let\atip@=\atip@xy \let\btip@=\btip@xy
 \let\Tip@=\Tip@xy \let\Ttip@=\Ttip@xy \let\tipjot@=\tipjot@xy}
```

They are local and thus can be switched on and/or off for individual pictures using the TEX grouping mechanism, *e.g.*,

```
\SelectTips{cm}{10}
\xy*{} \ar
 @{*{\UseTips\dir_{<<}}-*{\NoTips\dir{>}}}
 (20,5)*{} \endxy
```

will typeset



regardless of which tips are used otherwise in the document.

2.3.1 End & log

```
\xyendinput

% $Log: xytips.doc,v $
% Revision 3.3  1996/12/19 04:12:13  krisrose
% New for this maintenance release.
%
% Generalisation of xycmtip.doc,v 3.1 1995/09/05 20:31:32 kris Exp krisrose
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.6  1993/10/21  20:36:09  kris
% NEW file to go in version 2.7!
%
% Extracted from xypic.doc 2.6.1.1.
```

2.4 Line styles extension

Vers. 3.6 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This extension provides the ability to request various effects related to the appearance of straight lines; *e.g.*. thickness, non-standard dashing, and colour.

Header:

```
%% $Id: xyline.doc,v 3.6 1998/03/06 01:28:05 krisrose Exp $
%%
%% Xy-pic ``Line Width'' extension.
%% Copyright (c) 1993-1996 Ross Moore <ross@mpce.mq.edu.au>
```

```
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{line}{Line styles extension}{\stripRCS$Revision: 3.6 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

These are effects which are not normally available within TEX. Instead they require a suitable 'back-end' option to provide the necessary `\special` commands, or extra fonts, together with appropriate commands to implement the effects. Thus

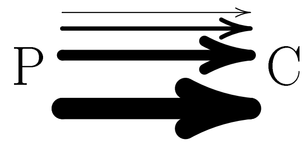
Using this extension will have no effect on the output unless used with a backend that explicitly supports it.

The extension provides special effects that can be used with any XY-pic ⟨object⟩, by defining `[`⟨shape⟩`]` modifiers. The modification is local to the ⟨object⟩ currently being built, so will have no effect if this object is never actually used.

Adjusting line thickness The following table lists the modifiers primarily to alter the thickness of lines used by XY-pic. They come in two types — either a single keyword, or using the key-character `|` with the following text parsed.

`[thicker]`	double line thickness
`[thinner]`	halve line thickness
`[|(`⟨num⟩`)]`	multiple of usual thickness
`[|<`⟨dimen⟩`>]`	set thickness to ⟨dimen⟩
`[|`⟨dimen⟩`]`	also sets to ⟨dimen⟩
`[|=`⟨word⟩`]`	make `[`⟨word⟩`]` set current style settings
`[|*]`	reuse previous style
`[butt]`	butt cap at ends
`[roundcap]`	round cap at ends
`[projcap]`	projecting square cap.

Later settings of the linewidth override earlier settings; multiple calls to `[thicker]` and `[thinner]` compound, but the other variants set an absolute thickness. The line-thickness specification affects arrow-tips as well as the thickness of straight lines and curves. Three kinds of line-caps are available; they are discussed below in the section on 'poly-lines'.



```
\xy/r8pc/:*++\txt\huge{C}="c"
,0*++\txt\huge{P}="p",
,"p",{\ar@*{[|(1)]}"p";"c"<20pt>}
,"p",{\ar@*{[|(4)]}"p";"c"<14pt>}
,"p",{\ar@*{[|(10)]}"p";"c"<4pt>}
,"p",{\ar@*{[|(20)]}"p";"c"<-16pt>}
\endxy
```

Using the PostScript back-end, the size of the arrow-head grows aesthetically with the thickness of the line used to draw it. This growth varies as the square-root of the thickness; thus for very thick lines (20+ times normal) the arrowhead begins to merge with the stem.

Load the `style` extension to establish the necessary infra-structure.

```
%\xydefcsname@{shape [|...]}#1{\xyFN@\xywidthchar@#1@@}%
\xydefcsname@{*stylechar@|@}#1{\xyFN@\xywidthchar@#1@@}%
```

```
%\xydefcsname@{shape [thinner]}{\xyshape@thinner@}
%\xydefcsname@{shape [thicker]}{\xyshape@thicker@}
\xydefcsname@{style [thinner]}{\xyshape@thinner@}
\xydefcsname@{style [thicker]}{\xyshape@thicker@}
```

Implementation Record the line thickness locally in a dimen register. The initial value is read from the `xydash10` font. Each time `\xylinethick@` is changed, its previous value is stored as the expansion of `\xyprevwidth@@`, in case this is needed by a specific back-end.

```
\xynew@{dimen}\xylinethick@ \xylinethick@=\xydashw@
```

```
\xydef@\xyshape@thicker@{\xylinewidth@i{\multiply\xylinethick@\tw@}}
\xydef@\xyshape@thinner@{\xylinewidth@i{\divide\xylinethick@\tw@}}
```

```
\xydef@\xylinewidth@i#1{\edef\xyprevwidth@@{\the\xylinethick@}%
 #1\edef\next@{\xylinethick@=\the\xylinethick@}%
 \edef\next@ii{\noexpand\addtotoks@{%
  \noexpand\xylinewidth@@{\next@}{\xyprevwidth@@}}}%
 \next@ii }
```

```
\xydef@\xywidthchar@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xywidthchar@}%gobble spaces
% \else\ifx @\next \DN@ @@{\addtotoks@{\checkXyStyle@}}%
 \else\ifx @\next \DN@ @@{\checkXyStyle@}%
 \else\ifx (\next \DN@ (##1)##2@@{%
  \xylinewidth@i{\xylinethick@=##1\xydashw@}}%
 \else\addLT@\ifx \next \addGT@{\addLT@\DN@ ##1}##2@@{%
  \xylinewidth@i{\xylinethick@=##1\relax}}%
```

```
% \else\ifx *\next \DN@ *@@{\addtotoks@{\checkXyStyle@}}%
% \else\ifx *\next \DN@ *@@{\checkXyStyle@}%
 \else\ifx *\next \DN@ *@@{\recoverXyStyle@}%
% \else\addEQ@\ifx \next \addEQ@\DN@##1@@{%
%  \addtotoks@{\checkXyStyle@\xynamestyle@{##1}}}%
 \else\addEQ@\ifx \next \addEQ@\DN@##1@@{\checkXyStyle@
  \addtotoks@{\xynamestyle@{##1}}}%
 \else\ifx C\next \DN@ C##1@@{\xysetlinecap@{##1}@}%
 \else\ifx J\next \DN@ J##1@@{\xysetlinejoin@{##1}@}%
 \else\ifx M\next \DN@ M(##1)@@{\xymiterlimit@{##1}@}%
 \else \DN@##1@@{\xylinewidth@i{\xylinethick@=##1\relax}}%
 \fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}
```

The diagram in figure 2.4, page 233, uses different line-thicknesses and colours.

Poly-lines By a 'poly-line' we mean a path built from straight line segments having no gaps where each segment abuts the next. The poly-line could be the edges of a polygon, either closed or open if the end-points are different.

The reason for considering a poly-line as a separate ⟨object⟩, rather than simply as a ⟨path⟩ built from straight lines, becomes apparent only when the lines have appreciable thickness. Then there are several standard ways to fashion the 'joins' (where segments meet). Also the shape of the 'caps' at either end of the poly-line can be altered.

The following modifiers are used to determine the shapes of the line 'caps' and 'joins':

`[|J`⟨val⟩`]`	join style, ⟨val⟩ = 0, 1 or 2
`[mitre]`	mitre-join, same as `[|J0]`
`[roundjoin]`	round join, same as `[|J1]`
`[bevel]`	bevel-join, same as `[|J2]`
`[|C`⟨val⟩`]`	end-cap, ⟨val⟩ = 0, 1 or 2
`[butt]`	"butt" cap, same as `[|C0]`
`[roundcap]`	round cap, same as `[|C1]`
`[projcap]`	"projecting square" cap, same as `[|C2]`
`[|M(`⟨num⟩`)]`	set mitrelimit to ⟨num⟩≥ 1

These effects are currently implemented only with the PostScript back-end or when using `\xypolyline` (described below) with a PostScript ⟨driver⟩. In this case the 'cap' setting can be applied to any segment, straight or curved, whether part of a poly-line or not; however the 'join' setting applies only to poly-lines. Arrow-tips are not affected. The defaults are to use round joins and round-cap ends.

Adjusting the miter-limit affects how far miters are allowed to protrude when two wide lines meet at small angles. The ⟨num⟩ is in units of the line-thickness. Higher values mean using bevels only at smaller angles, while the value of 1 is equivalent to using bevels at all angles. The default miter-limit is 10.

Current values are stored in case these are needed within nested portions of diagrams.

```
\xydef@\xylinecap@{2}
\xydef@\xylinejoin@{2}
\xydef@\xylinemiter@{10}
```

Recognise keywords for 'line-caps' and 'line-joins':

```
\xydefcsname@{style [bevel]}{\xyshape@bevel@}
\xydefcsname@{style [roundjoin]}{\xyshape@rdjoin@}
```

```
\xydefcsname@{style [miter]}{\xyshape@miter@}
%
\xydefcsname@{style [butt]}{\xyshape@butt@}
\xydefcsname@{style [roundcap]}{\xyshape@rdcap@}
\xydefcsname@{style [projcap]}{\xyshape@projcap@}

\xydef@\xyshape@butt@{\xysetlinecap@@0}
\xydef@\xyshape@rdcap@{\xysetlinecap@@1}
\xydef@\xyshape@projcap@{\xysetlinecap@@2}

\xydef@\xyshape@miter@{\xysetlinejoin@@0}
\xydef@\xyshape@rdjoin@{\xysetlinejoin@@1}
\xydef@\xyshape@bevel@{\xysetlinejoin@@2}

\xydef@\xysetlinecap@#1@{\DN@ii{#1}\count@=#1\relax
 \expandafter\DN@\expandafter{\the\count@}%
 \ifx\next@\next@ii\else
  \xywarning@{ignoring extra tokens in line-cap: \next@}\fi
 \expandafter\xysetlinecap@@\expandafter{\next@}}

\xydef@\xysetlinejoin@#1@{\DN@ii{#1}\count@=#1\relax
 \expandafter\DN@\expandafter{\the\count@}%
 \ifx\next@\next@ii\else
  \xywarning@{ignoring extra tokens in line-join: \next@}\fi
 \expandafter\xysetlinejoin@@\expandafter{\next@}}

\xydef@\xysetmiterlimit@#1@{\DN@ii{#1}\dimen@=#1\p@\relax
 \edef\next@{\expandafter\removePT@\the\dimen@}%
 \ifx\next@\next@ii\else
  \xywarning@{ignoring extra tokens in miter-limit: \next@}\fi
 \expandafter\xysetmiterlimit@@\expandafter{\next@}}
```

These use the more sophisticated approach that is easier to adjust for different ⟨driver⟩s.

```
\xydef@\xysetlinecap@@#1{\xy@linecap@i{\def\xylinecap@{#1}}}
\xydef@\xysetlinejoin@@#1{\xy@linejoin@i{\def\xylinejoin@{#1}}}
\xydef@\xysetmiterlimit@@#1{%
 \xy@linemiter@i{\def\xylinemiter@{#1}}}

\xydef@\xy@linecap@i#1{#1\addtotoks@{\xy@linecap@@{#1}}}
\xydef@\xy@linejoin@i#1{#1\addtotoks@{\xy@linejoin@{#1}}}
\xydef@\xy@linemiter@i#1{#1\addtotoks@{\xy@linemiter@{#1}}}
```

Implementation of line-caps:

```
\xydef@\xy@linecap@#1{\checkXyStyle@ \edef\xyprevcap@@{\xylinecap@}%
 #1\let\prexylinecap@=\prexylinecap@@ \let\postxylinecap@=\postxylinecap@@
 \applyLIFOstyle@\prexylinecap@{\xylinecap@}%
  \postxylinecap@{\xyprevcap@@}}

\xydef@\prexylinecap@@{\let\prexylinecap@=\eat@
 \let\postxylinecap@=\postxylinecap@@ \outputxylinecap@ }
\xydef@\postxylinecap@@{\let\postxylinecap@=\eat@
 \let\prexylinecap@=\prexylinecap@@ \resetxylinecap@@ }
\xylet@\prexylinecap@=\prexylinecap@@
\xylet@\postxylinecap@=\postxylinecap@@
```

```
\xydef@\outputxylinecap@#1{\xycapSpecial@@{#1}}
\xydef@\resetxylinecap@#1{\resetxylinecap@i{#1}\outputxylinecap@{#1}}
\xydef@\resetxylinecap@i#1{\def\xylinecap@{#1}}
\xylet@\resetxylinecap@@=\resetxylinecap@
```

```
\xylet@\xycapSpecial@@=\eat@
```

Implementation of line-joins:

```
\xydef@\xy@linejoin@#1{\checkXyStyle@
 \edef\xyprevjoin@@{\xylinejoin@}#1%
 \let\prexylinejoin@=\prexylinejoin@@
 \let\postxylinejoin@=\postxylinejoin@@
 \applyLIFOstyle@\prexylinejoin@{\xylinejoin@}%
  \postxylinecap@{\xyprevjoin@@}}
```

```
\xydef@\prexylinejoin@@{\let\prexylinejoin@=\eat@
 \let\postxylinejoin@=\postxylinejoin@@ \outputxylinejoin@ }
\xydef@\postxylinejoin@@{\let\postxylinejoin@=\eat@
 \let\prexylinejoin@=\prexylinejoin@@ \resetxylinejoin@@ }
\xylet@\prexylinejoin@=\prexylinejoin@@
\xylet@\postxylinejoin@=\postxylinejoin@@
```

```
\xydef@\outputxylinejoin@#1{\xyjoinSpecial@@{#1}}
\xydef@\resetxylinejoin@#1{\resetxylinejoin@i{#1}\outputxylinejoin@{#1}}
\xydef@\resetxylinejoin@i#1{\def\xylinejoin@{#1}}
\xylet@\resetxylinejoin@@=\resetxylinejoin@
```

```
\xylet@\xyjoinSpecial@@=\eat@
```

Implementation of miter-limit:

```
\xydef@\xy@linemiter@#1{\checkXyStyle@
 \edef\xyprevmiter@@{\xylinemiter@}#1%
 \let\prexylinemiter@=\prexylinemiter@@
 \let\postxylinemiter@=\postxylinemiter@@
 \applyLIFOstyle@\prexylinemiter@{\xylinemiter@}%
  \postxylinecap@{\xyprevmiter@@}}
```

```
\xydef@\prexylinemiter@@{\let\prexylinemiter@=\eat@
 \let\postxylinemiter@=\postxylinemiter@@ \outputxylinemiter@ }
\xydef@\postxylinemiter@@{\let\postxylinemiter@=\eat@
 \let\prexylinemiter@=\prexylinemiter@@ \resetxylinemiter@@ }
\xylet@\prexylinemiter@=\prexylinemiter@@
\xylet@\postxylinemiter@=\postxylinemiter@@
```

```
\xydef@\outputxylinemiter@#1{\xymiterSpecial@@{#1}}
\xydef@\resetxylinemiter@#1{\resetxylinemiter@i{#1}\outputxylinemiter@{#1}}
\xydef@\resetxylinemiter@i#1{\def\xylinemiter@{#1}}
\xylet@\resetxylinemiter@@=\resetxylinemiter@
```

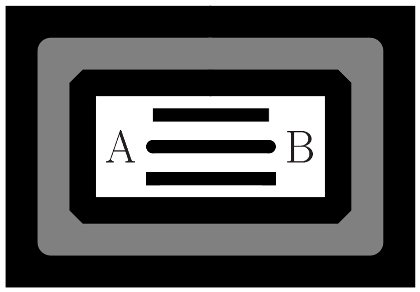
```
\xylet@\xymiterSpecial@@=\eat@
```

The path taken by the 'poly-line' this is read as the list of ⟨pos⟩itions in the current 'stack', ignoring size extents. The macro **\xypolyline** is used as a ⟨decor⟩; it reads the ⟨pos⟩itions from the stack, but leaves the stack intact for later use.

Implementation of 'poly-lines'.

```
\xydef@\xypolyline#1{\xypolyline@{#1}}

\xydef@\xypolyline@#1{\hbox\bgroup
 \enter@{\pfromthep@ \basefromthebase@ \cfromthec@}%
  \def\xyps@list{}\edef\smapp@@{\s@bot}%
  \smapxy@i{\enter@{\pfromthep@ \basefromthebase@ }%
   \edef\xyps@list{\expandafter\removePT@\the\X@c\space
    \expandafter\removePT@\the\Y@c\space \xyps@list}\leave@}%
 \leave@ \xy@polystyle@@{#1}\xydopoly@style\egroup
 \def\Drop@@{\styledboxz@}\def\Connect@@{\styledboxz@}}%
```

The following diagram illustrates the use of line-thickness, line-joins and line-caps with poly-lines. It contains an example of each of the styles.




```
\xycompileto{poly}%
{/r4pc/:,*[|<5pt>][thicker]\xybox{%
 *+(3,2){}="X"
;@={p+CU,p+LU,p+LD,p+RD,p+RU,p+CU}
 ,{0*[miter]\xypolyline{}}
 ,{\xypolyline{*}},@i@)
,"X",*+(2.5,1.5){}="X"
,@={!CU,!LU,!LD,!RD,!RU,!CU}
 ,{0*[gray][roundjoin]\xypolyline{}}
 ,{0*[gray]\xypolyline{*}},@i@)
,"X",*+(2,1){}="X"
,@={!CU,!LU,!LD,!RD,!RU,!CU}
 ,{0*[white]\xypolyline{*}}
 ,{0*[bevel]\xypolyline{}},@i@)
,"X"-(.7,0)*++\txt\LARGE{A}="a"
,"X"+(.7,0)*++\txt\LARGE{B}="b"
,{\ar@{-}@*{[butt][thinner]}"a";"b"<1pc>}
,{\ar@{-}@*{[roundcap][thinner]}"a";"b"}
,{\ar@{-}@*{[projcap][thinner]}"a";"b"<-1pc>}
}}
```

Note the use of `{0*[...]\xypolyline{..}}` to apply style-modifiers to a polyline. The `@={!..}` method for loading the stack gives equivalent results to using `;@={p+..}`, since `\xypolyline` ignores the edge extents of each ⟨pos⟩ in the stack.

Note also that the argument `#1` to `\xypolyline` affects what is typeset. Allowable arguments are:

`\xypolyline{}`	solid line
`\xypolyline{.}`	dotted line
`\xypolyline{-}`	dashed line
`\xypolyline{*}`	fill enclosed polygon

\xypolyline{?}	fill enclosed polygon using even-odd rule
\xypolyline{{*}}	use \dir{*} for lines
\xypolyline{<toks>}	using \dir{<toks>}

The latter cases one has **\dir{...} being used to connect the vertices of the polyline, with {{*}} being needed to get **\dir{*}. Similarly **\dir is used when a ⟨driver⟩ is not available to specifically support polylines; in particular the two 'fill' options * and ? will result in a dotted polygon outline the region intended to be filled.

In all cases it is up to the user to load the stack before calling \xypolyline{...}. A particularly common case is the outline of an existing XY-pic ⟨object⟩, as in the example above. Future extensions to \frm will provide a simplified mechanism whereby the user need not call \xypolyline explicitly for such effects.

```
\xylet@\xypolyline@Special=\eat@
\xylet@\xypolyfill@Special=\eat@
\xylet@\xypolyeofill@Special=\eat@
\xylet@\xypolydot@Special=\eat@
\xylet@\xypolydash@Special=\eat@

\xydef@\UnloadpolySpecials@{%
 \let\xypolyline@Special=\eat@
 \let\xypolyfill@Special=\eat@
 \let\xypolyeofill@Special=\eat@
 \let\xypolydot@Special=\eat@
 \let\xypolydash@Special=\eat@
}

\xydef@\xy@polystyle@#1{%
 \let\poly@style@@=\xypolyline@Special
 \DNii@{#1}\ifx\nextii@\empty
 \else\DN@{*}\ifx\next@\nextii@ \let\poly@style@@=\xypolyfill@Special
 \else\DN@{?}\ifx\next@\nextii@ \let\poly@style@@=\xypolyeofill@Special
 \else\expandafter\DN@\expandafter{\addDOT@{}}\ifx\next@\nextii@
  \let\poly@style@@=\xypolydot@Special
 \else\expandafter\DN@\expandafter{\addDASH@{}}\ifx\next@\nextii@
  \let\poly@style@@=\xypolydash@Special
 \else\DN@{{*}}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{*}}%
 \else \def\poly@style@@{\xypolystraight@{#1}}%
 \fi\fi\fi\fi\fi\fi }

\xydef@\xydopoly@style{\expandafter\poly@style@@\expandafter{\xyps@list}}
\xydef@\xynopolystyle@#1#2{\xyundefinedLine@{polyline}{{#1}}@@}
```

When there is no special support this handles the various cases using ordinary thin line types.

```
\xydef@\xypolynospec@#1{\DNii@{#1}%
 \ifx\nextii@\empty\def\poly@style@@{\xypolystraight@{-}}%
 \else\DN@{.}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}%
 \else\DN@{-}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{--}}%
 \else\DN@{*}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}%
 \else\DN@{?}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}%
 \else\DN@{{*}}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{*}}%
 \else \def\poly@style@@{\xypolystraight@{#1}}%
 \fi\fi\fi\fi\fi\fi }
```

Gobble the collected positions in `{\xyps@list}`; recover the vertices from the stack, then set straight lines in the appropriate style.

```
\xydef@\xypolystraight@#1#2{%
 \enter@{\pfromthep@ \basefromthebase@ \cfromthec@}%
 \edef\smapp@@{\s@bot}\csname S@0\endcsname\czeroEdge@\pfromc@\def\smapp@@{0}%
 \smapxy@i{\czeroEdge@\enter@{\pfromthec@}{\connect@\dir{#1}}\leave@}\leave@ }
```

Initially poly-lines are uninstalled, until loaded by a ⟨driver⟩.

```
\xylet@\xy@polystyle@@=\xypolynospec@
```

```
\xydef@\UninstallPolylines@{%
 \UnloadpolySpecials@
 \let\xy@polystyle@@=\xypolynospec@ }
\UninstallPolylines@
```

Installation Initially there is no support for the actual effects.

```
\xydef@\xyundefinedLine@#1#2@@{%
 \xywarning@{The #1 #2 effect is not implemented with the current driver.}}%
```

```
\xydef@\xynolinewidth@#1#2{%
 \xyundefinedLine@{linewidth}{\the\xylinethick@}@@}%
\xydef@\xynolinecap@#1{\xyundefinedLine@{linecap}{\xylinecap@}@@}%
\xydef@\xynolinejoin@#1{\xyundefinedLine@{linejoin}{\xylinejoin@}@@}%
\xydef@\xynolinemiter@#1{\xyundefinedLine@{miterlimit}{\xylinemiter@}@@}%
```

Nothing further can be done unless allowed by a specific ⟨driver⟩, which must establish an alternative expansions to be bound to `\xylinewidth@@` and the other hooks.

```
\xylet@\xylinewidth@@=\xynolinewidth@
\xylet@\xy@linecap@@=\xynolinecap@
\xylet@\xy@linejoin@@=\xynolinejoin@
\xylet@\xy@linemiter@@=\xynolinemiter@
```

```
\xydef@\UnloadLine@#1{%
 \ifx#1\relax\else\xyunload@{line width extension}\fi
 \let\resetxyline@@=\resetxyline@
 \let\resetxylinecap@@=\resetxylinecap@
 \let\resetxylinejoin@@=\resetxylinejoin@
 \let\resetxylinemiter@@=\resetxylinemiter@
 \let\transxyline@@=\DN@
 \let\xylineSpecial@@=\eat@
 \let\xycapSpecial@@=\eat@
 \let\xyjoinSpecial@@=\eat@
 \let\xymiterSpecial@@=\eat@
 \let\xylinewidth@@=\xynolinewidth@
 \let\xy@linecap@@=\xynolinecap@
 \let\xy@linejoin@@=\xynolinejoin@
 \let\xy@linemiter@@=\xynolinemiter@
 \UninstallPolylines@ }
```

```
\xyaddunsupport{line}{\UnloadLine@\relax}
```

This is a possible alternative for `\xylinewidth@@`. Indeed it is used with the PostScript support, which then redefines just \xylineSpecial@@.

```
\xydef@\xylinewidth@#1#2{\checkXyStyle@
 #1\let\prexyline@=\prexyline@@ \let\postxyline@=\postxyline@@
 \applyLIFOstyle@\prexyline@{\the\xylinethick@}\postxyline@{#2}}
\xydef@\prexyline@@{\let\prexyline@=\eat@
 \let\postxyline@=\postxyline@@ \outputxyline@ }
\xydef@\postxyline@@{\let\postxyline@=\eat@
 \let\prexyline@=\prexyline@@ \resetxyline@@ }
\xylet@\prexyline@=\prexyline@@
\xylet@\postxyline@=\postxyline@@

\xydef@\resetxyline@#1{\resetxyline@i{#1}\outputxyline@{#1}}
\xydef@\resetxyline@i#1{\xylinethick@=#1\relax}
\xydef@\outputxyline@#1{\transxyline@@{#1}%
 \expandafter\xylineSpecial@@\expandafter{\next@}}
\xylet@\resetxyline@@=\resetxyline@
```

These need to be rebound by a back-end which supports variation in the line-thickness.

```
\xydef@\transxyline@#1{\edef\next@{\removePT@#1}}

\xylet@\transxyline@@=\DN@
\xylet@\xylineSpecial@@=\eat@
```

The end & Log

```
\xyendinput

% $Log: xyline.doc,v $
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.3  1996/12/18  09:22:29  ross
% adjustments for styles now within the kernel code
% fixed problem with line-widths
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.12  1994/10/25  03:01:14  ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
%
% Revision 2.11  1994/07/05  09:27:49  ross
```

```
% fixed documentation bug
%
% Revision 2.10  1994/06/15  12:46:03  ross
% Second release 3beta.
% Colour and line style saving works; label colouring bug fixed.
%
% Revision 2.9  1994/06/09  14:39:49  ross
% Release 3beta.
%
% NEW for version 2.9 by Ross Moore.
```

2.5 Rotate and Scale extension

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This extension provides the ability to request that any object be displayed rotated at any angle as well as scaled in various ways.

Header:

```
%% $Id: xyrotate.doc,v 3.3 1996/12/18 09:40:16 ross Exp $
%%
%% Xy-pic ``Rotate and Scale extension'' option.
%% Copyright (c) 1993-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{rotate}{Rotate and Scale extension}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

These are effects which are not normally available within TEX. Instead they require a suitable 'back-end' option to provide the necessary `\special` commands, or extra fonts, together with appropriate commands to implement the effects. Thus

Using this extension will have no effect on the output unless used with a backend that explicitly supports it.

```
\xydef@\xyundefinedRotate@#1#2@@{%\egroup %
 \xywarning@{The #1 #2 effect is not implemented with the current driver.}}%
```

The extension provides special effects that can be used with any XY-pic ⟨object⟩ by defining `[`⟨shape⟩`]` modifiers. The modification is local to the ⟨object⟩ currently being built, so will have no effect if this object is never actually used.

The following table lists the modifiers that have so far been defined. They come in two types – either a single keyword, or a key-character with the following text treated as a single argument.

`[@]`	align with current direction
`[@`⟨direction⟩`]`	align to ⟨direction⟩
`[@!`⟨number⟩`]`	rotate ⟨number⟩ degrees
`[*`⟨number⟩`]`	scale by ⟨number⟩
`[*`⟨num⟩$_x$`,`⟨num⟩$_y$`]`	scale x and y separately
`[left]`	rotate anticlockwise by 90°
`[right]`	rotate (clockwise) by 90°
`[flip]`	rotate by 180°; same as `[*-1,-1]`
`[dblsize]`	scale to double size
`[halfsize]`	scale to half size

These `[`⟨shape⟩`]` modifiers specify transformations of the ⟨object⟩ currently being built. If the object has a rectangle edge then the size of the rectangle is transformed to enclose the transformed object; with a circle edge the radius is altered appropriately.

To Do: There should be an `[@*]` form which repeats the set of transformations applied to the last object which has been transformed, possibly in an earlier diagram. Also an `[@=`⟨word⟩`]` form allowing a set of transformations to be saved and used later, simply by specifying `[`⟨word⟩`]`.

Each successive transformation acts upon the result of all previous. One consequence of this is that the order of the shape modifiers can make a significant difference in appearance—in general, transformations do not commute. Even successive rotations can give different sized rectangles if taken in the reverse order.

Sometimes this change of size is not desirable. The following commands are provided to modify this behaviour.

`\NoResizing`	prevents size adjustment
`\UseResizing`	restores size adjustments

```
\xydef@\NoResizing{\xyNoResizing@}
\xydef@\UseResizing{\xyUseResizing@}

\xydef@\xyNoResizing@{%
 \global\let\origxyrescale@=\xyrescale@
 \global\let\origxyrotSizeMod@=\xyrotateSizeMod@
 \global\let\xyNoResizing@=\xyNoResizing@@
```

```
 \global\let\xyUseResizing@=\xyUseResizing@@
 \xyNoResizing@@ }
```

```
\xydef@\xyNoResizing@@{%
 \global\let\xyrescale@=\xyignorescale@
 \global\let\xyrotateSizeMod@=\xyignoreSizeMod@
 }
```

```
\xylet@\xyUseResizing@=\relax
\xydef@\xyUseResizing@@{%
 \global\let\xyrescale@=\origxyrescale@
 \global\let\xyrotateSizeMod@=\origxyrotSizeMod@
 }
```

The `\NoResizing` command is also useful to have at the beginning of a document being typeset using a driver that cannot support scaling effects, in particular when applied to whole diagrams. In any case an unscaled version will result, but now the spacing and positioning will be appropriate to the unscaled rather than the scaled size.

Interface Recognize the ⟨shape⟩ characters and keywords.

```
\xydefcsname@{shape [@...]}#1{\xyrotatechar@#1@@}%
\xydefcsname@{shape [<...]}#1{\xyshearchar@{#1}@@}%
\xydefcsname@{shape [*...]}#1{\xyscalechar@{#1}@@}%
```

```
\xydefcsname@{shape [left]}{\xyshape@left@}
\xydefcsname@{shape [right]}{\xyshape@right@}
\xydefcsname@{shape [flip]}{\xyshape@flip@}
\xydefcsname@{shape [halfsize]}{\xyshape@halfsize@}
\xydefcsname@{shape [dblsize]}{\xyshape@dblsize@}
\xydefcsname@{shape [vflip]}{\xyshape@vflip@}
\xydefcsname@{shape [hflip]}{\xyshape@hflip@}
```

Each style modifier adds code to one, or both, of two global lists. These contents of these lists, called `\preXYtransform@@` and `\postXYtransform@@`. An implementation must define macros which are to be bound to the control sequence names `\preXYtransform@` and `\postXYtransform@`, which are prepended and appended to the `\Drop@@` and `Connect@@` methods for the current ⟨object⟩.

Also `\Drop@@` and `\Connect@@` must be modified, if this has not already been done as indicated by `\xy@transform@` having expansion `\relax`.

```
\xydef@\preXYtransform@{}
\xydef@\postXYtransform@{}
```

```
\xywarnifdefined\preXYtransform@@
\xywarnifdefined\postXYtransform@@
 \gdef\preXYtransform@@{}
 \gdef\postXYtransform@@{}
\xydef@\xy@transform@{}
\xydef@\preShape@@{}
\xydef@\postShape@@{}
```

The transformations must be implemented by emulating the adjoint coordinate transformations. This requires that code be added to `preXYtransform@@` in reverse order to the order of occurrence of the ⟨shape⟩ modifiers. The code is added to `\postXYtransform@@` in natural order, so that each addition to `\preXYtransform@@` can be closed off, if necessary, in a correctly nested sequence.

```
\xydef@\modXYtransform@{%
 \ifx\xy@transform@\empty
  \transformDrop@\transformConnect@\resetTransform@\fi
%\show\preShape@@
 \DN@##1{\expandafter\def\expandafter\tmp@\expandafter{##1}}%
  \expandafter\next@\expandafter{\preShape@@}%
  \DN@##1{\expandafter\gdef\expandafter\preXYtransform@@\expandafter{%
   \tmp@ ##1}}\expandafter\next@\expandafter{\preXYtransform@@}%
 \DN@##1{\expandafter\def\expandafter\tmp@\expandafter{##1}}%
  \expandafter\next@\expandafter{\postShape@@}%
  \DN@##1{\expandafter\gdef\expandafter\postXYtransform@@\expandafter{%
   \postXYtransform@@ ##1}}\expandafter\next@\expandafter{\tmp@}}%
```

```
\xydef@\transformDrop@{\let\xy@transform@=\relax
 \expandafter\def\expandafter\Drop@@\expandafter{%
 \expandafter\preXYtransform@\Drop@@\postXYtransform@}}
```

```
\xydef@\transformConnect@{\let\xy@transform@=\relax
 \expandafter\def\expandafter\Connect@@\expandafter{%
 \expandafter\preXYtransform@\Connect@@\postXYtransform@}}
```

```
\xydef@\resetTransform@{\gdef\preXYtransform@@{}\gdef\postXYtransform@@{}}
```

Bug: The reference point does not move correctly in PostScript when a shift modifier has been applied before a rotation. The object prints outside of its bounding `\rectangleEdge`. Fix this!!

Global macros are used, so that the same styles can be reused by successive objects without having to re-interpret ⟨shape⟩ modifiers, as described next.

Saving transformations Once specified for an ⟨object⟩, the collection of transformations can be assigned a name, via `[>`⟨name⟩`]`. The ⟨name⟩ here is any collection of letters or other characters up to the closing `]`. Subsequent use of `[>`⟨name⟩`]`, with the same or other ⟨objects⟩s, will re-establish the saved transformations, acting on the new ⟨object⟩ with initial reference point appropriate to this ⟨object⟩.

Note: Such namings are global and permanent. They are intended to allow a particular set of transformations to be easily repeated for different objects inside various pictures and diagrams within the same document.

```
\xydef@\xynametransform@#1{%
 \DN@{\expandafter\gdef\expandafter#1\expandafter{\the\toks@}}\next@ }
```

Scaling and Scaled Text The ⟨shape⟩ modifier can contain either a single scale factor, or a pair indicating different factors in the x- and y-directions. Negative values are allowed, to obtain reflections in the coordinate axes, but not zero.

```
\xydef@\xyscalechar@#1@@{\xyscalechars@#1,#1,@@}
```

```
\xydef@\xyscalechars@#1,#2,#3@@{%
 \DN@{\xyrescale@(#1,#2)}%
 \dimen@=#1\p@
 \ifdim\dimen@>\z@ \else \ifdim\dimen@<\z@ \else
  \DN@{\xywarning@{invalid scaling factors #1, #2}}%
 \fi\fi  \dimen@=#2\p@
 \ifdim\dimen@>\z@ \else \ifdim\dimen@<\z@ \else
```

```
  \DN@{\xywarning@{invalid scaling factors #1, #2 }}%
 \fi\fi \next@ }
```

```
\xydef@\xyrescale@(#1,#2){%
 \setboxz@h{\xyscale@@{#1}{#2}%
  \dimen@=\ifx-#1\R@c\advance\dimen@-\R@c \dimen@=-\dimen@
  \else#1\L@c \advance\dimen@-\L@c\fi
 \kern\dimen@\boxz@\kern-\dimen@}%
 \R@c=\if-#1\L@c\else#1\R@c\fi
 \L@c=\if-#1\R@c\else#1\L@c\fi
 \U@c=\if-#2\D@c\else#2\U@c\fi
 \D@c=\if-#2\U@c\else#2\D@c\fi }
```

```
\xydef@\xyignorescale@(#1,#2){\xywarning@{Ignoring rescale: #1,#2}%
 \setboxz@h{\xyscale@@{#1}{#2}\boxz@}}%
```

Some particular scalings.

```
\xydef@\xyshape@halfsize@{\xyrescale@(.5,.5)}
\xydef@\xyshape@dblsize@{\xyrescale@(2,2)}
\xydef@\xyshape@flip@{\xyrescale@(-1,-1)}
\xydef@\xyshape@vflip@{\xyrescale@(1,-1)}
\xydef@\xyshape@hflip@{\xyrescale@(-1,1)}
```

Rotation and Rotated Text Within `[@...]` the `...` are parsed as a ⟨direction⟩ locally, based on the current direction. The value of count register `\Direction` contains the information to determine the requested direction. When no ⟨direction⟩ is parsed then `[@]` requests a rotation to align with the current direction.

The special sequence `[@!...]` is provided to pass an angle directly to the back-end. The Xy-pic size and shape of the ⟨object⟩ with `\rectangleEdge` is unchanged, even though the printed form may appear rotated. This is a feature that must be implemented specially by the back-end. For example, using the PostScript back-end, `[@!45]` will show the object rotated by 45° inside a box of the size of the unrotated object.

```
\xydef@\xyrotatechar@{\bgroup
 \afterDIRECTIONorEMPTY{\xyrotateSizeMod@}{\xyFN@\xySpecialRotate@i}}
```

```
\xydef@\xySpecialRotate@i{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xySpecialRotate@i}%
 \else\ifx @\next \let\next@=\xyrotateSizeMod@
 \else\addEQ@\ifx \next
  \addEQ@\DN@##1@@{%\transformDrop@
   \expandafter\xynametransform@\csname shape [##1]\endcsname\egroup}%
 \else\ifx !\next \DN@!{\egroup\getSpecialRotate@}%
 \else\ifx *\next \DN@*@@{\egroup\transformDrop@}%
 \else \DN@{\xySpecialRotate@@}%
 \fi\fi\fi\fi\fi \next@ }
```

```
\xydef@\getSpecialRotate@{\doSpecialRotate@@}
```

To Do: Provide example of repeated, named transformation.

Installation Initially there is no support for the actual effects. They will be processed according to size/shape requirements, but the actual rotation or re-sizing is ommitted. A ⟨driver⟩ file is required to implement the effects. It need to define alternative expansions to which `\xyRotate@@#1`, `\xySpecialRotate@@#1@@`, `\doSpecialRotate@@#1@@` and `\xyscale@@#1#2` can be bound.

```
\xydef@\xyRotate@#1{\xyundefinedRotate@{rotate}{#1}@@}
\xydef@\xySpecialRotate@#1@@{\xyundefinedRotate@{rotate}{#1}@@\egroup}
\xydef@\doSpecialRotate@#1@@{\xyundefinedRotate@{special rotate}{#1}@@}
\xydef@\xyscale@#1#2{\xyundefinedRotate@{scale}{#1,#2}@@}

\xydef@\UnloadRotate@#1{%
 \ifx#1\relax\else\xyunload@{rotation extension}\fi
 \let\xyRotate@@=\xyRotate@
 \let\xySpecialRotate@@=\xySpecialRotate@
 \let\doSpecialRotate@@=\doSpecialRotate@
 \let\xyscale@@=\xyscale@ }

\UnloadRotate@\relax

\xyaddunsupport{rotate}{\UnloadRotate@\relax}
```

This adjusts the sizes of `\L@c`, `\R@c`, `\U@c` and `\D@c` appropriately.

```
\xydef@\xyrotateSizeMod@ @@{%
 \L@p=\L@c \R@p=\R@c \U@p=\U@c \D@p=\D@c
 \ifdim\cosDirection\p@>\z@
  \R@c=\cosDirection\R@p  \L@c=\cosDirection\L@p
  \U@c=\cosDirection\U@p  \D@c=\cosDirection\D@p
 \else
  \R@c=\cosDirection\L@p  \L@c=\cosDirection\R@p
  \U@c=\cosDirection\D@p  \D@c=\cosDirection\U@p
 \fi
  \advance\L@c \if-\sinDirection\D@p\else\sinDirection\U@p\fi
  \advance\R@c \if-\sinDirection\U@p\else\sinDirection\D@p\fi
  \advance\U@c \if-\sinDirection\L@p\else\sinDirection\R@p\fi
  \advance\D@c \if-\sinDirection\R@p\else\sinDirection\L@p\fi
  \dimen@ii=\L@c \advance\dimen@ii-\L@p
 \edef\tmp@{\egroup \L@c=\the\L@c \R@c=\the\R@c \U@c=\the\U@c \D@c=\the\D@c
  \dimen@ii=\the\dimen@ii \noexpand\setboxz@h\bgroup
  \noexpand\xyRotate@@{\the\Direction}}\tmp@
  \xydropRotated@ }

\xydef@\xydropRotated@@{%
 \kern\dimen@ii \boxz@ \kern-\dimen@ii \modXYtransform@ \egroup }
\xylet@\xydropRotated@=\xydropRotated@@

\xydef@\xyignoreSizeMod@ @@{\xywarning@{Ignoring rotation}}
```

Some particular rotations.

```
\xydef@\xyshape@left@{\xyrotatechar@ v(0,1)@@}
\xydef@\xyshape@right@{\xyrotatechar@ v(0,-1)@@}
```

Reflections Reflections can be specified by a combination of rotation and a flip — either `[hflip]` or `[vflip]`.

```
\xydef@\xyshearchar@{\xyundefinedRotate@{shearing}}
```

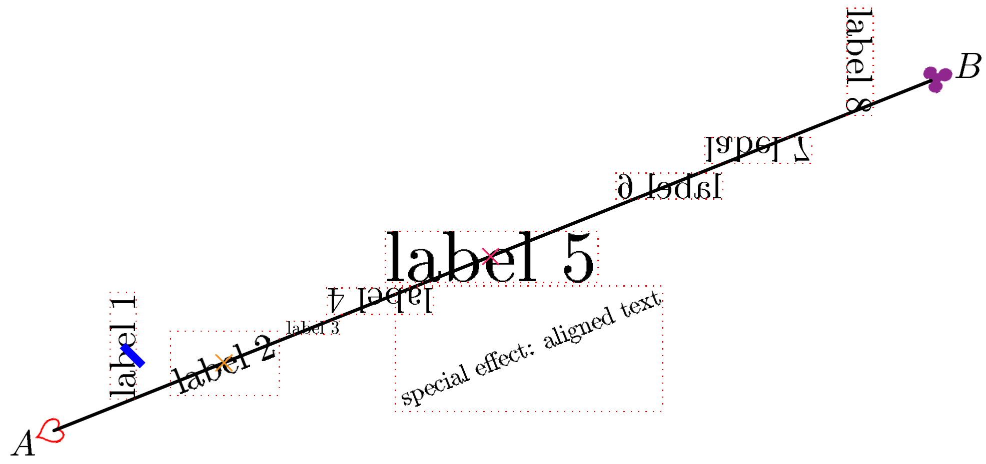


Figure 2.4: Rotations, scalings and flips

Shear transformations **To Do:** Provide the structure to support these; then implement it in PostScript.

The following diagram requires various options to be loaded and installed.

Example The diagram in figure 2.4 illustrates many of the effects described above as well as some additional ones defined by the `color` and `rotate` extensions.

Exercise 2.7: Suggest the code used by the author to typeset 2.4.

The actual code is given in the solution to the exercise. Use it as a test of the capabilities of your DVI-driver. The labels should fit snugly inside the accompanying rectangles, rotated and flipped appropriately.

Bug: This figure also uses colours, alters line-thickness and includes some PostScript drawing. The colours may print as shades of gray, with the line from A to B being thicker than normal. The wider band sloping downwards may have different width and length according to the DVI-driver used; this depends on the coordinate system used by the driver, when 'raw' PostScript code is included.

The end & Log

```
\xyendinput
```

```
% $Log: xyrotate.doc,v $
% Revision 3.3  1996/12/18  09:40:16  ross
% no changes
%
% Revision 3.2  1995/09/19 18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
```

```
% Ready for release of v3.
%
% Revision 2.12  1994/10/25  03:01:14  ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
%
% Revision 2.11  1994/07/05  06:38:37  ross
% some documentation was left out previously
%
% Revision 2.10  1994/06/15  12:46:03  ross
% Second release 3beta.
% Colour and line style saving works; label colouring bug fixed.
%
% Revision 2.9  1994/06/09  14:39:49  ross
% Release 3beta.
%
% NEW file to go in version 2.9 by Ross Moore.
```

2.6 Colour extension

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This extension provides the ability to request that any object be displayed in a particular colour.

It requires a suitable 'driver' option to provide the necessary `\special` commands to implement the effects. Thus

Using this extension will have no effect on the output unless used with a dvi-driver that explicitly supports it.

Header:

```
%% $Id: xycolor.doc,v 3.3 1996/12/19 03:50:08 ross Exp $
%%
%% Xy-pic ``Colour extension'' option.
%% Copyright (c) 1993-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
```

```
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{color}{Colour extension}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

All effects defined in the `color` extension can be implemented using most PostScript ⟨driver⟩s, and some non-PostScript ones, loaded using `\xyoption{xyps}`. This file provides 'generic' code which is known to work correctly with most drivers. For driver-specific variations, consult the appropriate ⟨driver⟩ file.

Colours are specified as a ⟨shape⟩ modifier which gives the name of the colour requested. It is applied to the whole of the current ⟨object⟩ whether this be text, an XY-pic line, curve or arrow-tip, or a composite object such as a matrix or the complete picture. However some DVI drivers may not be able to support the colour in all of these cases.

`[`⟨colour name⟩`]`	use named colour
`\newxycolor{`⟨name⟩`}{`⟨code⟩`}`	define colour
`\UseCrayolaColors`	load colour names

If the DVI-driver cannot support colour then a request for colour only produces a warning message in the log file. After two such messages subsequent requests are ignored completely.

```
\xydef@\xycolorwarning@@{\xycolormessage@
 \gdef\xycolorwarning@{\xycolormessage@
  \xywarning@{...no further colour warnings will be given.}%
 \global\let\xycolorwarning@=\relax }}
\xylet@\xycolorwarning@=\xycolorwarning@@

\xydef@\xycolormessage@{%
  \xywarning@{Current driver does not support colour.}}%
```

There are two methods, perhaps used simultaneously, for handling requests for colour. The `\xylocalColor@` is the normal method, colouring whatever objects TeX typesets from fonts. The more specialized `\xycolor@` is used with the PostScript back-end. This is necessary since some ⟨driver⟩s assume that the current colour should be 'black' when the contents of a `\special` is interpreted, e.g. within PostScript.

Thus a ⟨driver⟩ which supports both colour and PostScript must define a method to handle `\xylocalColor@` but should leave alone `\xycolor@`. The latter method will be adjusted appropriately when a PostScript ⟨driver⟩ is installed.

```
\xydef@\xyNoColor@#1{}
\xylet@\xycolor@@=\xyNoColor@
\xydef@\xyNoColor@@#1#2{\xycolorwarning@}

\xydef@\xycolor@{\xycolor@@}
\xylet@\xylocalColor@=\xyNoColor@@

\xydef@\UnloadColor@{%
 \let\xylocalColor@=\xyNoColor@@
% \let\xycolor@=\xyNoColor@
 \let\xycolor@@=\xyNoColor@
```

```
 \let\xycolorwarning@=\xycolorwarning@@ }
\xylet@\UnloadColor@@=\UnloadColor@
```

```
\xyaddunsupport{color}{\UnloadColor@@\relax}
```

This next macro is to inherit a default colour from the surrounding document, perhaps using one already defined by LaTeX 2_ε

```
\ifx\current@color\undefined \xydef@\xybasecolor@{}%
\else \def\xybasecolor@{\current@color}\fi
```

```
%\xydef@\xycmath@{\hbox\bgroup\dimen@=.55ex \checkxycolor@ \xyinside@}
%\xydef@\xycnomath@{\hbox\bgroup\dimen@=\z@ \checkxycolor@ \xyinside@}
```

```
%\xydef@\checkxycolor@{\ifx\xybasecolor@\empty\else
% \expandafter\imposexycolor@\fi}
%\xydef@\imposexycolor@{\xylocalColor@}
```

A commonly used method is to maintain a colour-stack. The following commands are for use with such a setup, however they should only be called from code installed from a ⟨driver⟩-file. Different drivers may do things differently; e.g. `dvips` uses a single `\special` via: `\def\xycolor@push@@#1{\special{color push #1}}`, but Textures uses two: `\xydef@\xycolor@push#1{\special{color push}\special{color #1}}`. Since the colour-stack idea was proposed by Tomas Rokicki,the `dvips` method is the default behaviour.

```
\xydef@\xystackcolor@{\bgroup \DN@##1##2{\egroup}%
 \ifInvisible@\else\ifHidden@\else\DNii@{\no@@}\ifx\nextii@\Connect@@
 \else\DN@{\xystackcolor@i}\fi\fi\fi \next@ }
```

```
\xydef@\xystackcolor@i#1#2{\checkXyStyle@
 \DN@{#2}\ifx\next@\empty
  \def\tmp@##1{\DN@{\xycolor@push{#1}##1\xycolor@pop}}%
 \else\def\tmp@##1{\DN@{\xycolor@push{#2 #1}##1\xycolor@pop}}\fi
 \toks@={\egroup \let\xy@style@=\relax \def\Drop@@}%
 \expandafter\tmp@\expandafter{\Drop@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\next@}\def\Connect@@}%
  \expandafter\tmp@\expandafter{\Connect@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\next@}}%
 \the\toks@ }%
```

```
\xydef@\xycolor@push@@#1{\special{color push #1}}
\xydef@\xycolor@pop{\special{color pop}}
\xylet@\xycolor@push=\xycolor@push@@
```

Named colours and colour models New colour names are created with `\newxycolor`, taking two arguments. Firstly a name for the colour is given, followed by the code which will ultimately be passed to the output device in order to specify the colour. If the current driver cannot support colour, or grayscale shading, then the new name will be recognised, but ignored during typesetting.

```
\xydef@\newxycolor#1#2{\newxyColor{#1}{}{}{#2}{}}
\xydef@\newxyColor#1#2#3#4{\newxycolor@{#1}{#2}{#3}{#4}{}}
```

For PostScript devices, the XY-ps PostScript dictionary defines operators `rgb`, `cmyk` and `gray` corresponding to the standard RGB and CMYK colour models and grayscale shadings. Colours and shades are described as: $r\ g\ b$ `rgb` or $c\ m\ y\ k$ `cmyk` or s `gray`, where the parameters are numbers in the range $0 \le r, g, b, c, m, y, k, s \le 1$. The operators link to the built-in colour models or, in the case of `cmyk` for earlier versions of PostScript, give a simple emulation in terms of the RGB model.

Existing color names can be overridden using `\newxycolor@#1#2\relax`.

```
\xydef@\newxycolor@#1#2#3#4#5{%
 \expandafter\let\expandafter\next@\csname shape [#1]\endcsname
 \ifx\next@\relax
  \DN@{#4}\ifx\next@\empty
   \DN@{#2}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalColor@{}{#3}\xycolor@{#3}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalColor@{#2}{#3}\xycolor@{#2 #3}}}%
   \fi
  \else\DN@{#2#3}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalColor@{#4}{}\xycolor@{#4}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalColor@{#2}{#3}\xycolor@{#4}}}%
  \fi \fi
 \else \def\tmp@{\relax}\DNii@{#5}%
 \ifx\nextii@\tmp@
  \DN@{#4}\ifx\next@\empty
   \DN@{#2}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalColor@{}{#3}\xycolor@{#3}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalColor@{#2}{#3}\xycolor@{#2 #3}}}%
   \fi
  \else
   \DN@{#2#3}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalColor@{#4}{}\xycolor@{#4}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalColor@{#2}{#3}\xycolor@{#4}}}%
  \fi \fi
 \else\DN@{}\fi\fi \next@\relax}
```

Standard colour names which are defined automatically correspond to the three primaries `red`, `green`, `blue` and their complements `cyan`, `magenta`, `yellow` as well as three extreme grayscale shades `black`, `white`, `gray`.

```
\xydef@\xystandardcolors@{%
% \newxycolor@{black}{1.}{gray}{}\relax  % Textures reverses gray-scales
% \newxycolor@{white}{0.}{gray}{}\relax  %
% \newxycolor@{black}{0.}{gray}{}\relax  % This is normal for PostScript
% \newxycolor@{white}{1.}{gray}{}\relax  %
 \newxycolor@{black}{0. 0. 0.}{rgb}{}\relax   % avoid the hassles.
 \newxycolor@{white}{1. 1. 1.}{rgb}{}\relax   %
 \newxycolor@{gray}{.5}{gray}{}\relax
 \newxycolor@{grey}{.5}{gray}{}\relax
 \newxycolor@{red}{1. 0. 0.}{rgb}{}\relax
 \newxycolor@{green}{0. 1. 0.}{rgb}{}\relax
 \newxycolor@{blue}{0. 0. 1.}{rgb}{}\relax
 \newxycolor@{cyan}{0. 1. 1.}{rgb}{}\relax
 \newxycolor@{magenta}{1. 0. 1.}{rgb}{}\relax
 \newxycolor@{yellow}{1. 1. 0.}{rgb}{}\relax
```

```
\xyuncatcodes }
```

Note: The driver file must call `\xystandardcolors@` before these colours become accessible. This is to allow any ⟨driver⟩-specific definitions to be active when these colours are defined.

Saving colour and styles When styles are saved using `[=`⟨word⟩`]`, see , then the current colour setting (if any) is saved also. Subsequent use of `[`⟨word⟩`]` recovers the colour and accompanying line-style settings.

Further colour names are defined by the command `\UseCrayolaColours` that loads the `crayon` option, in which more colours are defined. Consult the file `xyps-col.doc` for the colours and their specifications in the RGB or CMYK models.

```
\xydef@\UseCrayolaColors{\xyrequire{crayon}\xyuncatcodes}%
% \xyinputorelse@{xyps-col}{\xyerror@{Could not load xyps-col}{}}}
```

`xycrayon.doc:`

This option provides the command to install definitions for the 68 colours recognised by name by Tomas Rokicki's `dvips` driver [11]. This command must be called from a ⟨driver⟩-file which can actually support the colours.

```
%% $Id: xycrayon.doc,v 3.3 1996/12/18 08:58:16 ross Exp $
%%
%% Xy-pic ``Crayola colours'' PostScript backend support.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{crayon}{Crayola Colours}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

The colour definitions themselves are in the file `xyps-col.tex`. They are not loaded directly here since it is not known which ⟨driver⟩ is to be used. Different ⟨driver⟩s may redefine the `\xynewcolor@`

macro to set the colour by name or colour-model.

```
\xydef@\installCrayolaColors@{%
 \xyinputorelse@{xyps-col}{\xyerror@{Could not load Crayola colours}{}}%
 \let\installCrayolaColors@=\relax
 \xyuncatcodes }
```

```
\xyendinput
```

```
% $Log: xycrayon.doc,v $
% Revision 3.3  1996/12/18  08:58:16  ross
% cosmetic updates
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/05  07:58:43  ross
% Ready for v3 release?
%
#
% NEW for version 3.1 by Ross Moore  1995/03/18.
%
```

PostScript **colour** When PostScript driver support is available, the method for handling colour within diagrams may be different. The reason for this is that some dvi-drivers reset the colour to black before placing the contents of the `\special`. To counter this, the PostScript code maintains its own colour-stack. Support is established within a separate file `xyps-c.tex`, loaded from the appropriate ⟨driver⟩-files.

```
\xydef@\xyBEcolorcheck@{\ifx\xycolor@@\xyNoColor@\else
 \xywarning@{PostScript colour support should be OK.}\fi }%
```

```
\xywithoption{ps}{%
 \xyinputorelse@{xyps-c}{\xyerror@{Could not load xyps-c}{}}}%
 \xycatcodes
```

The end & Log

```
\xyendinput
```

```
% $Log: xycolor.doc,v $
% Revision 3.3  1996/12/19 03:50:08  ross
% Maintenance release.
%
% Revision 3.3  1996/12/18  08:56:49  ross
% minor updates
%
% Revision 3.2  1995/09/19  18:20:20  ross
% Bug fix release.
```

```
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.14  1995/07/06  02:56:02  kris
% Buglets...
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.10  1994/06/15  12:46:03  ross
% Second release 3beta.
% Colour and line style saving works; label colouring bug fixed.
%
% Revision 2.9  1994/06/09  14:39:49  ross
% Release 3beta.
%
% NEW for version 2.9 by by Ross Moore.
```

2.7 Pattern and Tile extension

Vers. 3.4 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This extension provides the ability to request that a filled region be tiled using a particular pattern.

This is an effect not normally available within TEX. Instead it requires a suitable ⟨driver⟩ option to provide the necessary `\special` commands, together with any extra commands needed to implement the effects. Thus

Using this extension will have no effect on the output unless used with a dvi-driver that explicitly supports it.

All effects defined in the `tile` extension can be implemented using most PostScript ⟨driver⟩s, loaded as `\xyoption{`⟨driver⟩`}`.

Header:

```
%% $Id: xytile.doc,v 3.4 1997/05/18 01:13:24 ross Exp $
%%
%% Xy-pic ``Pattern and Tile extension'' option.
%% Copyright (c) 1993-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
```

```
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{tile}{Pattern and Tile extension}{\stripRCS$Revision: 3.4 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

Like using color, tiling with patterns is an instance of a special style, since use of a pattern has no effect on the size or shape of the ⟨object⟩ being typeset. It only affects how the ⟨object⟩ will appear on the printed page or on-screen.

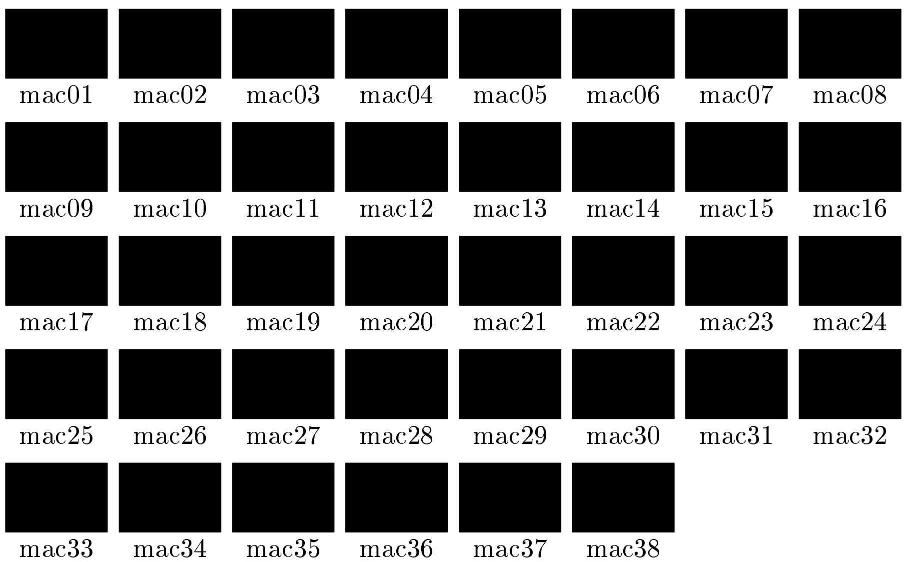


Figure 2.5: The 38 standard Macintosh patterns.

Patterns Patterns are specified as a ⟨shape⟩ modifier, similar to the way colours are specified by name. The pattern is applied to the whole of the current ⟨object⟩ whether this be text, an Xy-pic line, curve or arrow-tip, or a composite object such as a matrix or the complete picture. However some DVI-drivers may not support use of patterns in all cases.

If the current DVI-driver cannot support patterns then a request for one simply produces a warning message in the log file. After two such messages subsequent requests are ignored completely.

```
\xydef@\xypatternwarning@@{\xypatternmessage@
 \gdef\xypatternwarning@{\xypatternmessage@
```

```
  \xywarning@{...no further pattern warnings will be given.}%
 \global\let\xypatternwarning@=\relax }}
\xylet@\xypatternwarning@=\xypatternwarning@@

\xydef@\xypatternmessage@{%
  \xywarning@{Current driver does not support patterns.}}%
```

This allows new patterns to be specified from raw data. Make sure the usual catcodes are in place before the data is tokenised.

```
\xydef@\newxypattern{\begingroup\xycatcodes\newxypattern@i}
\xydef@\newxypattern@i#1#2{\edef\next@{\endgroup
 \noexpand\newxyPattern{#1}{}{#2}{}}\next@}
\xydef@\newxyPattern#1#2#3#4{\newxypattern@{#1}{#2}{#3}{#4}{}}
```

Existing pattern names can be overridden using `\newxypattern@#1#2\relax`.

```
\xydef@\newxypattern@#1#2#3#4#5{%
 \expandafter\let\expandafter\next@\csname shape [#1]\endcsname
 \ifx\next@\relax
  \DN@{#4}\ifx\next@\empty
   \DN@{#2}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalpattern@{}{#3}\xypattern@{#3}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalpattern@{#2}{#3}\xypattern@{#2 #3}}}%
   \fi
  \else\DN@{#2#3}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalpattern@{#4}{}\xypattern@{#4}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalpattern@{#2}{#3}\xypattern@{#4}}}%
  \fi\fi
 \else \def\tmp@{\relax}\DNii@{#5}%
 \ifx\nextii@\tmp@
  \DN@{#4}\ifx\next@\empty
   \DN@{#2}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalpattern@{}{#3}\xypattern@{#3}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalpattern@{#2}{#3}\xypattern@{#2 #3}}}%
   \fi
  \else
   \DN@{#2#3}\ifx\next@\empty
    \DN@{\newxystyle{#1}{\xylocalpattern@{#4}{}\xypattern@{#4}}}%
   \else
    \DN@{\newxystyle{#1}{\xylocalpattern@{#2}{#3}\xypattern@{#4}}}%
  \fi\fi
 \else\DN@{}\fi\fi \next@\relax }
```

`[`⟨name⟩`]` use named pattern

`\newxypattern{`⟨name⟩`}{`⟨data⟩`}`
specify new pattern using ⟨data⟩

`\UsePatternFile{`⟨file⟩`}`

sets default file for patterns

`\LoadAllPatterns{⟨file⟩}`
: load all patterns in ⟨file⟩

`\LoadPattern{⟨name⟩}{⟨file⟩}`
: load named pattern from ⟨file⟩

`\AliasPattern{⟨alias⟩}{⟨name⟩}{⟨file⟩}`
: let ⟨alias⟩ denote pattern from ⟨file⟩.

Although pattern data may be specified directly using `\newxypattern`, it is more usual to load it from a ⟨file⟩ in which many patterns are defined by name, each on a separate line. By convention such files always end in `.xyp` (XY-pattern) so no extension should be specified. The pattern is then requested using either the name supplied in the file or by an alias. Once `\UsePatternFile` has been used, then a null ⟨file⟩ argument to the other commands will still find patterns in the default file. The default remains in effect for the current level of TEX grouping.

For example, the following picture

uses 'filled' frames from the `frame` feature:

```
\AliasPattern{bricks}{mac12}{xymacpat}
\AliasPattern{bars}{mac08}{xymacpat}
\xy *+<5pc,3.1pc>{},{*[bricks]\frm{**}}
 ,*+<2.5pc>[o]{},*[bars]\frm{**}
\endxy
```

There are two methods, perhaps used simultaneously, for handling requests for a pattern. The `\xylocalpattern@` is the normal method, patterning whatever objects TEX typesets from fonts or rules. The more specialized `\xypattern@` is used with a PostScript ⟨driver⟩.

Thus a ⟨driver⟩ which supports both pattern and PostScript must define a method to handle `\xylocalpattern@` but should leave alone `\xypattern@`. The latter method will be adjusted appropriately when a PostScript ⟨driver⟩ is installed.

```
\xydef@\xyNopattern@#1{}
\xylet@\xypattern@@=\xyNopattern@
\xydef@\xyNopattern@@#1#2{\xypatternwarning@}
```

```
\xylet@\xypattern@=\xypattern@@
\xylet@\xylocalpattern@=\xyNopattern@@
```

This is to facilitate unloading patterns when the ⟨driver⟩ is changed to one that does not support this feature. This is the default state, until an appropriate ⟨driver⟩ is loaded.

```
\xydef@\Unloadpattern@{%
 \let\xylocalpattern@=\xyNopattern@@
 \let\xypattern@=\xyNopattern@
 \let\xypatternwarning@=\xypatternwarning@@ }
```

```
\Unloadpattern@
```

Pattern data A region is tiled using copies of a single 'cell' regularly placed so as to seamlessly tile the entire region. The ⟨data⟩ appearing as an argument to `\newxypattern` is ultimately passed to the dvi-driver.

The simplest form of pattern data is: ⟨num⟩ ⟨Hex-data⟩, where the data is a 16-character string of Hexadecimal digits; i.e. 0–9, A–F. Each Hex-digit equates to 4 binary bits, so this data contains 64 bits representing pixels in an 8×8 array. The ⟨num⟩ is an integer counting the number of '0's among the 64 bits. Taken as a fraction of 64, this number or its complement, represents the average density of 'on' pixels within a single cell of the pattern. Drivers unable to provide the fine detail of a pattern may simply use this number, or its complement, as a gray-level or part of a colour specification for the whole region to be tiled.

A desirable set of standard patterns could be specified to be always loaded. If such a set emerges indeed as being frequently desirable then it will be included here.

```
\xydef@\xystandardpatterns@{}%
```

Note: The driver file must call `\xystandardpatterns@` before these patterns become accessible. This is to allow any ⟨driver⟩-specific definitions to be active when these patterns are defined.

```
\xynew@{read}{\xypatread@}
\xydef@\defaultpattfile@{}

\xydef@\xygetfilepatterns@#1{\DN@{#1}%
 \ifx\next@\empty
  \ifx\defaultpattfile@\empty \xywarning@{No pattern file specified.}%
  \else \DN@{\expandafter\openpattfile@\expandafter{\defaultpattfile@}}\fi
 \else \DN@{\openpattfile@{#1}}\fi \next@ }

\xydef@\openpattfile@#1{\openin\xypatread@=#1.xyp %
 \ifeof\xypatread@\xywarning@{No patterns in file: #1.xyp}%
  \closein\xypatread@
 \else \expandafter \xyreadpatfile@ \fi }%

\xydef@\UsePatternFile#1{\def\defaultpattfile@{#1}}

\xydef@\LoadAllPatterns{\let\parsepattern@=\parsepattern@@
  \xygetfilepatterns@ }

\xydef@\LoadPattern#1{\let\parsepattern@=\filepattern@@
  \def\pattname@{#1}\xygetfilepatterns@}

\xydef@\AliasPattern#1#2{\let\parsepattern@=\aliaspattern@@
  \def\aliasname@{#1}\def\pattname@{#2}\xygetfilepatterns@}
```

The character `!` is used to delimit comments within pattern-data files.

```
\xydef@\xyreadpatfile@{%
 \loop@ \bgroup\xycatcodes
  \global\read\xypatread@ to\xypatline@
  \ifeof\xypatread@ \aftergroup\endpatread@
   \else\aftergroup\morepatread@\fi \edef\next@{%
  \egroup\noexpand\getxypattern@{\xypatline@!!!}\relax}\next@
 \repeat@ }
\xydef@\endpatread@{\closein\xypatread@\iffalse}
\xydef@\morepatread@{\iftrue}

%\xydef@\getxypattern@{\futurelet\next\getxypattern@@}
\xydef@\getxypattern@#1{\def\next{#1}\getxypattern@@}
\xydef@\getxypattern@@{\DN@{!!!}\ifx\next\next@\DN@{}%
 \else\DN@{\expandafter\parsepattern@\next}\fi \next@ }

\xydef@\parsepattern@@#1 #2!#3\relax{\newxypattern{#1}{#2}}
```

```
\xydef@\filepattern@@#1 #2!#3\relax{\DN@{#1}%
 \ifx\next@\pattname@
  \DN@{\expandafter\foundpattern@\expandafter{\pattname@}{#2}}%
 \else\DN@{}\fi \next@ }
\xydef@\foundpattern@#1#2{\newxypattern{#1}{#2}\endpatread@\fi
 \expandafter\eat@}
\xydef@\aliaspattern@@#1 #2!#3\relax{\DN@{#1}%
 \ifx\next@\pattname@
  \DN@{\expandafter\foundpattern@\expandafter{\aliasname@}{#2}}%
 \else\DN@{}\fi \next@ }

%\xydef@\parsepattern@#1 #2!#3\relax{\message{#1:#2}}
```

The file `xymacpat.xyp` contains defining data for the 38 standard patterns available with the Macintosh Operating system. Figure 2.5 displays all these patterns.

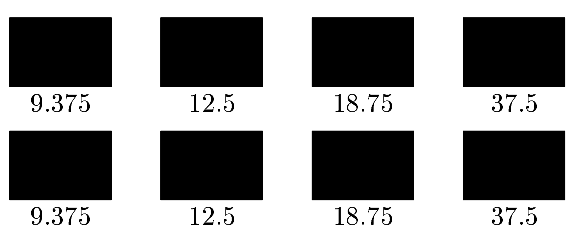
Rotating and Resizing Patterns Some implementations of patterns are sufficiently versatile to allow extra parameters to affect the way the pattern data is interpreted. PostScript is one such implementation in which it is possible to rotate the whole pattern and even to expand or contract the sizes of the basic cell.

Due to the raster nature of output devices, not all such requests can be guaranteed to produce aesthetic results on all devices. In practice only rotations through specific angles (e.g $30°$, $45°$, $60°$) and particular scaling ratios can be reliably used. Thus there is no sophisticated interface provided by XY-pic to access these features. However the 'PostScript escape' mechanism does allow a form of access, when a PostScript ⟨driver⟩ is handling pattern requests.

Special PostScript operators `pa` and `pf` set the pattern angle (normally 0) and 'frequency' measured in *cells per inch*. Hence, when used as an ⟨object⟩-modifier, `[! 30 pa 18.75 pq]` rotates the pattern by $30°$ clockwise and uses a smaller pattern cell (larger frequency). The default frequency of $12.5 = 300/(8 \times 3)$ means that each pixel in a pattern cell corresponds, on a device of resolution 300dpi, to a 3×3 square of device pixels; on such a device 18.75 uses 2×2 squares.

At 300dpi a frequency of $9.375 = 300/(8 \times 4)$ uses 4×4 squares. These match the natural size for pixels on a 75dpi screen and are pretty close for 72dpi screens. Though appropriate for screen displays, these are 'too chunky' for high quality printed work. Doubling the frequency is too fine for some patterns, hence the intermediate choice of 12.5 as default. In order for printed output to match the screen view, a PostScript operator `macfreq` has been defined to facilitate requests for 9.375, via `[!macfreq]`.

The next diagram displays changes to the frequency.



Saving patterns: When styles are saved using ⟨word⟩`]`, see note 1.4k of §1.4, then the current pattern (if any) is also saved. Subsequent use of `[`⟨word⟩`]` recovers the pattern as well as colour and line-style settings. This includes any explicit variations applied using the "Style Escape" mechanism.

Here is a variation of an earlier example, with extra effects.


```
\UsePatternFile{xymacpat}
\AliasPattern{bricks}{mac12}{}
\LoadPattern{mac28}{}\LoadPattern{mac05}{}
\xy *=0[! macfreq -45 pa][mac28][|=Bars]{}
,*+<12pc,4pc>{}*[bricks]\frm{**}
,-<3.5pc,0pt>,*+<2.65pc>[o]{},*[Bars]\frm{**}
,*[thicker]\frm{o},+<6pc,0pt>
,*+<5pc, 2.7pc>{},*[mac05]\frm{**},*\frm{-,}
,*[white]\txt\Large\bf\sf{Kilroy\\was here}
\endxy
```

Add this extension to the driver-tables as “unload”, unless an already-loaded driver can support it:

```
\DN@{\xyaddunsupport{tile}\Unloadpattern@}%
\ifx\xyeverywithoption@tile@\undefined
\else\ifx\xyeverywithoption@tile@\empty
\else
 \DN@{\xysetup@@{\xywithoption{tile}{\xyeverywithoption@tile@\xydriversloaded@@}}}%
\fi\fi \next@
```

The end & Log

```
\xyendinput
```

```
% $Log: xytile.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  09:43:06  ross
% documentation change
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
%
% NEW for version 3.0 by by Ross Moore.
```

2.8 Import graphics extension

Vers. 3.6 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This feature provides the ability to easy add labels and annotations to graphics prepared outside TEX or LATEX. An XY-pic graphics environment is established whose coordinates match that within the contents of the imported graphic, making it easy to specify exactly where a label should be placed, or arrow drawn to highlight a particular feature.

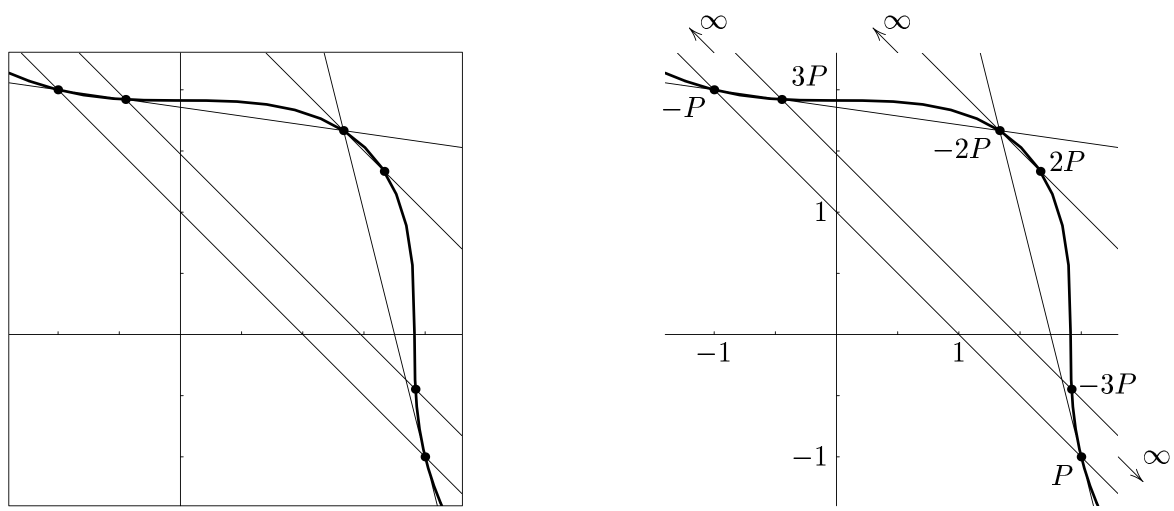
Header:

```
%% $Id: xyimport.doc,v 3.6 1998/03/06 01:28:05 krisrose Exp $
%%
%% Xy-pic ``Import feature'' option.
%% Copyright (c) 1993-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{import}{Import graphics extension}{\stripRCS$Revision: 3.6 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

A command `\xyimport` is defined which is used, in conjunction with imported graphics, to establish a coordinate system appropriate to the particular graphics. This enables ⟨pos⟩itions within the graphic to be easily located, either for labelling or adding extra embellishing features. It is used in either of the follow ways:

`\xyimport(`*width*`,`*height*`){`⟨graphic⟩`}`
`\xyimport(`*width*`,`*height*`)(`*x-off*`,`*y*-off`){`⟨graphic⟩`}`

Normally the ⟨graphics⟩ will be a box containing a graphic imported using the commands from packages such as `graphics`, `epsf` or `epsfig`, or using other commands provided by the local TEX implementation. However the ⟨graphic⟩ could be *any* balanced TEX material whatsoever; provided it occupies non-zero size, both vertically and horizontally.



framed contents of graphics file

Rational points on the elliptic curve: $x^3 + y^3 = 7$

Figure 2.6: importing a graphic for labelling

The *width* and *height* are ⟨number⟩s given in the coordinate system for the *contents of the* ⟨graphics⟩. These are not dimensions, but coordinate-lengths, using the units appropriate to the picture displayed by ⟨graphic⟩.

When provided, (x-*off*,y-off) give the distance in coordinate units from bottom-left corner to where the origin of coordinates should be located, usually within area covered by the ⟨graphic⟩. Usually the negatives of these numbers will give the coordinate location of the bottom-left corner of the ⟨graphic⟩. If no offsets are supplied then the origin is presumed to lie at the bottom-left corner.

```
\xydef@\xyimport#1#{\xyimport@{#1}}
\xydef@\xyimport@#1#2{\xydelayimport@{#1}{#2}}
\xydef@\xydelayimport@#1#2{\xy@@ix@{#1{#2}}%
 \xy@@{\expandafter\xyextern@\the\toks9}}

\xydef@\xyextern@(#1,#2){\hbox\bgroup\R@c=#1\p@ \U@c=#2\p@\xyextern@i}

\xydef@\xyextern@i#1#{\DN@{#1}%
 \ifx\next@\empty \L@c=\z@ \D@c=\z@ \DN@{\xyextern@x}%
 \else \DN@{\xyextern@ii#1}\fi \next@ }

\xydef@\xyextern@ii(#1,#2){\L@c=#1\p@ \D@c=#2\p@
 \advance\R@c-\L@c \advance\U@c-\D@c \xyextern@x}

\xydef@\xyimportdefault@{3}
\xydef@\xyexportwarning@#1#2#3{\xywarning@{%
 #1 for import has zero #2; using \xyimportdefault@#3 default}}

\xydef@\xyextern@x#1{\toks@={\egroup\L@c=}\setboxz@h{#1}%
 \dimen@ii=\L@c \advance\dimen@ii\R@c \dimen@=\wdz@
 \ifdim\dimen@=\z@ \xyexportwarning@{graphic}{width}{cm}%
  \dimen@=\xyimportdefault@ cm\fi
 \ifdim\dimen@ii=\z@ \xyexportwarning@{coords}{width}{unit}%
  \dimen@ii=\xyimportdefault@\p@\fi
```

```
 \quotient@\next@\dimen@\dimen@ii \dimen@=\next@\p@
 \edef\tmp@{\expandafter\removePT@\the\L@c}\L@c=\tmp@\dimen@
 \expandafter\addtotoks@\expandafter{\the\L@c\relax\R@c=}%
 \edef\tmp@{\expandafter\removePT@\the\R@c}\R@c=\tmp@\dimen@
 \expandafter\addtotoks@\expandafter{\the\R@c\relax\setbase@\z@\z@}%
%
 \expandafter\addtotoks@\expandafter{\expandafter{\the\dimen@}\z@\D@c=}%
 \dimen@ii=\U@c \advance\dimen@ii\D@c \dimen@=\ht\z@\advance\dimen@\dp\z@
 \ifdim\dimen@=\z@ \xyexportwarning@{graphic}{height}{cm}%
  \dimen@=\xyimportdefault@ cm\fi
 \ifdim\dimen@ii=\z@ \xyexportwarning@{coords}{height}{unit}%
  \dimen@ii=\xyimportdefault@\p@\fi
 \quotient@\next@\dimen@\dimen@ii \dimen@=\next@\p@
 \edef\tmp@{\expandafter\removePT@\the\D@c}\D@c=\tmp@\dimen@
 \expandafter\addtotoks@\expandafter{\the\D@c\relax\U@c=}%
 \edef\tmp@{\expandafter\removePT@\the\U@c}\U@c=\tmp@\dimen@
 \expandafter\addtotoks@\expandafter{\the\U@c\relax\setbase@@\z@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\the\dimen@}}%
%
 \setboxz@h{\kern-\L@c\lower\D@c\boxz@\kern\L@c}%
  \wd\z@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@ \the\toks@ \X@c=\z@ \Y@c=\z@
 \ifdim\X@min>-\L@c \X@min=-\L@c \fi \ifdim\X@max<\R@c \X@max=\R@c \fi
 \ifdim\Y@min>-\D@c \Y@min=-\D@c \fi \ifdim\Y@max<\U@c \Y@max=\U@c \fi
 \idfromc@{import}\POS }
```

Normally the `\xyimport` command is used at the beginning of an `\xy..\endxy` environment. It is not necessary to give any basis setup, for this is deduced by measuring the dimensions of the ⟨graphic⟩ and using the supplied *width*, *height* and offsets. The ⟨graphic⟩ itself defines named ⟨pos⟩ called `"import"`, located at the origin and having appropriate extents to describe the area covered by the ⟨graphic⟩. This makes it particularly easy to surround the ⟨graphic⟩ with a frame, as on the left side of figure 2.6, or to draw axes passing through the origin.

Here is the code used to apply the labelling in figure 2.6:

```
\def\ellipA{\resizebox{6cm}{!}{%
  \includegraphics{import1.eps}}}
\xy
\xyimport(3.7,3.7)(1.4,1.4){\ellipA}*\frm{-}
,!D+<2pc,-1pc>*+!U\txt{%
 framed contents of graphics file}\endxy
\qquad\qquad
\xy\xyimport(3.7,3.7)(1.4,1.4){\ellipA}
,!D+<2pc,-1pc>*+!U\txt{Rational points
 on the elliptic curve: $x^3+y^3=7$}
,(1,0)*+!U{1},(-1,0)*+!U{-1}
,(0,1)*+!R{1},(0,-1)*+!R{-1}
,(2,-1)*+!RU{P},(-1,2)*+!RU{-P}
,(1.3333,1.6667)*+!UR{-2P}
,(1.6667,1.3333)*!DL{\;2P}
,(-.5,1.9)*++!DL{3P},(1.9,-.5)*!DL{\;-3P}
,(-1,2.3)*+++!D{\infty}*=0{},{\ar+(-.2,.2)}
,(.5,2.3)*+++!D{\infty}*=0{},{\ar+(-.2,.2)}
```

```
,(2.3,-1)*+++!L{\infty}*=0{},{\ar+(.2,-.2)}
\endxy
```

This example uses the LaTeX 2_ε standard `graphics` package to import the graphics file `import1.eps`; other packages could have been used instead. e.g. `epsfig`, `epsf`, or the `\picture` or `\illustration` commands in TEXTURES on the Macintosh.

The only possible problems that can occur are when the graphics package is loaded after XY-pic has been loaded. Generally it is advisable to have XY-pic loading *after* all other macro packages.

The end & Log

```
\xyendinput

% $Log: xyimport.doc,v $
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  09:19:22  ross
% no changes
%
% Revision 3.1  1995/09/05 20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.14  1995/07/05  22:10:51  kris
% Buglets...
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
%
% NEW for version 3.0 by by Ross Moore.
```

2.9 Movie Storyboard extension

Vers. 3.5 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

This extension interprets the `\scene` primitive of the `movie` class, setting the progress indicators to dummy values. The following assumes that your are familiar with the `movie` class.

Header:

```
%% $Id: xymovie.doc,v 3.5 1997/05/28 13:05:01 krisrose Exp $
%%
%% Xy-pic ``Movie Storyboard extension'' option.
%% Copyright (c) 1996-1997 Kristoffer H. Rose <krisrose@brics.dk>
```

```
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{movie}{Movie Storyboard extension}{\stripRCS$Revision: 3.5 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

The size of the frame is determined by the command

`\MovieSetup{width=`*width*`,height=`*height*`,...}`

(the ... indicate the other arguments required by the `movie` class but silently ignored by the XY-pic `movie` extension).

Note: This extension still experimental and subject to change. The only documentation is in the `movie.cls` source file.

The implementation mimics the aspects of the `keyval` package needed here, namely reading the `height` and `width` parameters.

```
\xynew@{dimen}\m@height
\xynew@{dimen}\m@width
```

```
\xydef@\extractparameterm@#1#2#3{%
 \DN@##1,#1=##2,##3@{\csname m@#1\endcsname=##2\relax}%
 \next@,#2,#3,@}
```

```
\xydef@\MovieSetup#1{%
 \extractparameterm@{height}{#1}{height=2in}%
 \extractparameterm@{width}{#1}{width=2in}%
}
```

The progress macros all default to

```
\xydef@\theScene{0}
\xydef@\theF{0}
\xydef@\theFr{1}
```

```
\xydef@\F#1{#1(0)}
\xydef@\Fr#1{#1(1)}
```

We need to interpret and ignore the arguments to `\scene` except the `*` which is used on the last scene since this means that we should output the scene with progress values corresponding to “The End”.

```
\xynew@{if}\iflastframe@
\xynew@{count}\m@scene

\xydef@\scene{\advance\m@scene\@ne
 \DN@{\ifx*\next \lastframe@true \DN@*{\xyFN@\scene@}%
  \else \let\next@=\scene@ \fi \next@}%
 \xyFN@\next@}

\xydef@\scene@{%
 \ifx[\next \DN@[##1]{\scene@i}\else \let\next@=\scene@i \fi \next@}

\long\def\scene@i#1{{%
%
  \def\caption##1{\gdef\m@caption{Scene \the\m@scene. ##1}}%
  \edef\theScene{\the\m@scene}%
  \def\theF{0}\def\theFr{1}%
  \def\F##1{##1(0)}\def\Fr##1{##1(1)}%
  \scene@frame{#1}%
%
  \iflastframe@
   \def\caption##1{}%
   \gdef\m@caption{The End.}%
   \def\theF{1}\def\theFr{0}%
   \def\F##1{##1(1)}\def\Fr##1{##1(0)}%
   \scene@frame{#1}%
  \fi
  }\ignorespaces}

\def\scene@frame#1{\vbox{\null
  \dimen@=\m@width \advance\dimen@2\xydashw@
  \hrule width\dimen@
  \hbox to\dimen@{\vrule width\xydashw@ \hss
   \vbox to\m@height{\hsize=\m@width \null\vfil{#1}\vfil\null}%
   \hss \vrule width\xydashw@}%
  \hrule width\dimen@
  \setbox0=\hbox{%
   \setbox0=\hbox to\m@width{\hss\strut \m@caption\hss}%
   \dimen0=\dp0 \ht0=0pt \dp0=0pt \raise\dimen0\box0}%
  \ht0=0pt \dp0=0pt \box0\relax
  \null}}
```

The end & Log

```
\xyendinput

% $Log: xymovie.doc,v $
% Revision 3.5  1997/05/28 13:05:01  krisrose
```

```
% Fixed missing breaks bug.
%
% Revision 3.4  1997/05/18 02:37:17  krisrose
% Fits movie.cls 3.4...still experimental.
%
% Revision 1.1  1997/05/18 01:22:10  krisrose
% Initial revision
%
% NEW for version 3.4.
```

2.10 PostScript backend

Vers. 3.4 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

XY-ps is a 'back-end' which provides XY-pic with the ability to produce DVI files that use POSTSCRIPT `\specials` for drawing rather than the XY-pic fonts.

In particular this makes it possible to print XY-pic DVI files on systems which do not have the ability to load the special fonts. The penalty is that the generated DVI files will only function with one particular DVI driver program. Hence whenever XY-ps is activated it will warn the user:

XY-pic Warning: The produced DVI file is *not portable*: It contains POSTSCRIPT `\special`s for ⟨one particular⟩ driver

A more complete discussion of the pros and cons of using this backend is included below.

Header:

```
%% $Id: xyps.doc,v 3.4 1997/05/18 01:13:24 ross Exp $
%%
%% Xy-pic ``PostScript backend''.
%% Copyright (c) 1993-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
```

```
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{ps}{PostScript backend}{\stripRCS$Revision: 3.4 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

2.10.1 Choosing the DVI-driver

Including `\xyoption{ps}` within the document preamble, tells XY-pic that the PostScript alternative to the fonts should be used, provided the means to do this is also specified. This is done by also specifying a dvi-driver which is capable of recognising and interpreting `\special` commands. Although the file `xyps.tex` is read when the option request is encountered, the macros contained therein will have no effect until an appropriate driver has also been loaded.

With LaTeX 2_ε both the backend and driver may be specified, along with other options, via a single `\usepackage` command, see [4, page 317]; e.g.

```
\usepackage[ps,textures,color,arrow]{xy}
```

The rebindings necessary to support PostScript are not effected until the `\begin{document}` command is encountered. This means that an alternative driver may be selected, by another `\xyoption{`⟨driver⟩`}`, at any time until the `\begin{document}`. Only the macros relevant to last named ⟨driver⟩ will actually be installed.

The following table describes available support for PostScript drivers. Please consult the individual driver sections in part **??** for the exact current list. For each ⟨driver⟩ there is a corresponding file named `xy`⟨driver⟩`.tex` which defines the necessary macros, as well as a documentation file named `xy`⟨driver⟩`.doc`. The spelling is all lower-case, designed to be both descriptive and unique for the 1st 8 characters of the file names.

⟨driver⟩	Description
`dvips`	Tomas Rokicki's DVIPS
`dvips`	Karl Berry's `dvipsk`
`dvips`	Thomas Kiffe's DVIPS for Macintosh
`textures`	Blue Sky Research's TEXTURES v1.7+
`16textures`	Blue Sky Research's TEXTURES v1.6
`oztex`	Andrew Trevorrow's OzTeX v1.8+
`17oztex`	Andrew Trevorrow's OzTeX v1.7

Other DVI-drivers may also work using one of these files, if they use conventions similar to `dvips`, OzTeX or TEXTURES. Alternatively it should not be too difficult to write the files required, using these as a basis indicating the type of information needed to support the specific `\special` commands. Anyone attempting to do this should inform the author and convey a successful implementation to him for inclusion within the XY-pic distribution.

Note: In some previous versions of XY-pic the PostScript backend and driver were loaded simultaneously by a command of the form `\UsePSspecials{`⟨driver⟩`}`. For backward-compatibility these commands should still work, but now loading the latest version of the given ⟨driver⟩. However their future use is discouraged in favour of the option-loading mechanism, via `\xyoption{`⟨driver⟩`}`. This latter mechanism is more flexible, both in handling upgrades of the actual driver support and in being extensible to support more general forms of `\special` commands.

Once activated the PostScript backend can be turned off and on again at will, using the user following commands:

`\NoPSspecials`	cancels PostScript

`\UsePSspecials {}` restores PostScript

These obey normal TeX scoping rules for environments; hence it is sufficient to specify `\NoPSspecials` within an environment or grouping. Use of PostScript will be restored upon exiting from the environment.

The default level of PostScript is [4], see below.

```
\xydef@\UsePSspecials{\DNii@{{[4]}}\xyFN@\UsePSspecials@i}
\xydef@\UsePSspecials@i{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\UsePSspecials@i}%
 \else\ifx [\next \DN@[##1]{\DNii@{{[##1]}}\xyFN@\UsePSspecials@i}%
 \else\ifcat 5\noexpand\next \DN@##1{\DNii@{{[##1]}}\xyFN@\UsePSspecials@i}%
 \else\ifx\bgroup\next \DN@{\expandafter\UsePS@specials\nextii@}%
 \else \DN@{\UsePS@specials{[4]}}%
 \fi\fi\fi\fi \next@ }
```

To Do: There is redundancy between here and `\UsePSspecials@`. The value of `\setupxyPS@` PostScript level could be set here rather than later in `\PSspecials@`.

```
\xydef@\UsePS@specials#1#2{%
 \ifx\empty\whichPSspecials@
  \DN@{#2}\ifx\next@\empty
  \else
   \expandafter\let\expandafter\next\csname#2@\endcsname
%   \ifx\next\relax \DN@{\xyerror@{PostScript specials for `#2' not supported}{}}%
   \ifx\next\relax \DN@{\UsePSspecials@{#2}{}}%
%   \else \DN@{\expandafter\UsePSspecials@\csname#2@\endcsname#1}\fi
   \else \DN@{\UsePSspecials@{#2}{#1}}\fi
  \fi
 \else
  \DN@{#2}\ifx\next@\empty \DN@{\PSspecials@true}%
  \else
%   \edef\next@{\expandafter\string\csname#2@\endcsname}%
   \DN@{#2}%
   \edef\nextii@{\expandafter\string\whichPSspecials@}%
   \ifx\next@\nextii@\DN@{\PSspecials@true}%
   \else
    \ifx\firstPS@@\relax
     \DN@{\xyerror@{Only one PS <driver> allowed: \dvitype@ already loaded}{}}%
    \else
     \expandafter\let\expandafter\next\csname#2@\endcsname
     \ifx\next\relax
      \DN@{\xyerror@{PostScript specials for `#2' not supported}{}}%
     \else
      \xywarning@{Changing PS <driver> to #2 }%
%      \DN@{\expandafter\UsePSspecials@\csname#2@\endcsname#1}%
      \DN@{\UsePSspecials@{#2}{#1}}%
     \fi
    \fi
   \fi
  \fi
 \fi \next@}
```

The following control sequences will also load the corresponding driver. They are retained for backwards compatibility only; they may be phased out of future releases:

`\UseDVIPSspecials`	`dvips`
`\UsePostScriptSpecials`	`dvips`
`\UseTexturesPSspecials`	`Textures`
`\UseTexturesSpecials`	`Textures`

```
\xydef@\UseDVIPSspecials{\UsePSspecials@\dvips@}
\xydef@\UseTexturesPSspecials{\UsePSspecials@\Textures@}
\xydef@\UseTexturesSpecials{\UsePSspecials@\Textures@}
\xydef@\UsePostScriptSpecials{\UsePSspecials@\dvips@}
\xydef@\UseOzTeXspecials{\UsePSspecials@\OzTeX@}
\xydef@\UseDVITOPSspecials{\UsePSspecials@\dvitops@}
%\xylet@\UseDVIPSSpecials=\UseDVIPSspecials
%\xylet@\UsePSspecials=\UsePostScriptSpecials
%\xylet@\UseTeXturesPSspecials=\UseTexturesPSspecials
%\xylet@\UseTexturesSpecials=\UseTexturesPSspecials
%\xylet@\UseTeXturesSpecials=\UseTexturesSpecials
```

```
\message{drivers,}
```

First the driver is specified by `\UsePSspecials{`⟨driver⟩`}`. This causes a special driver file to be read. Use of fonts is restored at any point by calling `\NoPSspecials`, after which use of PostScript is re-instated by simply `\UsePSspecials`, without need of an argument. The commands described above for specific drivers are simply aliases for `\UsePSspecials@{`⟨driver⟩`}`.

Once the new bindings have been setup, then the use of `\special`s is governed by the value of the conditional `\ifPSspecials@`.

```
\xydef@\loadPSdriver@#1#2{\xyinputorelse@{xy#1}%
 {\xyrequire{#2}\expandafter\ifx\csname xy#2loaded\else %
 {\xyerror@{Unable to load xy#1.tex for #2 driver}{}}\fi}}
```

```
\xydef@\dvips@{\loadPSdriver@{ps-dvi}{dvips}}
\xydef@\Textures@{\loadPSdriver@{ps-txt}{Textures}}
\xydef@\OzTeX@{\loadPSdriver@{ps-oz}{OzTeX}}
\xydef@\dvitops@{\loadPSdriver@{ps-dto}{dvitops}}
\xydef@\dvipsone@{\loadPSdriver@{ps-one}{dvipsone}}
%\xydef@\dviwindo@{\loadPSdriver@{ps-wdo}{dviwindo}}
%\xydef@\dviwin@{\loadPSdriver@{ps-win}{dviwin}}
%\xydef@\pubps@{\loadPSdriver@{ps-pub}{pubps}}
```

The driver file contains definitions which are specific to the particular driver. Note that some drivers may not be able to support all of the PostScript effects that can be requested from within XY-pic. When an unsupported effect is encountered, it is simply ignored; a warning message will be produced unless too many such messages have already been issued.

```
\global\newif\ifPSspecials@
%\xydef@\UsePSspecials@#1{\gdef\whichPSspecials@{#1}\xyFN@\PSspecials@}
\xydef@\UsePSspecials@#1{\def\whichPSspecials@{#1}%
 \DN@{\Textures@}\ifx\next@\whichPSspecials@
  \DN@##1{\xyrequire{textures}%
   \def\whichPSspecials@{textures}\xyps@options}%
```

```
 \else\DN@{\dvips@}\ifx\next@\whichPSspecials@
  \DN@##1{\xyrequire{dvips}%
   \def\whichPSspecials@{dvips}\xyps@options}%
 \else\DN@{\OzTeX@}\ifx\next@\whichPSspecials@
  \DN@##1{\xyrequire{oztex}%
   \def\whichPSspecials@{oztex}\xyps@options}%
 \else\DN@{\dvitops@}\ifx\next@\whichPSspecials@
  \DN@##1{\xyrequire{dvitops}%
   \def\whichPSspecials@{dvitops}\xyps@options}%
 \else
  \DN@##1{\xyrequire{ps}\xyrequire{#1}\xyps@options}%
 \fi\fi\fi\fi \next@ }
```

```
\xydef@\xyps@options{%
 \xyrequire{ps}\xyrequire{line}\xyrequire{color}\xyrequire{crayon}%
 \xyrequire{rotate}\xyrequire{tile}\xyrequire{frame}\xyuncatcodes }
\xydef@\whichPSspecials@{}
```

```
\xydef@\NoPSspecials{\xyPSwarning@{off}%
 \PSspecials@false\aftergroup\resetPS@}
\xydef@\resetPS@{\ifPSspecials@\xyPSwarning@{back on}\fi}
\xydef@\UsePSspecials@x{\PSspecials@true\resetPS@}
```

```
\xydef@\xyPSwarning@#1{\xywarning@{PostScript switched #1}}
%
%\xydef@\xyPSraw@{%
% \ifPSspecials@\expandafter\PSraw@@\else\expandafter\eat@\fi }
```

Unloading the PostScript-backend is handled by the `unload` ⟨driver⟩.

```
\xyaddunsupport{ps}{\csname UnloadPS@@\endcsname}
```

adding new drivers Other drivers can be added by including an appropriate control-sequence name for the driver at this point. The purpose of an expansion such as `\xydef@\OzTeX@{\noexpand\OzTeX@}` is so that this name always refers to a unique token. The macro `\PSspecials@@` defined below, detects this token and then calls up the appropriate macro to make the necessary bindings.

Thus adding a new driver involves 5 steps:

- 1 Define a new control sequence, as just explained.
- 2 Define appropriately named macros to generate the desired `\special`s. (See the existing ones for examples of what is needed here.)
- 3 Define a macro that will perform the appropriate bindings of the different classes of `\special`. (See the existing ones for examples of what is needed here.)
- 4 Determine how frequently the XY-ps dictionary must be included within the PostScript file. Once (at the beginning) is the ideal, however it may be necessary to include it with every page—this is the case with Textures. Also if the dictionary can be loaded as a header or prolog, determine whether this can be done only once or must be for each page. Also check whether a specific name is required, as with OzTeX.
- 5 Check to see whether the `\dumpPSdict` macro correctly writes the dictionary to a file. The `\endlinechar` can be important here.

XY-ps works by rebinding existing control sequence names, in the XY-pic kernel and extensions, to have new expansions. These new expansions may eventually typeset nothing, or at most an empty box; instead they use a `\special` command to place POSTSCRIPT code directly into the `.dvi` file (or `DVI2` resources in the case of TEXTURESon the Macintosh). The new expansions alter the TEX processing, often simplifying it considerably hence leading to savings in both time and memory requirement.

It should not be possible to mix `\specials` intended for different drivers. Thus the first use of `\PSspecials@` will establish which driver is required then rebinds `\UsePSspecials@` so that no other driver can be used; subsequent attempts to use `\UsePSspecials@` simply result in the same driver being reinstated.

```
\xydef@\PSspecials@{%
 \ifx\next[\DN@[##1]{\setxyPSlevel@{##1}\xyFN@\PSspecials@}%
 \else\DN@{\whichPSspecials@\PSspecials@@}%
 \fi \next@ }%
```

```
\xydef@\PSspecials@@{%
  \ifx\und@fined\PSmessage\hidePSmessages\fi \setupxyPS@}
```

An optional argument to the `\UsePSspecials` command allows for some control over precisely when POSTSCRIPT `\specials` will be used. Similarly `\UseDVIPSspecials` and `\UseTexturesSpecials`, etc. can take an optional argument.

For example, `\UsePSspecials[1]` replaces only font characters with a POSTSCRIPT drawing of the same character. Both the POSTSCRIPT and the bitmapped fonts should produce (virtually) identical printed images. This is primarily a test mode.

`\UsePSspecials[2]` also only replaces font characters but such that the number of possible directions is 8192, so that arrowheads turns and hooks fit better.

`\UsePSspecials[3]` draws straight and dotted lines from a single `\special` command, and similarly for circle segments. The printed output should be identical to that obtained with `\UsePostScriptSpecials[2]`, but the size of the `.ps` file should be smaller.

`\UsePSspecials[4]` is the default level; all PostScript is turned on. Dotted curves use equally spaced dots, dashed curves have curved dashes; even dashed lines are better.

`\UsePSspecials[0]` does no rebinding of fonts at all. It allows the special effects, such as rotation, scaling, colour, etc defined in extensions, to be implemented while using the Xy-pic fonts.

```
\xydef@\setupxyPSlevel0@{\relax}
```

```
\xydef@\setxyPSlevel@#1{\ifcase#1%
    \gdef\setupxyPS@{\setupxyPSlevel0@}%
 \or\gdef\setupxyPS@{\setupxyPSlevelA@}%
 \or\gdef\setupxyPS@{\setupxyPSlevelB@}%
 \or\gdef\setupxyPS@{\setupxyPSlevelC@}%
 \or\gdef\setupxyPS@{\setupxyPSlevelD@}%
 \else\gdef\setupxyPS@{\setupxyPSlevelD@}\fi }
```

```
\xywarnifdefined\setupxyPS@
 \gdef\setupxyPS@{\setupxyPSlevelD@ \gdef\setupxyPS@{\PSspecials@true}}
```

```
\xydef@\PSincrease@#1{%
 \xywarning@{The PS level may only increase: #1 is already active}}
```

2.10.2 Why use POSTSCRIPT

At some sites users have difficulty installing the extra fonts used by XY-pic. The `.tfm` files can always be installed locally but it may be necessary for the `.pk` bitmap fonts (or the `.mf` METAFONT fonts) to be installed globally, by the system administrator, for printing to work correctly. If PostScript is available then XY-ps allows this latter step to be bypassed.

Note: with XY-ps it is still necessary to have the `.tfm` font metric files correctly installed, as these contain information vital for correct typesetting.

Other advantages obtained from using XY-ps are the following:

- Circles and circle segments can be set for arbitrary radii.
- solid lines are straighter and cleaner. They are no longer typeset as a collection of fixed-sized segments (drawn from the `xydash10` font). Previously special placement algorithms were required, to construct lines of arbitrary length in up to 8192 distinct directions, from the 128 characters in the font. These algorithms are no longer required, giving improved TEX processing, as well as having smooth lines of arbitrary length and direction limited only by the resolution of the PostScript device.
- The range of possible angles of directionals is greatly increased. For arrow tips, hooks, and turns, there are now 8192 possible orientations rather than just the 128 contained in the `xyatip10`, `xybtip10`, and `xybsql10` fonts.
- Spline curves are smoother. True dotted and dashed versions are now possible, using equally spaced segments which are themselves curved.
- The PostScript file produced by a driver from an XY-ps DVI file is in general significantly smaller than one produced by processing an 'ordinary' DVI file using the same driver. One reason for this is that no font information for the XY-pic fonts is required in the PostScript file; this furthermore means that the use of XY-pic does not in itself limit the PostScript file to a particular resolution.[2]
- The latest version of XY-pic now enables special effects such as variable line thickness, gray-level and colour. Also, rotation of text and (portions of) diagrams is now supported with some drivers. Similarly whole diagrams can be scaled up or down to fit a given area on the printed page. Future versions will allow the use of regions filled with colour and/or patterns, as well as other attractive effects.

Some of the above advantages are significant, but they come at a price. Known disadvantages of using XY-ps include the following:

- A DVI file with specials for a particular PostScript driver can only be previewed if a previewer is available that supports exactly the same `\special` format. A separate PostScript previewer will usually be required.

  However recent versions of `xdvi` support viewing of PostScript using either the GhostScript program or via "Display PostScript". The PostScript produced by XY-ps can be viewed this way
- DVI files created using XY-ps in fact lose their "device-independence". So please do not distribute DVI files with PostScript specials—send either the TEX source code, expecting the recipient to have XY-pic ☺, or send a (compressed) PostScript file.

The latter comment applies to files in which any special 'back-end' support is required, not just to PostScript. Of course it can be ignored when you know the configuration available to the intended recipient.

[2]Most TEX PostScript drivers store the images of characters used in the text as bitmaps at a particular resolution. This means that the PostScript file can only be printed without loss of quality (due to bitmap scaling) at exactly this resolution.

PostScript **header file:** With some DVI-drivers it is more efficient to have the PostScript commands that XY-ps needs loaded initially from a separate "header" file. To use this facility the following commands are available...

```
\UsePSheader {}
\UsePSheader {<filename>}
\dumpPSdict {<filename>}
\xyPSdefaultdict
```

Normally it is sufficient to invoke `\UsePSheader{}`, which invokes the default name of `xy36dict.pro`, referring to the current version of XY-pic. The optional ⟨filename⟩ allows a different file to be used. Placing `\dumpPSdict{..}` within the document preamble causes the dictionary to be written to the supplied ⟨filename⟩.

```
\DN@{\csname newif\endcsname\ifUsePSdict@@\DN@{}}
 \ifx\undefined\UsePSdict@@true\else\DN@{}\fi \next@

\ifx\undefined\xyPSdefaultdict
 \xydef@\xyPSdefaultdict{%
  \DN@##1.##2.##3@{\gdef\xyPSdictname{xy##1##2dict.pro}}%
 \expandafter\next@\xyversion.@}%
\fi

\ifx\undefined\UsePSheader@
 \xydef@\UsePSheader@#1{%
  \DN@{#1}\ifx\next@\empty
%  \ifx\xyPSdictname\undefined\xyPSdefaultdict\fi
  \else \gdef\xyPSdictname{#1}\fi
  \ifx\xyPSdictname\undefined\xyPSdefaultdict\fi
  \gdef\xyHeaderMessage@{%
   It includes a reference to the PostScript file \xyPSdictname.^^J}%
 \UsePSdict@@true}%
 \let\UsePSheader=\UsePSheader@
\fi
```

See the documentation for the specific driver to establish where the dictionary file should be located on any particular TEX system. Usually it is sufficient to have a copy in the current working directory. Invoking the command `\dumpPSdict{}` will place a copy of the requisite file, having the default name, in the current directory. This file will be used as the dictionary for the current processing, provided it is on the correct directory path, so that the driver can locate it when needed. Consult your local system administrator if you experience difficulties.

The 'dump' cannot be performed until a driver has been installed. This ensures that the correct 'end-of-line' character is used.

```
\xydef@\dumpPSdict#1{\DN@{#1}\ifx\next@\empty
  \ifx\undefined\xyPSdictname\xyPSdefaultdict\fi
 \else\gdef\xyPSdictname{#1}\fi
 \def\dumpPSdict@@{\writePSdict@@}%
 \ifx\xydriversselected@@\empty\DN@{\xysetup@@\null@xy@ps}%
 \else \DN@{\installxyps@x}\fi \next@}
```

Multiple instances of `\UsePSheader` and `\dumpPSdict` are possible, only the last will determine which file is used for the current document. The command `\xyPSdefaultdict` reverts to the default name.

Allowing an arbitrary ⟨filename⟩ with `\UsePSheader` is for flexibility, to accommodate systems that may impose special requirements on the filenames of files to be used as PostScript header files. OzTeX is one such. It also caters for advanced users of XY-pic who may wish to experiment with customised PostScript to obtain new effects.

2.10.3 Hooking into XY-pic

```
\message{hooks,}
```

This next macro streamlines the rebinding process.

```
\gdef\xyPSalternative@#1#2{%
\expandafter\global\expandafter\let\csname origxy#2@\endcsname=#1%
 \xdef#1{\noexpand\ifPSspecials@
   \noexpand\expandafter\expandafter\noexpand\csname xyPS#2@\endcsname
  \noexpand\else
   \noexpand\expandafter\expandafter\noexpand\csname origxy#2@\endcsname
  \noexpand\fi}}%
```

The commands `\setupxyPSlevelA@`, `\setupxyPSlevelB@`, `\setupxyPSlevelC@` and `\setupxyPSlevelD@` actually perform the rebindings. Each may be called precisely once, and each requires all lower levels are also set.

```
\xydef@\setupxyPSlevelA@{\xyPStips@%
 \xyPSalternative@{\Tip@@}{Tip}%
 \xyPSalternative@{\Ttip@@}{Ttip}%
 \xyPSalternative@{\Tip@@eu}{Tip@eu}%
 \xyPSalternative@{\Ttip@@eu}{Ttip@eu}%
 \xyPSalternative@{\hook@@}{hook}%
 \xyPSalternative@{\ahook@@}{ahook}%
 \xyPSalternative@{\bhook@@}{bhook}%
 \xyPSalternative@{\aturn@@}{aturn}%
 \xyPSalternative@{\bturn@@}{bturn}%
 \xyPSalternative@{\squiggle@@}{squiggle}%
 \xyPSalternative@{\stopper@@}{stopper}%
 \xyPSalternative@{\line@@}{dash}%
 \xyPSalternative@{\circhar@@}{circhar@}%
% \xywithoption{cmtip}{\xyPScmtips@}%
 \gdef\setupxyPSlevelA@{\PSincrease@{1}}%
 \global\let\UsePSspecials=\UsePSspecials@x \PSspecials@true}
%
\xydef@\setupxyPSlevelB@{%
 \let\xyPSfont@=\empty \let\xyPSsemifont@=\empty
 \xyPSalternative@{\cirrestrict@@}{cirrest}%
 \gdef\setupxyPSlevelB@{\PSincrease@{2}}%
 \setupxyPSlevelA@ }
%
\xydef@\setupxyPSlevelC@{%
 \xyPSalternative@{\cirbuild@}{cirbuild}%
 \xyPSalternative@{\CIRfull@}{CIRfull}%
 \xyPSalternative@{\solid@}{solid}%
 \xyPSalternative@{\point@}{point}%
```

```
 \xywithoption{curve}{%
  \xyPSalternative@{\splinesolid@}{splinesolid}}%
 \gdef\setupxyPSlevelC@{\PSincrease@{3}}%
 \setupxyPSlevelB@ }%
%
\xydef@\setupxyPSlevelD@{%
 \xyPSalternative@{\dash@}{dashed}%
% \xyPSalternative@{\squiggle@}{squiggled}%
 \xywithoption{curve}{%
  \xyPSalternative@{\splinedashed@}{splinedashed}%
  \xyPSalternative@{\splinedotted@}{splinedotted}}%
 \gdef\setupxyPSlevelD@{}%\PSincrease@{4}}%
 \setupxyPSlevelC@ }
```

The bindings are not performed until `\setupxyPS@` is called.

2.10.4 Kernel improvements

Directionals:

These macros standardise the way a character is described in PostScript. First give the `\Direction` code then the operator name, which is just a simple character string. The modifiers `\xyPSfont@` and `\xyPSsemifont@` are empty in all but the most primitive level of PostScript usage. When non-empty they restrict to using only angles corresponding to actual font characters.

```
\xydef@\xyPSchar@#1{%
 \PSmacro@{\the\Direction\space \xyPSfont@ #1}}
```

```
\xydef@\xyPSsemichar@#1{%
 \PSmacro@{\the\Direction\space \xyPSsemifont@ #1}}
```

These macros provide PostScript code to round a `\Direction` code to that corresponding to the nearest font character code, for Directional and semi-Directional fonts respectively.

```
\xydef@\xyPSfont@{f }%{xyfont }
\xydef@\xyPSsemifont@{fs }%{xysdfont }
```

Before any binding we save the original expansions of control-sequences whose names will be subject to rebinding. We give these first for each font.

tips Arrow heads in `\xyatipfont` and `\xybtipfont`

The tips are all set as zero-sized characters...

```
\xydef@\xyPStip@{\xyPSchar@{t}}%{\xyPSchar@{tip}}
%\xydef@\xyPSatip@{\xyPSchar@{a}}%{\xyPSchar@{atip}}
%\xydef@\xyPSbtip@{\xyPSchar@{b}}%{\xyPSchar@{btip}}
\xydef@\xyPSTip@{\xyPSchar@{/XT tt}}%{\xyPSchar@{Tip}}
\xydef@\xyPSTtip@{\setboxz@h\bgroup
  \reverseDirection@\line@ \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@
  \kern-\L@c \boxz@ \kern\L@c \xyPSchar@{/Xt tT}}%{\xyPSchar@{Ttip}}
\xydef@\xyPSTip@eu@{\xyPSchar@{/ET tt}}%{\xyPSchar@{Tip}}
\xydef@\xyPSTtip@eu@{\setboxz@h\bgroup
  \reverseDirection@\line@ \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@
  \kern-\L@c \boxz@ \kern\L@c \xyPSchar@{/ET tT}}%{\xyPSchar@{Ttip}}
```

The mechanism to handle Computer Modern tips is a little different. Here we must rebind `\xycm@` to avoid any calls to font names. When `\PSspecialstrue` then we expand a macro which will look at the following token to decide which type of tip is required after `\tip@x`.

```
%\xydef@\xyPScmtips@{\xyPSalternative@{\xycm@}{cm}}%
\xydef@\xyPScm@\tip@x{\xyFN@\xyPScm@@}%
\xydef@\xyPScm@@{%
      \ifx\next\tip@@  \DN@\tip@@{\tip@x\xyPScmtip@}%
 \else\ifx\next\atip@@ \DN@\atip@@{\tip@x\xyPScmatip@}%
 \else\ifx\next\btip@@ \DN@\btip@@{\tip@x\xyPScmbtip@}%
 \else\DN@{}%
 \fi\fi\fi \next@}%
%
\xydef@\xyPScmtip@{\xyPSchar@{ct}}%{\xyPSchar@{cmtip}}
\xydef@\xyPScmatip@{\xyPSchar@{ca}}%{\xyPSchar@{cmatip}}
\xydef@\xyPScmbtip@{\xyPSchar@{cb}}%{\xyPSchar@{cmbtip}}
```

From version 3.3 the font coding has changed, for greater flexibility and consistency. In particular `\xycm@` is no longer defined.

```
\xydef@\xyPStips@{%
 \xyPSalternative@{\atip@@}{atip}%
 \xyPSalternative@{\btip@@}{btip}%
 \xyPSalternative@{\tip@@}{abtip}%
}%
```

```
\xydef@\xyPSatip@{\xyPSwhichatip@
 \expandafter\xyPSchar@\expandafter{\xyPSfont@@}}%
\xydef@\xyPSbtip@{\xyPSwhichbtip@
 \expandafter\xyPSchar@\expandafter{\xyPSfont@@}}%
\xydef@\xyPSabtip@{\xyPSwhichabtip@
 \expandafter\xyPSchar@\expandafter{\xyPSfont@@}}%
```

```
\xydef@\xyPSwhichatip@{\ifx\xyatipfont\xy@@atfont
 \expandafter\xyPSatip@i \else\def\xyPSfont@@{10 /xy A}\fi }%
\xydef@\xyPSwhichbtip@{\ifx\xybtipfont\xy@@btfont
 \expandafter\xyPSbtip@i \else\def\xyPSfont@@{10 /xy B}\fi }%
\xydef@\xyPSwhichabtip@{\ifx\xyatipfont\xy@@atfont
 \expandafter\xyPSabtip@i \else\def\xyPSfont@@{10 /xy AB}\fi }%
\xydef@\xyPSatip@i{%
  \edef\xyPSfont@@{\tipsize@@\space/\tipfamily@@\space A}}
\xydef@\xyPSbtip@i{%
  \edef\xyPSfont@@{\tipsize@@\space/\tipfamily@@\space B}}
\xydef@\xyPSabtip@i{%
  \edef\xyPSfont@@{\tipsize@@\space/\tipfamily@@\space AB}}
```

```
\xydef@\xyPSTtips@{%
  \let\Ttip@xy=\Ttip@
  \let\Ttip@cm=\Ttip@
  \let\Ttip@eu=\Ttip@ }
```

hooks, turns and squiggles: squiggles from: `\xybsqlfont`

The hooks are zero-sized characters. . .

```
\xydef@\xyPShook@{\xyPSchar@{h}}%{\xyPSchar@{hook}}
\xydef@\xyPSahook@{\xyPSchar@{ha}}%{\xyPSchar@{ahook}}
\xydef@\xyPSbhook@{\xyPSchar@{hb}}%{\xyPSchar@{bhook}}
```

…so are the turns…

```
\xydef@\xyPSaturn@{\xyPSchar@{ta}}%{\xyPSchar@{aturn}}
\xydef@\xyPSbturn@{\xyPSchar@{tb}}%{\xyPSchar@{bturn}}
```

This handles squiggles as individual characters. Squiggled connections should be handled separately, e.g. by having `\xyPSsquiggle@` set the connection by something like `\edef\Connect@@{\noexpand\xyPSsquiggled@ {\the\Direction}}`.

```
\xydef@\xyPSsquiggle@{\xyPSchar@{g}}%{\xyPSchar@{squigl}}
```

To Do: define composite `\xyPSsquiggled@`
To Do: define composite `\xyPSdashsquiggled@`

dashes and stoppers: using characters from: `xydashfont`

A stopper contributes zero size. However a dash gets its size from the italic correction in the `xydashfont`, accessed here using `\origxyline@@`.

```
\xydef@\xyPSstopper@{\xyPSchar@{st}}%{\xyPSchar@{stopper}}

%\xydef@\xyPSfliptrue@{\DN@{\xyPSsemichar@{true dash}}}
\xydef@\xyPSfliptrue@{\DN@{\xyPSsemichar@{T d}}}

\xydef@\xyPSdash@{%
% \DN@{\xyPSsemichar@{false dash}}%
 \DN@{\xyPSsemichar@{F d}}%
 \ifnum\SemiDirectionChar=31
  \ifdim\d@Y<\z@
   \ifdim\d@X<\z@ \ifx\xyPSsemifont@\empty\xyPSfliptrue@\fi\fi
  \else
   \ifdim\z@>\d@X
    \ifx\xyPSsemifont@\empty\else\xyPSfliptrue@\fi
   \else\xyPSfliptrue@\fi
 \fi\fi
 \next@ \setboxz@h{\origxydash@}\kern\wdz@
 \setbox\z@=\box\voidb@x }
```

Circles

full circles and circle segments: `\xycircfont`

This handles the cases where a font character is called using `\circhar#1`.

```
\xydef@\xyPScirchar@#1{\count@=#1\edef\tmp@{\the\count@}%
 \ifcase#1%
  \xyPScirchar@@@{-180}{-135}{\tmp@}\dimen@=\partroottwo\R@
 \or\xyPScirchar@@@{-135}{-90}{\tmp@}\dimen@=\halfroottwo\R@
 \or\xyPScirchar@@@{-90}{-45}{\tmp@}\dimen@=\halfroottwo\R@
 \or\xyPScirchar@@@{-45}0{\tmp@}\dimen@=\partroottwo\R@
 \or\xyPScirchar@@@0{45}{\tmp@}\dimen@=\partroottwo\R@
 \or\xyPScirchar@@@{45}{90}{\tmp@}\dimen@=\halfroottwo\R@
 \or\xyPScirchar@@@{90}{135}{\tmp@}\dimen@=\halfroottwo\R@
 \or\xyPScirchar@@@{135}{180}{\tmp@}\dimen@=\partroottwo\R@
```

```
 \fi}
```

```
\xydef@\xyPScirrest@{\relax}%
```

This replaces just the font character, called using `\circhar@@#1`.

```
\xydef@\xyPScirchar@@#1{\relax\count@=#1\edef\tmp@{\the\count@}%
 \expandafter\xyPScirchar@@@\ifcase#1%
 {-180}{-135}{\tmp@}\kern\partroottwo\R@
 \or{-135}{-90}{\tmp@}\kern\halfroottwo\R@
 \or{-90}{-45}{\tmp@}\kern\halfroottwo\R@
 \or{-45}0{\tmp@}\kern\partroottwo\R@
 \or 0{45}{\tmp@}\kern\partroottwo\R@
 \or{45}{90}{\tmp@}\kern\halfroottwo\R@
 \or{90}{135}{\tmp@}\kern\halfroottwo\R@
 \or{135}{180}{\tmp@}\kern\partroottwo\R@
 \fi}
```

```
\xydef@\xyPScirchar@@@#1#2#3{\edef\tmp@{#1\space#2\space
 \expandafter\removePT@\the\R@\space #3\space c}%
 \PSmacro@{\tmp@}}
```

This next macro will allow for more general circle segments to be done in PostScript. The two parameters are the starting angle and finishing angle respectively, measured anti-clockwise.

```
\xydef@\xyPScirc@#1#2{\edef\tmp@{\expandafter\removePT@\the\R@}%
% \PSmessage{\tmp@\space #1 #2 circ}\PSmacro@{\tmp@\space #1 #2 circ}}
 \PSmacro@{\tmp@\space #1 #2 o}}
```

These gives full circles and circle segments built from quarter turns.

```
\xydef@\xyPSCIRfull@{\setboxz@h{\kern\R@\xyPScirc@0{360}\kern\R@}%
 \wd\z@=\z@ \ht\z@=\R@ \dp\z@=\R@ \boxz@}%
```

```
\xydef@\xyPScirbuild@{{%
 \count@=\CIRlo@@ \count@@=\CIRhi@@
 \multiply\count@ by45 \advance\count@-180 \relax
 \multiply\count@@ by45 \advance\count@@-180 \relax
 \def\tmp@{}%
 \ifx\CIRtest@@\CIRtest@inside
  \ifnum\count@@>\count@
   \edef\tmp@{\noexpand\xyPScirc@{\the\count@}{\the\count@@}}\fi
 \else\ifx\CIRtest@@\CIRtest@outside
  \ifnum\count@>-180 \advance\count@ by360 \relax
   \edef\tmp@{\noexpand\xyPScirc@{\the\count@@}{\the\count@}}%
  \else
   \ifnum\count@@<180 \relax
    \edef\tmp@{\noexpand\xyPScirc@{\the\count@@}{180}}%
   \fi\fi
 \fi\fi
 \kern\R@\tmp@\kern\R@}}
```

Lines

solid, dashed and dotted; without using segments.

This is based on `\straight@`. It is used by all three types of line, taking an appropriate macro as the parameter `#1`.

```
\xydef@\xyPSstraight@#1{\setupDirection@
 \edef\Creset@@{\cfromthec@ \pfromthep@ \DirectionfromtheDirection@}%
 \DN@##1##2{\def\checkoverlap@@{%
  \ifdim##1\X@p>##1\X@c \let\next@=\relax \fi
  \ifdim##2\Y@p>##2\Y@c \let\next@=\relax \fi}}%
 \edef\nextii@{{\sd@X}{\sd@Y}}\expandafter\next@\nextii@
 \noCshavep@@
  \edef\Cshavep@@{\pfromthep@ \noexpand\resetDirection@}%
 \noCshavec@@
  \edef\Cshavec@@{\cfromthec@ \noexpand\resetDirection@}%
 \ifHidden@\else
  \ifdim\Y@c>\Y@max \Y@max=\Y@c \fi  \ifdim\Y@p>\Y@max \Y@max=\Y@p \fi
  \ifdim\Y@c<\Y@min \Y@min=\Y@c \fi  \ifdim\Y@p<\Y@min \Y@min=\Y@p \fi
  \ifdim\X@c>\X@max \X@max=\X@c \fi  \ifdim\X@p>\X@max \X@max=\X@p \fi
  \ifdim\X@c<\X@min \X@min=\X@c \fi  \ifdim\X@p<\X@min \X@min=\X@p \fi \fi
 \ifInvisible@\let\next@=\relax \else
  \DN@{\setboxz@h{\kern\X@c \raise\Y@c\hbox{#1}}%
   \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ {\xykillstyles@\Drop@@}}%
 \fi
 \checkoverlap@@
 \ifdim\d@X=\z@ \ifdim\d@Y=\z@ \DN@{\relax}\fi\fi \next@
 \def\Cslidep@@{\noCslidep@@}\def\Cslidec@@{\noCslidec@@}%
 \def\Calong@@{\noCalong@@}\Creset@@ }
```

This is not needed unless styles are used, hence `xyps-s` is loaded.

```
\xylet@\xykillstyles@=\relax
```

Here is how solid lines are done.

```
%\xydef@\xyPSsolid@{\ifInvisible@ \let\next@=\no@@
% \else \DN@{\xyPSstraight@\xyPSsolid@@@}\fi \next@ }

\xydef@\xyPSsolid@{\straight@\xyPSsolidSpread@}%
\xydef@\xyPSsolidSpread@{\xyPSspread@{solid}{l}@@}%

\xydef@\xyPSdottedSpread@{\xyPSspread@{dotted}{dt}@@}%

\xydef@\xyPSdashed@{\line@ \def\Connect@@{\straight@\xyPSdashedSpread@}}
\xydef@\xyPSdashedSpread@{\xyPSspread@{dashed}{dd}@@}%

\xydef@\xyPSspread@#1#2@@#3\repeat@{\edef\tmp@{%
  \expandafter\removePT@\the\d@X\space\expandafter\removePT@\the\d@Y}%
 \setboxz@h{\kern\X@c \raise\Y@c\hbox{%
  \PSmessage{#1 line:}\PSmacro@{\tmp@\space #2}}}%
 \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ \boxz@}
```

This replaces the XY-pic `\dash@` to give dashed lines:

```
%\xydef@\xyPSdashed@{\line@ \def\Connect@@{\xyPSstraight@\xyPSdashed@@}}
%\xydef@\xyPSdashed@@{\edef\tmp@{%
%  \expandafter\removePT@\the\d@X\space\expandafter\removePT@\the\d@Y }%
%  \PSmessage{dashed line: \tmp@}\PSmacro@{\tmp@\space dd}}
```

This replaces the XY-pic `\point@` to give dotted lines. XY-pic defines `\zerodot` to be a small centred square with side length `\xydashw@`. The POSTSCRIPT substitute is a round dot of this radius.

```
\xydef@\xyPSpoint@{\xyPSzerodot@\egroup
 \Invisible@false \Hidden@false \def\Leftness@{.5}\def\U@pness@{.5}\ctipEdge@
% \def\Drop@@{\styledboxz@}\def\Connect@@{\xyPSstraight@\xyPSdotted@}}
 \def\Drop@@{\styledboxz@}\def\Connect@@{\straight@\xyPSdottedSpread@}}

%\xydef@\xyPSdotted@{\edef\tmp@{%
%  \expandafter\removePT@\the\d@X\space\expandafter\removePT@\the\d@Y }%
%  \PSmessage{dotted line: \tmp@}\PSmacro@{\tmp@\space dt}}
\xydef@\xyPSzerodot@{\PSmessage{dot:}\PSmacro@{p}}
```

Frames

These are used for dashed frames.

This may do circular frames.

Curves

```
\xydef@\xyPScubic@#1{\readsplineparams@
 \PSmessage{CUBIC:}%
 \DN@{#1}\ifx\next@\empty\PSmacro@{mark \cubicinfo@\space bz}%
 \else\PSmacro@{mark \cubicinfo@\space b#1}\fi}

\xydef@\xyPSsquine@#1{\readsplineparams@
 \PSmessage{SQUINE:}%
 \DN@{#1}\ifx\next@\empty\PSmacro@{mark \squineinfo@\space q#1}%
 \else\PSmacro@{mark \squineinfo@\space q#1}\fi}

\xydef@\xyPSsplinesolid@{\ifx\splineinfo@\squineinfo@
 \expandafter\xyPSsquine@\else\expandafter\xyPScubic@\fi{}}

\xydef@\xyPSsplinedashed@{\ifx\splineinfo@\squineinfo@
 \expandafter\xyPSsquine@\else\expandafter\xyPScubic@\fi{d}}

\xydef@\xyPSsplinedotted@{\ifx\splineinfo@\squineinfo@
 \expandafter\xyPSsquine@\else\expandafter\xyPScubic@\fi{t}}
```

The end & Log

```
\xyendinput

% $Log: xyps.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:50:08  ross
% Maintenance release.
%
% Revision 3.3  1996/12/18  09:31:46  ross
% revised interface to straight connections
% revised interface for tips, handles arbitrary scaling
% driver/option-loading now uses \xyrequire
%
```

```
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.14  1995/07/05  22:10:30  kris
% Buglets...
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.12  1994/10/25  03:01:14  ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
%
% Revision 2.11  1994/07/05  06:34:32  ross
% fixed bug with quadratic curves
% fixed minor documentation bugs
%
% Revision 2.10  1994/06/15  12:46:03  ross
% Second release 3beta.
% Colour and line style saving works; label colouring bug fixed.
%
% Revision 2.9  1994/06/09  14:50:54  ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% Revision 2.8  1994/04/08  10:36:40  ross
% Second 3alpha release.
%
% Revision 2.7  1994/03/28  10:57:02  ross
% First version.
%
% Initial version by Ross Moore based on XYps 2.6.
```

2.11 TPIC backend

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This option allows the XY-pic fonts to be replaced by TPIC `\specials`, when used with a dvi-driver capable of supporting them. Extra capabilities include smoother lines, evenly spaced dotted/dashed curves, variable line-widths, gray-scale fills of circles, ellipses and polygonal regions.

Header:

```
%% $Id: xytpic.doc,v 3.3 1996/12/18 09:50:38 ross Exp $
%%
%% Xy-pic ‘‘tpic’’ TPIC backend, for lines, curves and fills.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{tpic}{TPIC backend}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

Use of TPIC `\special`s offers an alternative to the XY-pic fonts. However they require a dvi-driver that is capable of recognizing and interpreting them. One such viewer is `xdvik`, Karl Berry's modification to the `xdvi` viewer on UNIX[3] systems running X-windows or a derivative. `dvipsk`, Karl Berry's modification to `dvips` also handles TPIC `\special`s, so `xdvik`/`dvipsk` is an good combination for quality screen-display and POSTSCRIPT printing.

Once loaded using `\xyoption{tpic}`, with an appropriate ⟨driver⟩ also specified either already or subsequently, the following commands are available to turn the TPIC backend off/on.

`\NoTPICspecials`	turns off TPIC specials.
`\UseTPICspecials`	reinstates TPIC specials.

```
\xynew@{if}\ifTPICspecials@
\xydef@\UseTPICspecials{\TPICspecials@true}
\xydef@\NoTPICspecials{\TPICspecials@false}
```

There is a limit to the number of points allowable in a path. For paths constructed by XY-pic, which includes spline curves, when the limit is reached the path is automatically flushed and a new path commenced. The following command can be used to customise this limit—initially set at 300 for use with XDVI—to suit alternative ⟨driver⟩s.

`\maxTPICpoints{`⟨num⟩`}`	set maximum for paths

[3] UNIX is a trademark of Bell Labs.

```
\xynew@{count}\xytpic@cnt
\xydef@\xytpic@max{300}
\xydef@\maxTPICpoints#1{\count@=#1 \relax
 \ifnum\count@>\tw@ \edef\xytpic@max{\the\count@}%
 \else\expandafter\xyTPICmaxwarning@\expandafter{\the\count@}\fi}
\xydef@\xyTPICmaxwarning@#1{max points must be at least 2, #1 invalid}
```

Each object involving TPIC `\special`s must set the style, using `\xytpic@set@` to store a macro as the expansion of `\xytpic@style@@`. Currently this is global, since a TPIC object must be completed once it has been started — perhaps this is not necessary.

Note that `\xytpic@set@` uses `\xdef`, but has a `\noexpand` in its expansion; this forces expansion of the argument, when there is one, to the control sequence passed as the first token following `\xytpic@set@`.

```
\xydef@\xytpic@style@@{\xytpic@fp}
\xydef@\xytpic@reset@{\gdef\xytpic@style@@{\xytpic@fp}%
 \global\xytpic@cnt=\z@ }
\xydef@\xytpic@set@#1{\xdef\xytpic@style@@{\noexpand#1}}
```

Here are the actual `\specials` that get placed in the dvi-file.

```
\xydef@\xytpic@pn#1{\xyTPIC@special{pn #1}}
\xydef@\xytpic@pa#1#2{\global\advance\xytpic@cnt\@ne\relax
 \ifnum\xytpic@cnt=\xytpic@max\relax \xyTPIC@special{pa #1 #2}%
  \kern-\X@c\raise-\Y@c\hbox{\xytpic@style@@}\kern\X@c
% \xytpic@ip
 \global\xytpic@cnt=\@ne \relax\fi \xyTPIC@special{pa #1 #2}}
\xydef@\xytpic@fp{\xyTPIC@special{fp}}
\xydef@\xytpic@ip{\xyTPIC@special{ip}}
\xydef@\xytpic@da#1{\xyTPIC@special{da #1}}
\xydef@\xytpic@dt#1{\xyTPIC@special{dt #1}}
\xydef@\xytpic@sp#1{\xyTPIC@special{sp #1}}
\xydef@\xytpic@ar#1#2#3#4#5#6{\xyTPIC@special{ar #1 #2 #3 #4 #5 #6}}
\xydef@\xytpic@ia#1#2#3#4#5#6{\xyTPIC@special{ia #1 #2 #3 #4 #5 #6}}
\xydef@\xytpic@sh#1{\xyTPIC@special{sh #1}}
%\xydef@\xytpic@psb#1{\xyTPIC@special{:[begin] #1}}
%\xydef@\xytpic@pse#1{\xyTPIC@special{:[end]}}
%
\xydef@\xytpic@wh{\xyTPIC@special{sh 0}}  % obsolete
\xydef@\xytpic@bk{\xyTPIC@special{sh 1}}  % obsolete

\xydef@\xyTPIC@special#1{\special{#1}}
\xydef@\xyTPIC@noisy#1{\W@{TPIC: #1}\special{#1}}
\xydef@\NoisyTPIC{\let\xyTPIC@special=\xyTPIC@noisy}
```

Some effects may require a TPIC `\special` to be placed both before and after the ⟨object⟩.

```
\xydef@\xyTPICsplit@#1#2{\setboxz@h{#1\boxz@#2}}%
```

TPIC uses units of 'milli-inches' for coordinates, but inches for dot-separations and dash-length.

```
\xydef@\xytpic@pt#1{\dimen@=#1\divide\dimen@ by72 %
 \multiply\dimen@ by 1000 \advance\dimen@.5\p@
 \count@=\dimen@ \divide\count@\p@ }
```

```
\xydef@\xytpic@in#1{\dimen@=#1\divide\dimen@ by72 %
 \edef\next@{\expandafter\removePT@\the\dimen@}}%
```

The TPIC location is calculated relative to the current TEX reference point. Hence most TPIC objects start at the $(0, 0)$ location.

```
\xydef@\xytpic@dot@@{\xytpic@pa00}
\xydef@\xytpic@width@{\xytpic@pt{\xylinethick@}%
 \expandafter\xytpic@pn\expandafter{\the\count@}}
%\xylet@\xytpic@width@=\relax
```

```
\xydef@\xytpic@line{%
 \xytpic@dot@@ \xytpic@pt{-\d@X}\count@@=\count@
 \xytpic@pt\d@Y \edef\next@{{\the\count@@}{\the\count@}}%
 \expandafter\xytpic@pa\next@ \xytpic@style@@ \xytpic@reset@ }
```

```
\xydef@\xytpic@solid@@{\xytpic@set@\xytpic@fp}
\xydef@\xytpic@dashed@@{\xytpic@in\xydashl@ \xytpic@dad}
\xydef@\xytpic@dotted@@{\dimen@=.5\xydashl@ \xytpic@in\dimen@ \xytpic@dtd}
\xydef@\xytpic@dad{\expandafter\xytpic@set@\expandafter{%
  \expandafter\xytpic@da\expandafter{\next@}}}
\xydef@\xytpic@dtd{\expandafter\xytpic@set@\expandafter{%
  \expandafter\xytpic@dt\expandafter{\next@}}}
```

```
\xydef@\xytpic@polysolid@{\xytpic@set@\xytpic@fp}
\xydef@\xytpic@polydashed@#1{\count@=#1 \xytpic@dad}
\xydef@\xytpic@polydotted@#1{\count@=#1 \xytpic@dtd}
\xydef@\xytpic@postpoly@{\xytpic@style@@ \xytpic@reset@}
```

re-bindings This is similar to the mechanism used by the PostScript back-end to rebind "hooks" to incorporate a switch according to the current value of `\ifTPICspecials@`.

```
\xydef@\TPICalternative@#1#2{%
\expandafter\global\expandafter\let\csname origxy#2@\endcsname=#1%
 \xdef#1{\noexpand\ifTPICspecials@
   \noexpand\expandafter\expandafter\noexpand\csname xytpic@#2@\endcsname
  \noexpand\else
   \noexpand\expandafter\expandafter\noexpand\csname origxy#2@\endcsname
  \noexpand\fi}}%
```

Allow only one instance of `\setupxyTPIC@`; thereafter simply do `\reloadxyTPIC@` to turn on anything that has been turned off.

```
\xydef@\setupxyTPIC@{\xywarning@{using TPIC specials}%
 \TPICalternative@{\solid@}{solid}%
 \TPICalternative@{\dash@}{dashed}%
 \TPICalternative@{\point@}{point}%
 \TPICalternative@{\circhar@@}{circhar}%
 \TPICalternative@{\splinesolid@}{splinesolid}%
% \TPICalternative@{\splinedashed@}{splinedashed}%
% \TPICalternative@{\splinedotted@}{splinedotted}%
 \xywithoption{frame}{\xyinputorelse@{xytp-f}{}\xyuncatcodes}%
 \let\setupxyTPIC@=\reloadxyTPIC@
 \activateTPIC@ }
```

```
\xydef@\xyunloadTPIC@{\xywarning@{TPIC specials turned off}%
 \xywithoption{curve}{\xysetup@@\uninstallTPICcurves@ }%
 \xywithoption{line}{\xysetup@@\uninstallTPICline@ }%
 \xywithoption{frame}{\xysetup@@\uninstallTPICframes@ }%
 \TPICspecials@false
 \let\setupxyTPIC@=\reloadxyTPIC@ }
```

```
\xydef@\activateTPIC@{%
 \xywithoption{line}{\xysetup@@\installTPICline@}%
 \xywithoption{curve}{\xysetup@@\installTPICcurves@}%
 \xywithoption{frame}{\xysetup@@\installTPICframes@}%
 \TPICspecials@true }
```

```
\xydef@\reloadxyTPIC@{\xywarning@{TPIC specials back on}\activateTPIC@ }
```

```
\xydef@\xytpic@solid@{%
 \ifInvisible@ \let\next@=\no@@
 \else \DN@{\xyTPICstraight@{\xytpic@solid@@ \xytpic@line}}\fi
 \next@ }
```

```
\xydef@\xytpic@dashed@{\line@
 \def\Connect@@{\xyTPICstraight@{\xytpic@dashed@@ \xytpic@line}}}
```

```
\xydef@\xytpic@point@{\xytpic@dot@@\xytpic@fp\egroup
 \Invisible@false \Hidden@false \def\Leftness@{.5}\def\Upness@{.5}\ctipEdge@
 \def\Drop@@{\styledboxz@}%
 \def\Connect@@{\xyTPICstraight@{\xytpic@dotted@@ \xytpic@line}}}
```

```
\xydef@\xyTPICstraight@#1{\setupDirection@
 \edef\Creset@@{\cfromthec@ \pfromthep@ \DirectionfromtheDirection@}%
 \DN@##1##2{\def\checkoverlap@@{%
  \ifdim##1\X@p>##1\X@c \let\next@=\relax \fi
  \ifdim##2\Y@p>##2\Y@c \let\next@=\relax \fi}}%
 \edef\nextii@{{\sd@X}{\sd@Y}}\expandafter\next@\nextii@
 \noCshavep@@ \edef\Cshavep@@{\pfromthep@ \noexpand\resetDirection@}%
 \noCshavec@@ \edef\Cshavec@@{\cfromthec@ \noexpand\resetDirection@}%
 \ifHidden@\else
  \ifdim\Y@c>\Y@max \Y@max=\Y@c \fi  \ifdim\Y@p>\Y@max \Y@max=\Y@p \fi
  \ifdim\Y@c<\Y@min \Y@min=\Y@c \fi  \ifdim\Y@p<\Y@min \Y@min=\Y@p \fi
  \ifdim\X@c>\X@max \X@max=\X@c \fi  \ifdim\X@p>\X@max \X@max=\X@p \fi
  \ifdim\X@c<\X@min \X@min=\X@c \fi  \ifdim\X@p<\X@min \X@min=\X@p \fi \fi
 \ifInvisible@\let\next@=\relax \else
  \DN@{\setboxz@h{\kern\X@c \raise\Y@c\hbox{#1}}%
   \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ {\Drop@@}}%
 \fi
 \checkoverlap@@
 \ifdim\d@X=\z@ \ifdim\d@Y=\z@ \DN@{\relax}\fi\fi \next@
 \def\Cslidep@@{\noCslidep@@}\def\Cslidec@@{\noCslidec@@}%
 \def\Calong@@{\noCalong@@}\Creset@@ }
```

straight lines

spline curves Of the curves defined in the xycurve extension, only solid spline curves are supported. This is done by treating the spline as a polygon (poly-line) with many segments. The dotted or dashed variants do not work correctly.

Implementations of TPIC draw dashed polygons such that the start and finish of each segment is solid. Since these segments can be very short, the effect is simply to create a solid line. Similarly the shortness of the segments tends to give nothing at all for large portions of a dotted curve. What is needed is an implementation whereby the on/off nature of a dashed or dotted polygon is determined by the accumulated length, not the length along just the current segment.

```
\xydef@\xytpic@splinesolid@{\xyTPICspline@\xytpic@fp }
\xydef@\xytpic@splinedashed@{\xytpic@in\xydashl@\expandafter\xyTPICspline@
 \expandafter{\expandafter\xytpic@da\expandafter{\next@}}}
\xydef@\xytpic@splinedotted@{\xytpic@in{.5\xydashl@}\expandafter\xyTPICspline@
 \expandafter{\expandafter\xytpic@dt\expandafter{\next@}}}

\xydef@\xyTPICspline@#1{\setbox\splinebox@=\copy\voidb@x %\zerodot
 \SloppyCurves %\splinetol@=\xydashl@  % \xytpic@ip
 \global\xytpic@cnt=\@ne \xytpic@set@{#1}%
 \def\splineextra@@{\xyTPICspline@dot}%
 \def\xycrvdrop@{ }\def\xycrvconn@{}\splineset@@ \xytpic@style@@}

\xydef@\xyTPICspline@dot{\xytpic@pt{\X@c}\count@@=\count@
 \xytpic@pt{-\Y@c}\edef\next@{{\the\count@@}{\the\count@}}%
 \expandafter\xytpic@pa\next@ }


\xydef@\xyTPIClinewidth@#1{\xytpic@pt{#1}\edef\next@{\the\count@}}

\xydef@\installTPICline@{%
 \let\xylinewidth@@=\xylinewidth@
 \let\xylineSpecial@@=\xytpic@pn
 \let\transxyline@@=\xyTPIClinewidth@ }

\xydef@\uninstallTPICline@{\UnloadLine@}%
```

line width

circles and ellipses Here we catch circles and ellipses specified using the \xycircle command from the xycurve extension. Since TPIC supports only solid circles, we catch it only when the ⟨style⟩ is either {} or {-}.

```
\xydef@\xydoTPICcircle@#1{\DN@{#1}%
 \ifx\next@\empty\DN@{\xyTPICfullcircle@}%
 \else\expandafter\DNii@\expandafter{\addDASH@{}}%
  \ifx\next@\nextii@\DN@{\L@=\R@ \xyTPICfullcircle@}%
 \else\DN@{\dosolidcircle@{#1}}\fi\fi \next@ }

\xydef@\xyTPICfullcircle@{\xyTPICpartcircle@{00}0\twoPi@}%

\xydef@\xyTPICpartcircle@#1#2#3{%
 \xytpic@pt{\R@}\expandafter\DNii@\expandafter{\the\count@}%
 \xytpic@pt{\L@}\edef\next@{#1{\nextii@}{\the\count@}{#2}{#3}}%
 \expandafter\xytpic@ar\next@ }

\xydef@\qartPi@{0.7853981}
\xydef@\tartPi@{2.3561944}
```

This replaces just the font character, called using `\circhar@@#1`.

```
\xydef@\xytpic@circhar@#1{\xytpic@width@\ifcase#1%
  \xytpic@pt{\R@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@{\tartPi@}{\fullPi@}\kern\partroottwo\R@
 \or
  \xytpic@pt{\halfroottwo\R@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@{\halfPi@}{\tartPi@}\kern\halfroottwo\R@
 \or
  \xytpic@pt{\z@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@{\qartPi@}{\halfPi@}\kern\halfroottwo\R@
 \or
  \xytpic@pt{-\halfroottwo\R@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@0{\qartPi@}\kern\partroottwo\R@
 \or
  \xytpic@pt{-\halfroottwo\R@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@{-\qartPi@}0\kern\partroottwo\R@
 \or
  \xytpic@pt{\z@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@{-\halfPi@}{-\qartPi@}\kern\halfroottwo\R@
 \or
  \xytpic@pt{\halfroottwo\R@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@{-\tartPi@}{-\halfPi@}\kern\halfroottwo\R@
 \or
  \xytpic@pt{\R@}\edef\next@{{{\the\count@}0}}%
  \expandafter\xyTPICcirchar@i\next@{-\fullPi@}{-\tartPi@}\kern\partroottwo\R@
 \fi}

\xydef@\xyTPICcirchar@i#1#2#3{\L@=\R@ \xyTPICpartcircle@{#1}{#2}{#3}}
```

```
\xydef@\xyTPIC@spline@ii@{\DN@{\xy@spline@ii}%
 \ifx\splineSTYLE@\empty\DN@{\xyTPICqspline@{0}}%
 \else\expandafter\DNii@\expandafter{\addDASH@{}}%
 \ifx\splineSTYLE@\nextii@\DN@{\xyTPICqspline@{0}}%
 \else\expandafter\DNii@\expandafter{\addDOT@{}}%
 \ifx\splineSTYLE@\nextii@\DN@{\xyTPICqspline@{-.5}}%
 \else\expandafter\expandafter\expandafter\DNii@\expandafter\expandafter
  \expandafter{\expandafter\addDASH@\addDASH@{}}%
 \ifx\splineSTYLE@\nextii@\DN@{\xyTPICqspline@{}}%
 \fi\fi\fi\fi
 \next@ }%

\xydef@\xyTPIC@accumulate{\xytpic@pt\X@c \count@@=\count@
 \xytpic@pt{-\Y@c}\edef\next@{{\the\count@@}{\the\count@}}%
 \expandafter\xytpic@pa\next@ }

\xydef@\xyTPICqspline@#1{\xytpic@width@
 \xytpic@in{#1\xydashl@}\expandafter\gdef\expandafter\xytpic@style@@
  \expandafter{\expandafter\xytpic@sp\expandafter{\next@}}%
 \save@ \cfromp@ \xyTPIC@accumulate
  \edef\smapp@@{\s@bot}\smapxy@i{\xyTPIC@accumulate}%
```

```
 \leave@ \xyTPIC@accumulate \xytpic@style@@ \xytpic@reset@
 \enter@{}\xy@spline@x }
```

```
\xydef@\installTPICcurves@{%
 \let\dosolidcircle@@=\xydoTPICcircle@
 \let\xy@spline@ii@@=\xyTPIC@spline@ii@
 }
\xydef@\uninstallTPICcurves@{%
 \let\dosolidcircle@@=\dosolidcircle@
 \let\xy@spline@ii@@=\xy@spline@ii }
```

quadratic splines

```
 \xywithoption{frame}{\xyinputorelse@{xytp-f}{}}
```

frames Implement color fills by reading from a `\TPICfillcolor`. Implement pattern fills by reading from a `\TPICfillpatt`. Read these whenever a fill is requested.

```
\xydef@\uninstallTPICframes@{\UnloadFrames@}
```

```
\xyendinput
```

```
% $Log: xytpic.doc,v $
% Revision 3.3  1996/12/18  09:50:38  ross
% adjusted Drop@@ for styles
% minor improvements to file-loading commands
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.1 by Ross Moore  1995/03/18.
% replaces xyps-col.doc
%
```

2.12 em-TeX backend

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

Eberhard Matte's em-TEX implementation provides a suite of `\special` commands to facilitate the drawing of lines, both on-screen and with various printing devices. This 'back-end' extension allows the lines in XY-pic diagrams to be drawn using these methods.

Note that this extension does not have to be used with em-TEX. Better results may be obtained using the POSTSCRIPT back-end and DVIPS ⟨driver⟩, since a version of DVIPS is available for em-TEX.

However, in particular for screen previewing purposes, it may be convenient to use this back-end. Furthermore note that DVIPS is capable of supporting em-TEX`\special`s.

Header:

```
%% $Id: xyemtex.doc,v 3.3 1996/12/18 09:18:00 ross Exp $
%%
%% Xy-pic ``em-TeX'' back-end extension.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{emtex}{em-TeX backend}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

Once loaded using `\xyoption{emtex}`, with an appropriate ⟨driver⟩ also specified either already or subsequently, the following commands are available to turn the em-TEX backend off/on.

`\NoEMspecials`	turns off em-TEX specials.
`\UseEMspecials`	reinstates em-TEX specials.

```
\xynew@{if}\ifEMspecials@
\xydef@\UseEMspecials{\EMspecials@true}
\xydef@\NoEMspecials{\EMspecials@false}
```

```
\xydef@\xyemtex@lineto{\xyEM@special{lineto}}
\xydef@\xyemtex@moveto{\xyEM@special{moveto}}
\xydef@\xyemtex@line#1{\xyEM@special{line #1}}
\xydef@\xyemtex@point#1{\xyEM@special{point #1}}
\xydef@\xyemtex@pen#1{\addtostyletoks@{\xyEM@special{linewidth #1}}}
```

The `\specials` These place the actual `\special`s and allow tracing:

```
{\catcode`:=12 \gdef\next#1{\special{em:#1}}}
\xylet@\xyEM@special=\next
```

```
{\catcode`:=12 \gdef\next#1{%
 \W@{EM:#1 (\the\X@p,\the\Y@p),(\the\X@c,\the\Y@c)}\special{em:#1}}}
\xylet@\xyEM@noisy=\next
```

```
\xydef@\NoisyEMTeX{\let\xyEM@special=\xyEM@noisy}
```

Currently these are not used by XY-pic.

```
%\xydef@\xyemtex@message#1{\xyEM@special{message #1}}
%\xydef@\xyemtex@graph#1{\xyEM@special{graph #1}}
```

re-bindings This is similar to the mechanism used by the PostScript back-end to rebind "hooks" to incorporate a switch according to the current value of `\ifEMspecials@`.

```
\xydef@\EMalternative@#1#2{%
\expandafter\global\expandafter\let\csname origxy#2@\endcsname=#1%
 \xdef#1{\noexpand\ifEMspecials@
   \noexpand\expandafter\expandafter\noexpand\csname xyemtex@#2@\endcsname
  \noexpand\else
   \noexpand\expandafter\expandafter\noexpand\csname origxy#2@\endcsname
  \noexpand\fi}}%
```

Allow only one instance of `\setupxyEM@`; thereafter simply do `\reloadxyEM@` to turn on anything that has been turned off.

```
\xydef@\setupxyEMTeX@{\xywarning@{using em-TeX specials}%
 \EMalternative@{\solid@}{solid}%
% \EMalternative@{\point@}{point}%
 \EMalternative@{\splinesolid@}{splinesolid}%
 \let\setupxyEM@=\reloadxyEM@
 \activateEM@ }
```

```
\xydef@\xyunloadEM@{\xywarning@{EM specials turned off}%
% \xywithoption{curve}{\xysetup@@\uninstallEMcurves@ }%
 \xywithoption{line}{\xysetup@@\uninstallEMline@ }%
% \xywithoption{frame}{\xysetup@@\uninstallEMframes@ }%
 \EMspecials@false
 \let\setupxyEM@=\reloadxyEM@ }
```

```
\xydef@\activateEM@{%
 \xywithoption{line}{\xysetup@@\installEMline@}%
% \xywithoption{curve}{\xysetup@@\installEMcurves@}%
% \xywithoption{frame}{\xysetup@@\installEMframes@}%
 \EMspecials@true }
```

```
\xydef@\reloadxyEM@{\xywarning@{EM specials back on}\activateEM@ }
```

```
\xydef@\xyemtex@solid@{%
 \ifInvisible@ \let\next@=\no@@
 \else \DN@{\xyEMstraight@}\fi
```

```
 \next@ }
```

```
\xydef@\xyEMstraight@{\setupDirection@
 \edef\Creset@@{\cfromthec@ \pfromthep@ \DirectionfromtheDirection@}%
 \DN@##1##2{\def\checkoverlap@@{%
  \ifdim##1\X@p>##1\X@c \let\next@=\relax \fi
  \ifdim##2\Y@p>##2\Y@c \let\next@=\relax \fi}}%
 \edef\nextii@{{\sd@X}{\sd@Y}}\expandafter\next@\nextii@
 \noCshavep@@ \edef\Cshavep@@{\pfromthep@ \noexpand\resetDirection@}%
 \noCshavec@@ \edef\Cshavec@@{\cfromthec@ \noexpand\resetDirection@}%
 \ifHidden@\else
  \ifdim\Y@c>\Y@max \Y@max=\Y@c \fi  \ifdim\Y@p>\Y@max \Y@max=\Y@p \fi
  \ifdim\Y@c<\Y@min \Y@min=\Y@c \fi  \ifdim\Y@p<\Y@min \Y@min=\Y@p \fi
  \ifdim\X@c>\X@max \X@max=\X@c \fi  \ifdim\X@p>\X@max \X@max=\X@p \fi
  \ifdim\X@c<\X@min \X@min=\X@c \fi  \ifdim\X@p<\X@min \X@min=\X@p \fi \fi
 \ifInvisible@\let\next@=\relax \else
  \DN@{%
   \setboxz@h{\kern\X@p \raise\Y@p\hbox{\xyemtex@moveto}}%
   \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ %
   \setboxz@h{\boxz@ \kern\X@c \raise\Y@c\hbox{\xyemtex@lineto}}%
   \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ {\Drop@@}}%
 \fi
 \checkoverlap@@
 \ifdim\d@X=\z@ \ifdim\d@Y=\z@ \DN@{\relax}\fi\fi \next@
 \def\Cslidep@@{\noCslidep@@}\def\Cslidec@@{\noCslidec@@}%
 \def\Calong@@{\noCalong@@}\Creset@@ }
```

straight lines

spline curves Of the curves defined in the xycurve extension, only solid spline curves are supported. This is done by treating the spline as a polygon (poly-line) with many segments.

```
\xydef@\xyemtex@splinesolid@{\setbox\splinebox@=\copy\voidb@x
 \multiply\splinetol@\thr@@
 \gdef\splineextra@@{\xyemtex@moveto\gdef\splineextra@@{\xyemtex@lineto}}%
 \def\xycrvdrop@{ }\def\xycrvconn@{}\splineset@@ }
```

```
\xydef@\xyEMlinewidth@#1{\dimen@=#1\relax\edef\next@{\the\dimen@}}
```

```
\xydef@\installEMline@{%
 \let\xylinewidth@@=\xylinewidth@
 \let\xylineSpecial@@=\xyemtex@pen
 \let\transxyline@@=\xyEMlinewidth@ }
```

```
\xydef@\uninstallEMline@{\UnloadLine@\relax}%
```

```
\xydef@\UseEMlinewidth{\installEMline@}%
\xydef@\NoEMlinewidth{\uninstallEMline@}%
```

line width

The end & Log

```
\xyendinput

% $Log: xyemtex.doc,v $
% Revision 3.3  1996/12/18  09:18:00  ross
% fixed bug affecting line-widths
%
% Revision 3.1  1995/09/05  20:28:57  ross
% Releasing version 3.1!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

2.13 Necula's extensions

Vers. 0.0 by George C. Necula ⟨necula@cs.cmu.edu⟩

Header:

```
%% $Id$
%%
%% Xy-pic ``Necula extensions'' option.
%% Copyright (c) 1998          George C. Necula <necula@cs.cmu.edu>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1998  Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{necula}{Necula's extensions}{\stripRCS$Revision: 0.0 $}%
 {George C. Necula}{necula@cs.cmu.edu}%
 {School of Computer Science,
```

```
  Carnegie Mellon University,
  5000 Forbes Avenue,
  Pittsburgh, PA 15213-3891, USA}
```

This option contains two extensions of the XY-pic kernel: A way to expand TEX macros in object ⟨modifier⟩s, and a way to specify arbitrary polygons as the ⟨shape⟩ of an object.

2.13.1 Expansion

The special syntax `e|`⟨macros⟩`|` is introduced in an object ⟨modifier⟩s and ⟨coord⟩inates. It expands the given TEX ⟨macros⟩ (with `\edef`) before reinterpretation as a ⟨modifier⟩ of ⟨coord⟩, respectively.

```
\def\expandbeforenext@#1{%
  \DN@ e|##1|{\edef\tmp@{##1}\expandafter\xyFN@\expandafter#1\tmp@}%
}
```

```
\xylet@\xy@oldOBJECT@=\OBJECT@
\xydef@\xy@newOBJECT@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xy@newOBJECT@}%gobble spaces
 \else\ifx e\next
  \expandbeforenext@\OBJECT@
 \else
  \DN@{\xy@oldOBJECT@}\fi\fi
 \next@}%
\let\OBJECT@=\xy@newOBJECT@
```

```
\xylet@\xy@oldCOORD@=\COORD@
\xydef@\xy@newCOORD@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xy@newCOORD@}%gobble spaces
 \else\ifx e\next
  \expandbeforenext@\COORD@
 \else
  \DN@{\xy@oldCOORD@}\fi\fi
 \next@}%
\let\COORD@=\xy@newCOORD@
```

This code may become part of the XY-pic kernel at a certain point.

2.13.2 Polygon shapes

A polygon ⟨shape⟩ is specified as

`[P:`⟨pos⟩`,`...`,`⟨pos⟩`]`

where `[P:`p_1`,`...`,`p_n`]` denotes the shape obtained by tracking the edge with each p_i a position relative to the object reference point. ⟨vector⟩s and ⟨corner⟩s can be used directly; otherwise use `-p` to get the relative position.

Note: Do not use `{}` or `[]` in the ⟨pos⟩itions.

Bug: The algorithm assumes that the reference point is always inside the polygon.

It is possible to frame polygons is also possible.

```
% Define the polygon as a stylechar so that it is deferred
\xydefcsname@{*stylechar@P@}#1{\Pshape@#1@}
\xynew@{count}\c@poly@count \c@poly@count=\z@ % Polygon identifiers
```

```
\xydef@\Pshape@:#1@{% Strip the mandatory :
  \addtotoks@{%
%              \xy@showthe c{Before parse}\xy@showthe p{Before parse}%
%              \xy@showdim{Before parse}%
              \def\poly@list{@}%
              \poly@parse #1,\relax,%Sets the poly@list
              \def\poly@cache{}%
              \edef\poly@id{\the\c@poly@count}%
              \global\advance\c@poly@count\@ne
              \poly@setEdge
              % Save the old values of Lc for shifting
              \edef\poly@saveLcshape{\the\L@c}%
              \polygonEdge@Outer% Set U,R,D,L
              \poly@setEdge
              % Now shift the object to keep it centered
              \dimen@=\poly@saveLcshape\advance\dimen@-\L@c
              \advance\R@p -\dimen@
%              \xy@showthe c{After parse}%
}}
```

```
% Set the edge
% A polygonal edge is represented as \polygonEdge x1,y1;x2,y2;...;xn,yn@id@
% where xi,yi define the vertices of the polygonal edge as displacement from
% the object reference point (xn=x1,yn=y1), and id is a numerical id used for
% caching the result of the last intersection operation. The cached values for
% polygon id are stored in the global macro poly@cache<id>.
\xydef@\poly@setEdge{\expandafter\poly@setEdge@\poly@list\poly@id}
\xydef@\poly@setEdge@#1@#2{\Edge@c={\polygonEdge#1@#2@}%
                        \expandafter\xdef\csname poly@cache#2%
                                          \endcsname{\poly@cache}}
```

```
% Define the polygon framing operations
\xydefcsname@{frm[P]{-}}{\expandafter\draw@polyframe\the\Edge@c{-}}
\xydefcsname@{frm[P]{.}}{\expandafter\draw@polyframe\the\Edge@c{.}}
\xydefcsname@{frm[P]{=}}{\expandafter\draw@polyframe\the\Edge@c{=}}
```

```
\xydef@\draw@polyframe\polygonEdge#1@#2@#3{%
   \def\poly@dir{#3}%
   \def\poly@list{#1@}%
   \def\poly@id{#2}%
   \edef\poly@cache{\csname poly@cache\poly@id\endcsname}%
   \draw@polygon}
```

```
% A polygon is maintained as a list of relative positions, as follows:
%    poly@list = Xdimen_0,Ydimen_0;...;Xdimen_n,Ydimen_n @
% The list is closed, i.e., the first and last element coincide
% Each polygon has an id storred in \poly@id
% Each polygon caches the result of the last intersection operation
%  as d@x,d@y,x1,y1,x2,y2 where the first two values define the slope
%  and the two sets of points are the two intersection points
%  These values are storred in \poly@cache<id>.
%
```

```
%
% Helper function for dealing with polygon lists

% Map over a polygon list. Make sure you define
% \let\poly@map@next=\poly@map before. Then, you can define
% \let\poly@map@next=\poly@map@stop if you want to stop the traversal

\xydef@\poly@empty{}
\xydef@\poly@map#1#2,#3;#4@{%
   #1{#2}{#3}%
   \def\tmp@{#4}%
   \ifx \poly@empty\tmp@ \else
     \poly@map@next#1#4@%
   \fi
}

\xydef@\poly@map@stop#1@{}

\xydef@\poly@mapExpand#1#2{%
   \edef\tmp@{\noexpand\poly@map\noexpand #1#2}%
   \tmp@}

% Parse a polygon definition

\xydef@\poly@parse #1,{%
 \ifx #1\relax %Done. Copy the head of the list at its end
   \poly@close
   \let\next@=\relax
 \else
   % Save everything
   \save@
      % Process the position using POS
      \POS #1%
      % Now add the new c to the list
      \edef\tmp@{{\the\X@c}{\the\Y@c}}%
      \expandafter\poly@append\tmp@
   \restore
   \let\next@=\poly@parse % continue parsing
 \fi
 \next@
}
              % Append to the list of polygon points
\xydef@\poly@append#1#2{\expandafter\poly@append@\poly@list{#1,#2;}}
\xydef@\poly@append@#1@#2{\global\def\poly@list{#1#2@}}

               % Close the polygon
\xydef@\poly@close{\expandafter\poly@close@\poly@list}
\xydef@\poly@close@#1,#2;#3@{\poly@append{#1}{#2}}

% Draw a polygon at the current location

\xydef@\draw@polygon{% First separate the head
%  \W@{Drawing polygon \poly@list}\xy@showthe c{ Centered at}%
  \save@
    % Set the origin to point at the reference point
```

```
    \X@origin=\X@c \Y@origin=\Y@c
    \poly@setp % Set p at the beginning of the polygon and set poly@rest
    % Zero the edges
    \U@c=\z@\R@c=\z@\D@c=\z@\L@c=\z@\U@p=\z@\R@p=\z@\D@p=\z@\L@p=\z@
    \Edge@c={\zeroEdge}\Edge@p={\zeroEdge}%
    \let\poly@map@next=\poly@map
    \poly@mapExpand\poly@drawseg\poly@rest
  \restore
}

               % Set p to the first element. Requires the origin to be
               % at the reference point
\xydef@\poly@setp{\expandafter\poly@setp@\poly@list}
\xydef@\poly@setp@#1,#2;#3@{%
   \X@p=#1\advance\X@p\X@origin \Y@p=#2\advance\Y@p\Y@origin%
   \global\def\poly@rest{#3@}}

\xydef@\poly@drawseg#1#2{%
  \dimen@=#1\X@c=\the\dimen@\advance\X@c \X@origin
  \dimen@=#2\Y@c=\the\dimen@\advance\Y@c \Y@origin
%  \W@{Next segment is at offset #1,#2 and absolute \the\X@c,\the\Y@c}%
  \expandafter\connect@\expandafter\dir\poly@dir%
  \X@p=\X@c\Y@p=\Y@c}

% Polygonal edge
% Requires c to be the reference point
\xydef@\polygonEdge#1@#2@#3{%
  \def\poly@list{#1@}%
  \def\poly@id{#2}%
  \edef\poly@cache{\csname poly@cache\poly@id\endcsname}%
  \ifcase#3\relax
      \DN@{\polygonEdge@Inters
%          \ifpoly@badinters
%             \xyerror@{Could not find intersection for polygon}\fi
          }%0
  \or \DN@{\polygonEdge@Under}%1
  \or \DN@{\polygonEdge@Dist}%2
  \or \DN@{\rectangleProp@}%3  I do not understand Prop!
  \or \DN@{\polygonEdge@Inner}%4
  \or \DN@{\polygonEdge@Outer}%5
  \else \DN@{}\fi
  \next@}

\newif\ifpoly@badinters

\xydef@\polygonEdge@Inters{%
%  \W@{Edge intersection with (\the\X@c, \the\Y@c) -> (\the\X@p,\the\Y@p)}%
%  \W@{ for polygon: \poly@id}%
%  \W@{ with list: \poly@list}%
%  \W@{ with cache: \poly@cache}%
  % Check the cache first
  \ifx\poly@cache\poly@empty
```

```
    \poly@intersdoit
  \else
    \expandafter\poly@intersprobecache\poly@cache @%
  \fi
%  \xy@showthe c{Inters res}%
}

%
% Redefine poly@cachehit to poly@cachehitdisable to diable polygon caching
\xydef@\poly@cachehit#1#2{%
%    \W@{ intersection point found in cache}%
    \X@c=#1\Y@c=#2}

\xydef@\poly@cachehitdisable#1#2{\poly@intersdoit}
%\let\poly@cachehit=\poly@cachehitdiasable% Uncomment this line to disable $

% Probes the cache
%
\xydef@\poly@intersprobecache#1,#2,#3,#4,#5@{%
  \dimen@=#1\advance\dimen@-\d@X
  \ifdim\zz@\dimen@
    \dimen@=#2\advance\dimen@-\d@Y
    \ifdim\zz@\dimen@
       \poly@cachehit{#3}{#4}
    \else
       \poly@intersdoit
     \fi
   \else
     \poly@intersdoit
   \fi
}

% Computes the intersection between the line cp and the edge.
% Both intersection points are stored in the cache, the one in the
% direction towards p first.
\xydef@\poly@intersdoit{%
%  \W@{ intersection not in cache. Computing it}%
  % Save A@ and B@ because we cannot change them
  \edef\polyoldA@{\the\A@}\edef\polyoldB@{\the\B@}%
  % Initialize the intersection points
  \def\poly@intersneg{}\def\poly@interspos{}%
  \save@
    \poly@setorigin % Set origin at the beginning of the polygon and
                    % set poly@rest
    \def\zeroDivide@{\poly@badinterstrue}% Handle the division by zero except.
    \let\poly@map@next=\poly@map
    \poly@mapExpand\poly@interseg\poly@rest
    \ifx\poly@intersneg\poly@empty
      \poly@badinterstrue
    \else
      \ifx\poly@interspos\poly@empty
```

```
        \poly@badinterstrue
      \else
        \poly@badintersfalse
      \fi
    \fi
  \restore
  \A@=\polyoldA@\B@=\polyoldB@ % Restore A@ and B@
  \ifpoly@badinters \else
    \edef\tmp@{\poly@interspos,\poly@intersneg @}%
    \expandafter\poly@intersfinish\tmp@
  \fi
}

\xydef@\poly@intersfinish#1,#2,#3@{%
  \X@c=#1\Y@c=#2\relax
  \xdef\poly@cache{\the\d@X,\the\d@Y,#1,#2,#3}%
  \poly@setEdge
}

               % Set origin to the first point. Define poly@rest
               % Requires c to be set to the reference point
\xydef@\poly@setorigin{\expandafter\poly@setorigin@\poly@list}
\xydef@\poly@setorigin@#1,#2;#3@{%
   \X@origin=#1\advance\X@origin\X@c \Y@origin=#2\advance\Y@origin\Y@c%
   \global\def\poly@rest{#3@}}


%
% Compute one intersection with a given edge
% c is set to the reference point, p is set to the end of the ray
% origin is set to the start of the segment
\xydef@\poly@interseg#1#2{%
  % Default is no intersection
%  \W@{ Intersection with edge (\the\X@origin,\the\Y@origin) -> (#1,#2)}%
  % Compute the absolute values
  \dimen@=#1\advance\dimen@\X@c \edef\poly@saveXcinters{\the\dimen@}%
  \dimen@=#2\advance\dimen@\Y@c \edef\poly@saveYcinters{\the\dimen@}%
  \save@
    \poly@badintersfalse
    % Set R@c and U@c to the distance to end of segment
    \R@c=\poly@saveXcinters \advance\R@c -\X@origin
    \U@c=\poly@saveYcinters \advance\U@c -\Y@origin
    % Now call intersect to set X@c and Y@c to the intersection point
    \intersect@
    \A@=\X@c \B@=\Y@c % Save result
  \restore
  \ifpoly@badinters
%      \W@{ \space failed after intersect}%
  \else
%      \W@{ \space after intersect@ (\the\A@,\the\B@)}%
      % Verify that it is on the segment from origin to (#1, #2)
```

```
\poly@isonseg\X@origin\Y@origin\A@\B@\poly@saveXcinters\poly@saveYcinters
      \ifpoly@badinters
%          \W@{ \space failed after seg check}%
      \else
        % Verify that (A,B) is on the ray from c to p
        \poly@isonray\X@c\Y@c\A@\B@\X@p\Y@p
        \ifpoly@badinters
           % It is on the negative ray
%           \W@{ \space a negative intersection}%
           \edef\poly@intersneg{\the\A@,\the\B@}%
        \else
%           \W@{ \space a positive intersection}%
           \edef\poly@interspos{\the\A@,\the\B@}%
        \fi
      \fi
  \fi
  % Set the origin to the start of the next segment
  \X@origin=\poly@saveXcinters\Y@origin=\poly@saveYcinters
}

%
% Computes the distance from reference point to the intersection
%
\xydef@\polygonEdge@Dist{\xyerror@{Dist is not yet implemented for polygons}}

%
% Checks that #1 <= #2 <= #3 or that #3 <= #2 <= #1
% Sets ifpoly@badinters otherwise
% ifpoly@closedrange decides whether the second inequality is also checked
% All checks are done with a precision of 100*almostz@ =~ 5000sp = 0.08pt
\newif\ifpoly@closedrange
\xydef@\poly@isinrange#1#2#3{%
  \ifpoly@badinters \else
    \dimen@=#1\dimen@i=#2\dimen@ii=#3\relax
%    \W@{check if in \ifpoly@closedrange closed\else open\fi\space
%                      range  1=\the\dimen@,2=\the\dimen@i,3=\the\dimen@ii}%
    \advance\dimen@ii -\dimen@ \advance\dimen@i -\dimen@
    \ifdim\dimen@ii<0pt\relax
      \ifdim\dimen@i>100\almostz@
          \poly@badinterstrue
      \fi
      \dimen@i=-\dimen@i \dimen@ii=-\dimen@ii
    \else
      \ifdim\dimen@i<-100\almostz@\relax
          \poly@badinterstrue
      \fi
    \fi
    \ifpoly@closedrange
      \advance\dimen@ii 100\almostz@
      \ifdim\dimen@i>\dimen@ii
```

```
        \poly@badinterstrue
      \fi
    \fi
%    \ifpoly@badinters \W@{ failed}\else\W@{ succeeded}\fi
  \fi}
%
% Checks that (#3,#4) is on a segment defined by (#1,#2)->(#5,#6)
\xydef@\poly@isonseg#1#2#3#4#5#6{%
  % Check X first
  \poly@closedrangetrue
  \poly@isinrange{#1}{#3}{#5}%
  \poly@isinrange{#2}{#4}{#6}%
}
%
% Checks that (#3,#4) is on a ray defined by (#1,#2)->(#5,#6)
\xydef@\poly@isonray#1#2#3#4#5#6{%
  % Check X first
  \poly@closedrangefalse
  \poly@isinrange{#1}{#3}{#5}%
  \poly@isinrange{#2}{#4}{#6}%
}
%
% Test whether (Xp,Yp) is inside the polygon (or on the edge)
% Sets \ifInside@ accordingly
\xydef@\polygonEdge@Under{%
%  \W@{}\W@{Edge Under with c=(\the\X@c,\the\Y@c) and p=(\the\X@p,\the\Y@p)}%
%  \W@{ for polygon: \poly@list}%
  % Save A@ and B@
  \edef\polysaveA@under{\the\A@}\edef\polysaveB@under{\the\B@}%
  % Save X@c and Y@c
  \edef\poly@saveXcUnder{\the\X@c}\edef\poly@saveYcUnder{\the\Y@c}%
  % Compute the intersection
  \polygonEdge@Inters
  \ifpoly@badinters% p is very close to c
      \Inside@true
  \else
    \A@=\X@c\B@=\Y@c
    % Restore c
    \X@c=\poly@saveXcUnder\Y@c=\poly@saveYcUnder
    % Now verify that the intersection point is on the ray c->p
    \poly@isonseg\X@c\Y@c\X@p\Y@p\A@\B@
    \ifpoly@badinters \Inside@false \else \Inside@true\fi
    \A@=\polysaveA@under\B@=\poly@saveB@under
  \fi
%  \ifInside@\W@{->inside}\else\W@{->outside}\fi
}
%
% Compute the inner rectangle
```

```
%
\xydef@\polygonEdge@Inner{%
%  \W@{}\W@{Edge Inner with (\the\X@c, \the\Y@c) -> (\the\X@p,\the\Y@p)}%
%  \W@{ for polygon: \poly@list}%
%  \W@{ with cache: \poly@cache}%
  % Save everything except c
  \enter@{\basefromthebase@ \pfromthep@ \DirectionfromtheDirection@}%
    % Save c in Lc,Dc
    \L@c=\X@c \D@c=\Y@c
    % Compute the two intersection points
    \polygonEdge@Inters
    \ifpoly@badinters
       \czeroEdge@
    \else
       % Save the result in Rc,Uc and in X@c,Y@c
       \expandafter\poly@getinterspoints\poly@cache @%
       % Now compute the inner rectangle centered
       \ifdim\X@c>\R@c
         \L@c=\R@c \R@c=\X@c
       \else
         \L@c=\X@c
       \fi
       \X@c=0.5\L@c \advance\X@c 0.5\R@c \advance\R@c -\X@c \L@c=\R@c
       \ifdim\Y@c>\U@c
         \D@c=\U@c \U@c=\Y@c
       \else
         \D@c=\Y@c
       \fi
       \Y@c=0.5\D@c \advance\Y@c 0.5\U@c \advance\U@c -\Y@c \D@c=\U@c
       \Edge@c={\rectangleEdge}%
    \fi
  \leave@
%  \xy@showthe c{After inner}%
}

\xydef@\poly@getinterspoints#1,#2,#3,#4,#5,#6@{%
  \R@c=#3\U@c=#4\X@c=#5\Y@c=#6}

%
% Compute the outer rectangle (set Uc,Dc,Lc,Rc and Edgec).
% Does not change Xc,YC
%
\xydef@\polygonEdge@Outer{%
%  \W@{Computing outer for \poly@list}\xy@showthe c{Before outer}%
  % Save everything except c
  \enter@{\basefromthebase@ \pfromthep@ \DirectionfromtheDirection@}%
    \czeroEdge@ % Zero out c
    \let\poly@map@next=\poly@map
    \poly@mapExpand\poly@findextent\poly@list
    \Edge@c={\rectangleEdge}%
  \leave@
```

```
%  \xy@showthe c{After outer}
}

\xydef@\poly@findextent#1#2{%
%  \xy@showthe c{Before find extent}%
%  \W@{ extx=#1, exty=#2}%
  \dimen@=#1\dimen@=\the\dimen@% it fails if I remove the second assign
  \ifdim\dimen@>\R@c \R@c=\dimen@ \fi
  \ifdim -\dimen@>\L@c \L@c=-\dimen@ \fi
  \dimen@=#2\dimen@=\the\dimen@% it fails if I remove the second assign
  \ifdim\dimen@>\U@c \U@c=\dimen@ \fi
  \ifdim -\dimen@>\D@c \D@c=-\dimen@ \fi
%  \xy@showthe c{ After extent}%
}

% Change to account for polynomial shapes, in addition to circular ones
%
\xydef@\Fshape@#1:{\def\whichframe@@{{#1}}\let\whichoptions@@=\empty
 \DN@{{}}\ifx\whichframe@@\next@ \def\whichframe@@{{-}}\fi
 \expandafter\xyFN@\expandafter\Fshape@whichframe\the\Edge@c}

\xydef@\Fshape@whichframe{%
 \ifx\next\circleEdge
   \edef\whichframe@@{[o]\whichframe@@}%
   \DN@##1{\xyFN@\Fshape@i}%
 \else
 \ifx\next\polygonEdge
   \edef\whichframe@@{[P]\whichframe@@}%
   \DN@\polygonEdge##1@##2@{\xyFN@\Fshape@i}%
 \else
   \DN@##1{\xyFN@\Fshape@i}%
 \fi\fi
 \next@
}

\xyendinput
```

Bug: This code should be merged with the 'frame' and 'poly' options.
The example at the end of §**??** illustrates the extensions.

```
% $Log$
% Received from George.
```

Chapter 3

Features

This chapter describes the options that support facilities that can be obtained using the kernel and extensions yet are much easier to obtain using the provided special syntax.

3.1 All features

Vers. 3.3 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

As a special convenience, this feature loads a subset of XY-pic,[1] namely the extensions: `curve` (*cf.* §2.1), `frame` (§2.2), `cmtip` (§2.3), `line` (§2.4), `rotate` (§2.5), `color` (§2.6), and the following features: `matrix` (§3.5), `arrow` (§3.3), and `graph` (§3.6).

```
%% $Id: xyall.doc,v 3.3 1996/12/19 03:31:56 krisrose Exp $
%%
%% Xy-pic ``All features'' option.
%% Copyright (c) 1993-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

[1] The name 'all' hints at the fact that these were all the available options at the time 'all' was added.

```
\xyprovide{all}{All features}{\stripRCS$Revision: 3.3 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

Here we go:

```
\xyrequire{curve}
\xyrequire{frame}
\xyrequire{cmtip}
\xyrequire{line}
\xyrequire{rotate}
\xyrequire{color}
```

```
\xyrequire{matrix}
\xyrequire{arrow}
\xyrequire{graph}
```

That is all.

```
\xyendinput
```

```
% $Log: xyall.doc,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.11  1994/07/05  10:37:32  kris
% Third 3beta release [bug fixes].
% Experimental graph feature included (for ECCT-94 presentation).
%
% Revision 2.9  1994/06/09  14:59:19  kris
% Release 3beta.
%
% NEW file to go in version 2.9!
```

3.2 Dummy option

Vers. 3.3 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

This option is provided as a template for new options, it provides neither features nor extensions but it does count how many times it is requested. Even though the option does nothing it still has a standard XY-pic header:

```
%% $Id: xydummy.doc,v 3.3 1996/12/19 03:31:56 krisrose Exp $
```

```
%%
%% Xy-pic ``Dummy'' option.
%% Copyright (c) 1993-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{dummy}{Dummy option}{\stripRCS$Revision: 3.3 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

Next follows two declarations that are used to count the number of invocations:

```
\xynew@{count}\xydummyrequires@@
```

This is used by the option to output a message every time it is loaded with `\xyoption` or `\xyrequire`:

```
\xyeveryrequest{dummy}{\global\advance\xydummyrequires@@\@ne
 \W@{The `dummy' option was used \the\xydummyrequires@@\space times!}}
```

(`\W@` is an XY-pic primitive that outputs its argument.)

Then the option ends:

```
\xyendinput
```

Finally `xydummy.doc` is maintained using RCS and thus contains the following revision log:

```
% $Log: xydummy.doc,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.1  1995/09/05 20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
```

```
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% NEW file to go in version 2.7!
```

3.3 Arrow and Path feature

Vers. 3.5 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

This feature provides Xy-pic with the arrow paradigm presented in [12].

Note: `\PATH` command incompatibly changed for version 3.3 (the `\ar` command is unaffected).

The basic concept introduced is the *path*: a connection that *starts* from c (the current object), *ends* at a specified object, and may be split into several *segments* between intermediate specified objects that can be individually labelled, change style, have breaks, etc.

§3.3.1 is about the `\PATH` primitive, including the syntax of paths, and §3.3.2 is about the `\ar` customisation of paths to draw arrows using Xy-pic directional objects.

Header:

```
%% $Id: xyarrow.doc,v 3.5 1997/05/28 13:05:01 krisrose Exp $
%%
%% Xy-pic ``Arrows and Paths'' feature.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{arrow}{Arrow and Path feature}{\stripRCS$Revision: 3.5 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
```

Syntax			Action
`\PATH` ⟨path⟩			interpret ⟨path⟩
`\afterPATH{`⟨decor⟩`}` ⟨path⟩			interpret ⟨path⟩ and then run ⟨decor⟩
⟨path⟩	⟶	`~` ⟨action⟩ `{` ⟨stuff⟩ `}` ⟨path⟩	set ⟨action⟩[3.3a] to ⟨stuff⟩
	|	`~` ⟨which⟩ `{` ⟨labels⟩ `}` ⟨path⟩	add ⟨labels⟩ prefix for some segments[3.3b]
	|	`~` `{` ⟨stuff⟩ `}` ⟨path⟩	set failure continuation[3.3c] to ⟨stuff⟩
	|	`'` ⟨segment⟩ ⟨path⟩	make straight segment[3.3d]
	|	`` ` `` ⟨turn⟩ ⟨segment⟩ ⟨path⟩	make turning segment[3.3f]
	|	⟨segment⟩	make last segment[3.3g]
⟨turn⟩	⟶	⟨diag⟩ ⟨turnradius⟩	1/4 turn[3.3f] starting in ⟨diag⟩
	|	⟨cir⟩ ⟨turnradius⟩	explicit turn[3.3f]
⟨turnradius⟩	⟶	⟨empty⟩	use default turn radius
	|	`/` ⟨dimen⟩	set *turnradius* to ⟨dimen⟩
⟨segment⟩	⟶	⟨path-pos⟩ ⟨slide⟩ ⟨labels⟩	segment[3.3e] with ⟨slide⟩ and ⟨labels⟩
⟨slide⟩	⟶	⟨empty⟩ | `<` ⟨dimen⟩ `>`	optional slide[3.3h]: ⟨dimen⟩ in the "above" direction
⟨labels⟩	⟶	`^` ⟨anchor⟩ ⟨it⟩ ⟨alias⟩ ⟨labels⟩	label with ⟨it⟩[3.3i] *above* ⟨anchor⟩
	|	`_` ⟨anchor⟩ ⟨it⟩ ⟨alias⟩ ⟨labels⟩	label with ⟨it⟩[3.3i] *below* ⟨anchor⟩
	|	`|` ⟨anchor⟩ ⟨it⟩ ⟨alias⟩ ⟨labels⟩	break with ⟨it⟩[3.3j] at ⟨anchor⟩
	|	⟨empty⟩	no more labels
⟨anchor⟩	⟶	`-` ⟨anchor⟩ | ⟨place⟩	label/break placed relative to the ⟨place⟩ where `-` is a synonym for `<>(.5)`
⟨it⟩	⟶	⟨digit⟩ | ⟨letter⟩ | `{`⟨text⟩`}` | ⟨cs⟩	⟨it⟩ is a default label[3.3k]
	|	`*` ⟨object⟩	⟨it⟩ is an ⟨object⟩
	|	`@` ⟨dir⟩	⟨it⟩ is a ⟨dir⟩ectional
	|	`[` ⟨shape⟩ `]` ⟨it⟩	use `[`⟨shape⟩`]` for ⟨it⟩
⟨alias⟩	⟶	⟨empty⟩ | `="`⟨id⟩`"`	optional name for label object[3.3l]

Figure 3.1: ⟨path⟩s

```
{BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
 DK--8000 Aarhus~C}
```

3.3.1 Paths

```
\message{path,}
```

The fundamental commands of this feature are `\PATH` and `\afterPATH` that will parse the ⟨path⟩ according to the grammar in figure 3.1 with notes below.

Parsing: `\afterPATH` sets up `\afterPATH@` and initialise all the actions (see note 3.3a below) before invoking the actual ⟨path⟩ parser. `\PATH` is just a dummy interface.

```
\xydef@\PATH{\afterPATH{}}
```

```
\xylet@\afterPATH@=\empty
```

```
\xydef@\afterPATH#1{\save
 \DN@##1{\def\afterPATH@{\restore \def\afterPATH@{##1}#1}}%
 \expandafter\next@\expandafter{\afterPATH@}%
 \let\PATHfail@@=\empty
 \let\PATHbefore@@=\empty
 \let\PATHafter@@=\empty
 \let\PATHlabelsevery@@=\empty
 \let\PATHlabelsnext@@=\empty
 \let\PATHlabelslast@@=\empty
 \xy@{\afterPATH{#1}}{\def\PATHslide@@{\z@}}%
 \def\PATHcontinue@@{\xyFN@\PATH@}%
 \xyFN@\PATH@}
```

`\PATH@` is the parser for ⟨path⟩: it eats the ⟨path⟩ until it fails and then calls the 'failure continuation' which will eventually become `\empty` which ends the parsing. **Hack:** The '`\ifPATHsingle@`' switch switches the ⟨segment⟩ parsing *off*; this is used by the graph feature to parse stand-alone ⟨slide⟩ ⟨labels⟩ sequences.

```
\xynew@{if}\ifPATHsingle@
```

```
\xydef@\PATH@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PATH@}%
 \else\ifPATHsingle@ \let\next@=\PATH@single
 \else \let\next@=\PATH@normal \fi\fi \next@}
```

```
\xydef@\PATH@normal{%
 \ifx ~\next \DN@ ~{\xyFN@\PATHsetting@}%
 \else \addRQ@\ifx \next \addRQ@\DN@{\xy@'{}\xyFN@\PATHstraight@}%
 \else \addLQ@\ifx \next \addLQ@\DN@{\xy@`{}\xyFN@\PATHturn@}%
 \else\ifx \PATHfail@@\PATH@x \DN@{\xyFN@\PATHfinal@}%
 \else
  \DNii@{\let\PATHfail@@=\PATH@x \xyFN@\PATH@}%
  \DN@{\expandafter\nextii@\PATHfail@@}%
 \fi\fi\fi\fi \next@}
```

`\PATHsetting@` just reads an ⟨action⟩ or ⟨which⟩ determining which action stuff to set and then does it—we treat the failure continuation as an action here even though it is not, strictly speaking. The four actions are explained in the appropriate notes below.

```
\xylet@\PATHfail@@=\empty
```

```
\xylet@\PATHbefore@@=\empty
\xylet@\PATHafter@@=\empty
```

```
\xylet@\PATHlabelsevery@@=\empty
\xylet@\PATHlabelsnext@@=\empty
\xylet@\PATHlabelslast@@=\empty
```

```
\xydef@\PATHsetting@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PATHsetting@}%
 \else\ifx \bgroup\next
  \DN@##1{\xy@{~{##1}}{}\def\PATHfail@@{##1}\xyFN@\PATH@}%
 \else \addEQ@\ifx \next
  \addEQ@\DN@##1{\xy@{~={##1}}{}\def\PATHbefore@@{##1}\xyFN@\PATH@}%
```

```
 \else \addLT@\ifx \next
  \addLT@\DN@##1{\xy@{~<{##1}}{}\def\PATHlabelsnext@@{##1}\xyFN@\PATH@}%
 \else \addGT@\ifx \next
  \addGT@\DN@##1{\xy@{~>{##1}}{}\def\PATHlabelslast@@{##1}\xyFN@\PATH@}%
 \else \addPLUS@\ifx \next
  \addPLUS@\DN@##1{\xy@{~+{##1}}{}\def\PATHlabelsevery@@{##1}\xyFN@\PATH@}%
 \else \ifx /\next
  \DN@/##1{\xy@{~/{##1}}{}\def\PATHafter@@{##1}\xyFN@\PATH@}%
 \else \addDASH@\ifx \next
  \xywarning@{Obsolete `-' PATH <action> translated to `='.}%
  \addDASH@\DN@##1{\xy@{~-{##1}}{}\def\PATHbefore@@{##1}\xyFN@\PATH@}%
 \else
  \xyerror@{Unknown \string~ setting: \meaning\next}%
         {See the Xy-pic arrow feature documentation for help.}%
 \fi\fi\fi\fi\fi\fi\fi\fi \next@}
```

Next the setup for straight segments including the last: they set `\PATHinit@@` to initialise, *i.e.*, slide, `\PATHextra@@` to do what is needed after the `=` action, and finally `\PATHpost@@` is set to any operations to be done after the entire segment is typeset but before the next is read. `\PATHcontinue@@` is used internally to distinguish the ordinary segments from the last.

```
\xylet@\PATHinit@@=\empty

\xylet@\PATHextra@@=\empty
\xylet@\PATHpost@@=\empty
\xylet@\PATHcontinue@@=\empty

\xydef@\PATHstraight@{%
 \def\PATHinit@@{\PATHinitstraight@}%
 \let\PATHextra@@=\empty
 \let\PATHpost@@=\empty
 \let\PATHlabelsextra@@=\relax
 \xy@@\pfromc@ \PATHafterPOS{\xyFN@\PATHsegment@}}

\xydef@\PATHfinal@{%
 \def\PATHinit@@{\PATHinitstraight@}%
 \def\PATHextra@@{\let\PATHcontinue@@=\afterPATH@}%
 \let\PATHpost@@=\empty
 \let\PATHlabelsextra@@=\PATHlabelsextralast@
 \xy@@\pfromc@ \PATHafterPOS{\xyFN@\PATHsegment@}}

\xydef@\PATHlabelsextralast@{\let\PATHlabelsextra@@=\relax
 \expandafter\xyFN@\expandafter\PATHlabels@\PATHlabelslast@@}

\xydef@\PATHinitstraight@{\xy@@{\setupDirection@ \dimen@=\PATHslide@@
   \dimen@ii=-\sinDirection\dimen@
   \ifPATHomitslide@@\else\advance\X@p\dimen@ii\fi \advance\X@c\dimen@ii
   \dimen@ii= \cosDirection\dimen@
   \ifPATHomitslide@@\else\advance\Y@p\dimen@ii\fi \advance\Y@c\dimen@ii
   \PATHomitslide@@false \resetupDirection@}}
```

Setup for turning segments is in note 3.3f.

A ⟨segment⟩ is interpreted as follows after p has been set to the previous end object, c to the ⟨pos⟩ given at the start of the segment, and `\PATHslide@@` to the slide:

```
\xydef@\PATHslide@@{\z@}
\xynew@{if}\ifPATHomitslide@@
```

```
\xydef@\PATHsegment@{%
 \addLT@\ifx\next
  \addGT@{\addLT@\DN@##1}{%
   \xy@{<##1>}{\dimen@=##1\relax \edef\PATHslide@@{\the\dimen@}}%
   \xyFN@\PATHsegment@@}%
 \else \let\next@=\PATHsegment@@
 \fi \next@}
```

1. Expand `\PATHinit@@` (for straight segments this just sets up the direction and applies the ⟨slide⟩).

2. Expand `\PATHaction={`⟨stuff⟩`}` as set using `~={`⟨stuff⟩`}`.

3. Sets `\PATHcontinue@@` to any continuation set with `~{`⟨stuff⟩`}`).

4. Construct list of ⟨labels⟩ specified by the user through `~`⟨which⟩ setup. Those applicable to the present segment are inserted before the user's ⟨labels⟩ in the sequence `<>=`.

5. Store p, c as *start*,*end* of segment.

6. Interpret ⟨labels⟩ (see below), including the ones added by the `~`⟨when⟩... ⟨setup⟩.

`\PATHsegment@@` performs this except the last two points:

```
\xydef@\PATHsegment@@{\PATHinit@@ \addEQ@\PATHaction@\PATHbefore@@ \PATHextra@@
 \expandafter\toks@\expandafter{\PATHlabelsnext@@}\let\PATHlabelsnext@@=\empty
 \expandafter\addtotoks@\expandafter{\PATHlabelsevery@@}%
 \expandafter\def\expandafter\PATHlabels@@\expandafter{\the\toks@}%
 \toks@={}\expandafter\xyFN@\expandafter\PATHlabels@\PATHlabels@@}
```

`\PATHlabels@` parses all ⟨labels⟩ of a ⟨segment⟩. This sets `\PATHlabelit@@` to the operation building the label in question. Then some parsing stores the tokens `<>(.5)` for each `-` in the ⟨anchor⟩ and finally passes control to the kernel ⟨place⟩ parser with these tokens as the head. This then in turn calls `\PATHit@` that parses ⟨it⟩ and applies the chosen operation. After the last we continue with the accumulated subsegment actions.

```
\xylet@\PATHlabelit@@=\empty
```

```
\xydef@\PATHlabels@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PATHlabels@}%
 \else \ifx ^\next
  \DN@##1{\xy@^{}\let\PATHlabelit@@=\PATHlabelabove@@
   \DNii@{}\xyFN@\PATHanchor@}%
 \else \ifx _\next
  \DN@##1{\xy@_{}\let\PATHlabelit@@=\PATHlabelbelow@@
   \DNii@{}\xyFN@\PATHanchor@}%
 \else \ifx |\next
  \DN@##1{\xy@|{}\let\PATHlabelit@@=\PATHlabelbreak@
   \DNii@{}\xyFN@\PATHanchor@}%
 \else \let\next@=\PATHfinishsegments@ \fi\fi\fi\fi \next@}
```

```
\xydef@\PATHanchor@{\begingroup \toks@={}\PATHanchor@i}
```

```
\xydef@\PATHanchor@i{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PATHanchor@i}%
 \else\addDASH@\ifx \next
  \addDASH@\DN@{\expandafter\addtotoks@\expandafter{\PATHanchor@toks}%
   \xyFN@\PATHanchor@i}%
 \else
  \DNii@##1{\endgroup\afterPLACE{\xyFN@\PATHit@}##1}%
  \DN@{\expandafter\nextii@\expandafter{\the\toks@}}%
 \fi\fi \next@}
```

```
{\xyuncatcodes \gdef\next{<>(.5)}}
\xylet@\PATHanchor@toks=\next
```

```
\xylet@\PATHitshapes@@=\empty
```

```
\xydef@\PATHit@{\let\PATHitshape@@=\empty \xyFN@\PATHit@i}
```

```
\xydef@\PATHit@i{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PATHit@i}%
 \else \ifx *\next \DN@*##1##{\PATHlabelit@@{!C##1}}%
 \else \addAT@\ifx\next \addAT@\DN@##1##{\PATHlabelit@@{\dir##1}}%
 \else \ifx [\next \DN@[##1]{%
  \expandafter\def\expandafter\PATHitshape@@\expandafter{\PATHitshape@@[##1]}%
  \xyFN@\PATHit@i}%
 \else \DN@{\PATHlabelit@@{}}%
 \fi\fi\fi\fi \next@}
```

The individual operations bound to `\PATHlabelit@@` are discussed as appropriate in the notes. After the last break and label we typeset the last piece of the connection.

```
\xydef@\PATHfinishsegments@{%
 \ifx\PATHlabelsextra@@\relax \expandafter\PATHfinishsegments@i
 \else
  \expandafter\PATHlabelsextra@@ \fi}
```

```
\xydef@\PATHfinishsegments@i{%
 \xy@@{\Clast@@}\the\toks@ \toks@={}%
 \xy@@\setupDirection@
 \PATHpost@@ \PATHaction@/\PATHafter@@
 \PATHcontinue@@}
```

Bug: The order of the breaks determines the order of the subsegments. This is maybe a feature.

Notes

3.3a. An ⟨action⟩ can be either of the characters `=/`. The associated ⟨stuff⟩ is saved and used to call

`\PATHaction`⟨action⟩`{`⟨stuff⟩`}`

before and *after* each segment (including all ⟨labels⟩) for `=` and `/`, respectively.

The default `\PATHaction` macro just expands to "`\POS` ⟨stuff⟩ `\relax`" thus ⟨stuff⟩ should be of the form ⟨pos⟩ ⟨decor⟩. The user can redefine this—in fact the `\ar` command described in §3.3.2 below is little more than a special `\PATHaction` command and a clever defaulting mechanism.

Here is the default `\PATHaction`:

```
\xydef@\PATHaction@default#1#2{\xy@{PATHaction#1{#2}}{}\POS#2\relax}
```

```
\xylet@\PATHaction=\PATHaction@default
```

It is called using `\PATHaction@` to expand the action control sequences back to their ⟨stuff⟩ first:

```
\xydef@\PATHaction@#1#2{\expandafter\PATHaction\expandafter#1\expandafter{#2}}
```

3.3b. It is possible to include a number of default ⟨labels⟩ *before* the ⟨labels⟩ of the actual ⟨segment⟩ are interpreted, using `~`⟨which⟩`{`⟨labels⟩`}`. The specified ⟨which⟩ determines for which segments the indicated ⟨labels⟩ should be prefixed as follows:

⟨which⟩	applied to...
`<`	next segment only
`>`	last segment only
`=`	every segment

(when several apply to the same segment they are inserted in the sequence `<>+`).

This is useful to draw connections with a 'center marker' in particular with arrows, *e.g.*, the 'mapsto' example explained below can be changed into a 'breakto' example: typing

```
\xy*+{0}\PATH
 ~={**\dir{-}}
 ~>{|>*\dir{>}}
 ~+{|*\dir{/}}
 '(10,1)*+{1} '(20,-2)*+{2} (30,0)*+{3}
\endxy
```

will typeset

Note, however, that what goes into `~+{...}` is ⟨labels⟩ and thus not a ⟨pos⟩ – it is not an action in the sense explained above.

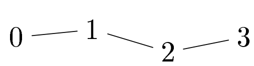
3.3c. Specifying `~{`⟨stuff⟩`}` will set the "failure continuation" to ⟨stuff⟩. This will be inserted when the last ⟨segment⟩ is expected—it can even replace it or add more ⟨segment⟩s, *i.e.*,

```
\xy *+{0} \PATH ~={**\dir{-}}
  ~{'(20,-2)*+{2} (30,0)*+{3}} '(10,1)*+{1}
\endxy
```

is equivalent to

```
\xy *+{0} \PATH ~={**\dir{-}}
  '(10,1)*+{1} '(20,-2)*+{2} (30,0)*+{3}
\endxy
```

typesetting



because when `\endxy` is seen then the parser knows that the next symbol is neither of the characters `~'`` ` and hence that the last ⟨segment⟩ is to be expected. Instead, however, the failure continuation is inserted and parsed, and the ⟨path⟩ is finished by the inserted material.

Failure continuations can be nested:

```
\xy *+{0} \PATH ~={**\dir{-}}
```

```
 ~{~{(30,0)*+{3}}
 '(20,-2)*+{2}} '(10,1)*+{1}
\endxy
```

will also typeset the connected digits.

3.3d. A "straight segment" is interpreted as follows:

1. First p is set to the end object of the previous segment (for the first segment this is c just before the path command) and c is set to the ⟨pos⟩ starting the ⟨segment⟩, and the current ⟨slide⟩ is applied.
2. Then the `=` and `<` *segment actions* are expanded (in that sequence) and the `<` action is cleared. The resulting p and c become the *start* and *end* object of the segment.
3. Then all ⟨labels⟩ (starting with the ones defined as described in note 3.3b below).

The code expanding the actions is part of the parsing above.

3.3e. A *segment* is a part of a ⟨path⟩ between a previous and a new *target* given as a ⟨path-pos⟩: normally this is just a ⟨pos⟩ as described in §1.3 but it can be changed to something else by changing the control sequence `\PATHafterPOS` to be something other than `\afterPOS`.

```
\xylet@\PATHafterPOS@default=\afterPOS
\xylet@\PATHafterPOS=\PATHafterPOS@default
```

3.3f. A *turning* segment is one that does not go all the way to the given ⟨pos⟩ but only as far as required to make a turn towards it. The c is set to the actual turn object after a turning segment such that subsequent turning or other segments will start from there, in particular the last segment (which is always straight) can be used to finish a winding line.

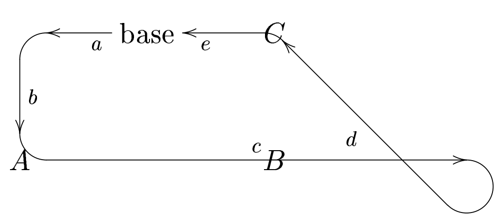
What the turn looks like is determined by the ⟨turn⟩ form:

⟨empty⟩ Nothing between the ` and the ⟨pos⟩ is interpreted the same as giving just the ⟨diag⟩ last used *out* of a turn.

⟨diag⟩ Specifying a single ⟨diag⟩ d is the same as specifying either of the ⟨cir⟩cles d`^` or d`_`, depending on whether the specified ⟨pos⟩ has its center 'above' or 'below' the line from p in the ⟨diag⟩onal direction.

⟨cir⟩ When a full explicit ⟨cir⟩cle is available then the corresponding ⟨cir⟩cle object is placed such that its ingoing direction is a continuation of a straight connection from p and the outgoing direction points such that a following straight (or last) segment will connect it to c (with the same slide).

Here is an example using all forms of ⟨turn⟩s:



was typeset by

```
\xy <4pc,0pc>:(0,0)
 *+\txt{base}="base"
```

```
 \PATH ~={**\dir{-}?>*\dir{>}}
       `l   (-1,-1)*{A} ^a
       `     (1,-1)*{B} ^b
       `_ul  (1, 0)*{C} ^c
       `ul^l "base"     ^d
             "base"     ^e
\endxy
```

Bug: Turns are only really resonable for paths that use straight lines like the one above.

Note: Always write a valid ⟨pos⟩ after a ⟨turn⟩, otherwise any following `^` or `_` labels can confuse the parser. So if you intend the `^r` in `` `^r `` to be a label then write `` `,^r ``, using a dummy `,` ⟨pos⟩ition.

We need to keep track of the current *in* and *out* ⟨diag⟩ directions as well as the current radius.

```
\xydef@\PATHlastout@@{3}
```

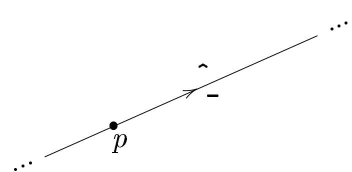
The startup routine is just a selection of the two kinds of ⟨turn⟩:

```
\xydef@\PATHturn@{\afterCIRorDIAG\PATHturn@cir\PATHturn@diag}
```

In both cases the trick is to set up a `\PATHinit@@` method that typesets the right circle segment at the right location. The simplest one is when there is a full ⟨cir⟩ available because that should just be used directly. The only complication is that we forbid 180° turns:

```
\xydef@\PATHturn@cir{\toks@={\xy@@{%
  \count@=\CIRin@@ \ifnum\count@<4\else\advance\count@-4\fi
  \count@@=\CIRout@@ \ifnum\count@@<4\else\advance\count@@-4\fi
  \ifnum\count@=\count@@ \xyerror@{<turn> cannot be half or full}{%
You asked for a <turn>ed segment with parallel in- and out-direction.^^J%
This is not allowed because it is not possible to position it uniquely.}\fi}}%
 \edef\next@{{\CIRin@@}{\expandafter\noexpand\CIRorient@@}{\CIRout@@}}%
 \expandafter\PATHturn@i\next@}
```

Procedure for ⟨diag⟩ *turns*: Builds initial part of `\PATHinit@@` that computes the missing *orient* and *out* from *in* and the *c* passed through the segment. This initial segment should set *orient* as follows:



i.e., as the sign of the inner product $\vec{pc} \cdot \hat{in}$, and set *out* as for the ⟨cir⟩cle *in orient*.

```
\xydef@\PATHturn@diag{\toks@={\xy@@{%
   \setupDirection@ \count@=\CIRin@@ \dimen@=\xydashl@ \ABfromdiag@
   \ifdim \sinDirection\A@>\cosDirection\B@ \def\CIRorient@@{\CIRacw@}%
    \advance\count@\ifnum\count@<6 \tw@ \else -6\fi
   \else \def\CIRorient@@{\CIRcw@}%
    \advance\count@\ifnum\count@>\@ne -\tw@ \else 6\fi
   \fi
   \edef\CIRout@@{\the\count@}}}%
 \edef\next@{{\CIRin@@}{}{}}%
 \expandafter\PATHturn@i\next@}
```

`\PATHturn@i` is where `\PATHinit@@` is defined to do the following:

l.775 Setup the default *in*, *orient*, and *out* as defined by the call from the above macros. Set *radius* parameter to the current *turnradius* value. The commands in `\toks@` are run to do any last-minute fixing of the defaults.

l.779 The segment will leave p in direction *in* and pretend to be meeting c in direction *out*. Thus the center of the ⟨cir⟩cle object to be used for the actual ⟨turn⟩ object is located at the intersection of the line through $p + \mathit{radius}\hat{v}_{\mathit{in}}$ in direction $\vec{v}_{\mathit{in}}$ and the line through $c + \mathit{radius}\hat{v}_{\mathit{out}}$ in direction $\vec{v}_{\mathit{out}}$. Here we compute $p + R\hat{v}_{\mathit{in}}$ using $R = \mathit{radius}$ if this is the initial turn and $R = \mathit{radius} \pm \mathit{slide}$ if it is a continuation turn.

l.782 Save p and *base* for later.

l.785 Compute $c + \mathit{radius}\hat{v}_{\mathit{out}}$ and use the kernel `\intersect@` primitive to compute the turn circle segment center.

l.793 However, if the intersection point lies on the 'wrong side' of p, *i.e.*, if $\vec{pc} \cdot \mathit{in}$ is negative, then we should not use the intersection point but the point as close to p as possible because the connection from p to the turn should at least have zero length—in fact we set it to have 8sp length to make sure that a sugsegment is typeset!

l.806 Modify *radius* to use for the actual circle segment by the *slide*: either positively (for *orient* `_`) or negatively (for *orient* `^`), and drop the circle segment (bypassing `\cir` to use the internal settings from above).

l.809 Store in `\PATHpostpos@@` the true end of the circle segment and code to prevent the application of the *slide* for the start of the next ⟨segment⟩.

l.815 Move c to true beginning of the circle segment.

l.818 Restore saved p and *base*, and slide p finally.

```
771 \xylet@\PATHpostpos@@=\empty
773 \xydef@\PATHturn@i#1#2#3{%
774  \DN@##1{\def\PATHinit@@{\xy@@{%
775     \def\CIRin@@{#1}\def\CIRorient@@{#2}\def\CIRout@@{#3}
776     \ifnum\CIRin@@=8 \let\CIRin@@=\PATHlastout@@ \fi
777     \R@=\turnradius@
778     ##1\relax}%
779    \xy@@{\count@=\CIRin@@
780     \ifPATHomitslide@@ \dimen@=-\PATHslide@@
781      \ABfromdiag@ \advance\X@p-\B@ \advance\Y@p\A@ \fi
782     \enter@{\pfromthep@ \basefromthebase@}}
783    \xy@@{\dimen@=\expandafter\ifx\CIRorient@@\CIRcw@-\fi\R@
784     \ABfromdiag@ \advance\X@p-\B@ \advance\Y@p\A@
785     \X@origin=\X@p \Y@origin=\Y@p}
786    \xy@@{\dimen@=\xydashl@ \ABfromdiag@ \R@c=\A@ \U@c=\B@
787     \count@=\CIRout@@
788     \dimen@=\expandafter\ifx\CIRorient@@\CIRcw@-\fi\R@
789     \ABfromdiag@ \advance\X@c-\B@ \advance\Y@c\A@
790     \dimen@=\xydashl@ \X@p=\X@c \Y@p=\Y@c
791     \ABfromdiag@ \advance\X@p-\A@ \advance\Y@p-\B@
792     \intersect@
793     \count@=\CIRin@@ \dimen@=\p@ \ABfromdiag@}
794    \xy@@{\edef\next@{\expandafter\removePT@\the\A@}%
```

```
   \edef\nextii@{\expandafter\removePT@\the\B@}%
   \A@=\X@c \advance\A@-\X@origin \B@=\Y@c \advance\B@-\Y@origin
   \ifdim \next@\A@<-\nextii@\B@
    \dontleave@ \cfromp@
    \count@=\CIRin@@ \advance\count@\ifnum\count@<4 +4\else -4\fi
    \dimen@=\xydashl@ \Directionfromdiag@
    \the\Edge@c\z@
    \count@=\CIRin@@ \dimen@=\expandafter\ifx\CIRorient@@\CIRcw@-\fi\R@
    \ABfromdiag@ \advance\X@c-\B@ \advance\Y@c\A@
    \dimen@=5sp \ABfromdiag@ \advance\X@c\A@ \advance\Y@c\B@
   \fi
   \dimen@ii=\expandafter\ifx\CIRorient@@\CIRacw@-\fi\PATHslide@@
   \advance\R@\dimen@ii
   \drop@\literal@{\hbox\bgroup\cir@i}}%
  \xy@@{\X@p=\X@c \Y@p=\Y@c
   \count@=\CIRout@@
   \dimen@=\expandafter\ifx\CIRorient@@\CIRacw@-\fi\R@c
   \ABfromdiag@ \advance\X@p-\B@ \advance\Y@p\A@
   \edef\PATHpostpos@@{\X@c=\the\X@p \Y@c=\the\Y@p \noexpand\czeroEdge@
    \noexpand\PATHomitslide@@true}}%
  \xy@@{\count@=\CIRin@@
   \dimen@=\expandafter\ifx\CIRorient@@\CIRacw@-\fi\R@c
   \ABfromdiag@ \advance\X@c-\B@ \advance\Y@c\A@ \czeroEdge@}%
  \xy@@{\leave@
   \count@=\CIRin@@ \dimen@=\PATHslide@@
   \ABfromdiag@ \advance\X@p-\B@ \advance\Y@p\A@
   \edef\PATHlastout@@{\CIRout@@}%
   \count@=\CIRout@@ \dimen@=\xydashl@ \Directionfromdiag@}}}%
 \expandafter\next@\expandafter{\the\toks@}\toks@={}%
 \let\PATHextra@@=\empty
 \def\PATHpost@@{\xy@@\PATHpostpos@@}%
 \let\PATHlabelsextra@@=\relax
 \xyFN@\PATHturn@ii}
```

The final macro takes care of the v2.6 format for changing *turnradius*: using `/`⟨dimen⟩ right after the ⟨turn⟩.

```
\xydef@\PATHturn@ii{%
 \ifx /\next
  \DN@ /{\afterassignment\nextii@\dimen@=}%
  \DNii@{%
   \edef\next@{\noexpand\xy@@{\edef\noexpand\turnradius@{\the\dimen@}}}\next@
   \xy@@\pfromc@ \PATHafterPOS{\xyFN@\PATHsegment@}}%
 \else \DN@{\xy@@\pfromc@ \PATHafterPOS{\xyFN@\PATHsegment@}}\fi
 \next@}
```

This just stores a ⟨diag⟩ vector of the specified lenght in `<`A`,`B`>` thus acting as an optimised version of `\Directionfromdiag@`:

```
\xydef@\ABfromdiag@{\ifcase\count@\relax
     \A@=-.7071\dimen@ \B@=-.7071\dimen@ \or \A@=\z@      \B@=-\dimen@
```

```
 \or \A@=+.7071\dimen@ \B@=-.7071\dimen@ \or \A@=\dimen@  \B@=\z@
 \or \A@=+.7071\dimen@ \B@=+.7071\dimen@ \or \A@=\z@      \B@=+\dimen@
 \or \A@=-.7071\dimen@ \B@=+.7071\dimen@ \or \A@=-\dimen@ \B@=\z@
 \else\xybug@{impossible <diag>?}\fi}
```

The default used for *turnradius* can be set by the operation

---

`\turnradius` ⟨add op⟩ `{`⟨dimen⟩`}`

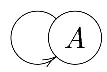
---

that works like the kernel `\objectmargin` etc. commands; it defaults to 10pt.

```
\xydef@\turnradius@{10pt}
```

```
\xydef@\turnradius{\afterADDOP{\Addop@@\turnradius@}}
```

Exercise 3.1: Typeset



using ⟨turn⟩s.

3.3g. The last segment is exactly as a straight one except that the `>` action (if any) is executed (and cleared) just after the `<` action.

(The code implementing this is merged with the parsing.)

3.3h. "Sliding" a segment means moving each of the p, c objects in the direction perpendicular to the current direction at each.

(The code implementing this is merged with the parsing, using the `\PATHslide@@` control sequence to hold the current slide.)

3.3i. Labelling means that ⟨it⟩ is dropped relative to the current segment using a `?` ⟨pos⟩ition. This thus depends on the user setting up a connection with a `**` ⟨pos⟩ as one of the actions—typically the `=` action is used for this (see note 3.3d for the details). The only difference between `^` and `_` is that they shift the label in the `^` respectively `_` direction; for straight segments it is placed in the "superscript" or "subscript" position.

The operations used by `\PATHlabels@` to put labels above and below (with `^` and `_`) are trivial, taking two arguments and placing the label immediately; for convenience the `\drop` head are named separately such that they may be redefined:

```
\xydef@\PATHlabelabove@#1#2{\droplabel@\belowDirection@{#1}{#2}%
 \let\afteraliases@@=\empty \xyFN@\PATHlabelalias@}
```

```
\xydef@\PATHlabelbelow@#1#2{\droplabel@\aboveDirection@{#1}{#2}%
 \let\afteraliases@@=\empty \xyFN@\PATHlabelalias@}
```

```
\xylet@\PATHlabelabove@@=\PATHlabelabove@
\xylet@\PATHlabelbelow@@=\PATHlabelbelow@
```

```
\xydef@\droplabel@#1#2#3{\xy@@{\enter@\DirectionfromtheDirection@}%
 \DN@{#2}\ifx\next@\empty
  \expandafter\xy@@ix@\expandafter{\PATHitshape@@\labelbox{#3}}%
 \else \expandafter\xy@@ix@\expandafter{\PATHitshape@@#2{#3}}\fi
 \xy@@{\setbox\z@=\expandafter\object\the\toks9 %
  \advance\L@c\labelmargin@ \advance\R@c\labelmargin@
  \advance\D@c\labelmargin@ \advance\U@c\labelmargin@
```

```
  \setboxz@h{\kern\labelmargin@\boxz@\kern\labelmargin@}%
  \A@=\X@c \B@=\Y@c #1\xydashl@ \the\Edge@c\thr@@
  \advance\A@-\X@c \advance\B@-\Y@c \advance\X@c 2\A@ \advance\Y@c 2\B@
  \ht\z@=\U@c \dp\z@=\D@c \dimen@=\L@c \advance\dimen@\R@c \wd\z@=\dimen@
  \setbox\lastobjectbox@=\hbox{\X@c=\L@c \Y@c=\z@ \Drop@@}%
  \drop@{\box\lastobjectbox@}{}%
  \leave@}}
```

```
\xydef@\PATHlabelbreak@#1#2{%
 \DN@{#1}\ifx\next@\empty \drop\labelbox{#2}\else \drop#1{#2}\fi
 \def\afteraliases@@{\xy@@\Cbreak@@}\xyFN@\PATHlabelalias@}
```

These emulate the effect of the ⟨modifier⟩s `!C +<2`*labelmargin*`>` ⟨perp⟩ `!P` ⟨unperp⟩ where ⟨perp⟩,⟨unperp⟩ are `^`,`_` for `^`-labels and `_`,`^` for `_`-labels. The last hack using `\lastobjectbox@` aims at both setting that right and setting the label as an object (such that modifiers take effect) even though its placement is peculiar.

Labels will be separated from the connection by the *labelmargin* that you can set with the operation

`\labelmargin` ⟨add op⟩ `{`⟨dimen⟩`}`

that works like the kernel `\objectmargin` command; in fact *labelmargin* defaults to use *objectmargin* if not set.

```
\xylet@\labelmargin@=\objectmargin@
```

```
\xydef@\labelmargin{\afterADDOP{\Addop@@\labelmargin@}}
```

3.3j. Breaking means to "slice a hole" in the connection and insert ⟨it⟩ there. This is realized by typesetting the connection in question in *subsegments*, one leading to the break and one continuing after the break as described in notes 3.3a and 3.3d.

The special control sequence `\hole` is provided to make it easy to make an empty break.

```
\xydef@\hole{\hbox{\dimen@=\objectmargin@ \kern2\dimen@
  \vrule height\dimen@ depth\dimen@ width\z@}}
```

Subsegments are typeset using the '`\Cbreak@@`' method.

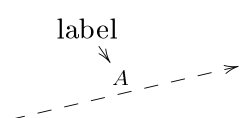
3.3k. Unless ⟨it⟩ is a full-fledged ⟨object⟩ (by using the `*` form), it is typeset using a `\labelbox` object (initially similar to `\objectbox` of basic XY-pic but using `\labelstyle` for the style).

Remark: You can only omit the `{}`s around single letters, digits, and control sequences.

```
\xylet@\labelstyle=\scriptstyle
\xydef@\labelbox#1{\hbox{$\m@th\labelstyle{#1}$}}
```

3.3l. A label is an object like any other in the XY-picture. Inserting an ⟨alias⟩ `="`⟨id⟩`"` saves the label object as `"`⟨id⟩`"` for later reference.

Exercise 3.2: Typeset



```
\xydef@\PATHlabelalias@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PATHlabelalias@}%
 \else \addEQ@\ifx \next
  \addEQ@\DN@"##1"{\savealias@{##1}\xyFN@\PATHlabelalias@}%
 \else \DN@{\afteraliases@@\PATHlabels@}\fi\fi \next@}

\xylet@\afteraliases@@=\empty

%{\xyuncatcodes \gdef\next#1{\POS!C="#1"\relax}}
\xydef@\savealias@#1{\xy@@{\enter@\cfromthec@
    \advance\X@c-\L@c \advance\L@c\R@c \L@c=.5\L@c \R@c=\L@c \advance\X@c\L@c
    \advance\Y@c-\D@c \advance\D@c\U@c \D@c=.5\D@c \U@c=\D@c \advance\Y@c\D@c
    \idfromc@{#1}\leave@}}
```

3.3.2 Arrows

```
\message{\string\ar,}
```

Arrows are paths with a particularly easy syntax for setting up arrows with *tail*, *stem*, and *head* in the style of [12]. This is provided by a single ⟨decor⟩ation the syntax of which is described in figure 3.2 (with the added convention that a raised '*' means 0 or more repetitions of the preceeding nonterminal).

Parsing: The `\ar` command effectively translates the ⟨arrow⟩ into a `\PATH` command that initialises the tail, stem, and head of it as required to get the desired ⟨arrow⟩.

The translation of the ⟨arrow⟩ is saved in the following variables:

```
\xylet@\arvariant@@=\empty
\xylet@\arstemprefix@@=\empty
\xylet@\artail@@=\empty
\xylet@\arstem@@=\empty
\xylet@\arhead@@=\empty
\xylet@\armodifiers@@=\empty
\xylet@\arlabels@@=\empty
\xylet@\arafterPOS@@=\empty
\xylet@\arinit@@=\empty
\xylet@\arexit@@=\empty

\xylet@\arcomponent@@=\relax
\xylet@\arcomponenttype@@=\relax

\xylet@\afterar@@=\relax
```

`\ar` stores the defaults; it loads an intermediate command to avoid problems when used as first command in compiled matrix entry...

```
\xydef@\ar{\relax\arSAFE}

\xydef@\arSAFE{%
 \let\arvariant@@=\empty
 \def\arstemprefix@@{\dir}%
 \edef\artail@@{\arvariant@@{}}%
 \edef\arstem@@{\arvariant@@{-}}%
 \edef\arhead@@{\arvariant@@{>}}%
 \def\armodifiers@@{}%
```

```
 \def\arafterPOS@@{}%
 \def\arlabels@@{}%
 \def\arinit@@{}%
 \def\arexit@@{}%
 \let\PATHlabelabove@@=\PATHlabelabove@
 \let\PATHlabelbelow@@=\PATHlabelbelow@
 \xyFN@\ar@}
```

`\ar@` is where we parse the sequence of ⟨form⟩s and continue with typesetting the arrow afterwards.

```
\xydef@\ar@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\ar@}%
 \else \addAT@\ifx\next \addAT@\DN@{\xyFN@\ar@form}%
 \else\ifx |\next
  \DN@ |{\ar@anchor|}%
 \else\ifx ^\next
  \DN@ ^{\ar@anchor^}%
 \else\ifx _\next
  \DN@ _{\ar@anchor_}%
 \else \let\next@=\ar@x \fi\fi\fi\fi\fi \next@}
```

Note: Makes use of the *new compilation principle* to parse a ⟨place⟩ vacuously…

```
\xydef@\addtoarinit@#1{%
 \expandafter\def\expandafter\arinit@@\expandafter{\arinit@@ #1}}
```

```
\xydef@\addtoarlabels@#1{%
 \expandafter\def\expandafter\arlabels@@\expandafter{\arlabels@@ #1}}
```

```
\xydef@\ar@anchor#1{\begingroup
 \def\PATHlabelit@@##1##2{%
  \DN@{##1}\ifx\next@\empty
   \DN@####1{\expandafter\endgroup\expandafter\addtoarlabels@
    \expandafter{\the\toks@####1{##2}}\xyFN@\ar@}%
  \else
   \DN@####1{\expandafter\endgroup\expandafter\addtoarlabels@
    \expandafter{\the\toks@*!C####1##1{##2}}\xyFN@\ar@}\fi
  \expandafter\next@\expandafter{\PATHitshape@@}}%
 \toks@={#1}%
 \def\xy@##1##2{\addtotoks@{##1}}\change@oxy@\xy@ \let\xy@@ix@=\eat@
 \DNii@{}\xyFN@\PATHanchor@}
```

`\ar@form` handles one ⟨form⟩:

```
\xydef@\ar@form{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\ar@form}%
 \else\ifx ^\next \DN@ ^{\xyFN@\ar@style}\edef\arvariant@@{\string^}%
 \else\ifx _\next \DN@ _{\xyFN@\ar@style}\edef\arvariant@@{\string_}%
 \else\ifx 0\next \DN@ 0{\xyFN@\ar@style}\def\arvariant@@{0}%
 \else\ifx 1\next \DN@ 1{\xyFN@\ar@style}\def\arvariant@@{1}%
 \else\ifx 2\next \DN@ 2{\xyFN@\ar@style}\def\arvariant@@{2}%
 \else\ifx 3\next \DN@ 3{\xyFN@\ar@style}\def\arvariant@@{3}%
 \else\ifx \bgroup\next \let\next@=\ar@style
 \else\ifx [\next \DN@[##1]{\ar@modifiers{[##1]}}%]
 \else\ifx *\next \DN@ *{\ar@modifiers}%
```

```
 \else\addLT@\ifx\next \let\next@=\ar@slide
 \else\ifx /\next \let\next@=\ar@curveslash
 \else\ifx (\next \let\next@=\ar@curveinout %)
 \else\addRQ@\ifx\next \addRQ@\DN@{\ar@curve@}%
 \else\addLQ@\ifx\next \addLQ@\DN@{\xyFN@\ar@curve}%
 \else\addDASH@\ifx\next \addDASH@\DN@{\defarstem@-\xyFN@\ar@}%
 \else\addEQ@\ifx\next \addEQ@\DN@{\def\arvariant@@{2}\defarstem@-\xyFN@\ar@}%
 \else\addDOT@\ifx\next \addDOT@\DN@{\defarstem@.\xyFN@\ar@}%
 \else\ifx :\next \DN@:{\def\arvariant@@{2}\defarstem@.\xyFN@\ar@}%
 \else\ifx ~\next \DN@~{\defarstem@~\xyFN@\ar@}%
 \else\ifx !\next \DN@!{\dasharstem@\xyFN@\ar@}%
 \else\ifx ?\next \DN@?{\ar@upsidedown\xyFN@\ar@}%
 \else \let\next@=\ar@error
 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}

\xydef@\defarstem@#1{\edef\arstem@@{\arvariant@@{\string#1}}%
 \DNii@##1##{\next@}%
 \edef\next@##1{\def\noexpand\artail@@{\arvariant@@{##1}}}%
  \expandafter\nextii@\artail@@
 \edef\next@##1{\def\noexpand\arhead@@{\arvariant@@{##1}}}%
  \expandafter\nextii@\arhead@@}

\xydef@\dasharstem@{%
 \DN@##1##{\nextii@{##1}}\DNii@##1##2{\def\arstem@@{##1{##2##2}}}%
 \expandafter\next@\arstem@@}

\xydef@\ar@error#1{\xyerror@{Illegal <form>ation (\meaning\next)}{}%
 \xyFN@\ar@}
```

`\ar@i` sets the remaining defaults and then allows `\ar@ii-iv` to pick up any tail, stem, and head using a `$` as terminator.

```
\xydef@\ar@style{%
 \ifx \bgroup\next \def\artail@@{{}}%
  \edef\arstem@@{\arvariant@@{-}}\edef\arhead@@{\arvariant@@{>}}%
  \expandafter\ar@i
 \else \resetvariant@\artail@@ \resetvariant@\arstem@@ \resetvariant@\arhead@@
  \expandafter\xyFN@\expandafter\ar@ \fi}

\xydef@\resetvariant@#1{%
 \DN@##1##{\DN@{##1}\ifx\next@\empty
   \DN@{\expandafter\nextii@\expandafter{\arvariant@@}}%
  \else \DN@{\nextii@{##1}}\fi \next@}%
 \DNii@##1##2{\def#1{##1{##2}}}%
 \expandafter\next@#1}

\xydef@\ar@i#1{\DN@{#1}%
 \ifx\next@\empty \edef\arstem@@{\arvariant@@{}}\edef\arhead@@{\arvariant@@{}}%
  \DN@{\xyFN@\ar@}%
 \else
  \let\arcomponent@@=\ar@ii \let\arcomponenttype@@=\artip@
  \DN@{\xyFN@\arcomponent@#1$}%
 \fi \next@}

\xydef@\ar@ii{\ifx $\next \let\next@=\ar@iv
```

```
\else \expandafter\def\expandafter\artail@@\expandafter{\the\toks@}%
 \let\arcomponent@@=\ar@iii \let\arcomponenttype@@=\arconn@
 \DN@{\xyFN@\arcomponent@}\fi \next@}

\xydef@\ar@iii{%
 \expandafter\def\expandafter\arstem@@\expandafter{\the\toks@}%
 \resetvariant@\artail@@
 \let\arcomponent@@=\ar@iv \let\arcomponenttype@@=\artip@
 \xyFN@\arcomponent@}

\xydef@\ar@iv{%
 \expandafter\def\expandafter\arhead@@\expandafter{\the\toks@}%
 \ifx $\next \DN@ ${\xyFN@\ar@}%
 \else \xyerror@{illegal <arrow>: \meaning\next\space not valid}{}\fi \next@}
```

`\ar@x` is the final macro that passes control to the ⟨path⟩ interpreter. It is a bit clumsy because we must pass the right character tokens; it could definitely be more efficient by calling ⟨path⟩ internals directly…

```
\xydef@\ar@x{%
 \let\arsavedPATHafterPOS@@=\PATHafterPOS \let\PATHafterPOS=\arafterPOS@
 \toks@={\ar@PATH}%
 \expandafter\addtotoks@\expandafter{\expandafter{\artail@@}}%#1
 \expandafter\addtotoks@\expandafter{\expandafter{\arstem@@}}%#2
 \expandafter\addtotoks@\expandafter{\expandafter{\arstemprefix@@}}%#3
 \expandafter\addtotoks@\expandafter{\expandafter{\arhead@@}}%#4
 \expandafter\addtotoks@\expandafter{\expandafter{\armodifiers@@}}%#5
 \expandafter\addtotoks@\expandafter{\expandafter{\arinit@@}}%#6
 \expandafter\addtotoks@\expandafter{\expandafter{\arexit@@}}%#7
 \expandafter\addtotoks@\expandafter{\expandafter{\arlabels@@}}%#8
 \addtotoks@{\afterar@@}%#9
 \expandafter\DNii@\expandafter{\the\toks@}\toks@={}%
%
%{\let\x=\expandafter \x\x\x\DNii@\x\x\x{\nextii@}\W@{RUN: \codeof\nextii@}}%
%
 \nextii@}

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{%
 \def\next{%
  \afterPATH{#9}%
  ~={#6\preconnect#5#3#2}%
  ~/{#7}%
  ~<{|<*h#5\dir#1}%
  ~>{|>*h#5\dir#4}%
  ~+{#8}%
  }%
 \next}}

\xylet@\ar@PATH=\next

\xydef@\arafterPOS@#1{%
 \arsavedPATHafterPOS@@{\let\PATHafterPOS=\arsavedPATHafterPOS@@
  \DN@{#1}\expandafter\next@\arafterPOS@@}}%

\xylet@\arsavedPATHafterPOS@@=\relax
```

⟨tip⟩s and ⟨conn⟩ectors are interpreted by these that leave any ⟨variant⟩ in `\toks@ii` and the sequence of ⟨tipchar⟩s or ⟨connchar⟩s characters in `\toks@`:

The work is really done in `\arcomponent@` that builds `\toks@` with something conforming to the `\arcomponenttype@@` and then passes control to `\arcomponent@@`:

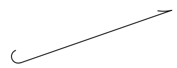
```
\xydef@\arcomponent@{%
 \ifx ^\next      \toks@ii={^}\DN@ ^{\xyFN@\arcomponent@i}%
 \else\ifx _\next \toks@ii={_}\DN@ _{\xyFN@\arcomponent@i}%
 \else\ifx 1\next \toks@ii={1}\DN@ 1{\xyFN@\arcomponent@i}%
 \else\ifx 2\next \toks@ii={2}\DN@ 2{\xyFN@\arcomponent@i}%
 \else\ifx 3\next \toks@ii={3}\DN@ 3{\xyFN@\arcomponent@i}%
 \else\ifx \bgroup\next \expandafter\toks@ii\expandafter{\arvariant@@}%
  \let\next@=\arcomponent@i
 \else\ifx *\next \DN@*##1##{\arcomponent@ii{##1}}%
 \else \expandafter\toks@ii\expandafter{\arvariant@@}\toks@={}%
  \let\next@=\arcomponenttype@@
 \fi\fi\fi\fi\fi\fi\fi
 \next@}

\xydef@\arcomponent@i#1{\toks@={#1}\arcomponent@x}

\xydef@\arcomponent@ii#1#2{\toks@={*#1{#2}}\xyFN@\arcomponent@@}

\xydef@\artip@{%
 \addGT@\ifx\next \addGT@\addtotoks@ \addGT@\DN@{\xyFN@\artip@}%
 \else\addLT@\ifx\next \addLT@\addtotoks@ \addLT@\DN@{\xyFN@\artip@}%
 \else\ifx (\next \addtotoks@(\DN@({\xyFN@\artip@}%
 \else\ifx )\next \addtotoks@)\DN@){\xyFN@\artip@}%
 \else\ifx |\next \addtotoks@|\DN@|{\xyFN@\artip@}%
 \else\addLQ@\ifx\next \addLQ@\addtotoks@ \addLQ@\DN@{\xyFN@\artip@}%
 \else\addRQ@\ifx\next \addRQ@\addtotoks@ \addRQ@\DN@{\xyFN@\artip@}%
 \else\addPLUS@\ifx \next \addPLUS@\addtotoks@ \addPLUS@\DN@{\xyFN@\artip@}%
 \else\ifx /\next \addtotoks@/\DN@/{\xyFN@\artip@}%
 \else\ifcat A\noexpand\next \DN@##1{\addtotoks@{##1}\xyFN@\artip@}%
 \else\ifx\space@\next \addtotoks@{ }\expandafter\DN@\space{\xyFN@\artip@}%
 \else \let\next@=\arcomponent@x
 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}

\xydef@\arconn@{%
 \addDASH@\ifx\next \addDASH@\addtotoks@ \addDASH@\DN@{\xyFN@\arconn@}%
 \else\addEQ@\ifx\next \addEQ@\addtotoks@ \addEQ@\DN@{\xyFN@\arconn@}%
  \ifx\arvariant@@\empty \def\arvariant@@{2}\fi
 \else\addDOT@\ifx\next \addDOT@\addtotoks@ \addDOT@\DN@{\xyFN@\arconn@}%
 \else\ifx :\next \addtotoks@:\DN@:{\xyFN@\arconn@}%
  \ifx\arvariant@@\empty \def\arvariant@@{2}\fi
 \else\ifx ~\next \addtotoks@~\DN@~{\xyFN@\arconn@}%
 \else \let\next@=\arcomponent@x
 \fi\fi\fi\fi\fi \next@}

\xydef@\arcomponent@x{%
 \DN@##1{\toks@=\expandafter{\the\toks@ii{##1}}}%
 \expandafter\next@\expandafter{\the\toks@}%
 \xyFN@\arcomponent@@}
```

Notes

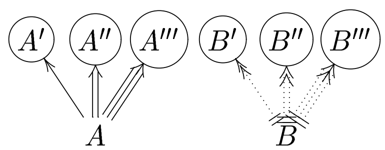
3.3m. Building an ⟨arrow⟩ is simply using the specified directionals (using `\dir` of §1.6.1) to build a path: the first ⟨tip⟩ becomes the *arrow tail* of the arrow, the ⟨conn⟩ection in the middle becomes the *arrow stem*, and the second ⟨tip⟩ becomes the *arrow head*. If a ⟨variant⟩ is given before the `{` then that variant `\dir` is used for all three. For example,

```
\xy\ar @^{(->} (20,7)\endxy
```

typesets



Exercise 3.3: Typeset these arrows:



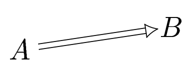
The above is a flexible scheme when used in conjunction with the kernel `\newdir` to define all sorts of arrowheads and -tails. For example,

```
\newdir{|>}{!/4.5pt/\dir{|}
            *:(1,-.2)\dir^{>}
            *:(1,+.2)\dir_{>}}
```

defines a new arrow tip that makes

```
\xy (0,0)*+{A}
 \ar @{=|>} (20,3)*+{B}
\endxy
```

typeset

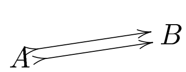


Notice that the fact that the directional uses only ⟨tipchar⟩ characters means that it blends naturally with the existing tips.

Exercise 3.4: Often tips used as ‘tails’ have their ink on the wrong side of the point where they are placed. Fortunately space is also a ⟨tipchar⟩ so we can define `\dir{ >}` to generate a ‘tail’ arrow. Do this such that

```
\xy (0,0)*+{A}="a", (20,3)*+{B}="b"
 \ar @{>->}  "a";"b" < 2pt>
 \ar @{ >->} "a";"b" <-2pt>
\endxy
```

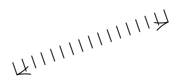
typesets



3.3n. Specifying a ⟨dir⟩ as a ⟨tip⟩ or ⟨conn⟩ means that `\dir`⟨dir⟩ is used for that ⟨tip⟩ or ⟨conn⟩. For example,

```
\xy\ar @{<^{|}>} (20,7)\endxy
```

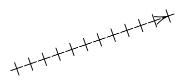
typesets



When using this you must specify a `{}` dummy ⟨dir⟩ectional in order to ignore one of the tail, stem, or tip components, *e.g.*,

```
\xy\ar @{{}{+}>} (20,7)\endxy
```

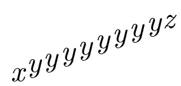
typesets



In particular `*`⟨object⟩ is a ⟨dir⟩ so any ⟨object⟩ can be used for either of the tail, stem, or head component:

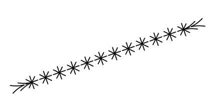
```
\xy\ar @{*{x}*{y}*{z}} (20,7)\endxy
```

typesets



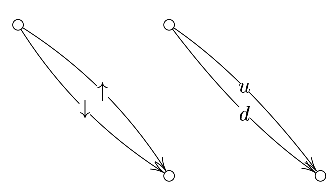
Note: A `*` introduces an ⟨object⟩ whereas the directional '•' is typeset by the ⟨dir⟩ `{*}`.

Exercise 3.5: Typeset



using only one `\ar` command.

3.3o. *Curving* the arrow using `/`$d\ell$`/`, where d is a ⟨direction⟩ and ℓ a ⟨dimen⟩sion, makes the stem a curve which is similar to a straight line but has had it's center point 'dragged' the distance ℓ in d:



was typeset by

```
\xy
 \POS (0,10)  *\cir<2pt>{} ="a"
    , (20,-10)*\cir<2pt>{} ="b"
 \POS"a" \ar @/^1ex/ "b"|\uparrow
 \POS"a" \ar @/_1ex/ "b"|\downarrow
%
 \POS (20,10) *\cir<2pt>{} ="a"
    , (40,-10)*\cir<2pt>{} ="b"
 \POS"a" \ar @/u1ex/ "b"|u
 \POS"a" \ar @/d1ex/ "b"|d
\endxy
```

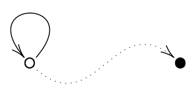
ℓ defaults to `.5pc` if omitted.

This is really just a shorthand for curving using the more general form described next: `@/`$d\ell$`/` is the same as `` @`{{**{} ?+/ ``d 2ℓ`` /}} `` which makes the (quadratic) curve pass through the point defined by the ⟨pos⟩ `**{} ?+/`$d\ell$`/`.

```
\xydef@\ar@curveslash/#1/{\expandafter\ar@curve@\ar@slashing{#1}}

\xydef@\ar@slashing#1{{\xy@{**{} ?+/#1/+/#1/ @+c}{\setupDirection@
   \vfromslide@{#1}%
   \X@c=2\X@c \advance\X@c\X@p \advance\X@c.5\d@X
   \Y@c=2\Y@c \advance\Y@c\Y@p \advance\Y@c.5\d@Y
   \czeroEdge@
   \spushc@}}}
```

3.3p. Using `@(`d_2`,`d_2`)` where d_1, d_2 are simple ⟨direction⟩s (as described in note 1.4l except it is not possible to use `()`s) will typeset the arrow curved such that it leaves the source in direction d_1 and enters the target from direction d_2.

Exercise 3.6: Typeset



```
{\xyuncatcodes \catcode`\#=6 %
 \gdef\next(#1,#2){{+/#1 3pc/,p+/#2 3pc/}}}
\xylet@\ar@curveinout@=\next

\xydef@\ar@curveinout{\expandafter\ar@curve@load\ar@curveinout@}
```

To Do: implement this efficiently and properly get rid of the no-`()` restriction!

3.3q. The final curve form is the most general one: `` @`{ ``⟨control point lists⟩`}` sets the control points explicitly to the ones in the ⟨control point lists⟩ (where they should be separated by `,`). See the curve extension described in §2.1 for the way the control points are used; when the control points list is parsed p is the source and c the target of the arrow.

Curving simply uses `\crvi` instead of `\dir`. The only tricky bit is to put the control points on the stack right.

```
\xydef@\ar@curve{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\ar@curve}%gobble spaces
 \else\ifx \bgroup\next \let\next@=\ar@curve@load
 \else\ifx "\next \DN@"##1"{\ar@curve@{"##1"}}%
 \else \xyerror@{@= <form> must be followed by \string"<id>\string" or
        {<control point list>}}{}%
 \fi\fi\fi \next@}

{\xyuncatcodes \gdef\next#1#2{\def#1##1{#2{;@={##1}}}}}
\next\ar@curve@load\ar@curve@

\xydef@\ar@curve@#1{\curve@check
 \setcurvearinit@{#1}\setcurvearexit@\arexit@@
 \def\arstemprefix@@{\crvi}\xyFN@\ar@}

{\xyuncatcodes \gdef\next#1{\save @(,#1\restore}}
```

```
\xylet@\setcurvearinit@i=\next
\xydef@\setcurvearinit@#1{%
 \expandafter\addtoarinit@\expandafter{\setcurvearinit@i{#1}}}
```

```
{\xyuncatcodes \gdef\next#1{\def#1{\POS @i @) }}}
\xylet@\setcurvearexit@=\next
```

```
\xydef@\curve@check{%
 \xyerror@{Forms @/.../, @(...), and @`{...}, only available when curve
        extension loaded}{}}
\xywithoption{curve}{\let\curve@check=\relax}
```

To Do: The parsing is trivial and should be extended to include more exotic control point forms.

3.3r. `@[...]` and `@*{...}` formations define what object ⟨modifier⟩s should be used when building objects that are part of the arrow. This is mostly useful in conjunction with extensions that define additional `[`⟨shape⟩`]` modifiers, *e.g.*, if a `[red]` ⟨modifier⟩ changes the colour of an object to red then `@[red]` will make the entire arrow red; similarly if it is desired to make and entire arrow invisible then `@*{i}` can be used.

```
\xydef@\ar@modifiers#1{%
 \expandafter\def\expandafter\armodifiers@@\expandafter{\armodifiers@@#1}%
 \xyFN@\ar@}
```

3.3s. `@<`D`>` will slide (each segment of) the arrow the dimension D as explained in note 3.3h.

```
{\xyuncatcodes \catcode`\#=6 \catcode`\@=11 %
 \gdef\next<#1>{\def\arafterPOS@@{<#1>}\xyFN@\ar@}}
\xylet@\ar@slide=\next
```

3.3t. `@?` reverse the meaning of 'above' and 'below' for this particular arrow.

```
\xydef@\ar@upsidedown{\let\next=\PATHlabelabove@@
 \let\PATHlabelabove@@=\PATHlabelbelow@@ \let\PATHlabelbelow@@=\next}
```

All the features of ⟨path⟩s described above are available for arrows.

End & log

```
\xyendinput
```

```
% $Log: xyarrow.doc,v $
% Revision 3.5  1997/05/28 13:05:01  krisrose
% Fixed missing breaks bug.
%
% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.2  1995/09/19  18:22:27  kris
% Bug fix release.
%
```

```
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.11  1994/07/05  10:37:32  kris
% Third 3beta release [bug fixes].
% Experimental graph feature included (for ECCT-94 presentation).
%
% Revision 2.10  1994/06/15  12:55:07  kris
% Second 3beta release: bug fixes.
%
% Revision 2.9  1994/06/09  14:59:19  kris
% Release 3beta.
%
% Revision 2.8  1994/04/08  04:30:00  kris
% Second (bug fix) 3alpha release.
%
% Revision 2.7  1994/03/08  10:25:18  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% NEW for version 2.7 based of path and arrow code in xypic.doc 2.6.1.1.
```

3.4 Two-cell feature

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This feature is designed to facilitate the typesetting of curved arrows, either singly or in pairs, together with labels on each part and between. The intended mathematical usage is for typesetting categorical "2-cell" morphisms and "pasting diagrams", for which special features are provided. These features also allow attractive non-mathematical effects.

Header:

```
%% $Id: xy2cell.doc,v 3.3 1996/12/18 14:21:23 ross Exp $
%%
%% Xy-pic ``2-cell'' feature.
%% Copyright (c) 1993-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
```

```
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this macro package; if not, write to the Free Software Foundation,
%% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{2cell}{Two-cell feature}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

The 2-cell feature makes use of facilities from the 'curve' extension which is therefore automatically loaded.

```
\xyrequire{curve}\xycatcodes
```

3.4.1 Typesetting 2-cells in Diagrams

Categorical "2-cell" morphisms are used in the study of tensor categories and elsewhere. The morphisms are displayed as a pair of curved arrows, symmetrically placed, together with an orientation indicated by a short broad arrow, or *Arrow*. Labels may be placed on all three components.

```
\message{two-cells,}
```

Bug: This document still uses version 2-style commmands, as described in appendix B.1.

$$\begin{tikzcd} A \arrow[r, bend left, "f"] \arrow[r, bend right, "g"'] \arrow[r, phantom, "\Downarrow"] & B \end{tikzcd}$$

```
\diagram
A\rtwocell^f_g &B\\
\enddiagram
```

$$\begin{tikzcd} A \arrow[r, bend left=40, "f"] \arrow[r, "g"'] \arrow[r, bend right=40, "h"'] \arrow[r, phantom, shift left=3, "\Downarrow\alpha"] \arrow[r, phantom, shift right=3, "\Downarrow\beta"] & B \end{tikzcd}$$

```
\diagram
A\ruppertwocell^f{\alpha}
```

```
  \rlowertwocell_h{\beta}
  \rto_(.35)g & B\\
\enddiagram
```

These categorical diagrams frequently have a matrix-like layout, as with commutative diagrams. To facilitate this there are control sequences of the form: `\rtwocell` , `\ultwocell` , `\xtwocell` , ... analogous to the names defined in `xyv2` for use in diagrams produced using `xymatrix`. As this involves the definition of 21 new control sequences, many of which may never be used, these are not defined immediately upon loading `xy2cell`. Instead the user must first specify `\UseTwocells`.

As in the second example above, just the upper or lower curved arrow may be set using control sequences of the form `\..uppertwocell` and `\..lowertwocell`. These together with the `\..compositemap` family, in which two abutting arrows are set with an empty object at the join, allow for the construction of complicated “pasting diagrams” (see figure 3.3 for an example).

The following initialise the families of control sequences for use in matrix diagrams.

`\UseTwocells`	two curves
`\UseHalfTwocells`	one curve
`\UseCompositeMaps`	2 arrows, end-to-end
`\UseAllTwocells`	(all the above)

```
\xydef@\UseTwocells{\definesupermorphism{twocell}{%
 \xy@{start of 2-cell}{\begingroup}%
 \twocell@}}

\xydef@\UseHalfTwocells{%
 \definesupermorphism{uppertwocell}{%
  \xy@{start of 2-cell}{\begingroup}%
  \uppertwocell@}%
 \definesupermorphism{lowertwocell}{%
  \xy@{start of 2-cell}{\begingroup}%
  \lowertwocell@}}

\xydef@\UseCompositeMaps{\definesupermorphism{compositemap}{%
  \xy@{start composite-map}{\begingroup}%
  \compositemap@}}

\xydef@\UseAllTwocells{%
 \UseTwocells \UseHalfTwocells \UseCompositeMaps }
```

The families of connections are defined using the utility macro...`\definesupermorphism` which defines control sequences for morphisms between neighbouring cells, and next-to-neighbouring cells.

More distant cells use `\xtwocell`, `\xcompositemap`, etc. with ⟨hop⟩s e.g. `\xtwocell[1,3]` and `\xtwocell[llddd]`.

```
\xydef@\supermorphism#1[#2]#3{\def\afterMORPHISM{[#2]#3}#1}
\xydef@\afterMORPHISM{}

\xydef@\definesupermorphism#1#2{%
 \expandafter\def\csname x#1\endcsname{\supermorphism{#2}}%
 \expandafter\def\csname u#1\endcsname{\supermorphism{#2}[-1,0]{}}%
 \expandafter\def\csname d#1\endcsname{\supermorphism{#2}[1,0]{}}%
 \expandafter\def\csname l#1\endcsname{\supermorphism{#2}[0,-1]{}}%
 \expandafter\def\csname r#1\endcsname{\supermorphism{#2}[0,1]{}}%
```

```
\expandafter\def\csname uu#1\endcsname{\supermorphism{#2}[-2,0]{}}%
\expandafter\def\csname dd#1\endcsname{\supermorphism{#2}[2,0]{}}%
\expandafter\def\csname ll#1\endcsname{\supermorphism{#2}[0,-2]{}}%
\expandafter\def\csname rr#1\endcsname{\supermorphism{#2}[0,2]{}}%
\expandafter\def\csname ur#1\endcsname{\supermorphism{#2}[-1,1]{}}%
\expandafter\def\csname ul#1\endcsname{\supermorphism{#2}[-1,-1]{}}%
\expandafter\def\csname dr#1\endcsname{\supermorphism{#2}[1,1]{}}%
\expandafter\def\csname dl#1\endcsname{\supermorphism{#2}[1,-1]{}}%
\expandafter\def\csname uul#1\endcsname{\supermorphism{#2}[-2,-1]{}}%
\expandafter\def\csname uur#1\endcsname{\supermorphism{#2}[-2,1]{}}%
\expandafter\def\csname ull#1\endcsname{\supermorphism{#2}[-1,-2]{}}%
\expandafter\def\csname urr#1\endcsname{\supermorphism{#2}[-1,2]{}}%
\expandafter\def\csname ddl#1\endcsname{\supermorphism{#2}[2,-1]{}}%
\expandafter\def\csname ddr#1\endcsname{\supermorphism{#2}[2,1]{}}%
\expandafter\def\csname dll#1\endcsname{\supermorphism{#2}[1,-2]{}}%
\expandafter\def\csname drr#1\endcsname{\supermorphism{#2}[1,2]{}}%
\expandafter\def\csname uull#1\endcsname{\supermorphism{#2}[-2,-2]{}}%
\expandafter\def\csname uurr#1\endcsname{\supermorphism{#2}[-2,2]{}}%
\expandafter\def\csname ddll#1\endcsname{\supermorphism{#2}[2,-2]{}}%
\expandafter\def\csname ddrr#1\endcsname{\supermorphism{#2}[2,2]{}}}
```

Alternatively 2-cells can be set directly in XY-pictures without using the matrix feature. In this case the above commands are not needed. This is described in §3.4.6.

Furthermore a new directional `\dir{=>}` can be used to place an "Arrow" anywhere in a picture, after the direction has been established appropriately. It is used with all of the 2-cell types.

```
\newdir{=>}{!/5pt/\dir{=}!/2.5pt/\dir{=}*!/-5pt/\dir2{>}}
```

```
\xydef@\arrowobject#1{\def\Arrowobject@{#1}}
\xydef@\Arrowobject@{\dir{=>}}
```

Default positions: These set up default values for the parameters which specify the position of each component as well as the flags to indicate what type of 2-cell is to be typeset.

`\toks6`	position of control point, normal to line $\overline{pc}$ at midpoint
`\toks5`	position of central Arrow
`\toks7`	position of `^` label
`\toks8`	position of `_` label
`\toks3`	flag for number of pieces...
`f`	full: i.e. both curved arrows
`o`	one: `\uppertwocell` or `\lowertwocell`
`t`	two maps: `\compositemaps`

`@` means to calculate from the value in `\toks6`, see `\tw@cell@@@`.

```
\xydef@\twocell@{\toks6={3.5}\toks5={@}\toks7={@}\toks8={@}\toks3={f}%
 \tw@cell@ }
\xydef@\uppertwocell@{\toks6={7}\toks5={@}\toks7={@}\toks8={@}\toks3={o}%
 \tw@cell@ }
\xydef@\lowertwocell@{\toks6={-7}\toks5={@}\toks7={@}\toks8={@}\toks3={o}%
```

```
 \tw@cell@ }
\xydef@\compositemap@{\toks6={3.5}\toks5={@}\toks7={@}\toks8={@}\toks3={t}%
 \tw@cell@ }
```

This collects together all the relevant data, and resolves the default positions.

```
\xydef@\tw@cell@@@{%
 \expandafter\ifx\the\toks6\omit\relax\toks6={}\else
%
%    default position of label on upper arrow,
%    relative to where it meets the perpendicular bisector
%
 \expandafter\ifx\expandafter @\the\toks7\relax\expandafter\dimen@\the\toks6\p@
  \expandafter\if\the\toks3t\relax\dimen@=\p@
  \else\expandafter\if\the\toks3o\divide\dimen@\tw@
   \ifdim\dimen@<\z@ \dimen@ii-\p@ \advance\dimen@ii\dimen@
    \dimen@=\dimen@ii \else \dimen@\p@ \fi
   \else\dimen@\p@
  \fi\fi
  \edef\tmp@{\expandafter\removePT@\the\dimen@}%
  \expandafter\toks\expandafter7\expandafter{\tmp@}\fi
%
%    default position of label on lower arrow,
%    relative to where it meets the perpendicular bisector
%
 \expandafter\ifx\expandafter @\the\toks8\relax
  \expandafter\dimen@\the\toks6\p@
  \expandafter\if\the\toks3t\relax\dimen@\p@
  \else\expandafter\if\the\toks3o\divide\dimen@\tw@
   \ifdim\dimen@<\z@ \dimen@ii-\p@ \advance\dimen@ii\dimen@
    \dimen@=\dimen@ii \else \dimen@\p@ \fi
   \else\dimen@\p@
  \fi\fi
  \edef\tmp@{\expandafter\removePT@\the\dimen@}%
  \expandafter\toks\expandafter8\expandafter{\tmp@}\fi
%
%  default position of the Arrow, relative to midpoint
%
 \expandafter\ifx\expandafter @\the\toks5\relax
  \expandafter\dimen@\the\toks6\p@
  \expandafter\if\the\toks3o\divide\dimen@ by-4\relax
   \else\dimen@\z@\fi
  \edef\tmp@{\expandafter\removePT@\the\dimen@}%
  \expandafter\toks\expandafter5\expandafter{\tmp@}\fi
 \fi
%
%   possible modifications for module maps
%
 \twocellmod@
%
%   pass all the parameters to  |\twocell@@@@|
```

```
%
\expandafter\if\the\toks3t\relax
%
%   composed maps:   ^ : first map   _ : second map
%
 \edef\tmp@{.[\noexpand\twocell@@@@{\the\toks6}{\the\toks5}%
  {\the\toks7}{\the\toks8}{\the\toks3}%
  {\the\toks2}{\the\toks1}{\the\toks@}{\the\toks4}!!!].\afterMORPHISM}%
\else
%
%    curved arrow(s)
%
 \edef\tmp@{.[\noexpand\twocell@@@@{\the\toks6}{\the\toks5}%
  {\the\toks7}{\the\toks8}{\the\toks3}%
  {\the\toks1}{\the\toks2}{\the\toks@}{\the\toks4}!!!].\afterMORPHISM}%
\fi
\toks@={\endgroup}%
\expandafter\def\expandafter\next@\expandafter{%
\expandafter\def\expandafter\twocellhead@\expandafter{\twocellhead@}}%
\expandafter\addtotoks@\expandafter{\next@}%
\expandafter\def\expandafter\next@\expandafter{%
\expandafter\def\expandafter\twocelltail@\expandafter{\twocelltail@}}%
\expandafter\addtotoks@\expandafter{\next@}%
\expandafter\addtotoks@\expandafter{\expandafter\twocellPATH\tmp@}%
\the\toks@ }%
```

See below for the possible forms of `\twocellPATH`.

Labels are placed labels on the upper and lower arrows, more correctly 'anti-clockwise' and 'clockwise', using `^` and `_`. These are entirely optional with the following token, or grouping, giving the contents of the label. When used with `\..compositemap` the `^` and `_` specify labels for the first and second arrows, respectively.

```
\xydef@\twocellstyle{\scriptstyle}
```

```
\xydef@\droptwocelllabel@#1{\xyFN@\droptwocelllabel@@#1@!}
\xydef@\droptwocelllabel@@{%
 \ifx*\next\DN@*{\droptwocelldrop@}%
 \else\DN@##1@!{\droptwocelltext@{##1}}%
 \fi \next@ }
```

```
{\xyuncatcodes \gdef\next#1{\drop+!C{\twocellstyle #1}}}
\xylet@\droptwocelltext@=\next
```

```
\xydef@\droptwocelldrop@#1@!{\bgroup \let\xy@=\oxy@
 \let\objectstyle=\twocellstyle \drop#1{}%
 \edef\tmp@{\egroup \X@min=\the\X@min \X@max=\the\X@max
  \Y@min=\the\Y@min \Y@max=\the\Y@max}\tmp@ }
```

Normally the label is balanced text, set in TeX's math mode, with `\twocellstyle` setting the style. The default definition is given by ...

`\def\twocellstyle{\scriptstyle}`

This can be altered using `\def` in versions of TeX or `\redefine` in LaTeX. However labels are not restricted to being simply text boxes. Any effect obtainable using the XY-pic kernel language can be set within an \xybox and used as a label. Alternatively if the first character in the label is * then the label is set as an XY-pic ⟨object⟩, as if with \drop<object> or *<object> in the kernel language. The current direction is tangential to the curved arrows. Extra braces are needed to get a * as the label, as in ^{{{*}}} or _{{}*}.

The position of a label normal to the tangential direction can also be altered by *nudging* (see below). Although it is possible to specify multiple labels, only the last usage of each of ^ and _ is actually set, previous specifications being ignored.

Similarly a label for the central Arrow must be given, after the other labels, by enclosing it within braces {...}. An empty group {} gives an empty label; this is necessary to avoid misinterpretation of subsequent tokens. As above if the first character is * then the label is set as an XY-pic ⟨object⟩, the current direction being along the Arrow.

Parsing: Open a new group so the following registers can be used locally... \toks@=\toks0, \toks1, ..., \toks8 the final parameter set is stored briefly in \tmp@ before being used.

OPTIONS:

		spaces are ignored
<	<num>	determines position of curves control point
	<0>	uses a single solid line,
	<\omit>	causes only the central Arrow (with label) to be set
\omit		omits the curved arrows, but allows their labels
^		place "above" label (more correctly "anti-clockwise")
_		place "below" label (more correctly "clockwise")
~'		change the arrowhead
~`		place/change the arrowtail

Label data e.g. ^{#1} _{#1} is stored temporarily in \toks@ then passed via \tw@@cell@@ to \tw@cell@@ for further parsing, as: \the\toks@ @

~' and ~` change the values of \twocellhead@ and \twocelltail@ Normally \twocelltail@ is {}, so ~` also changes a flag to indicate that \twocelltail@ is indeed required.

When the next token is a grouping {...} then it gives the label for the central Arrow. No other label specifications may follow.

Default head and tail ornaments...

```
\xydef@\twocellhead#1{\def\twocellhead@{#1}}
\xydef@\twocelltail#1{\def\twocelltail@{#1}}
\xydef@\twocellhead@{\dir{>}}
\xydef@\twocelltail@{}
```

clear \toks.. registers for the new connection labels.

```
\xywarnifdefined\twocellmod@
\xydef@\tw@cell@{\begingroup
 \toks@={{}}\toks1={{}}\toks2={{}}\toks4={{}}%
 \def\twocellmod@{}\xyFN@\tw@@cell@}%
```

```
\xydef@\tw@@cell@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\tw@@cell@}%
 \else\ifx\bgroup\next \let\next@=\toks@\afterassignment\tw@@cell@@
 \else\ifx ^\next \DN@^##1{\toks@={##1}\nudgepos@71\tw@@cell@}%
 \else\ifx _\next \DN@_##1{\toks@={##1}\nudgepos@82\tw@@cell@}%
 \else\addLT@\ifx \next \addGT@{\addLT@\DN@##1}{%
```

```
  \toks6={##1}\ifx\omit##1\relax\toks3={t}\toks5={0}\fi\xyFN@\tw@@cell@}%
 \else\ifx\omit\next \DN@\omit{\omitarrows@\xyFN@\tw@@cell@}%
 \else\ifx~\next \DN@~{\xyFN@\whichCurveObject@}%
 \else \DN@{\xyFN@\tw@cell@@\empty @}%
 \fi\fi\fi\fi\fi\fi\fi%
 \next@ }
```

```
\xydef@\whichCurveObject@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\whichCurveObject@}%
 \else\ifx\next^\DN@^##1{\xy@{~^{##1}}{\uppercurveobject{##1}}\xyFN@\tw@@cell@}%
 \else\ifx\next_\DN@_##1{\xy@{~_{##1}}{\lowercurveobject{##1}}\xyFN@\tw@@cell@}%
 \else\addRQ@\ifx\next \addRQ@\DN@##1{%
  \xy@{~'{##1}}{\def\twocellhead@{##1}}\xyFN@\tw@@cell@}%
 \else\addLQ@\ifx\next \addLQ@\DN@##1{%
  \xy@{~`{##1}}{\def\twocelltail@{##1}}\xyFN@\tw@@cell@}%
 \else\ifx!\next \DN@!{\xyFN@\checkmodmap@}%
 \else\ifx\next\bgroup\DN@##1{\xy@{~{##1}}{\curveobject{##1}}\xyFN@\tw@@cell@}%
 \else\DN@##1{\xy@{~##1}{\curveobject{##1}}\xyFN@\tw@@cell@}%
 \fi\fi\fi\fi \fi\fi\fi
 \next@ }
```

```
\xydef@\checkmodmap@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\checkmodmap@}%
 \else \def\twocellmod@{\modmap@}%
  \ifx\next!\DN@!##1{%
   \xy@{~!!{##1}}{\def\modmapobject@{##1}}\xyFN@\tw@@cell@}%
  \else \DN@{\xyFN@\tw@@cell@}\fi
 \fi \next@ }
```

With no arrows, still allow labels: default positions are at 0, $+X$, $-X$, with $X =$*default value*. These values are altered by “nudging”.

```
\xydef@\omitarrows@{\toks5={0}\toks6={}\toks7={0}\toks8={0}}
```

3.4.2 Standard Options

The orientation of the central Arrow may be reversed, turned into an equality, or omitted altogether. In each case a label may still be specified, so in effect the Arrow may be replaced by anything at all.

These effects are specified by the first token in the central label, which thus has the form: `{`⟨tok⟩⟨label⟩`}` where ⟨tok⟩ may be one of ...

`_`	Arrow points clockwise
`^`	Arrow points anti-clockwise
`=`	no tip, denotes equality
`\omit`	no Arrow at all.

When none of these occurs then the default of `_` is assumed. If the label itself starts with one of these characters then specify `_` explicitly, or enclose the label within a group `{...}`. See *Extra Options 1*, for more values of ⟨tok⟩. Also note that `*` has a special role when used as the first character; however it is considered to be part of the ⟨label⟩, see above.

OPTIONS for the central Arrow:

c l	ignore spaces
^	points anti-clockwise
_	points clockwise (default)
=	no `\Tip`, equality
`\omit`	no Arrow at all, just squine(s) or composed arrows
'	no Arrow, opposite squine reversed
`	no Arrow, forward squine reversed
"	no Arrow, tips on both ends
!	no Arrow, no tips on squines

the position of the Arrow is determined by "nudging" (see below) e.g. `{^<num>label}`, where the actual dimen is `<num>\xydashl@`

$$\begin{tikzcd}
A \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow\alpha"] & B \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\beta\Uparrow"] & C \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\gamma\|"] & D \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Psi"] & E
\end{tikzcd}$$

```
\xymatrixcolsep{4pc}
\xymatrixrowsep{1pc}
\diagram
A \rtwocell{_\alpha} & B \rtwocell{^\beta} &
C \rtwocell{=\gamma} & D \rtwocell{\omit\Psi} & E \\
\enddiagram
```

```
\xydef@\tw@@cell@@{\expandafter\xyFN@\expandafter\tw@cell@@\the\toks@ @}

\xydef@\tw@cell@@{\ifx \space@\next\expandafter\DN@\space{\xyFN@\tw@cell@@}%
 \else\ifx ^\next
  \DN@^##1@{\toks@={##1}\toks4={^}\nudgepos@50\tw@cell@@@}%
 \else\addEQ@\ifx\next
  \addEQ@\DN@##1@{\toks@={##1}\toks4={=}\nudgepos@50\tw@cell@@@}%
 \else\ifx _\next
  \DN@_##1@{\toks@={##1}\toks4={_}\nudgepos@50\tw@cell@@@}%
 \else\addLT@\ifx \next
   \DN@##1@{\toks@={##1}\toks4={_}\nudgepos@50\tw@cell@@@}%
 \else\ifx\omit\next \DN@\omit##1@{\toks4={@}\toks@={##1}\tw@cell@@@}%
 \else\addRQ@\ifx\next \addRQ@\DN@##1@{\toks4={@}\toks3={/}\toks@={##1}%
  \nudgepos@50\tw@cell@@@}%
 \else\addLQ@\ifx\next \addLQ@\DN@##1@{\toks4={@}\toks3={b}\toks@={##1}%
  \nudgepos@50\tw@cell@@@}%
 \else\ifx"\next \DN@"##1@{\toks4={@}\toks3={B}\toks@={##1}%
  \nudgepos@50\tw@cell@@@}%
 \else\ifx!\next \DN@!##1@{\toks4={@}\toks3={@}\toks@={##1}%
  \nudgepos@50\tw@cell@@@}%
 \else\ifx\next\empty\DN@##1@{\toks@={{}}\toks4={_}\tw@cell@@@}%
 \else\DN@##1@{\toks@={##1}\toks4={_}\tw@cell@@@}%
 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@ }
```

Parsing of the Arrow's label

3.4.3 Nudging

Positions of all labels may be adjusted, as can the amount of curvature for the curved arrows. The way this is done is by specifying a "nudge" factor $\langle$num$\rangle$ at the beginning of the label. Here $\langle$num$\rangle$ is a number which specifies the actual position of the label in units of `\xydashl@` (the length of a single dash, normally 5pt) except with `\..compositemap`, see below. Movement is constrained to the perpendicular bisector of the line $\overline{cp}$. When nudging the label for the central Arrow it is the whole Arrow which is moved, along with its label.

Curvature of the arrows themselves is altered by a nudge of the form `\..twocell`$\langle$num$\rangle$.... The separation of the arrows, along the bisector, is set to be $\langle$num$\rangle$`\xydashl@`. When $\langle$num$\rangle$ is zero, that is `\..twocell<0>...`, the result is a single straight arrow, its mid-point being the origin for nudging labels. A negative value for $\langle$num$\rangle$ is also acceptable; but check the orientation on the Arrow and which of `^` and `_` correspond to which component.

The origin for nudging labels is where the arrow crosses the bisector. Positive nudges move the label outwards while negative nudges move towards $\overline{pc}$ and possibly beyond. The default position of a label is on the outside, with edge at the origin.

The origin for nudging the Arrow is at the midpoint of $\overline{pc}$. A positive nudge moves in the clockwise direction. This will be the direction of the arrowhead, unless it has been reversed using `^`.

Labels on a `\..compositemap` are placed relative to the midpoint of the component arrows. Nudges are in units of 1pt. Movement is in the usual XY-pic *above* and *below* directions, such that a positive nudge is always outside the triangle formed by the arrows and line $\overline{pc}$.

The special nudge value `<\omit>` typesets just the Arrow, omitting the curved arrows entirely. When used with labels, the nudge value `<\omit>` causes the following label to be ignored.

Exercise 3.7: Give code to typeset figure 3.3.
Such code is relatively straight-forward, using "nudging" and `\omit` to help position the arrows, curves and Arrows. It also uses an *excursion*, as described below in the subsection *Extra Options 3*.

NUDGE position of labels

```
\xywarnifdefined\whichnudge
\xywarnifdefined\whichinfo
\xywarnifdefined\afternudge
\xydef@\nudgepos@#1#2#3{\def\whichnudge{\toks#1}\def\whichinfo{\toks#2}%
 \def\afternudge{#3}\expandafter\xyFN@\expandafter\nudgepos@@\the\toks@ @}
```

`#1`	which token register to store nudge amount
`#2`	which token register to store label
`#3`	what to do next, after parsing

Parsing: $\langle$num$\rangle\langle$label$\rangle$`@`, $\langle$num$\rangle$ = nudge amount, as multiple of `\xydashl@`.

```
\xydef@\nudgepos@@{\ifx \space@\next\expandafter\DN@\space{\xyFN@\nudgepos@}%
 \else\addLT@\ifx \next
  \addGT@{\addLT@\DN@##1}##2@{%
    \ifx\omit##1\relax\whichnudge={0}\whichinfo={}%
    \else\whichnudge={##1}\whichinfo={##2}\fi\xyFN@\afternudge}%
 \else \DN@##1@{\whichinfo={##1}\xyFN@\afternudge}%
 \fi\fi \next@ }
```

3.4.4 Extra Options

The following features are useful in non-mathematical applications.

1. no Arrow

This is determined by special values for ⟨tok⟩ as the first (or only) character in the central label, as in the above description of the standard switches.

'	arrowheads pointing clockwise;
`	arrowheads pointing anti-clockwise;
"	arrow tips on both ends;
!	no tips at all.

The central Arrow is omitted, leaving symmetrically placed curved connections with arrowheads at the specified ends. A label can be placed where the Arrow would have been.

If a special arrowhead is specified using `~'{..}` (see Extra Options 2, below) then this will be used instead of the standard `\dir{>}`.

$$\begin{tikzcd}[column sep=5pc]
\text{Clouds} \arrow[r, bend left, "\text{\tiny precipitation}"] \arrow[r, phantom, "\mathbf{H_2O}"] & \text{Oceans} \arrow[l, bend left, "\text{\tiny evaporation}"]
\end{tikzcd}$$

```
\xymatrixcolsep{5pc}
\diagram
 \relax\txt{Clouds }\rtwocell<10>
 _{\hbox{\tiny evaporation }}
 ^{\hbox{\tiny precipitation }}
{'{\mathbf{H_2 O}}}
&\relax\txt{Oceans}\\
\enddiagram
```

$$\begin{tikzcd}[column sep=5pc]
\text{Mathematics} \arrow[r, bend left, <->, "\text{\tiny theory}"] \arrow[r, bend right, <->, "\text{\tiny experiment}"'] & \text{Physics}
\end{tikzcd}$$

```
\xymatrixcolsep{5pc}
\diagram
\relax\txt{\llap{Math}ematics }\rtwocell
_{\hbox{\tiny experiment }}
^{\hbox{\tiny theory }}{"}
& \relax\txt{Physics} \\
\enddiagram
```

2. Changing Tips and Module Maps

The following commands are provided for specifying the ⟨object⟩ to be used when typesetting various parts of the twocells.

command	*default*
`\modmapobject{`⟨object⟩`}`	`\dir{`|`}`
`\twocellhead{`⟨object⟩`}`	`\dir{>}`
`\twocelltail{`⟨object⟩`}`	`\dir{}`
`\arrowobject{`⟨object⟩`}`	`\dir{=>}`
`\curveobject{`⟨object⟩`}`	
`\uppercurveobject{`⟨object⟩`}`	`{}`
`\lowercurveobject{`⟨object⟩`}`	`{}`

These commands set the object to be used for all subsequent 2-cells at the same level of TEX grouping. `\curveobject` specifies both of the upper- and lower-curve objects. For some of these there is also a way to change the object for the current 2-cell only. This requires a `~`-⟨switch⟩ which is described below, except for the `\..curveobject` types, which are discussed in *Extra Options 4*.

These effects are specified by placing switches after the `\..twocell` control sequence, *e.g.* `\rtwocell` *switches labels*... . Each switch is either a single token ⟨tok⟩, or a `~`⟨tok⟩ with a single argument: `~`⟨tok⟩`{`*arg*`}`. Possibilities are listed in the following table, in which `{..}` denotes the need for an argument.

`\omit`	no arrows, Arrow and label only;
!	place module-map indicator;
`~'{..}`	change arrow-head to `{..}`;
``~`{..}``	place/change tail on arrow(s);
`~{..}`	change object used to set curves;
`~^{..}`	use object `{..}` to set upper curve;
`~_{..}`	use object `{..}` to set lower curve;

Here we discuss the use of `!`, `~'`, ``~` `` and `\omit`. The description of `~^`, `~_` and `~{..}` is given in *Extra Options 4*.

The default module map indicator places a single dash crossing the arrow at right-angles, located roughly midway along the actual printed portion of the arrow, whether curved or straight. This takes into account the sizes of the objects being connected, thereby giving an aesthetic result when these sizes differ markedly. This also works with `\..compositemap` where an indicator is placed on each arrow. The actual object can be changed using `\modmapobject`.

Any of the standard XY-pic tips may be used for arrow-heads. This is done using `~'{..}`, for example `~'{\dir{>>}}` gives double-headed arrows. Similarly ``~`{..}`` can be used to place an arrow-tail. Normally the arrow-tail is , so is not placed; but if a non-empty tail has been specified then it will be placed, using `\drop`. No guarantee is offered for the desired result being obtained when an arrow-tail is mixed with the features of *Extra Options 1*.

$$\begin{tikzcd} P \arrow[r, bend left, "M"] \arrow[r, bend right, "M'"'] \arrow[r, phantom, "f\,\Vert"] & S \end{tikzcd}$$

```
\modmapobject{\objectbox{\otimes}}
\xymatrixcolsep{5pc}
\diagram
P\rtwocell~!~'{\dir{>>}}~`{\dir{|}}
 ^{<1.5>M}_{<1.5>M'}{=f} & S \\
\enddiagram
```

switch for module maps... ...places decoration called `\modmapobject` half way along arrows

```
\xydef@\modmap@{%
 \expandafter\ifx\the\toks4@\toks4={!}%
 \else\expandafter\ifx\the\toks4=\toks4={H}%
 \else\expandafter\ifx\the\toks4^\toks4={A}%
 \else\expandafter\ifx\the\toks4_\toks4={V}\fi\fi\fi\fi
 \expandafter\DN@\expandafter{\the\toks6}%
 \ifx\next@\empty\relax\else\expandafter\moddefs@\fi}
```

also adjust curvature and label positions

```
\xydef@\moddefs@{%
 \expandafter\dimen@\the\toks6\p@
  \ifdim\dimen@=\z@\relax
  \expandafter\dimen@\the\toks7\p@
   \ifdim\dimen@<\z@\advance\dimen@.5\p@\else\advance\dimen@-.5\p@\fi
   \edef\tmp@{\noexpand\toks7={\expandafter\removePT@\the\dimen@}}\tmp@
  \expandafter\dimen@\the\toks8\p@
   \ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
   \expandafter\ifx\the\toks3t\advance\dimen@-1\p@\fi
   \edef\tmp@{\noexpand\toks8={\expandafter\removePT@\the\dimen@}}\tmp@
 \else
  \ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
  \edef\tmp@{\noexpand\toks6={\expandafter\removePT@\the\dimen@}}\tmp@
%  \expandafter\dimen@\the\toks5\p@
%   \ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
%   \edef\tmp@{\noexpand\toks5={\expandafter\removePT@\the\dimen@}}\tmp@
  \expandafter\dimen@\the\toks7\p@
   \ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
   \expandafter\ifx\the\toks3t\advance\dimen@-2\p@\fi
   \edef\tmp@{\noexpand\toks7={\expandafter\removePT@\the\dimen@}}\tmp@
  \expandafter\dimen@\the\toks8\p@
   \ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
   \expandafter\ifx\the\toks3t\advance\dimen@-2\p@\fi
   \edef\tmp@{\noexpand\toks8={\expandafter\removePT@\the\dimen@}}\tmp@
 \fi }%
```

```
%\def\moddefs@{}
```

default object is `\stop`, user can change this with `\modmapobject#1`

```
\xydef@\modmapobject#1{\def\modmapobject@{#1}}
\xydef@\modmapobject@{\dir{|}}
\xydef@\@modmapobject@{{\if\@mod@\relax\expandafter\empty
 \else\expandafter\modmapobject@\fi}}
```

3. Excursions

Syntax for `\xcompositemap` and `\x..twocell` types is a little different to what might be expected from that for `\xto`, `\xline`, etc. For example,

---

`\xtwocell[`⟨hop⟩`]{`⟨displace⟩`}...`

---

connects to the ⟨pos⟩ displaced by ⟨displace⟩ from the relative cell location specified by ⟨hop⟩. The displacement can be any string of valid XY-pic commands, but they must be enclosed within a group `{...}`. When the cell location is the target, a null grouping `{}` *must* be given.

When used with the `<\omit>` nudge, such excursions allow a labelled Arrow to be placed anywhere within an XY-pic diagram; furthermore the Arrow can be oriented to point in any direction.

```
\xydef@\curveobject#1{\def\ucurveObject@{~*#1}\def\dcurveObject@{~*#1}}
\xydef@\uppercurveobject#1{\def\ucurveObject@{~*#1}}
\xydef@\lowercurveobject#1{\def\dcurveObject@{~*#1}}
\xydef@\ucurveObject@{}
\xydef@\dcurveObject@{}
```

The empty object gives the default of closely spaced tiny dots, resulting in a "smooth" curve.

The following macro is used in conjunction with compiling diagrams, to include the global parameters in the compiled file.

```
\xydef@\savetwocellobjects{%
 \xy@{twocell objects}{%
  \expandafter\uppercurveobject\expandafter{\ucurveObject@}%
  \expandafter\lowercurveobject\expandafter{\dcurveObject@}%
  \expandafter\modmapobject\expandafter{\modmapobject@}%
  \expandafter\twocellhead\expandafter{\twocellhead@}%
  \expandafter\twocelltail\expandafter{\twocelltail@}%
  \expandafter\arrowobject\expandafter{Arrowobject@}%
 }}
```

4. Fancy curves

By specifying `\curveobject` an arbitrary object may be used to construct the curved arrows. Indeed with a `\..twocell` different objects can be used with the upper and lower curves by specifying `\uppercurveobject` and `\lowercurveobject`.

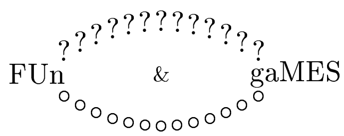
These specifications apply to all 2-cells subsequently constructed at the same level of TEX grouping. Alternatively using a `~`-switch, as in *Extra Options 2*, allows such a specification for a single 2-cell or curved part.

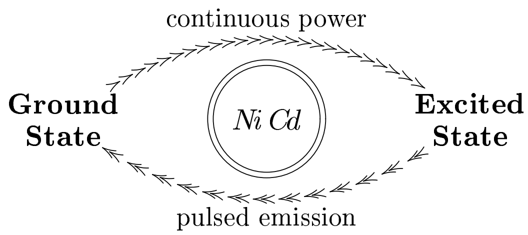
Objects used to construct curves can be of two types. Either a single ⟨object⟩ is set once, with copies placed along the curve. Alternatively a directional object can be aligned with the tangent along the curve. In this case use a specification takes the form:

`\curveobject{`⟨spacer⟩`~**`⟨object⟩`}`.

Here ⟨spacer⟩ may be any ⟨object⟩ of non-zero size. Typically it is empty space, *e.g.* `+`⟨dimen⟩`{}`.

Exercise 3.8: Give code to typeset the following diagrams.





3.4.5 After Parsing...

Values for `\twocellPATH`, depends on the mode of picture/diagram as to whether the `@c` and `@p` are known yet, or still have to be read.

```
\xydef@\twocell@path.[#1].{%
 \expandafter\ifx\csname Q@@c\endcsname\relax
  \DN@{\enter@{\cfromthec@ \pfromthep@ \basefromthebase@}%
   \xy@{}{\expandafter\edef\csname Q@@\endcsname{\cfromthec@}}%
   \afterPOS{\xy@{}%
    {\expandafter\edef\csname Q@@c\endcsname{\cfromthec@}}%
   \xy@{}%
   {\cfromid@{@}\expandafter\edef\csname Q@@p\endcsname{\cfromthec@}}%
   \xy@{twocell #1}{#1}\leave@\aftertwoCELL};p,}%
 \else
  \DN@{\enter@{\cfromthec@ \pfromthep@ \basefromthebase@}%
   \xy@{twocell #1}{#1}\leave@\aftertwoCELL}%
 \fi \next@ }
```

```
\xydef@\twocellpath.[#1].{%
  \expandafter\edef\csname Q@@p\endcsname{\cfromthec@}%
  \xy@{}{\enter@{\pfromthep@}}%
   \xy@{}{\enter@{\cfromthec@ \pfromthep@ \basefromthebase@}%
    \enter@{\pfromthec@}}%
   \afterPOS{\xy@{}{\leave@\setupDirection@}#1\restore\restore
 \xy@{end of 2-cell}\xysaveMinMax@ }}%
```

```
\xydef@\xysaveMinMax@{\edef\tmp@{\endgroup
 \noexpand\ifdim\X@max<\the\X@max \X@max=\the\X@max\noexpand\fi
 \noexpand\ifdim\X@min>\the\X@min \X@min=\the\X@min\noexpand\fi
 \noexpand\ifdim\Y@max<\the\Y@max \Y@max=\the\Y@max\noexpand\fi
 \noexpand\ifdim\Y@min>\the\Y@min \Y@min=\the\Y@min\noexpand\fi
 }\tmp@ }
```

```
\xylet@\twocellPATH=\twocellpath
```

```
\xydef@\aftertwo@CELL{\let\twocellPATH=\twocellpath \def\aftertwoCELL{}}
\xydef@\aftertwoCELL{}
```

It remains to collect everything together and place it on the queue.

```
\xydef@\twocell@@@@#1!!!{\xy@{twocell #1}{\twocell@@#1!!!}}
```

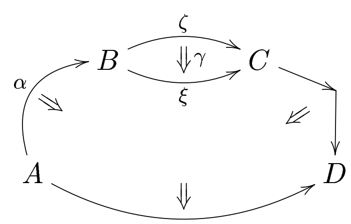
3.4.6 2-cells in general XY-pictures

Two-cells can also be set directly within any XY-picture, without the matrix feature, using either `\drop` or `\connect`.

```
\let\xystatus=\xystatus@
\xydef@\twocell{\hbox\bgroup\@twocell}
\xydef@\uppertwocell{\hbox\bgroup\@uppertwocell}
\xydef@\lowertwocell{\hbox\bgroup\@lowertwocell}
\xydef@\compositemap{\hbox\bgroup\@compositemap}

\xydef@\twocelll#1#{\hbox\bgroup\xy@\save\save@\@twocelll{#1}}
\xydef@\@twocelll#1#2{%
 \afterPOS{\@twocelll@\xyFN@\twocell@#1{#2}}}

\xydef@\uppertwocelll#1#{\hbox\bgroup\xy@\save\save@\@uppertwocelll{#1}}
\xydef@\@uppertwocelll#1#2{%
 \afterPOS{\@twocelll@\xyFN@\uppertwocell@#1{#2}}}

\xydef@\lowertwocelll#1#{\hbox\bgroup\xy@\save\save@\@lowertwocelll{#1}}
\xydef@\@lowertwocelll#1#2{%
 \afterPOS{\@twocelll@\xyFN@\lowertwocell@#1{#2}}}

\xydef@\compositemapp#1#{\hbox\bgroup\xy@\save\save@\@compositemapp{#1}}
\xydef@\@compositemapp#1#2{%
 \afterPOS{\@twocelll@\xyFN@\compositemap@#1{#2}}}
%
\xydef@\@twocelll@{%
% \idfromc@{@c}\swap@\idfromc@{@p}\swap@
 \expandafter\edef\csname Q@@c\endcsname{\cfromthec@}\swap@
 \expandafter\edef\csname Q@@p\endcsname{\cfromthec@}\swap@
 \let\twocellPATH=\twocell@path
% \def\aftertwoCELL{\twocell@DONE}}
 \def\aftertwoCELL{\twocell@DONE \xy@\restore \leave@
% \cfromid@{@c}\no@@ \xystatus@{end: }
 }}

\xydef@\@twocell#1{\DN@{\@twocell@\twocell@}\xyFN@\next@[#1]}
\xydef@\@uppertwocell#1{\DN@{\@twocell@\uppertwocell@}\xyFN@\next@[#1]}
\xydef@\@lowertwocell#1{\DN@{\@twocell@\lowertwocell@}\xyFN@\next@[#1]}
\xydef@\@compositemap#1{\DN@{\@twocell@\compositemap@}\xyFN@\next@[#1]}
```

```
\def\myPOS#1{\POS}\def\goVia#1{%
  \afterPOS{\connect#1\myPOS}}
\xy
 *+{A}="A",+<1cm,1.5cm>*+{B}="B",
 +<2.0cm,0pt>*+{C}="C",
 +<1cm,-1.5cm>*+{D}="D",
```

```
"A";\goVia{\uppertwocell^\alpha{}}"B"{}
;\goVia{\twocell^\zeta_\xi{\gamma}}"C"{}
;\goVia{\compositemap{}}"D"{},
"A";\goVia{\lowertwocell{}}"D"{}
\endxy
```



The code shown is a compact way to place a chain of 2-cells within a picture. It illustrates a standard technique for using `\afterPOS` to find a ⟨pos⟩ to be used for part of a picture, then subsequently reuse it. Also it is possible to use `\drop` or ⟨decor⟩s to specify the 2-cells, giving the same picture.

```
\xy *+{A}="A",+<1cm,1.5cm>*+{B}="B",
 +<2cm,0pt>*+{C}="C",
 +<1cm,-1.5cm>*+{D}="D",
"A";"B",{\uppertwocell^\alpha{}},
"B";"C",{\twocell^\zeta_\xi{\gamma}},
"C"; \afterPOS{\drop\compositemap{}}"D"
\POS "A";
 \afterPOS{\drop\lowertwocell{}}"D"
\endxy
```

The `\connect` variant is usually preferable as this maintains the size of the object at c, while the `\drop` variant leaves a rectangular object having p and c on opposite sides.

This is used outside of diagrams.

```
\xydef@\@twocell@#1{\ifx\next[\DN@[{\toks@={#1}\xyFN@\@@twocell@}%
 \else\DN@{#1{}}\fi
%%
% \idfromc@{@c}\swap@\idfromc@{@p}\swap@
 \expandafter\edef\csname Q@@c\endcsname{\cfromthec@}\swap@
 \expandafter\edef\csname Q@@p\endcsname{\cfromthec@}\swap@
 \let\twocellPATH=\twocell@path
 \def\aftertwoCELL{\twocell@DONE}\next@}
```

This hack is necessary, since `\DN@[##1]{#1##1}` will strip braces.

```
\xydef@\@@twocell@{\ifx\next\bgroup\DN@##1]{\the\toks@{##1}}%
  \else\DN@##1]{\the\toks@##1}\fi\next@}
```

To Do: These are supposed to allow `\dir\...twocell{...}` to work. At present it seems to be impossible to do this.

```
% \expandafter\xydefcsname@\expandafter{\codeof\twocell}{\@twocell}
% \expandafter\xydefcsname@\expandafter{\codeof\uppertwocell}{\@uppertwocell}
% \expandafter\xydefcsname@\expandafter{\codeof\lowertwocell}{\@lowertwocell}
% \expandafter\xydefcsname@\expandafter{\codeof\compositemap}{\@compositemap}
```

```
%% \DN@{\twocell}%
%% \expandafter\def\csname*dir@@\codeof\next@ @\endcsname{\@twocell}
%% \DN@{\uppertwocell}%
%% \expandafter\def\csname*dir@@\codeof\next@ @\endcsname{\@uppertwocell}
%% \DN@{\lowertwocell}%
%% \expandafter\def\csname*dir@@\codeof\next@ @\endcsname{\@lowertwocell}
%% \DN@{\compositemap}%
%% \expandafter\def\csname*dir@@\codeof\next@ @\endcsname{\@compositemap}
```

...since the contents of `{..}` is variable.

Finish off the box, set up the object size, establish the `\Drop@@` and `\Connect@@` methods.

```
\xydef@\twocell@DONE{%
 \edef\tmp@{\egroup \X@min=\the\X@min \X@max=\the\X@max
  \Y@min=\the\Y@min \Y@max=\the\Y@max}\tmp@
 \L@c=\X@c \advance\L@c-\X@min \R@c=\X@max \advance\R@c-\X@c
 \D@c=\Y@c \advance\D@c-\Y@min \U@c=\Y@max \advance\U@c-\Y@c
 \ht\z@=\U@c \dp\z@=\D@c \dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@
 \computeLeftUpness@
%
\setboxz@h{\kern-\X@p \raise-\Y@c\boxz@ }%
\dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@ \ht\z@=\U@c \dp\z@=\D@c
%
 \Edge@c={\rectangleEdge}\Invisible@false \Hidden@false
 \edef\Drop@@{\noexpand\drop@Twocell
  \noexpand\def\noexpand\Leftness@{\Leftness@}%
  \noexpand\def\noexpand\Upness@{\Upness@}}%
 \edef\Connect@@{\noexpand\connect@Twocell
  \noexpand\ifdim\X@max<\the\X@max \X@max=\the\X@max\noexpand\fi
  \noexpand\ifdim\X@min>\the\X@min \X@min=\the\X@min\noexpand\fi
  \noexpand\ifdim\Y@max<\the\Y@max \Y@max=\the\Y@max\noexpand\fi
  \noexpand\ifdim\Y@min>\the\Y@min \Y@min=\the\Y@min\noexpand\fi }}%
```

modified `\Drop@@` and `\Connect@@`

```
\xydef@\drop@Twocell{\boxz@ }
```

```
\xydef@\connect@Twocell{%
 \setboxz@h{\kern\X@p \raise\Y@c\box\lastobjectbox@ }%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \Drop@@ }
```

`\twocell@@` is the main switching engine, taking 9 parameters determining which pieces to set and where to position them. This includes curvature of the squines, whether to draw two squines or only one, or to use straight lines, whether to have a central Arrow and its orientation, contants of labels and their positions.

The current p and c are the extents of the connection these are stored as `"@p"` and `"@c"` for repeated use;

`#1`	determines location of control point `"@m"`
`#2`	location of `=>`, as multiple of `\xydashl@` from `"@m"`
`#3`	location of `_` label (source arrow)
`#4`	location of `^` label (target arrow)
`#5`	flag for how many squines to draw & which tips
`#6`	label for source arrow
`#7`	label for target arrow
`#8`	label for `=>` (2-cell morphism)
`#9`	determines orientation of 2-cell

The following tokens may appear as parameters in `#9`: `!`, `^`, `_`, `=`, `A`, `V`, `H`

```
\xywarnifdefined\@mod@
\xywarnifdefined\Arrowtok@

\xydef@\twocell@@#1#2#3#4#5#6#7#8#9!!!{%
 \def\@mod@{@}\DN@{#9}%
 \ifx #9!\def\Arrowtok@{@}\else\ifx #9A\def\Arrowtok@{^}%
  \else\ifx #9V\def\Arrowtok@{_}\else\ifx #9H\def\Arrowtok@{=}%
  \else\def\@mod@{\relax}%
  \ifx #5/\def\Arrowtok@{@}\else\ifx #5b\def\Arrowtok@{@}%
   \else\ifx #5B\def\Arrowtok@{@}\else\ifx #5@\def\Arrowtok@{@}%
   \else\def\Arrowtok@{#9}%
 \fi\fi\fi\fi\fi\fi\fi\fi
   \addEQ@\ifx #9\def\Arrowtok@{=}\fi
 \DN@{\omit}\ifx\next@\Arrowtok@\def\Arrowtok@{}\else
  \edef\Arrowtok@{\codeof\Arrowtok@}\fi
 \save@ \idfromc@{@c}\swap@\idfromc@{@p}\swap@
 \cfromid@{@c}\no@@
%
 \ifdim \X@c<\X@p \X@min=\X@c \X@max=\X@p \else \X@min=\X@p \X@max=\X@c \fi
 \ifdim \Y@c<\Y@p \Y@min=\Y@c \Y@max=\Y@p \else \Y@min=\Y@p \Y@max=\Y@c \fi
%
 \Calong@@{.5}\idfromc@{@m}\idfromc@{@m1}\idfromc@{@m2}%
%
 \ifx #5t\relax
  \enter@{\cplusthec@\pfromthep@}%
  \enter@\DirectionfromtheDirection@ \begingroup\aboveDirection@ \xydashl@
  \DN@{#1}\ifx\next@\empty\dimen@=\z@\else\dimen@=#1\p@\fi
  \ifdim\dimen@=\z@ \DN@{\vfromslide@i{\xydashl@}@}\relax
  \else\DN@{\vfromslide@i{#1\xydashl@}@}\fi \next@
  \czeroEdge@ \leave@ \no@@ \drop@{+}{}\idfromc@{@m1}%
  \DN@{#1}\ifx\next@\empty\DN@{\no@@}\else\DN@{\connect@{\dir}{-}}\fi
 \else
  \DN@{#1}\ifx\next@\empty\dimen@=\z@\else\dimen@=#1\p@\fi
  \ifdim\dimen@=\z@ \enter@{\cplusthec@\pfromthep@}%
   \enter@\DirectionfromtheDirection@ \begingroup\aboveDirection@\xydashl@
    \vfromslide@i{\xydashl@}@\czeroEdge@ \leave@ \idfromc@{@m1}%
    \cfromid@{@p}\swap@\cfromid@{@c}%
```

```
   \DN@{#1}\ifx\next@\empty\DN@{\no@@}\else\DN@{\connect@{\dir}{-}}\fi
  \else \DN@{\enter@\cplusthec@
   \enter@\DirectionfromtheDirection@ \begingroup\aboveDirection@ \xydashl@
    \vfromslide@i{#1\xydashl@}@\czeroEdge@ \leave@
   \idfromc@{@m1}\cfromid@{@p}\swap@\cfromid@{@c}%
   \edef\next@{\codeof\ucurveObject@}%
   \ifx\next@\empty\DN@{\connect@\crv{"@m1"}}%
   \else
    \DNii@####1{\connect@\crv{####1}}\expandafter\DN@\expandafter{%
     \expandafter\nextii@\expandafter{\ucurveObject@"@m1"}}%
   \fi \next@ }%
%%   \ifx\next@\empty\def\ucurveObject@{.}\fi
%%   \enter@\cfromthec@\sinit@\cfromid@{@m1}\senter@\leave@
%%   \expandafter\connect@\expandafter\crvs\expandafter{\ucurveObject@}%
%%   \sinit@\sleave@ }%
  \fi
 \fi \next@
%
 \DN@{#1}\ifx\next@\empty\relax\else
 \if\@mod@ @\relax
  \enter@\pfromthep@ \Creset@@
  \ifx #5t\relax
   \PLACEedgec@ \PLACEedgep@ \def\PLACEf@{{.5}}%
  \else \def\next@{\crvreset@}%
   \ifx\next@\Creset@@
    \gettwocelledges@ \edef\PLACEf@{{\expandafter\removePT@\the\dimen@}}%
   \else \PLACEedgec@ \PLACEedgep@ \def\PLACEf@{{.5}}\fi
  \fi
  \expandafter\Calong@@\PLACEf@ \czeroEdge@ \leave@
  \edef\tmp@{\codeof\modmapobject@}%
  \ifx\tmp@\empty \DN@{\drop@{\dir}{|}}%
  \else \DNii@##1##{\drop@{##1}}%
   \DN@{\expandafter\nextii@\modmapobject@{}}%
  \fi \next@
 \fi\fi
%
 \DN@{#6}\ifx\next@\empty\DN@{\relax}\else
 \DNii@{{}}\ifx\next@\nextii@\DN@{\relax}\else
  \ifx @#3\relax\DN@{\relax}\else
  \DN@{\enter@\pfromthep@ \Creset@@
   \Calong@@{.5}\czeroEdge@ \leave@
  \enter@\cplusthec@
   \enter@\DirectionfromtheDirection@\begingroup
    \DN@{#1}\ifx\next@\empty\dimen@\z@\else\dimen@=#1\p@\fi
    \ifdim\dimen@<\z@ \belowDirection@ \xydashl@
    \else \aboveDirection@ \xydashl@ \fi
%     \vfromslide@i{#3\xydashl@}@\czeroEdge@ \leave@
%  \droptwocelllabel@{#6}}%
    \dimen@=#3\p@ \if\@mod@ @\relax
```

```
     \advance\dimen@\ifdim\dimen@<\z@-\fi.5\p@\fi
     \edef\next@{\expandafter\removePT@\the\dimen@}%
    \expandafter\vfromslide@i\expandafter{\next@\xydashl@}@\leave@
    \droptwocelllabel@{#6}}%
 \fi\fi\fi \next@
%
%
 \DN@{#1}\ifx\next@\empty\relax\else
 \ifx #5b\relax         % tip at start only
  \enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
   \Calong@@{0}\czeroEdge@ \leave@
  \edef\next@{\codeof\twocelltail@}%
   \ifx\next@\empty \DN@{\dir{>}}%
    \ifx\next@\twocellhead@\drop@{\dir}{<}%
    \else
     \reverseDirection@ \DNii@##1##{\drop@{##1}}%
     \expandafter\nextii@\twocellhead@{}\reverseDirection@
    \fi
   \else
    \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocelltail@{}%
   \fi
 \else\ifx #5@\relax   % no tips at all
 \else                 % tip at end...
  \enter@\pfromthep@ \Creset@@ \PLACEedgec@
   \Calong@@{1}\czeroEdge@ \leave@
  \edef\next@{\codeof\twocellhead@}%
   \ifx\next@\empty\drop@{\dir}{>}%
   \else
    \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocellhead@{}%
   \fi
  \ifx #5B\relax        % ... and also at start.
   \enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
    \Calong@@{0}\czeroEdge@ \leave@
   \edef\next@{\codeof\twocelltail@}%
   \ifx\next@\empty \DN@{\dir{>}}%
    \ifx\next@\twocellhead@\drop@{\dir}{<}%
    \else
     \reverseDirection@\DNii@##1##{\drop@{##1}}%
     \expandafter\nextii@\twocellhead@{}\reverseDirection@
    \fi
   \else
    \DNii@##1##{\drop@{##1}}\expandafter\drop\twocelltail@{}%
   \fi
  \else\ifx #5/\relax
  \else \edef\next@{\codeof\twocelltail@}%
   \ifx\next@\empty\relax\else
    \enter@\pfromthep@ \Creset@@ \PLACEedgep@
     \Calong@@{0}\czeroEdge@ \leave@
    \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocelltail@{}%
```

```
  \fi\fi\fi
 \fi\fi\fi
%
 \cfromid@{@p}\swap@\cfromid@{@c}\no@@ \cfromid@{@m}%
 \ifx #5t\relax
  \idfromc@{@m2}\cfromid@{@m1}\swap@ \cfromid@{@c}\no@@
  \DN@{#1}\ifx\next@\empty\DN@{\no@@}\else\DN@{\connect@{\dir}{-}}\fi
 \else\ifx #5o\relax
  \idfromc@{@m2}\DN@{}%
 \else     %  second curved arrow
  \DN@{#1}\ifx\next@\empty\dimen@=\z@\else\dimen@=#1\p@\fi
  \ifdim\dimen@=\z@
   \enter@{\cplusthec@\pfromthep@}%
   \enter@\DirectionfromtheDirection@ \begingroup \belowDirection@\xydashl@
    \vfromslide@i{\xydashl@}@\czeroEdge@ \leave@ \idfromc@{@m2}%
   \DN@{\cfromid@{@p}\swap@\cfromid@{@c}\no@@}%
  \else \DN@{\enter@\cplusthec@
   \enter@\DirectionfromtheDirection@ \begingroup \belowDirection@ \xydashl@
    \vfromslide@i{#1\xydashl@}@\czeroEdge@ \leave@
   \idfromc@{@m2}\cfromid@{@p}\swap@\cfromid@{@c}%
   \edef\next@{\codeof\dcurveObject@}%
   \ifx\next@\empty\DN@{\connect@\crv{"@m2"}}%
   \else
    \DNii@####1{\connect@\crv{####1}}\expandafter\DN@\expandafter{%
     \expandafter\nextii@\expandafter{\dcurveObject@"@m2"}}%
   \fi \next@ }\fi
%%   \enter@\cfromthec@\sinit@\cfromid@{@m2}\senter@\leave@
%%   \expandafter\connect@\expandafter\crvs\expandafter{\dcurveObject@}%
%%   \sinit@\sleave@
%%  }\fi
 \fi\fi \next@
%
 \DN@{#1}\ifx\next@\empty\relax\else \ifx #5o\relax\else
 \if\@mod@ @\DN@{\enter@\pfromthep@ \Creset@@
  \ifx #5t\relax
   \PLACEedgec@ \PLACEedgep@ \def\PLACEf@{{.5}}%
  \else \def\next@{\crvreset@}%
   \ifx\next@\Creset@@
    \gettwocelledges@ \edef\PLACEf@{{\expandafter\removePT@\the\dimen@}}%
   \else \PLACEedgec@ \PLACEedgep@ \def\PLACEf@{{.5}}\fi
  \fi
   \expandafter\Calong@@\PLACEf@ \czeroEdge@ \leave@
  \edef\tmp@{\codeof\modmapobject@}\ifx\tmp@\empty\DN@{\drop@{\dir}{|}}%
  \else \DNii@####1####{\drop@{####1}}%
   \DN@{\expandafter\nextii@\modmapobject@{}}\fi
  \next@ }\else\DN@{\relax}\fi
 \next@ \fi\fi
%
 \DN@{#7}\ifx\next@\empty\DN@{\relax}\else
```

```
\DNii@{{}}\ifx\next@\nextii@\DN@{\relax}\else
 \ifx @#4\relax\DN@{\relax}\else
 \DN@{\enter@\pfromthep@ \Creset@@
  \Calong@@{.5}\czeroEdge@ \leave@
 \enter@\cplusthec@
 \enter@\DirectionfromtheDirection@ \begingroup
  \DN@{#1}\ifx\next@\empty\dimen@\z@ \else\dimen@=#1\p@\fi
  \ifx #5t\relax \ifdim\dimen@<\z@ \belowDirection@ \xydashl@
   \else \aboveDirection@ \xydashl@ \fi
  \else \ifdim\dimen@<\z@ \aboveDirection@ \xydashl@
   \else \belowDirection@ \xydashl@ \fi
  \fi \dimen@=#4\p@ \if\@mod@ @\relax
    \advance\dimen@\ifdim\dimen@<\z@-\fi.5\p@\fi
    \edef\next@{\expandafter\removePT@\the\dimen@}%
   \expandafter\vfromslide@i\expandafter{\next@\xydashl@}@\leave@
  \droptwocelllabel@{#7}}%
\fi\fi\fi \next@
%
\DN@{#1}\ifx\next@\empty\relax\else
\ifx #5o\relax
\else\ifx #5b\relax  % tip at start only
 \enter@ {\pfromthep@ }\Creset@@ \PLACEedgec@
  \Calong@@{1}\czeroEdge@ \leave@
 \edef\next@{\codeof\twocelltail@}%
 \ifx\next@\empty \DN@{\dir{>}}%
  \ifx\next@\twocellhead@\drop@{\dir}{>}%
  \else
   \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocellhead@{}%
  \fi
 \else
  \DNii@##1##{\drop@{##1}}\expandafter\drop\twocelltail@{}%
 \fi
\else\ifx #5/\relax
 \enter@{\pfromthep@}\Creset@@ \PLACEedgep@
  \Calong@@{0}\czeroEdge@ \leave@
 \edef\next@{\codeof\twocelltail@}%
  \ifx\next@\empty \DN@{\dir{>}}%
   \ifx\next@\twocellhead@\drop@{\dir}{<}%
   \else
    \reverseDirection@\DNii@##1##{\drop@{##1}}%
    \expandafter\nextii@\twocellhead@{}\reverseDirection@
   \fi
  \else
   \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocelltail@{}%
  \fi
\else\ifx #5@\relax    % no tips at all
\else                  % tip at end...
 \enter@\pfromthep@ \Creset@@ \PLACEedgec@
  \Calong@@{1}\czeroEdge@ \leave@
```

```
  \edef\next@{\codeof\twocellhead@}%
   \ifx\next@\empty\drop@{\dir}{>}%
   \else
    \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocellhead@{}%
   \fi
  \ifx #5B\relax        % ... and also at start.
   \enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
    \Calong@@{0}\czeroEdge@ \leave@
   \edef\next@{\codeof\twocelltail@}%
   \ifx\next@\empty\DN@{\dir{>}}%
    \ifx\next@\twocellhead@\drop@{\dir}{<}%
    \else
     \reverseDirection@\DNii@##1##{\drop@{##1}}%
     \expandafter\nextii@\twocellhead@{}\reverseDirection@
    \fi
   \else
    \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocelltail@{}%
   \fi
  \else
   \edef\next@{\codeof\twocelltail@}%
   \ifx\next@\empty\relax\else
    \enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
     \Calong@@{0}\czeroEdge@ \leave@
    \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocelltail@{}%
   \fi\fi
  \fi\fi\fi
 \fi\fi
%
%   the Arrow
%
 \DN@{#1}\ifx\next@\empty\dimen@=\z@\else\dimen@=#1\p@\fi
 \ifdim\dimen@<\z@ \cfromid@{@m2}\swap@\cfromid@{@m1}%
 \else \cfromid@{@m1}\swap@\cfromid@{@m2}%
 \fi \no@@ \cfromid@{@m}%
%
 \ifx\Arrowtok@\empty
%
 \else\DN@{_}\edef\next@{\codeof\next@}\ifx\next@\Arrowtok@
  \enter@{\pfromthep@ \cplusthec@}\dimen@=#2\xydashl@
   \enter@\DirectionfromtheDirection@ \begingroup
    \expandafter\vfromslide@i\expandafter{\the\dimen@}@\czeroEdge@
  \leave@ \idfromc@{@m}%
 \DNii@##1##{\drop@{##1}}\expandafter\nextii@\Arrowobject@{}%
% \drop@{\dir}{=>}%
%
 \else\DN@{^}\edef\next@{\codeof\next@}\ifx\next@\Arrowtok@
   \DN@{_}\edef\Arrowtok@{\codeof\next@}%
  \enter@{\pfromthep@ \cplusthec@}\dimen@=#2\xydashl@
   \enter@\DirectionfromtheDirection@ \begingroup
```

```
    \expandafter\vfromslide@i\expandafter{\the\dimen@}@\czeroEdge@
  \leave@ \idfromc@{@m}\reverseDirection@
  \DNii@##1##{\drop@{##1}}\expandafter\nextii@\Arrowobject@{}%
%  \drop@{\dir}{=>}%
%
 \else \DN@{=}\edef\next@{\codeof\next@}\ifx\next@\Arrowtok@
  \enter@{\pfromthep@ \cplusthec@}%
   \dimen@=#2\xydashl@ \advance\dimen@\xydashl@
   \enter@\DirectionfromtheDirection@ \begingroup
    \expandafter\vfromslide@i\expandafter{\the\dimen@}@\czeroEdge@
  \leave@ \idfromc@{@m2}%
  \cfromid@{@m}%
  \enter@{\pfromthep@ \cplusthec@}%
   \dimen@=#2\xydashl@ \advance\dimen@-\xydashl@
   \enter@\DirectionfromtheDirection@ \begingroup
    \expandafter\vfromslide@i\expandafter{\the\dimen@}@\czeroEdge@
  \leave@ \idfromc@{@m1}%
  \cfromid@{@m2}\swap@\cfromid@{@m1}\connect@{\dir 2}{-}%
 \fi\fi\fi\fi
%
%   label on the Arrow
%
 \DN@{#8}\ifx\next@\empty\DN@{}\else\DN@{%
  \ifx\Arrowtok@\empty
   \Calong@@{.5}%
  \else\DN@{=}\edef\next@{\codeof\next@}\ifx\next@\Arrowtok@
   \Calong@@{.5}%
  \else\DN@{_}\edef\next@{\codeof\next@}\ifx\next@\Arrowtok@
   \cfromid@{@m}%
  \fi\fi\fi
  \enter@\cplusthec@
   \enter@\DirectionfromtheDirection@ \begingroup \aboveDirection@\xydashl@
    \ifx\Arrowtok@\empty
     \def\tmp@{0pt}%
    \else\DN@{=}\edef\next@{\codeof\next@}\ifx\next@\Arrowtok@
     \def\tmp@{\xydashl@}%
    \else\DN@{_}\edef\next@{\codeof\next@}\ifx\next@\Arrowtok@
     \def\tmp@{1.2\xydashl@}%
    \else\def\tmp@{0pt}%
    \fi\fi\fi
    \expandafter\vfromslide@i\expandafter{\tmp@}@\czeroEdge@ \leave@
  \droptwocelllabel@{#8}%
 }\fi \next@
 \leave@ }%
```

These are used to establish the location for the module-map indicator, giving best aesthetic appeal. It finds the location whose parameter value is the average of half-way along the complete curve and the average of parameter values at the the edges.

```
\xydef@\gettwocelledges@{%
  \bgroup \the\crvpts@
```

```
  \edef\next@{\edges@}%
  \expandafter\gettwocelledges@@\next@ }
```

```
\xydef@\gettwocelledges@@#1;#2,#3;#4,{%
 \dimen@=#2\advance\dimen@-#4\relax
 \advance\dimen@ 2\p@ \divide\dimen@ by4 \relax
 \edef\next@{\egroup\dimen@=\the\dimen@ }\next@}%
```

The end & Log

DOCMODE3

```
\xyendinput
```

```
% $Log: xy2cell.doc,v $
% Revision 3.3  1996/12/18 14:21:23  ross
% Ross's version
%
% Revision 3.3.1.1  1996/12/18  08:49:34  ross
% *** empty log message ***
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.14  1995/07/05  22:11:25  kris
% Buglets...
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.12  1994/10/25  03:01:14  ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
%
% Revision 2.10  1994/07/01  01:19:46  ross
% removed undefined references
%
% Revision 2.9  1994/06/09  12:51:27  ross
% Release 3beta.
%
% NEW for version 2.9 based on unreleased code for version 2.6+.
```

3.5 Matrix feature

Vers. 3.4 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

This option implements "Xy-matrices", *i.e.*, matrices where it is possible to refer to the entry objects by their row/column address. We first describe the general form of Xy-matrices in §3.5.1, then in §3.5.2 we summarise the new ⟨coord⟩inate forms used to refer to entries. In §3.5.3 we explain what parameters can be set to change the spacing and orientation of the matrix, and in §3.5.4 we explain how the appearance of the entries can be changed.

Header:

```
%% $Id: xymatrix.doc,v 3.4 1997/05/18 01:14:25 krisrose Exp $
%%
%% Xy-pic ``Matrix'' feature.
%% Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{matrix}{Matrix feature}{\stripRCS$Revision: 3.4 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

3.5.1 XY-matrices

The fundamental command of this feature is

`\xymatrix` ⟨setup⟩ `{`⟨rows⟩`}`

that reads a matrix of entries in the generic TEX row&column format, *i.e.*, where rows are separated with `\\` and contain columns separated with `&` (we discuss in the following sections what ⟨setup⟩ can be). Thus a matrix with *maxrow* rows and *maxcol* columns where each entry contains *row*`,`*col* is entered as

$$\begin{array}{llll}
\texttt{\textbackslash xymatrix\{} & & & \\
\texttt{1,1 \&} & \texttt{1,2 \&} & \cdots & \texttt{1,}\mathit{maxcol}\ \texttt{\textbackslash\textbackslash} \\
\texttt{2,1 \&} & \texttt{2,2 \&} & & \texttt{2,}\mathit{maxcol}\ \texttt{\textbackslash\textbackslash} \\
\vdots & & \ddots & \\
\mathit{maxrow}\texttt{,1 \&} & \mathit{maxrow}\texttt{,2 \&} & & \mathit{maxrow}\texttt{,}\mathit{maxcol}\ \texttt{\}}
\end{array}$$

(TEXnically the `&` character represents any 'alignment tab', *i.e.*, character with category code 4).

A ⟨matrix⟩ can appear either in an XY-picture (as ⟨decor⟩) or "stand-alone".

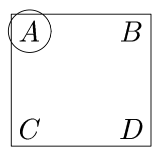
The aspects in which `\xymatrix` differs from ordinary matrix constructions, such as Plain TEX's `\matrix{...}` and LATEX's `array` environment, are

- arbitrary XY-pic ⟨decor⟩ations may be specified in each entry and will be interpreted in a state where c is the current entry,
- the entire matrix is an object itself with reference point as the top left entry, and
- a progress message "`<xymatrix` *rows*`x`*cols* *size*`>`" is printed for each matrix with *rows* × *cols* entries and XY-pic complexity *size* (the number of primitive operations performed), unless the declaration `\SilentMatrices` is issued.
- Entries starting with a `*` are special (described in §3.5.4)[2], so use `{*}` to get a *.

For example,

```
\xy
 \xymatrix{A&B\\C&D}
 \drop\frm{-}
 \drop\cir<8pt>{}
\endxy
```

will typeset



Bug: Matrix nesting is not safe.

Matrices are often quite slow to typeset so as a convenience all matrices can be set to compile (and not) automatically with the declarations

```
\CompileMatrices
\NoCompileMatrices
```

Matrices can be compiled or not individually, by using the explicit commands `\xymatrixcompile` and `\xymatrixnocompile` as well as by encapsulating in the usual `\xycompileto{`*name*`}{...}` (see note 1.5e).

Note: Matrices will only compile correctly if all entries start with a nonexpandable token, *i.e.*, `{` or `\relax` or some non-active character.

On the following code: The complexity of this option stems from the fact that we cannot build the actual XY-picture until after the matrix has been typeset so that we know the sizes and places of all the entries. The handling described here is thus separated into several subprocedures.

Setup: Not much—only to keep track of the rows and columns. These are not accesible to the user—should they?

```
\xynew@{count}\Row
\xynew@{count}\Col
```

[2] In general it is recommended that entries start with a non-expanding token, *i.e.*, an ordinary (non-active) character, `{`, or `\relax`, for compilation to work.

```
\xydef@\maxrow@{0}             % highest row number used
\xydef@\maxcol@{0}             % highest column number used
\xydef@\maxcolrow@{0}          % a row with maximal number of columns
```

There are some user-defined parameters summarised later.

The following is an experimental set of 'optimal' queueing macros using two queues and two counters to keep track of the progress.

```
\xynew@{toks}\queue@
\xynew@{toks}\queue@@

\xynew@{count}\qcount@
\xynew@{count}\qcount@@

\xydef@\clearq@{%
 \global\queue@={}\global\qcount@=\@ne
 \global\queue@@={}\global\qcount@@=\z@}

\xydef@\addq@#1{\global\advance\matrixsize@\@ne
 \ifnum\qcount@@<\qcount@ \let\addq@@=\addq@i
 \else \let\addq@@=\addq@ii \fi
 \expandafter\addq@@\the\queue@@\addq@@{#1}}

\xylet@\addq@@=\relax

\xydef@\addq@i#1\addq@@#2{%
 \global\advance\qcount@@\@ne \global\queue@@={#1#2}}

\xydef@\addq@ii#1\addq@@#2{%
 \global\advance\qcount@\@ne \global\qcount@@=\z@
 \global\queue@=\expandafter{\the\queue@#1#2}\global\queue@@={}}

\xydef@\finishq@{\expandafter\addq@ii\the\queue@@\addq@@{}}
```

Hack: intended to make the addition of tokens to the token list faster: we first add to the 'small' `\queue@@` as many times as we have added to the 'big' `\queue@` before that. So we know we should always expand `\queue@@`: this is done first (last in the macro text). Then we call `\addq@@` that will either add to the small one or add the small one to the big one and then clear the small one!

Usage: Call `\clearq@` first. Then use `\addq@` as many times as desired. To get the queue in `\queue@` call `\finishq@` and use it. Finally clear `\queue@` (globally) or continue.

Main procedure: Here is the main code which is concerned with initialisations and output of messages explaining how far the matrix construction has progressed.

```
\xynew@{count}\matrixsize@
\xynew@{if}\ifnoisymatrices@ \noisymatrices@true

\xydef@\SilentMatrices{\noisymatrices@false}
\xydef@\NoisyMatrices{\noisymatrices@true}

\xydef@\xymatrixnocompile{\global\matrixsize@=\z@
 \ifnoisymatrices@\message{<xymatrix}\fi
 \if\inxy@
  \DN@{\nter@{}\xy@\xymatrix{\enter@{%
     \def\noexpand\xy@minX@@{\xy@minX@@}\def\noexpand\xy@maxX@@{\xy@maxX@@}%
     \def\noexpand\xy@minY@@{\xy@minY@@}\def\noexpand\xy@maxY@@{\xy@maxY@@}%
     \edef\noexpand\xymatrixprefix@@{\codeof\xymatrixprefix@@}}}}%
 \else \DN@{\xy \nter@\endxy \xy@\xymatrix{\nter@{}}}\fi \next@
```

```
 \xy@@{\save@ \edef\matrixorigin@{\X@c=\the\X@c \Y@c=\the\Y@c}%
  \edef\xy@minX@@{\the\X@c}\edef\xy@maxX@@{\the\X@c}%
  \edef\xy@minY@@{\the\Y@c}\edef\xy@maxY@@{\the\Y@c}}%
 \xdef\maxrow@{0}\xdef\maxcol@{0}\xdef\maxcolrow@{0}%
 \mkHWdefaults@ \xymatrix@prefix""}
```

```
\xylet@\xymatrix=\xymatrixnocompile
```

```
\xydef@\xymatrixprefix@@{}
```

```
\xydef@\xymatrix@prefix"#1"{\DN@{#1}\edef\xymatrixprefix@@{\codeof\next@}%
 \DNii@##1{\xy@{"#1"}{\edef\xymatrixprefix@@{##1}}}%
 \expandafter\nextii@\expandafter{\xymatrixprefix@@}%
 \xyFN@\xymatrix@setup}
```

```
\xydef@\xymatrix@setup{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xymatrix@setup}%gobble spaces
 \else\ifx \bgroup\next \let\next@=\xymatrix@ii
 \else\addAT@\ifx\next \addAT@\DN@{\xy@{@}{}\xyFN@\xymatrix@at}%
 \else\ifx "\next \let\next@=\xymatrix@prefix
 \else \DN@{\xyerror@{\string\xymatrix<setup>{<rows>} expected}{}}%
 \fi\fi\fi\fi \next@}
```

```
\xydef@\xymatrix@ii#1{%
%
%\W@{ H@max=\meaning\H@max}%
%\W@{ W@max=\meaning\W@max}%
%\W@{ HW@max=\meaning\HW@max}%
%\W@{ mkHrow@={\meaning\mkHrow@}}%
%\W@{ mkWcol@={\meaning\mkWcol@}}%
%\W@{ mkHmax@={\meaning\mkHmax@}}%
%\W@{ mkWmax@={\meaning\mkWmax@}}%
%\W@{ mkHWmax@={\meaning\mkHWmax@}}%
%
 \xymatrix@pretypeset{#1}%
%  \ifnum\maxcol@=\z@ \mkWcol@ \mkWmax@ %\mkHrow@ \mkHmax@
  \ifnum\maxcol@=\z@ \mkWcol@ \mkWmax@ \mkHrow@ \mkHmax@
   \xdef\maxcol@{\the\Col}\xdef\maxcolrow@{\the\Row}\fi
  \ifnoisymatrices@\message{\maxcol@ x\maxrow@}\fi
  \xymatrix@measureit@@ \the\queue@ \global\queue@={}%
  \xymatrix@typeset{#1}\restore\xymatrix@insert
  \xy@@\leave@ \leave@
  \ifnoisymatrices@\message{\the\matrixsize@>}\fi \ignorespaces}
```

Here is where compilation of matrices is turned on/off.

```
\xydef@\xymatrixcompile#1#{\xymatrixcompile@{#1}}
\xydef@\xymatrixcompile@#1#2{\xycompile{\xymatrixnocompile#1{#2}}}
```

```
\xydef@\CompileMatrices{\let\xymatrix=\xymatrixcompile}
\xydef@\NoCompileMatrices{\let\xymatrix=\xymatrixnocompile}
```

To Do: Proper nesting that ensures that the matrix state is reestablished after the current one has been typeset.

Each subprocedure is explained below.

Set row-rotation: This sets the general direction of the rows as the current direction indicates. This direction is not known until XY-time so it gets put into the row/column definitions.

```
\xydef@\xy@cossign@@{+}
\xydef@\xy@sinsign@@{+}
\xydef@\xy@cosabs@@{1}
\xydef@\xy@sinabs@@{0}
\xydef@\xy@tanabs@@{0}
\xydef@\xy@cotabs@@{100}

\xydef@\xymatrix@rotation{
 \xy@@{%
  \edef\xy@cossign@@{\ifdim\cosDirection\p@<\z@-\else+\fi}%
  \edef\xy@sinsign@@{\ifdim\sinDirection\p@<\z@-\else+\fi}%
  \edef\xy@cosabs@@{%
   \if\xy@cossign@@\cosDirection\else\xy@cossign@@\cosDirection\fi}%
  \edef\xy@sinabs@@{%
   \if\xy@sinsign@@\sinDirection\else\xy@sinsign@@\sinDirection\fi}%
  \dimen@=\xy@cosabs@@\p@ \dimen@ii=\xy@sinabs@@\p@
  \ifdim\dimen@ii<.01\p@ \def\xy@cotabs@@{100}%
  \else \quotient@\xy@cotabs@@\dimen@\dimen@ii \fi
  \dimen@=\xy@cosabs@@\p@ \dimen@ii=\xy@sinabs@@\p@
  \ifdim\dimen@<.01\p@ \def\xy@tanabs@@{100}%
  \else \quotient@\xy@tanabs@@\dimen@ii\dimen@ \fi}}
```

Pretypeset: The purpose of this is to

- set box8 to an halign with all the entries properly typeset to be measured in the next step, and
- store in a queue the operations that will define each entry as an object once the measurement is done and the distances between rows and columns is known.

This is achieved as follows: Row/column counters, maxima, and queues are reset. The tab- and interlineskip is removed to make taking the constructed box apart easier. And the halign template is used to apply an appropriate action for each entry that updates the counters and queue.

The special `\xymatrixsavedCOORD@` is used to store the original `\COORD@` except when diagrams are nested.

```
\xylet@\xymatrixsavedCOORD@=\relax

\xydef@\xymatrix@pretypeset#1{%
 \global\Row=\z@ \xdef\maxcol@{0}\clearq@
 \setbox8=\vtop{%
  \ifx\xymatrixsavedCOORD@\relax \let\xymatrixsavedCOORD@=\COORD@ \fi
  \let\COORD@=\xymatrixCOORD@
  \everycr{}\let\\=\cr \tabskip=\z@ \offinterlineskip
  \halign{\prentry@ ##!@&&\xyFN@\prentry@@##!@\cr#1\crcr}%
  \prentry@@@}%
 \xdef\maxrow@{\the\Row}%
 \finishq@}
```

The first entry in each row is special; in particular it moves the entries from the columns of the previous row (if any) into the second-level queue:

```
\xydef@\prentry@{\relax
 \ifnum\Row>\z@
  \ifnum\maxcol@<\Col\xdef\maxcol@{\the\Col}\xdef\maxcolrow@{\the\Row}\fi
  \expandafter\xdef\csname maxcol@\the\Row\endcsname{\the\Col}\fi
 \global\advance\Row\@ne \global\Col=\z@ \xyFN@\prentry@@}
```

The common part should first typeset the object. Then the resulting ⟨setup⟩ commands should be emitted as described below.

While typesetting the entry, XY-commands are *disabled* by redefining `\xy@` to just 'eat' its arguments, so they don't interfere. . .

```
\xylet@\lastprentry@@=\relax

\xydef@\xyeatall@{\let\xy@=\xyeat@ \change@oxy@\xyeat@ \let\xy@@ix@=\eat@}

\xydef@\prentry@@{\global\advance\Col\@ne
 \let\next@=\prentry@@norm
 \ifoldxymatrix@\else \ifx *\next \DN@*{\xyFN@\prentry@@star}\fi \fi
 \next@}

\xydef@\prentry@@star{\ifx *\next \DN@*{\xyFN@\prentry@@star@ii}%
 \else \DN@##1##{\prentry@@star@i{##1}}\fi \next@}

\xydef@\prentry@@star@i#1#2#3!@{\gdef\lastprentry@@{*#1{#2}#3}%
 \setbox\lastobjectbox@=\object#1{#2}%
 \prentry@@x}

\xydef@\prentry@@star@ii{%
 \ifx [\next
  \DN@[##1]{\DN@####1{\def\entrymodifiers@{[##1]####1}}%
    \expandafter\next@\expandafter{\entrymodifiers@}\prentry@@norm}%]
 \else
  \DN@##1{\DN@####1{\def\entrymodifiers@{##1####1}}%
    \expandafter\next@\expandafter{\entrymodifiers@}\prentry@@norm}%
 \fi \next@}

\xydef@\prentry@@norm#1!@{\gdef\lastprentry@@{#1}%
 \DN@{\setbox\lastobjectbox@=\object}%
 \expandafter\next@\entrymodifiers@{\xyeatall@ \everyentry@ #1}%
 \prentry@@x}

\xydef@\prentry@@x{%
 \edef\next@{{\noexpand\pre@emit{\the\Row}{\Upness@}{\the\Col}{\Leftness@}%
  {\the\L@c}{\the\R@c}{\the\D@c}{\the\U@c}{\expandafter\noexpand\the\Edge@c}}}%
 \expandafter\addq@\next@ \box\lastobjectbox@}
```

Finally the following ensures that the last row in the diagram is treated correctly: if the last entry was an empty row then we ignore the row entirely if the `\ifoldxymatrix@` switch is set just as we ignore `*`-entries above.

```
\xynew@{if}\ifoldxymatrix@ \oldxymatrix@false

\xydef@\prentry@@@{\relax \let\next@=\prentry@@@i
 \ifoldxymatrix@ \ifnum\Col=\@ne \ifx\lastprentry@@\empty
    \global\advance\Row\m@ne \let\next@=\relax \fi\fi\fi
 \global\let\lastprentry@@=\relax \next@}

\xydef@\prentry@@@i{%
```

```
430  \ifnum\maxcol@<\Col \xdef\maxcol@{\the\Col}\xdef\maxcolrow@{\the\Row}\fi
431  \expandafter\xdef\csname maxcol@\the\Row\endcsname{\the\Col}}
```

Furthermore the code (1.464) sets the edge and extents of the object and (1.465) adds the object to the size of the matrix. **Bug:** Currently ignores upness and leftness parameters.

```
439 \xydef@\xy@minX@@{\z@}
440 \xydef@\xy@maxX@@{\z@}
441 \xydef@\xy@minY@@{\z@}
442 \xydef@\xy@maxY@@{\z@}
```

```
444 \xydef@\pre@emit#1#2#3#4#5#6#7#8#9{%
445  % #1=R,#2=Upn,#3=C,#4=Leftn,#5=L,#6=R,#7=D,#8=U,#9=Edge.
446  \global\Row=#1\global\Col=#3\relax
447  \ifnum#3=\@ne
448   \ifnum#1=\@ne \xy@@{\matrixorigin@}
449   \else \count@@=#1\advance\count@@\m@ne
450    \edef\next@{\noexpand\xy@@{\noexpand\cfromid@
451      {\xymatrixprefix@@\the\count@@,1}}}\next@
452    \pre@emit@ \W@maxout\Hrow@c@@\W@maxin\Hrow@in\xy@tanabs@@
453     \X@c\xy@sinsign@@\xy@sinabs@@\xymatrixrowsep@
454    \pre@emit@ \Hrow@c@@\W@maxout\Hrow@in\W@maxin\xy@cotabs@@
455     \Y@c{-\xy@cossign@@}\xy@cosabs@@\xymatrixrowsep@
456   \fi
457  \else \count@@=#3\advance\count@@\m@ne
458    \pre@emit@ \Wcol@c@@\H@maxout\Wcol@in\H@maxin\xy@cotabs@@
459     \X@c\xy@cossign@@\xy@cosabs@@\xymatrixcolsep@
460    \pre@emit@ \H@maxout\Wcol@c@@\H@maxin\Wcol@in\xy@tanabs@@
461     \Y@c\xy@sinsign@@\xy@sinabs@@\xymatrixcolsep@
462  \fi
463  \DN@##1{\xy@@{\L@c=#5\R@c=#6\D@c=#7\U@c=#8\Edge@c={#9}\idfromc@{##1#1,#3}}}%
464  \expandafter\next@\expandafter{\xymatrixprefix@@}
465  \xy@@{
466   \dimen@=\X@c \advance\dimen@-\L@c
467    \ifdim\dimen@<\xy@minX@@ \edef\xy@minX@@{\the\dimen@}\fi
468   \dimen@=\X@c \advance\dimen@+\R@c
469    \ifdim\dimen@>\xy@maxX@@ \edef\xy@maxX@@{\the\dimen@}\fi
470   \dimen@=\Y@c \advance\dimen@-\D@c
471    \ifdim\dimen@<\xy@minY@@ \edef\xy@minY@@{\the\dimen@}\fi
472   \dimen@=\Y@c \advance\dimen@+\U@c
473    \ifdim\dimen@>\xy@maxY@@ \edef\xy@maxY@@{\the\dimen@}\fi}}
```

The following procedure represents the similarity of the four lines for the coordinates in the computation above... **To Do:** describe this properly...

```
481 \xydef@\pre@emit@#1#2#3#4{#1\A@ #2\B@ #3\R@ #4\dimen@
482  \edef\next@{{\the\A@}{\the\B@}{\the\R@}{\the\dimen@}}%
483  \expandafter\pre@emit@i\next@}
```

```
485 \xydef@\pre@emit@i#1#2#3#4#5{\xy@@{%
486   % Emit R := min(#1,#5#2) + min(#3,#5#4)
487   \dimen@=#2\R@=#5\dimen@ \dimen@=#1\relax \ifdim\R@>\dimen@ \R@=\dimen@ \fi
488   \dimen@=#4\B@=#5\dimen@ \dimen@=#3\relax \ifdim\B@>\dimen@ \B@=\dimen@ \fi
```

```
  \advance\R@\B@}%
 \pre@emit@ii}

\xydef@\pre@emit@ii#1#2#3#4{\xy@@{%
  % Emit #1 := #1 + #2*[ R + #3#4 ]
  \dimen@=#4\dimen@=#3\dimen@ \advance\R@\dimen@ \advance#1#2\R@}}

\xydef@\Wcol@in#1{#1=\csname Wcol@\the\Col\endcsname #1=.5#1}
\xydef@\Wcol@out#1{#1=\csname Wcol@\the\Col\endcsname #1=.5#1}
\xydef@\Wcol@c@@#1{#1=\csname Wcol@\the\count@@\endcsname #1=.5#1}
\xydef@\W@maxin#1{#1=\W@max #1=.5#1}
\xydef@\W@maxout#1{#1=\W@max #1=.5#1}

\xydef@\Hrow@in#1{#1=\csname Hrow@\the\Row\endcsname #1=.5#1}
\xydef@\Hrow@out#1{#1=\csname Hrow@\the\Row\endcsname #1=.5#1}
\xydef@\Hrow@c@@#1{#1=\csname Hrow@\the\count@@\endcsname #1=.5#1}
\xydef@\H@maxin#1{#1=\H@max #1=.5#1}
\xydef@\H@maxout#1{#1=\H@max #1=.5#1}
```

Bug: It should be possible to change te adjustment more profoundly for individual entries, rows, and columns.

Measure: Take the created box8 apart to define the macros `\Wcol@`*col* and `\Hrow@`*row* containing the width and height of the bounding box of the entries in the row/column. Finally the commands collected in the queue are executed (hopefully they'll invoke `\pre@emit` a number of times). This is a hook because some special cases won't need it…

```
\xydef@\H@max{\z@}
\xydef@\W@max{\z@}
\xydef@\HW@max{\z@}

\xydef@\mkHrow@{}
\xydef@\mkWcol@{}
\xydef@\mkHmax@{}
\xydef@\mkWmax@{}
\xydef@\mkHWmax@{}

\xydef@\mkHWdefaults@{\let\xymatrix@measureit@@=\xymatrix@measureit
 \let\mkHrow@=\Hrow@R \let\mkWcol@=\Wcol@C
 \def\mkHmax@{\ifdim\dimen@>\H@max\relax \xdef\H@max{\the\dimen@}\fi}%
 \def\mkWmax@{\ifdim\dimen@>\W@max\relax \xdef\W@max{\the\dimen@}\fi}%
 \let\mkHWmax@=\HWmax@max}

\xydef@\Hrow@R{\expandafter\xdef\csname Hrow@\the\Row\endcsname{\the\dimen@}}
\xydef@\Wcol@C{\expandafter\xdef\csname Wcol@\the\Col\endcsname{\the\dimen@}}

\xydef@\Hrow@max{\expandafter\gdef\csname Hrow@\the\Row\endcsname{\H@max}}
\xydef@\Wcol@max{\expandafter\gdef\csname Wcol@\the\Col\endcsname{\W@max}}

\xydef@\Hrow@HWmax{\expandafter\gdef\csname Hrow@\the\Row\endcsname{\HW@max}}
\xydef@\Wcol@HWmax{\expandafter\gdef\csname Wcol@\the\Col\endcsname{\HW@max}}

\xydef@\HWmax@max{%
 \ifdim\H@max>\W@max\global\let\HW@max=\H@max\else\global\let\HW@max=\W@max\fi}

\xydef@\HWmax@set#1{\gdef\HW@max{#1}%
 \global\let\H@max=\HW@max \global\let\W@max=\HW@max}
```

```
\xydef@\xymatrix@measureit{%
 \xdef\H@max{\z@}\xdef\W@max{\z@}%
 \setbox6=\vbox{\unvbox8 \global\Row=\maxrow@
  \loop@ \dimen@ii=\lastskip\unskip \ifnum\z@<\Row
   \setbox4=\lastbox \dimen@=\dp4 \advance\dimen@\ht4 \mkHrow@ \mkHmax@
   \ifnum\Row=\maxcolrow@
    \setbox2=\hbox{\unhbox4 %
     \global\Col=\maxcol@
     \loop@ \dimen@ii=\lastskip\unskip \ifnum\z@<\Col
      \setbox\z@=\lastbox \dimen@=\wdz@ \mkWcol@ \mkWmax@
      \global\advance\Col\m@ne
     \repeat@}\fi
    \global\advance\Row\m@ne
   \repeat@}%
 \mkHWmax@}
```

```
\xylet@\xymatrix@measureit@@=\relax
```

To Do: Allow non-default spacing between particular rows/columns as well as different justification of the entire matrix. . .

Typeset: Retypeset the entire diagram using a hacked halign: its template typesets each entry as

`\POS` "*row* , *col*" `*` *object* `\relax` *Xy-commands* `\relax`

where *object* is the contents of the entry translated into Xy-pic ⟨object⟩ form and *Xy-commands* entails the commands used in this particular entry.

```
\xydef@\xymatrix@typeset#1{%
 \clearq@
 \hbox{\vtop{\xyqall@
   \xy@@{\ifx\xymatrixsavedCOORD@\relax
     \enter@{\noexpand\let\noexpand\COORD@\noexpand\xymatrixsavedCOORD@}%
     \let\xymatrixsavedCOORD@=\COORD@ \let\COORD@=\xymatrixCOORD@
    \else \enter@{}\fi}%
   \ifx\xymatrixsavedCOORD@\relax \let\xymatrixsavedCOORD@=\COORD@ \fi
   \let\COORD@=\xymatrixCOORD@
   \kern\z@ \global\Row=\z@
   \everycr{}\let\\=\cr \tabskip=\z@ \offinterlineskip
   \halign{\entry@##!@&&\entry@@##!@\cr#1\crcr}%
   \xy@@{\leave@}}}%
 \finishq@ \expandafter\xy@@\expandafter{\the\queue@}\global\queue@={}}
```

Here are the template execution macros: We reset the `\queue@` for each entry, `\drop` it the usual way except first all Xy-commands are 'redirected' to the queue, and then typeset them.

```
\xydef@\entry@{\relax \global\advance\Row\@ne \global\Col=\z@ \entry@@}
```

Again some parsing determines whether this is a `*`-object or a normal object:

```
\xydef@\entry@@{\global\advance\Col\@ne
 \DN@##1,##2,##3@{\xy@{ENTRY "##3##1,##2"}{\Row=##1 \Col=##2\relax
   \cfromid@{##3##1,##2}\pfromc@}}%
 \edef\nextii@{\the\Row,\the\Col,\xymatrixprefix@@ @}%
 \expandafter\next@\nextii@
```

```
 \xyFN@\entry@@i}

\xydef@\entry@@i{\let\next@=\entry@@norm
 \ifoldxymatrix@\else \ifx *\next \DN@*{\xyFN@\entry@@star}\fi \fi
 \next@}

\xydef@\entry@@star{\ifx *\next \DN@*{\xyFN@\entry@@star@ii}%
 \else \DN@##1##{\entry@@star@i{##1}}\fi \next@}

\xydef@\entry@@star@i#1#2#3!@{%
 \xy@@ix@{{#1}{#2}}\xy@@{\expandafter\dropentry@\the\toks9}%
 \setboxz@h{\xyqall@
  \ifx\xymatrixsavedCOORD@\relax \let\xymatrixsavedCOORD@=\COORD@ \fi
  \let\COORD@=\xymatrixCOORD@ \everyentry@ #3}%
 \setbox\z@=\copy\voidb@x}

\xydef@\entry@@star@ii{%
 \ifx [\next
  \DN@[##1]{\DN@####1{\def\entrymodifiers@{[##1]####1}}%
    \expandafter\next@\expandafter{\entrymodifiers@}\entry@@norm}%]
 \else
  \DN@##1{\DN@####1{\def\entrymodifiers@{##1####1}}%
    \expandafter\next@\expandafter{\entrymodifiers@}\entry@@norm}%
 \fi \next@}

\xydef@\entry@@norm#1!@{%
 \DN@##1{\expandafter\xy@@ix@
  \expandafter{\expandafter{\entrymodifiers@}{##1}}}%
 \expandafter\next@\expandafter{\everyentry@#1}%
 \xy@@{\expandafter\dropentry@\the\toks9}%
 \setbox\z@=\expandafter\object\entrymodifiers@{\xyqall@
  \ifx\xymatrixsavedCOORD@\relax \let\xymatrixsavedCOORD@=\COORD@ \fi
  \let\COORD@=\xymatrixCOORD@ \everyentry@ #1}%
 \setbox\z@=\copy\voidb@x}
```

Dropping the entry is just retypesetting it at the right location again ignoring the XY-stuff:

```
\xydef@\dropentry@#1#2{\drop@{#1}{\xyeatall@
  \ifx\xymatrixsavedCOORD@\relax \let\xymatrixsavedCOORD@=\COORD@ \fi
  \let\COORD@=\xymatrixCOORD@ #2}}
```

The macros used for queuing are these; only the toks9 replacement is tricky; furthermore queueing is *not* done if we are saving because the effect will give the same result without risk of extremely long lines in the compiled file!

```
\xydef@\xyqall@{%
 \ifxysaving@ \let\xy@=\xyqs@ \let\xy@@ix@=\xysave@@toksix@
 \else \let\xy@=\xyq@ \let\xy@@ix@=\xyq@@toksix@ \fi
 \change@oxy@\xy@}

\xydef@\xyq@#1#2{\addq@{#2}}

\xydef@\xyq@@toksix@{\begingroup
 \xyuncatcodes \afterassignment\xyq@@toksix@i \global\toks9=}

\xydef@\xyq@@toksix@i{%
 \DN@##1{\endgroup \xy@@{\global\toks9={##1}}}%
```

```
\expandafter\next@\expandafter{\the\toks9}}
\xydef@\xyqs@{\global\advance\matrixsize@\@ne \xysave@}
```

Insert: Finally we insert the matrix in the XY-picture logically by ensuring that it has the right size

```
\xydef@\xymatrix@insert{%
 \xy@@{\edef\nextii@{{\xy@minX@@}{\xy@maxX@@}{\xy@minY@@}{\xy@maxY@@}}%
  \expandafter\xymatrix@insert@\nextii@}}
\xydef@\xymatrix@insert@#1#2#3#4{\matrixorigin@
  \L@c=+\X@c \advance\L@c-#1\relax
  \R@c=-\X@c \advance\R@c+#2\relax
  \D@c=+\Y@c \advance\D@c-#3\relax
  \U@c=-\Y@c \advance\U@c+#4\relax
  \Edge@c={\rectangleEdge}}
```

To Do: Should handle other forms than reference point at center of "1,1" as hardcoded here. In particular reference point at baseline of "1,1" would be useful. . .

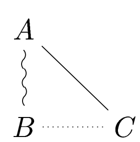
3.5.2 New coordinate formats

It is possible within entries to refer to all the entries of the XY-matrix using the following special ⟨coord⟩inate forms:

"r,c"	Position and extents of entry in row r, column c (top left is "1,1")
[Δr,Δc]	Δr rows below, Δc columns right of current entry
[⟨hop⟩*]	entry reached by ⟨hop⟩s; each ⟨hop⟩ is one of `dulr` describing one 'move' to a neighbor entry
[⟨hop⟩$^+$ ⟨place⟩]	⟨place⟩ on straight line to *non-empty* [⟨hop⟩*]

So the current entry has the synonyms [0,0], [], [rl], [ud], [dudu], etc., as well as its 'absolute' name "r,c".

These forms are useful for defining diagrams where the entries are related, *e.g.*,



was typeset by

```
$$\xy
\xymatrix{
 A \POS[];[d]**\dir{~},
       [];[dr]**\dir{-}    \\
 B & C \POS[];[l]**\dir{.} }
\endxy$$
```

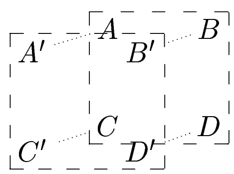
If an entry outside the XY-matrix is referenced then an error is reported.

In case several matrices are used in the same diagram, and they refer to each other, then it is useful to give the matrices different "⟨prefix⟩" ⟨setup⟩ such that they can refer to each other using the following special coordinate forms that all have the same meaning except the target entry is picked from a particular matrix:

"⟨prefix⟩r,c"
["⟨prefix⟩"Δr,Δc]
["⟨prefix⟩" ⟨hop⟩*]
["⟨prefix⟩" ⟨hop⟩$^+$ ⟨place⟩]

In fact absolute references *must* always be given using "⟨prefix⟩⟨row⟩,⟨col⟩", *even inside the matrix itself*.

Here is an example using this:



was typeset (using the 'frame' extension and 'arrow' feature) by

```
\xy
  \xymatrix"*"{%
   A & B \\
   C & D }%
  \POS*\frm{--}
  \POS-(10,3)
  \xymatrix{%
   A’ \ar@{.}["*"] & B’ \ar@{.}["*"] \\
   C’ \ar@{.}["*"] & D’ \ar@{.}["*"] }%
  \POS*\frm{--}
\endxy
```

These handle the parsing and interpretation of bracketed ⟨coord⟩inates; the number is computed already at parse time:

```
\xylet@\thematrixCOORD@@=\relax
\xylet@\thematrixprefix@@=\relax

\xydef@\xymatrixCOORD@{%
 \ifx [\next %]
  \DN@[##1]{\xy@{[##1]}{\def\thematrixCOORD@@{[##1]}}%
    \xyFN@\xymatrixCOORD@i##1]}%
 \else \let\next@=\xymatrixsavedCOORD@ \fi \next@}

\xydef@\xymatrixCOORD@i{\count@=\Row \count@@=\Col
 \let\thematrixprefix@@=\xymatrixprefix@@
 \ifcat A\noexpand\next
  \DN@##1{\expandafter\notrelaxorelse@\csname xymatrixCOORD@i@##1\endcsname
    \xymatrixCOORD@other}%[[
 \else\ifx ]\next \DN@]{\xymatrixCOORD@x}%
 \else\ifx "\next \DN@"##1"{\DN@{##1}\edef\thematrixprefix@@{\codeof\next@}%
     \xyFN@\xymatrixCOORD@ii}%
 \else \let\next@=\xymatrixCOORD@other \fi\fi\fi \next@}

\xydef@\xymatrixCOORD@i@u{\advance\count@ \m@ne\xyFN@\xymatrixCOORD@ii}
\xydef@\xymatrixCOORD@i@d{\advance\count@ \@ne \xyFN@\xymatrixCOORD@ii}
```

```
\xydef@\xymatrixCOORD@i@l{\advance\count@@\m@ne\xyFN@\xymatrixCOORD@ii}
\xydef@\xymatrixCOORD@i@r{\advance\count@@\@ne \xyFN@\xymatrixCOORD@ii}
```

```
\xydef@\xymatrixCOORD@ii{%
 \ifcat A\noexpand\next
  \DN@##1{\expandafter\notrelaxorelse@\csname xymatrixCOORD@i@##1\endcsname
   \xymatrixCOORD@error}%
 \else\ifx ]\next \DN@]{\xymatrixCOORD@x}%
 \else \let\next@=\xymatrixCOORD@xx \fi\fi \next@}
```

```
\xydef@\xymatrixCOORD@error#1{\DN@{#1}%
 \xyerror@{illegal <coord> (\codeof\next@): illegal [...] target form}{}%
 \xyFN@\xymatrixCOORD@ii}
```

```
\xydef@\xymatrixCOORD@other#1,#2]{%
 \advance\count@#1\advance\count@@#2\xymatrixCOORD@x}
```

The next does the actual lookup at XY-time where we can still print an error message using the saved original form.

```
\xydef@\xymatrixCOORD@x{%
 \DN@##1{\xy@@{%
   \expandafter\let\expandafter\next@\csname Q@##1\endcsname
   \ifx\next@\relax \xyerror@RC{##1}\else \next@ \fi}%
  \afterCOORD@}%
 \edef\nextii@{{\thematrixprefix@@\the\count@,\the\count@@}}%
 \expandafter\next@\nextii@}
```

```
\xydef@\xymatrixCOORD@xx{%
 \DN@##1{\xy@@{%
   \expandafter\let\expandafter\next@\csname Q@##1\endcsname
   \ifx\next@\relax \xyerror@RC{##1}\else \next@\no@@ \fi}%
  \afterPLACE\xymatrixCOORD@after}%
 \edef\nextii@{{\thematrixprefix@@\the\count@,\the\count@@}}%
 \expandafter\next@\nextii@}
```

```
\xydef@\xymatrixCOORD@after{%
 \ifx ]\next \DN@]{\afterCOORD@}%
 \else \let\next@=\xymatrixCOORD@error \fi \next@}
```

```
\xydef@\xyerror@RC#1{\xyerror@{in entry \string"\the\Row,\the\Col\string":
        No \codeof\thematrixCOORD@@\space(is \string"#1\string") from here}{}}
```

3.5.3 Spacing and rotation

Any matrix can have its spacing and orientation changed by adding ⟨setup⟩ 'switches' between `\xymatrix` and the opening `{`.

The default spacing between entries of matrix is changed with the switches

`@R`⟨add op⟩ ⟨dimen⟩
`@C`⟨add op⟩ ⟨dimen⟩
`@` ⟨add op⟩ ⟨dimen⟩

that change row spacing, column spacing, and both, respectively, as indicated by the ⟨add op⟩ and ⟨dimen⟩, where the ⟨dimen⟩ may be omitted and can be given as one of `R` and `C` to indicate the current value of the parameter in question. **Note:** there is *no default*.

In addition, XY-pic can be instructed to use a 'fixed grid' for the matrix with the switches

---

`@!R`
`@!C`
`@!`

---

that ensure that the row spacing, column spacing, and both, respectively, pretending that *all* entries have the size of the largest entry (without modifying the real size of the entries, of course, only the spacing – to get the entries to *really* have the same size use a `@*`... ⟨setup⟩ described in §3.5.4 below). The special variants

---

`@!0`
`@!=`⟨dimen⟩

---

pretend that entries have zero or ⟨dimen⟩ height and width for computing row and column spacing; as above inserting `R` or `C` just after the `!` makes this affect only the row *or* column spacing, *e.g.*, `@!R0` means that the row spacing only is between the centers of the rows.

Finally, the spacing of things that are typeset can be adjusted separately:

`@M`⟨add op⟩ ⟨dimen⟩
`@W`⟨add op⟩ ⟨dimen⟩
`@H`⟨add op⟩ ⟨dimen⟩
`@L`⟨add op⟩ ⟨dimen⟩

---

will adjust the entry margin, entry width, entry height, and label separation used (the latter is actually passed to the arrow feature).

```
\xylet@\xymatrix@addop@@=\relax

\xydef@\xymatrix@at{\def\xymatrix@addop@@{\xymatrix@R\xymatrix@C}%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xymatrix@at}%gobble spaces
 \else\ifx !\next \DN@!{\xyFN@\xymatrix@fix}%
 \else\addPLUS@\ifx \next \let\next@=\xymatrix@addop
 \else\addDASH@\ifx \next \let\next@=\xymatrix@addop
 \else\addEQ@\ifx \next \let\next@=\xymatrix@addop
 \else\ifx 1\next \DN@1{\def\everyentry@{\vphantom(}%)
  \let\objectmargin@=\p@ \xyFN@\xymatrix@setup}%
 \else\ifx R\next \def\xymatrix@addop@@{\xymatrix@R}\DN@ R{\xymatrix@addop}%
 \else\ifx C\next \def\xymatrix@addop@@{\xymatrix@C}\DN@ C{\xymatrix@addop}%
 \else\ifx M\next \def\xymatrix@addop@@{\xymatrix@M}\DN@ M{\xymatrix@addop}%
 \else\ifx L\next \def\xymatrix@addop@@{\xymatrix@L}\DN@ L{\xymatrix@addop}%
 \else\ifx H\next \def\xymatrix@addop@@{\xymatrix@H}\DN@ H{\xymatrix@addop}%
 \else\ifx W\next \def\xymatrix@addop@@{\xymatrix@W}\DN@ W{\xymatrix@addop}%
 \else\ifx *\next \DN@*{\xyFN@\xymatrix@mods}%
 \else \DN@{\afterDIRECTIONorEMPTY
    {\xymatrix@rotation \xyFN@\xymatrix@setup}{%
     \xyerror@{<addop> or <direction> or one of RCMLHW*! expected}{}%
     \xyFN@\xymatrix@setup}}%
 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}

\xydef@\xymatrix@fix{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xymatrix@fix}%gobble spaces
```

```
 \else\ifx R\next \let\mkHrow@=\Hrow@max \DN@ R{\xyFN@\xymatrix@fix@}%
 \else\ifx C\next \let\mkWcol@=\Wcol@max \DN@ C{\xyFN@\xymatrix@fix@}%
 \else \let\mkHrow@=\Hrow@HWmax \let\mkWcol@=\Wcol@HWmax
  \let\next@=\xymatrix@fix@
 \fi\fi\fi \next@}
```

```
\xydef@\xymatrix@fix@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xymatrix@fix@}%gobble spaces
 \else\ifx 0\next \def\mkHWmax@{\HWmax@set\z@}\DN@ 0{\xyFN@\xymatrix@setup}%
 \else\addEQ@\ifx\next \addEQ@\DN@{\afterassignment\xymatrix@fix@x\dimen@ii=}%
 \else \let\mkHWmax@=\HWmax@max \let\next@=\xymatrix@setup \fi\fi\fi \next@}
```

```
\xydef@\xymatrix@fix@x{%
 \edef\mkHWmax@{\noexpand\HWmax@set{\the\dimen@ii}}\xyFN@\xymatrix@setup}
```

```
\xydef@\xymatrix@addop{\afterADDOP{\xyFN@\xymatrix@addop@}}
\xydef@\xymatrix@addop@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xymatrix@addop@}%gobble spaces
 \else\ifx R\next \DN@ R{\xy@{R}{}\dimen@ii=\xymatrixrowsep@\xymatrix@addop@x}%
 \else\ifx C\next \DN@ C{\xy@{C}{}\dimen@ii=\xymatrixcolsep@\xymatrix@addop@x}%
 \else \DN@{\afterassignment\xymatrix@addop@x\dimen@ii=}\fi\fi\fi \next@}
```

```
\xydef@\xymatrix@addop@x{\expandafter\xy@\expandafter{\the\dimen@ii}{}%
 \xymatrix@addop@@ \xyFN@\xymatrix@setup}
```

```
\xydef@\xymatrix@R{\Addop@@\xymatrixrowsep@\dimen@ii}
\xydef@\xymatrix@C{\Addop@@\xymatrixcolsep@\dimen@ii}
```

```
\xydef@\xymatrix@M{\Addop@@\objectmargin@\dimen@ii}
\xydef@\xymatrix@L{\Addop@@\labelmargin@\dimen@ii}
```

```
\xydef@\xymatrix@H{\Addop@@\objectheight@\dimen@ii}
\xydef@\xymatrix@W{\Addop@@\objectwidth@\dimen@ii}
```

The spacing can also be changed for en entire TeX group by the declarations

---

`\xymatrixrowsep` ⟨add op⟩ `{`⟨dimen⟩`}`
`\xymatrixcolsep` ⟨add op⟩ `{`⟨dimen⟩`}`

The default spacing for both is `2pc`.

```
\xydef@\xymatrixrowsep@{2pc}
\xydef@\xymatrixcolsep@{2pc}
```

```
\xydef@\xymatrixrowsep{\afterADDOP{\Addop@@\xymatrixrowsep@}}
\xydef@\xymatrixcolsep{\afterADDOP{\Addop@@\xymatrixcolsep@}}
```

To Do: optimize processing of matrices with constant grid.

An entire matrix can be rotated by adding a rotation ⟨setup⟩ of the form

---

`@`⟨direction⟩

This will set the orientation of the rows to ⟨direction⟩ (the default corresponds to `r`, *i.e.*, rows are oriented left to right).

The code is the first procedure of the main matrix code above (1.306); factors are used by `\pre@emit`.

3.5.4 Entries

The appearance of a single entry can be modified by entering it as

---

`*` ⟨object⟩ ⟨pos⟩ ⟨decor⟩

This makes the particular entry ignore the entry modifiers and typeset as a kernel object with the same reference point as the (center of) the default object would have had.

Additional object ⟨modifier⟩s may be added to an otherwise ordinary entry by using the forms

---

`**[`⟨shape⟩`]` ⟨entry⟩
`**{`⟨modifier⟩*`}` ⟨entry⟩

The first sets the default ⟨shape⟩ for objects (*cf.* note 1.4j), the second a default size (change, *cf.* note 1.4g), and the last makes it possible to add any ⟨object⟩ modifier of §1.4, *e.g.*, for recentering entries after the default entry form which is equivalent to '`!C +<`$2 \times$ *objectmargin*`>`' (with the effect of centering the object and add the *objectmargin*) to all sides.

Exercise 3.9: Typeset the following diagram:

$$\begin{tikzcd} A \times B \arrow[r, "/A"] \arrow[d, "/B"'] & B \arrow[d, "\times A"] \\ A \arrow[r, "B\times"'] & B \times A \end{tikzcd}$$

It is also possible to use these `@`⟨setup⟩s (as usual between `\xymatrix` and the leading `{`):

`@*[`⟨shape⟩`]`
`@*` ⟨add op⟩ ⟨size⟩

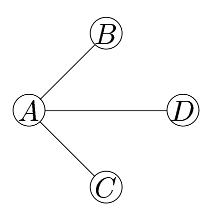
---

which are equivalent to changing all entries to behave as if they had started with the similar `**`-form.
To Do: Allow `**`⟨add op⟩⟨size⟩ ⟨entry⟩ for entries.

If the default set of entry modifiers should be changed then the following declaration must be issued before the `\xymatrix` command; this is the only way to actually switch the initial default centering and spacing off:

`\entrymodifiers={` ⟨modifier⟩* `}`

Exercise 3.10: How did the author typeset the following matrix?



Bug: The four constructions `@*[...]`, `**[...]`, `@*` ⟨add op⟩ ⟨size⟩, and, `**{...}`, *accumulate in reverse order*. Only entries starting with a single `*` completely override the modifiers ⟨setup⟩ with a `@*`-construction.

```
\xydef@\xymatrix@mods{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\xymatrix@mods}%gobble spaces
 \else\ifx [\next
  \DN@ [##1]{\xy@{[##1]}{}\DN@####1{\def\entrymodifiers@{[##1]####1}}%
    \expandafter\next@\expandafter{\entrymodifiers@}\xyFN@\xymatrix@setup}%]
 \else\addPLUS@\ifx \next \DN@##1{\DNii@{##1}\xyFN@\xymatrix@change}%
 \else\addDASH@\ifx \next \DN@##1{\DNii@{##1}\xyFN@\xymatrix@change}%
 \else\addEQ@\ifx \next \DN@##1{\DNii@{##1}\xymatrix@set}%
 \else \let\next@=\xymatrix@mods@error \fi\fi\fi\fi\fi \next@}

\xydef@\xymatrix@set{%
 \begingroup \plainxy@
 \afterVECTORorEMPTY\xymatrix@mods@x\xymatrix@mods@error}

\xydef@\xymatrix@change{%
 \addEQ@\ifx\next \DN@##1{\expandafter\DNii@\expandafter{\nextii@##1}%
   \expandafter\xy@\expandafter{\nextii@}{}\begingroup \plainxy@
   \afterVECTORorEMPTY\xymatrix@mods@x\xymatrix@mods@xx}%
 \else \DN@{\expandafter\xy@\expandafter{\nextii@}{}\begingroup \plainxy@
   \afterVECTORorEMPTY\xymatrix@mods@x\xymatrix@mods@xx}\fi
 \next@}

\xydef@\xymatrix@mods@x{\expandafter\xy@\expandafter{\nextii@}{}%
 \def\next@##1##2##3{\def\nextii@####1{##1##2####1##3}}%
 \addGT@{\addLT@{\expandafter\next@\expandafter{\nextii@}}}%
 \DN@##1{\gdef\tmp@@@{{##1}}}%
 \edef\nextiii@{{\the\X@c,\the\Y@c}}%
 \expandafter\expandafter\expandafter\next@
  \expandafter\expandafter\expandafter{\expandafter\nextii@\nextiii@}%
 \endgroup
 \DN@##1{\xy@{##1}{}\DN@####1{\def\entrymodifiers@{##1####1}}%
   \expandafter\next@\expandafter{\entrymodifiers@}\xyFN@\xymatrix@setup}%
 \expandafter\next@\tmp@@@}

\xydef@\xymatrix@mods@xx{\expandafter\xy@\expandafter{\nextii@}{}%
 \expandafter\gdef\expandafter\tmp@@@\expandafter{\expandafter{\nextii@}}%
 \endgroup
 \DN@##1{\xy@{##1}{}\DN@####1{\def\entrymodifiers@{##1####1}}%
   \expandafter\next@\expandafter{\entrymodifiers@}\xyFN@\xymatrix@setup}%
 \expandafter\next@\tmp@@@}

\xydef@\xymatrix@mods@error{\endgroup
 \xyerror@{Impossible @*<addop><size> <setup>.}{}\xyFN@\xymatrix@setup}

\xywarnifdefined\entrymodifiers
\expandafter\def\addEQ@\entrymodifiers#1{\def\entrymodifiers@{#1}\ignorespaces}

\xydef@\entrymodifiers@{\entrybox}

\xydef@\entrybox#1{\hbox{\setbox\z@=\objectbox{#1}%
  \L@p=\wdz@ \D@p=\ht\z@ \advance\D@p\dp\z@
  \dimen@=\objectwidth@ \advance\dimen@-\L@p
  \ifdim\dimen@>\z@ \advance\L@p\dimen@ \R@p=.5\dimen@ \else \R@p=\z@ \fi
  \dimen@=\objectheight@ \advance\dimen@-\D@p
  \ifdim\dimen@>\z@ \advance\D@p\dimen@ \fi
```

```
\dimen@=\objectmargin@ \advance\L@p2\dimen@ \advance\D@p2\dimen@
\U@p=-.5\ht\z@ \advance\U@p.5\dp\z@ \advance\R@p\objectmargin@
\setboxz@h{\kern\R@p \raise\U@p\boxz@}%
\wdz@=\L@p \ht\z@=.5\D@p \dp\z@=.5\D@p \boxz@}}
```

Finally, the following is specially suited for 'one-line' matrices included in text:

`@1`

---

will insert a 'math strut' (`\vphantom(`) into each entry to make its centering appropriate for text.

This is just an abbreviation using the following declaration which will setup ⟨decor⟩ that should be inserted before everything else in each entry. Initially it is empty but

---

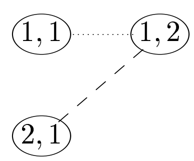
`\everyentry={` ⟨decor⟩ `}`

---

will insert ⟨decor⟩ first in each entry; inside the counter registers `\Row` and `\Col` are set to the current entry's row and column, respectively. For example,

```
\everyentry={{\the\Row,\the\Col}}
\xymatrix @*[F]@*[o] {
 {} \POS[];[r]**\dir{..} & \\
 {} \POS[];[ur]**\dir{--}
}
```

will typeset

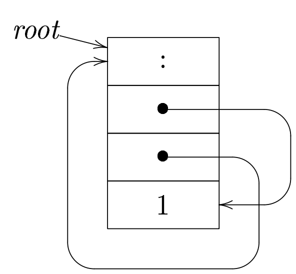



```
\xywarnifdefined\everyentry
\expandafter\def\addEQ@\everyentry#1{\def\everyentry@{#1}\ignorespaces}
```

```
\xylet@\everyentry@=\empty
```

Note: When using compilation, changes to `\everyentry` and `\entrymodifiers` will *not* result in recompilation even when the constructed matrix changes – you may have to remove the `.xyc` file manually.

Exercise 3.11: How did the author typeset the following diagram?



Hints: The arrow feature was used to make the bending arrows and the frame extension for the frames around each cell.

End & log

```
\xyendinput

% $Log: xymatrix.doc,v $
% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.14  1995/07/06  02:56:02  kris
% Buglets...
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.11  1994/07/05  10:37:32  kris
% Third 3beta release [bug fixes].
% Experimental graph feature included (for ECCT-94 presentation).
%
% Revision 2.9  1994/06/09  15:02:49  kris
% Release 3beta.
%
% Revision 2.8  1994/04/08  04:30:00  kris
% Second (bug fix) 3alpha release.
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% NEW for version 2.7 extracted from diagram code in xypic.doc 2.6.1.1.
```

3.6 Graph feature

Vers. 3.6 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

This option implements 'XY-graph', a special *combinatoric drawing language* suitable for diagrams like flow charts, directed graphs, and various forms of trees. The base of the language is reminiscent of the PIC [5] language because it uses a notion of the 'current location' and is based on 'moves'. But the central construction is a 'map' combinator that is borrowed from functional programming.

Header:

```
%% $Id: xygraph.doc,v 3.6 1998/03/06 01:28:05 krisrose Exp krisrose $
%%
%% Xy-pic ``Graph Combinator feature'' option.
%% Copyright (c) 1994-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{graph}{Graph feature}{\stripRCS$Revision: 3.6 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

XY-graph makes use of facilities of the 'arrow' feature option of §3.3, which is therefore required.

```
\xyrequire{arrow}\xycatcodes
```

Figure 3.5 summarises the syntax of a ⟨graph⟩ with notes below. A ⟨graph⟩ can appear either in an XY-picture (as ⟨decor⟩) or "stand-alone".[3] **Note:** If you use `\xygraph{...}` inside constructions where `&` is significant (like plain TEX's `\halign` or LATEX's `array` environment) then make sure to add an extra level of braces around it.

Parsing: The parser does almost everything that this feature does because it is all implemented directly on top of other constructions, notably the kernel stack and the ⟨arrow⟩ construction.

The graph state always has the current value of `?` both as the stacktop and in the special ⟨id⟩ `"?NODE"`.

```
\xydef@\xygraph#1{\relax\graph#1\endgraph}
\xydef@\graph{\if\inxy@ \DN@{\xy@@{\nter@{}}}%
```

[3] In fact LATEX users can use a `graph` environment.

```
 \else \DN@{\xy \xy@\xy{\nter@\endxy}}\fi \next@
 \xy@\xygraph{\senter@ \idfromc@{?NODE}\spushc@
  \expandafter\let\expandafter\next@\csname Q@graphbase\endcsname
  \ifx\next@\relax \setbase@\z@\z@{3pc}\z@
  \else \expandafter\next@ \fi}%
 \GRAPH[]}

\xydef@\endgraph{\ignorespaces}

\xydef@\GRAPH{\xyFN@\GRAPH@}

\xydef@\GRAPH@{%
%
%\W@{GRAPH: next=\meaning\next}%
%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\GRAPH@}%
 \else\ifx :\next \DN@:{\xy@:{}\expandafter\GRAPHar@\GRAPHeveryarrow@@}%
 \else\addDASH@\ifx\next
  \addDASH@\DN@{\xy@-{}\expandafter\GRAPHar@\GRAPHeveryline@@}%
 \else\ifx (\next
  \DN@({\xy@({\spushid@{?NODE}\idfromc@{?NODE}}\xyFN@\GRAPH@}%
 \else\ifx ,\next
  \DN@,{\xy@,{\cfromid@{?NODE}}\xyFN@\GRAPH@}%
 \else\ifx )\next
  \DN@){\xy@){\cfromid@{?NODE}%
    \tests@\z@{\idfroms@{?NODE}\z@ \spop@}{\xyerror@{) without (}{}}}%
   \xyFN@\GRAPH@}%
 % HACK: ONLY POP WHEN NONEMPTY...WHY EVER NONEMPTY?
 \else\ifx \endgraph\next \DN@{\xy@@{\tests@\z@\spop@\sleave@\leave@}}%
 \else\ifx \end\next \DN@{\xy@@{\tests@\z@\spop@\sleave@\leave@}}%
 \else\ifx ~\next
  \DN@~{\xy@@{\enter@\cfromthec@}\afterNODE{\leave@ \xyFN@\GRAPH@}!}%
 \else \DN@{\afterNODE{\xyFN@\GRAPH@}}%
 \fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}

\xylet@\GRAPHsaved@PATHafterPOS=\PATHafterPOS
\xylet@\GRAPHsaved@afterar@@=\afterar@@

\xydef@\GRAPHar@{%
 \let\PATHafterPOS=\afterNODE \let\afterar@@=\GRAPHafterar@ \ar}

\xydef@\GRAPHafterar@{%
 \let\PATHafterPOS=\GRAPHsaved@PATHafterPOS
 \let\afterar@@=\GRAPHsaved@afterar@@
 \xy@@{\cfromid@{}}\xyFN@\GRAPH@}

\xylet@\afterNODE@@=\empty

\xydef@\afterNODE#1{\xy@@{\enter@\pfromthep@}%
 \DN@##1{\def\afterNODE@@{\xy@@\leave@ \def\afterNODE@@{##1}#1}}%
 \expandafter\next@\expandafter{\afterNODE@@}%
 \xyFN@\NODE@}

\xylet@\theMOVE@@=\relax

\xydef@\NODE@{%
```

```
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\NODE@}%
 \else\ifx [\next \DN@[##1]{\xy@{[##1]}{}%
   \def\theMOVE@@{[##1]}\xyFN@\MOVE@##1]}%
 \else\ifx "\next \DN@ "##1"{%
   \xy@{"##1"}{\cfromid@{##1}}\def\theMOVE@@{"##1"}\xyFN@\NODE@i}%
 \else\ifx ?\next \DN@ ?{%
   \xy@?{\cfromid@{?NODE}}\def\theMOVE@@{?}\xyFN@\NODE@i}%
 \else\ifx !\next \def\theMOVE@@{[]}\let\next@=\NODE@i
 \else \testAND@\iffalse
  \DN@##1{\xy@{}{\pfromc@ \advance\X@c\X@xbase \advance\Y@c\Y@xbase
    \czeroEdge@}\jumpMOVE@ \xyFN@\NODE@i}%
 \else\ifx \\\next
  \DN@\\{\xy@\\{\pfromc@ \advance\X@c\X@xbase \advance\Y@c\Y@xbase
    \R@c=\X@ybase \U@c=\Y@ybase \intersect@
    \advance\X@c-\X@ybase \advance\Y@c-\Y@ybase
    \czeroEdge@}\def\theMOVE@@{\\}\xyFN@\NODE@i}%
 \else \xyerror@{Illegal <node>: \string`\meaning\next\string' unexpected}{}%
 \fi\fi\fi\fi\fi\fi\fi \next@}

\xydef@\jumpMOVE@@{&}
\xydef@\jumpMOVE@{\let\theMOVE@@=\jumpMOVE@@}

\xydef@\MOVE@{\xy@@{\count@=\z@ \count@@=\z@}%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\MOVE@}%
 \else\ifcat A\noexpand\next \let\next@=\MOVE@letter
 \else \let\next@=\MOVE@x \fi\fi \next@}

\xydef@\MOVE@letter{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\MOVE@letter}%
 \else\ifx u\next \DN@ u{\xy@@{\advance\count@\@ne}\xyFN@\MOVE@letter}%
 \else\ifx d\next \DN@ d{\xy@@{\advance\count@\m@ne}\xyFN@\MOVE@letter}%
 \else\ifx l\next \DN@ l{\xy@@{\advance\count@@\m@ne}\xyFN@\MOVE@letter}%
 \else\ifx r\next \DN@ r{\xy@@{\advance\count@@\@ne}\xyFN@\MOVE@letter}%
 \else \let\next@=\MOVE@x \fi\fi\fi\fi\fi \next@}

\xydef@\MOVE@x{%[[
 \ifx ]\next \DN@ ]{\MOVE@hop \xyFN@\NODE@i}%
 \else\ifx (\next \DN@{\MOVE@hop \xyFN@\MOVE@place}%)
 \else\ifx !\next \DN@{\MOVE@hop \xyFN@\MOVE@place}%
 \else\ifx /\next \DN@{\MOVE@hop \xyFN@\MOVE@place}%
 \else\addLT@\ifx\next \DN@{\MOVE@hop \xyFN@\MOVE@place}%
 \else\addGT@\ifx\next \DN@{\MOVE@hop \xyFN@\MOVE@place}%
 \else\addDASH@\ifx\next \addDASH@\DN@{\xyerror@{- not implemented yet}{}}%
 \else \xyerror@{illegal <move>: \codeof\theMOVE@@}{}\let\next@=\NODE@i
 \fi\fi\fi\fi\fi\fi\fi \next@}

\xydef@\MOVE@hop{\xy@@{\pfromc@ \czeroEdge@
  \edef\next@{\the\count@@,\the\count@ @}\expandafter\vfromcartesian@@\next@
  \advance\X@c-\X@origin \advance\Y@c-\Y@origin
  \advance\X@c\X@p \advance\Y@c\Y@p}}

\xydef@\MOVE@place{\xy@@\no@@ \afterPLACE{\xyFN@\MOVE@}}

\xydef@\NODE@i{\let\next@=\NODE@x
```

```
%
%\W@{NODE@i: next=\meaning\next}%
%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\NODE@i}%
 \else\addEQ@\ifx \next
  \addEQ@\DN@"##1"{\xy@{="##1"}{\idfromc@{##1}}\xyFN@\NODE@i}%
 \else\ifx !\next \DN@!{\xyFN@\NODE@escape}%
 \else\ifx :\next
 \else\addDASH@\ifx\next
 \else\ifx (\next
 \else\ifx ,\next
 \else\ifx )\next
 \else\ifx ~\next
 \else\ifx [\next
 \else\ifx "\next
 \else\ifx ?\next
%
 \else\ifx ^\next
 \else\ifx _\next
 \else\ifx |\next
 \else\addLQ@\ifx\next
 \else\addRQ@\ifx\next
%
 \else \testAND@ \iffalse
 \else\ifx \\\next
%
 \else\ifx \endgraph\next
 \else\ifx \end\next
%
 \else \let\next@=\NODE@it
 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}

\xydef@\NODE@x{\xy@@{\idfromc@{}}\afterNODE@@}

\xydef@\NODE@it{\let\PATHlabelit@@=\NODEit@ \PATHit@}

{\xyuncatcodes \gdef\next{+}}
\xylet@\GRAPHnodemodifiers@@=\next

\xydef@\NODEit@#1{%
 \DN@{#1}\ifx\next@\empty
  \DN@{\expandafter\NODEit@i\expandafter{\GRAPHnodemodifiers@@}}%
 \else \DN@{\NODEit@i{#1}}\fi \next@}

\def\NODEit@i#1#2{\xy@{*#1{#2}}{\drop@{#1}{#2}}%
 \DN@{\addEQ@\ifx\next\else \xy@@{\idfromc@{#2}}\fi \NODE@i}%
 \xyFN@\next@}
```

Notes

3.6a. A *move* is to establish a new *current node*.

3.6b. To *draw* something is simply to draw a line or the specified ⟨arrow⟩ from the current node to the specified target node. The target then becomes the current node. All the features of arrows as described in §3.3 can be used, in particular arrows can be labelled and segmented, but with the change that ⟨path-pos⟩ means ⟨node⟩ as explained in note §3.3e.

3.6c. To *map over a list* is simply to save the current node and then interpret the ⟨list⟩ with the following convention:

- Start each element of the list with the current node as saved and p as the previous list element, and
- let the `?` ⟨node⟩ refer to the saved current node explicitly.

3.6d. The `&` and `\\` special moves are included to make it simple to enter 'matrix-like' things as graphs – note that they will not be automatically aligned, however, for that you should use the `!M` escape.

`&` is the same as `[r]` and `\\` is the same as `[r]!{y+(0,-1)-(0,0)}` which uses a kernel escape to moves to the first column in the next row (where the first column is on the y-axis of the current coordinate system).

Note: If you use the form `*{...}` for nodes then you don't have to change them if you decide to use an XY-matrix.

3.6e. Typeset ⟨it⟩ and make it the current node. Also saves ⟨it⟩ for later reference using `"`⟨id⟩`"`: if ⟨it⟩ is a simple letter, or digit, then just as `"`⟨it⟩`"`; if ⟨it⟩ is of the form `{`*text*`}` or `*`...`{`*text*`}` then as `"`*text*`"`.

With the `=` addition it is possible to save explicitly in case several nodes have the same text or a node has a text that it is impractical to use for reference. In fact using the form ⟨it⟩`="`⟨id⟩`"` will *only* save the node as `"`⟨id⟩`"` and *not* as `"`⟨it⟩`"`! As a special convenience `""` (thus the empty ⟨id⟩) always refers to the last completed node, so adding `=""` after a node merely means it should *not* be saved under its proper name.

Exercise 3.12: How did the author typeset this?

$$\begin{tikzcd} A \arrow[r, bend left] & A \arrow[r, bend left] & A \arrow[ll, bend left] \end{tikzcd}$$

3.6f. Moving by a series of *hops* is simply moving in a grid as the sequence of `dulr` (for down/up/left/right) indicates. The grid is a standard cartesian coordinate system with 3pc unit unless the current base is redefined using `[]!{...}` with an appropriate ⟨pos⟩ition containing `:` and `::` as described in note 1.3d.

To Do: Describe the use of ⟨move⟩s with ⟨place⟩s in detail ... in particular (1) 'until perpendicular to ... ' and (2) 'until intercepts with ... ' can be coded...

3.6g. This 'escapes' into the XY-pic kernel language and interprets the ⟨pos⟩ ⟨decor⟩. The current node is then set to the resulting c object and the grid from the resulting *base*.

The effect of the ⟨pos⟩ ⟨decor⟩ can be completely hidden from XY-graph by entering it as `{\save` ⟨pos⟩ ⟨decor⟩ `\restore}`.

```
\xydef@\NODE@escape{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\NODE@escape}%
 \else\ifcat A\noexpand\next \DN@##1{\rungraphescape@{##1}}%
 \else\ifcat !\noexpand\next \DN@##1{\rungraphescape@{##1}}%
```

```
 \else\ifx \bgroup\next \DN@##1{\xy@{!{##1}}{}\POS##1\relax \xyFN@\NODE@i}%
 \else\ifx ~\next \let\next@=\graphsetup@
 \else \DN@{\xyerror@{Unknown graph <escape>}{}\xyFN@\NODE@i}%
 \fi\fi\fi\fi\fi \next@}
```

3.6h. It is possible to insert a ⟨matrix⟩ in a graph provided the 'matrix' option described in §3.5 has been loaded: it overwrites the node with the result of `\xymatrix`⟨matrix⟩. Afterwards the graph grid is set as the top left 'square' of the matrix, *i.e.*, with `[d]` and `[r]` adjusted as they work in the top left entry.

Bug: `[dr]` immediately after the matrix will work as expected, *e.g.*, make the center of `"2,2"` the current node, but others might not, *e.g.*, `[rr]` will not necessarily place the current node on top of `"1,3"`.

```
\xydefcsname@{graph !M}#1#{\NODE@matrix{#1}}

\xydef@\NODE@matrix#1#2{\xyerror@{matrix feature not loaded}{}\xyFN@\GRAPH@}

\xydef@\NODE@matrix@#1#2{\xy@{!M#1{#2}}{}%
 \def\everyentry@{\GRAPHeveryentry}%
 \xymatrix#1{#2}%
 \let\everyentry@=\empty \insertmatrixingraph@ \xyFN@\NODE@i}

\xydef@\GRAPHeveryentry{\def\:{\ar}\def\="##1"{\xy@@{\idfromc@{##1}}}}

\xywithoption{matrix}{\let\NODE@matrix=\NODE@matrix@}

\xydef@\insertmatrixingraph@{%
 \xy@@{\enter@{\cfromthec@ \pfromthep@}%
 \expandafter\let\expandafter\next@\csname Q@1,1\endcsname
 \ifx\next@\relax\else
  \next@ \swap@
  \expandafter\let\expandafter\next@\csname Q@1,2\endcsname
  \ifx\next@\relax\else \next@ \setbase@\X@p\Y@p\X@c\Y@c
   \expandafter\let\expandafter\next@\csname Q@2,1\endcsname
   \ifx\next@\relax\else
    \next@ \setbase@@{-\X@c}{-\Y@c}\fi\fi\fi
 \leave@}}
```

3.6i. It is possible to insert a ⟨polygon⟩ or an ⟨ellipse⟩ in a graph provided the `poly` option described in §3.7 or the `arc` option described in §3.9 has been loaded, respectively: it will have c as the current node, p as the previous one, and the the current base has the ⟨hop⟩s `[r]` and `[u]` as base vectors.

Note: lattices, knots, *etc.*, can also be used but no special syntax is useful since the `!{...}` syntax is adequate.

```
\xydefcsname@{graph !P}#1#{\NODE@poly{#1}}
\xydef@\NODE@poly#1#2{\xyerror@{poly(gon) feature not loaded}{}\xyFN@\GRAPH@}
\xydef@\NODE@poly@#1#2{\xy@{!P#1{#2}}{}\xypoly@#1{#2}\xyFN@\NODE@i}
\xywithoption{poly}{\let\NODE@poly=\NODE@poly@}

\xydefcsname@{graph !E}#1#{\NODE@arc{#1}!}
\xydef@\NODE@arc#1#2{\xyerror@{(ellipse+)arc feature not loaded}{}%
 \xyFN@\GRAPH@}
\xydef@\NODE@arc@#1#2{\xy@{!E#1{#2}}{}\ellipse#1{#2}\xyFN@\NODE@i}
```

```
\xywithoption{arc}{\let\NODE@arc=\NODE@arc@}
```

3.6j. This allows setting of some parameters of the graph: `!~`⟨setup⟩ should be one of the following:

`!~:{` ⟨arrow⟩ `}`	include with every `:` arrow
`!~-{` ⟨arrow⟩ `}`	include with every `-` line
`!~*{` ⟨modifiers⟩ `}`	include with every non-`*` node
`!~`⟨letter⟩`{` ⟨graph⟩ `}`	define new graph escape `!`⟨letter⟩

These are destructive: the previous value is lost; the default is established by the sequence `!~:{}` `!~-{@{-}}` `!~*{+}` making `:` create simple arrows, `-` plain lines, and formatting default nodes in math mode with the default objectmargin.

```
\xylet@\GRAPHeveryarrow@@=\empty
{\xyuncatcodes \gdef\next{@{-}}}
\xylet@\GRAPHeveryline@@=\next
\xydef@\graphsetup@~{\xyFN@\graphsetup@i}
\xydef@\graphsetup@i{%
 \ifx :\next \DN@:##1{\def\GRAPHeveryarrow@@{##1}\xyFN@\NODE@i}%
 \else\addDASH@\ifx\next
  \addDASH@\DN@##1{\def\GRAPHeveryline@@{##1}\xyFN@\NODE@i}%
 \else\ifx *\next \DN@*##1{\def\GRAPHnodemodifiers@@{##1}\xyFN@\NODE@i}%
 \else\ifcat A\next \DN@##1##2{\newgraphescape{##1}{##2}\xyFN@\NODE@i}%
 \else \DN@{\xyerror@{illegal graph setup \codeof\next@}{}\xyFN@\NODE@i}%
 \fi\fi\fi\fi \next@}
```

The last possibility is also available as a command

`\newgraphescape{`⟨letter⟩`}{`⟨graph⟩`}`

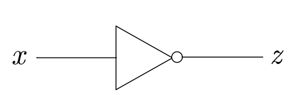
that makes the specified escape generate the ⟨graph⟩ as a macro; with it it is possible to pass arguments to the ⟨graph⟩ using the standard TeX `\def` method: The declaration code

```
\newgraphescape{i}#1#2{
 []!{+0="o#2"*=<10pt>{};p!#1**{},"o#2"
  -/4pt/*!E\cir<2pt>{}
  +0;p-/:a(-30)24pt/**\dir{-}="X2"
  ;p-/:a(-60)24pt/="X1"**\dir{-}
  ;?(.5),="i#2",
  p-/:a(-60)24pt/**\dir{-},
  "o#2"."i#2"."X1"."X2"}}
```

is (rather complicated kernel code) that makes the node escape `!i`dn typeset an 'inverter' oriented with the d corner as the output with input named `"i`n`"` and output named `"o`n`"` such that the graph

```
\xygraph{ []!iR1 ("i1"[l]x - "i1") - [r]z }
```

will typeset

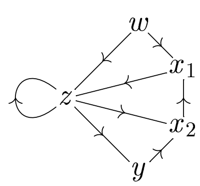


The `\newgraphescape`⟨letter⟩ declaration defines a macro with name `\graph !`⟨letter⟩. It is slightly complicated by the fact that we accept LaTeX-style `[n]` arguments (when using LaTeX, that is ☺). Thus an internal macro named `\G!`⟨letter⟩ is defined using the appropriate command definition primitve; this is in turn invoked by `\graph !`⟨letter⟩ after activating the ⟨graph⟩ parser.

```
\xydef@\newgraphescape#1{%
 \expandafter\let\expandafter\next\csname graph !#1\endcsname
 \ifx\next\relax\else \expandafter\let\csname G!#1\endcsname=\undefined
  \xywarning@{Redefining graph <escape> !#1}\fi
 \expandafter\gdef\csname graph !#1\endcsname{%
  \expandafter\let\expandafter\next\csname G!#1\endcsname
  \expandafter\xyFN@\expandafter\NODE@\next}%
 \DN@{\newgraphescape@{#1}}\xyFN@\next@}

\xydef@\newgraphescape@#1{%
 \ifx [\next %]
  \ifx\newcommand\undefined \xyerror@{[n] only with LaTeX}%
  \else
   \def\nextiv@{\expandafter\newcommand\expandafter{\csname G!#1\endcsname}}\fi
 \else \def\nextiv@{\expandafter\gdef\csname G!#1\endcsname}\fi \nextiv@}

\xydef@\rungraphescape@#1{%
 \expandafter\let\expandafter\next\csname graph !#1\endcsname
 \ifx\next\relax \xyerror@{Undefined graph <escape> !#1}\fi \next}
```

The final exercise illustrates much of the above.

Exercise 3.13: Typeset



End & log

```
\xyendinput

% $Log: xygraph.doc,v $
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.2  1995/09/19  18:22:27  kris
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
```

```
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.11  1994/07/05  10:37:32  kris
% Third 3beta release [bug fixes].
% Experimental graph feature included (for ECCT-94 presentation).
%
% New for v3...
```

3.7 Polygon feature

Vers. 3.4 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This feature provides a means for specifying the locations of vertices for regular polygons, with any number (≥ 3) of sides. Polygons can be easily drawn and/or the vertex positions used to constuct complex graphics within an XY-picture. Many non-regular polygons can be specified by setting a non-square basis.

Header:

```
%% $Id: xypoly.doc,v 3.4 1997/05/18 01:13:24 ross Exp krisrose $
%%
%% Xy-pic ``Polygon'' feature.
%% Copyright (c) 1994-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this macro package; if not, write to the Free Software Foundation,
%% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{poly}{Polygon feature}{\stripRCS$Revision: 3.4 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

Some of the effects use `\ar` so make sure the 'arrow' feature is loaded.

```
\xyrequire{arrow}\xycatcodes
```

```
\message{polygons,}
```

A polygon is most easily specified using . . .

`\xypolygon`⟨number⟩`{}` with ⟨number⟩ sides;
`\xypolygon`⟨number⟩`{`⟨tok⟩`}` ⟨tok⟩ at vertices;
`\xypolygon`⟨number⟩`{`⟨object⟩`}`
with a general ⟨object⟩ at each vertex;

Here ⟨number⟩ is a sequence of digits, giving the number of sides. If used within an `\xy`. . .`\endxy` environment then the polygon will be centred on c, the current ⟨pos⟩. However an `\xypolygon` can be used outside such an environment, as "stand-alone" polygon; the whole picture must be specified within the `\xypolygon` command.

In either case the shape is obtained by spacing vertices equally around the "unit circle" with respect to the current basis. If this basis is non-square then the vertices will lie on an ellipse. Normally the polygon, with at most 12 vertices, is oriented so as to have a flat base when specified using a standard square basis. With more than 12 vertices the orientation is such that the line from the centre to the first vertex is horizontal, pointing to the right. Any other desired orientation can be obtained, with any number of vertices, by using the `~={`. . .`}` as described below.

The general form for `\xypolygon` is . . .

`\xypolygon`⟨number⟩`"`⟨prefix⟩`"{`⟨switches⟩. . .`}`

---

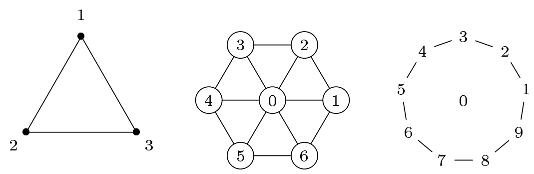
where the `"`⟨prefix⟩`"` and ⟨switches⟩ are optional. Their uses will be described shortly.

A `\xypolygon` establishes positions for the vertices of a polygon. At the same time various things may be typeset, according to the specified ⟨switches⟩. An ⟨object⟩ may be dropped at each vertex, "spokes" drawn to the centre and successive vertices may be connected as the polygon's "sides". Labels and breaks can be specified along the spokes and sides.

Each vertex is automatically named: `"1"`, `"2"`, . . . , `"`⟨number⟩`"` with `"0"` as centre. When a ⟨prefix⟩ has been given, names `"`⟨prefix⟩`0"`, . . . , `"`⟨prefix⟩⟨number⟩`"` are used instead. While the polygon is being constructed the macro `\xypolynum` expands to the number of sides, while `\xypolynode` expands to the number of each vertex, spoke and side at the time it is processed. This occurs in the following order: *vertex* 1, *spoke* 1, *vertex* 2, *spoke* 2, *side* 1, *vertex* 3, *spoke* 3, *side* 2, . . . , *vertex* n, *spoke* n, *side* $n-1$, *side* n where the final side joins the last vertex to the first.

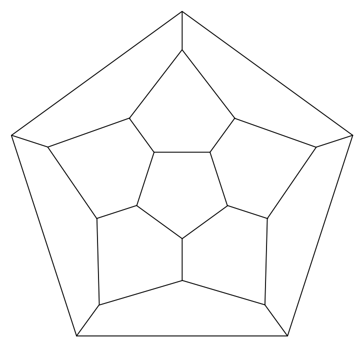
The macro `\xypolyname` holds the name of the polygon, which is ⟨prefix⟩ if supplied. In this case the value of `\xypolynum` is also stored as `\`⟨prefix⟩`NUMSIDES`, accessible outside the polygon.

As stated above, a polygon with up to 12 vertices is oriented so as to have a flat base, when drawn using a standard square basis. Its vertices are numbered in anti-clockwise order, commencing with the one at horizontal-right of centre, or the smallest angle above this (see example below). With more than 12 vertices then vertex `"1"` is located on the horizontal, extending to the right from centre (assuming a standard square basis). By providing a switch of the form `~={`⟨angle⟩`}` then the vertex `"1"` will be located on the unit circle at ⟨angle⟩° anti-clockwise from "horizontal" — more correctly, from the X-direction in the basis to be used when setting the polygon, which may be established using a `~:{...}` switch.



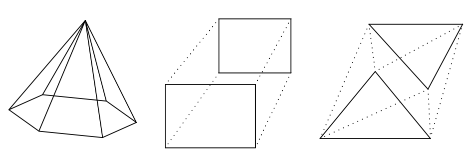
Exercise 3.14: Give code to typeset these.

One important use of ⟨prefix⟩ is to allow the vertices of more than one polygon to be accessed subsequently within the same picture. Here are some examples of this, incorporating the `~:{...}` switch to perform simple rescalings. Firstly the edges of a dodecahedron as a planar graph:



```
\xy /l1.5pc/:,{\xypolygon5"A"{}},
{\xypolygon5"B"{~:{(1.875,0):}~>{}}},
{\xypolygon5"C"{~:{(-2.95,0):}~>{}}},
{\xypolygon5"D"{~:{(-3.75,0):}}},
{"A1"\PATH~={**@{-}}'"B1"'"C4"'"B2"},
{"A2"\PATH~={**@{-}}'"B2"'"C5"'"B3"},
{"A3"\PATH~={**@{-}}'"B3"'"C1"'"B4"},
{"A4"\PATH~={**@{-}}'"B4"'"C2"'"B5"},
{"A5"\PATH~={**@{-}}'"B5"'"C3"'"B1"},
"C1";"D1"**@{-},"C2";"D2"**@{-},
"C3";"D3"**@{-},"C4";"D4"**@{-},
"C5";"D5"**@{-} \endxy
```

Next a hexagonal pyramid, a rectangular box and an octahedral crystal specified as a triangular anti-prism. Notice how the `~:{...}` switch is used to create non-square bases, allowing the illusion of 3D-perspective in the resulting diagrams:



```
\xy/r2pc/: ="A", +(.2,1.5)="B","A",
{\xypolygon6{~:{(1,-.1):(0,.33)::}
 ~<>{;"B"**@{-}}}}\endxy
\quad \xy /r2pc/:
{\xypolygon4"A"{~:{(0,.7)::}}},+(.7,1.1),
```

```
{\xypolygon4"B"{~:{(.8,0):(0,.75)::}}},
"A1";"B1"**@{.},"A2";"B2"**@{.},
"A3";"B3"**@{.},"A4";"B4"**@{.}
\endxy\quad \xy /r2pc/:
{\xypolygon3"A"{~:{(0,.7)::}}},+(.7,1.1),
{\xypolygon3"B"{~:{(-.85,0):(-.15,.8)::}}}
,"A1"\PATH~={**@{.}}'"B2"'"A3"'"B1"
'"A2"'"B3"'"A1" \endxy
```

`\xypolynode` simply reads the value of a counter called `\xypolynode@`. It is set initially at -1, to indicate no polygon yet. This is used to establish the default naming in case polygons are nested.

```
\xynew@{count}\xypolynode@\xypolynode@=\m@ne
\xydef@\xypolynode{\the\xypolynode@}
```

```
\xydef@\xypolyNUMSIDES@{}
\xydef@\xypolynum{\xypolyNUMSIDES@}
```

To allow recursion the values of `\xypolyNUMSIDES@` and `\xypolynode@` are saved upon entering an `\xypolygon`, to be restored at the end.

```
\xydef@\xypolygon{\xypoly@}%
```

```
\xydef@\xypoly@{%
 \if\inxy@
  \if\inxypoly@
   \edef\next@{\noexpand\xy@@{\noexpand\nter@{\xypolynode@=\the\xypolynode@
    \noexpand\def\noexpand\xypolyNUMSIDES@{\xypolyNUMSIDES@}}}}%
  \else
   \DN@{\xy@@{\def\inxypoly@{TT}%
    \nter@{\def\inxypoly@{TF}\xypolynode@=\m@ne}}}%
  \fi
 \else
  \if\inxypoly@
   \edef\next@{\noexpand\xy
    \noexpand\xy@@{\noexpand\nter@{\xypolyside=\the\xypolyside
     \noexpand\def\noexpand\xypolyNUMSIDES@{\xypolyNUMSIDES@}\endxy}}}%
  \else
   \DN@{\xy \xy@@{\def\inxypoly@{TT}%
    \nter@{\def\inxypoly@{TF}\xypolynode@=\m@ne \endxy}}}%
  \fi
 \fi
 \next@ \count@=\z@ \xyFN@\xypoly@n}
```

```
\xydef@\inxypoly@{TF}%
\xydef@\xypoly@x{\xy@@{\leave@\leave@}}
```

Read the number of vertices by parsing digits.

```
\xydef@\xypoly@n{%
 \DN@##1{\multiply\count@10\relax \advance\count@##1\relax \xyFN@\xypoly@n}%
 \ifx \space@\next
  \ifnum\count@>\z@ \expandafter\DN@\space@{\xyFN@\xypoly@i}%
  \else \expandafter\DN@\space@{\xyFN@\xypoly@n}\fi
 \else\ifx 0\next \else\ifx 1\next \else\ifx 2\next \else\ifx 3\next
```

```
 \else\ifx 4\next \else\ifx 5\next \else\ifx 6\next \else\ifx 7\next
 \else\ifx 8\next \else\ifx 9\next
 \else \let\next@=\xypoly@i \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi \next@}
```

```
\xydef@\xypoly@i{\edef\xypolyNUMSIDES@{\the\count@}\xyFN@\xygetpolyprefix@}
```

Next we check whether a prefix has been supplied for creating the vertex names. If so it is stored as `\xypolyPREFIX@`, which otherwise expands to empty. The number of sides is also stored using the prefix.

```
\xydef@\xypolyPREFIX@{}%
\xydef@\xypolyPREFIX@@{}%
\xydef@\xypolyname{\xypolyPREFIX@}%
```

```
\xydef@\xygetpolyprefix@{%
 \ifx\next"\DN@"##1"{\def\xypolyPREFIX@@{##1}\expandafter
  \edef\csname\xypolyPREFIX@@ NUMSIDES\endcsname{\xypolyNUMSIDES@}%
  \xypolyprefix@@ }%
 \else
  \ifnum\xypolynode@<\z@\def\xypolyPREFIX@@{}%
  \else
   \edef\xypolyPREFIX@@{\xypolyPREFIX@\the\xypolynode@,}%
  \fi \DN@{\xypolyprefix@@}%
 \fi \next@ }
```

```
\xydef@\xypolyprefix@@{%
 \expandafter\toks@\expandafter{\addEQ@\POS"}%
 \expandafter\addtotoks@\expandafter{\xypolyPREFIX@@0"\relax}%
 \the\toks@
 \ifnum\xypolynode@<\z@
  \expandafter\def\expandafter\xypolyPREFIX@\expandafter{\xypolyPREFIX@@}%
  \xy@@{\enter@{\xypolynode@=\the\xypolynode@
   \noexpand\def\noexpand\xypolyPREFIX@{\xypolyPREFIX@}}\xypolynode@=\z@ }%
 \else
  \xy@@{\enter@{\xypolynode@=\the\xypolynode@
   \noexpand\def\noexpand\xypolyPREFIX@{\xypolyPREFIX@}}\xypolynode@=\z@ }%
  \expandafter\def\expandafter\xypolyPREFIX@\expandafter{\xypolyPREFIX@@}%
 \fi
 \xypoly@@i }
```

Currently any tokens remaining before the opening brace are discarded, with an accompanying message.

```
\xydef@\xypoly@@i#1#{\begingroup
 \DN@{#1}\ifx\next@\empty\else
 \xywarning@{discarding unused tokens: #1}\fi \xypoly@@ }
```

Next it is time to analyse the braced information, e.g. to set switches and/or adjust the scale. If this information is empty `{}` then the default values are used.

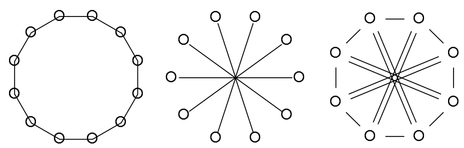
```
\xydef@\xypoly@@#1{\xypolydefaults@
 \DN@{#1}\ifx\next@\empty\DN@{\xypoly@@@}%
 \else \DN@{\xyFN@\xypolyparams@#1~!}\fi \next@}
```

```
\xydef@\xypolydefaults@{\def\xypolySCALE@{}%
 \def\xypolySPOKES@{{}}\def\xypostSPOKES@{}\let\xypreSPOKES@\relax
```

```
\def\xypolySIDES@{{-}}\def\xypostSIDES@{}\let\xypreSIDES@\relax
\def\xypolyVERTEX@{}\def\xypolyALIGN@{}}
```

Vertex object: Unless the first character is `~`, signifying a "switch", then the whole of the braced material is taken as specifying the ⟨object⟩ for each vertex. It will be typeset with a circular edge using `\drop[o]...`, except when there is just a single token ⟨tok⟩. In this case it is dropped as `\drop=0{`⟨tok⟩`}`, having zero size. An object can also be dropped at each vertex using the switch `~*{...}`, in which case it will be circular, with the current *objectmargin* applied.

The next example illustrates three different ways of specifying a `\circ` at the vertices.



```
\xy/r2pc/: {\xypolygon12{\circ}},
+/r5pc/,{\xypolygon10{~<{-}~>{}{\circ}}},
+/r5pc/,{\xypolygon8{~*{\circ}~<=}}\endxy
```

```
{\xyuncatcodes \gdef\next{*=0}}
\xylet@\xypolynullvertex@=\next
{\xyuncatcodes \gdef\next{*[o]}}
\xylet@\xypolyvertex@=\next
{\xyuncatcodes \gdef\next{*+[o]}}
\xylet@\xypolywidevertex@=\next
\xydef@\xypolyparams@{%
 \ifx\space@\next \expandafter\DN@\space{\xyFN@\xypolyparams@}%
 \else\ifx\next~\DN@~{\xyFN@\xypolywhich@}%
 \else\ifx\next\bgroup\DN@##1~!{%
%  \ifx\empty\xypolyVERTEX@\def\xypolyVERTEX@{*[o]{##1}}%
  \ifx\empty\xypolyVERTEX@\expandafter
   \def\expandafter\xypolyVERTEX@\expandafter{\xypolyvertex@{##1}}%
  \else\xypolyVERTEXerror@{##1}\fi \xypoly@@@}%
 \else\DN@##1~!{\DN@{##1}\ifx\next@\empty\else
   \ifx\empty\xypolyVERTEX@\def\xypolyVERTEX@{*##1}%
%     \checkpolytok@\xypolyVERTEX@{*=0}##1@@!\relax
    \expandafter\checkpolytok@\expandafter\xypolyVERTEX@
     \expandafter{\xypolynullvertex@}##1@@!\relax
   \else\xypolyVERTEXerror@{##1}\fi\fi \xypoly@@@}%
 \fi\fi\fi \next@ }
\xydef@\xypolyVERTEXerror@#1{\xywarning@{%
 vertex already specified, discarding unused tokens:#1}}
```

Switches

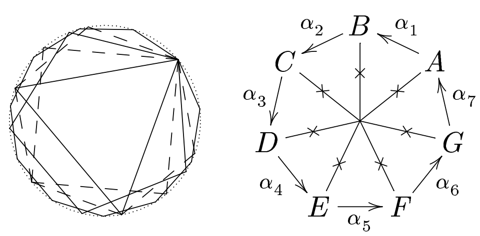
The allowable switches are given in the following table:

`~:{...}`	useful for rescaling.
`~*{`⟨object⟩`}`	⟨object⟩ at each vertex.

`~={`⟨angle⟩`}`	align first vertex.
`~<{...}`	directional for “spokes”;
`~<<{`⟨arrow⟩`}`	use ⟨arrow⟩ for spokes;
`~<>{...}`	labels & breaks on spokes.
`~>{...}`	directional for “sides”;
`~><{`⟨arrow⟩`}`	use ⟨arrow⟩ for sides;
`~>>{...}`	labels & breaks on sides.

Using `~<<{`⟨arrow⟩`}` or `~><{`⟨arrow⟩`}` is most appropriate when arrowheads are required on the sides or spokes, or when labels/breaks are required. Here ⟨arrow⟩ is as in figure 3.2, so it can be used simply to specify the style of directional to be used. Thus `~<<{}` sets each spoke as a default arrow, pointing outwards from the centre; `~<<{@{-}}` suppresses the arrowhead, while `~><{@{}}` uses an empty arrow along the sides. Labels and breaks are specified with `~<>{...}` and `~>>{...}`, where the `{...}` use the notation for a ⟨label⟩, as in figure 3.1.

When no tips or breaks are required then the switches `~<{...}` and `~>{...}` are somewhat faster, since less processing is needed. Labels can still be specified with `~<>{...}` and `~>>{...}`, but now using the kernel’s ⟨place⟩ notation of figure 1.1. In fact any kernel code can be included using these switches. With `~<>` the current p and c are the centre and vertex respectively, while for `~>>` they are the current vertex and the previous vertex. (The connection from vertex `"`⟨number⟩`"` to vertex `"1"` is done last.) The pyramid above is an example of how this can be used. Both `~<{...}` and `~<<{`⟨arrow⟩`}` can be specified together, but only the last will actually be used; similarly for `~>{...}` and `~><{`⟨arrow⟩`}`.




```
\def\alphanum{\ifcase\xypolynode\or A
\or B\or C\or D\or E\or F\or G\or H\fi}
\xy/r3pc/: {\xypolygon3{~={40}}},
{\xypolygon4{~={40}~>{{--}}}},
{\xypolygon5{~={40}}},
{\xypolygon6{~={40}~>{{--}}}},
{\xypolygon11{~={40}}},
{\xypolygon50{~={40}~>.}}, +/r8pc/,
{\xypolygon7{~<<{@{-}}~><{}
 ~<>{|*@{x}}~*{\alphanum}
 ~>>{_{\alpha_\xypolynode^{}}}}}
\endxy
```

Use of the `~={...}` switch was described earlier. When using the `~:{...}` more can be done than just setting the base. In fact any kernel code can be supplied here. It is processed prior to any other part of the polygon. The graphics state has c at the centre of the polygon, p at the origin of coordinates within the picture and has basis unchanged from what has previously been established. The current point c will be reset to the centre following any code interpreted using this switch.

Note that `~!` was appended by `\xypoly@@`, in order to indicate the end of the braced tokens. The macro `\xypolywhich@` is the main parser, with `\xypoly@SPOKES@` and `\xypoly@SIDES@` handling the subcases for `~<` and `~>`.

```
\xydef@\xypolywhich@{%
              \ifx\next!\DN@!{\xypoly@@@}%
 \else \addLT@\ifx\next \addLT@\DN@{\xyFN@\xypoly@SPOKES@}%
 \else \addGT@\ifx\next \addGT@\DN@{\xyFN@\xypoly@SIDES@}%
 \else \addEQ@\ifx\next \addEQ@\DN@##1{%
  \def\xypolyALIGN@{##1}\xyFN@\xypolyparams@ }%
 \else \ifx :\next \DN@:##1{%
  \def\xypolySCALE@{##1}\xyFN@\xypolyparams@ }%
% \else\ifx\next*\DN@*##1{\def\xypolyVERTEX@{*+[o]{##1}}\xyFN@\xypolyparams@}%
 \else\ifx\next*\DN@*##1{\expandafter
   \def\expandafter\xypolyVERTEX@\expandafter{\xypolywidevertex@{##1}}%
  \xyFN@\xypolyparams@}%
 \fi\fi\fi\fi\fi\fi \next@ }
```

```
\xydef@\xypoly@SPOKES@{%
       \addLT@\ifx\next \addLT@\DN@##1{%
  \def\xypreSPOKES@{##1}\def\xypolySPOKES@{}\xyFN@\xypolyparams@ }%
 \else \addGT@\ifx\next \addGT@\DN@##1{%
  \def\xypostSPOKES@{##1}\xyFN@\xypolyparams@ }%
 \else \ifx\next\bgroup \DN@##1{%
  \def\xypolySPOKES@{##1}\let\xypreSPOKES@\relax
  \checkpolytok@\xypolySPOKES@{}##1@@!{\xyFN@\xypolyparams@}}%
 \else \DN@##1{%
  \let\xypreSPOKES@\relax \def\xypolySPOKES@{{##1}}\xyFN@\xypolyparams@}%
 \fi\fi\fi \next@ }
```

```
\xydef@\xypoly@SIDES@{%
       \addLT@\ifx\next \addLT@\DN@##1{%
  \def\xypreSIDES@{##1}\def\xypolySIDES@{}\xyFN@\xypolyparams@ }%
 \else \addGT@\ifx\next \addGT@\DN@ ##1{%
  \def\xypostSIDES@{##1}\xyFN@\xypolyparams@ }%
 \else \ifx\next\bgroup \DN@##1{%
  \def\xypolySIDES@{##1}\let\xypreSIDES@\relax
  \checkpolytok@\xypolySIDES@{}##1@@!{\xyFN@\xypolyparams@}}%
 \else \DN@##1{%
  \let\xypreSIDES@\relax \def\xypolySIDES@{{##1}}\xyFN@\xypolyparams@}%
 \fi\fi\fi \next@ }
```

A further simplification exists for sides and spokes without ⟨arrow⟩s. If ⟨tok⟩ is a single character then `~>`⟨tok⟩, `~>{`⟨tok⟩`}`, `~>{{`⟨tok⟩`}}` all specify the directional `\dir{`⟨tok⟩`}`; similarly with the `~<` switch. On the other hand, compound directionals require all the braces, e.g. `~>{{--}}` and `~>{2{.}}`.

This is achieved with the macro `\checkpolytok@#1#2#3#4@!#5` which handles the bare ⟨tok⟩ case by adding braces around ⟨tok⟩ which has been passed as `#3`, preceeding it with `#2`. This is then `\def`'d to the control-sequence name passed as `#1`. Finally continue with the command passed as `#5`.

```
\xydef@\checkpolytok@#1#2#3#4@!#5{\DNii@{#1}\DN@{#4}%
 \ifx\next@\empty\expandafter\def\nextii@{{}}%
 \else\ifx#4@\expandafter\def\nextii@{#2{#3}}\fi\fi #5}
```

After all switches have been processed, remaining tokens are used to specify the ⟨object⟩ for each vertex. Such tokens will be used directly after a `\drop`, so can include object ⟨modifier⟩s as in figure 1.5. If an ⟨object⟩ has already been specified, using the `~*` switch, then the following message will be written to the TEX log:

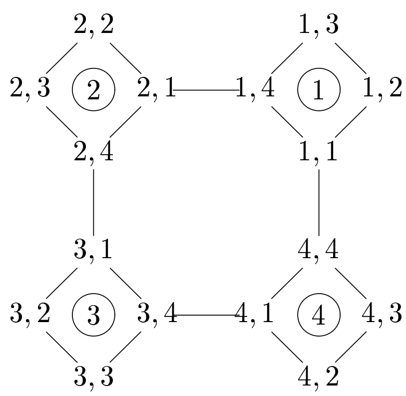
```
Xy-pic Warning:  vertex already specified, discarding unused tokens:
```

with tokens at the end indicating what remains unprocessed. Similarly extra tokens before the `{...}` generate a message:

```
Xy-pic Warning: discarding unused tokens:
```

Nested Polygons

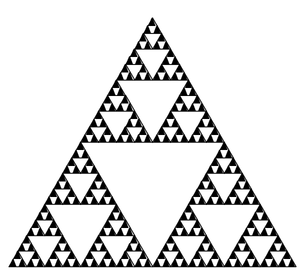
When `\xypolygon` is specified within either a `~<>{...}` or `~>>{...}` switch for another polygon, then the inner polygon inherits a name which incorporates also the number of the part on which it occurs, as given by `\xypolynode`. This name is accessed using `\xypolyname`. In the following example the inner polygon is placed using `~<>` in order to easily adjust its orientation to the outward direction of the spokes.




```
\xypolygon4{~:{/r5pc/:}
 ~<>{*\frm<8pt>{o}\xypolygon4{~:{/-2pc/:}
   ~*{\xypolyname\xypolynode}}}
 [o]=<5pc>{\xypolynode}}
```

Notice how nested polygons inherit names `"1,1"`, `"1,2"`, ..., `"4,1"`, ..., `"4,4"` for their vertices. If a ⟨prefix⟩ is supplied at the outermost level then the names become: `"`⟨prefix⟩i, j`"`. Specifying a ⟨prefix⟩ for the inner polygon overrides this naming scheme. The same names may then be repeated for each of the inner polygons, allowing access afterwards only to the last—possibly useful as a memory saving feature when the vertices are not required subsequently.

Four levels of nesting gives a quite acceptable "Sierpinski gasket". The innermost triangle is provided by `\blacktriangle` from the $\mathcal{AMS}$ symbol font `msam5`, at 5-point size. Further levels can be achieved using the PostScript backend, otherwise line segments become too small to be rendered using XY-fonts.



```
\font\msamv=msam5 at 5pt
\def\blacktriangle{{\msamv\char'116}}
\def\objectstyle{\scriptscriptstyle}
\xypolygon3{~:{/r5.2pc/:}
 ~>{}~<>{?\xypolygon3"a"{~:{(.5,0):}
 ~>{}~<>{?\xypolygon3"b"{~:{(.5,0):}
 ~>{}~<>{?\xypolygon3"c"{~:{(.5,0):}
 ~>{}~<>{?\xypolygon3"d"{~:{(.5,0):}
~<>{?*!/d.5pt/=0\hbox{\blacktriangle}}
}} }} }} }} }
```

Note the use of naming in this example; when processing this manual it saves 13,000+ words of main memory and 10,000+ string characters as well as 122 strings and 319 multi-letter control sequences.

```
\xydef@\xytrigerror@#1#2{\string#1 only works from 0 to 12}

\xydef@\cosPIon#1{\ifcase#1 1\or-1\or0\or.5\or\halfroottwo\or
 .809017\or\halfrootthree\or.900969\or.92388\or.939693\or.951057\or
 .959493\or.965926\else1\xytrigerror@\cosPIon{#1}\fi}
\xydef@\sinPIon#1{\ifcase#1 0\or0\or1\or\halfrootthree\or\halfroottwo\or
 .587785\or.5\or.433884\or.382683\or.34202\or.309017\or
 .281733\or.258819\else0\xytrigerror@\sinPIon{#1}\fi}

\xydef@\cosTwoPIon#1{\ifcase#1 1\or1\or-1\or-\cosPIon3\or\cosPIon2\or
 \sinPIon{10}\or\cosPIon3\or.62349\or\cosPIon4\or
 .766044\or\cosPIon5\or.841254\or\cosPIon6%
 \else1\xytrigerror@\cosTwoPIon{#1}\fi}
\xydef@\sinTwoPIon#1{\ifcase#1 0\or0\or0\or-\sinPIon3\or\sinPIon2\or
 \cosPIon{10}\or\sinPIon3\or.781831\or\sinPIon4\or
 .642788\or\sinPIon5\or.540641\or\sinPIon6%
 \else0\xytrigerror@\sinTwoPIon{#1}\fi}

\xydef@\cosThreePIon#1{\ifcase#1 1\or-1\or0\or-1\or-\cosPIon4\or
 -\cosPIon5\or\cosPIon2\or.222521\or\sinPIon8\or\cosPIon3\or
 \sinTwoPIon5\or.654861\or\cosPIon4%
 \else1\xytrigerror@\cosThreePIon{#1}\fi}
\xydef@\sinThreePIon#1{\ifcase#1 0\or0\or1\or0\or\sinPIon4\or
 \sinTwoPIon5\or\sinPIon2\or.974928\or\cosPIon8\or\sinPIon3\or
 \cosTwoPIon5\or.75575\or\sinPIon4%
 \else0\xytrigerror@\sinThreePIon{#1}\fi}

\xydef@\cosFourPIon#1{\ifcase#1 1\or1\or1\or-\cosPIon3\or-1\or-\cosPIon5\or
 \cosTwoPIon3\or-\cosThreePIon7\or0\or.173648\or\cosTwoPIon5\or
 .415415\or\cosPIon3%
 \else1\xytrigerror@\cosFourPIon{#1}\fi}
\xydef@\sinFourPIon#1{\ifcase#1 0\or0\or0\or-\sinPIon3\or0\or\sinPIon5\or
 \sinTwoPIon3\or\sinThreePIon7\or1\or.984808\or\sinTwoPIon5\or
 .909632\or\sinPIon3%
 \else0\xytrigerror@\sinFourPIon{#1}\fi}

\xydef@\cosFivePIon#1{\ifcase#1 1\or-1\or0\or\cosPIon3\or-\cosPIon4\or
 -1\or-\cosPIon6\or-\cosTwoPIon7\or-\cosThreePIon8\or-\cosFourPIon9\or
 0\or.142315\or\sinPIon{12}%
 \else1\xytrigerror@\cosFivePIon{#1}\fi}
```

```
\xydef@\sinFivePIon#1{\ifcase#1 0\or0\or1\or-\sinPIon3\or-\sinPIon4\or
 0\or\sinPIon6\or\sinTwoPIon7\or\sinThreePIon8\or\sinFourPIon9\or1\or
 .989821\or\cosPIon{12}%
 \else0\xytrigerror@\sinFivePIon{#1}\fi}
```

Coordinates for the vertices are read from trigonometry tables using control sequences

`\cosPIon`n	`\sinPIon`n
`\cosTwoPIon`n	`\sinTwoPIon`n
`\cosThreePIon`n	`\sinThreePIon`n
`\cosFourPIon`n	`\sinFourPIon`n
`\cosFivePIon`n	`\sinFivePIon`n

which expand to the values of $\cos\frac{\pi}{n}$, $\sin\frac{\pi}{n}$, $\cos\frac{2\pi}{n}$, $\ldots$, $\sin\frac{5\pi}{n}$ and are also available for other uses. The parameter n must be a non-negative integer up to 12. The complete table is in figure 3.6.

Specific Polygons: All the information from the switches is passed as 8 separate parameters to a macro appropriate to the number of sides. The parameters have the following uses, and are set as shown:

`#1`	pre-arrow: set by `~<<{...}` else `;`
`#2`	vertex ⟨object⟩, set by `~*{...}`
`#3`	prefix for naming
`#4`	post-vertex: empty with `\ars` else `**@{...}` where `{...}` is set by `~<{...}`
`#5`	post-arrow/vertex: set by `~<>{...}`
`#6`	pre-edge/arrow, set by `~><{...}`
`#7`	post-edge: empty with `\ars` else `**@{...}` where `{...}` is set by `~>{...}`
`#8`	post-arrow/edge, set by `~>>{...}`

The organisation of this information is done by `\xyPolygon@@@#1` which takes as a single parameter a macro appropriate to the number of sides, supplied by `\xypoly@@@`.

```
\xydef@\xypoly@@@{\count@=\xypolyNUMSIDES@\relax
 \ifx\xypolyALIGN@\empty
  \ifnum\count@<\thr@@\relax\DN@{\xypolytoosmall@{\the\count@}}%
  \else \advance\count@-\thr@@
   \ifcase\count@ \DN@{\xyPolygon@@@\xytriangle@@}%
   \or \DN@{\xyPolygon@@@\xysquare@@}%
   \or \DN@{\xyPolygon@@@\xypentagon@@}%
   \or \DN@{\xyPolygon@@@\xyhexagon@@}%
   \or \DN@{\xyPolygon@@@\xyheptagon@@}%
   \or \DN@{\xyPolygon@@@\xyoctagon@@}%
   \or \DN@{\xyPolygon@@@\xynonagon@@}%
   \or \DN@{\xyPolygon@@@\xydecagon@@}%
   \or \DN@{\xyPolygon@@@\xyundecagon@@}%
   \or \DN@{\xyPolygon@@@\xydodecagon@@}%
   \else \DN@{\xylargePolygon@{0}{\xypolyNUMSIDES@}}%
  \fi\fi
```

```
 \else
  \DN@{\xylargePolygon@{\xypolyALIGN@}{\xypolyNUMSIDES@}}%
 \fi \next@ \xypoly@@@x}

\xydef@\xypoly@@@x{\restore\xypoly@x}

\xydef@\xypolytoosmall@#1{\xyerror@{#1 sides is too few for a polygon}}
```

Here is `\xyPolygon@@@`. It constructs a list of tokens to be interpreted, after having closed the grouping that was current while parsing. It starts by setting the value of `\xypolyNUMSIDES@`, then a `\save` before adjusting the scale according to the value of `\xypolySCALE@`.

```
\xydef@\xyPolygon@@@#1{%
 \toks@={\def\xypolyNUMSIDES@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\xypolyNUMSIDES@}\save}%
%
 \ifx\empty\xypolySCALE@ \addtotoks@{\POS"}%
 \else \addtotoks@{\POS;0;}%
  \expandafter\addtotoks@\expandafter{\xypolySCALE@\relax\POS"}%
 \fi
 \expandafter\addtotoks@\expandafter{\xypolyPREFIX@0"\relax}%
%
 \ifx\xypreSPOKES@\relax
  \addtotoks@{#1{;}}%
  \expandafter\addtotoks@\expandafter{\expandafter{\xypolyVERTEX@}}%
  \expandafter\addtotoks@\expandafter{\expandafter{\xypolyPREFIX@}}%
  \DN@##1\next@{\def\tmp@{**\dir##1}}%
   \expandafter\next@\expandafter{\xypolySPOKES@}\next@
   \expandafter\addtotoks@\expandafter{\expandafter{\tmp@}}%
 \else
  \expandafter\DN@\expandafter{\expandafter\ar\xypreSPOKES@}%
  \addtotoks@{#1}%
  \expandafter\addtotoks@\expandafter{\expandafter{\next@}}%
  \expandafter\addtotoks@\expandafter{\expandafter{\xypolyVERTEX@}}%
  \expandafter\addtotoks@\expandafter{\expandafter{\xypolyPREFIX@}{}}%
 \fi
 \expandafter\addtotoks@\expandafter{\expandafter{\xypostSPOKES@}}%
%
 \ifx\xypreSIDES@\relax
  \addtotoks@{{;}}%
  \DN@##1\next@{\def\tmp@{**\dir##1}}%
   \expandafter\next@\expandafter{\xypolySIDES@}\next@
   \expandafter\addtotoks@\expandafter{\expandafter{\tmp@}}%
 \else
  \expandafter\DN@\expandafter{\expandafter\ar\xypreSIDES@}%
  \expandafter\addtotoks@\expandafter{\expandafter{\next@}{}}%
 \fi
 \expandafter\addtotoks@\expandafter{\expandafter{\xypostSIDES@}}%
 \ifx\xypolyALIGN@\empty
  \ifnum\xypolyNUMSIDES@>12\addtotoks@{{0}}\fi
 \else
  \expandafter\addtotoks@\expandafter{\expandafter{\xypolyALIGN@}}%
```

```
 \fi
 \expandafter\DN@\expandafter{\expandafter\endgroup\the\toks@ }%
 \next@}
```

Note how `\endgroup` closes the grouping. All the information for the layout of the polygon is contained in `\xypolyNUMSIDES@` and the specific polygon macro contained in the final `\next@`.

Here is where the polygons are defined, individually.

```
\xydef@\xytriangle@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
 \POS#1"#30"+(0,1)#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(-\cosPIon6, -.5)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\cosPIon6,-.5)#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#33"#6"#31"#7#8\relax }
```

```
\xydef@\xysquare@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
 \POS#1"#30"+(\halfroottwo,\halfroottwo)#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(-\halfroottwo,\halfroottwo)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\halfroottwo,\halfroottwo)#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(-\halfroottwo,\halfroottwo)#2="#34"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#33"#6"#34"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#34"#6"#31"#7#8\relax }
```

```
\xydef@\xypentagon@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
 \POS#1"#30"+(\cosPIon{10},\sinPIon{10})#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(0,1)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(-\cosPIon{10},\sinPIon{10})#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
```

```
 \POS"#30"#1"#30"-(\sinPIon5,\cosPIon5)#2="#34"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#33"#6"#34"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\sinPIon5,-\cosPIon5)#2="#35"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#34"#6"#35"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#35"#6"#31"#7#8\relax }

\xydef@\xyhexagon@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
 \POS#1"#30"+(1,0)#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(.5,\halfrootthree)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(-.5,\halfrootthree)#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(1,0)#2="#34"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#33"#6"#34"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(.5,\halfrootthree)#2="#35"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#34"#6"#35"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(-.5,\halfrootthree)#2="#36"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#35"#6"#36"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#36"#6"#31"#7#8\relax }

\xydef@\xyheptagon@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
 \POS#1"#30"+(\sinTwoPIon7,\cosTwoPIon7)#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(0,1)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(-\sinTwoPIon7,\cosTwoPIon7)#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\sinThreePIon7,\cosThreePIon7)#2="#34"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#33"#6"#34"#7#8\relax
```

```
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\sinPIon7,\cosPIon7)#2="#35"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#34"#6"#35"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\sinPIon7,-\cosPIon7)#2="#36"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#35"#6"#36"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\sinThreePIon7,-\cosThreePIon7)#2="#37"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#36"#6"#37"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#37"#6"#31"#7#8\relax }

\xydef@\xyoctagon@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
 \POS#1"#30"+(\cosPIon8,\sinPIon8)#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(\sinPIon8,\cosPIon8)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\sinPIon8,-\cosPIon8)#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\cosPIon8,-\sinPIon8)#2="#34"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#33"#6"#34"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\cosPIon8,\sinPIon8)#2="#35"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#34"#6"#35"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\sinPIon8,\cosPIon8)#2="#36"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#35"#6"#36"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\sinPIon8,-\cosPIon8)#2="#37"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#36"#6"#37"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\cosPIon8,-\sinPIon8)#2="#38"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#37"#6"#38"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#38"#6"#31"#7#8\relax }

\xydef@\xynonagon@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
```

```
 \POS#1"#30"+(\sinFourPIon9,\cosFourPIon9)#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(\sinTwoPIon9,\cosTwoPIon9)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(0,1)#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(-\sinTwoPIon9,\cosTwoPIon9)#2="#34"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#33"#6"#34"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(-\sinFourPIon9,\cosFourPIon9)#2="#35"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#34"#6"#35"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\cosPIon6,\sinPIon6)#2="#36"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#35"#6"#36"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\sinPIon9,\cosPIon9)#2="#37"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#36"#6"#37"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\sinPIon9,-\cosPIon9)#2="#38"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#37"#6"#38"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\cosPIon6,-\sinPIon6)#2="#39"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#38"#6"#39"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#39"#6"#31"#7#8\relax }

\xydef@\xydecagon@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
 \POS#1"#30"+(1,0)#2="#31"#4#5\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#30"#1"#30"+(\cosPIon5,\sinPIon5)#2="#32"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#31"#6"#32"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\sinPIon{10},\cosPIon{10})#2="#33"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#32"#6"#33"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\sinPIon{10},-\cosPIon{10})#2="#34"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
```

```
\POS"#33"#6"#34"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\cosPIon5,-\sinPIon5)#2="#35"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#34"#6"#35"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(1,0)#2="#36"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#35"#6"#36"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\cosPIon5,\sinPIon5)#2="#37"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#36"#6"#37"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\sinPIon{10},\cosPIon{10})#2="#38"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#37"#6"#38"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(\sinPIon{10},-\cosPIon{10})#2="#39"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#38"#6"#39"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(\cosPIon5,-\sinPIon5)#2="#310"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#39"#6"#310"#7#8\relax
\xy@@{\advance\xypolynode@\@ne}%
\POS"#310"#6"#31"#7#8\relax }

\xydef@\xyundecagon@@#1#2#3#4#5#6#7#8{%
\xy@@{\xypolynode@=\@ne}%
\POS#1"#30"+(\sinFourPIon{11},\cosFourPIon{11})#2="#31"#4#5\relax
\xy@@{\advance\xypolynode@\@ne}%
\POS"#30"#1"#30"+(\sinTwoPIon{11},\cosTwoPIon{11})#2="#32"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#31"#6"#32"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(0,1)#2="#33"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#32"#6"#33"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(-\sinTwoPIon{11},\cosTwoPIon{11})#2="#34"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#33"#6"#34"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(-\sinFourPIon{11},\cosFourPIon{11})#2="#35"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#34"#6"#35"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\sinFivePIon{11},\cosFivePIon{11})#2="#36"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
```

```
\POS"#35"#6"#36"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\sinThreePIon{11},\cosThreePIon{11})#2="#37"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#36"#6"#37"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\sinPIon{11},\cosPIon{11})#2="#38"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#37"#6"#38"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(\sinPIon{11},-\cosPIon{11})#2="#39"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#38"#6"#39"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(\sinThreePIon{11},-\cosThreePIon{11})#2="#310"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#39"#6"#310"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(\sinFivePIon{11},-\cosFivePIon{11})#2="#311"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#310"#6"#311"#7#8\relax
\xy@@{\advance\xypolynode@\@ne}%
\POS"#311"#6"#31"#7#8\relax }

\xydef@\xydodecagon@@#1#2#3#4#5#6#7#8{%
 \xy@@{\xypolynode@=\@ne}%
\POS#1"#30"+(\cosPIon{12},\sinPIon{12})#2="#31"#4#5\relax
\xy@@{\advance\xypolynode@\@ne}%
\POS"#30"#1"#30"+(\halfroottwo,\halfroottwo)#2="#32"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#31"#6"#32"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"+(\sinPIon{12},\cosPIon{12})#2="#33"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#32"#6"#33"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\sinPIon{12},-\cosPIon{12})#2="#34"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#33"#6"#34"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\halfroottwo,-\halfroottwo)#2="#35"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#34"#6"#35"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\cosPIon{12},-\sinPIon{12})#2="#36"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
\POS"#35"#6"#36"#7#8\relax
\xy@@{\advance\xypolynode@\tw@}%
\POS"#30"#1"#30"-(\cosPIon{12},\sinPIon{12})#2="#37"#4#5\relax
\xy@@{\advance\xypolynode@\m@ne}%
```

```
 \POS"#36"#6"#37"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\halfroottwo,\halfroottwo)#2="#38"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#37"#6"#38"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"-(\sinPIon{12},\cosPIon{12})#2="#39"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#38"#6"#39"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\sinPIon{12},-\cosPIon{12})#2="#310"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#39"#6"#310"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\halfroottwo,-\halfroottwo)#2="#311"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#310"#6"#311"#7#8\relax
 \xy@@{\advance\xypolynode@\tw@}%
 \POS"#30"#1"#30"+(\cosPIon{12},-\sinPIon{12})#2="#312"#4#5\relax
 \xy@@{\advance\xypolynode@\m@ne}%
 \POS"#311"#6"#312"#7#8\relax
 \xy@@{\advance\xypolynode@\@ne}%
 \POS"#312"#6"#31"#7#8\relax }
```

```
\xydef@\xylargePolygon@#1#2{%
 \dimen@=#1\p@
 \xyPolygon@@@\xypolygon@@}
% \xyerror@{Sorry, polygons with #2 sides are not implemented; maximum 12.}{}}
```

We need a looping construction that will not interfere with others used at a high level.

```
\xydef@\xypolyrepeat@{\fi}
\xydef@\xypolybody@{}
\xydef@\xypolyiterate@{\xypolybody@\expandafter\xypolyiterate@\else\fi}
\xydef@\xypolyloop@#1\xypolyrepeat@{\def\xypolybody@{#1}\xypolyiterate@}
```

The counter `\xypolynode@` keeps track of the current vertex, spoke and side. Since the user may choose to place another `\xypolygon` along any such piece, it is necessary to calculate the angle for each vertex afresh, using only information passed via the parameters `#1`,. . . ,`#9`.

```
\xydef@\xypolygon@@#1#2#3#4#5#6#7#8#9{\save
 \vfromcartesianangle@{#9}\edef\next@{\the\X@c,\the\Y@c\addGT@\empty}%
  \xy@@{\xypolynode@=\@ne}%
  \toks@={\POS#1"#30"+}%
  \expandafter\expandafter\expandafter\addtotoks@\expandafter
   \expandafter\expandafter{\addLT@\empty}%
  \expandafter\addtotoks@\expandafter{\next@#2="#31"#4#5\relax}%
  \xy@@\dontleave@ \the\toks@
%
 \xypolyloop@
  \dimen@=360\p@ \count@@=\xypolyNUMSIDES@\relax \advance\dimen@ .5\count@@
  \divide\dimen@ \xypolyNUMSIDES@ \relax
```

```
  \multiply\dimen@\xypolynode@ \relax \advance\dimen@ #9\p@
  \edef\tmp@{%
    \noexpand\vfromcartesianangle@{\expandafter\removePT@\the\dimen@}}%
  \tmp@ \edef\next@{\the\X@c,\the\Y@c\addGT@\empty}%
  \xy@@{\advance\xypolynode@\@ne}\count@=\xypolynode@
  \toks@={\POS"#30"#1"#30"+}\edef\nextii@{\the\count@}%
  \expandafter\expandafter\expandafter\addtotoks@\expandafter
   \expandafter\expandafter{\addLT@\space}%
  \expandafter\addtotoks@\expandafter{\next@#2="#3}%
  \expandafter\addtotoks@\expandafter{\nextii@"#4#5\relax}%
  \the\toks@
%
  \count@=\xypolynode@ \xy@@{\advance\xypolynode@\m@ne}%
  \toks@={\POS"#3}\expandafter\addtotoks@\expandafter{\the\xypolynode@"#6"#3}%
  \expandafter\addtotoks@\expandafter{\the\count@"#7#8\relax}%
  \the\toks@
  \xy@@{\advance\xypolynode@\@ne}\count@=\xypolyNUMSIDES@ \relax
  \ifnum\xypolynode@<\count@ \relax \xypolyrepeat@
%
  \toks@={\POS"#3}%
  \expandafter\addtotoks@\expandafter{\xypolyNUMSIDES@"#6"#31"#7#8\relax}%
  \the\toks@ \restore }
```

The end & Log

DOCMODE3

```

\xyendinput
```

```
% $Log: xypoly.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  09:28:35  ross
% cosmetic changes in documentation
%
% Revision 3.1  1995/09/05  20:28:57  ross
% Releasing version 3.1!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.12  1994/10/25  03:01:14  ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
%
% Created by Ross Moore, September 1994.
```

3.8 Lattice and web feature

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This feature provides macros to facilitate typesetting of arrangements of points within a 2-dimensional lattice or "web-like" structure.

Header:

```
%% $Id: xyweb.doc,v 3.3 1996/12/18 09:52:11 ross Exp $
%%
%% Xy-pic ``Lattice and web'' feature.
%% Copyright (c) 1994-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{web}{Lattice and web feature}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\message{lattices,}
```

We need two counters and macros to access their values.

```
\xynew@{count}\lattice@A
\xynew@{count}\lattice@B
\xydef@\latticeA{\lattice@A}
\xydef@\latticeB{\lattice@B}
```

Currently the only routines implemented with this feature are some "quick and dirty" macros for dropping objects at the points of an integer lattice. **To Do:** More sophisticated routines will be developed for later versions of Xy-pic, as the need arises.

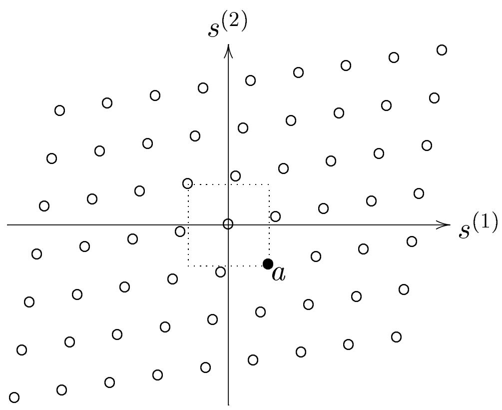
Mathematically speaking, let $\vec{u}$ and $\vec{v}$ be vectors pointing in independent directions within the plane. Then the lattice spanned by $\vec{u}$ and $\vec{v}$ is the infinite set of points L given by:

$$L \;=\; \{a\,\vec{u} + b\,\vec{v}\;;\;\; \text{for } a,\, b \text{ integers}\}\,.$$

Within XY-pic the vectors $\vec{u}$ and $\vec{v}$ can be established as the current coordinate basis vectors. The following macros typeset a finite subset of an abstract lattice.

`\xylattice#1#2#3#4`	points in lattice
`\croplattice#1#2#3#4#5#6#7#8`	...in specific rectangle.

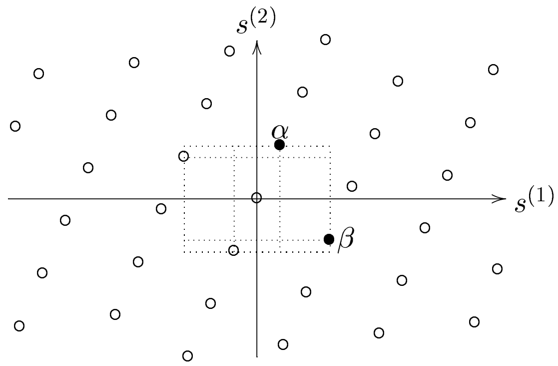
The parameters `#1` ... `#4` are to be integers $a_{\rm min}$, $a_{\rm max}$, $b_{\rm min}$ and $b_{\rm max}$, so that the portion of the lattice to be typeset is that collection of vectors in L for which $a_{\rm min} \le a \le a_{\rm max}$ and $b_{\rm min} \le b \le b_{\rm max}$.




```
\def\latticebody{%
\ifnum\latticeA=1 \ifnum\latticeB=-1 %
\else\drop{\circ}\fi\else\drop{\circ}\fi}
\xy *\xybox{0;<1.5pc,1mm>:<1mm,1.5pc>::
,0,{\xylattice{-4}4{-3}3}
 ,(1,-1)="a"*{\bullet}*+<2pt>!UL{a}
 ,(-1,1)."a"*\frm{.}="L"
,{"L"+L \ar "L"+R*+!L{s^{(1)}}}
,{"L"+D \ar "L"+U*+!D{s^{(2)}}}
\endxy
```

In the above code, notice how the basis is first established then the `\xylattice` typeset. Doing this within an `\xybox` allows axes to be sized and placed appropriately. Since lattice points are determined by their (integer) coordinate displacements, they can be revisited to add extra ⟨object⟩s into the overall picture. More generally, the origin for lattice-coordinates is the current ⟨pos⟩ c, when the `\xylattice` command is encountered. Easy accessibility is maintained, as seen in the next example.

When the basis vectors $\vec{u}$ and $\vec{v}$ are not perpendicular the collection of points with a, b in these ranges will fill out a skew parallelogram. Generally it is useful to plot only those points lying within a fixed rectangle. This is the purpose of `\croplattice`, with its extra parameters `#5` ... `#8` determining the 'cropping' rectangle within which lattice points will be typeset. Other points will not be typeset even when a and b are within the specified ranges. Explicitly the horizontal range of the cropping rectangle is $X_{\rm min}$ to $X_{\rm max}$, with $X_{\rm min}$ being the X-coordinate of the vector `#5` $\times\, \vec{u}$, where `#5` is a ⟨number⟩ (not necessarily an integer). Similarly $X_{\rm max}$ is the X-coordinate of `#6` $\times\, \vec{u}$. The vertical extents are $Y_{\min}$ and $Y_{\max}$, given by the Y-coordinates of `#7` $\times \vec{v}$ and `#8` $\times \vec{v}$ respectively.




```
\def\latticebody{%
\ifnum\latticeA=1 \ifnum\latticeB=-1 %
\else \drop{\circ}\fi\else
\ifnum\latticeA=0 \ifnum\latticeB=1\else
\drop{\circ}\fi\else\drop{\circ}\fi\fi}
\xy +(2,2)="o",0*\xybox{%
 0;<3pc,1.5mm>:<0.72pc,1.65pc>::,{"o"
 \croplattice{-4}4{-4}4{-2.6}{2.6}{-3}3}
 ,"o"+(0,1)="a"*{\bullet}*+!D{\alpha}
 ,"o"+(1,-1)="b"*{\bullet}*+!L{\beta}
 ,"o"+(0,-1)="c","o"+(-1,1)="d"
 ,"a"."c"="e",!DR*{};"a"**\dir{.}
 ,"e",!UL*{};"c"**\dir{.}
 ,"b"."d"="f",!DL*{};"b"**\dir{.}
 ,"f",!UR*{};"d"**\dir{.}
 ,"e"."f"*\frm{.}}="L","o"."L"="L"
 ,{"L"+L \ar "L"+R*+!L{s^{(1)}}}
 ,{"L"+D \ar "L"+U*+!D{s^{(2)}}}
\endxy
```

The routines `\xylattice` and `\croplattice` simply cycle through the allowable range of values for `\lattice@A` and `\lattice@B`, with `\lattice@B` varying within the inner loop.

```
\xydef@\xylattice#1#2#3#4{\xy@{LATTICE}{\xylattice@{#1}{#2}{#3}{#4}}}%

\xydef@\xylattice@#1#2#3#4{%
 \lattice@A=#1\relax
 \loop \bgroup\lattice@B=#3\relax
  \innerlatticeloop@{#4}%
  \edef\tmp@{\egroup
   \X@min=\the\X@min \X@max=\the\X@max
   \Y@min=\the\Y@min \Y@max=\the\Y@max
  }\tmp@ \ifnum\lattice@A<#2\relax \advance\lattice@A\@ne
 \repeat }

\xydef@\innerlatticeloop@#1{%
 \loop \enter@{\cfromthec@}\enter@{\cplusthec@}%
  \vfromcartesian@@ \the\lattice@A,\the\lattice@B @%
  \edef\latticeX{\expandafter\removePT@\the\X@c}%
  \edef\latticeY{\expandafter\removePT@\the\Y@c}%
```

```
 \leave@ \latticebody \leave@
\ifnum\lattice@B<#1\relax \advance\lattice@B\@ne
\repeat }
```

With `\xylattice` getting the picture size correct is simply a matter of passing `\X@min`, `X@max`, etc. outside of the inner group. However with `\croplattice` these are first set to correspond to the 'cropping rectangle' then subsequently merged with their original values.

```
\xydef@\croplattice#1#2#3#4#5#6#7#8{\xy@{LATTICE}%
 {\xycroplattice@{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}}%
```

```
\xydef@\xycroplattice@#1#2#3#4#5#6#7#8{%
 \enter@{\X@min=\the\X@min \X@max=\the\X@max
  \Y@min=\the\Y@min \Y@max=\the\Y@max}%
 \enter@{\cfromthec@}\enter@{\cplusthec@}%
  \vfromcartesian@@ #5,0 @\leave@ \X@min=\X@c \leave@
 \enter@{\cfromthec@}\enter@{\cplusthec@}%
  \vfromcartesian@@ #6,0 @\leave@ \X@max=\X@c \leave@
 \enter@{\cfromthec@}\enter@{\cplusthec@}%
  \vfromcartesian@@ 0,#7 @\leave@ \Y@min=\Y@c \leave@
 \enter@{\cfromthec@}\enter@{\cplusthec@}%
  \vfromcartesian@@ 0,#8 @\leave@ \Y@max=\Y@c \leave@
 \lattice@A=#1\relax
 \loop
  \bgroup\lattice@B=#3\relax
   \enter@{\cfromthec@}\innercroplatticeloop@{#4}\leave@
  \egroup
  \ifnum\lattice@A<#2\relax \advance\lattice@A\@ne
 \repeat \mergecropextents@ }
```

```
\xydef@\innercroplatticeloop@#1{%
 \loop
  \enter@{\cfromthec@}\enter@{\cplusthec@}%
   \vfromcartesian@@ \the\lattice@A,\the\lattice@B @%
   \edef\latticeX{\expandafter\removePT@\the\X@c}%
   \edef\latticeY{\expandafter\removePT@\the\Y@c}%
  \leave@ \DN@{\latticebody}%
  \ifdim\X@c<\X@min\DN@{}%
  \else\ifdim\X@c>\X@max\DN@{}%
  \else\ifdim\Y@c<\Y@min\DN@{}%
  \else\ifdim\Y@c>\Y@max\DN@{}%
  \fi\fi\fi\fi \next@ \leave@
 \ifnum\lattice@B<#1\relax \advance\lattice@B\@ne
 \repeat }
```

```
\xydef@\mergecropextents@{%
 \edef\tmp@{%
  \noexpand\ifdim\X@min>\the\X@min \X@min=\the\X@min\noexpand\fi
  \noexpand\ifdim\X@max<\the\X@max \X@max=\the\X@max\noexpand\fi
  \noexpand\ifdim\Y@min>\the\Y@min \Y@min=\the\Y@min\noexpand\fi
  \noexpand\ifdim\Y@max<\the\Y@max \Y@max=\the\Y@max\noexpand\fi}%
 \leave@ \tmp@ }
```

The `\latticebody` macro. At each lattice point within the specified range for a, b (and within the cropping rectangle when `\croplattice` is used), a macro called `\latticebody` is expanded. This is meant to be user-definable, so as to be able to adapt to any specific requirement. It has a default expansion given by ...

`\def\latticebody{\drop{\bullet}}`.

The following macros may be useful when specifying what to do at each point of the lattice.

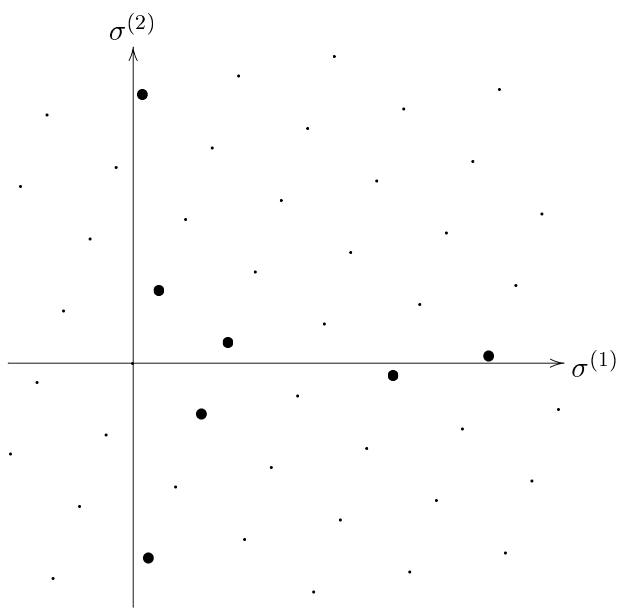
`\latticebody`	expanded at lattice points
`\defaultlatticebody`	resets to default
`\latticeA`	a-value of lattice point
`\latticeB`	b-value of lattice point
`\latticeX`	X-coord, offset in `pts`...
`\latticeY`	Y-coord, ...from lattice origin.

As in the examples presented above, the object dropped at the lattice point can be varied according to its location, or omitted altogether.

The default `\latticebody` macro ...

```
\xydef@\deflatticebody@{\def\latticebody{\drop{\bullet}}}
\xydef@\defaultlatticebody{\deflatticebody@}
\deflatticebody@
```

In the final example the `\latticebody` macro performs a calculation to decide which lattice points should be emphasised:




```
\def\latticebody{\dimen0=\latticeX pt
 \ifdim\dimen0>0pt \divide\dimen0 by 64
 \dimen0=\latticeY\dimen0 \relax
 \ifdim 0pt>\dimen0 \dimen0=-\dimen0 \fi
 \ifdim 10pt>\dimen0 \drop{\bullet}%
 \else\drop{.}\fi \else\drop{.}\fi}
\xy*\xybox{0;<3pc,2.57mm>:<.83pc,2.25pc>::
,0,{\croplattice{-3}5{-5}5
```

```
  {-1.3}{4.5}{-3.4}{4.4}}}="L"
,{"L"+L \ar "L"+!R*+!L{\sigma^{(1)}}}
,{"L"+D \ar "L"+!U*+!D{\sigma^{(2)}}}
\endxy
```

The end & Log
DOCMODE3

```
\xyendinput
```

```
% $Log: xyweb.doc,v $
% Revision 3.3  1996/12/18  09:52:11  ross
% no changes
%
% Revision 3.1  1995/09/05 20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/05  10:36:22  ross
% Ready for v3 release?
%
% NEW for version 3.
```

3.9 Circle, Ellipse, Arc feature

Vers. 3.4 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This feature provides a means to a specify circles of arbitrary radius, drawn with a variety of line styles. Similarly ellipses may be specified, having arbitrary major/minor axes aligned in any direction. A circular arc joining two points can be constructed with specified tangent direction at one end.

All the curves described here—circles, ellipses and sectors of these—are constructed using the curves from the `xycurve` extension. As such any comments given there concerning memory requirements are equally valid here, perhaps even more so. Use of the `xyps` PostScript back-end is highly recommended.

Header:

```
%% $Id: xyarc.doc,v 3.4 1997/05/18 01:13:24 ross Exp $
%%
%% Xy-pic ``Circles, Ellipses and Arcs'' feature.
%% Copyright (c) 1995-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The XY-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
```

```
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The XY-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{arc}{Circle, Ellipse, Arc feature}{\stripRCS$Revision: 3.4 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

The arcs are not truly circular or elliptical, but are approximations given by cubic Bézier segments. Hence the `xycurve` feature must be loaded.

```
\xyrequire{curve}
\xycatcodes
```

The basic ⟨object⟩ defined here is `\arc`. So far it is only implemented as a ⟨decor⟩ation, called `\ellipse`.

```
\xydef@\ellipse#1#{\xy@{ELLIPSE}{\hbox\bgroup\save@}%
 \let\endarc@=\endellipse@@ \xyFN@\preARC@ #1@}
```

```
\xydef@\endellipse@@{\xy@{endELLIPSE}{\leave@\origDirection@
 \edef\next@{\egroup \X@min=\the\X@min \X@max=\the\X@max
  \Y@min=\the\Y@min \Y@max=\the\Y@max }\next@}\let\bstartPLACE@=\relax}
```

```
\xydef@\preARC@{%
 \ifx\next\space@ \expandafter\DN@\space{\xyFN@\preARC@}%
 \else\ifx\next @\DN@{\xy@{ellipseARC}\ellipticARC@ \arc@}%
 \else\ifx\next ~\DN@~##1{\tildeARC@{##1}}%
 \else\DN@{\xy@{ellipticARC}\ellipticARC@ \arc@}%
 \fi\fi\fi \next@ }
```

```
%\xydef@\arc#1#{\xy@{ARC}{\hbox\bgroup\save@}%
%  \let\endarc@=\endarc@@\arc@{#1}}
%\xydef@\endarc@@{\xy@{endARC}{\leave@\origDirection@
```

We start by recording the curve ⟨style⟩ and initialising some methods and ⟨pos⟩ names, to be able to easily access important ⟨pos⟩ and ⟨direction⟩ information.

When the ... in `\arc...{`⟨style⟩`}` is particularly simple, e.g. ⟨empty⟩ or `^` or `_`, then we can branch immediately to `\fullCircle@`. Otherwise more parsing is required.

```
\xydef@\arc@#1@#2{\DN@{#2}%
 \ifx\next@\empty \xy@{arcSTYLE}{\def\arcSTYLE{-}}%
 \else \xy@{arcSTYLE}{\def\arcSTYLE{#2}}\fi \arc@i
 \DNii@{#1}%
 \ifx\nextii@\empty \DN@{\fullCircle@}%
```

```
  \else \DN@{_}%
   \ifx\next@\nextii@ \xy@{_}{\def\arcOrient@@{_}}%
    \let\arcOrient@@=\next@ \DN@{\fullCircle@}%
  \else \DN@{^}%
   \ifx\next@\nextii@ \xy@{^}{\def\arcOrient@@{^}}%
    \let\arcOrient@@=\next@ \DN@{\fullCircle@}%
  \else \DN@{\def\afterARCextents@@{%
    \xy@{aftARCext}{\let\doCircleArc@=\doCircleArc@@}%
    \ellipseArc@ \xy@{doCircARCs}{\doCircleArcs@}\endarc@ }%
%   \xy@{doCircARCs}{\leave@ \doCircleArcs@}\endarc@ }%
   \xyFN@\getarcparams@#1@}\fi\fi\fi
  \next@ }
```

When further parsing is required, using `\getarcparams@`, then an `@` token is used to mark the end of the relevant tokens.

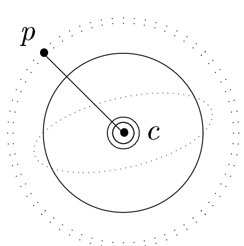
Before any parsing at all the current p and c are recorded as named ⟨pos⟩itions *0@p* and *0@c*, as well as a point *0@m* along the line in the current direction from p. A method `\origDirection@` is defined to artificially re-impose this direction independent of the p and c that may be current at the time. `\startDirection@` is initialised to this method, though parsing may change its value. Other methods are initialised, in particular the default orientation; their values can be expected to change with subsequent parsing.

```
\xydef@\arc@i{\xy@{ARCi}{%
  \save@\idfromc@{0@c}\cfromp@\idfromc@{0@p}%
  \enter@\cplusthec@ \enter@\relax\begingroup\vfromslide@i{}@\czeroEdge@
  \leave@\idfromc@{0@m}\leave@ \zeroDivideLimit@{50}%
  \edef\origDirection@{\Direction=\the\Direction
   \relax\noexpand\imposeDirection@}%
  \let\startDirection@=\origDirection@ \def\startDirection@@{}%
  \def\endDirection@{}\def\endDirection@@{}\def\arcOrient@@{^}}%
  \def\arcOrient@@{^}\let\processDirections@=\checkstartDirection@
  \def\arcRadius@{}\def\ellipseRatio@{}\def\arcScale@{\arcScale@@}}
```

```
\message{circles,}
```

3.9.1 Full Circles

The `xyarc` feature allows a much wider range of possibilities for typesetting circles than is available with `\cir`. Firstly the radius is no longer restricted to a finite collection of sizes. Secondly fancy line (curve) styles are available, as with curved arrows. Finally there are a variety of ways of specifying the desired radius, relative to other parts of the picture being built, as in the following example.




```
\xy 0;/r5pc/:*\dir{*}="p",*+!DR{p};
p+(.5,-.5)*\dir{*}="c",*+++!L{c}**\dir{-}
```

```
,{\ellipse<>{:}},{\ellipse(.5){}}
,0;(.5,.5)::,"p";"c",{\ellipse(.5){.}}
,{\ellipse<5pt>{=}}\endxy
```

The following give circles centred at c.

`\ellipse<>{`⟨style⟩`}`	radius = dist(p, c)
`\ellipse<`⟨dimen⟩`>{..}`	radius is the ⟨dimen⟩
`\ellipse(`⟨num⟩`){`⟨style⟩`}`	unit circle scaled ⟨num⟩, in the current basis.

Note that if the current basis is not square then the latter variant, namely `\ellipse(`⟨num⟩`)`, will typeset an ellipse rather than a circle. On the other hand the first two variants always specify true circles. In the 2nd case, i.e. when ⟨dimen⟩ is ⟨empty⟩, the size of the object at p is taken into account when drawing the circle; if this is not desired then kill the size using a null object, e.g. `;*{};`.

Currently the `\ellipse` macro works only as a ⟨decor⟩. In future versions there will be an ⟨object⟩ called `\arc` having elliptical shape, via `\circleEdge` with possibly unequal extents. Also it will be possible to `\connect\arc`, which will set the current connection so that any place on the full ellipse, not just the visible sector, will be accessible using an extension to the usual ⟨place⟩ mechanism.

To Do: make this be!!

The simplest cases of full circles go straight to `\circleArc@`.

```
\xydef@\fullCircle@{\xy@{fullCIRC}{\let\endDirection@=\relax
 \let\doCircleArc@=\doMergeEndArc@@}\circleArc@ }%
```

If the first non-space token following `\arc` is `<` then the current c is taken to be the centre of the circle or ellipse, or sector thereof. The first parsing is done to interpret information concerning the radius, (or axes of an ellipse) along with an orientation. This can be specified either as absolute dimensions, enclosed in `<`...`>`, or relative to the current basis by `(`...`)`, or based on the line $\overline{pc}$ using `<>` or `<,`⟨dimen⟩`>` or `(,`⟨num⟩`)`.

```
\xydef@\getarcparams@{%
 \ifx\space@\next \expandafter\DN@\space{\xyFN@\getarcparams@}%gobble spaces
 \else\addLT@\ifx \next \xy@{<}{}%
  \addGT@{\addLT@\DN@##1}{\splitRadius@dim##1,,@}%
% \else\ifx \next (\xy@{(}{}\DN@(##1){\splitRadius@##1,,@}%
 \else\ifx\next (\DN@({\xyFN@\checkRadOpen@}%
 \else\ifx\next @\xy@{@}{}\DN@ @{\fullEllipse@}%
 \else \addLQ@\ifx \next \addLQ@\DN@{\xy@`{}\ARCturn@}%
 \else \DN@{\getarcExtents@}%
 \fi\fi\fi\fi\fi \next@ }

\xydef@\checkRadOpen@{%
 \ifx\next *\DN@{\xyFN@\getarcExtents@ (}%
 \else \xy@{(}{}\DN@##1){\splitRadius@##1,,@}%
 \fi \next@ }
```

The centre is now stored as `"0@0"`, since the value of `"0@c"` will be changed to agree with `"0@p"`. This is so that start/finish directions will agree, ensuring all four quadrants be drawn. The following method is actually executed after all the radius/axes information has been parsed.

```
\xydef@\fixedRadiusArc@{%
 \xy@{fixRadArc}{\save@ \idfromc@{0@0}\cfromp@\idfromc@{0@c}\leave@}%
 \xyFN@\fixedRadiusArc@i }

\xydef@\fixedRadiusArc@i{%
```

```
\ifx\next\space@ \expandafter\DN@\space{\xyFN@\fixedRadiusArc@i}%
\else\ifx\next @\xy@{@}{}\DN@ @{\fullEllipse@}%
\else\DN@{\let\processDirections@=\processArcDirections@
 \def\afterARCextents@@{\xy@{aftARCext}{\let\doCircleArc@=\doCircleArc@@
   \ellipseSector@}}\xyFN@\getarcExtents@}%
\fi\fi \next@ }
```

When a full circle or ellipse is required then there are three methods which must be invoked when establishing the local basis change.

`\arcScale@` Initially this has value `\arcScale@@` which is based on $\overline{pc}$. It is changed only by `(`⟨num⟩`)` and `(`⟨num⟩`,`⟨num⟩`)`; see the code for `\splitRadius@` below. When executed, this method will establish the required basis, at least partially.

```
\xydef@\arcScale@@{\xy@{arcSCALE@@}{\save@
\cfromid@{0@0}\pfromc@\cfromid@{0@p}\setbase@\X@p\Y@p\X@c\Y@c}}
```

`\arcRadius@` Initially ⟨empty⟩, this changes with `<`⟨dimen⟩`,`⟨dimen⟩`>` and `<`⟨dimen⟩`>` to `\arcScale@dim` which gets the radius by a ⟨slide⟩ of given ⟨dimen⟩ along the line $\overline{pc}$.

```
\xydef@\arcScale@dim#1{\xy@{arcSCALEdim}{\save@
\cfromid@{0@0}\pfromc@ \cfromid@{0@p}\setupDirection@\cfromp@
\enter@\cplusthec@ \begingroup\vfromslide@i#1@%
\setbase@\X@p\Y@p\X@c\Y@c \idfromc@{0@p}}}
```

`\ellipseRatio@` Initially ⟨empty⟩, this changes with `<`⟨dimen⟩`,`⟨dimen⟩`>` to `\ellipseScale@dim` which establishs the (`\X@basey`, `\Y@basey`) by a ⟨slide⟩ perpendicular to the line $\overline{pc}$, or to `\ellipseScale@` with `(,`⟨num⟩`)` to simply set the perpendicular basis direction using ⟨num⟩ as a coordinate.

```
\xydef@\ellipseScale@dim#1{\xy@{ellSCALEdim}{\setupDirection@\cfromp@
\enter@\cplusthec@ \aboveDirection@\xydashl@ \vfromslide@{#1}\leave@
\setbase@@\X@c\Y@c}}
```

```
\xydef@\ellipseScale@#1{\xy@{ellSCALE@}%
{\vfromcartesian@@0,#1@\setbase@@\X@c\Y@c}}
```

The methods `\splitRadius@dim` and `\splitRadius@` complete the interpretation of the radius/axes specification. They must determine whether one or two pieces of information have been explicitly provided, and act accordingly.

```
\xydef@\splitRadius@dim#1,#2,#3@{%
\DN@{#1}\ifx\next@\empty\dimen@=\z@\else\dimen@=#1\relax%\splinetracing
 \expandafter\def\expandafter\arcRadius@
  \expandafter{\arcScale@dim{#1}}\fi
\DN@{#2}\ifx\next@\empty\dimen@ii=\z@\def\ellipseRatio@{}%
\else\dimen@ii=#2\relax \expandafter
 \def\expandafter\ellipseRatio@\expandafter{\ellipseScale@dim{#2}}%
\fi \fixedRadiusArc@ }
```

```
\xydef@\splitRadius@#1,#2,#3@{%
\DN@{#1}\ifx\next@\empty\dimen@=\z@\else\dimen@=#1\p@\relax\fi
\DN@{#2}\ifx\next@\empty\dimen@ii=\z@\else\dimen@ii=#2\p@\relax\fi
\ifdim\dimen@=\z@ \ifdim\dimen@ii=\z@\else \expandafter
 \def\expandafter\ellipseRatio@\expandafter{\ellipseScale@{#2}}\fi
```

```
 \else
  \ifdim\dimen@ii=\z@
   \def\arcScale@{\xy@{arcSCALE@}{\save@\cfromid@{0@0}%
    \X@origin=\X@c \Y@origin=\Y@c
    \X@xbase=#1\X@xbase \Y@xbase=#1\Y@xbase
    \X@ybase=#1\X@ybase \Y@ybase=#1\Y@ybase
    \advance\X@c\X@xbase \advance\Y@c\Y@xbase \czeroEdge@
    \idfromc@{0@p}\idfromc@{0@c}}}%
  \else
   \def\arcScale@{\xy@{arcSCALE@}{\save@\cfromid@{0@0}%
    \X@origin=\X@c \Y@origin=\Y@c
    \X@xbase=#1\X@xbase \Y@xbase=#1\Y@xbase
    \X@ybase=#2\X@ybase \Y@ybase=#2\Y@ybase
    \advance\X@c\X@xbase \advance\Y@c\Y@xbase \czeroEdge@
    \idfromc@{0@p}\idfromc@{0@c}}}%
  \fi
 \fi \fixedRadiusArc@ }
```

```
\xydef@\baseOrient@{\DN@{_}\ifx\next@\arcOrient@@
 \X@ybase=-\X@ybase \Y@ybase=-\Y@ybase \fi }
```

```
\message{ellipses,}
```

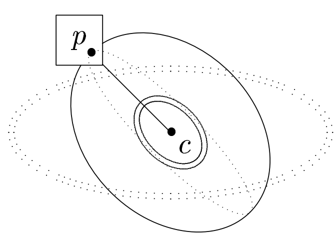
3.9.2 Ellipses

There are several ways to specify an ellipse, apart from the method illustrated above in which the basis must be changed from square. Basically we must specify the lengths of the major and minor axes. Also it is necessary to specify an alignment for one axis.

In the following, the ellipse is centred on c and one axis is aligned along the line $\overline{pc}$, except with the final variant where it aligns with the current basis. When used ⟨num⟩ is treated as a scale factor, multiplying an appropriate length.

`\ellipse<`⟨dimen⟩`,`⟨dimen⟩`>{`..`}`	given axes lengths
`\ellipse<,`⟨dimen⟩`>{`⟨style⟩`}`	one axis is $\overline{pc}$
`\ellipse(,`⟨num⟩`){`⟨style⟩`}`	...perp. axis scaled
`\ellipse(`⟨num⟩`,`⟨num⟩`){`..`}`	scaled axes aligned with basis.

In the latter variant, if the second ⟨num⟩ is ⟨empty⟩ then this is equivalent to both ⟨num⟩s having the same value, which is in turn equivalent to the final variant for circles.




```
\xy 0;/r5pc/:*\dir{*},*++!DR(.5){p}
*\frm{-};p+(.5,-.5)*\dir{*}="c",
```

```
**\dir{-},**+!UL{c},"c",
,{\ellipse(1,.4){:}},{\ellipse(,.75){}}
,{\ellipse<15pt,10pt>{=}}
;*{};{\ellipse<,10pt>{.}}\endxy
```

As with full circles, full ellipses require little preparation.

```
\xydef@\fullEllipse@{\xy@{fullELLIPSE}%
 {\def\startDirection@{}\let\doCircleArc@=\doMergeEndArc@@}%
 \ellipseArc@
 \xy@{doCIRCarcs}{\leave@ \doCircleArcs@}\endarc@ }
```

3.9.3 Drawing arcs

This method is common to full circles, ellipses and circular arcs. It first executes the methods `\arcScale@`, `\arcRadius@`, `\ellipseRatio@` where necessary, then examines `\arcOrient@@` to decide if the arc is to be typeset clockwise or counter-clockwise.

```
\xydef@\ellipseArc@{%
 \arcScale@ \arcRadius@ \ellipseRatio@
 \xy@{arcCONTROLS}{\baseOrient@ \arcControls@ \let\bstartPLACE@=\relax }}
```

Here is where the "control points" are defined, for each of the four Bézier cubic segments, using coordinates in the constructed basis. The endpoints of the segments—though, not necessarily of the arc—are stored as *0p*, *1c*, *2c*, *3c* and the tangential control points are stored as *1m*, *2m*, ..., *8m*.

```
\xydef@\cubicARCcontrols@@{\czeroEdge@
 \vfromcartesian@@0,1@\idfromc@{1@c}%
 \vfromcartesian@@-1,0@\idfromc@{2@c}%
 \vfromcartesian@@0,-1@\idfromc@{3@c}%
 \vfromcartesian@@1,\arcparam@ @\idfromc@{1@m}%
 \vfromcartesian@@\arcparam@,1@\idfromc@{2@m}%
 \vfromcartesian@@-\arcparam@,1@\idfromc@{3@m}%
 \vfromcartesian@@-1,\arcparam@ @\idfromc@{4@m}%
 \vfromcartesian@@-1,-\arcparam@ @\idfromc@{5@m}%
 \vfromcartesian@@-\arcparam@,-1@\idfromc@{6@m}%
 \vfromcartesian@@\arcparam@,-1@\idfromc@{7@m}%
 \vfromcartesian@@1,-\arcparam@ @\idfromc@{8@m}%
 }
```

Drawing the circle/ellipse/arc means tracing along the requisite cubic segments. The method `\doCircleArc@@` is used for full circles/ellipses when complete segments are to be typeset. The methods `\doMergeStartArc@`, `\doMergeEndArc@@` and `\doMergeBothArc@` are used when a segment is not complete. The work by "merging" the initial/final point of the curved segment with the appropriate point required to be at the extremity of the arc. Typesetting of the curve should not take place within the region covered by this merged object.

To overcome the difficulty of the curve leaving then re-entering this region, a second merge is performed to extend it suitably. This is the purpose of `\cfromcontrols@`. Due to the "convex-hull property" of the Bézier cubic segment, a point can be constructed from the end-points and control points which can be used to extend the rectangle to cover a suitable portion of the curve.

```
\xydef@\doCircleArc@@#1#2{\save@ \ifcase#2\relax
 \doCircleArc@@@{0@p}{1@m}{2@m}{1@c}{#1}\or
 \doCircleArc@@@{1@c}{3@m}{4@m}{2@c}{#1}\or
```

```
 \doCircleArc@@@{2@c}{5@m}{6@m}{3@c}{#1}\or
 \doCircleArc@@@{3@c}{7@m}{8@m}{0@p}{#1}\or
 \doCircleArc@@@{0@p}{1@m}{2@m}{1@c}{#1}\or
 \doCircleArc@@@{1@c}{3@m}{4@m}{2@c}{#1}\or
 \doCircleArc@@@{2@c}{5@m}{6@m}{3@c}{#1}\or
 \doCircleArc@@@{3@c}{7@m}{8@m}{0@p}{#1}\fi \leave@ }

\xydef@\doCircleArc@@@#1#2#3#4#5{%
 \cfromid@{#1}\pfromc@\senter@
 \cfromid@{#2}\spushc@\cfromid@{#3}\spushc@\cfromid@{#4}%
 \connect@\crvs{#5}\relax\sinit@\sleave@}

\xydef@\doMergeEndArc@@#1#2{\save@\ifcase#2\relax
 \doMergeEnd@@@{0@p}{1@m}{2@m}{1@c}{#1}\relax\or
 \doMergeEnd@@@{1@c}{3@m}{4@m}{2@c}{#1}\relax\or
 \doMergeEnd@@@{2@c}{5@m}{6@m}{3@c}{#1}\relax\or
 \doMergeEnd@@@{3@c}{7@m}{8@m}{0@p}{#1}@\fi \leave@ }

\xydef@\doMergeEnd@@@#1#2#3#4#5#6{%
 \cfromid@{#1}\pfromc@\senter@
 \cfromid@{#2}\spushc@\cfromid@{#3}\spushc@\cfromid@{#4}%
 \ifx#6\relax
  \L@c=.1\p@ \R@c=\L@c \U@c=\L@c \D@c=\L@c \Edge@c={\rectangleEdge}\fi
 \enter@\cmergethec@\cfromid@{0@c}\leave@
 \enter@\cmergethec@\cfromcontrols@{#1}{#2}{#3}{#4}\leave@
 \connect@\crvs{#5}\relax
 \sinit@\sleave@}

\xydef@\doMergeStartArc@#1#2{\save@\ifcase#2\relax
 \doMergeStart@@{0@p}{1@m}{2@m}{1@c}{#1}@\or
 \doMergeStart@@{1@c}{3@m}{4@m}{2@c}{#1}\relax\or
 \doMergeStart@@{2@c}{5@m}{6@m}{3@c}{#1}\relax\or
 \doMergeStart@@{3@c}{7@m}{8@m}{0@p}{#1}\relax\fi \leave@ }

\xydef@\doMergeStart@@#1#2#3#4#5#6{%
 \cfromid@{#1}\ifx#6\relax
  \L@c=.1\p@ \R@c=\L@c \U@c=\L@c \D@c=\L@c \Edge@c={\rectangleEdge}\fi
 \enter@\cmergethec@\cfromid@{0@x}\leave@
 \enter@\cmergethec@\cfromcontrols@{#4}{#3}{#2}{#1}\leave@
 \pfromc@
 \senter@ \cfromid@{#2}\spushc@\cfromid@{#3}\spushc@\cfromid@{#4}%
 \connect@\crvs{#5}\relax\sinit@\sleave@}

\xydef@\doMergeBothArc@#1#2{\save@\ifcase#2\relax
 \doMergeBoth@@{0@p}{1@m}{2@m}{1@c}{#1}@\or
 \doMergeBoth@@{1@c}{3@m}{4@m}{2@c}{#1}\relax\or
 \doMergeBoth@@{2@c}{5@m}{6@m}{3@c}{#1}\relax\or
 \doMergeBoth@@{3@c}{7@m}{8@m}{0@p}{#1}\relax\fi \leave@ }

\xydef@\doMergeBoth@@#1#2#3#4#5#6{%
 \cfromid@{#1}\ifx#6\relax
  \L@c=.1\p@ \R@c=\L@c \U@c=\L@c \D@c=\L@c \Edge@c={\rectangleEdge}\fi
 \enter@\cmergethec@\cfromid@{0@x}\leave@
 \enter@\cmergethec@\cfromcontrols@{#4}{#3}{#2}{#1}\leave@
```

```
 \pfromc@
 \senter@ \cfromid@{#2}\spushc@\cfromid@{#3}\spushc@\cfromid@{#4}%
 \enter@\cmergethec@\cfromid@{0@c}\leave@
 \enter@\cmergethec@\cfromcontrols@{#1}{#2}{#3}{#4}\leave@
 \connect@\crvs{#5}\relax\sinit@\sleave@}
```

```
\xydef@\doMergeEndArc@#1#2{\count@@=#2\relax
 \ifnum\count@@>\thr@@ \advance\count@@-4\fi\relax
 \doMergeEndArc@@{#1}\count@@ }
```

```
\xydef@\doStraightArc@@#1{\xy@{STRAITarc}{\save@
 \cfromid@{0@p}\pfromc@\cfromid@{0@c}%
 \senter@\connect@\crvs{#1}\sinit@\sleave@\leave@}}
```

```
\xydef@\cfromcontrols@#1#2#3#4{\bgroup
  \cfromid@{#1}\L@p=\X@c \U@p=\Y@c
  \cfromid@{#4}\R@p=\X@c \D@p=\Y@c
  \cfromid@{#3}\A@=\X@c \B@=\Y@c \cfromid@{#2}%
  \ifdim\R@p>\L@p \ifdim\A@>\R@p \R@p=\A@
    \ifdim\X@c>\R@p \R@p=\X@c\fi\fi
  \else \ifdim\A@<\R@p \R@p=\A@
    \ifdim\X@c<\R@p \R@p=\X@c\fi\fi
  \fi
  \ifdim\D@p>\U@p \ifdim\B@>\D@p \D@p=\B@
    \ifdim\Y@p>\D@p \D@p=\Y@p\fi\fi
  \else \ifdim\B@<\D@p \D@p=\B@
    \ifdim\Y@p<\D@p \D@p=\Y@p\fi\fi
  \fi
  \edef\next@{\egroup \X@c=\the\R@p \Y@c=\the\D@p}\next@ }
```

The next method controls which segments are typeset. It is rather primitive and could well be improved; e.g. to be more like `\doEllipseSectors@` with `\forwardSectors@` and `\backwardSectors@`.

First it calculates the `\startDirection@@` as a number, using the current base. Next it calculates `\quadDirection@` which records the angle of the y-axis in the current base. Depending on the required orientation, read from `\arcOrient@@`, this latter number is adjusted to be minimal greater than `\startDirection@@` for a counter-clockwise arc, or maximal less than it for a clockwise one. Similarly the value of `\endDirection@@` is made larger or smaller than `\startDirection@@`, according to orientation, by adding ± 8192 if necessary.

Each required segment is typeset using `\doCircleArc@@` except that the final segment uses the value of `\doCircleArc@`. When the end-point is the original c then this value is `\doMergeEndArc@@`, so that the arc will stop at the edge of this ⟨pos⟩, otherwise `\doCircleArc@@` is used.

(The following macro can be shortened by making use of the methods `\forwardSectors@` and `\backwardSectors@`, as in `\doEllipseSectors@`.

```
\xydef@\doCircleArcs@{\save@
 \cfromid@{0@0}\pfromc@\cfromid@{0@p}\setupDirection@
 \startDirection@ \edef\startDirection@@{\the\Direction}%
 \cfromid@{1@c}\setupDirection@ \edef\quadDirection@{\the\Direction}\relax
 \count@=\quadDirection@ \relax\def\nextii@{^}%
 \let\preDirection@@=\startDirection@@
 \ifx\nextii@\arcOrient@@
  \ifnum\quadDirection@>\startDirection@@ \relax
  \else \advance\count@ by8192 \relax\edef\quadDirection@{\the\count@}\fi
```

```
  \ifx\endDirection@@\empty
   \ifx\endDirection@\relax
    \bgroup\cfromid@{0@c}\setupDirection@
     \edef\next@{\egroup \count@@=\the\Direction}\next@ \relax
    \ifnum\count@@<\startDirection@@ \advance\count@@ 8192 \fi
    \edef\endDirection@@{\the\count@@}%
   \else
    \count@@=\startDirection@@
    \advance\count@@ by8192 \edef\endDirection@@{\the\count@@}%
   \fi
  \else
   \count@@=\endDirection@@ \relax\ifnum\count@@<\startDirection@@ \relax
    \advance\count@@ by8192 \edef\endDirection@@{\the\count@@}\fi
  \fi % \forwardSectors@
 \ifnum\endDirection@@>\quadDirection@
  \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{0}}%
 \else
  \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{0}}%
 \fi
%
 \else % \arcOrient@@ = _
%
  \ifnum\count@<\startDirection@@\relax\else\advance\count@ by-8192 \relax
  \edef\quadDirection@{\the\count@}\fi
  \ifx\endDirection@@\empty
   \ifx\endDirection@\relax
    \bgroup\cfromid@{0@c}\setupDirection@
     \edef\next@{\egroup \count@@=\the\Direction}\next@ \relax
    \ifnum\count@@>\startDirection@@ \advance\count@@-8192 \fi
    \edef\endDirection@@{\the\count@@}%
   \else
    \count@@=\startDirection@@
    \advance\count@@ by-8192 \edef\endDirection@@{\the\count@@}%
   \fi
  \else
   \count@@=\endDirection@@ \relax \ifnum\count@@>\startDirection@@
    \advance\count@@ by-8192 \edef\endDirection@@{\the\count@@}\fi
  \fi %\backwardSectors@
  \ifnum\endDirection@@<\quadDirection@
   \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{0}}%
  \else \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{0}}\fi
 \fi
 \next@ \relax
%
 \count@=\quadDirection@ \relax\def\nextii@{^}\DN@{}%
 \ifx\nextii@\arcOrient@@
  \count@@=\startDirection@@ \advance\count@@ by4096 \relax
 \ifnum \count@<\endDirection@@ \relax
 \ifnum\endDirection@@>\count@@
```

```
  \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}%
 \else \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{1}}\fi
 \fi
 \else
  \count@@=\startDirection@@ \advance\count@@ by-4096 \relax
 \ifnum \count@>\endDirection@@ \relax
 \ifnum\endDirection@@<\count@@
  \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}%
 \else \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{1}}\fi
 \fi\fi
 \next@ \relax
%
 \ifx\next@\empty \else \relax
 \count@=\startDirection@@ \relax\def\nextii@{^}\DN@{}%
 \ifx\nextii@\arcOrient@@ \advance\count@ by4096 \relax
  \count@@=\quadDirection@ \advance\count@@ by4096 \relax
 \ifnum \count@<\endDirection@@
 \ifnum\endDirection@@>\count@@
  \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{2}}%
 \else \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{2}}\fi
 \fi
 \else \advance\count@ by-4096 \relax
  \count@@=\quadDirection@ \advance\count@@ by-4096 \relax
 \ifnum \count@>\endDirection@@ \relax
 \ifnum\endDirection@@<\count@@
  \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{2}}%
 \else \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{2}}\fi
 \fi\fi
 \next@ \relax
 \fi
%
 \ifx\next@\empty \else \relax
 \count@=\quadDirection@ \def\nextii@{^}\DN@{}%
 \ifx\nextii@\arcOrient@@ \advance\count@ by4096 \relax
 \ifnum \count@<\endDirection@@
 \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{3}}\fi
 \else \advance\count@ by-4096 \relax
 \ifnum \count@>\endDirection@@
 \DN@{\expandafter\doCircleArc@\expandafter{\arcSTYLE}{3}}\fi
 \fi \next@ \relax
 \fi
%
 \leave@ }
```

```
\message{elliptical arcs,}
```

3.9.4 Circular and Elliptical Arcs

The xyarc feature handles arcs to be specified in two essentially different ways, according to what information is provided by the user. We call these the “radius-unknown/end-points known” and the "radius-known/end-points unknown" cases.

radius unknown, end-points known

The simplest case, though not necessarily the most common, is that of a circular arc from p to c, with radius and centre unspecified. To uniquely specify the arc, the tangent direction at p is taken to be along the current direction, given by `\Direction`, as set by the latest ⟨connect⟩ion. If no connection has been used, then the default ⟨direction⟩ is "up".

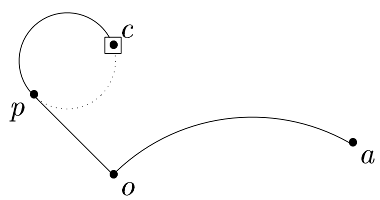
`\ellipse_{`⟨style⟩`}`	clockwise arc from p to c
`\ellipse^{`⟨style⟩`}`	counter-clockwise arc
`\ellipse{`⟨style⟩`}`	also counter-clockwise

With this information only, a unique circle can be found whose radius and centre need not be specified in advance. For a unique arc it is sufficient to specify the orientation around the circle.

The exception is when the current direction is from p to c, in which case no circle exists. Instead a straight line is typeset accompanied by the following message:

```
Xy-pic Warning: straight arc encountered
```

The following example shows how, given three points o, p and c, to continue the line $\overline{op}$ by a circular arc to c joining smoothly at p.




```
\xy 0;/r5pc/:*=+\dir{*}*+!UR{p};
p+(.5,-.5)*\dir{*}="o",*+!UL{o}
,+(0,.81)*=<6.1pt>\dir{*}*\frm{-}="c"
,*+!DL{c},"o",**\dir{-},
"c",{\ellipse_{}},{\ellipse^{.}}
%
,"o"+(1.5,.2)*\dir{*}="a"*+!UL{a}
,"o";p+/_1pc/,**{},"a",{\ellipse_{}}
\endxy
```

Note how the remainder of the circle can be specified separately. The example also shows how to specify an arc which leaves a particular point perpendicular to a specific direction.

Slightly more complicated is when the tangent direction at p is specified, but different from the current direction; a unique circular arc can still be defined. More complicated is when a specific tangent direction is required also at c. In this case the arc produced is a segment of an ellipse. (If the required tangent at p points to c then a straight segment is drawn, as in the circular case described above.)

`\ellipse`⟨dir⟩$_p$`,`⟨orient⟩`{..}`	circular
`\ellipse`⟨dir⟩$_p$`,`⟨orient⟩`,`⟨dir⟩$_c$`{..}`	elliptical
`\ellipse`⟨dir⟩$_p$`,`⟨orient⟩⟨dir⟩$_c$`{..}`	elliptical
`\ellipse`⟨dir⟩$_p$`,`⟨orient⟩`,=`⟨dir⟩$_c$`{..}`	elliptical

`\ellipse`‘⟨coord⟩⟨orient⟩`{..}`	elliptical

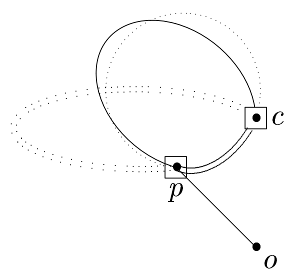
In these cases $\langle\text{dir}\rangle_p$ and $\langle\text{dir}\rangle_c$ are ⟨direction⟩ specifications, as in figure 1.5 and note 1.4l, and ⟨orient⟩ must be either `^` or `_` for anti-/clockwise respectively, defaulting to `^` if ⟨empty⟩. Beware that the `(*`⟨pos⟩⟨decor⟩`*)` form *must* be used for this ⟨direction⟩ variant, as if an object modifier.

The second and third cases in the above table generally give identical results. The second ‘`,`’ is thus optional, except in two specific situations:

1. ⟨orient⟩ is empty and $\langle\text{dir}\rangle_c$ has `^` or `_` as the first token;

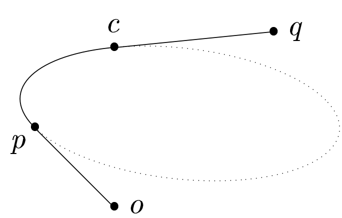
2. ⟨orient⟩ is `^` and $\langle\text{dir}\rangle_c$ has `^` as first token. Without the `,` then `^^` would be interpreted by TEX as part of a special ligature for a hexadecimal character code.

If both ⟨orient⟩ and $\langle\text{dir}\rangle_c$ are ⟨empty⟩ then even the first ‘`,`’ can be omitted.




```
\xy 0;/r5pc/:*=<8.1pt>\dir{*}="p",*\frm{-}
,*++!U{p},"p";p+(.5,-.5)*+\dir{*}="o"
,*+!UL{o},+(0,.81)*=<8.1pt>\dir{*}="c"
,*\frm{-},*++!L{c},"o"**\dir{-},"c"
,{\ellipse :a(50),_:0{:}}
,{\ellipse :a(30),_:a(-45){}}
,{\ellipse :a(40),_{.}},
;*{};{\ellipse :a(20),^=_{=}}\endxy
```

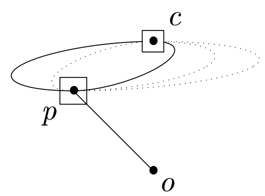
Note that only the slope of $\langle\text{dir}\rangle_p$ and $\langle\text{dir}\rangle_c$ is significant; rotations by 180° being immaterial.




```
\xy 0;/r5pc/:*\dir{*}="p",*+!UR{p}
;p+(.5,-.5)*\dir{*}="o",*++!L{o}**\dir{-}
,p+(.5,.5)*\dir{*}="c",*++!D{c},"c"
;p+(1,.1)*\dir{*}="q",*++!L{q}**\dir{-}
,"o";"p",**{};"c"
,{\ellipse![["o";"p"]],_![["q";"c"]]{}}
,{\ellipse![["o";"p"]],![["c";"q"]]{.}}
\endxy
```

The `=` variant establishes the ⟨direction⟩ parsing to begin with the direction resulting from $\langle\text{dir}\rangle_p$ instead of the original direction. If $\langle\text{dir}\rangle_c$ is required to be the original direction then use `:0`. It cannot be ⟨empty⟩ since this is interpreted as requiring a circular arc with unspecified tangent at c; see the example above. However when $\langle\text{dir}\rangle_p$ and $\langle\text{dir}\rangle_c$ are parallel there is a whole family[4] of possible ellipses with the specified tangents.

With no further hint available, a choice is made based on the distance between p and c. If the required direction is perpendicular to $\overline{pc}$ this choice results in a circular arc. The optional factor in `=(`⟨num⟩`)` is used to alter this choice; the default `(1)` is assumed when nothing follows the `=`. This factor is used to "stretch" the ellipse along the specified direction. For a negative ⟨num⟩ the orientation reverses.




```
\xy ;/r5pc/:*+=<10.1pt>\dir{*}="p";p*\frm{-}
,*++!UR{p},p+(.5,-.5)*\dir{*}="o",**\dir{-}
,*+!UL{o},+(0,.81)*=<8.1pt>\dir{*}="c"
,*\frm{-},*++!DL{c},"c"
,{\ellipse r,={}},{\ellipse r,=(2){.}}
,{\ellipse r,^=(3){.}},{\ellipse r,=(-2){}}
,{\ellipse r,=(-1){.}}\endxy
```

The final variant uses the directions from p and c to the given ⟨coord⟩. If ⟨orient⟩ is ⟨empty⟩ then the orientation is determined to give the shortest path along the ellipse. Specifying an ⟨orient⟩ of `^` or `_` will force the orientation, even if this means travelling 'the long way' around the ellipse. For example, see next figure.

Alternative curves In some cases the circular or elliptic curve can be replaced by a curve with different shape, having the same tangent directions at the end-points. When a full circle/ellipse is specified then one gets instead a closed curve constructed from 4 spline segments. Other variants use a single segment, 2 or 3 segments, or some portion of all 4 segments. Possibilities are given in the following table.

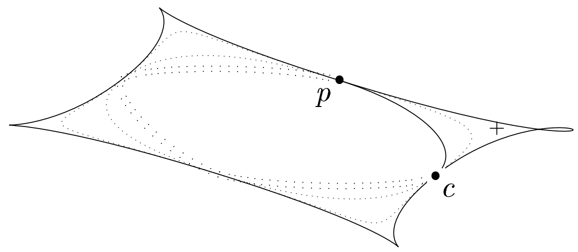
`\ellipse~e ...{`⟨..⟩`}`	elliptical, as above
`\ellipse~q ...{`⟨..⟩`}`	parabolic segments
`\ellipse~c ...{`⟨..⟩`}`	cubic segments
`\ellipse~i ...{`⟨..⟩`}`	interpolating cubic
`\ellipse~p ...{`⟨..⟩`}`	cuspidal cubic
`\ellipse~c(`⟨num⟩`)...{`⟨..⟩`}`	cubic segments, with "looseness"

In the latter case the ⟨num⟩, typically between 0 and 1, controls how soon the curve begins to bend away from the tangent direction. Smaller values give tighter curves — 0 for straight lines — with `~c` being the same as `~c(1)` and `~q` is `~c(.66667)`, that is ⟨num⟩$=\frac{2}{3}$.

The curve produced by the "interpolating" variant `~i` actually passes through the control point `"x"`, with slope parallel to the line $\overline{pc}$. Since the tangents at p and c point toward `"x"` the curvature is quite gentle until near `"x"` where the curve bends rapidly, yet smoothly. This is obtained also by using `~c(1.33333)`, that is ⟨num⟩$=\frac{4}{3}$. Since `<num>` > 1 the "convex hull property" does not hold; indeed the curve is entirely outside the convex hull of p, c and `"x"`, apart from those points themselves.

[4]Indeed this is always so. The algorithm used for the general case tends toward parallel lines—clearly unsuitable.

The 'cuspidal' variant `~p` is equivalent to `~c(2)`. It exhibits a cusp. For `<num>` > 2 the curve is so "loose" that it exhibits loops. (The author offers no guarantees on the usefulness of such curves for any particular purpose; however they do look nice. ☺)




```
\xy 0;/r6pc/:*+\dir{*}="p",*+!UR{p},"p";
p+(.5,-.5)*+\dir{*}="c",*+!UL{c}
,"p"+(.825,-.25)="x"*\dir{+},"c"
,{\xycompile{\ellipse`"x"{-}}}
,{\xycompile{\ellipse~q`"x"^{.}}}
,{\xycompile{\ellipse~c`"x"{.}}}
,{\xycompile{\ellipse~c(.3)`"x"^{:}}}
,{\xycompile{\ellipse~c(2.3)`"x"{-}}}
,{\xycompile{\ellipse~i`"x"^{.}}}
,{\xycompile{\ellipse~p`"x"^{-}}}
\endxy
```

Hint: When exploring to find the best location for the "control-point" (e.g. the `"x"` in the above example), then use `\xycompile` as shown, changing the location outside of the compilation. This speeds up the reprocessing with the changed value.

Furthermore, the ellipses are not true ellipses, but are constructed from four Bézier cubic curves, one for each quadrant of the circle or ellipse. To do this it is sufficient to establish the control points for each cubic segment. This is straightforward, using the following "magic number", given a square basis such that the desired circle is the unit circle, or rectangular basis for which the "unit circle" is the desired ellipse. Thus `~e` is equivalent to `~c(.5517847)`.

```
\xydef@\arcmagic@@{0.5517847}
\xylet@\arcmagic=\arcmagic@@
```

This is the magic number, exactly given by $\frac{1}{12}(\sqrt{385} - 13)$, that helps construct the Bézier cubic curve that best approximates a quarter-turn arc of a circle. It does so with remarkable accuracy, differing by at most .5% of the radius at any angle; the average deviation along the whole quadrant being less than .13%.

```
\xydef@\tildeARC@#1{%
 \if q#1\DN@{\xy@{quadARC}\quadARC@ \arc@}%
 \else\if c#1\DN@{\xyFN@\cubicARC@i}%
 \else\if e#1\DN@{\xy@{ellipticARC}\ellipticARC@ \arc@}%
 \else\if i#1\DN@{\xy@{interpARC}\interpARC@ \arc@}%
 \else\if p#1\DN@{\xy@{cuspidalARC}\cuspidalARC@ \arc@}%
 \else\xywarning@{unknown arc type: #1 }%
  \DN@{\xy@{ellipticARC}\ellipticARC@ \arc@}%
 \fi\fi\fi\fi\fi \next@ }
```

```
\xydef@\cubicARC@i{%
```

```
  \ifx\next\space@ \expandafter\DN@\space{\xyFN@\cubicARC@i}%
  \else\ifx\next (\DN@(##1){\xy@{cubicARC(##1)}{\cubicARC@{##1}}\arc@}%
  \else \DN@{\xy@{cubicARC(1)}{\cubicARC@{1}}\arc@}%
  \fi\fi \next@ }

\xydef@\ellipticARC@{\let\arcControls@=\cubicARCcontrols@@
  \edef\arcparam@{\arcmagic@@}}

\xydef@\cubicARC@#1{\let\arcControls@=\cubicARCcontrols@@
  \dimen@=#1\p@ \edef\arcparam@{\expandafter\removePT@\the\dimen@}}

\xydef@\quadARC@{\let\arcControls@=\cubicARCcontrols@@
  \def\arcparam@{.66667}}

\xydef@\cuspidalARC@{\let\arcControls@=\cubicARCcontrols@@
  \def\arcparam@{2}}

\xydef@\interpARC@{\let\arcControls@=\cubicARCcontrols@@
  \def\arcparam@{1.33333}}
```

Avoiding overflows If $\langle\text{dir}\rangle_p$ and $\langle\text{dir}\rangle_c$ are intended to be equal then the method of the previous paragraph should be used. However it may happen that "nearly parallel" directions may be specified, perhaps by accident. There is then the possibility of "numerical overflow" or a "division by zero" error. The latter may be accompanied by a warning message:

```
Xy-pic Warning: division by 0 in
     \intersect@, replaced by 50
```

This indicates that the number 50 has been used as the result of a division by zero. In many contexts this will produce an acceptable result. However it may lead to an "overflow" in other situations, or to drawing beyond the normal page boundary. This can be controlled using a ⟨decor⟩ of type `,{\zeroDivideLimit{`⟨num⟩`}},` prior to specifying the `\ellipse`. The value 50 will be replaced by ⟨num⟩ whenever a "division by zero" would otherwise be encountered in an intersection calculation.

```
\xydef@\ARCturn@{\xy@@\bgroup\afterCOORD{%
   \xy@@{\edef\next@{\egroup \A@=\the\X@c \B@=\the\Y@c}\next@}\ARCturn@@}}

\xydef@\ARCturn@@#1@{\DN@{#1}\DNii@{^}%
  \ifx\next@\nextii@ \xy@{^}{\def\arcOrient@@{^}}%
  \else\DNii@{_}\ifx\next@\nextii@ \xy@{_}{\def\arcOrient@@{_}}%
   \else\xy@{}{\def\arcOrient@@{}}\fi\fi
  \xy@@\ARCturn@@@ \processDirections@@ }

\xydef@\ARCturn@@@{%
  \bgroup\cfromid@{0@p}\X@p=\X@c \Y@p=\Y@c \X@c=\A@ \Y@c=\B@
   \setupDirection@ \edef\next@{\egroup\count@=\the\Direction}\next@\relax
   \edef\startDirection@@{\the\count@ }%
  \bgroup\cfromid@{0@c}\X@p=\A@ \Y@p=\B@
   \setupDirection@ \edef\next@{\egroup\count@=\the\Direction}\next@\relax
  \edef\endDirection@@{\the\count@}%
  \ifx\arcOrient@@\empty
   \bgroup\cfromid@{0@p}\X@p=\X@c \Y@p=\Y@c \cfromid@{0@c}%
    \setupDirection@ \edef\next@{\egroup\count@@=\the\Direction}\next@\relax
   \count@=-\count@@ \advance\count@\startDirection@@ \relax \count@=-\count@
```

```
 \ifnum\count@>4096 \def\arcOrient@@{_}%
 \else\ifnum\count@<-4096 \def\arcOrient@@{^}%
 \else\ifnum\count@>\z@ \def\arcOrient@@{^}%
 \else\def\arcOrient@@{_}\fi\fi\fi
\fi }
```

```
\xydef@\circleArc@{%
 \xy@{circARC@}{\cfromid@{0@p}\pfromc@\cfromp@\startDirection@
  \arcCenter@\idfromc@{0@0}%
  \def\startDirection@{\bgroup
    \cfromid@{0@0}\pfromc@\cfromid@{0@p}\setupDirection@
    \edef\next@{\egroup \Direction=\the\Direction}\next@
   \relax\imposeDirection@}}%
 \ifx\endDirection@@\empty
  \ifx\arcRadius@\empty \ifx\ellipseRatio@\empty\else
   \DN@{\arcScale@@}\ifx\arcScale@\next@\else
   \xy@@{\bgroup\cfromid@{0@0}\pfromc@\cfromid@{0@c}\setupDirection@
     \edef\next@{\egroup \Direction=\the\Direction}\next@
   \edef\endDirection@@{\the\Direction}}\fi\fi\fi\fi
 \ellipseArc@ \xy@@{\leave@ \doCircleArcs@}\endarc@ }
% \ellipseArc@ \xy@@{\doCircleArcs@}\endarc@ }
```

The following methods calculate the centre of a circle using the perpendicular to the required angle at p and the perpendicular bisector of the line $\overline{pc}$.

```
\xydef@\arcbisector{%
 \cfromid@{0@p}\pfromc@\cfromid@{0@c}\setupDirection@
 \advance\X@c\X@p\divide\X@c\tw@
 \advance\Y@c\Y@p\divide\Y@c\tw@ \czeroEdge@ \pfromc@
 \enter@\cplusthec@ \enter@\relax\begingroup
  \aboveDirection@\xydashl@ \vfromslide@i{}@\leave@ }
```

```
\xydef@\arcCenter@{%
 \cfromid@{0@p}\pfromc@\cfromid@{0@m}\setupDirection@
 \cfromp@\enter@\cplusthec@ \enter@\relax\begingroup
  \aboveDirection@\xydashl@ \vfromslide@i{}@\leave@\no@@
 \arcCenter@i }
```

```
\xydef@\arcCenter@i{%
 \enter@{\pfromthep@}\setupDirection@ \Creset@@
  \let\PLACEedgep@@=\PLACEedgep@ \let\PLACEedgec@@=\PLACEedgec@
  \enter@{\pfromthep@\basefromthebase@}\bgroup
   \arcbisector \edef\next@{\egroup
     \X@c =\the\X@c \Y@c=\the\Y@c \X@p=\the\X@p \Y@p=\the\Y@p }%
     \next@ \Cintercept@@ \leave@\leave@ }
```

radius known, end-points unknown

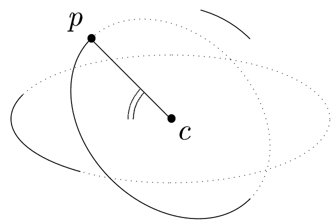
The language for these is a combination of most of that used above, but the interpretation of the $\langle\text{direction}\rangle$s is different...

$$\texttt{\\ellipse<}\langle\text{radius}\rangle\texttt{>}\langle\text{dir}\rangle_1\texttt{,}\langle\text{orient}\rangle\texttt{,}\langle\text{dir}\rangle_2\texttt{\{..\}}$$

`\ellipse<`$\langle\text{radius}\rangle$`>`$\langle\text{dir}\rangle_1$`,`$\langle\text{orient}\rangle$`,=`$\langle\text{dir}\rangle_2$`{..}`

---

where $\langle\text{radius}\rangle$ is one of the forms used above to describe a circle or ellipse. Not all of the ellipse will be typeset—only that arc starting with $\langle\text{dir}\rangle_1$ as tangent vector, tracing via $\langle\text{orient}\rangle$ until the tangent points in direction $\langle\text{dir}\rangle_2$. This effectively extends the notation used with `\cir` in 1.6.2. Note that rotating a given $\langle\text{dir}\rangle_i$ by 180° specifies a different arc on the same ellipse/circle. Reversing the $\langle\text{orient}\rangle$ no longer gives the complementary arc, but this complement rotated 180°.




```
\xy 0;/r5pc/:*\dir{*}="p",*+!DR{p};
p+(.5,-.5)*\dir{*}="c",*+!UL{c}**\dir{-}
,"c",{\ellipse<15pt>_,=:a(45){=}}
  ,{\ellipse<>__,=:a(30){-}}
,{\ellipse(1,.4){.}}
  ,{\ellipse(1,.4)_,=:a(120){-}}
,{\ellipse(,.75){.}}
  ,{\ellipse(,.75)_,^,^{-}}\endxy
```

```
\xydef@\getarcExtents@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\getarcExtents@}%
 \else\ifx\next @\xy@{@}{}\DN@ @{\fullEllipse@}%
 \else\DN@{\xy@{arcEXTENTs}{\save@
   \def\startDirection@{\Direction=\startDirection@@ \imposeDirection@}}%
  \xyFN@\getarcExtents@ii}\fi\fi \next@ }

\xydef@\getarcExtents@ii{%
 \ifx\next!\xy@{![[}{}\DN@![[##1]]{\getxyDirection@{##1}\getarcExtents@iv}%
 \else\DN@{\getarcExtents@iv}\fi \next@ }

\xydef@\getarcExtents@iv{\afterDIRECTIONorEMPTY
  {\xy@{startDIR}{\edef\startDirection@@{\the\Direction}\leave@}%
   \xyFN@\getarcOrientation@}%
  {\xy@{startDIR}{\edef\startDirection@@{\the\Direction}\leave@}%
   \xyFN@\getarcOrientation@}}

\xydef@\getarcOrientation@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\getarcOrientation@}%
 \else\ifx\next @\xy@{@}{}\DN@{\getEndDirection@x}%
 \else\ifx\next,\xy@{,}{}\DN@,{\xyFN@\getarcOrientation@i}%
 \else\DN@{\getarcOrientation@i}\fi\fi\fi \next@ }%

\xydef@\getarcOrientation@i{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\getarcOrientation@i}%
 \else\ifx\next @\xy@{@}{}\DN@{\getEndDirection@x}%
 \else\ifx\next^{}\DN@##1{\xy@{^}{\def\arcOrient@@{^}}%
```

```
  \def\arcOrient@@{^}\xyFN@\getEndDirection@}%
 \else\ifx\next_\DN@##1{\xy@{_}{\def\arcOrient@@{_}}%
   \def\arcOrient@@{_}\xyFN@\getEndDirection@}%
 \else\DN@{\xyFN@\getEndDirection@}%
 \fi\fi\fi\fi \next@ }

\xydef@\getEndDirection@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\getEndDirection@}%
 \else\ifx\next,\DN@,{\xy@{,}{}\xyFN@\getEndDirection@}%
 \else\addEQ@\ifx\next\addEQ@\DN@{\xy@{=}{\save@\startDirection@}%
  \xyFN@\getEqDirection@}%
 \else \ifx\next!\DN@![[##1]]{\xy@{![[}{\save@}\getxyDirection@{##1}%
   \xy@{endDIR}{\edef\endDirection@@{\the\Direction}}%
  \xyFN@\getEndDirection@i}%
 \else \DN@{\xy@{origDIR}{\save@\origDirection@}\getEndDirection@i}%
 \fi\fi\fi\fi \next@ }

\xydef@\getEqDirection@{%
 \ifx\space@\next\expandafter\DN@\space{\xyFN@\getEqDirection@}%
 \else\ifx\next @\xy@{@}{}\DN@ @{%
  \xy@{sameDIR}{\edef\endDirection@@{\startDirection@@}\leave@}%
  \parEllipse@{1}{}}%
% \else\ifx\next (\DN@(##1)##2@{\xy@{()}{\leave@}\parEllipse@{##1}{##2}}%
 \else\ifx\next (\DN@({\xyFN@\checkEqOpen@}%
 \else \DN@{\getEndDirection@i}\fi\fi\fi \next@ }

\xydef@\checkEqOpen@{%
 \ifx\next *\DN@{\getEndDirection@i (}%
 \else \DN@##1)##2@{\xy@{()}\leave@ \parEllipse@{##1}{##2}}%
 \fi \next@ }

\xydef@\getEndDirection@i{\afterDIRECTIONorEMPTY
  {\xy@{endDIR}{\edef\endDirection@@{\the\Direction}\leave@}%
   \getEndDirection@x }%
  {\xy@{endDIR}\leave@ \getEndDirection@x }}

\xydef@\getEndDirection@x#1@{%
 \DN@{#1}\ifx\next@\empty\relax \else\xywarning@{\omittingTokens@}\fi
 \processDirections@ }

\xydef@\omittingTokens@{omitting unused tokens: \codeof\next@ }

\xydef@\processDirections@@{%
 \ifx\endDirection@@\empty \skewCircle@
 \else \skewEllipse@ \expandafter \afterARCextents@@ \fi }

\xydef@\processArcDirections@{%
 \xy@{ARCdirs}{\ifx\endDirection@@\empty
   \origDirection@ \edef\endDirection@@{\the\Direction}\fi}%
 \ellipseArc@
 \xy@{procARCdirs}\processArcDirections@@
 \let\arcScale@\ellsave@ \let\arcRadius@\ellsave@ \def\ellipseRatio@{}%
 \xy@{circSECTORS}{\doCircleSectors@}\endarc@ }

\xydef@\ellsave@{\xy@@{\save@}}
```

```
\xydef@\processArcDirections@@{\save@
 \bgroup \cfromid@{0@0}\pfromc@ \Direction=\startDirection@@
  \DN@{_}\ifx\next@\arcOrient@@
   \advance\Direction \ifnum\Direction<\z@-\fi 4096 \relax\fi
  \imposeDirection@
  \d@X=\cosDirection\Y@ybase \advance\d@X-\sinDirection\X@ybase
  \d@Y=-\cosDirection\Y@xbase \advance\d@Y\sinDirection\X@xbase
  \setupDirection@i
  \d@X=-\cosDirection\X@ybase \advance\d@X\sinDirection\X@xbase
  \d@Y=-\cosDirection\Y@ybase \advance\d@Y\sinDirection\Y@xbase
  \enter@\cplusthec@\X@c=\d@X \Y@c=\d@Y \leave@ \czeroEdge@
  \setupDirection@i
  \czeroEdge@
 \edef\next@{\egroup \cfromthec@ \count@=\the\Direction}\next@
 \edef\startDirection@@{\the\count@}\idfromc@{0@x}%
%
 \bgroup \cfromid@{0@0}\pfromc@ \Direction=\endDirection@@
 \DN@{_}\ifx\next@\arcOrient@@
  \advance\Direction \ifnum\Direction<\z@-\fi 4096 \relax\fi
 \imposeDirection@
  \d@X=\cosDirection\Y@ybase \advance\d@X-\sinDirection\X@ybase
  \d@Y=-\cosDirection\Y@xbase \advance\d@Y\sinDirection\X@xbase
  \setupDirection@i
  \d@X=-\cosDirection\X@ybase \advance\d@X\sinDirection\X@xbase
  \d@Y=-\cosDirection\Y@ybase \advance\d@Y\sinDirection\Y@xbase
  \enter@\cplusthec@\X@c=\d@X \Y@c=\d@Y \leave@
 \czeroEdge@
  \setupDirection@i
  \edef\next@{\egroup \cfromthec@ \count@=\the\Direction}\next@
 \edef\endDirection@@{\the\count@}\idfromc@{0@c}%
%
 \count@=\startDirection@@ \relax \count@@=\endDirection@@ \relax
 \DN@{^}\ifx\arcOrient@@\next@
  \ifnum\count@@>\count@ \else\advance\count@@ 8192 \relax\fi
 \else
  \ifnum\count@@<\count@ \else\advance\count@@-8192 \relax\fi\fi
 \edef\startDirection@@{\the\count@}%
 \edef\endDirection@@{\the\count@@}}%

\xydef@\doCircleSectors@{%
 \let\doCircleArc@=\doMergeEndArc@@ \leave@ \leave@
 \cfromid@{0@0}\pfromc@\cfromid@{0@p}\setupDirection@
 \count@=\Direction \DN@{^}\ifx\arcOrient@@\next@
  \ifnum\count@>\startDirection@@ \advance\count@-8192 \fi\relax
  \count@@=\count@ \advance\count@@ 8192 \relax
  \ifnum\count@@<\endDirection@@ \advance\count@@ 8192 \fi\relax
 \else
  \ifnum\count@<\startDirection@@ \advance\count@ 8192 \fi\relax
  \count@@=\count@ \advance\count@@-8192 \relax
  \ifnum\count@@>\endDirection@@ \advance\count@@-8192 \fi\relax
```

```
 \fi \relax
 \edef\preDirection@@{\the\count@}\edef\postDirection@@{\the\count@@}%
 \cfromid@{1@c}\setupDirection@ \count@=\Direction
 \def\nextii@{^}\ifx\nextii@\arcOrient@@
  \DN@{\forwardSectors@\doEllipseSectors@}%
  \ifnum\count@<\preDirection@@ \advance\count@ 8192 \relax
  \else \count@@=\count@\advance\count@-8192 \relax
   \ifnum\count@@<\preDirection@@ \else\count@=\count@@ \fi\fi
 \else
  \DN@{\backwardSectors@\doEllipseSectors@}%
  \ifnum\count@>\preDirection@@ \advance\count@-8192 \relax
  \else \count@@=\count@\advance\count@ 8192 \relax
   \ifnum\count@@>\preDirection@@ \else\count@=\count@@ \fi\fi
 \fi \edef\quadDirection@{\the\count@}%
 \next@ }

\xydef@\forwardSectors@{%
 \ifnum \startDirection@@<\quadDirection@ \def\startSector@@{0}%
 \else\count@=\preDirection@@ \advance\count@ 4096 \relax
  \ifnum \startDirection@@<\count@ \def\startSector@@{1}%
  \else\count@=\quadDirection@ \advance\count@ 4096 \relax
   \ifnum\startDirection@@<\count@ \def\startSector@@{2}%
   \else \def\startSector@@{3}\fi\fi\fi
%
 \ifnum\quadDirection@ <\endDirection@@ \relax
  \count@=\preDirection@@ \advance\count@ 4096 \relax
  \ifnum\count@<\endDirection@@ \relax
   \count@=\quadDirection@ \advance\count@ 4096 \relax
   \ifnum\count@<\endDirection@@ \relax
    \count@=\preDirection@@ \advance\count@ 8192 \relax
    \ifnum\count@<\endDirection@@ \relax
     \count@=\quadDirection@ \advance\count@ 8192 \relax
     \ifnum\count@<\endDirection@@ \relax
      \count@=\preDirection@@ \advance\count@ 12288 \relax
      \ifnum\count@<\endDirection@@ \relax
       \count@=\quadDirection@ \advance\count@ 12288 \relax
       \ifnum\count@<\endDirection@@ \def\endSector@@{7}%
       \else \def\endSector@@{6}\fi
      \else \def\endSector@@{5}\fi
     \else \def\endSector@@{4}\fi
    \else \def\endSector@@{3}\fi
   \else \def\endSector@@{2}\fi
  \else \def\endSector@@{1}\fi
 \else \def\endSector@@{0}\fi }

\xydef@\backwardSectors@{%
 \ifnum \startDirection@@>\quadDirection@ \def\startSector@@{0}%
 \else\count@=\preDirection@@ \advance\count@-4096 \relax
  \ifnum \startDirection@@>\count@ \def\startSector@@{1}%
  \else\count@=\quadDirection@ \advance\count@-4096 \relax
   \ifnum\startDirection@@>\count@ \def\startSector@@{2}%
```

```
   \else \def\startSector@@{3}\fi\fi\fi
%
 \ifnum\quadDirection@>\endDirection@@ \relax
  \count@=\preDirection@@ \advance\count@-4096 \relax
  \ifnum\count@>\endDirection@@ \relax
   \count@=\quadDirection@ \advance\count@-4096 \relax
   \ifnum\count@>\endDirection@@ \relax
    \count@=\preDirection@@ \advance\count@-8192 \relax
    \ifnum\count@>\endDirection@@ \relax
     \count@=\quadDirection@ \advance\count@-8192 \relax
     \ifnum\count@>\endDirection@@ \relax
      \count@=\preDirection@@ \advance\count@-12288 \relax
      \ifnum\count@>\endDirection@@ \relax
       \count@=\quadDirection@ \advance\count@-12288 \relax
       \ifnum\count@>\endDirection@@ \def\endSector@@{7}%
       \else \def\endSector@@{6}\fi
      \else \def\endSector@@{5}\fi
     \else \def\endSector@@{4}\fi
    \else \def\endSector@@{3}\fi
   \else \def\endSector@@{2}\fi
  \else \def\endSector@@{1}\fi
 \else \def\endSector@@{0}\fi }
```

```
\xydef@\doEllipseSectors@{%
 \ifx\startSector@@\endSector@@
  \DN@{\expandafter\doMergeBothArc@\expandafter{\arcSTYLE}{\endSector@@}}%
 \else
   \expandafter\doMergeStartArc@\expandafter{\arcSTYLE}{\startSector@@}%
  \edef\countArcs@@{\startSector@@}%
  \loop@ \count@=\countArcs@@ \relax \advance\count@\@ne
   \ifnum \count@ < \endSector@@ \edef\countArcs@@{\the\count@}%
    \bgroup\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{\countArcs@@}%
    \egroup
  \repeat@
  \DN@{\expandafter\doMergeEndArc@\expandafter{\arcSTYLE}{\endSector@@}}%
 \fi \next@ }
```

parsing of directions When the starting direction is along the line $\overline{pc}$ then a straight segment is typeset, ignoring any requested ending direction.

```
\xydef@\checkstartDirection@{%
 \bgroup\no@@ \edef\next@{\egroup\count@@=\the\Direction}\next@\relax
 \ifnum\count@@=\startDirection@@ \relax \DN@{\straightArc@}%
 \else \advance\count@@-\startDirection@@ \relax
  \ifnum\count@@=\ifnum\count@@<\z@-\fi 4096 \relax \DN@{\straightArc@}%
  \else \DN@{\processDirections@@}\fi\fi \next@ }
```

```
\xydef@\straightArc@{\xywarning@{straight arc encountered}%
 \xy@@{\let\bstartPLACE@=\relax
  \expandafter\doStraightArc@@\expandafter{\arcSTYLE}}\endarc@ }%
```

```
\xydef@\skewEllipse@{\xy@@{\doskewEllipse@}%
```

```
\def\afterARCextents@@{\xy@{aftARCext}{\let\doCircleArc@=\doMergeEndArc@@}%
 \ellipseArc@ \xy@{doCircARCs}{\doCircleArcs@}\endarc@ }%
\let\arcScale@=\relax \def\arcRadius@{}\def\ellipseRatio@{}}

\xydef@\skewCircle@{\xy@@{\setupDirection@}%
 \ifnum\Direction=\startDirection@@\relax\DN@{\halfEllipse@{1}}%
 \else
  \xy@@{\def\startDirection@{\Direction=\startDirection@@\imposeDirection@}%
   \cfromid@{0@p}\enter@\cplusthec@ \startDirection@
   \enter@\relax\begingroup\vfromslide@i{}@\czeroEdge@\leave@\idfromc@{0@m}%
   \let\endDirection@=\relax \let\doCircleArc@=\doMergeEndArc@@}%
  \DN@{\circleArc@}%
 \fi \next@ }

\xydef@\doskewEllipse@{%
 \def\startDirection@{\Direction=\startDirection@@ \relax
  \imposeDirection@}\startDirection@ \cfromid@{0@p}%
 \enter@\cplusthec@ \enter@\relax\begingroup\vfromslide@i{}@\leave@
 \idfromc@{0@x}%
 \ifx\endDirection@@\empty \parallelDirection@
 \else
  \def\endDirection@{\Direction=\endDirection@@ \relax
   \imposeDirection@ }\endDirection@ \skewEllB@
  \bgroup\setupDirection@\edef\next@{\egroup\count@@=\the\Direction}\next@
  \edef\endDirection@@{\the\count@@}%
 \fi
 \cfromid@{0@p}\bgroup \setupDirection@
  \edef\next@{\egroup \count@@=\the\Direction}\next@
 \edef\startDirection@@{\the\count@@}}

\xydef@\halfEllipse@#1{\xy@@{%
 \cfromid@{0@p}\pfromc@\cfromid@{0@c}\no@@
 \edef\endDirection@@{\the\Direction}%
 \advance\X@c\X@p \divide\X@c\tw@ \advance\Y@c\Y@p \divide\Y@c\tw@
 \idfromc@{0@0}\pfromc@ \cfromid@{0@p}\setbase@\X@p\Y@p\X@c\Y@c
 \dimen@=\X@origin \advance\dimen@-\X@c \dimen@ii=\cosDirection\dimen@
 \dimen@=\Y@origin \advance\dimen@-\Y@c \advance\dimen@ii \sinDirection\dimen@
 \X@c=\X@origin \Y@c=\Y@origin \enter@\cplusthec@
 \dimen@=#1\p@ \ifdim\dimen@<\z@ \DN@{^}\dimen@ii=-#1\dimen@ii
  \ifx\next@\arcOrient@@\def\arcOrient@@{_}\else\def\arcOrient@@{^}\fi
 \else \dimen@ii=#1\dimen@ii\fi
 \startDirection@ \enter@\relax\begingroup\expandafter
  \vfromslide@i\expandafter{\the\dimen@ii}@\leave@
 \idfromc@{1@c}\setbase@@ \X@c \Y@c \adjustBaseOrient@
 \count@@=\endDirection@@ \relax
 \DN@{^}\advance\count@@\ifx\arcOrient@@\next@\else-\fi 4096 \relax
 \edef\startDirection@@{\the\count@@}%
 \def\startDirection@{\Direction=\startDirection@@ \imposeDirection@ }%
 \let\doCircleArc@=\doMergeEndArc@@ }%
 \let\arcScale@=\relax \ellipseArc@
% \xy@@{\leave@ \doCircleArcs@}\endarc@ }
```

```
 \xy@@{\doCircleArcs@}\endarc@ }
```

```
\xydef@\parallelDirection@{\show\parallelDirection@}%
```

```
\xydef@\parEllipse@#1#2{\DN@{#2}\ifx\next@\empty
 \else\xywarning@{\omittingTokens@ }\fi \halfEllipse@{#1}}
```

```
\xydef@\skewEllB@{\cfromid@{0@c}\pfromc@\enter@\cplusthec@
 \enter@\relax\begingroup\vfromslide@i{}@\leave@
 \enter@{\pfromthep@\basefromthebase@}\no@@
 \bgroup \cfromid@{0@p}\pfromc@\cfromid@{0@x}\edef\next@{\egroup
  \X@c =\the\X@c \Y@c=\the\Y@c \X@p=\the\X@p \Y@p=\the\Y@p }\next@
 \Cintercept@@\leave@ \idfromc@{0@x}\cfromid@{0@p}\enter@\cplusthec@
 \cfromid@{0@c}\leave@ \enter@\cplusthec@ \cfromid@{0@x}%
 \X@c=-\X@c \Y@c=-\Y@c \leave@ \idfromc@{0@0}\pfromc@\cfromid@{0@p}%
 \setbase@\X@p\Y@p\X@c\Y@c \cfromid@{0@c}\setbase@@\X@c\Y@c
 \adjustBaseOrient@ }
```

```
\xydef@\adjustBaseOrient@{%
 \edef\tmp@{\expandafter\removePT@\the\X@xbase}\dimen@=\Y@ybase
 \divide\dimen@ 64 \relax \dimen@ii=\tmp@\dimen@
 \edef\tmp@{\expandafter\removePT@\the\X@ybase}\dimen@=\Y@xbase
 \divide\dimen@ 64 \relax \dimen@=\tmp@\dimen@
 \ifdim\dimen@>\dimen@ii \X@ybase=-\X@ybase \Y@ybase=-\Y@ybase \fi }
```

The end & Log

```
\xyendinput
```

```
% $Log: xyarc.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18 14:21:23  ross
% Ross's version
%
% Revision 3.3.1.1  1996/12/18  08:50:58  ross
% *** empty log message ***
%
% Revision 3.2  1995/09/19  18:20:20  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

3.10 Knots and Links feature

Vers. 3.4 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This feature provides a language for specifying knots, links and general arrangements of crossing strings.

Header:

```
%% $Id: xyknot.doc,v 3.4 1997/05/18 01:13:24 ross Exp $
%%
%% Xy-pic ‘‘Knots and Links’’ feature.
%% Copyright (c) 1994-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this macro package; if not, write to the Free Software Foundation,
%% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{knot}{Knots and Links feature}{\stripRCS$Revision: 3.4 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

To Do: Document this feature!

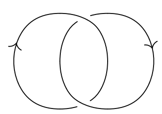
```
\message{knots and links,}
```

This `knot` feature is really a ‘construction kit’, providing pieces which may be placed appropriately to form knots and links. The types of pieces provided are of two kinds: the “crossings”, representing one string crossing over or under another; and “joins” which are used to connect what would otherwise be loose ends. Several types of each are provided, along with a simple way of specifying where to place arrowheads and labels.

All the pieces ultimately use curves from the `curve` extension, usually indirectly via the `arrow` feature. As such, processing can be memory-intensive and may seem rather slow. All the warnings and advice given elsewhere on techniques to handle pages and individual diagrams with many curves are especially applicable when using this feature.

Most constructions use `\ar` so make sure this feature is loaded.

```
\xyrequire{arrow}\xycatcodes
```



simple link

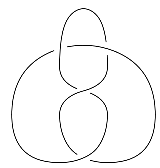
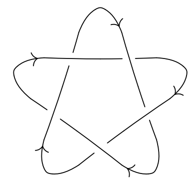


figure-8 knot



cinquefoil

Crossings

A "crossing" is intended to represent two strings passing close by, but not meeting. The macros provided specify typesetting within a square cell of coordinate values; using a non-square basis alters this shape, but see also note 3.10c below, for the technique that was used in the "cinquefoil" example above.

Notes

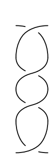
3.10a. Several families of crossing are provided. Those having names as `\v...` and `\h...` are designed to stack respectively vertically and horizontally. More precisely the current ⟨pos⟩ starts at the top-left corner and finishes at either the bottom-left or top-right. Say that a crossing is either a 'vertical crossing' or 'horizontal crossing' respectively.

This certainly applies to the `\..cross..` and `\..twist..` families, see figure 3.8 in which the strings enter and leave the square all with vertical tangents or all with horizontal tangents. Indeed *all* crossings are either vertical or horizontal, with the final letter indicating which for the `\xover..` families.

Furthermore there is a natural *orientation* for each crossing, as well as along each strand. This corresponds to the order in which ink is applied to the printed page, following the natural parametrization of each strand as a curved connection or arrow. This orientation determines whether a crossing is 'over' (mathematically, positive or right-handed) or 'under' (mathematically, negative or left-handed). It is used in determining the location of labels and the direction of arrowheads placed along the strings. Note that `\..cross..` and `\..twist..` crossings may set the same curves, but with different orientation and label-positioning.

Figure 3.8 displays the orientation on all the crossings, grouping them into subfamilies consisting of right-handed, left-handed and non-crossings. Also indicated are the default positions for labels and arrow-tips; each piece uses the same code for tips and labels, e.g. `\vover<>|>>><{x}|{y}>{z}`.

The `\x...` crossings do not stack easily since their tangents are at 45° to the coordinate axes. It is the last letter in the name which denotes whether the particular crossing is vertical or horizontal. On the other hand `\vover`, `\vunder` etc. stack vertically on top of a `\vcross`, `\vtwist` etc.; similarly `\hover` stacks at the left of `\hcross`, `\htwist` etc.



```
$$\xy 0;/r1pc/:
 ,{\vunder\vtwist\vtwist\vunder-}\endxy
\qquad\qquad\qquad \xy 0;/r1pc/:+(0,-1.5)
```

```
,{\hover\hcross\hcross\hover-}\endxy$$
```

Parsing

```
\xydef@\xykparsecross@{%
 \def\xykSCALE@@{}\edef\xyknotPLACE{\xykmidPLACE@}%
 \let\xykparser@=\xykparsecross@@
 \def\xykdefaultbreak@{\let\xykbreak@=\xykforetemp@
  \def\xyknotPLACE{(.5)}\appendtoholder@}%
 \xyFN@\xykparsescale@ }
```

```
{\xyuncatcodes \catcode`@=11
\gdef\next{%
 \ifx\space@\next \expandafter\DN@\space{\xyFN@\xykparsecross@@}%
 \else\addLT@\ifx\next \addLT@\DN@{\xyprecross@}%
 \else\addGT@\ifx\next \addGT@\DN@{\xypostcross@}%
 \else\addEQ@\ifx\next \addEQ@\DN@{\xyFN@\xykparsetips@@}%
 \else\ifx|\next\DN@|{\xymidcross@}%
 \else\addAT@\ifx\next \addAT@\DN@(##1){%
  \xyadjustBREAK@i##1@@\xyFN@\xykparsecross@@}%
 \else\DN@{\afterknot@}%
 \fi\fi\fi\fi\fi\fi \next@ }}
\xylet@\xykparsecross@@=\next
```

```
\xydef@\xyprecross@{%
 \def\xykholder@{\xykprebreak@@}%
 \edef\xyknotPLACE{\xykprePLACE@}%
 \xyFN@\xykparsebreak@}
```

```
\xydef@\xypostcross@{%
 \def\xykholder@{\xykpostbreak@@}%
 \edef\xyknotPLACE{\xykpostPLACE@}%
 \xyFN@\xykparsebreak@}
```

```
\xydef@\xymidcross@{%
 \def\xykholder@{\xykmidbreak@@}%
 \edef\xyknotPLACE{\xykmidPLACE@}%
 \xyFN@\xykparsebreak@}
```

These macros are common to most crossings.

```
\xydef@\xykcross@#1{\xykcheckTIPS@
 \expandafter\toks@\expandafter{\expandafter#1\knotSTYLE}}
```

```
\xydef@\xykoverstring@{\addtotoks@{{}}\xykmidbreak@ \xyknottips@}
\xydef@\xykunderstring@{\xykprebreak@ \xykpostbreak@ \xyknottips@}
```

“cross” crossings:

The initialisation…

```
\xydef@\vcross{\begingroup\def\afterknot@{\xyvcross}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
```

```
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\hcross{\begingroup\def\afterknot@{\xyhcross}%
  \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\vcrossneg{\begingroup\def\afterknot@{\xyvcrossneg}%
  \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\hcrossneg{\begingroup\def\afterknot@{\xyhcrossneg}%
  \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\vuncross{\begingroup\def\afterknot@{\xyvuncross}%
  \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%
\xydef@\huncross{\begingroup\def\afterknot@{\xyhuncross}%
  \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%
```

The interface...

```
\xydef@\xyvcross{\xykcross@\xykcrossv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyhcross{\xykcross@\xykcrossh
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}

\xydef@\xyvcrossneg{\xykcross@\xykcrossv
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyhcrossneg{\xykcross@\xykcrossh
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(1,0)}}

\xydef@\xyvuncross{\xykcross@\xykuncrossv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyhuncross{\xykcross@\xykuncrossh
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}
```

The drawing code...

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<",{\xykz_\ar #1@'{"_<"**{}?(.25)@+,?+(.375,0)@+,
  "_>";"^>"**{}?-(.375,0)@+,?(.25)@+}"_>"#3#4#5},
 "_<",{\xykz^\ar #2@'{"^<"**{}?(.25)@+,?+(.375,0)@+,
  "^>";"_>"**{}?-(.375,0)@+,?(.25)@+}"^>"#6#7#8}
 \restore\POS #9}}
\xylet@\xykcrossv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "_<",{\xykz_\ar #1@'{"_>"**{}?(.25)@+,?+(0,.375)@+,
```

```
  "^>";"^<"**{}?-(0,.375)@+,?(.25)@+}"^>"#3#4#5},
 "_>",{\xykz^\ar #2@'{"_<"**{}?(.25)@+,?+(0,.375)@+,
  "^<";"^>"**{}?-(0,.375)@+,?(.25)@+}"^<"#6#7#8}
 \restore\POS #9}}
\xylet@\xykcrossh=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<",{\xykz^\ar #1@'{"_<"**{}?(.25)@+,?(.375)+(.375,0)@+,
  "^>";"_>"**{}?(.375)-(.375,0)@+,?(.25)@+}"^>"#3#4#5},
 "_<",{\xykz^\ar #2@'{"^<"**{}?(.25)@+,?(.375)+(.375,0)@+,
  "_>";"^>"**{}?(.375)-(.375,0)@+,?(.25)@+}"_>"#6#7#8}
 \restore\POS #9}}
\xylet@\xykuncrossv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "_<",{\xykz^\ar #1@'{"_>"**{}?(.25)@+,?(.375)+(0,.375)@+,
  "^<";"^>"**{}?(.375)-(0,.375)@+,?(.25)@+}"^<"#3#4#5},
 "_>",{\xykz^\ar #2@'{"_<"**{}?(.25)@+,?(.375)+(0,.375)@+,
  "^>";"^<"**{}?(.375)-(0,.375)@+,?(.25)@+}"^>"#6#7#8}
 \restore\POS #9}}
\xylet@\xykuncrossh=\next
```

"over" crossings:

The initialisation…

```
\xydef@\vover{\begingroup\def\afterknot@{\xyvover}%
 \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.8)}\def\xykpostbreak@@{|(.725)\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\hover{\begingroup\def\afterknot@{\xyhover}%
 \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.8)}\def\xykpostbreak@@{|(.725)\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\vunder{\begingroup\def\afterknot@{\xyvunder}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.8)}%
 \def\xykmidPLACE@{(.2)}\def\xykpostbreak@@{|(.275)\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\hunder{\begingroup\def\afterknot@{\xyhunder}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.8)}%
 \def\xykmidPLACE@{(.2)}\def\xykpostbreak@@{|(.275)\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\vunover{\begingroup\def\afterknot@{\xyvunover}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.8)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%

\xydef@\hunover{\begingroup\def\afterknot@{\xyhunover}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.8)}\def\xykpostbreak@@{}%
```

```
 \xyFN@\xykparsecross@ }%

\xydef@\xoverv{\begingroup\def\afterknot@{\xyxoverv}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}%
 \def\xykmidPLACE@{(.15)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\xunderv{\begingroup\def\afterknot@{\xyxunderv}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}%
 \def\xykmidPLACE@{(.15)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\xoverh{\begingroup\def\afterknot@{\xyxoverh}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}%
 \def\xykmidPLACE@{(.15)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\xunderh{\begingroup\def\afterknot@{\xyxunderh}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}%
 \def\xykmidPLACE@{(.85)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%

\xydef@\xunover{\begingroup\def\afterknot@{\xyxunover}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}%
 \def\xykmidPLACE@{(.15)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%

\xydef@\xunoverv{\begingroup\def\afterknot@{\xyxunoverv}%
 \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}%
 \def\xykmidPLACE@{(.2)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%

\xydef@\xunoverh{\begingroup\def\afterknot@{\xyxunoverh}%
 \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}%
 \def\xykmidPLACE@{(.8)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%
```

The interface. . .

```
\xydef@\xyvover{\xykcross@\xykoverv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyhover{\xykcross@\xykoverh
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}

\xydef@\xyvunder{\xykcross@\xykoverv
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyhunder{\xykcross@\xykoverh
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(1,0)}}

\xydef@\xyvunover{\xykcross@\xykunoverv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyhunover{\xykcross@\xykunoverh
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}

\xydef@\xyxoverv{\xykcross@\xykxoverv
```

```
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}
\xydef@\xyxunderv{\xykcross@\xykxoverv
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(0,-1)}}
\xydef@\xyxoverh{\xykcross@\xykxoverh
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}
\xydef@\xyxunderh{\xykcross@\xykxoverh
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(1,0)}}
\xydef@\xyxunover{\xykcross@\xykxoverv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}
\xydef@\xyxunoverv{\xykcross@\xykxunoverv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}
\xydef@\xyxunoverh{\xykcross@\xykxunoverh
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}
```

The drawing code…

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<",{\xykz^\ar #1@'{"^>"**{}?(.25)@+,?(.625)+(0,-.125)@+,
  "_>";"^>"**{}?(.625)-(.125,0)@+,?(.25)@+}"_>"#3#4#5},
 "_<",{\xykz^\ar #2@'{"^<"**{}?(.25)@+,?(.625)+(.125,0)@+,
  "^>";"^<"**{}?(.625)-(0,.125)@+,?(.25)@+}"^>"#6#7#8},\restore\POS#9}}
\xylet@\xykoverv=\next
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "_<",{\xykz^\ar #1@'{"^<"**{}?(.25)@+,?(.625)+(.125,0)@+,
  "^>";"^<"**{}?(.625)-(0,.125)@+,?(.25)@+}"^>"#3#4#5},
 "_>",{\xykz^\ar #2@'{"_<"**{}?(.25)@+,?(.625)+(0,.125)@+,
  "^<";"_<"**{}?(.625)+(.125,0)@+,?(.25)@+}"^<"#6#7#8},\restore\POS#9}}
\xylet@\xykoverh=\next
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<",{\xykz^\ar #1@'{"^>"**{}?(.25)@+,?-(0,.175)@+,?(.75)@+}
  "^>"#3#4#5},
 "_<",{\xykz^\ar #2@'{"^<"**{}?(.25)@+,?+(.125,0)@+,
   "_>"**{}?+(0,.7)@+,"_>";"^>"**{}?-(.125,0)@+,?(.25)@+}
  "_>"#6#7#8},\restore\POS#9}}
\xylet@\xykunoverv=\next
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "_<",{\xykz^\ar #1@'{"^<"**{}?(.25)@+,?+(.175,0)@+,?(.75)@+}
  "^<"#3#4#5},
 "_>",{\xykz^\ar #2@'{"_<"**{}?(.25)@+,?+(0,.125)@+,
   "^>"**{}?-(.7,0)@+,"^>";"^<"**{}?-(0,.125)@+,?(.25)@+}
  "^>"#6#7#8},\restore\POS#9}}
\xylet@\xykunoverh=\next
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<";"^>"**{}?="^_","_<";"_>"**{}?;"^_",**{}?="^_",
 "^>",{\xykz^\ar #1@'{"^_"@+}"_<"#3#4#5},
 "^<",{\xykz_\ar #2@'{"^_"@+}"_>"#6#7#8},\restore\POS#9}}
\xylet@\xykxoverv=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<";"_<"**{}?="^_","^>";"_>"**{}?;"^_",**{}?="^_",
 "^<",{\xykz_\ar #1@'{"^_"@+}"_>"#3#4#5},
 "_<",{\xykz_\ar #2@'{"^_"@+}"^>"#6#7#8},\restore\POS#9}}
\xylet@\xykxoverh=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<";"^>"**{}?="^_","_<";"_>"**{}?;"^_",**{}?="^_",
 "^>",{\xykz^\ar #1@'{"^_"@+}"_>"#3#4#5},
 "^<",{\xykz^\ar #2@'{"^_"@+}"_<"#6#7#8},\restore\POS#9}}
\xylet@\xykxunoverv=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<";"_<"**{}?="^_","^>";"_>"**{}?;"^_",**{}?="^_",
 "^<",{\xykz^\ar #1@'{"^_"@+}"^>"#3#4#5},
 "_<",{\xykz^\ar #2@'{"^_"@+}"_>"#6#7#8},
  \restore\POS#9}}
\xylet@\xykxunoverh=\next
```

"twist" crossings:

The initialisation…

```
\xydef@\vtwist{\begingroup\def\afterknot@{\xyvtwist}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\htwist{\begingroup\def\afterknot@{\xyhtwist}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\vtwistneg{\begingroup\def\afterknot@{\xyvtwistneg}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\htwistneg{\begingroup\def\afterknot@{\xyhtwistneg}%
 \def\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.9)}%
 \def\xykmidPLACE@{(.9)}\def\xykpostbreak@@{|\knothole}%
 \xyFN@\xykparsecross@ }%
\xydef@\vuntwist{\begingroup\def\afterknot@{\xyvuntwist}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}%
 \def\xykmidPLACE@{(.85)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%
\xydef@\huntwist{\begingroup\def\afterknot@{\xyhuntwist}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}%
 \def\xykmidPLACE@{(.85)}\def\xykpostbreak@@{}%
 \xyFN@\xykparsecross@ }%
```

The interface…

```
\xydef@\xyvtwist{\xykcross@\xyktwistv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}
```

```
\xydef@\xyvtwistneg{\xykcross@\xyktwistv
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyvuntwist{\xykcross@\xykuntwistv
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(0,-1)}}

\xydef@\xyhtwist{\xykcross@\xyktwisth
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}

\xydef@\xyhuntwist{\xykcross@\xykuntwisth
 \xykoverstring@ \xykunderstring@ \xykhvobject@{+(1,0)}}

\xydef@\xyhtwistneg{\xykcross@\xyktwisth
 \xykunderstring@ \xykoverstring@ \xykhvobject@{+(1,0)}}
```

The drawing code…

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^>",{\xykz_\ar #1@'{"_>"**{}?(.25)@+,?-(.375,0)@+,
  "_<";"^<"**{}?+(.375,0)@+,?(.25)@+}"_<"#3#4#5},
 "^<",{\xykz^\ar #2@'{"_<"**{}?(.25)@+,?+(.375,0)@+,
  "_>";"^>"**{}?+(-.375,0)@+,?(.25)@+}"_>"#6#7#8}
 \restore\POS #9}}
\xylet@\xyktwistv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^>",{\xykz^\ar #1@'{"_>"**{}?(.125)@+,?(.375)-(.25,0)@+,
  ?(.625)-(.25,0)@+,?(.875)@+}"_>"#3#4#5},
 "^<",{\xykz^\ar #2@'{"_<"**{}?(.125)@+,?(.375)+(.25,0)@+,
  ?(.625)+(.25,0)@+,?(.875)@+}"_<"#6#7#8}
 \restore\POS #9}}
\xylet@\xykuntwistv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<",{\xykz_\ar #1@'{"^>"**{}?(.25)@+,?+(0,-.375)@+,
  "_>";"_<"**{}?+(0,.375)@+,?(.25)@+}"_>"#3#4#5},
 "_<",{\xykz^\ar #2@'{"_>"**{}?(.25)@+,?+(0,.375)@+,
  "^>";"^<"**{}?-(0,.375)@+,?(.25)@+}"^>"#6#7#8}
 \restore\POS #9}}
\xylet@\xyktwisth=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
 "^<",{\xykz^\ar #1@'{"^>",**{}?(.125)@+,?(.375)+(0,-.25)@+,
  ?(.625)+(0,-.25)@+,?(.875)@+}"^>"#3#4#5},
 "_<",{\xykz^\ar #2@'{"_>",**{}?(.125)@+,?(.375)+(0,.25)@+,
  ?(.625)+(0,.25)@+,?(.875)@+}"_>"#6#7#8},\restore\POS #9}}
\xylet@\xykuntwisth=\next
```

3.10b. The above examples also show how to use `-` to get the mirror-image of a particular crossing. Any numerical scale factor can be used by enclosing it within `[..]` e.g. `[2.3]` scaling a single piece without affecting the rest of the picture. The scale-factor *must* occur before any label or arrow-tip specifiers, see below). Vertical crossings remain vertical under scalings; the current ⟨pos⟩ still moves by 1 coordinate unit in the 'down' direction. Similarly horizontal crossings remain horizontal. The single character `-` is a shorthand version for `[-1]`, effectively giving a half-turn rotation in a rectangular basis.

```
\xydef@\xykparsescale@{%
 \ifx\space@\next \expandafter\DN@\space{\xyFN@\xykparsescale@}%
 \else\addDASH@\ifx\next \addDASH@\DN@{\def\xykSCALE@@{-1}%
  \xyFN@\xykparser@}%
 \else\ifx[\next \DN@[##1]{\dimen@=##1\p@
  \ifdim\dimen@=\z@\xywarning@{\xykscaleerror@}%
  \else\edef\xykSCALE@@{\expandafter\removePT@\the\dimen@}\fi
  \xyFN@\xykparser@}%
 \else\ifx~\next \DN@~##1##2##3##4{%
  \def\xykSCALE@@{@{##1}{##2}{##3}{##4}}\xyFN@\xykparser@}%
 \else\DN@{\xyFN@\xykparser@}%
 \fi\fi\fi\fi \next@}
```

```
\xydef@\xykscaleerror@{Ignoring zero scale factor}
```

3.10c. A knot-piece need not be rectangular. By specifying `~`$\langle\text{pos}\rangle_1\langle\text{pos}_2\rangle\langle\text{pos}_3\rangle\langle\text{pos}_4\rangle$ the four corners `UL`, `UR`, `DL`, `DR` are set to the given $\langle\text{pos}\rangle$s respectively. The local basis is established so that

$$\begin{aligned} r\text{–hop} &\leftrightarrow \tfrac{1}{2}(\langle\text{pos}_2\rangle - \langle\text{pos}_1\rangle + \langle\text{pos}_4\rangle - \langle\text{pos}_3\rangle) \\ u\text{–hop} &\leftrightarrow \tfrac{1}{2}(\langle\text{pos}_1\rangle - \langle\text{pos}_3\rangle + \langle\text{pos}_2\rangle - \langle\text{pos}_1\rangle)\ . \end{aligned}$$

3.10d. With a non-rectangularly shaped piece it will usually be necessary to adjust the place where the 'hole' occurs in the 'under' string. This is done by specifying `@(`$\langle\text{num}\rangle$`)`, with $0 \leq \langle\text{num}\rangle \leq 1$ being the parameter value of the new location for the hole.

```
%\xydef@\xykobject@{%
% \ifx\xykSCALE@@\empty\DN@{\xykkeepscale@}%
% \else \DN@{\xyksavescale@{}}%
% \fi \next@ }
```

```
\xydef@\xykhobject@{\aftergroup\xykhadjust@
 \ifx\xykSCALE@@\empty\DN@{\xykkeepscale@}%
 \else \DN@{\xyksavescale@{}}%
 \fi \next@ }
```

```
\xydef@\xykvobject@{\aftergroup\xykvadjust@
 \ifx\xykSCALE@@\empty\DN@{\xykkeepscale@}%
 \else \DN@{\xyksavescale@{}}%
 \fi \next@ }
```

```
\xydef@\xykhvobject@{\aftergroup\xykhvadjust@
 \ifx\xykSCALE@@\empty\DN@{\xykkeepscale@}%
 \else \DN@{\xyksavescale@{}}%
 \fi \next@ }
```

```
\xydef@\xykhuobject@{\aftergroup\xykhuadjust@
 \ifx\xykSCALE@@\empty\DN@{\xykkeepscale@}%
 \else \DN@{\xyksavescale@{}}%
 \fi \next@ }
```

```
\xydef@\xyksavescale@#1#2{%
 \expandafter\ifx\the\Edge@c\zeroEdge\relax \def\xyksaveEdgec{}%
 \else
  \edef\xyksaveEdgec{\noexpand\Edge@c={\expandafter\noexpand\the\Edge@c}%
```

```
   \L@c=\the\L@c \R@c=\the\R@c \D@c=\the\D@c \U@c=\the\U@c}%
 \fi
%
 \edef\next@##1##2{\endgroup
  \def\noexpand\xyksaveEdgec{}%
  \def\noexpand\xykSCALE@@\noexpand{##1}%
  \def\noexpand\xykjoincontrol\noexpand{\xykjoincontrol}%
  \noexpand\toks@\noexpand{##2}%
  \noexpand\xy@@{\def\noexpand\xyksaveEdgec{}%
   \def\noexpand\xykSCALE@@{##1}%
   \def\noexpand\xykjoincontrol{\xykjoincontrol}}}%
 \expandafter\expandafter\expandafter\next@
 \expandafter\expandafter\expandafter{\expandafter\xykSCALE@@\expandafter}%
 \expandafter{\the\toks@}%
%
 \xykrescale@ #1\the\toks@\restore\POS#2}

\xydef@\xykkeepscale@{%
 \edef\next@##1{\endgroup \noexpand\toks@={##1}%
  \def\noexpand\xykjoincontrol{\xykjoincontrol}}%
 \expandafter\next@\expandafter{\the\toks@}%
 \def\xykSCALE@@{}\def\xyksaveEdgec{}%
 \xy@@{\def\xykSCALE@@{}\def\xyksaveEdgec{}}%
 \edef\next@{\def\noexpand\xykjoincontrol{\xykjoincontrol}}%
 \expandafter\xy@@\expandafter{\next@}%
 \xyknoflexscale@ \the\toks@}

\xydef@\xykrescale@{\expandafter\xykcheckscale@\xykSCALE@@ !}%

\xydef@\xykcheckscale@#1#2!{%
 \expandafter\DN@\expandafter{\codeof{#1}}%
 \expandafter\DNii@\expandafter{\codeof{@}}%
 \ifx\next@\nextii@\relax\DN@{\xykflexiscale@#2%
 \xy@@{\divide\X@xbase2\divide\X@ybase2\divide\Y@xbase2\divide\Y@ybase2%
  \def\xykSCALE@@{}}\def\xykSCALE@@{}%
  \expandafter\toks@\expandafter{\afterFLEX@}}%
 \else\DN@{\save \xykrescale@@}\fi \next@ }

\xydef@\xykflexiscale@{%
 \expandafter\def\expandafter\afterFLEX@\expandafter{\the\toks@}%
 \xykflexiscale@@ }

\xydef@\xykrescale@@{%
 \edef\next@{\X@xbase=\xykSCALE@@\X@xbase \X@ybase=\xykSCALE@@\X@ybase
  \Y@xbase=\xykSCALE@@\Y@xbase \Y@ybase=\xykSCALE@@\Y@ybase}%
 \expandafter\xy@@\expandafter{\next@}%
 \ifdim\xykSCALE@@\p@<\z@\relax \DN@{\xyknegflexscale@}%
 \else\DN@{\xyknoflexscale@}\fi \next@ }

{\xyuncatcodes \gdef\next#1#2#3#4{\save="@",
  #1,="^<",\POS"@",#2,="^>",\POS"@",#3,="_<",\POS"@",#4,="_>",
 (0,0);"^>"-"^<"+"_>"-"_<":"^<"-"_<"+"^>"-"_>"::}}
\xylet@\xykflexiscale@@=\next
```

```
{\xyuncatcodes \gdef\next{\save ="^<",
 +(1,0)="^>",-(1,1)="_<",+(1,0)="_>"\restore }}
\xylet@\xyknoflexscale@@=\next
\xylet@\xyknoflexscale@=\next

{\xyuncatcodes \gdef\next{\save ="_>",
 +(-1,1)="^<",+(1,0)="^>",-(1,1)="_<"\restore }}
\xylet@\xyknegflexscale@@=\next
\xylet@\xyknegflexscale@=\next

{\xyuncatcodes \gdef\next{\save ="_<",
 +(1,0)="_>",+(-1,1)="^<",+(1,0)="^>"\restore }}
\xylet@\xykhscale@@=\next

{\xyuncatcodes \gdef\next{\save ="^>",
 -(1,0)="^<",+(1,-1)="_>",-(1,0)="_<"\restore }}
\xylet@\xykvscale@@=\next

\xydef@\xykhadjust@{%
 \let\xyknoflexscale@=\xykhscale@@
 \let\xyknegflexscale@=\xyknegflexscale@@ }

\xydef@\xykvadjust@{%
 \let\xyknoflexscale@=\xyknoflexscale@@
 \let\xyknegflexscale@=\xykhscale@@ }

\xydef@\xykhvadjust@{%
 \let\xyknoflexscale@=\xyknoflexscale@@
 \let\xyknegflexscale@=\xyknegflexscale@@ }

\xydef@\xykhuadjust@{%
 \let\xyknoflexscale@=\xykhscale@@
 \let\xyknegflexscale@=\xykvscale@@ }
```

3.10e. The `knot` feature allows for the easy placement of the following objects along the strings of a crossing:

- labels on the strings;
- arrowheads for direction or orientation;
- holes in strings, allowing another string to be drawn passing over.

3.10f. The characters `<`, `>` and `|` are used to indicate to which string portion the object is associated; with `|` denoting the string which crosses the other, while `<` and `>` denote the initial and final portions of the 'crossed' string.

```
\xydef@\xykprebreak@{%
 \ifx\xykprebreak@@\empty\addtotoks@{{}}\else
  \expandafter\addtotoks@\expandafter{\expandafter{\xykprebreak@@}}\fi}

\xydef@\xykpostbreak@{%
 \ifx\xykpostbreak@@\empty\addtotoks@{{}}\else
  \expandafter\addtotoks@\expandafter{\expandafter{\xykpostbreak@@}}\fi}

\xydef@\xykmidbreak@{%
 \ifx\xykmidbreak@@\empty\addtotoks@{{}}\else
  \expandafter\addtotoks@\expandafter{\expandafter{\xykmidbreak@@}}\fi}
```

3.10g. A simple label enclosed in braces, for example `\vcross>{x}`, is set in math-mode using the `\labelstyle`, at a pre-determined place on the string portion, shifted in either the 'above' or 'below' direction from the curve at this point. (For each crossing depicted in figure 3.8 only default values are used for the place and shift-direction.)

3.10h. If the first character within the braces `{..}` is `*` e.g. `\htwist>{*`⟨object⟩`}`, then a general ⟨object⟩ may be placed as a label. Furthermore if the first character is `^` or `_` or `|`, then the interpretation is, e.g. `\vtwist<{^`⟨anchor⟩⟨it⟩`}`, as in 3.2 to place ⟨it⟩ as a label along an `\ar` of the `arrow` feature.

3.10i. A second character `<` or `>` specifies that an arrowhead should appear at the pre-determined place on the chosen string. Here `>` denotes an arrowhead pointing with the natural orientation, while `<` points against. Due to the curvature of the strings, it is usually best to `\UseComputerModernTips` rather than normal arrow-tips.

3.10j. To generate a 'hole' use `\knothole`, or simply `\khole`, as following token. This generates a 'break', in the sense of 3.3j. Indeed such a 'hole' is used to separate the two portions of the 'crossed' string. Default size for the hole is 5pt, which is alterable via `\knotholesize{`⟨dimen⟩`}`; normally used to set the size for *all* holes in a diagram.

3.10k. If the resulting `\khole` is either too large or perhaps non-existent, this could be due to a technicality in the way breaks in curves are handled. This problem should not occur with the standard crossings, using a rectangular basis, but it may occur with non-rectangular bases. An easy 'fix' is to include an extra *null-break* on the string, using `<|`, `>|` or `||`, which should place the zero-sized break at parameter value .5 on the curve. The specification should precede a `\khole` at a higher parameter value, or come after one at a lower value.

The aim of the above 'fix' is to position the null-break as close as possible to where the curve is farthest from the line joining its end-points, which is usually at parameter value `(.5)`.

```
\xydef@\xykparsebreak@{%
 \ifx\space@\next \expandafter\DN@\space{\xyFN@\xykparsebreak@}%
 \else \addLT@\ifx\next
  \expandafter\appendtoholder@\xykbacktemp@\next@
  \addLT@\DN@{\xyFN@\xykparser@}%
 \else \addGT@\ifx\next
  \expandafter\appendtoholder@\xykforetemp@\next@
  \addGT@\DN@{\xyFN@\xykparser@}%
 \else \ifx|\next
  \expandafter\appendtoholer@\xyknulltemp@\next@
  \DN@|{\xyFN@\xykparser@}%
 \else \ifx\next \bgroup\DN@##1{\checkgroupbreak@##1@}%
 \else \ifx (\next\DN@(##1){\xyadjustknotPLACE@i##1@@%
  \xyFN@\xykparsebreak@}%
 \else \ifx\next \knothole@\DN@##1{\expandafter\appendtoholder@@
   \expandafter{\expandafter|\xyknotPLACE\knothole}\next@
  \xyFN@\xykparser@}%
 \else
  \expandafter\xykdefaultbreak@\xykbreak@\next@
  \DN@{\xyFN@\xykparser@}%
 \fi\fi\fi\fi\fi\fi\fi \next@ }
```

Templates for arrowheads.

```
{\xyuncatcodes \gdef\next{^>*\dir{>}}}
\xylet@\xykforetips@=\next
```

```
{\xyuncatcodes \gdef\next{^<*\dir{<}}}
\xylet@\xykbacktips@=\next
```

```
{\xyuncatcodes \gdef\next{*!/-1pt/=0\dir{>}}}
\xylet@\xykforetemp@=\next
```

```
{\xyuncatcodes \gdef\next{*!/1pt/=0\dir{<}}}
\xylet@\xykbacktemp@=\next
```

```
{\xyuncatcodes \gdef\next{*=<\xykholesize>[o]{}}}
\xylet@\xykholetemp@=\next
\xydef@\knothole@{\hbox to\xykholesize{%
 \dimen@=\xykholesize \divide\dimen@\tw@
 \hfill \vrule height\dimen@ depth\dimen@ width\z@ }}
\xylet@\knothole=\knothole@
\xylet@\khole=\knothole@
```

```
{\xyuncatcodes \gdef\next{*=<0pt>{}}}
\xylet@\xyknulltemp@=\next
\xylet@\xykbreak@=\xyknulltemp@
```

When the next token is a begin-group character `{` then look at the first token inside the group. If it is `^`,`_` or `|` then the group contents will be an ⟨anchor⟩ for user-chosen places along the string. Otherwise the place is taken to be the already specified `\xyknotPLACE` and the group contains a label, to be set in the usual way for labels, using `_` due to the orientation of the curve. However if the first token is `*` then the user is assumed to be dropping his own label, possibly with ⟨object⟩ modifiers.

```
\xydef@\checkgroupbreak@#1#2@{%
 \ifx|#1\DN@{\appendtoholder@@#1#2\next@ \xyFN@\xykparser@}%
 \else \ifx_#1\DN@{\appendtoholder@@_#2\next@ \xyFN@\xykparser@}%
 \else \ifx^#1\DN@{\appendtoholder@@^#2\next@ \xyFN@\xykparser@}%
 \else
 \def\next##1\next{\expandafter\DNii@\expandafter{\xyk@@##1}}%
 \expandafter\next\xyknotPLACE\next
 \ifx*#1%
  \expandafter\DN@\expandafter{%
    \expandafter\appendtoholder@@\nextii@#1#2\next@ \xyFN@\xykparser@}%
 \else %\expandafter\DNii@\expandafter{\expandafter\xyk@@\xyknotPLACE}%
   \expandafter\DN@\expandafter{%
    \expandafter\appendtoholder@@\nextii@{#1#2}\next@ \xyFN@\xykparser@}%
 \fi\fi\fi\fi \next@ }
```

```
{\xyuncatcodes \gdef\next{%
 \expandafter{\expandafter^\xyknotPLACE##1}}}%
 \DN@{\xydef@\xykholder@@##1}\DNii@{\expandafter\toks@}%
 \expandafter\expandafter\expandafter\next@
 \expandafter\expandafter\expandafter{\expandafter\nextii@\next}
```

```
\xydef@\appendtoholder@#1\next@{\xykholder@@{#1}%
 \DN@##1{\expandafter\DN@\expandafter{\xykholder@##1}}%
 \expandafter\next@\expandafter{\the\toks@}\appendtoholder@@@ }
```

```
\xydef@\appendtoholder@@#1\next@{%
 \DN@##1\next@{\expandafter\DN@\expandafter{\xykholder@##1}}%
  \next@#1\next@ \appendtoholder@@@}%

\xydef@\appendtoholder@@@{%
 \expandafter\expandafter\expandafter\DN@
  \expandafter\expandafter\expandafter{\next@}%
 \DNii@{\expandafter\def\xykholder@}%
  \expandafter\nextii@\expandafter{\next@}}

{\xyuncatcodes \gdef\next{%
 \expandafter{\expandafter|\xyknotPLACE##1}}}%
\DN@{\xydef@\xykholer@@##1}\DNii@{\expandafter\toks@}%
 \expandafter\expandafter\expandafter\next@
 \expandafter\expandafter\expandafter{\expandafter\nextii@\next}

\xydef@\appendtoholer@#1\next@{\xykholer@@{#1}%
 \DN@##1{\expandafter\DN@\expandafter{\xykholder@##1}}%
 \expandafter\next@\expandafter{\the\toks@}\appendtoholder@@@ }
```

Multiple breaks, arrow-heads and labels may be specified along the two strings of a crossing; simply place their specifications one after another; e.g. `<>|>>><{x}|{y}>{z}` was used in figure 3.8.

The only proviso is that all 'breaks' along a single strand must occur with increasing order of parameter position. On the 'crossed' string this includes the automatic 'hole' to create space for the other string. Hence it is advisable to use just the `(+..)` and `(-..)` variants for small adjustments, and to keep these correctly ordered.

```
\xydef@\xykz#1{\let\xyk@@=#1\xy@@{\let\xyk@@=#1}}
{\xyuncatcodes \global\let\next=_}
\xylet@\xyk@@=\next
```

Adjustment of position along the strings can be achieved using a ⟨factor⟩, as in `\vover|(+.1)>`. Allowed syntax is `(`⟨sign⟩⟨num⟩`)` where ⟨sign⟩ is `+` or `-` to increment or decrement from the predefined value. Also allowable are `=` or ⟨empty⟩ to set the parameter position to ⟨num⟩, which must lie between 0 and 1 to have any meaning.

This parses adjustments of the `\xyknotPLACE`.

```
\xydef@\xyadjustknotPLACE@i{%
 \ifx\xyknotPLACE\empty\DN@{\dimen@=.5\p@}%
 \else\DN@{\expandafter\xygetknotPLACE@@\xyknotPLACE}%
 \fi \next@ \relax \let\xywhichknotPLACE@@=\xysetknotPLACE@@
 \xyFN@\xyadjustknotPLACE@}

\xydef@\xyadjustknotPLACE@{\expandafter\ifx\space@\next
  \expandafter\DN@\space{\xyFN@\xyadjustknotPLACE@}%
 \else\addDASH@\ifx\next\DN@{\xyadjustknotPLACE@@}%
 \else\addPLUS@\ifx\next\DN@{\xyadjustknotPLACE@@}%
 \else\addEQ@\ifx\next\addEQ@\DN@{\dimen@=\z@ \xyadjustknotPLACE@@}%
 \else \DN@{\dimen@=\z@ \xyadjustknotPLACE@@}%
 \fi\fi\fi\fi \next@}

\xydef@\xyadjustknotPLACE@@#1@@{\advance\dimen@#1\p@
 \expandafter\xysetknotPLACE@\expandafter{\the\dimen@}}
```

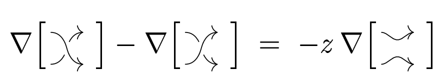
```
\xydef@\xysetknotPLACE@#1{%
 \expandafter\xywhichknotPLACE@@\expandafter{\removePT@#1}}
{\xyuncatcodes\gdef\next#1{\def\xyknotPLACE{(#1)}}}%
\xylet@\xysetknotPLACE@@=\next
{\xyuncatcodes \catcode`@=11
 \gdef\next(#1){\DN@\next@{#1}\ifx\next@\empty
  \dimen@=\z@\else\dimen@=#1\p@ \fi}}%
\xylet@\xygetknotPLACE@@=\next
```

These use the same mechanism to adjust parameters stored in `\xykpostbreak@@` and `\xykjoincontrol@@`.

```
%\xydef@\xyadjustJOIN@i#1@@{\xykjoincontrol@i{#1}}%
\xydef@\xyadjustJOIN@i{%
 \ifx\xykjoincontrol\empty\DN@{\dimen@=.75\p@}%
 \else\DN@{\expandafter\xygetknotPLACE@@\expandafter(\xykjoincontrol)}%
 \fi \next@ \relax \let\xywhichknotPLACE@@=\xysetjoincontrol@
 \xyFN@\xyadjustknotPLACE@}
%\xydef@\xyadjustBREAK@i#1@@{\def\xykpostbreak@@{|(##1)\knothole}}%
\xydef@\xyadjustBREAK@i{%
 \ifx\xykpostbreak@@\empty\DN@{\dimen@=.5\p@}%
 \else\DN@{\expandafter\xygetpostBREAK@\xykpostbreak@@}%
 \fi \next@ \relax \let\xywhichknotPLACE@@=\xysetpostbreak@
 \xyFN@\xyadjustknotPLACE@}
\xydef@\xygetpostBREAK@|#1\knothole{\DN@{#1}%
 \ifx\next@\empty\dimen@=.5\p@\else \xygetknotPLACE@@#1\fi}
\xydef@\xysetpostbreak@#1{\def\xykpostbreak@@{|(#1)\knothole}}
\xydef@\xysetjoincontrol@#1{\edef\xykjoincontrol{#1}}
```

Arrowheads can also be placed at either, or both, ends of of the strings forming a crossing. This is governed by a pair of booleans, initially `{FF}`. It is changed for *all* subsequent strings in a diagram by `\knottips{..}` where the recognised values are `{FF}`, `{FT}`, `{TF}` and `{TT}`, denoting tips (`T`) or not (`F`) at the start and end of each string. To add arrowtips at the start of strings in a particular crossing, append the 2-character combination `=<`; similarly `=>` adds tips at the ends, if not already requested. The combinations `==` and `=!` specify both (`{TT}`) and none (`{FF}`) respectively. These 2-character pairs can be mixed in with any specifications for labels and breaks, etc. Multiple pairs compound their effect; in particular `=<=>` gives the same result as `==`, while `=!=<` is needed to change `{FT}` into `{TF}`.

These are best used with single pieces, as in the following equation.



```
\UseComputerModernTips \knottips{FT}
\def\Conway#1{\mathord{\nabla\Bigl[\,
 \raise5pt\xybox{0;/r1pc/:#1}\,\Bigr]}}
$$
\Conway\htwist - \Conway\htwistneg
 \;=\; -z\,\Conway\huntwist $$
```

```
\xydef@\xykparsetips@{%
```

```
 \def\xyknotTIPS@@{}\def\xyknotPLACE{}%
 \let\xykparser@=\xykparsetips@@
 \xyFN@\xykparser@ }
```

```
\xydef@\xykparsetips@@{%
 \ifx\space@\next \expandafter\DN@\space{\xyFN@\xykparsetips@@}%
 \else\addLT@\ifx\next
  \ifx\xyknotTIPS@@\empty \def\xyknotTIPS@@{TF}%
  \else\DN@{FF}\ifx\next@\xyknotTIPS@@ \def\xyknotTIPS@@{TF}%
  \else\DN@{FT}\ifx\next@\xyknotTIPS@@ \def\xyknotTIPS@@{TT}%
  \fi\fi\fi
  \addLT@\DN@{\xyFN@\xykparser@}%
 \else\addGT@\ifx\next
  \ifx\xyknotTIPS@@\empty \def\xyknotTIPS@@{FT}%
  \else\DN@{FF}\ifx\next@\xyknotTIPS@@ \def\xyknotTIPS@@{FT}%
  \else\DN@{TF}\ifx\next@\xyknotTIPS@@ \def\xyknotTIPS@@{TT}%
  \fi\fi\fi
  \addGT@\DN@{\xyFN@\xykparser@}%
 \else\addEQ@\ifx\next \def\xyknotTIPS@@{TT}%
  \addEQ@\DN@{\xyFN@\xykparser@}%
 \else \ifx!\next \def\xyknotTIPS@@{FF}\DN@!{\xyFN@\xykparser@}%
 \else\DN@{\xyFN@\xykparser@}%
 \fi\fi\fi\fi\fi \next@ }
```

This reads the setting for tips on string ends.

```
\xydef@\xykcheckTIPS@{%
 \ifx\xyknotTIPS@@\empty \edef\xyknotTIPS@@{\xyknotTIPS@}\fi
      \DN@{FF}\ifx\next@\xyknotTIPS@@\def\tmp@{}%
 \else\DN@{FT}\ifx\next@\xyknotTIPS@@
  \expandafter\def\expandafter\tmp@\expandafter{\xykforetips@}%
 \else\DN@{TF}\ifx\next@\xyknotTIPS@@
  \expandafter\def\expandafter\tmp@\expandafter{\xykbacktips@}%
 \else
  \DN@##1\next@{\DN@{\xykforetips@##1}}%
   \expandafter\next@\xykbacktips@\next@
  \DNii@{\expandafter\def\expandafter\tmp@\expandafter}%
   \expandafter\nextii@\expandafter{\next@}%
 \fi\fi\fi \def\xyknotTIPS@@{}}
```

```
\xydef@\xyknottips@{%
 \ifx\tmp@\empty\addtotoks@{{}}\else
  \expandafter\addtotoks@\expandafter{\expandafter{\tmp@}}\fi}
```

Joins

3.10l. The "joins" are used to connect the loose ends of crossing strings. In particular "loops" and "caps" are for placing on ends of horizontal or vertical 'twist' and 'cross' crossings, leaving the current ⟨pos⟩ fixed. The "bends" join non-adjacent crossings of the same type, either horizontal or vertical.

The `\xcap..` pieces are designed to join adjacent `\xover..` pieces; they move c either vertically or horizontally, as appropriate. Finally the `\xbend..` pieces allow for smooth joins of 45° slopes to horizontal or vertical slopes. For these the actual positioning of the piece, see figure 3.9, is not entirely obvious.

Figure 3.9 displays the orientation on the joins. Also indicated are default positions for labels and arrow-tips; each piece uses the same code, e.g. `\vloop <>|>>><{x}|{y}>{z}`. Furthermore the current ⟨pos⟩ before the piece is drawn is marked using $^\circ$; that afterwards is indicated by $^\times$ or $^+$.

The ability to scale in size and place arrow-tips, breaks, labels etc. apply also to ⟨join⟩ pieces. The only difference is…

3.10m. The three places referred to by `<`,`|`,`>` are all on a single string. In particular `|` is always at the middle of the ⟨join⟩, whereas `<` and `>` are at *earlier* and *later* parameter values respectively. Any adjustments[3.10k] involving breaks should occur in increasing parameter order.

Parsing

```
\xydef@\xykparsejoin@{%
 \def\xykSCALE@@{}%
 \edef\xyknotPLACE{\xykmidPLACE@}%
 \let\xykparser@=\xykparsejoin@@
 \def\xykdefaultbreak@{\let\xykbreak@=\xykforetemp@
  \def\xyknotPLACE{(.5)}\appendtoholder@}%
%  \edef\xyknotPLACE{\xykmidPLACE@}\appendtoholder@}%
 \xyFN@\xykparsescale@ }
```

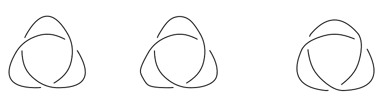
```
{\xyuncatcodes \catcode`@=11
\gdef\next{%
 \ifx\space@\next \expandafter\DN@\space{\xyFN@\xykparsejoin@@}%
 \else\addLT@\ifx\next \let\xykbreak@=\xykbacktemp@
  \def\xykdefaultbreak@{\def\xyknotPLACE{(.5)}%
%  \def\xykdefaultbreak@{\edef\xyknotPLACE{\xykmidPLACE@}%
  \appendtoholder@}\addLT@\DN@{\xyprecross@}%
 \else\addGT@\ifx\next \let\xykbreak@=\xykforetemp@
  \def\xykdefaultbreak@{\def\xyknotPLACE{(.5)}%
%  \def\xykdefaultbreak@{\edef\xyknotPLACE{\xykmidPLACE@}%
  \appendtoholder@}\addGT@\DN@{\xypostcross@}%
 \else\ifx|\next \let\xykbreak@=\xyknulltemp@
% \else\ifx|\next \let\xykbreak@=\xykholetemp@
  \def\xykdefaultbreak@{\def\xyknotPLACE{(.5)}%
%  \def\xykdefaultbreak@{\edef\xyknotPLACE{\xykmidPLACE@}%
  \appendtoholer@}\DN@|{\xymidcross@}%
 \else\addEQ@\ifx\next \addEQ@\DN@{\xyFN@\xykparsetips@@}%
% \else\addAT@\ifx\next \addAT@\DN@{\xykjoincontrol@}%
 \else\addAT@\ifx\next \addAT@\DN@(##1){%
  \xyadjustJOIN@i##1@@\xyFN@\xykparsejoin@@}%
 \else\DN@{\afterknot@}%
 \fi\fi\fi\fi\fi\fi \next@ }}
\xylet@\xykparsejoin@@=\next
```

3.10n. A parameter can be altered, using `@⟨adjust⟩`, to effect subtle adjustments to the shape of any join. Within a rectangular basis the horizontal or vertical tangents are preserved and overall reflection or rotation symmetry is preserved. Thus this parameter affects the 'flatness' of a cap or loop, or the amount of curvature is s-bends and z-bends. For `\xcap..`s and `\xbend..`s the 45° angle is altered; this is especially useful to match the tangents when a knot-piece has been specified using the technique of note 3.10c.

The normal range for these parameters is between 0 and 1. Other values can be used with interesting results—the parameter determines the location of control points for a Bézier cubic curve.

piece	value	effect on...
`\..cap`	.25	flatness of cap;
`\..loop`	.75	flatness of loop;
`\sbend..`	.75	curvature in the 's';
`\zbend..`	.75	curvature in the 'z';
`\xcap..`	.5	height of cap, slope at base;
`\xbend..`	.5	curvature, slope at base.

The following example gives three ways of specifying a 'trefoil' knot, using the `poly` feature to establish the location of the vertices for knot-pieces. In each the ⟨crossing⟩s are calculated to fit together smoothly; a different way of creating ⟨join⟩s is used in each. Also the third displays subtle changes of the [3.10n] join control.



```
\def\TrefoilA{\xygraph{!{0;/r.75pc/:}
 !P3"a"{~>{}}!P9"b"{~:{(1.3288,0):}~>{}}
 !P3"c"{~:{(2.5,0):}~>{}}
 !{\vover~{"b2"}{"b1"}{"a1"}{"a3"}}
 !{"b4";"b2"**\crv{"c1"}}
 !{\vover~{"b5"}{"b4"}{"a2"}{"a1"}}
 !{"b7";"b5"**\crv{"c2"}}
 !{\vover~{"b8"}{"b7"}{"a3"}{"a2"}}
 !{"b1";"b8"**\crv{"c3"}}}}
%
\def\TrefoilB{\xygraph{!{0;/r.75pc/:}
 !P3"a"{~>{}}!P9"b"{~:{(1.3288,0):}~>{}}
 !P3"c"{~:{(2.5,0):}~>{}}
 !{\vover~{"b2"}{"b1"}{"a1"}{"a3"}}
 !{\vcap~{"c1"}{"c1"}{"b4"}{"b2"}@(+.1)}
 !{\vover~{"b5"}{"b4"}{"a2"}{"a1"}}
 !{\vcap~{"c2"}{"c2"}{"b7"}{"b5"}@(+.2)}
 !{\vover~{"b8"}{"b7"}{"a3"}{"a2"}}
 !{\vcap~{"c3"}{"c3"}{"b1"}{"b8"}}}}
%
\def\TrefoilC{\xygraph{!{0;/r.75pc/:}
 !P3"a"{~>{}}
 !P12"b"{~:{(1.414,0):}~>{}}
```

```
 !{\vover~{"b2"}{"b1"}{"a1"}{"a3"}}
 !{\save 0;"b2"-"b5":"b5",
   \xcaph @(+.1)\restore}
 !{\vover~{"b6"}{"b5"}{"a2"}{"a1"}}
 !{\save 0;"b6"-"b9":"b9",
   \xcaph @(+.2)\restore}
 !{\vover~{"b10"}{"b9"}{"a3"}{"a2"}}
 !{\save 0;"b10"-"b1":"b1",
    \xcaph @(+.3)\restore} }}
$$\TrefoilA\quad\TrefoilB
   \quad\TrefoilC$$
```

```
\xydef@\xykjoincontrol@@{.75}%
\xydef@\xyk@joincontrol@#1{\DN@{#1}%
 \ifx\next@\empty\edef\xykjoincontrol@@{.75}%
 \else\dimen@=#1\p@
  \edef\xykjoincontrol@@{\expandafter\removePT@\the\dimen@}\fi}
\xylet@\xykjoincontrol\empty %\xyk@joincontrol@

\xydef@\xykjoincontrol@(#1){\DN@{#1}%
 \ifx\next@\empty\edef\xykjoincontrol{\xykjoincontrol@@}%
 \else\dimen@=#1\p@
  \edef\xykjoincontrol{\expandafter\removePT@\the\dimen@}%
 \fi \xyFN@\xykparsejoin@@ }

\xydef@\xykjoincontrol@i#1{\DN@{#1}%
 \ifx\next@\empty\edef\xykjoincontrol{\xykjoincontrol@@}%
 \else\dimen@=#1\p@
  \edef\xykjoincontrol{\expandafter\removePT@\the\dimen@}\fi}
```

These macros are common to most joins.

```
\xydef@\xykjoin@#1{%
 \expandafter\toks@\expandafter{\expandafter#1\knotSTYLE}}

\xydef@\xyksetjoin@{\xykcheckTIPS@
 \xykprebreak@ \xykmidbreak@ \xykpostbreak@ \xyknottips@ }
```

“cap” joins:

The initialisation…

```
\xydef@\hcap{\begingroup\def\afterknot@{\xyhcap}%
 \def\xykprePLACE@{(.385)}\def\xykpostPLACE@{(.615)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.25}%
 \xyFN@\xykparsejoin@ }%
\xydef@\vcap{\begingroup\def\afterknot@{\xyvcap}%
 \def\xykprePLACE@{(.385)}\def\xykpostPLACE@{(.615)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.25}%
 \xyFN@\xykparsejoin@ }%
\xydef@\xcapv{\begingroup\def\afterknot@{\xyxcapv}%
 \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.5}%
 \xyFN@\xykparsejoin@ }%
```

```
\xydef@\xcaph{\begingroup\def\afterknot@{\xyxcaph}%
 \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.5}%
 \xyFN@\xykparsejoin@ }%
```

The interface…

```
\xydef@\xyvcap{\xykjoin@\xykcapv \xyksetjoin@ \xykhobject@{}}
\xydef@\xyhcap{\xykjoin@\xykcaph \xyksetjoin@ \xykvobject@{}}
\xydef@\xyxcapv{\xykjoin@\xykxcapv \xyksetjoin@ \xykvobject@{+(0,-1)}}
\xydef@\xyxcaph{\xykjoin@\xykxcaph \xyksetjoin@ \xykhobject@{+(1,0)}}
```

The drawing code…

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "_>",{\xyksaveEdgec},{\xykz_\ar #1@'{"^>"**{}?(\xykjoincontrol)@+
 ,?-(.25,0)@+,"_<";"^<"**{}?+(.25,0)@+,?(\xykjoincontrol)@+}
 "_<"#3#4#5#6},\restore \POS#7}}
\xylet@\xykcapv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "_<",{\xyksaveEdgec},{\xykz_\ar #1@'{"_>"**{}?(\xykjoincontrol)@+,
  ?+(0,.25)@+,"^<";"^>"**{}?-(0,.25)@+,?(\xykjoincontrol)@+}
 "^<"#3#4#5#6}
 \restore \POS#7}}
\xylet@\xykcaph=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "^>";"_>"**{}?="^_","^<";"_<"**{}?;"^_",**{}?(\xykjoincontrol)="^_",
 "_<",{\xyksaveEdgec},{\xykz_\ar #1@'{"^_"@+}
  "^<"#3#4#5#6},\restore \POS#7}}
\xylet@\xykxcapv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "^<";"^>"**{}?="^_","_<";"_>"**{}?;"^_",**{}?(\xykjoincontrol)="^_",
 "_>",{\xyksaveEdgec},{\xykz_\ar #1@'{"^_"@+}
  "_<"#3#4#5#6},\restore \POS#7}}
\xylet@\xykxcaph=\next
```

"loop" joins:

The initialisation and interface…

```
\xydef@\hloop{\begingroup\def\afterknot@{\xyhloop}%
 \def\xykprePLACE@{(.07)}\def\xykpostPLACE@{(.93)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.75}%
 \xyFN@\xykparsejoin@ }%
\xydef@\vloop{\begingroup\def\afterknot@{\xyvloop}%
 \def\xykprePLACE@{(.07)}\def\xykpostPLACE@{(.93)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.75}%
 \xyFN@\xykparsejoin@ }%

\xydef@\xyvloop{\xykjoin@\xykloopv \xyksetjoin@ \xykhobject@{}}
\xydef@\xyhloop{\xykjoin@\xykxlooph \xyksetjoin@ \xykvobject@{}}
```

The drawing code…

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "_>",{\xykz_\ar #1@'{"^>"**{}?(\xykjoincontrol)@+,?(1.25)-(.25,0)@+,
  "_<";"^<"**{}?(1.25)+(.25,0)@+,?(\xykjoincontrol)@+}"_<"#3#4#5#6}
 \restore \POS#7}}
\xylet@\xykloopv=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "_<",{\xykz_\ar #1@'{"_>"**{}?(\xykjoincontrol)@+,?(1.25)+(0,.25)@+,
  "^<";"^>"**{}?(1.25)+(0,-.25)@+,?(\xykjoincontrol)@+}"^<"#3#4#5#6}
 \restore \POS#7}}
\xylet@\xykslooph=\next
```

"bend" joins:

The initialisation…

```
\xydef@\zbendh{\begingroup\def\afterknot@{\xyzbendh}%
 \def\xykprePLACE@{(.25)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}%
 \xyFN@\xykparsejoin@ }%
\xydef@\sbendv{\begingroup\def\afterknot@{\xysbendv}%
 \def\xykprePLACE@{(.25)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}%
 \xyFN@\xykparsejoin@ }%
\xydef@\sbendh{\begingroup\def\afterknot@{\xysbendh}%
 \def\xykprePLACE@{(.25)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}%
 \xyFN@\xykparsejoin@ }%
\xydef@\zbendv{\begingroup\def\afterknot@{\xyzbendv}%
 \def\xykprePLACE@{(.25)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}%
 \xyFN@\xykparsejoin@ }%
```

```
\xydef@\xbendr{\begingroup\def\afterknot@{\xyxbendr}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.5}%
 \xyFN@\xykparsejoin@ }%
\xydef@\xbendl{\begingroup\def\afterknot@{\xyxbendl}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.5}%
 \xyFN@\xykparsejoin@ }%
\xydef@\xbendu{\begingroup\def\afterknot@{\xyxbendu}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.5}%
 \xyFN@\xykparsejoin@ }%
\xydef@\xbendd{\begingroup\def\afterknot@{\xyxbendd}%
 \def\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.75)}%
 \def\xykmidPLACE@{(.5)}\edef\xykjoincontrol{.5}%
 \xyFN@\xykparsejoin@ }%
```

The interface…

```
\xydef@\xyzbendh{\xykjoin@\xykzbendh \xyksetjoin@ \xykhvobject@{+(1,-1)}}
\xydef@\xysbendv{\xykjoin@\xyksbendv \xyksetjoin@ \xykhvobject@{+(1,-1)}}
\xydef@\xysbendh{\xykjoin@\xyksbendh \xyksetjoin@ \xykhuobject@{+(1,1)}}
\xydef@\xyzbendv{\xykjoin@\xykzbendv \xyksetjoin@ \xykhuobject@{+(1,1)}}
```

```
\xydef@\xyxbendr{\xykjoin@\xykxbendr \xyksetjoin@ \xykhvobject@{+(1,-1)}}
\xydef@\xyxbendl{\xykjoin@\xykxbendl \xyksetjoin@ \xykhvobject@{+(0,-1)}}
\xydef@\xyxbendu{\xykjoin@\xykxbendu \xyksetjoin@ \xykhvobject@{+(1,0)}}
\xydef@\xyxbendd{\xykjoin@\xykxbendd \xyksetjoin@ \xykhvobject@{+(1,-1)}}
```

The implementation…

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "^<",{\xyksaveEdgec},
 {\xykz^\ar #1@'{"^>"**{}?(\xykjoincontrol)@+,
  "_>";"_<"**{}?(\xykjoincontrol)@+}"_>"#3#4#5#6}%
 \restore \POS#7}}
\xylet@\xykzbendh=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "^<",{\xyksaveEdgec},
 {\xykz_\ar #1@'{"_<"**{}?(\xykjoincontrol)@+,
  "_>";"^>"**{}?(\xykjoincontrol)@+}"_>"#3#4#5#6}%
 \restore \POS#7}}
\xylet@\xyksbendv=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "_<",{\xyksaveEdgec},
 {\xykz_\ar #1@'{"_>"**{}?(\xykjoincontrol)@+,
  "^>";"^<"**{}?(\xykjoincontrol)@+}"^>"#3#4#5#6}%
 \restore \POS#7}}
\xylet@\xyksbendh=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "_<",{\xyksaveEdgec},
 {\xykz^\ar #1@'{"^<"**{}?(\xykjoincontrol)@+,
  "^>";"_>"**{}?(\xykjoincontrol)@+}"^>"#3#4#5#6}%
 \restore \POS#7}}
\xylet@\xykzbendv=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "^<"-(0,.5),{\xyksaveEdgec},
 {\xykz_\ar #1@'{"_<";"_>"**{}?(\xykjoincontrol)@+,}"_>"#3#4#5#6}%
 \restore \POS#7}}
\xylet@\xykxbendr=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "^>"-(0,.5),{\xyksaveEdgec},
 {\xykz^\ar #1@'{"_>";"_<"**{}?(\xykjoincontrol)@+,}"_<"#3#4#5#6}%
 \restore \POS#7}}
\xylet@\xykxbendl=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "_<"+(.5,0),{\xyksaveEdgec},
 {\xykz_\ar #1@'{"_>";"^>"**{}?(\xykjoincontrol)@+,}"^>"#3#4#5#6}%
```

```
 \restore \POS#7}}
\xylet@\xykxbendu=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\save,
 "^<"+(.5,0),{\xyksaveEdgec},
 {\xykz^\ar #1@'{"^>";"_>"**{}?(\xykjoincontrol)@+,}"_>"#3#4#5#6}%
 \restore \POS#7}}
\xylet@\xykxbendd=\next
```

Changing the string-style

It is not necessary to use solid curves; any style available to curves and arrows can be chosen using…

`\knotstyle{`⟨char⟩`}`	use `\dir{`⟨char⟩`}`
`\knotstyles{`⟨char⟩`}{`⟨char⟩`}`	two styles
`\knotSTYLE{`⟨code⟩`}`	use ⟨code⟩

In each case the new style applies to *all* subsequent knot pieces, except that the two styles apply only to crossings. The latter case allows use of object ⟨modifier⟩s. The ⟨code⟩ consists of two groups `{..}{..}`, each containing ⟨arrow⟩ forms, as in 3.2 and notes 3.3m, 3.3r. Only the first ⟨arrow⟩ form is used with ⟨join⟩s whereas the two forms are used respectively with the two strings of a ⟨crossing⟩ in the order that they are drawn.

```
{\xyuncatcodes \gdef\next{{@{-}}{@{-}}}}
\xylet@\knotSTYLE=\next

{\xyuncatcodes \gdef\next#1{\def\knotSTYLE{{@{#1}}{@{#1}}}}}
\xylet@\knotstyle=\next

{\xyuncatcodes \gdef\next#1#2{\def\knotSTYLE{{@{#1}}{@{#2}}}}}
\xylet@\knotstyles=\next

{\xyuncatcodes \gdef\next#1{\def\knotSTYLE{#1}}}
\xylet@\KNOTstyle=\next
```

Initialisation: default values for the methods.

```
\xydef@\knotholesize@#1{\def\xykholesize{#1}}
\xydef@\xykholesize{5pt}
\xylet@\knotholesize=\knotholesize@
\xylet@\holesize=\knotholesize@

\xydef@\knottips#1{\def\xyknotTIPS@{#1}}
\xydef@\xyknotTIPS@{FF}
\xydef@\xyknotTIPS@@{}

\xydef@\xykSCALE@@{}

%\xydef@\xyknotbreak@@{}%
\xydef@\xykprebreak@@{}%
\xydef@\xykpostbreak@@{}%
\xydef@\xykmidbreak@@{}%

\xydef@\xykprePLACE@{(.25)}%
\xydef@\xykpostPLACE@{(.75)}%
\xydef@\xykmidPLACE@{}%
```

```
\xylet@\xykPLACE@=\xykmidPLACE@
```

```
{\gdef\next#1#2#3#4{%\message{#1:#2:#3:#4}%
 \save(0,0);"#2"-"#1"+"#4"-"#3":(.5,0):
 "#3"-"#1"+"#4"-"#2"::(0,.5)::}}
\xylet@\xykflexbase@=\next
\xylet@\xykflexbase=\xykflexbase@
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8{\xykflexbase{#1}{#2}{#3}{#4}%
% \save(0,0);"#2"-"#1"+"#4"-"#3":(.5,0):
% "#3"-"#1"+"#4"-"#2"::(0,.5)::
 "#1",{\ar @{#5}@'{"#1"+(0,.125)@++(.1875,.125)@+,
 "#4"-(.1875,.25)@++(.1875,.125)@+}"#4" #6},
 "#2",{\ar @{#5}@'{"#2"+(0,.125)@++(-.1875,.125)@+,
 "#3"+(.1875,-.25)@++(-.1875,.125)@+}"#3" #7}%
\restore \POS#8}}
\xylet@\Xtwist=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8{\xykflexbase{#1}{#2}{#3}{#4}%
% \save(0,0);"#2"-"#1"+"#4"-"#3":(.5,0):
% "#3"-"#1"+"#4"-"#2"::(0,.5)::
 "#1",{\ar @{#5}@'{"#1"+(0,.0625)@++(.1875,.0625)@+,
 "#3"+(.1875,-.125)@++(-.1875,.0625)@+}"#3" #6},
 "#2",{\ar @{#5}@'{"#2"+(0,.0625)@++(-.1875,.0625)@+,
 "#4"-(.1875,.125)@++(.1875,.0625)@+}"#4" #7}%
\restore \POS#8}}
\xylet@\Xuntwist=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\xykflexbase{#1}{#2}{#3}{#4}%
% \save(0,0);"#2"-"#1"+"#4"-"#3":(.5,0):
% "#3"-"#1"+"#4"-"#2"::(0,.5)::
 "#1",{\ar @{#5}@'{"#1"+(.1875,.0625)@++(.1875,.1875)@+,
 "#4"+(-#9,-.125)@+}"#4" #6},
 "#2",{\ar @{#5}@'{"#2"+(-.1875,.0625)@++(-.1875,.1875)@+,
 "#3"+(#9,-.125)@+}"#3" #7}%
\restore \POS#8}}
\xylet@\Xover=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\xykflexbase{#1}{#2}{#3}{#4}%
% \save(0,0);"#2"-"#1"+"#4"-"#3":(.5,0):
% "#3"-"#1"+"#4"-"#2"::(0,.5)::
 "#1",{\ar @{#5}@'{"#1"+(.1875,.0625)@++(.1875,.1875)@+,
 "#2"+(-.1875,.0625)@+}"#2" #6},
% "#3"+(#9,-.125)@+}"#3" #6},
 "#3",{\ar @{#5}@'{"#3"+(#9,-.125)@+,"#3"+(.25,-.25)@+,
% "#2",{\ar @{#5}@'{"#2"+(-.1875,.0625)@++(-.1875,.1875)@+,
 "#4"+(-#9,-.125)@+}"#4" #7}%
\restore \POS#8}}
\xylet@\Xunover=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7{\xykflexbase{#1}{#2}{#3}{#4}%
% \save(0,0);"#2"-"#1"+"#3"-"#4":(.5,0):
```

```
% "#4"-"#1"+"#3"-"#2"::(0,.5)::
 "#1",{\ar @{#5}@'{"#1"+(0,.125)@++(.125,.25)@+,
 "#4"+(-.125,.375)@++(.125,-.25)@+}"#4" #6},
\restore \POS#7}}
\xylet@\Xcap=\next
```

```
{\xyuncatcodes \catcode`@=11
 \gdef\next#1#2#3#4#5#6#7#8#9{\dimen@=#1\p@
 \ifdim\dimen@=\z@ \DN@{\Xunover{#2}{#3}{#4}{#5}{#8}{#6}{#7}{"#5"}}%
 \else\ifdim\dimen@>\z@
  \DNii@{#6}\ifx\nextii@\empty
    \DN@{\Xover{#2}{#3}{#4}{#5}{#8}{|(.45)\knothole}{#7}{"#5"}}%
   \else \DN@{\Xover{#2}{#3}{#4}{#5}{#8}{#6}{#7}{"#5"}}\fi
 \else
  \DNii@{#7}\ifx\nextii@\empty
   \DN@{\Xover{#2}{#3}{#4}{#5}{#8}{#6}{|(.45)\knothole}{"#5"}}%
  \else\DN@{\Xover{#2}{#3}{#4}{#5}{#8}{#6}{#7}{"#5"}}\fi
 \fi\fi
 \def\tmp@{{#9}}\ifx\tmp@\empty\def\tmp@{{0}}\else\dimen@=#9\p@\fi
 \expandafter\next@\tmp@ }}
\xylet@\XXover=\next
```

```
{\xyuncatcodes \catcode`@=11
 \gdef\next#1#2#3#4#5#6#7#8{\dimen@=#1\p@
 \ifdim\dimen@=\z@ \DN@{\Xuntwist{#2}{#4}{#3}{#5}{#8}{#6}{#7}{"#5"}}%
 \else\ifdim\dimen@>\z@
  \DNii@{#6}\ifx\nextii@\empty
    \DN@{\Xtwist{#2}{#4}{#3}{#5}{#8}{|(.45)\knothole}{#7}{"#5"}}%
   \else \DN@{\Xtwist{#2}{#4}{#3}{#5}{#8}{#6}{#7}{"#5"}}\fi
 \else
  \DNii@{#7}\ifx\nextii@\empty
   \DN@{\Xtwist{#2}{#4}{#3}{#5}{#8}{#6}{|(.45)\knothole}{"#5"}}%
  \else\DN@{\Xtwist{#2}{#4}{#3}{#5}{#8}{#6}{#7}{"#5"}}\fi
 \fi\fi \next@ }}
\xylet@\XXtwist=\next
```

simplified switch-like interface

```
{\xyuncatcodes \catcode`@=11
 \gdef\next#1#2#3#4{\dimen@=#1\p@
 \ifdim\dimen@=\z@ \DN@{\xykuncrossv{#4}{#2}{#3}{+(0,-1)}}%
 \else\ifdim\dimen@>\z@
  \DNii@{#2}\ifx\nextii@\empty
    \DN@{\xyktwistv{#4}{|\knothole}{#3}{+(1,0)}}%
   \else \DN@{\xyktwistv{#4}{#2}{#3}{+(1,0)}}\fi
 \else
  \DNii@{#3}\ifx\nextii@\empty
   \DN@{\xyktwistv{#4}{#2}{|\knothole}{+(0,-1)}}%
  \else\DN@{\xyktwistv{#4}{#2}{#3}{+(0,-1)}}\fi
 \fi\fi \next@ }}
\xylet@\Vcross=\next
```

```
{\xyuncatcodes \catcode`@=11
 \gdef\next#1#2#3#4{\dimen@=#1\p@
 \ifdim\dimen@=\z@ \DN@{\xykuncrossh{#4}{#2}{#3}{+(0,-1)}}%
 \else\ifdim\dimen@<\z@
  \DNii@{#2}\ifx\nextii@\empty
    \DN@{\xyktwisth{#4}{|\knothole}{#3}{+(1,0)}}%
   \else \DN@{\xyktwisth{#4}{#2}{#3}{+(1,0)}}\fi
 \else
  \DNii@{#3}\ifx\nextii@\empty
   \DN@{\xyktwisth{#4}{}{|\knothole}{+(1,0)}}%
  \else\DN@{\xyktwisth{#4}{#2}{#3}{+(1,0)}}\fi
 \fi\fi \next@ }}
\xylet@\Hcross=\next
```

```
{\xyuncatcodes \catcode`@=11
 \gdef\next#1#2#3{\dimen@=#1\p@ \save
 \ifdim\dimen@<\z@
  \DN@{\POS="@",p="@@",(0,0);(-1,0):(0,-1)::"@@";"@",}%
 \else\DN@{}\fi
 \next@ \xykcaph{#3}{#2}\restore }}
\xylet@\Hcap=\next
```

```
{\xyuncatcodes \catcode`@=11
 \gdef\next#1#2#3{\dimen@=#1\p@ \save
 \ifdim\dimen@<\z@
  \DN@{\POS="@",p="@@",(0,0);(0,-1)::"@@";"@",}%
 \else\DN@{}\fi
 \next@ \xykcapv{#3}{#2}\restore }}
\xylet@\Vcap=\next
```

The end & Log

```
\xyendinput
```

```
% $Log: xyknot.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  09:20:49  ross
% no changes
%
% Revision 3.1  1995/09/05 20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Created by Ross Moore, September 1994.
```

3.11 Smart Path option

Vers. 0.0 by George C. Necula ⟨necula@cs.cmu.edu⟩

Header:

```
%% $Id$
%%
%% Xy-pic ``Smart Path feature'' option.
%% Copyright (c) 1998          George C. Necula <necula@cs.cmu.edu>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1998  Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{smartpath}{Smart Path option}{\stripRCS$Revision: 0.0 $}%
 {George C. Necula}{necula@cs.cmu.edu}%
 {School of Computer Science,
  Carnegie Mellon University,
  5000 Forbes Avenue,
  Pittsburgh, PA 15213-3891, USA}
```

This extends the ‘arrow’ feature, which is therefore required, with a “smart” ⟨path⟩ between two ⟨pos⟩itions.

```
\xyrequire{arrow}\xycatcodes
```

The ⟨turn⟩ syntax is extended with the construction

⟨turn⟩ ⟶ `‘s` ⟨diag⟩ `_` ⟨diag⟩ ⟨turnradius⟩

`\ar`*in*`_`*out*`/5pt` which draws a connector leaving p in the *in* ⟨diag⟩onal direction and arrives at c in the *out* ⟨diag⟩onal direction, using `5pt` turns. The connector contains only horizontal or vertical lines and $\frac{1}{8}$ sectors of circles of the given (optional) ⟨turnradius⟩.

Bug: Any labels are placed at the end of the connection.

```
% The smart connection algorithm is as follows:
%
```

```
% A smart connector consists only of horizontal and vertical straight
% segments connected with 1/4 circles. 1/8 circles might be use at the start
% and end of the connector to reach a horizontal or a vertical.
%
% All possible connectors can be reduced to a series of equivalence classes,
% where two connectors are considered equivalent if they are identical after
% removing from each of them all the horizontal and vertical straight
% segments. Such a connector is called a representative.
%
% The algorithm first computes 10 representatives that start and end in the
% given directions and have at most one point where the orientation is
% changed. 5 of the representatives start counter-clockwise and the other
% 5 start clockwise. Among the counter-clockwise representatives, 1 does not
% have any change in orientation (one arc of circle starting and ending in the
% given directions). The other 4 are obtained by changing the orientation in
% the four points where the circle is tangent to a horizontal or a vertical.
%
% For each representative the algorithm determines if it can be extended with
% horizontal and vertical segments to reach the destination point when
% starting in the source point (the directions are already right by
% construction). For example, horizontal straight lines could be inserted
% in all the points where the representative is tangent to a horizontal line.
% In each such point a straight line segment can be inserted to extend the
% connector horizontally but only in the same direction of the connector at
% that point (the direction of the connetor in a point is towards the
% destination). For example, if the horizontal distance between the source and
% destination points is Dx and the horizontal distance between the start and
% the end of the representative connector is dx then the distance ``Dx - dx''
% must be covered using horizontal straight lines. This distance is
% distributed equally among all the horizontal extension points of the right
% direction. If no such points exist, then the representative is discarded.
% The cost of a connector is the length of the representative plus the total
% length of straight extension segments, i.e., |Dx-dx| + |Dy-dy|.
%
% The algorithm selects among all the representatives that are not discarded,
% the one leading to the shortest connection.

% Change the PATHturn@ to recognize `s as a smart PATH.
\let\origPATHturn@=\PATHturn@
\def\PATHturn@{%
 \ifx \space@\next \expandafter\DN@\space{\xyFN@\PATHturn@}%gobble spaces
 \else\ifx s\next
    \let\origPATHturn@i=\PATHturn@i\let\PATHturn@i=\PATHsmartturn@i
    \let\origPATHturn@cir=\PATHturn@cir\let\PATHturn@cir=\PATHsmartturn@cir
    \DN@ s{\xyFN@\origPATHturn@}% Let the original routines do the parsing
 \else
    \DN@{\origPATHturn@}%
 \fi\fi
 \next@}
```

```
% Changed PATHturn@cir to allow half or full turns
\xydef@\PATHsmartturn@cir{%
  \edef\next@{{\CIRin@@}{\expandafter\noexpand\CIRorient@@}{\CIRout@@}}%
  \expandafter\PATHturn@i\next@}

% This is the changed function for placing the turn
\xydef@\PATHsmartturn@i#1#2#3{%
 \DN@##1{%
   \def\PATHinit@@{%
     \xy@@{%
        \def\sm@CIRin{#1}\def\sm@CIRout{#3}% The IN and OUT directions. Ignore
                                             % the orientation
        \ifx\sm@CIRout\empty %Only on direction is given. Use it as OUT and
                             %use the previous direction as IN
           \let\sm@CIRout=\sm@CIRin
           \let\sm@CIRin=\PATHlastout@@
        \fi
        ##1\relax}%
     \xy@@{\enter@{\basefromthebase@}}%
     \xy@@{\sm@conn}% Draw the connection
     \xy@@{\X@p=\X@c \Y@p=\Y@c \czeroEdge@% Save the start of the segment
            \count@=\sm@CIRout\count@=\the\count@% Move forward a dash to
                                                 % touch the edge
            \dimen@=\xydashl@ \ABfromdiag@ \advance\X@c\A@ \advance\Y@c\B@
           \edef\PATHpostpos@@{\X@c=\the\X@c \Y@c=\the\Y@c \noexpand\czeroEdge@
                               \noexpand\PATHomitslide@@true}}%
     \xy@@{\leave@ %
           \edef\PATHlastout@@{\sm@CIRout}% Store the last direction
           \count@=\sm@CIRout \dimen@=\xydashl@ \Directionfromdiag@}}}%
 \expandafter\next@\expandafter{\the\toks@}\toks@={}%
 \let\PATHextra@@=\empty
 \def\PATHpost@@{\xy@@\PATHpostpos@@}%
 \let\PATHlabelsextra@@=\relax
 \let\PATHturn@i=\origPATHturn@i
 \let\PATHturn@cir=\origPATHturn@cir
 \xyFN@\PATHturn@ii}

\xydef@\sm@nil{}
\xydef@\sm@nnil{\sm@nil}
\xydef@\sm@maxcost{1000mm}% A very long connector

% This computes and draws the connection
\xydef@\sm@conn{
%  \xy@showthe p{Source}\xy@showthe c{Dest}%
%  \W@{Computing connector from p in \sm@CIRin\space to c in \sm@CIRout
%      .Radius is \turnradius@}%
  %Adjust the end points to the edge of the objects in the given directions
  \enter@{\cfromthec@ \DirectionfromtheDirection@}\cfromp@% Adjust p
  \count@=\sm@CIRin\count@=\the\count@
  \dimen@=\xydashl@ \ABfromdiag@ \advance\X@p\A@\advance\Y@p\B@
  \setupDirection@ \the\Edge@c\z@ \czeroEdge@
```

```
  \pfromc@
  \reverseDirection@
  % Now leave room for one dash to start with (for placing arrow tails)
  \count@=\sm@CIRin\count@=\the\count@
  \dimen@=\xydashl@ \ABfromdiag@
  % Drop the arrow tail in the right place
  \edef\tmp@{\noexpand\dir\artail@@}\expandafter\sm@drop@\tmp@
  \X@c=\X@p\advance\X@c\A@ \Y@c=\Y@p\advance\Y@c\B@ \czeroEdge@
  % Draw a straight connection from p to c
  \sm@stri
  % Set p to the end of the connection
  \X@p=\X@c \Y@p=\Y@c
  \leave@
%  \xy@showthe p{After adjust}%
  \enter@{\pfromthep@ \DirectionfromtheDirection@}\pfromc@% Adjust c
  \count@=\sm@CIRout\count@=\the\count@
  \dimen@=\xydashl@ \ABfromdiag@ \advance\X@p-\A@\advance\Y@p-\B@
  \setupDirection@ \the\Edge@c\z@
  \advance\X@c-\A@\advance\Y@c-\B@% Leave room for a dash to terminate the seg
  \czeroEdge@                % The final segment will be drawn by PATH@segment
  % Clear the tail setting for the last segment
  \def\PATHlabelsnext@@{}%
  \leave@
%  \xy@showthe c{After adjust}%
  % Set the turn radius
  \R@=\turnradius@
  \cirrestrict@@%Adjust the radius to fit the circles
  % Initialize variables
  \let\sm@circles=\sm@nil
  \edef\sm@bestcost{\sm@maxcost}% Best so far is very bad
  \sm@trycircles  1%Try counter clockwise first
  \sm@trycircles{-1}%Try clockwise after
%
% Now check the best
%
  \ifdim\sm@bestcost<\sm@maxcost
%    \W@{}\W@{Best: \meaning\sm@bestconn}%
    \cfromp@
%    \xy@showthe p{Source}%
    \expandafter\sm@conndraw\sm@bestconn
  \else
    \xyerror@{Cannot draw the smart connector}{}%
  \fi
}

% Compute and examine 5 representatives, for the orientation given
% as #1 (1 for counter-clockwise and -1 for clockwise).
\xydef@\sm@trycircles#1{%
  \ifnum #1>0% Compute the opposite orientation
     \def\sm@CIRorienti{-1}%
```

```
  \else
     \def\sm@CIRorienti{1}%
  \fi
  % Generate lists of triples (IN,ORIENT,OUT) so that OUT_1=IN_2 and
  % IN=IN_1 and OUT=OUT_2, and such that ORIENT_1=#1m ORIENT_2=-ORIENT_1
  % Then try each list in turn.
  %Try one circle first (a singleton list)
  \edef\tmp@{{{\sm@CIRin}{#1}{\sm@CIRout}}}%
  \expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
  %Now try lists of two circles.
  %Advance "in" so that it is a multiple of 90 degrees (that is, odd)
  \count@@=\sm@CIRin\count@@=\the\count@@\count@=\count@@
  \sm@roundcount@{#1}%
  \edef\sm@savecount@{\the\count@}%
  % See if we need to add a 1/8 starting circle
  \ifnum\the\count@=\the\count@@ \else
%
     \edef\tmp@{{{\sm@CIRin}{#1}{\the\count@}}%
                {{\the\count@}{\sm@CIRorienti}{\sm@CIRout}}}%
     \expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
  \fi
%
  \count@=\sm@savecount@\count@=\the\count@
  \sm@advancecount@ 2{#1}\edef\sm@savecount@{\the\count@}% Advance 90 deg.
  \edef\tmp@{{{\sm@CIRin}{#1}{\the\count@}}%
             {{\the\count@}{\sm@CIRorienti}{\sm@CIRout}}}%
  \expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
%
  \count@=\sm@savecount@\count@=\the\count@
  \sm@advancecount@ 2{#1}\edef\sm@savecount@{\the\count@}% Advance 90 deg.
  \edef\tmp@{{{\sm@CIRin}{#1}{\the\count@}}%
             {{\the\count@}{\sm@CIRorienti}{\sm@CIRout}}}%
  \expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
%
  \count@=\sm@savecount@\count@=\the\count@
  \sm@advancecount@ 2{#1}\edef\sm@savecount@{\the\count@}% Advance 90 deg.
  \edef\tmp@{{{\sm@CIRin}{#1}{\the\count@}}%
             {{\the\count@}{\sm@CIRorienti}{\sm@CIRout}}}%
  \expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
}

% Advance the direction in \count@ by #1 (each unit is 45deg) in the
% dirction #2, taking care of wrap-around
\xydef@\sm@advancecount@#1#2{%
%  \W@{Advancing count from \the\count@\space by #1\space
%                                                  in \xy@showorient #2}%
  \ifnum #2>0
     \edef\tmp@{#1}%
  \else
     \edef\tmp@{-#1}%
```

```
  \fi
  \advance\count@\tmp@\count@=\the\count@
  \ifnum\the\count@<0 \advance\count@ 8\fi
  \ifnum\the\count@>7 \advance\count@ -8\fi
  \count@=\the\count@
%  \W@{    Resulting count@=\the\count@}%
}

% Round counter to the next odd element (multiple of 90deg)
% in the given direction
\xydef@\sm@roundcount@#1{%
  \ifcase\the\count@
      \advance\count@ #1\or
  \or \advance\count@ #1\or
  \or \advance\count@ #1\or
  \or \advance\count@ #1\fi
  \count@=\the\count@
  \ifnum\the\count@<0 \advance\count@ 8\fi
  \ifnum\the\count@>7 \advance\count@ -7\fi
  \count@=\the\count@
}

\newif\ifsm@firstseg
\newif\ifsm@acceptable

% Given a representative described as a list of triplets ending with sm@nil,
% verify if is can be extended to the right destination
\xydef@\sm@trycirclelist#1{%
  \R@p=\z@\U@p=\R@p%                Clear the deltas
  \def\sm@exthp{0}\def\sm@exthm{0}%
  \def\sm@extvp{0}\def\sm@extvm{0}% Clear the extension points
  \def\sm@dxp{0pt}\def\sm@dxm{0pt}%
  \def\sm@dyp{0pt}\def\sm@dym{0pt}% Reset the extensions
  \def\sm@segs{}%                 % Clear the segment list
  \sm@firstsegtrue                % Mark the first segment
  \def\sm@connlen{0pt}%           % Initialize the connector length
  \let\sm@tryclcont=\sm@trycirclelist@i
  \sm@trycirclelist@i #1%Strip {} to expose the list elements
%
% The connector was processed
%
% Add an extra segment at then end, if ending in a multiple of 90 deg
  \count@@=\sm@CIRout\count@@=\the\count@\count@=\the\count@
  \sm@roundcount@{1}% Direction does not matter
  \ifnum\count@=\count@@
     \edef\sm@segs{\sm@segs{{\the\count@}{1}{\the\count@}}}%Accum segment
  \fi
%  \W@{After trying CL deltax=\the\R@p, deltay=\the\U@p}%
%  \sm@showext{\space}%
%  \W@{ \space Segs: \meaning\sm@segs}%
  %Now see if the connector can be extended enough
```

```
  \A@=\X@p\advance\A@\R@p\advance\A@ -\X@c\A@=-\A@ % A=Dx
  \B@=\Y@p\advance\B@\U@p\advance\B@ -\Y@c\B@=-\B@ % B=Dy
  \sm@acceptabletrue
%  \W@{ \space Dx=\the\A@, Dy=\the\B@}%
  \ifdim\A@>0pt
    \ifnum\sm@exthp>0 \dimen@=\A@ \divide\dimen@\sm@exthp
                      \edef\sm@dxp{\the\dimen@}%
    \else  \sm@acceptablefalse\fi\fi
  \ifdim\A@<0pt
    \ifnum\sm@exthm>0 \dimen@=\A@ \divide\dimen@\sm@exthm
                      \edef\sm@dxm{\the\dimen@}%
    \else  \sm@acceptablefalse\fi
    \A@=-\A@%Make it positive
  \fi
  \ifdim\B@>0pt
    \ifnum\sm@extvp>0 \dimen@=\B@ \divide\dimen@\sm@extvp
                      \edef\sm@dyp{\the\dimen@}%
    \else  \sm@acceptablefalse\fi\fi
  \ifdim\B@<0pt
    \ifnum\sm@extvm>0 \dimen@=\B@ \divide\dimen@\sm@extvm
                      \edef\sm@dym{\the\dimen@}%
    \else  \sm@acceptablefalse\fi
    \B@=-\B@
  \fi
  \ifsm@acceptable
     %Compute the cost of the connector
     \dimen@=\sm@connlen\advance\dimen@\A@\advance\dimen@\B@
%     \W@{ \space Cost is \the\dimen@}%
     %See if it is best so far
     \ifdim\dimen@<\sm@bestcost
        \edef\sm@bestcost{\the\dimen@}%
        \edef\sm@bestconn{{\sm@dxp}{\sm@dxm}{\sm@dyp}{\sm@dym}{\sm@segs}}%
     \fi
  \fi
}

\xydef@\sm@showext#1{%
  \W@{#1\space hp=\sm@exthp,hm=\sm@exthm,vp=\sm@extvp
                                         ,vm=\sm@extvm,len=\sm@connlen}}

% Process the next element in the list that describes the representative
%
\xydef@\sm@trycirclelist@i#1{%
  \def\@tmp{#1}%
  \ifx \@tmp\sm@nnil \let\sm@tryclcont=\relax\else
       \expandafter\sm@tryclcar\@tmp \fi
  \sm@tryclcont}

\xydef@\sm@tryclcar#1#2#3{%
%  \W@{Trying one turn in #1\space orient #2\space out #3}%
  %Compute deltax in R@p and deltay in U@p. Move them to center first.
```

```
  \dimen@=\ifnum #2<0 -\fi\R@
  \count@=#1\count@=\the\count@ \ABfromdiag@
  \advance\R@p -\B@ \advance\U@p \A@
%  \xy@showdim{   After center}%
  % Move them to exit
  \count@=#3\count@=\the\count@ \ABfromdiag@
  \advance\R@p \B@ \advance\U@p -\A@
%  \xy@showdim{   After turn}%
%  \xy@showthe p{          }%
  % Now update the extension points
  \sm@computeext{#1}{#2}{#3}%
}
%
% Compute the extensions
%
\xydef@\sm@computeext#1#2#3{%
%  \W@{Computing extents for turn in #1\space orient #2\space out #3}%
  \ifsm@firstseg
     \sm@accumext#1%
     \sm@firstsegfalse
  \fi
  \count@@=#1\count@@=\the\count@@\count@=\count@@
  \dimen@=\sm@connlen
%  \W@{Before rounding count@=\the\count@, count@@=\the\count@@}%
  \sm@roundcount@{#2}%
%  \W@{After rounding count@=\the\count@, count@@=\the\count@@}%
  \ifnum \count@=\count@@
    \sm@advancecount@ 1{#2}% Advance by 90 degrees if odd. By 45 first
    \advance\dimen@ 0.7854\R@% It was incremented with 1/8 circle
    \ifnum \count@=#3% Check if done
    \else
       \sm@advancecount@ 1{#2}% Another 45
       \advance\dimen@ 0.7854\R@% It was incremented with 1/8 circle
    \fi
  \else
    \advance\dimen@ 0.7854\R@% It was incremented with 1/8 circle
  \fi
  \edef\sm@connlen{\the\dimen@}% Save the connector length
  \sm@accumext{\the\count@}%
  \edef\sm@segs{\sm@segs{{#1}{#2}{\the\count@}}}%Accum segment
  \ifnum\the\count@=#3 \else
    \edef\tmp@{{\the\count@}{#2}{#3}}%Recursive call
    \expandafter\sm@computeext\tmp@
  \fi
}
%
% Accumulate the extensions
%
\xydef@\sm@accumext#1{%
```

```
% \W@{Accumulating extent for #1}%
% \sm@showext{Before accum:}%
 \ifcase #1%
   \or    \count@@=\sm@extvm\advance\count@@ by1%
          \edef\sm@extvm{\the\count@@}% 1 is V-
   \or\or \count@@=\sm@exthp\advance\count@@ by1%
          \edef\sm@exthp{\the\count@@}% 3 is H+
   \or\or \count@@=\sm@extvp\advance\count@@ by1%
          \edef\sm@extvp{\the\count@@}% 5 is V+
   \or\or \count@@=\sm@exthm\advance\count@@ by1%
          \edef\sm@exthm{\the\count@@}% 7 in H-
 \fi
% \sm@showext{After accum:}%
}
```

```
% Draw a computed connector.
% 1,2,3,4 - dx+, dx-, dv+, dv- (lengths of straight connectors in each
%                              direction)
% 5       - a list of segments, each enclosed in a pair of braces.
\xydef@\sm@conndraw#1#2#3#4#5{%
   \def\sm@contlist{\sm@drawseglist}%
   \edef\sm@dxp{#1}%
   \edef\sm@dxm{#2}%
   \edef\sm@dyp{#3}%
   \edef\sm@dym{#4}%Save deltas
   \sm@drawseglist #5\sm@nil}
\xydef@\sm@drawseglist#1{% Get the first segment
  \ifx #1\sm@nil \def\sm@contlist{}\else
     \sm@drawseg #1\fi
  \sm@contlist}
```

```
%
% Draw a segment
%
\xydef@\sm@drawseg#1#2#3{% A segment is CIRin+CIRorient+CIRout
%  \W@{Drawing segment in=#1\space out=\space #3}\xy@showthe c{Seg start}%
  \def\CIRin@@{#1}\def\CIRout@@{#3}%
  \sm@straight%See if a straight line is needed here. Insert it and
              %reset c if needed
  \ifnum\CIRin@@=\CIRout@@ \else
      % Compute the center of the circle
      \count@=\CIRin@@
      \dimen@=\ifnum #2<0 -\fi\R@
      \ABfromdiag@
      \advance\X@c -\B@ \advance\Y@c \A@
%      \xy@showthe c{Center of circle}%
%      \xy@showdim{Before center}%
      \ifnum #2>0 \def\CIRorient@@{\CIRacw@}%
            \else \def\CIRorient@@{\CIRcw@}\fi
      \drop@\literal@{\hbox\bgroup\cir@i}% Place the circle
```

```
      % Now move the the end of the arc
      \dimen@=\ifnum #2<0 -\fi\R@
      \count@=\CIRout@@\count@=\count@%Wierd. If I remove this last assignm
%                                     %\count@ is not set
      \ABfromdiag@
%      \xy@showthe c{Before advance}%
%      \xy@showdim{Before advance}%
      \advance\X@c\B@ \advance\Y@c-\A@
%      \xy@showthe c{After circle}%
%      \xy@showdim{After circle}%
  \fi
}
\xydef@\sm@straight{%
  \U@c=\z@\D@c=\U@c\L@c=\U@c\R@c=\U@c
  \pfromc@                    %Set p at the beginning of the segment
  \ifcase \CIRin@@ \or% 1 is V-
      \A@=\sm@dym\advance \Y@c\A@ %Wierd. I have to use the temporary A@!!
  \or\or              % 3 is H+
      \A@=\sm@dxp\advance \X@c\A@
  \or\or              % 5 is V+
      \A@=\sm@dyp\advance \Y@c\A@
  \or\or              % 7 is H-
      \A@=\sm@dxm\advance \X@c\A@ \fi
  \ifdim\X@c=\X@p\ifdim\Y@c=\Y@p\else \sm@stri \fi\else \sm@stri \fi
}
\xydef@\sm@stri{%
%   \W@{Drawing straight connection}\xy@showthe p{StrSrc}\xy@showthec{StrDst}%
    \edef\tmp@{\expandafter\noexpand\arstemprefix@@\arstem@@}%
    \expandafter\sm@connect@\tmp@}
\xydef@\sm@connect@#1#{%
    \DN@##1{\connect@{#1}{##1}}\next@}
\xydef@\sm@drop@#1#{%
    \DN@##1{\drop@{#1}{##1}}\next@}
% Compute and draw a connector. External interface
% 1 - in (for p)
% 2 - out (for c)
% 3 - suggested radius
\xydef@\smconn#1#2#3{%
  \edef\sm@CIRin{#1}\edef\sm@CIRout{#2}%Memorize the directions
  \R@=#3\R@=\the\R@
  \sm@conn}
\xyendinput
```

Bug: This code should probably be merged with the 'arrow' feature.
Finally the log.

```
% $Log$
% Received from George.
```

<table>
<tr><th colspan="3">Syntax</th><th>Action</th></tr>
<tr><td colspan="3"><code>\ar</code> ⟨arrow⟩ ⟨path⟩</td><td>make ⟨arrow⟩ along ⟨path⟩</td></tr>
<tr><td>⟨arrow⟩</td><td>⟶</td><td>⟨form⟩*</td><td>⟨arrow⟩ has the ⟨form⟩s</td></tr>
<tr><td>⟨form⟩</td><td>⟶</td><td><code>@</code> ⟨variant⟩</td><td>use ⟨variant⟩ of arrow</td></tr>
<tr><td></td><td>|</td><td><code>@</code> ⟨variant⟩ <code>{</code> ⟨tip⟩ <code>}</code></td><td>build arrow[3.3m] using ⟨variant⟩ of a standard stem and ⟨tip⟩ for the head</td></tr>
<tr><td></td><td>|</td><td><code>@</code> ⟨variant⟩ <code>{</code> ⟨tip⟩ ⟨conn⟩ ⟨tip⟩ <code>}</code></td><td>build arrow[3.3m] using ⟨variant⟩ of ⟨tip⟩, ⟨conn⟩, ⟨tip⟩ as arrow tail, stem, and head (in that order)</td></tr>
<tr><td></td><td>|</td><td><code>@</code> ⟨connchar⟩</td><td>change stem to the indicated ⟨connchar⟩</td></tr>
<tr><td></td><td>|</td><td><code>@!</code></td><td>dash the arrow stem by doubling it</td></tr>
<tr><td></td><td>|</td><td><code>@/</code> ⟨direction⟩ ⟨dist⟩ <code>/</code></td><td>curve[3.3o] arrow the ⟨dist⟩ance towards ⟨direction⟩</td></tr>
<tr><td></td><td>|</td><td><code>@(</code> ⟨direction⟩ <code>,</code> ⟨direction⟩ <code>)</code></td><td>curve to fit with in-out directions[3.3p]</td></tr>
<tr><td></td><td>|</td><td><code>@‘ {</code> ⟨control point list⟩ <code>}</code></td><td>curve setup[3.3q] with explicit control points</td></tr>
<tr><td></td><td>|</td><td><code>@[</code> ⟨shape⟩ <code>]</code></td><td>add <code>[</code>⟨shape⟩<code>]</code> to object ⟨modifier⟩s[3.3r] for all objects</td></tr>
<tr><td></td><td>|</td><td><code>@* {</code> ⟨modifier⟩* <code>}</code></td><td>add object ⟨modifier⟩s[3.3r] for all objects</td></tr>
<tr><td></td><td>|</td><td><code>@<</code> ⟨dimen⟩ <code>></code></td><td>slide arrow[3.3s] the ⟨dimen⟩</td></tr>
<tr><td></td><td>|</td><td><code>|</code> ⟨anchor⟩ ⟨it⟩</td><td>break each segment at ⟨anchor⟩ with ⟨it⟩</td></tr>
<tr><td></td><td>|</td><td><code>^</code> ⟨anchor⟩ ⟨it⟩ | <code>_</code> ⟨anchor⟩ ⟨it⟩</td><td>label each segment at ⟨anchor⟩ with ⟨it⟩</td></tr>
<tr><td></td><td>|</td><td><code>@?</code></td><td>reverse meaning of above and below[3.3t]</td></tr>
<tr><td>⟨variant⟩</td><td>⟶</td><td>⟨empty⟩ | <code>^</code> | <code>_</code>
| <code>0</code> | <code>1</code> | <code>2</code> | <code>3</code></td><td>
⟨variant⟩: plain, above, below, double, or triple</td></tr>
<tr><td>⟨tip⟩</td><td>⟶</td><td>⟨tipchar⟩*</td><td>directional named as the sequence of ⟨tipchar⟩s</td></tr>
<tr><td></td><td>|</td><td>⟨dir⟩</td><td>any ⟨dir⟩ectional[3.3n]</td></tr>
<tr><td>⟨tipchar⟩</td><td>⟶</td><td><code><</code> | <code>></code> | <code>(</code> | <code>)</code> | <code>|</code> | <code>’</code> | <code>‘</code> | <code>+</code> | <code>/</code></td><td>recognised tip characters</td></tr>
<tr><td></td><td>|</td><td>⟨letter⟩ | ⟨space⟩</td><td>more tip characters</td></tr>
<tr><td>⟨conn⟩</td><td>⟶</td><td>⟨connchar⟩*</td><td>directional named as the sequence of ⟨connchar⟩s</td></tr>
<tr><td></td><td>|</td><td>⟨dir⟩</td><td>any ⟨dir⟩ectional[3.3n]</td></tr>
<tr><td>⟨connchar⟩</td><td>⟶</td><td><code>-</code> | <code>.</code> | <code>~</code> | <code>=</code> | <code>:</code></td><td>recognised connector characters</td></tr>
</table>

Figure 3.2: ⟨arrow⟩s.

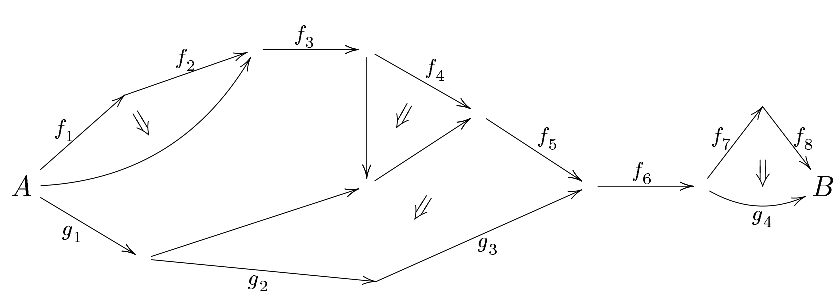


Figure 3.3: Pasting diagram.

Syntax			Action
⟨twocell⟩	⟶	⟨2-cell⟩⟨switches⟩⟨Arrow⟩	typeset ⟨2-cell⟩ with the ⟨switches⟩ and ⟨Arrow⟩
⟨2-cell⟩	⟶	`\..twocell`	typeset two curved arrows
	|	`\..uppertwocell`	typeset upper curved arrow only
	|	`\..lowertwocell`	typeset lower curved arrow only
	|	`\..compositemap`	use consecutive straight arrows
⟨Arrow⟩	⟶	`{`⟨tok⟩⟨text⟩`}`	specifies orientation and label
	|	`{`⟨nudge⟩⟨text⟩`}`	adjust position, use default orientation
	|	`{`⟨text⟩`}`	use default position and orientation
	|	`{`⟨tok⟩`*`⟨object⟩`}`	use ⟨object⟩ as the label
	|	`{`⟨nudge⟩`*`⟨object⟩`}` | `{*`⟨object⟩`}`	
⟨tok⟩	⟶	`^` | `_` | `=`	oriented anti-/clockwise/equality
	|	`\omit`	no Arrow, default is clockwise
	|	`` ` `` | `'` | `"` | `!`	no Arrow; tips on two curved arrows as: anti-/clockwise/double-headed/none
⟨switches⟩	⟶	⟨switch⟩⟨switches⟩	list of optional modifications
⟨switch⟩	⟶	⟨empty⟩	use defaults
	|	`^` ⟨label⟩	place ⟨label⟩ on the upper arrow
	|	`_` ⟨label⟩	place ⟨label⟩ on the lower arrow
	|	⟨nudge⟩	set the curvature, based on ⟨nudge⟩ value
	|	`\omit`	do not set the curved arrows
	|	`!`	place `\modmapobject` midway along arrows
	|	`~` ⟨what⟩ `{` ⟨object⟩ `}`	use ⟨object⟩ in place specified by ⟨what⟩
⟨what⟩	⟶	⟨empty⟩	set curves using the specified ⟨object⟩
	|	`^` | `_`	use ⟨object⟩ with upper/lower curve
	|	`` ` `` | `'`	use ⟨object⟩ for arrow head/tail
⟨label⟩	⟶	⟨text⟩ | ⟨nudge⟩ ⟨text⟩	set ⟨text⟩, displaced by ⟨nudge⟩
	|	`*`⟨object⟩ | ⟨nudge⟩`*`⟨object⟩	set ⟨object⟩, displaced by ⟨nudge⟩
⟨nudge⟩	⟶	`<`⟨number⟩`>`	use ⟨number⟩ in an appropriate way, *e.g.*, to position object or label along a fixed axis
	|	`<\omit>`	do not typeset the object/label

Figure 3.4: ⟨twocell⟩s

Syntax			Action
`\xygraph{`⟨graph⟩`}`			typeset ⟨graph⟩
⟨graph⟩	⟶	⟨step⟩*	interpret ⟨step⟩s in sequence
⟨step⟩	⟶	⟨node⟩	move[3.6a] to the ⟨node⟩
	$\mid$	`-`⟨arrow⟩ ⟨node⟩ ⟨labels⟩	draw[3.6b] line to ⟨node⟩, with ⟨labels⟩
	$\mid$	`:`⟨arrow⟩ ⟨node⟩ ⟨labels⟩	draw[3.6b] ⟨arrow⟩ to ⟨node⟩, with ⟨labels⟩
	$\mid$	`(` ⟨list⟩ `)`	map[3.6c] current node over ⟨list⟩
⟨node⟩	⟶	`[` ⟨move⟩ `]`	new node ⟨move⟩d relative to current
	$\mid$	`&` $\mid$ `\\`	new node in next column/row[3.6d]
	$\mid$	`"`⟨id⟩`"`	previously saved[3.6e] node
	$\mid$	`?`	currently mapped[3.6c] node
	$\mid$	⟨node⟩ ⟨it⟩	⟨node⟩ with ⟨it⟩ typeset and saved[3.6e] there
	$\mid$	⟨node⟩ `=` `"`⟨id⟩`"`	⟨node⟩ saved[3.6e] as `"`⟨id⟩`"`
	$\mid$	⟨node⟩ `!` ⟨escape⟩	augment node with material in another mode
⟨move⟩	⟶	⟨hop⟩*	⟨hop⟩s[3.6f] (`dulr`) from current node
	$\mid$	⟨hop⟩* ⟨place⟩ ⟨move⟩	do ⟨hop⟩s[3.6f] but use its ⟨place⟩ and ⟨move⟩ again
⟨list⟩	⟶	⟨graph⟩ `,` ⟨list⟩ $\mid$ ⟨graph⟩	list of subgraphs[3.6c]
⟨escape⟩	⟶	`{` ⟨pos⟩ ⟨decor⟩ `}`	perform ⟨pos⟩ ⟨decor⟩[3.6g]
	$\mid$	`M` ⟨matrix⟩	insert ⟨matrix⟩[3.6h]
	$\mid$	`P` ⟨polygon⟩	insert ⟨polygon⟩[3.6i]
	$\mid$	`E` ⟨ellipse⟩	insert ⟨ellipse⟩[3.6i]
	$\mid$	`~` ⟨setup⟩	setup parameters[3.6j]

Figure 3.5: ⟨graph⟩s

n	$\sin\frac{\pi}{n}$	$\cos\frac{\pi}{n}$	$\sin\frac{2\pi}{n}$	$\cos\frac{2\pi}{n}$	$\sin\frac{3\pi}{n}$	$\cos\frac{3\pi}{n}$	$\sin\frac{4\pi}{n}$	$\cos\frac{4\pi}{n}$	$\sin\frac{5\pi}{n}$	$\cos\frac{5\pi}{n}$
0	0	1	0	1	0	1	0	1	0	1
1	0	−1	0	1	0	−1	0	1	0	−1
2	1	0	0	−1	1	0	0	1	1	0
3	.8660254	.5	−.8660254	−.5	0	−1	−.8660254	−.5	−.8660254	.5
4	.70710678	.70710678	1	0	.70710678	−.70710678	0	−1	−.70710678	−.70710678
5	.587785	.809017	.951057	.309017	.951057	−.809017	.587785	−.809017	0	−1
6	.5	.8660254	.8660254	.5	1	0	−.8660254	−.5	.5	−.8660254
7	.433884	.900969	.781831	.62349	.974928	.222521	.974928	−.222521	.781831	−.62349
8	.382683	.92388	.70710678	.70710678	.92388	.382683	1	0	.92388	−.382683
9	.34202	.939693	.642788	.766044	.8660254	.5	.984808	.173648	.984808	−.173648
10	.309017	.951057	.587785	.809017	.309017	.951057	.951057	.309017	1	0
11	.281733	.959493	.540641	.841254	.75575	.654861	.909632	.415415	.989821	.142315
12	.258819	.965926	.5	.8660254	.70710678	.70710678	.8660254	.5	.965926	.258819

Figure 3.6: Trigonometry tables for Polygon vertices.

<table>
<tr><th colspan="3">Syntax</th><th>Action</th></tr>
<tr><td>⟨knot-piece⟩</td><td>⟶</td><td>⟨piece⟩⟨scale⟩⟨knot-labels⟩</td><td>interpret knot-piece</td></tr>
<tr><td>⟨piece⟩</td><td>⟶</td><td>⟨crossing⟩ | ⟨join⟩</td><td>piece is a crossing[3.10a] or a join[3.10l]</td></tr>
<tr><td>⟨scale⟩</td><td>⟶</td><td>⟨empty⟩ | <code>-</code> | <code>[</code>⟨num⟩<code>]</code></td><td>invert or scale the knot piece[3.10b];</td></tr>
<tr><td></td><td></td><td>| <code>~</code>⟨pos⟩⟨pos⟩⟨pos⟩⟨pos⟩</td><td>alter size and shape[3.10c] using the ⟨pos⟩s</td></tr>
<tr><td>⟨knot-labels⟩</td><td>⟶</td><td>⟨empty⟩ | ⟨knot-tips⟩⟨knot-labels⟩</td><td>arrowtips at ends, aligned with orientation</td></tr>
<tr><td></td><td></td><td>| ⟨where⟩⟨what⟩⟨knot-labels⟩</td><td>list[3.10k] of arrowtips, breaks and labels[3.10e]</td></tr>
<tr><td></td><td></td><td>| <code>@</code>⟨adjust⟩⟨knot-labels⟩</td><td>adjust hole[3.10d] position for a ⟨crossing⟩;
adjust other parameter[3.10n] for a ⟨join⟩.</td></tr>
<tr><td>⟨knot-tips⟩</td><td>⟶</td><td><code>==</code> | <code>=!</code></td><td>arrowtips[3.10k] at both/neither end</td></tr>
<tr><td></td><td></td><td>| <code>=<</code> | <code>=></code></td><td>arrowtips[3.10k] also at start/finish</td></tr>
<tr><td>⟨where⟩</td><td>⟶</td><td><code>|</code> | <code>|</code>⟨adjust⟩</td><td>‘over’ string on a ⟨crossing⟩;[3.10f]
middle[3.10m] place on a ⟨join⟩.</td></tr>
<tr><td></td><td></td><td>| <code><</code> | <code><</code>⟨adjust⟩</td><td>initial portion of ‘under’ string on a ⟨crossing⟩;[3.10f]
earlier[3.10m] place on a ⟨join⟩.</td></tr>
<tr><td></td><td></td><td>| <code>></code> | <code>></code>⟨adjust⟩</td><td>final portion of ‘under’ string on a ⟨crossing⟩;[3.10f]
later[3.10m] place on a ⟨join⟩.</td></tr>
<tr><td>⟨adjust⟩</td><td>⟶</td><td><code>(+</code>⟨num⟩<code>)</code> | <code>(-</code>⟨num⟩<code>)</code></td><td>adjustment[3.10k] from current value of parameter</td></tr>
<tr><td></td><td></td><td>| <code>(=</code>⟨num⟩<code>)</code> | <code>(</code>⟨num⟩<code>)</code></td><td>set parameter value[3.10k]</td></tr>
<tr><td>⟨what⟩</td><td>⟶</td><td><code>></code> | <code><</code></td><td>arrowhead aligned with/against orientation[3.10i]</td></tr>
<tr><td></td><td></td><td>| <code>\knothole</code> | <code>\khole</code></td><td>leave hole in the string[3.10j]</td></tr>
<tr><td></td><td></td><td>| <code>{</code>⟨text⟩<code>}</code></td><td>set[3.10g] ⟨text⟩ as label, using <code>\labelstyle</code></td></tr>
<tr><td></td><td></td><td>| <code>{*</code>⟨object⟩<code>}</code></td><td>drop ⟨object⟩[3.10h]</td></tr>
<tr><td></td><td></td><td>| <code>{</code>⟨anchor⟩⟨it⟩<code>}</code></td><td>⟨break⟩ or label[3.10h] as on an ⟨arrow⟩</td></tr>
<tr><td></td><td></td><td>| <code>|</code></td><td>null-break[3.10k]</td></tr>
</table>

Figure 3.7: ⟨knot-piece⟩ construction set

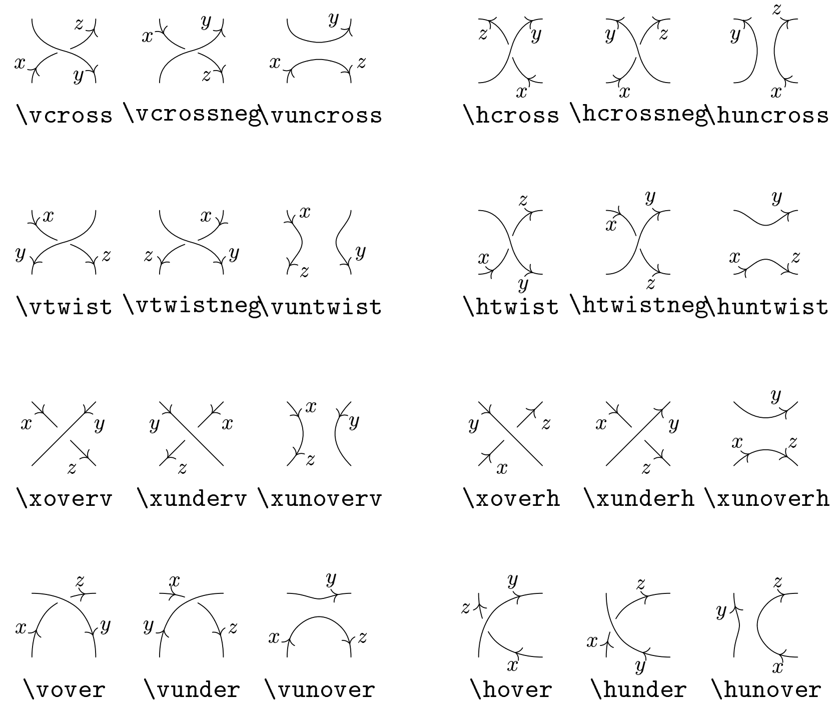


Figure 3.8: knot crossings with orientations and label positions

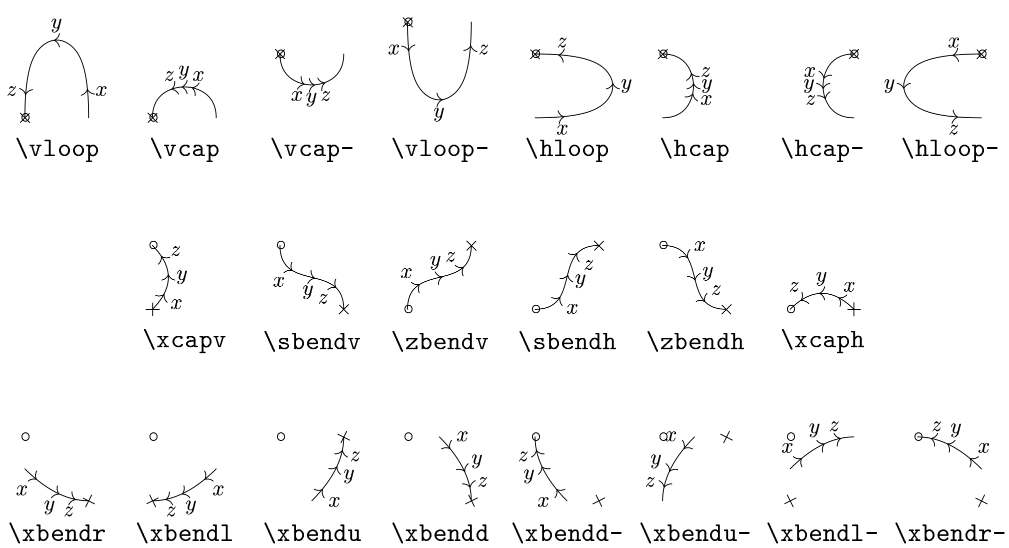


Figure 3.9: knot joins, with orientations, labels and shifts

Chapter 4

Drivers

This chapter describes the driver options that support customisation of the XY-pic output to particular output devices. The first section describes the actual drivers, the second a set of modifications specially applicable to PostScript printing.

4.1 Support for Specific Drivers

4.2 dvidrv driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This driver provides support for the "emtex" `\special` commands, when using one of the standard dvi-drivers: `dvidot`, `dvihplj`, `dvimsp`, `dviscr` or `dvivik`, that come with Eberhard Mattes' em-TeX distribution.

Header:

```
%% $Id: xydvidrv.doc,v 3.3 1996/12/18 09:12:40 ross Exp $
%%
%% Xy-pic ``dvidrv-driver'' option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
```

```
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{dvidrv}{dvidrv driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\dvidrv@@#1{\dvimessage@{dvidrv}{#1}}
\newdriver{%
 \xyaddsupport{emtex}\dvidrvEmTeX@@
 \xyaddsupport{line}\dvidrvLine@@
}
```

Driver installation Supported `\special` effects are...

- em-TeX line-drawing `\special`s.

```
\xydef@\dvidrvEmTeX@@{\dvidrvEmTeX@}%
\xydef@\dvidrvEmTeX@{\setupxyEMTeX@}%
```

- variable line-widths

```
\xydef@\dvidrvLine@@{\dvidrvLine@}%
\xydef@\dvidrvLine@{\xywithoption{emtex}{\installEMline@}}%
```

The end & Log

```
\xyendinput
% $Log: xydvidrv.doc,v $
% Revision 3.3  1996/12/18  09:12:40  ross
% no changes
%
% Revision 3.1  1995/09/05 20:28:57  ross
% Releasing version 3.1!
%
%
% NEW for version 3.1
```

4.3 DVIPS driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This driver provides support for *all extensions* when using the DVIPS driver by Tomas Rokicki [11]. It has been tested with dvips version 5.55a and dvipsk version 5.58f.

Header:

```
%% $Id: xydvips.doc,v 3.3 1996/12/18 09:13:14 ross Exp $
%%
```

```
%% Xy-pic ``dvips-driver'' option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{dvips}{DVIPS driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\dvips@@#1{\dvimessage@{dvips}{#1}}
```

```
\newdriver{%
 \xyaddsupport{color}\dvipsColor@@
 \xyaddsupport{crayon}\dvipsCrayola@@
 \xyaddsupport{ps}\dvipsPS@@
 \xyaddsupport{rotate}\dvipsRotate@@
 \xyaddsupport{line}\dvipsLine@@
 \xyaddsupport{frame}\dvipsFrames@@
 \xyaddsupport{tile}\dvipsTiles@@
 \xyaddsupport{tpic}\dvipsTPIC@
 \xyaddsupport{emtex}\dvipsEM@
 \xyaddsupport{cmtip}\relax
}
```

Driver installation Supported `\special` effects are...

- colour, using direct color specials and PostScript.

  DVIPS uses the colour-stack method, due to Tomas Rokicki.

```
\xydef@\dvips@Color@{\dvipsPStypes@
 \let\xylocalColor@=\xystackcolor@
 \let\xycolor@push\xycolor@push@@
 \def\xycolor@{\xycolor@@}\let\xycolorwarning@=\relax
```

```
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}\installPScolor@ }%
 \else\DN@{}\fi \next@
 \ifx\xycolor@@\xyNoColor@ \let\xycolor@@=\xycolor@raw@@\fi
 \ifx\xycrayolaloaded\empty \DN@{\xystandardcolors@\dvipsCrayola@{}}%
 \else \DN@{\xystandardcolors@}\fi \next@
 \let\dvipsColor@@=\relax
 \let\UnloadColor@@=\NoDVIPScolor@
 }%
```

```
\xydef@\dvipsColor@@{\dvips@Color@}
```

```
\xydef@\NoDVIPScolor@{%
 \def\xycolor@{\xycolor@@}%
 \let\xycolor@@=\xyNoColor@
 \let\xylocalColor@=\xyNoColor@@
 \let\xycolorwarning@=\xycolorwarning@@
 \let\dvipsColor@@=\dvips@Color@
 \let\UnloadColor@@=\UnloadColor@
 }
```

If neither support file `colordvi.tex` nor `colordvi.sty` can be found, then the normal colour support will not be available. However the colour support for the PostScript back-end can still be used.

```
\xydef@\NoDVIPScolordvi@{\xywarning@{%
 Neither  colordvi.tex  nor  colordvi.sty  could be found.}%
 \xyBEcolorcheck@}%
```

- crayon colours.

  The 68 colours that `dvips` recognizes by name are not loaded unless the `crayon` option has been requested.

```
\xydef@\dvipsCrayola@@{\dvipsColor@@\dvipsCrayola@}
```

```
\xydef@\dvipsCrayola@{%
 \let\prevxycolor@=\newxycolor@
  \def\newxyDVIPScolor@##1##2##3{\prevxycolor@{##1}{}{##1}}%
  \let\newxycolor@=\newxyDVIPScolor@ \dvipsInclude@{color.pro}%
%  \xyinputorelse@{colordvi.tex}%
%   {\NoDVIPScolordvi@\let\newxycolor@=\prevxycolor@}%
  \installCrayolaColors@
 \let\newxycolor@=\prevxycolor@ }%
```

Although any order of loading options: `ps`, `dvips`, `color` and `crayon` produces the desired result visually, the PostScript code can be different with different loading order. The most easily readable is obtained when `crayon` is requested last.

- PostScript back-end.

```
\xydef@\dvipsPS@@{\dvipsPS@}
```

```
\xydef@\dvipsPS@unload{\UnloadPS@
 \let\UnloadPS@@=\UnloadPS@
 \let\dvipsPS@@=\dvipsPS@ }
```

dvips has three kinds of PostScript \special which can be used. These may be used with various features other than just the PostScript backend.

```
\xydef@\dvipsPStypes@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \ifx\PSspecial@@\dvipsSpecial@\else
  \let\PSspecial@@=\dvipsSpecial@
  \let\PSmacro@@=\dvipsMacro@
  \let\PSdict@@=\dvipsDict@
  \let\PSspecialdict@@=\dvipsDict@
  \let\PSraw@@=\dvipsRaw@
  \let\PSinclude@@=\dvipsInclude@
  \let\xyPSobeylines@@=\obeyDVIPSlines@
  \let\xyPScurrpt@@=\dvipsCurrpt@@
  \dvips@@{PostScript}%
 \fi
% \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \XYpredict@ }
```

```
\xywarnifdefined\dvipsSpecial@
\xywarnifdefined\dvipsMacro@
\xywarnifdefined\dvipsDict@
\xywarnifdefined\dvipsInclude@
\xywarnifdefined\dvipsRaw@
{\catcode`\"=11 \catcode`\!=11 \catcode`\:=11
 \gdef\dvipsSpecial@#1{\special{ps:: SDict begin #1 end}}
 \gdef\dvipsMacro@#1{\firstPS@@\special{" {#1}xy}}
% \gdef\dvipsDict@#1{\special{! mark{#1}stopped cleartomark}}
 \gdef\dvipsDict@#1{\special{! #1}}
 \gdef\dvipsInclude@#1{\special{header=#1}}
 \gdef\dvipsRaw@#1{\special{ps: SDict begin #1 end}}
```

```
\catcode`\^^M=\active %
 \gdef\next{\catcode`\^^M=\active \let^^M=\space }}%
\xylet@\obeyDVIPSlines@=\next
```

All the PostScript definitions passed to dvips by reading the dictionary file are stored in a PostScript dictionary which it defines, called SDict. This must be the current dictionary whenever an XY-ps command is to be executed. This will be the case whenever the \special{" *commands*} or \special{! *commands*} forms are used, but not when the \special{ps:: *commands*} form is used. Thus the \dvipsSpecial@ macro includes code to open SDict and close it when finished; so also does the \dvipsRaw@ type, since this is used with XY-pic PostScript operators whose definitions have been placed within SDict.

The currentpoint is defined for the "raw" specials.

```
\xydef@\dvipsCurrpt@{\let\xyPScurrpt@@=\dvipsCurrpt@@}%
\xydef@\dvipsCurrpt@@{\PSraw@@{xyp}}
```

The PostScript operator called xyp is defined in xyps-ps for storing the location read from currentpoint.

This installs the PostScript backend.

```
\xydef@\dvipsPS@{\dvipsPStypes@
```

```
 \let\dvipsPS@@=\PSspecials@true
 \let\UnloadPS@@=\dvipsPS@unload
 \let\installxyps@@=\relax \installxyps@ }
```

- rotated/scaled diagrams and text, using PostScript.

```
\xydef@\dvipsRotate@@{\dvipsRotate@}
\xydef@\dvipsRotate@{\dvipsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%
  \dvipsCurrpt@ \dvipsRotScale@ \installPSrotscale@ }%
 \else\DN@{\xydriverfail@{rotations are}UnloadRotate@\relax}\fi
 \next@ }
\xydef@\dvipsRotScale@{%
 \def\xyrot@start(##1){{xyct ##1\space xyrs }}%
 \def\xyrot@end{}%
 \def\xyscale@start(##1,##2){{xyct ##1\space##2\space xyss }}%
 \def\xyscale@end{}}
```

- variable line-widths and poly-lines, using PostScript.

```
\xydef@\dvipsLine@@{\dvipsLine@}
\xydef@\dvipsLine@{\dvipsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}\installPSline@ }%
 \else\DN@{\xydriverfail@{line-widths are}\UnloadLine@\relax}\fi
 \next@ }
```

- extra frames and fills, using PostScript.

```
\xydef@\dvipsFrames@@{\dvipsFrames@}
\xydef@\dvipsFrames@{\dvipsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}\installPSframes@ }%
 \else\DN@{\xydriverfail@{frames are}\UnloadFrames@\relax}\fi
 \next@ }
```

- patterns and tiles, using PostScript.

```
\xydef@\dvipsTiles@@{\dvipsTiles@}
\xydef@\dvipsTiles@{\dvipsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}\installPSpatterns@ }%
 \else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
 \next@ }
```

- tpic drawing commands.

```
\xydef@\dvipsTPIC@@{\dvipsTPIC@}
\xydef@\dvipsTPIC@{\setupxyTPIC@}
```

- em-TEX drawing commands.

```
\xydef@\dvipsEM@@{\dvipsEM@}
\xydef@\dvipsEM@{\setupxyEMTeX@}
```

The end & Log

```
\xyendinput
```

```
% $Log: xydvips.doc,v $
% Revision 3.3  1996/12/18  09:13:14  ross
% minor improvements to file-loading commands
%
% Revision 3.2  1995/09/19  18:20:20  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

4.4 DVITOPS driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This file provides support for the DVITOPS driver by James Clark. As of September 1995, it has not been fully tested.

Header:

```
%% $Id: xydvitops.doc,v 3.3 1996/12/18 09:14:36 ross Exp $
%%
%% Xy-pic ``DVITOPS-driver'' option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
```

```
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{dvitops}{DVITOPS driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\dvitops@@#1{\dvimessage@{DVITOPS}{#1}}
```

```
\newdriver{%
 \xyaddsupport{color}\dvitopsColor@@
 \xyaddsupport{crayon}\dvitopsCrayon@@
 \xyaddsupport{ps}\dvitopsPS@@
 \xyaddsupport{rotate}\dvitopsRotate@@
 \xyaddsupport{line}\dvitopsLine@@
 \xyaddsupport{frame}\dvitopsFrames@@
 \xyaddsupport{tile}\dvitopsTiles@@
% \xyaddsupport{tpic}\dvitopsTPIC@@
 \xyaddsupport{cmtip}\relax
}
```

Supported `\special` effects are...

- colour, using direct color specials for `gray`, `rgb` and `hsb` colour models; and PostScript colour within diagrams;

  dvitops has a method for using colours locally, however it only colours "regions", which cannot be nested. To use this within XY-pic diagrams would be just asking for trouble.

```
\xydef@\dvitopsColor@@{\dvitops@Color@}
```

```
\xydef@\dvitops@Color@{\dvitopsPStypes@
 \let\xylocalColor@=\xyDVITOPScolor@
 \let\xycolor@push\xycolor@push@@
 \def\xycolor@{\xycolor@@}\let\xycolorwarning@=\relax
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}\xycatcodes
  \installPScolor@ }\else\DN@{}\fi \next@
 \ifx\xycolor@@\xyNoColor@ \let\xycolor@@=\xycolor@raw@@\fi
 \ifx\xycrayolaloaded\empty \DN@{\xystandardcolors@\dvitopsCrayon@{}}%
 \else\let\next@=\xystandardcolors@
 \fi \next@
 \let\dvitops@Color@@=\relax
 \let\UnloadColor@@=\NoDVITOPScolor@
 }%
```

```
\xydef@\NoDVITOPScolor@{%
```

```
 \def\xycolor@{\xycolor@@}%
 \let\xycolor@@=\xyNoColor@
 \let\xylocalColor@=\xyNoColor@@
 \let\xycolorwarning@=\xycolorwarning@@
 \let\dvitops@Color@@=\dvitops@Color@
 }
```

```
\xydef@\xyDVITOPScolor@{\bgroup \DN@##1##2{\egroup}%
 \ifInvisible@\else\ifHidden@\else\DNii@{\no@@}\ifx\nextii@\Connect@@
 \else\DN@{\xyDVITOPScolor@i}\fi\fi\fi \next@ }
```

```
\xydef@\xyDVITOPScolor@i#1#2{\checkXyStyle@
 \DN@{#2}\ifx\next@\empty
  \def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{#1}}}%
 \else\DNii@{gray}\ifx\next@\nextii@
  \def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{gray}{#1}}}%
 \else\DNii@{rgb}\ifx\next@\nextii@
  \def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{rgbcolor}{#1}}}%
 \else\DNii@{hsb}\ifx\next@\nextii@
  \def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{hsbcolor}{#1}}}%
 \else \DN@{\xywarning@{DVITOPS cannot support color: #2 #1}}%
 \fi\fi\fi\fi
 \toks@={\egroup \let\xy@style@=\relax \def\Drop@@}%
 \expandafter\tmp@\expandafter{\Drop@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\next@}\def\Connect@@}%
  \expandafter\tmp@\expandafter{\Connect@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\next@}}%
 \the\toks@ }%
```

```
\xydef@\xycolor@begin{\dvitopsBegin@{color}}
\xydef@\xycolor@end{\dvitopsEnd@{color}\dvitopsColor@x}
```

- crayon colours.

  The 68 colours are those that `dvips` recognizes by name, thanks to Tomas Rokicki.

```
\xydef@\dvitopsCrayon@@{\dvitopsColor@@\dvitopsCrayon@}
\xydef@\dvitopsCrayon@{\installCrayolaColors@}%
```

- PostScript back-end.

```
\xydef@\dvitopsPS@@{\dvipsPS@}
```

```
\xydef@\dvitopsPS@unload{\UnloadPS@
 \let\UnloadPS@@=\UnloadPS@
 \let\dvitopsPS@@=\dvitopsPS@ }
```

dvitops has two kinds of `\special` which can be used; 'inline' and 'prolog' for including header files. Also there are the 'begin' and 'end' types for defining regions.

```
\xydef@\dvitopsPStypes@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \ifx\PSspecial@@\dvitopsSpecial@\else
  \let\PSspecial@@=\dvitopsSpecial@
  \let\PSmacro@@=\dvitopsMacro@
```

```
  \let\PSdict@@=\dvitopsDict@
  \let\PSspecialdict@@=\dvitopsDict@
  \let\PSraw@@=\dvitopsRaw@
  \let\PSinclude@@=\dvitopsInclude@
  \let\xyPSobeylines@@=\obeyDVIPSlines@
  \let\xyPScurrpt@@=\dvitopsCurrpt@@
  \dvitops@@{PostScript}%
 \fi
% \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \XYpredict@ }
```

```
\xywarnifdefined\dvitopsSpecial@
\xywarnifdefined\dvitopsMacro@
\xywarnifdefined\dvitopsDict@
\xywarnifdefined\dvitopsInclude@
\xywarnifdefined\dvitopsRaw@
\xywarnifdefined\dvitopsOrigin@
\xywarnifdefined\dvitopsBegin@
\xywarnifdefined\dvitopsEnd@
\xywarnifdefined\dvitopsRotate@x
\xywarnifdefined\dvitopsTransform@x
\xywarnifdefined\dvitopsColor@x
{\catcode`\:=11
 \gdef\dvitopsSpecial@#1{\special{dvitops: inline #1}}
 \gdef\dvitopsMacro@#1{\special{dvitops: inline {#1}xy}}
 \gdef\dvitopsDict@#1{\special{dvitops: inline #1}}
 \gdef\dvitopsInclude@#1{\special{dvitops: prolog #1}}
 \gdef\dvitopsRaw@#1{\special{dvitops: inline #1}}
%
 \gdef\dvitopsOrigin@#1{\special{dvitops: origin #1}}
 \gdef\dvitopsBegin@#1{\special{dvitops: begin #1}}
 \gdef\dvitopsEnd@#1{\special{dvitops: end #1}}
 \gdef\dvitopsRotate@x#1{\special{dvitops: rotate #1}}
 \gdef\dvitopsTransform@x#1{\special{dvitops: transform #1}}
 \gdef\dvitopsColor@x#1#2{\special{dvitops: #1 color #2}}
}
```

```
\catcode`\^^M=\active %
 \gdef\next{\catcode`\^^M=\active \let^^M=\space }}%
\xylet@\obeyDVIPSlines@=\next
```

Rotations and transformations use a different mechanism, so `\xyPScurrpt@@` can be set to `\relax`.

```
\xydef@\dvitopsCurrpt@{\global\let\xyPScurrpt@@=\relax}%
```

This installs the PostScript backend.

```
\xydef@\dvitopsPS@{\dvitopsPStypes@
 \let\installxyps@@=\relax \installxyps@ }
```

- rotated/scaled diagrams and text, using DVITOPS specials; however these may not be nested.

```
\xydef@\dvitopsRotate@@{\dvitopsRotate@}
```

```
\xydef@\dvitopsRotate@{\dvitopsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty
  \DN@{\xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%
  \installDVITOPSrotate@ }%
 \else\DN@{\xydriverfail@{rotations are}\UnloadRotate@\relax}\fi
 \next@ }
```

```
\xydef@\dvitopsRotScale@{%
 \def\xyrot@start{\dvitopsBegin@{rot}\dvitopsOrigin@{rot}}%
 \def\xyrot@end(##1){\dvitopsEnd@{rot}\dvitopsRotate@x{##1}}%
 \def\xyscale@start{\dvitopsBegin@{scal}\dvitopsOrigin@{scal}}%
 \def\xyscale@end(##1,##2){%
  \dvitopsEnd@{scal}\dvitopsTransform@x{##1 0 ##2 0}}%
 }
```

```
\xydef@\installDVITOPSrotate@{\installPSstyle@
 \let\xyRotate@@=\dvitopsRotate@
 \let\doSpecialRotate@@=\dvitopsSpecialRotate@
 \let\xyscale@@=\dvitopsScale@
 \dvitopsCurrpt@
 \dvitopsRotScale@ \xyPSshapes@true }
```

```
\xydef@\UnloadDVITOPSrotate@{\UnloadRotate@\relax}
```

```
 \xydef@\dvitopsRotate@#1{\xyPSsplitPS@
  {\dvitopsBegin@{rot}\dvitopsOrigin@{rot}}%
  {\dvitopsEnd@{rot}\dvitopsRotate@x{#1}}}%
```

```
 \xydef@\dvitopsScale@#1#2{\xyPSsplitPS@
  {\dvitopsBegin@{scal}\dvitopsOrigin@{scal}}%
  {\dvitopsEnd@{scal}\dvitopsTransform@x{#1 0 #2 0}}}
```

```
 \xydef@\dvitopsSpecialRotate@#1@@{\dvitopsRotate@{#1}}
```

- variable line-widths and poly-lines, using PostScript.

```
\xydef@\dvitopsLine@@{\dvitopsLine@}
```

```
\xydef@\dvitopsLine@{\dvitopsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}\installPSline@ }%
 \else\DN@{\xydriverfail@{line-widths are}\UnloadLine@\relax}\fi
 \next@ }
```

- extra frames and fills, using PostScript.

```
\xydef@\dvitopsFrames@@{\dvitopsFrames@}
```

```
\xydef@\dvitopsFrames@{\dvitopsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}\installPSframes@ }%
 \else\DN@{\xydriverfail@{frames are}\UnloadFrames@\relax}\fi
 \next@ }
```

- patterns and tiles, using PostScript

```
\xydef@\dvitopsTiles@{\dvitopsPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}\installPSpatterns@ }%
 \else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
 \next@ }
```

- TPIC drawing commands.

```
\xydef@\dvitopsTPIC@@{\dvitopsTPIC@}
```

```
\xydef@\dvitopsTPIC@{\setupxyTPIC@}
```

The end & Log

```
\xyendinput
```

```
% $Log: xydvitops.doc,v $
% Revision 3.3  1996/12/18  09:14:36  ross
% minor improvements to file-loading commands
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:28:57  ross
% Releasing version 3.1!
%
% NEW for version 3.
```

4.5 OzTeX driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This driver provides the necessary interface to support the PostScript back-end and other PostScript effects when using the DVI driver of versions 1.8+ of OzTEX by Andrew Trevorrow,[1] *Earlier versions of OzTEX should instead use the driver option* `\xyoption{17oztex}`.

Effects such as colour, line-thickness and rotated or scaled diagrams are only partially supported in that the effects cannot be applied to any text or symbols placed using fonts. This is due to the nature of OzTEX ⟨driver⟩, whose optimization of the placement of font-characters precludes the applicability of such effects. Furthermore the PostScript dictionary must be available in a file called `global.ps` or appended to the `OzTeXdict.pro`. However with version 1.8 and later of OzTEX, there is the alternative of using the `dvips` ⟨driver⟩, which does support all the PostScript effects available in XY-pic.

Note: To use XY-pic effectively with OzTEX requires changing several memory parameters. In particular a 'Big-TEX' is needed, along with an increase in the `pool_size` parameter. Explicit instructions are contained in the file `INSTALL.OzTeX` of the XY-pic distribution.

Header:

```
%% $Id: xyoztex.doc,v 3.3 1996/12/18 09:26:26 ross Exp $
%%
```

[1]OzTEX is a shareware implementation of TEX for Macintosh available from many bulletin boards and ftp sites; v1.5 and earlier versions were freeware. Email contact: ⟨akt@kagi.com⟩.

```
%% Xy-pic ``OzTeX'' driver interface option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{oztex}{OzTeX driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\OzTeX@@#1{\dvimessage@{OzTeX}{#1}}
```

```
\newdriver{%
 \xyaddsupport{color}\oztexColor@@
 \xyaddsupport{crayon}\oztexCrayon@@
 \xyaddsupport{ps}\oztexPS@@
 \xyaddsupport{rotate}\oztexRotate@@
 \xyaddsupport{line}\oztexLine@@
 \xyaddsupport{frame}\oztexFrames@@
 \xyaddsupport{tile}\oztexTiles@@
 \xyaddsupport{cmtip}\relax
}
```

Supported `\special` effects are...

- colour, using PostScript, but not of font-characters.

```
\xydef@\oztexColor@@{\OzTeXColours@}
```

```
\xydef@\OzTeXColours@{\OzTeXPStypes@
 \let\xylocalColor@=\xyNoColor@@ \def\xycolor@{\xycolor@@}%
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%\XYpredict@
  \xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}%
  \installPScolor@ }\else\DN@{}\fi \next@
 \ifx\xycrayolaloaded\empty \DN@{\xystandardcolors@\OzTeXCrayon@@{}}%
 \else\let\next@=\xystandardcolors@\fi \next@
```

```
 \let\xycolorwarning@=\relax
 \let\oztexColor@@=\relax
 \let\UnloadColor@@=\NoOzTeXColor@
}
\xydef@\NoOzTeXColor@{%
 \let\xylocalColor@=\xyNoColor@@
 \let\xycolor@=\xyNoColor@
 \let\xycolorwarning@=\xycolorwarning@@
 \let\oztexColor@@=\OzTeXColours@
 \let\UnloadColor@@=\UnloadColor@
 }
```

- crayon colours, similarly restricted.

  The 68 colours are those that `dvips` recognizes by name, thanks to Tomas Rokicki.

```
\xydef@\OzTeXCrayon@@{\installCrayolaColors@}
```

- PostScript back-end.

```
\xydef@\oztexPS@@{\oztexPS@}
\xydef@\oztexPSunload@{\UnloadPS@
 \def\xyPSdictname{}\UsePSdict@@false
 \let\installxyps@@=\relax
 \let\UnloadPS@@=\UnloadPS@
 \xyPSshapes@true
 \let\oztexPS@@=\oztexPS@ }
\xydef@\OzTeXSpecial@#1{\special{empty.ps {#1}xy}}
\xydef@\OzTeXMacro@#1{\special{empty.ps {#1}xy}}
\xydef@\OzTeXDict@#1{\special{empty.ps @bxy{#1}stopped pop @exy}}
\xydef@\OzTeXinclude@#1{\special{#1}}
\xydef@\OzTeXraw@#1{\special{empty.ps @bxy #1 @exy}}%
\xydef@\OzTeXPStypes@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \ifx\PSspecial@@\OzTeXSpecial@\else
  \let\PSspecial@@=\OzTeXSpecial@
  \let\PSmacro@@=\OzTeXMacro@
  \let\PSdict@@=\OzTeXDict@
  \let\PSraw@@=\OzTeXraw@
  \let\PSinclude@@=\OzTeXinclude@
  \let\xyPSobeylines@@=\obeyOzTeXlines@
  \let\PSspecialdict@@=\OzTeXDict@
  \OzTeX@@{PostScript}%
 \fi
% \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \OzTeXCurrpt@ }
{\catcode`\^^M=\active %
 \gdef\next{\catcode`\^^M=\active \let^^M=\space \obeyspaces}}
\xylet@\obeyOzTeXlines@=\next
```

The PostScript operators @bxy and @exy are defined in xyps-ps for bypassing the mechanism used in OzTeX to isolate code passed in \special commands. Although it is possible (albeit difficult) to devise PostScript to deduce the current location for use with the "raw" specials, this information cannot be used to do rotations and scaling as with other ⟨driver⟩s. Hence the value of \xyPScurrpt@@ is taken to be empty.

```
\xydef@\OzTeXCurrpt@{\let\xyPScurrpt@@=\OzTeXCurrpt@@}%
\xydef@\OzTeXCurrpt@@{\PSraw@@{xyp}}
```

This avoids dumping the dictionary into the dvi-file, making sure that instead it goes into a file, specified by \xyPSdictname, which defaults to global.ps.

```
\xydef@\oztexPS@{\OzTeXPStypes@
 \let\oztexPS@@=\relax
 \let\UnloadPS@@=\oztexPSunload@
 \let\installxyps@@=\@OzTeX@PS@
 \installxyps@ }
```

```
\xydef@\@OzTeX@PS@{\UsePSdict@@true
 \let\includePSmessage@=\xyOzTeXheader
 \ifx\undefined\xyPSdictname\def\xyPSdictname{global.ps}\fi}
```

```
\xydef@\xyOzTeXheader#1#2#3{\UsePSdict@@true}
```

The parameters #1#2#3 of \xyOzTeXheader gobble tokens that are unnecessary with OzTeX.

- variable line-widths and poly-lines, using PostScript.

```
\xydef@\oztexLine@@{\oztexLine@}
```

```
\xydef@\oztexLine@{\OzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%\XYpredict@
  \xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}\installPSline@ }%
 \else\DN@{\xydriverfail@{line-styles are}\UnloadLine@\relax}\fi
 \next@ }
```

- extra frames and fills, using PostScript.

```
\xydef@\oztexFrames@@{\OzTeXFrames@}
```

```
\xydef@\OzTeXFrames@{\OzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}\installPSframes@ }%
 \else\DN@{\xydriverfail@{frames are}\UnloadPSFrames@\relax}\fi
 \next@ }
```

- patterns and tiles, using PostScript.

```
\xydef@\oztexTiles@@{\OzTeXTiles@}
```

```
\xydef@\OzTeXTiles@{\OzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}\installPSpatterns@
  \xystandardpatterns@ }%
 \else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
 \next@ }
```

- rotated/scaled diagrams and text, recognised but not supported.

  In fact OzTEX cannot support rotations and scaling. Instead we go through all the motions of providing PostScript support, then set a flag that causes a limited number of warning messages to be generated as requests are made for these effects.

```
\xydef@\oztexRotate@@{\OzTeXRotate@}
```

```
\xydef@\OzTeXRotate@{\OzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%
  \OzTeXRotScale@ \installPSrotscale@ \xyPSshapes@false }%
 \else\DN@{\xydriverfail@{rotations are}\UnloadRotate@\relax}\fi
 \next@ }
```

This never worked properly. It requires too much hacking at Trevorrow's code for little gain, since text-rotation can never work anyway. Use the dvips option instead!

```
\xydef@\OzTeXRotScale@{%
 \def\xyrot@start(##1){%
  {xyp xyt ##1\space xyr}{OzXy begin gsave}}%
 \def\xyscale@start(##1,##2){%
  {xyp xyt ##1\space##2\space xys}{OzXy begin gsave}}%
 \def\xyrot@end{grestore end}\def\xyscale@end{grestore end}}
```

The default code will just do nothing as far as the output is concerned...

```
\let\OzTeXRotScale@=\xyRotScale@@
```

The end & Log

```
\xyendinput
```

```
% $Log: xyoztex.doc,v $
% Revision 3.3  1996/12/18  09:26:26  ross
% improvements to file-loading commands
% fixed missing @ bug
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

4.6 OzTeX v1.7 driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This option provides the necessary interface to support the PostScript back-end and other PostScript effects when using the DVI driver of version 1.7 of OzTEX by Andrew Trevorrow,[2] *Later versions of OzTEX should instead use the driver option* `\xyoption{oztex}`*.* Upgrading to version 1.9+ ofOzTEX is recommended.

Does not support rotations, scaling and coloured text within diagrams and the PostScript dictionary must be available in a file called `global.ps`.

Note: To use XY-pic effectively with OzTEX requires changing several memory parameters. In particular a 'Big-TEX' is needed, along with an increase in the `pool_size` parameter. Explicit instructions are contained in the file `INSTALL.OzTeX` of the XY-pic distribution.

Header:

```
%% $Id: xy17oztex.doc,v 3.3 1996/12/18 14:21:23 ross Exp $
%%
%% Xy-pic ``OzTeX v1.7'' driver interface option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{17oztex}{OzTeX v1.7 driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\OzTeX@@#1{\dvimessage@{OzTeX v1.7}{#1}}
\newdriver{%
 \xyaddsupport{color}\oldoztexColor@@
```

[2] OzTEX is a shareware implementation of TEX for Macintosh available from many bulletin boards and ftp sites; v1.5 and earlier versions were freeware. Email contact: ⟨akt@kagi.com⟩.

```
 \xyaddsupport{crayon}\oldoztexCrayon@@
 \xyaddsupport{ps}\oldoztexPS@@
 \xyaddsupport{rotate}\oldoztexRotate@@
 \xyaddsupport{line}\oldoztexLine@@
 \xyaddsupport{frame}\oldoztexFrames@@
 \xyaddsupport{tile}\oldoztexTiles@@
 \xyaddsupport{cmtip}\relax
}
```

Supported `\special` effects are...

- colour, using PostScript, but not of font-characters.

```
\xydef@\oldoztexColor@@{\oldOzTeXColours@}
```

```
\xydef@\oldOzTeXColours@{\oldOzTeXPStypes@
 \let\xylocalColor@=\xyNoColor@@ \def\xycolor@{\xycolor@@}%
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%\XYpredict@
  \xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}%
  \installPScolor@ }\else\DN@{}\fi \next@
 \ifx\xycrayolaloaded\empty \DN@{\xystandardcolors@\oldOzTeXCrayon@@{}}%
 \else\let\next@=\xystandardcolors@\fi \next@
 \let\xycolorwarning@=\relax
 \let\oldoztexColor@@=\relax
 \let\UnloadColor@@=\NooldOzTeXColor@
 }
```

```
\xydef@\NooldOzTeXColor@{%
 \let\xylocalColor@=\xyNoColor@@
 \let\xycolor@=\xyNoColor@
 \let\xycolorwarning@=\xycolorwarning@@
 \let\oldoztexColor@@=\oldOzTeXColours@
 \let\UnloadColor@@=\UnloadColor@
 }
```

- crayon colours, similarly restricted.

  The 68 colours are those that `dvips` recognizes by name, thanks to Tomas Rokicki.

```
\xydef@\oldOzTeXCrayon@@{\installCrayolaColors@}
```

- PostScript back-end.

```
\xydef@\oldoztexPS@@{\oldoztexPS@}
```

```
\xydef@\oldoztexPSunload@{\UnloadPS@
 \def\xyPSdictname{}\UsePSdict@@false
 \let\installxyps@@=\relax
 \let\UnloadPS@@=\UnloadPS@
 \xyPSshapes@true
 \let\oldoztexPS@@=\oldoztexPS@ }
```

```
\xydef@\oldOzTeXSpecial@#1{\special{empty.ps {#1}xy}}
\xydef@\oldOzTeXMacro@#1{\special{empty.ps {#1}xy}}
\xydef@\oldOzTeXDict@#1{\special{empty.ps @obxy{#1}stopped pop @eoxy}}
\xydef@\oldOzTeXinclude@#1{\special{#1}}
```

```
\xydef@\oldOzTeXraw@#1{\special{empty.ps @boxy #1 @eoxy}}%
\xydef@\oldOzTeXPStypes@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \ifx\PSspecial@@\oldOzTeXSpecial@\else
  \let\PSspecial@@=\oldOzTeXSpecial@
  \let\PSmacro@@=\oldOzTeXMacro@
  \let\PSdict@@=\oldOzTeXDict@
  \let\PSraw@@=\oldOzTeXraw@
  \let\PSinclude@@=\oldOzTeXinclude@
  \let\xyPSobeylines@@=\obeyoldOzTeXlines@
  \let\PSspecialdict@@=\oldOzTeXDict@
  \oldOzTeX@@{PostScript}%
 \fi
% \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \oldOzTeXCurrpt@ }
{\catcode`\^^M=\active %
 \gdef\next{\catcode`\^^M=\active \let^^M=\space \obeyspaces}}
\xylet@\obeyoldOzTeXlines@=\next
```

The PostScript operators `@boxy` and `@eoxy` are defined in `xyps-ps` for bypassing the mechanism used in OzTeX to isolate code passed in `\special` commands. Although it is possible (albeit difficult) to devise PostScript to deduce the current location for use with the "raw" specials, this information cannot be used to do rotations and scaling as with other ⟨driver⟩s. Hence the value of `\xyPScurrpt@@` is taken to be empty.

```
\xydef@\oldOzTeXCurrpt@{\let\xyPScurrpt@@=\OzTeXCurrpt@@}%
\xydef@\oldOzTeXCurrpt@@{}
```

This avoids dumping the dictionary into the dvi-file, making sure that instead it goes into a file, specified by `\xyPSdictname`, which defaults to `global.ps`.

```
\xydef@\oldoztexPS@@{\oldOzTeXPStypes@
 \let\installxyps@@=\old@OzTeX@PS@
 \installxyps@ }
% The parameter |#1| in this next macro is to gobble the |\dumpPSdict@|
% which would otherwise dump the dictionary into the dvi-file. However
% if |\xyPSdictname| is defined then this dump must occur into a file.
%
%\xydef@\old@OzTeX@PS@#1{%
% \let\includePSmessage@=\xyoldOzTeXheader
% \ifx\undefined\xyPSdictname\DN@{}%
% \else\DN@{#1}\fi\next@ \UsePSdict@@true}
\xydef@\old@OzTeX@PS@{\UsePSdict@@true
 \let\includePSmessage@=\xyoldOzTeXheader
 \ifx\undefined\xyPSdictname\def\xyPSdictname{global.ps}\fi}
\xydef@\xyoldOzTeXheader#1#2#3{\UsePSdict@@true}
```

The parameters `#1#2#3` of `\xyOzTeXheader` gobble tokens that are unnecessary with OzTeX.

- variable line-widths and poly-lines, using PostScript.

```
\xydef@\oldoztexLine@@{\oldoztexLine@}

\xydef@\oldoztexLine@{\oldOzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}\installPSline@ }%
 \else\DN@{\xydriverfail@{line-styles are}\UnloadLine@\relax}\fi
 \next@ }
```

- extra frames and fills, using PostScript.

```
\xydef@\oldoztexFrames@@{\oldOzTeXFrames@}

\xydef@\oldOzTeXFrames@{\oldOzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}\installPSframes@ }%
 \else\DN@{\xydriverfail@{frames are}\UnloadPSFrames@\relax}\fi
 \next@ }
```

- patterns and tiles, using PostScript.

```
\xydef@\oldoztexTiles@@{\oldOzTeXTiles@}

\xydef@\oldOzTeXTiles@{\oldOzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}\installPSpatterns@
  \xystandardpatterns@ }%
 \else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
 \next@ }
```

- rotated/scaled diagrams and text, recognised but not supported.

  In fact OzTeX cannot support rotations and scaling. Instead we go through all the motions of providing PostScript support, then set a flag that causes a limited number of warning messages to be generated as requests are made for these effects.

```
\xydef@\oldoztexRotate@@{\oldOzTeXRotate@}

\xydef@\oldOzTeXRotate@{\oldOzTeXPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%\XYpredict@
  \xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%
  \xyRotScale@@ \installPSrotscale@ \xyPSshapes@false }%
 \else\DN@{\xydriverfail@{rotations are}\UnloadRotate@\relax}\fi
 \next@ }
```

The end & Log

```
\xyendinput

% $Log: xy17oztex.doc,v $
% Revision 3.3  1996/12/18 14:21:23  ross
% Ross's version
%
% Revision 3.3.1.1  1996/12/18  08:47:45  ross
% *** empty log message ***
%
% Revision 3.2  1995/09/19  18:20:20  ross
```

```
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

4.7 Textures driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This driver provides support for version 1.7+ of Blue Sky Research's TEXTURES application for Macintosh[3]. It incorporates support for colour and all of XY-pic's POSTSCRIPT effects. Earlier versions of TEXTURES should instead use the driver option `\xyoption{16textures}`.

Notice that version 1.7 suffers from a printing bug which may cause a POSTSCRIPT error. A fix is kludged by making sure the first page has been shown in the viewer before any pages with diagrams are sent to the printer.

Header:

```
%% $Id: xytextures.doc,v 3.3 1996/12/18 09:41:45 ross Exp $
%%
%% Xy-pic ``Textures'' driver interface option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
```

[3]Macintosh is a trademark of Apple Computer Inc.

```
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{textures}{Textures driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\Textures@@#1{\dvimessage@{Textures}{#1}}
\newdriver{%
 \xyaddsupport{color}\texturesColor@@
 \xyaddsupport{crayon}\texturesCrayon@@
 \xyaddsupport{ps}\texturesPS@@
 \xyaddsupport{rotate}\texturesRotate@@
 \xyaddsupport{line}\texturesLine@@
 \xyaddsupport{frame}\texturesFrames@@
 \xyaddsupport{tile}\texturesTiles@@
 \xyaddsupport{cmtip}\relax
}
```

Supported `\special` effects are...

- colour, both on-screen and with PostScript

  Textures uses the colour-stack method, but it requires a separate `\special` to set the new colour after having pushed the previous colour onto the stack.

```
\xydef@\TexturesColor@push#1{\special{color push}\special{color #1}}
\xydef@\TexturesColours@{\texturesPStypes@
 \let\xylocalColor@=\xystackcolor@
 \let\xycolor@push=\TexturesColor@push
 \def\xycolor@{\xycolor@@}%
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}%
  \installPScolor@ }\else\DN@{}\fi \next@
 \ifx\xycolor@@\xyNoColor@ \let\xycolor@@=\xycolor@raw@@\fi
 \ifx\xycrayolaloaded\empty \DN@{\xystandardcolors@\texturesCrayon@@{}}%
 \else\let\next@=\xystandardcolors@ \fi \next@
 \let\xycolorwarning@=\relax \Textures@@{color}%
 \let\texturesColor@@=\relax
 \let\UnloadColor@@=\NoTexturesColor@
 }
\xydef@\texturesColor@@{\TexturesColours@}
\xydef@\NoTexturesColor@{%
 \let\xylocalColor@=\xyNoColor@@
 \let\xycolor@=\xyNoColor@
 \let\xycolorwarning@=\xycolorwarning@@
 \let\texturesColor@@=\TexturesColours@
 \let\UnloadColor@@=\UnloadColor@
 }
```

- crayon colours.

The 68 colours are those that `dvips` recognizes by name, thanks to Tomas Rokicki.

```
\xydef@\texturesCrayon@@{\texturesColor@@\texturesCrayon@}
\xydef@\texturesCrayon@{\installCrayolaColors@}%
```

- PostScript back-end.

```
\xydef@\texturesPS@@{\texturesPS@}
\xydef@\texturesPSunload@{\UnloadPS@
 \let\UnloadPS@@=\UnloadPS@
 \let\texturesPS@@=\texturesPS@ }
```

Versions 1.7 and later of Textures have the following kinds of `\special`, the first two executing in the environment defined by the dictionary `userdict`. In particular these versions of Textures now allow the PostScript dictionary to be loaded into the main working dictionary.

- `\special{rawpostscript #1}` puts code directly into the PostScript file.
- `\special{postscript #1}` wraps the code within `sps`...`eps`, which involves a basis change within a `save/restore` pair.
- `\special{prePostScript #1}` places the PostScript code within the header dictionary, called `TeXdict`.

Each of these has a corresponding version for reading the PostScript commands from a file.

```
\xydef@\texturesPStypes@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \ifx\PSspecial@@\TexturesSpecial@\else
  \let\PSspecial@@=\TexturesSpecial@
  \let\PSmacro@@=\TexturesMacro@
  \let\PSdict@@=\TexturesDict@
  \let\PSraw@@=\TexturesRaw@
  \let\PSinclude@@=\TexturesInclude@
  \let\PSspecialdict@@=\TexturesSpecialDict@
  \let\xyPSobeylines@@=\obeyTextureslines@
  \Textures@@{PostScript}%
 \fi \texturesCurrpt@ \XYpredict@ }

\xydef@\TexturesSpecial@#1{\firstPS@@\special{postscript #1}}
\xydef@\TexturesMacro@#1{\special{postscript {#1}xy}}
\xydef@\TexturesDict@#1{\special{prePostScript #1}}
\xydef@\TexturesSpecialDict@#1{\special{prePostScript #1}}
\xydef@\TexturesInclude@#1{\firstPS@@\special{prePostScriptfile #1}}
\xydef@\TexturesRaw@#1{\firstPS@@\special{rawpostscript #1}}%

{\catcode`\^^M=\active%
 \gdef\next{\catcode`\^^M=\active \let^^M=\par%
 \let\PSspecial@@=\TexturesDict@ \obeyspaces}}
\xylet@\obeyTextureslines@=\next
```

The `currentpoint` is defined for the “raw” specials. For Textures this is the same as `\xyPScurrpt@@@`, in `xyps-ps.doc`.

```
\xydef@\texturesCurrpt@{\let\xyPScurrpt@@=\texturesCurrpt@@}%
\xydef@\texturesCurrpt@@{\PSspecial@@{xyi}\PSraw@@{userdict begin xyx end}}
```

```
%\xydef@\texturesCurrpt@@{\PSspecial@@{%
% 0 0 transform grestore gsave itransform }%
% \PSraw@@{userdict begin /xyYpos exch def /xyXpos exch def end}}
```

The PostScript operator called `xyx` is defined in `xyps-ps` for storing the location placed on the stack by `xyi`.

This installs the PostScript backend.

```
\xydef@\texturesPS@{\texturesPStypes@
 \let\texturesPS@@=\relax
 \let\UnloadPS@@=\texturesPSunload@
 \let\installxyps@@=\relax \installxyps@ }
```

- rotated/scaled diagrams and text, using PostScript.

```
\xydef@\texturesRotate@@{\texturesRotate@}
```

```
\xydef@\texturesRotate@{\texturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%
  \xyRotScale@@ \installPSrotscale@ }%
 \else\DN@{\xydriverfail@{rotations are}\UnloadRotate@\relax}\fi
 \next@}
```

- variable line-widths and poly-lines, using PostScript.

```
\xydef@\texturesLine@@{\texturesLine@}
```

```
\xydef@\texturesLine@{\texturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}%
  \installPSline@ }%
 \else\DN@{\xydriverfail@{line-widths are}\UnloadLine@\relax}\fi
 \next@ }
```

- extra frames and fills, using PostScript.

```
\xydef@\texturesFrames@@{\texturesFrames@}
```

```
\xydef@\texturesFrames@{\texturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}%
  \installPSframes@ }%
 \else\DN@{\xydriverfail@{frames are}\UnloadPSFrames@\relax}\fi
 \next@ }
```

- patterns and tiles, using PostScript.

```
\xydef@\texturesTiles@@{\texturesTiles@}
```

```
\xydef@\texturesTiles@{\texturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}%
  \installPSpatterns@ \xystandardpatterns@}%
 \else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
```

```
\next@ }
```

The end & Log

```
\xyendinput
```

```
% $Log: xytextures.doc,v $
% Revision 3.3  1996/12/18  09:41:45  ross
% minor improvements to file-loading commands
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

4.8 Textures v1.6 driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This driver provides support for versions 1.5b and 1.6 of Blue Sky Research's TEXTURES application for Macintosh[4]. It incorporates support for POSTSCRIPT colour and the Xy-ps POSTSCRIPT back-end. This will *not* work with versions 1.7 and later; these require the ⟨driver⟩ option `\xyoption{textures}`.

Header:

```
%% $Id: xy16textures.doc,v 3.3 1996/12/18 14:21:23 ross Exp $
%%
%% Xy-pic ``Textures v1.6'' driver interface option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
```

[4]Macintosh is a trademark of Apple Computer Inc.

```
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{16textures}{Textures v1.6 driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\oldTextures@@#1{\dvimessage@{Textures v1.6}{#1}}
```

```
\newdriver{%
 \xyaddsupport{color}\oldtexturesColor@@
 \xyaddsupport{crayon}\oldtexturesCrayon@@
 \xyaddsupport{ps}\oldtexturesPS@@
 \xyaddsupport{rotate}\oldtexturesRotate@@
 \xyaddsupport{line}\oldtexturesLine@@
 \xyaddsupport{frame}\oldtexturesFrames@@
 \xyaddsupport{tile}\oldtexturesTiles@@
 \xyaddsupport{cmtip}\relax
}
```

Supported `\special` effects are...

- colour, using PostScript

  Textures v1.6 cannot manage colours, except within imported graphics. It can put colours into the PostScript output.

```
\xydef@\oldTexturesColours@{\oldtexturesPStypes@
 \let\xylocalColor@=\xyNoColor@@ \def\xycolor@{\xycolor@@}%
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}%
  \installPScolor@ }\else\DN@{}\fi \next@
 \ifx\xycolor@@\xyNoColor@ \let\xycolor@@=\xycolor@raw@@\fi
 \ifx\xycrayolaloaded\empty \DN@{\xystandardcolors@\oldtexturesCrayon@@{}}%
 \else\let\next@=\xystandardcolors@ \fi \next@
 \let\xycolorwarning@=\relax
 \let\oldtexturesColor@@=\relax
 \let\UnloadColor@@=\NooldTexturesColor@
 }
```

```
\xydef@\oldtexturesColor@@{\oldTexturesColours@}
```

```
\xydef@\NooldTexturesColor@{%
 \let\xylocalColor@=\xyNoColor@@
 \let\xycolor@=\xyNoColor@
 \let\xycolorwarning@=\xycolorwarning@@
 \let\oldtexturesColor@@=\oldTexturesColours@
```

```
 \let\UnloadColor@@=\UnloadColor@
 }
```

- crayon colours.

  The 68 colours are those that `dvips` recognizes by name, thanks to Tomas Rokicki.

```
\xydef@\oldtexturesCrayon@@{\oldtexturesColor@@\oldtexturesCrayon@}
\xydef@\oldtexturesCrayon@{\installCrayolaColors@}%
```

- PostScript back-end.

```
\xydef@\oldtexturesPS@@{\oldtexturesPS@}
```

```
\xydef@\oldtexturesPSunload@{\UnloadPS@
 \ifx\xyrealshipout@\undefined\else
  \ifx\shipout\xyrealshipout@
  \else \let\shipout=\xyrealshipout@\fi\fi
 \let\installxyps@@=\relax
 \let\UnloadPS@@=\UnloadPS@
 \let\oldtexturesPS@@=\oldtexturesPS@ }
```

We must return the binding of `\shipout` to its initial value.

Early versions (1.5–1.6) of Textures have two kinds of `\special`, for inserting PostScript code into the dvi-file and two more for reading such code from a file:

- `\special{rawpostscript #1}` puts code directly into the PostScript file.
- `\special{postscript #1}` wraps the code within `sps`...`eps`, which involves graphics state changes within a `gsave/grestore` pair.

Each of these has a corresponding version for reading the PostScript commands from a file.

```
\xydef@\oldTexturesSpecial@#1{\firstPS@@\special{postscript #1}}
\xydef@\oldTexturesMacro@#1{\special{postscript {#1}xy}}
\xydef@\oldTexturesInclude@#1{\firstPS@@\special{rawpostscriptfile #1}}
\xydef@\oldTexturesRaw@#1{\firstPS@@\special{rawpostscript #1}}
```

```
\xydef@\oldtexturesPStypes@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \ifx\PSspecial@@\oldTexturesSpecial@\else
  \let\PSspecial@@=\oldTexturesSpecial@
  \let\PSmacro@@=\oldTexturesMacro@
  \let\PSdict@@=\oldTexturesRaw@
  \let\PSraw@@=\oldTexturesRaw@
  \let\PSinclude@@=\oldTexturesInclude@
  \let\PSspecialdict@@=\oldTexturesRaw@
  \let\xyPSobeylines@@=\obeyoldTextureslines@
  \oldTextures@@{PostScript}%
 \fi
% \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \oldtexturesCurrpt@ \@PSshipout }
```

```
{\catcode`\^^M=\active %
 \gdef\next{\catcode`\^^M=\active \let^^M=\par \obeyspaces}}
```

```
\xylet@\obeyoldTextureslines@=\next
```

The `currentpoint` is defined for the “raw” specials. For Textures this is the same as `\xyPScurrpt@@@`, in `xyps-ps.doc`.

```
\xydef@\oldtexturesCurrpt@{\let\xyPScurrpt@@=\oldtexturesCurrpt@@}%
\xydef@\oldtexturesCurrpt@@{\PSspecial@@{xyi}\PSraw@@{userdict begin xyx end}}
```

The PostScript operator called `xyx` is defined in `xyps-ps` for storing the location placed on the stack by `xyi`.

Textures v1.6 requires the PostScript dictionary to be shipped-out with every page. To achieve this efficiently we rebind `\shipout`, as described in `xyps-ps.doc`. Next set the flag `\UsePSdict@@true` after having rebound `\includePSmessage@` to gobble the attempt to load the dictionary directly into the dvi-file.

This installs the PostScript backend.

```
\xydef@\oldtexturesPS@{\oldtexturesPStypes@
 \let\oldtexturesPS@@=\relax
 \let\UnloadPS@@=\oldtexturesPSunload@
 \let\installxyps@@=\@PSshipout \installxyps@ }
```

- rotated/scaled diagrams and text, using PostScript.

```
\xydef@\oldtexturesRotate@@{\oldtexturesRotate@}
```

```
\xydef@\oldtexturesRotate@{\oldtexturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%
  \xyRotScale@@ \installPSrotscale@ }%
 \else\DN@{\xydriverfail@{rotations are}\UnloadRotate@\relax}\fi
 \next@ }
```

- variable line-widths and poly-lines, using PostScript.

```
\xydef@\oldtexturesLine@@{\oldtexturesLine@}
```

```
\xydef@\oldtexturesLine@{\oldtexturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}%
  \installPSline@ }%
 \else\DN@{\xydriverfail@{line-styles are}\UnloadLine@\relax}\fi
 \next@ }
```

- extra frames and fills, using PostScript.

```
\xydef@\oldtexturesFrames@@{\oldtexturesFrames@}
```

```
\xydef@\oldtexturesFrames@{\oldtexturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}%
  \installPSframes@ }%
 \else\DN@{\xydriverfail@{frames are}\UnloadPSFrames@\relax}\fi
 \next@ }
```

- patterns and tiles, using PostScript.

```
\xydef@\oldtexturesTiles@@{\oldtexturesTiles@}

\xydef@\oldtexturesTiles@{\oldtexturesPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}%
  \installPSpatterns@ \xystandardpatterns@}%
 \else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
 \next@ }
```

The end & Log

```
\xyendinput

% $Log: xy16textures.doc,v $
% Revision 3.3  1996/12/18 14:21:23  ross
% Ross's version
%
% Revision 3.3.1.1  1996/12/18  08:45:28  ross
% *** empty log message ***
%
% Revision 3.2  1995/09/19  18:20:20  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

4.9 XDVI driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

This driver provides support for extensions when using variants of the `xdvi` driver, by Eric Cooper, Bob Scheifler, Mark Eichin and others. It has been used successfully with `xdvi` patchlevel 20, by Paul Vojta, and `xdvik` version 18f, by Karl Berry.

Some of the supported features assume that the implementation of `xdvi` is linked to a PostScript renderer; e.g. *Ghostscript* or Display PostScript. If such support is not available, then invoking `xdvi` using the command `xdvi -hushspecials` will suppress warning messages that might otherwise be produced. One drawback of such a setup is that much of the PostScript is not rendered until after all of the font characters, etc. have been placed on the page. Thus text that was meant to be placed on top of a filled or patterned region may appear to be obscured by it. However when printed, using a PostScript printer, the correct placement is obtained.

Header:

```
%% $Id: xyxdvi.doc,v 3.3 1996/12/18 09:53:22 ross Exp $
%%
%% Xy-pic ``xdvi-driver'' option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{xdvi}{XDVI driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xydef@\xdvi@@#1{\dvimessage@{xdvi}{#1}}

\newdriver{%
 \xyaddsupport{color}\xdviColor@@
 \xyaddsupport{crayon}\xdviCrayola@@
 \xyaddsupport{ps}\xdviPS@@
 \xyaddsupport{rotate}\xdviRotate@@
 \xyaddsupport{line}\xdviLine@@
 \xyaddsupport{frame}\xdviFrames@@
 \xyaddsupport{tile}\xdviTiles@@
 \xyaddsupport{tpic}\xdviTPIC@
}
```

Driver installation Supported `\special` effects are...

- colour, using PostScript.

  Not all versions of `xdvi` support color `\specials`, so there is no direct support for colour. However parts of pictures rendered using PostScript may appear coloured, due to interpretation of colour commands within the PostScript.

  Disable the `\xylocalColor@` but enable the `\xycolor@` for use by PostScript.

```
\xydef@\xdvi@Color@{\xdviPStypes@
 \let\xylocalColor@=\xyNoColor@@
% \let\xylocalColor@=\xystackcolor@
% \let\xycolor@push\xycolor@push@@
 \def\xycolor@{\xycolor@@}\let\xycolorwarning@=\relax
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}\installPScolor@ }%
 \else\DN@{}\fi \next@
 \ifx\xycolor@@\xyNoColor@ \let\xycolor@@=\xycolor@raw@@\fi
 \ifx\xycrayolaloaded\empty \DN@{\xystandardcolors@\xdviCrayola@{}}%
 \else \DN@{\xystandardcolors@}\fi \next@
 \let\xdviColor@@=\relax
 \let\UnloadColor@@=\NoXDVIcolor@
 }%
```

```
\xydef@\xdviColor@@{\xdvi@Color@}
```

```
\xydef@\NoXDVIcolor@{%
 \def\xycolor@{\xycolor@@}%
 \let\xycolor@@=\xyNoColor@
 \let\xylocalColor@=\xyNoColor@@
 \let\xycolorwarning@=\xycolorwarning@@
 \let\xdviColor@@=\xdvi@Color@
 \let\UnloadColor@@=\UnloadColor@
 }
```

If neither support file `colordvi.tex` nor `colordvi.sty` can be found, then the normal colour support will not be available. However the colour support for the PostScript back-end can still be used.

```
\xydef@\NoXDVIcolordvi@{\xywarning@{%
 Neither  colordvi.tex  nor  colordvi.sty  could be found.}%
 \xyBEcolorcheck@}%
```

- crayon colours.

  The 68 colours that `xdvi` recognizes by name are not loaded unless the `crayon` option has been requested.

```
\xydef@\xdviCrayola@@{\xdviColor@@\xdviCrayola@}
```

```
\xydef@\xdviCrayola@{%
 \let\prevxycolor@=\newxycolor@
  \def\newxyXDVIcolor@##1##2##3{\prevxycolor@{##1}{}{##1}}%
  \let\newxycolor@=\newxyXDVIcolor@
  \installCrayolaColors@
 \let\newxycolor@=\prevxycolor@ }%
```

  Although any order of loading options: `ps`, `xdvi`, `color` and `crayon` produces the desired result visually, the PostScript code can be different with different loading order. The most easily readable is obtained when `crayon` is requested last.

- PostScript back-end.

```
\xydef@\xdviPS@@{\xdviPS@}
```

```
\xydef@\xdviPS@unload{\UnloadPS@
 \let\UnloadPS@@=\UnloadPS@
 \let\xdviPS@@=\xdviPS@ }
```

The latest versions of `xdvi` recognise most forms of `\special` command that are recognised by `dvips`; these are used to support PostScript effects.

```
\xydef@\xdviPStypes@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}%
 \ifx\PSspecial@@\xdviSpecial@\else
  \let\PSspecial@@=\xdviSpecial@
  \let\PSmacro@@=\xdviMacro@
  \let\PSdict@@=\xdviDict@
  \let\PSspecialdict@@=\xdviDict@
  \let\PSraw@@=\xdviRaw@
  \let\PSinclude@@=\xdviInclude@
  \let\xyPSobeylines@@=\obeyXDVIlines@
  \let\xyPScurrpt@@=\xdviCurrpt@@
  \xdvi@@{PostScript}%
 \fi \XYpredict@ }
```

```
\xywarnifdefined\xdviSpecial@
\xywarnifdefined\xdviMacro@
\xywarnifdefined\xdviDict@
\xywarnifdefined\xdviInclude@
\xywarnifdefined\xdviRaw@
{\catcode`\"=11 \catcode`\!=11 \catcode`\:=11
 \gdef\xdviSpecial@#1{\special{ps:: SDict begin #1 end}}
 \gdef\xdviMacro@#1{\firstPS@@\special{" {#1}xy}}
% \gdef\xdviDict@#1{\special{! mark{#1}stopped cleartomark}}
 \gdef\xdviDict@#1{\special{! #1}}
 \gdef\xdviInclude@#1{\special{header=#1}}
 \gdef\xdviRaw@#1{\special{ps: SDict begin #1 end}}
```

```
\catcode`\^^M=\active %
 \gdef\next{\catcode`\^^M=\active \let^^M=\space }}%
\xylet@\obeyXDVIlines@=\next
```

All the PostScript definitions passed to `xdvi` by reading the dictionary file are stored in a PostScript dictionary which it defines, called `SDict`. This must be the current dictionary whenever an XY-ps command is to be executed. This will be the case whenever the `\special{"` *commands*} or `\special{!` *commands*} forms are used, but not when the `\special{ps::` *commands*} form is used. Thus the `\xdviSpecial@` macro includes code to open `SDict` and close it when finished; so also does the `\xdviRaw@` type, since this is used with XY-pic PostScript operators whose definitions have been placed within `SDict`.

The `currentpoint` is defined for the "raw" specials.

```
\xydef@\xdviCurrpt@{\let\xyPScurrpt@@=\xdviCurrpt@@}%
\xydef@\xdviCurrpt@@{\PSraw@@{xyp}}
```

The PostScript operator called `xyp` is defined in `xyps-ps` for storing the location read from `currentpoint`.

This installs the PostScript backend.

```
\xydef@\xdviPS@{\xdviPStypes@
 \let\xdviPS@@=\PSspecials@true
 \let\UnloadPS@@=\xdviPS@unload
 \let\installxyps@@=\relax \installxyps@ }
```

• rotated/scaled diagrams and text, using PostScript.

```
\xydef@\xdviRotate@@{\xdviRotate@}
```

```
\xydef@\xdviRotate@{\xdviPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%
  \xdviCurrpt@ \xdviRotScale@ \installPSrotscale@ }%
 \else\DN@{\xydriverfail@{rotations are}UnloadRotate@\relax}\fi
 \next@ }
```

```
\xydef@\xdviRotScale@{%
 \def\xyrot@start(##1){{xyct ##1\space xyrs }}%
 \def\xyrot@end{}%
 \def\xyscale@start(##1,##2){{xyct ##1\space##2\space xyss }}%
 \def\xyscale@end{}}
```

• variable line-widths and poly-lines, using PostScript.

```
\xydef@\xdviLine@@{\xdviLine@}
```

```
\xydef@\xdviLine@{\xdviPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}\installPSline@ }%
 \else\DN@{\xydriverfail@{line-widths are}\UnloadLine@\relax}\fi
 \next@ }
```

• extra frames and fills, using PostScript.

```
\xydef@\xdviFrames@@{\xdviFrames@}
```

```
\xydef@\xdviFrames@{\xdviPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}\installPSframes@ }%
 \else\DN@{\xydriverfail@{frames are}\UnloadFrames@\relax}\fi
 \next@ }
```

• patterns and tiles, using PostScript.

```
\xydef@\xdviTiles@@{\xdviTiles@}
```

```
\xydef@\xdviTiles@{\xdviPStypes@
 \expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%
  \xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}\installPSpatterns@ }%
 \else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
 \next@ }
```

• tpic drawing commands.

```
\xydef@\xdviTPIC@@{\xdviTPIC@}
```

```
\xydef@\xdviTPIC@{\setupxyTPIC@}
```

The end & Log

```
\xyendinput
```

```
% $Log: xyxdvi.doc,v $
% Revision 3.3  1996/12/18  09:53:22  ross
% minor improvements to file-loading commands
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:28:57  ross
% Releasing version 3.1!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

4.10 CMacTeX driver

Vers. 3.3 by Ross Moore ⟨ross@mpce.mq.edu.au⟩

Thomas Kiffe's CMacTEX application uses DVIPS as its dvi-driver.

Header:

```
%% $Id: xycmactex.doc,v 3.3 1996/12/18 08:54:25 ross Exp $
%%
%% Xy-pic ``CMacTeX'' driver interface option.
%% Copyright (c) 1995-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
```

```
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi

\xyprovide{cmactex}{CMacTeX driver}{\stripRCS$Revision: 3.3 $}%
 {Ross Moore}{ross@mpce.mq.edu.au}%
 {Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```
\xyrequire{dvips}
```

The end & Log

```
\xyendinput

% $Log: xycmactex.doc,v $
% Revision 3.3  1996/12/18  08:54:25  ross
% no changes
%
% Revision 3.1  1995/09/05 20:28:57  ross
% Releasing version 3.1!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.
```

4.11 Extra features with PostScript support

4.11.1 xyps-ps.doc

The included file `xyps-ps.tex` (version3.6) provides support for PostScript `\special` commands used by the `ps` backend extension as well as PostScript-based options, to produce special effects not available directly with TeX.

Header:

```
%% $Id: xyps-ps.doc,v 3.6 1998/03/06 01:28:05 krisrose Exp $
%%
%% Xy-pic ``PS-PS'' PostScript support.
%% Copyright (c) 1993-1997      Ross Moore      <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997      Kristoffer H. Rose  <krisrose@ens-lyon.fr>
```

```
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\expandafter\ifx\csname xyps-psloaded\endcsname\empty
 \message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-psloaded\endcsname=\empty \xycatcodes
```

PostScript `\special` commands The PostScript `\special`s which are used by XY-ps fall into four broad classes:

1. execute a piece of code, e.g. to draw some graphic object;
2. add a new PostScript definition to the existing dictionary of commands;
3. change the value of some parameters, storing them for later use; and
4. read PostScript commands from a pre-existing file.

Since different drivers may provide different syntax for these classes of command, XY-ps has different macros to optimize to interface to the different drivers. Initially these control-sequence names are bound to macros that do nothing. Upon specifying a driver, they will be bound to a macro appropriate for that driver.

```
\xylet@\PSmacro@@=\eat@
\xylet@\PSdict@@=\eat@
\xylet@\PSspecial@@=\eat@
\xylet@\PSread@@=\eat@
\xylet@\PSinclude@@=\eat@
\xylet@\PSraw@@=\eat@
\xylet@\xyPSobeylines@@=\relax
\xylet@\PSspecialdict@@=\eat@
```

```
\xydef@\UnloadPSps@{%
 \let\PSmacro@@=\eat@
 \let\PSdict@@=\eat@
 \let\PSspecial@@=\eat@
 \let\PSread@@=\eat@
 \let\PSinclude@@=\eat@
 \let\PSraw@@=\eat@
```

```
 \let\xyPSobeylines@@=\relax
 \let\PSspecialdict@@=\eat@
}
```

Unloading PostScript requires also turning off the PostScript-backend, if it has been loaded.

```
\xydef@\UnloadPS@{\UnloadPSps@
 \expandafter\ifx\csname xypsloaded\endcsname\unknown\PSspecials@false
 \else\expandafter\NoPSspecials\fi}
\xylet@\UnloadPS@@=\UnloadPS@
```

```
%\xyaddunsupport{ps}\UnloadPS@@
```

```
\ifx\undefined\dvitype@\xydef@\dvitype@{null}\fi
```

Some drivers, in particular versions of Textures earlier than 1.7, require the PostScript dictionary to be included with every page shipped out. We do this by providing a routine which rebinds the `\shipout` primitive to a macro `\xyPSshipout@` which prepends the XY-ps PostScript dictionary to the box being shipped-out. This routine `\@PSshipout` is called, if necessary, when the driver has been specified.

```
\xylet@\xyrealshipout@=\shipout
\xydef@\xyPSshipout@{\setbox9=\copy\voidb@x
 \afterassignment\xyPSshipout@i\setbox9=}
\xydef@\xyPSshipout@i{\ifvoid9 \expandafter\aftergroup\fi\xyPSshipout@ii}
\xydef@\xyPSshipout@ii{\xyrealshipout@\vbox{\XYpredict@@\box9}}
```

```
\xydef@\@PSshipout{\global\let\shipout=\xyPSshipout@ }
```

The box register `\box9` is assumed to be a scratch register, used globally according to TeX conventions.

When the PostScript backend is loaded then we need to rebind `\shipout` to include the full dictionary.

```
\xydef@\xyPSshipout@PSii{%
 \ifUsePSdict@@\let\XYshipdict@=\includeXYPSdict@@
 \else\def\XYshipdict@{\XYpredict@@\XYdict@@}\fi
 \xyrealshipout@\vbox{\XYshipdict@\box9}}
```

These two control sequences are used in the DVI-driver files to control when the PostScript ⟨driver⟩ can be altered.

```
\xydef@\firstPS@{%\fixedxydriver@
 \global\let\firstPS@@=\relax \global\let\firstPS@=\relax}
\xylet@\firstPS@@=\firstPS@
```

4.11.2 Installation

To install use of PostScript requires loading the PostScript dictionary. However this can only be done when a ⟨driver⟩ is known to be able to support it. Hence the `\installxyps@` command should be called only from within a ⟨driver⟩-file.

Furthermore, the ⟨driver⟩-file should define a value for `\installxyps@@` to including anything specific to that ⟨driver⟩; even if only `\let\installxyps@@=\relax`. If `\dumpPSdict` has been specified, then the dictionary must be written first, so it can be used with the subsequent processing (see the subsection below).

```
\ifx\undefined\dumpPSdict@@ \xylet@\dumpPSdict@@=\relax \fi
```

```
\xydef@\installxyps@{%
 \xyinputorelse@{xyps-ps}{\xydrivernoload@{ps-ps}}\xycatcodes
 \installxyps@@ \installxyps@x }

\xylet@\installxyps@@=\relax
\xylet@\setupxyPS@@=\relax
\xywithoption{ps}{\let\setupxyPS@@=\setupxyPS@}

\xydef@\installxyps@x{\dumpPSdict@@
 \ifx\xyrealshipout@\undefined \let\xyrealshipout@=\shipout\fi
 \ifx \shipout \xyrealshipout@
  \ifUsePSdict@@ \DN@{\includePSmessage@\includeXYPSdict@\message{>}}%
  \else \DN@{\XYdict@\endXYdict@}\fi
 \else \DN@{\let\xyPSshipout@ii=\xyPSshipout@PSii \@PSshipout}\fi
 \let\firstPS@=\relax \next@
 \let\XYpredict@=\relax
 \setupxyPS@@ }

\let\realinstallxyps@x=\installxyps@x
\ifx\undefined\AfterBeginDocument
 \def\installxyps@x{\xysetup@@{\realinstallxyps@x}}%
\else
 \def\installxyps@x{\AfterBeginDocument{\realinstallxyps@x}}%
\fi
```

In case no ⟨driver⟩ was known when a call to the `\installxyps@` method was queued, e.g. by `\dumpPSdict{}` before any `\xyoption{`⟨driver⟩`}`, then a check is made to see there is now a known ⟨driver⟩ which supports PostScript. If so then this installation is cancelled since another should follow, otherwise the default ⟨driver⟩ of `dvips` is assumed. The command `\xyPSdriver#1` is provided to allow users to change this default ⟨driver⟩. This may be necessary when multiple ⟨driver⟩s are required and the final one specified does not support PostScript.

```
\xydef@\xydefaultdriver@{dvips}
\xydef@\xyPSdriver#1{\def\xydefaultdriver@{#1}}

\xydef@\installxyps@orig{\DN@{}%
 \ifx\xydriversselected@@\empty
  \DN@{\expandafter\defaultinstallps@\expandafter{\xydefaultdriver@}}%
 \else \def\do##1{%
   \expandafter\ifx\csname##1@xy@ps\endcsname\relax
    \DN@{\defaultinstallps@{##1}}}%
  \xydriversselected@@
 \else \DN@##1{}\fi\fi \next@ }

%\xylet@\installxyps@@=\installxyps@orig

\xydef@\defaultinstallps@#1#2{%
 \xywarning@{no driver specified, using #1 }%
 \xyrequire{#1}\csname #1@xy@ps\endcsname{}}

\xydef@\null@xy@ps{\installxyps@}
```

PostScript escape

An extra ⟨shape⟩ modifier key allows arbitrary PostScript code to be applied to the current ⟨object⟩.

`[!⟨postscript code⟩]`	for special effects
`[psxy]`	stores current location.

```
\xydefcsname@{shape [!...]}#1{%\tracingall%
 \expandafter\ifx\csname xyps-sloaded\endcsname\empty \DN@{\xyPSescapestyle@{#1}@@}%
 \else\ifxyPSshapes@\DN@{\xyPSshapechar@{#1}@@}%
 \else \DN@{\xyPScharwarning@}\fi\fi \next@ }
\xydefcsname@{shape [psxy]}{\ifxyPSshapes@\DN@{\xyPSpsxy@}%
 \else \DN@{\xyPScharwarning@}\fi \next@ }
```

Normally the ⟨postscript code⟩ will be a simple command to alter the PostScript graphics state: e.g. `[!1 0 0 setrgbcolor]` changes the colour used to render parts of the ⟨object⟩. Any number of such ⟨shape⟩ modifiers is allowable, however it is more efficient to combine them into a single modifier, whenever possible.

It is very important that braces `{` and `}` do not appear explicitly in any ⟨postscript code⟩, as this may upset the XY-pic ⟨object⟩ parsing. However it is acceptable to have a control sequence name here, expanding into more intricate PostScript code. This will not be expanded until a later (safe) time.

Due to differences within the DVI-drivers, such simple PostScript commands need not affect every part of an ⟨object⟩. In particular the lines, curves and arrowheads generated by XY-pic use a different mechanism, which should give the same result with all drivers. This involves redefining some PostScript procedures which are always read prior to rendering one of these objects. One simple way to specify a red line is as follows; the `xycolor` extension provides more sophisticated support for colour. The ⟨shape⟩ modifiers described in the previous section also use this mechanism, so should work correctly with all drivers.

```
\def\colorxy(#1){%
 /xycolor{#1 setrgbcolor}def}
...
\connect[!\colorxy(1 0 0)]\dir{-}
...
```

Note how the braces are inserted within the expansion of the control sequence `\colorxy`, which happens after parsing of the ⟨connection⟩. The following table shows which graphics parameters are treated in this way, their default settings, and the type of PostScript code needed to change them.

colour	`/xycolor{0 setgray}def`
line-width	`/xywidth{.4 setlinewidth}def`
dashing	`/xydash{[] 0 setdash}def`
line-cap	`/xycap{1 setlinecap}def`
line-join	`/xyjoin{1 setlinejoin}def`

This feature is meant primarily for modifying the rendering of objects specified in TEX and XY-pic, not for drawing new objects within PostScript. No guarantee can be given of the current location, or scale, which may be different with different drivers. However a good PostScript programmer will be able to overcome such difficulties and do much more. To aid in this the special modifier `[psxy]` is provided to record the location where the reference point of the current ⟨object⟩ will be placed. Its coordinates are stored with keys `xyXpos` and `xyYpos`.

When the `\special` is placed the following registers contain important values: `\L@p` = horizontal displacement of Xy-pic reference point from the TeX reference point (i.e. left-hand end of the box) of the initial object; `\D@p` = vertical displacement of Xy-pic reference point from the TeX reference point (i.e. the baseline) of the initial object; `\R@p` = horizontal offset, resulting from ⟨shape⟩ modifiers; `\U@p` = vertical offset, initially `-\D@p` but alterable by ⟨shape⟩ modifiers.

```
\xydef@\xyPSpsxy@{%
 \setboxz@h{\dimen@=\L@p \advance\dimen@-\R@p
 \dimen@ii=-\U@p \raise\dimen@ii\hbox to\z@{%
  \kern\dimen@\xyPScurrpt@\kern-\dimen@}\boxz@}}%
```

```
\xydef@\xyPScurrpt@{\ifxyPSshapes@\expandafter\xyPScurrpt@@\fi}
```

It is necessary to know the current PostScript location. Unfortunately the `currentpoint` operator frequently has no value. The following code overcomes this difficulty.

Some drivers may need to define this differently...

```
\xydef@\xyPScurrpt@@@{\PSspecial@@{xyi}\PSraw@@{userdict begin xyx end}}
\ifx\undefined\xyPScurrpt@@ \xylet@\xyPScurrpt@@=\xyPScurrpt@@@ \fi
```

```
%\xydef@\xyPScurrpt@@@{\PSspecial@@{userdict begin
% 0 0 transform grestore gsave itransform
% /xyYpos exch def /xyXpos exch def end}}
```

```
%\xydef@\xyPScurrpt@@@{%
% \PSspecial@@{0 0 transform grestore gsave itransform }%
% \PSraw@@{userdict begin /xyYpos exch def /xyXpos exch def end}}
```

The PostScript operator called `xyx` is defined below, for storing the location placed on the stack by `xyi`.

Technical Note

: The scoping is achieved by using two `\specials` so that the resulting PostScript file should ultimately look like:

... { special code before} ⟨object⟩ {special code after} ...

The "code after" is to cancel the effect, returning the graphics state to what it was prior to the "code before". Not all DVI-drivers can achieve this sequencing. In particular OzTeX collects all `\specials` in the DVI-file and places their contents at the beginning of the PostScript file: any effect would be cancelled immediately after it has been established.

The user can add code to both parts by expressing the ⟨shape⟩ modifier as follows:

`*...!`⟨code before⟩`<>`⟨code after⟩`]...` .

Further Technical Note

: The "code before" does two things in addition to that code given explicitly by the user. Firstly it issues a `gsave` then it opens a new dictionary on the dictionary stack. The "code after" contains the matching `grestore` after closing the new dictionary.

Since objects can be built as `\composites` and diagrams can be nested, there is the possibility of generating long chains of nested PostScript dictionaries. For this reason the dictionary is kept small, allowing only 8 key-value pairs to be `def`ined within it. If more are required, the user should define a private dictionary to hold the extra key-value pairs, making sure that it is open when its entries need to be accessed.

```
\xywarnifdefined\xyPSshapechar@
\xywarnifdefined\xyPSsplitshape@

{\xyuncatcodes \catcode`@=11
 \gdef\xyPSshapechar@#1@@{\xyPSsplitshape@#1<><>@@}%
 \gdef\xyPSsplitshape@#1<>#2<>#3@@{\xyPSsplitPS@{#1}{#2}}}

%\xydef@\xyPSsplitPS@#1#2{%\aftergroup\show\aftergroup\relax
% \def\xyPSpre@{#1}\def\xyPSpost@{#2}\modPSboxz@ }

\xydef@\xyPSrawA@#1{\expandafter\xyPSrawAA@ #1{}}%
\xydef@\xyPSrawAA@#1#2{\PSraw@@{{pu #1}xyg #2}%
  \PSmessage{{pu #1}xyg #2}}%
\xydef@\xyPSrawZ@#1{\PSraw@@{{#1 pp}xyf}\PSmessage{{#1 pp}xyf}}%
```

The PostScript operators `pu`, `pp`, `xyg` and `xyf` are defined below, for push/pop of the `XYddict` stack, preceded/followed by a `gsave`/`grestore` of the PostScript graphics state.

```
\xydef@\modPSboxz@{%
 \expandafter\DN@\expandafter{\xyPSpre@{}}\ifx\next@\empty\else
 \setboxz@h{\expandafter\xyPSrawA@\expandafter{\xyPSpre@{}}\boxz@
  \expandafter\xyPSrawZ@\expandafter{\xyPSpost@}}\DN@{}\fi}%

\xydef@\modPSdrop@{%
 \expandafter\DNii@\expandafter{%
  \expandafter\xyPSrawA@\expandafter{\xyPSpre@{}}}%
 \DN@##1{\expandafter\def\expandafter\tmp@\expandafter{\nextii@ ##1}}%
  \expandafter\next@\expandafter{\Drop@@}%
 \expandafter\DNii@\expandafter{%
  \expandafter\xyPSrawZ@\expandafter{\xyPSpost@}}%
 \DN@##1{\expandafter\def\expandafter\Drop@@\expandafter{\tmp@ ##1}}%
  \expandafter\next@\expandafter{\nextii@}}%

\xydef@\modPSconnect@{%
 \expandafter\DNii@\expandafter{%
  \expandafter\xyPSrawA@\expandafter{\xyPSpre@{}}}%
 \DN@##1{\expandafter\def\expandafter\tmp@\expandafter{\nextii@ ##1}}%
  \expandafter\next@\expandafter{\Connect@@}%
 \expandafter\DNii@\expandafter{%
  \expandafter\xyPSrawZ@\expandafter{\xyPSpost@}}%
 \DN@##1{\expandafter\def\expandafter\Connect@@\expandafter{\tmp@ ##1}}%
  \expandafter\next@\expandafter{\nextii@}}%

\xydef@\xyPSpre@{{}{}}
\xydef@\xyPSpost@{}
```

Some utility macros for controlling writing to the log-file.

```
\xydef@\PSmessage@#1{\W@{PS: #1, direction=\the\Direction}}
\xydef@\PSignore@#1{}%
\xydef@\showPSmessages{\let\PSmessage=\PSmessage@}
\xydef@\hidePSmessages{\let\PSmessage=\PSignore@}
\xylet@\PStracing=\showPSmessages
```

```
\hidePSmessages
```

```
\xydef@\PSmacro@#1{\PSmacro@@{#1}\PSmessage{#1}}
```

PostScript Header file

This creates a flag to indicate whether the user wishes to get the PostScript dictionary from an external header file.

```
\xydef@\includeXYPSdict@@{%
 \expandafter\PSinclude@@\expandafter{\xyPSdictname}}
\xydef@\includeXYPSdict@r{\message{PS: includeXYPSdict}\includeXYPSdict@@
 \let\includeXYPSdict@=\relax}
\xylet@\includeXYPSdict@\includeXYPSdict@r
```

```
\xydef@\endXYdict@{}
```

```
\xydef@\includePSmessage@@{\message{<xyps: including \xyPSdictname}}
\xylet@\includePSmessage@=\includePSmessage@@
```

```
\xydef@\xyPSdictknown@{%
% \immediate\closein\xyPSdictread@
 \immediate\closein\xyread@
 \xywarning@{File \xyPSdictname\space already exists}
 \message{ Do you wish to overwrite it \string?  y/n }%
 \read16 to \next
 \DNii@{\xywarning@{You did not type Y or N -- try again please}%
  \xyPSdictknown@}%
 \DN@{Y }\ifx\next@\next \DNii@{\writePSdict@@}%
 \else\DN@{y }\ifx\next@\next \DNii@{\writePSdict@@}%
 \else\DN@{n }\ifx\next@\next \DNii@{\cancelPSdict@}%
 \else\DN@{N }\ifx\next@\next \DNii@{\cancelPSdict@}%
 \fi\fi\fi\fi \nextii@ }
```

```
\xydef@\cancelPSdict@{%
 \xywarning@{cancelling dump of PostScript dictionary}}
```

```
\xydef@\noDictmessage@{%
 \xywarning@{\dvitype@\space does not support the use of a header file}}
```

These macros are also defined in `xyps.doc`. To avoid warning messages about redefinitions, that mechanism has been disabled.

```
\DN@{\csname newif\endcsname\ifUsePSdict@@\DN@{}}
 \ifx\undefined\UsePSdict@@true\else\DN@{}\fi \next@
```

```
\ifx\undefined\xyPSdefaultdict
 \xydef@\xyPSdefaultdict{%
  \DN@##1.##2.##3@{\gdef\xyPSdictname{xy##1##2dict.pro}}%
 \expandafter\next@\xyversion.@}%
\fi
```

```
\ifx\undefined\UsePSheader@
 \xydef@\UsePSheader@#1{%
  \DN@{#1}\ifx\next@\empty
%  \ifx\xyPSdictname\undefined\xyPSdefaultdict\fi
  \else \gdef\xyPSdictname{#1}\fi
```

```
  \ifx\xyPSdictname\undefined\xyPSdefaultdict\fi
  \gdef\xyHeaderMessage@{%
   It includes a reference to the PostScript file \xyPSdictname.^^J}%
  \UsePSdict@@true}%
 \let\UsePSheader=\UsePSheader@
\fi
```

This is used to write the dictionary to a file, suitable for inclusion as a resource within any PostScript document. It is fully conforming to Adobe's document structuring guidelines.

```
\xywarnifdefined\xyPS@@
\xywarnifdefined\xyPS@@@
{\catcode`|=14 \catcode`\%=12 \catcode`\<=1 \catcode`\>=2 |
 \gdef\xyPS@@<%%>\gdef\xyPS@@@<%!>}
```

```
%\xydef@\writePSdict@@{\writePSdict@ \let\writePSdict@@=\relax}
\xydef@\writePSdict@@{\message{PS: writePSdict@@}\writePSdict@
 \def\writePSdict@@{\message{write PS relaxed}}}
{\catcode`|=14 \catcode`\%=12
 \gdef\writePSdict@{{\begingroup|
  \immediate\openout\xywrite@=\xyPSdictname \relax
  \message{<xyps: dumping \xyPSdictname}|
   \def\PSdict@@@##1{\immediate\write\xywrite@{##1}}|
   \let\xyPSobeylines@@=\obeyoutlines@
   \let\firstPS@@=\relax
   \let\endXYdict@=\relax
   \PSdict@@@{\xyPS@@@ PS-Adobe-3.0 Resource-procset}|
   \PSdict@@@{\xyPS@@ Title: \xyPSdictname}|
   \PSdict@@@{\xyPS@@ Version: REPLACEWITHVERSION}|
   \PSdict@@@{\xyPS@@ Creator: Xy-ps backend to Xy-pic}|
   \PSdict@@@{\xyPS@@ DocumentSuppliedProcSets: XYdict}|
   \ifx\dvitype@\undefined
    \PSdict@@@{\xyPS@@ For: use with Xy-pic}\else
    \PSdict@@@{\xyPS@@ For: \dvitype@ use of Xy-pic}\fi
   \PSdict@@@{\xyPS@@ BeginResource: procset XYdict}|
   \def\PSspecialdict@@##1{\PSdict@@@{##1^^J}}|
   \let\PSdict@@=\PSdict@@@ \let\xy@begindvi=\literal@|
   \UsePSdict@@false\XYpredict@@|
   \XYdict@@|
   \PSdict@@@{\xyPS@@ EndResource}|
   \immediate\closeout\xywrite@
  \message{>}|
 \endgroup}}}
```

Within the expansion of `\XYdict@` the end-of-line tokens are still active. The following expansion seems to work on all systems so far tested.

```
\xywarnifdefined\obeyoutlines@
{\catcode`\^^M=\active%
 \gdef\obeyoutlines@{\catcode`\^^M=\active \def^^M{^^J}%
  \newlinechar=`\^^J\obeyspaces}}
```

The dictionary of PostScript commands is split into two pieces according to whether they are relevent to general PostScript effects or only to the PostScript backend. These pieces are generated from code within the files `xyps-pro.tex` and `xypsdict.tex`. These files have corresponding `.doc` versions for documentation.

The following commands cause the parts of the PostScript dictionary to be read from the appropriate files. If `\ifUsePSdict@@` gives `\iftrue` then there may be no need to do anything, since the definitions will be subsequently loaded from the header file. Normally it is only necessary to read these files once, using `\XYpredict@`. However some ⟨driver⟩s may need it more frequently, so instead call `\XYpredict@@` directly.

```
\xydef@\XYpredict@@{\ifUsePSdict@@\DN@{}\else
 \DN@{\xy@begindvi{\xyinputorelse@{xyps-pro}%
   {\xyerror@{Could not load xyps-pro}{}}\xyuncatcodes}}%
 \next@
 \DN@{\xy@begindvi{\xyinputorelse@{xypspatt}%
   {\xyerror@{Could not load xypspatt}{}}\xyuncatcodes}}\fi
 \next@ }

\xydef@\XYpredict@r{\XYpredict@@ \let\XYpredict@=\relax \installxyps@}
\xylet@\XYpredict@\XYpredict@r

\xydef@\XYdict@{\XYdict@@\def\XYdict@{\message{PS relaxed}}}%
\xydef@\XYdict@@{\DN@{\xyinputorelse@{xypsdict}%
   {\xyerror@{Could not load xypsdict}{}}\xyuncatcodes}%
 \expandafter\xy@begindvi\expandafter{\next@}}
```

This command is for use in macro-packages, providing an easy way to force the dictionary to load, when it has not done so using the usual mechanisms.

```

\xydef@\forceload@XYdict{%
  \let\includeXYPSdict@=\includeXYPSdict@r
  \let\XYpredict@=\XYpredict@r
  \let\XYdict@=\XYdict@@ \expandafter\installxyps@ }
```

4.11.3 Extensions

Several included files handle standard PostScript implementations of XY-pic extensions.

A conditional `\ifxyPSshapes@` is used to indicate whether the special shape effects implemented in `xyps-r.tex` can actually be fully supported by the current driver. When not available, then an attempt to use these effects simply results in a warning message. After two attempts the warning message "turns itself off"; subsequent attempts are simply ignored cleanly.

```
\xynew@{if}\ifxyPSshapes@ \xyPSshapes@true

\xydef@\xyPScharwarning@@{\xyPScharmessage@
 \gdef\xyPScharwarning@@{\xyPScharmessage@
  \xywarning@{...no further PostScript warnings will be given}%
 \global\let\xyPScharwarning@@=\relax }}

\xydef@\xyPScharmessage@{\xywarning@{%
 the current <driver> does not fully support PostScript effects}}%
```

Allow new POSTSCRIPT effects to be defined. This section describes how `\newxyPSshape` is used to define a new effect which is available only in POSTSCRIPT; *i.e.* having no analogue elsewhere within XY-pic.

The new effect will be called via `[`⟨name⟩`]`, which gets interpreted as referring to `\csname xyshape@`⟨name⟩`@\endcsname`. Hence a control sequence of this form must first be created, if it does not already exist. Its initial expansion is simply `\xyundefinedEffect@{`⟨name⟩`}{}@@` which produces a warning message.

```
\xydef@\newxyshape#1#2#3{%
 \DN@{#3}\ifx\next@\empty
  \xydefcsname@{shape [#1]}{\csname xyshape@#1@\endcsname}%
 \else \expandafter\def\csname shape [#1]\endcsname{%
  \csname xyshape@#1@\endcsname}\fi
 \DN@{#2}\ifx\next@\empty
  \expandafter\def\csname xyshape@#1@\endcsname{%
   \xyundefinedEffect@{#1}{}@@}%
  \else \expandafter\def\csname xyshape@#1@\endcsname{#2}\fi }

\xydef@\xyundefinedEffect@#1#2@@{%
 \xywarning@{The #1 #2 effect is not implemented with the current driver}}%
```

When an implementation of the effect is available then `\csname xyshape@`⟨name⟩`@\endcsname` is rebound to `\csname xyPSshape@`⟨name⟩`@\endcsname`, which expands to the contents of `#2` in `\newxyPSshape#1#2`. If `#2` is empty then the expansion is the `\xyPSnotimplemented@@‘`⟨name⟩`’@@` warning message, which is especially useful during development.

```
\xydef@\newxyPSshape#1#2{%
 \expandafter\DN@\expandafter{\csname shape [#1]\endcsname}%
 \DNii@{\relax}\ifx\next@\nextii@ \newxyshape{#1}{}\relax\fi
 \expandafter\DNii@\expandafter{\csname xyPSshape@#1@\endcsname}%
 \expandafter\def\nextii@{#2}%
 \ifx\nextii@\empty\expandafter\def\next@{\xyPSnotimplemented@@‘#1’@@}%
 \else
  \edef\tmp@{\noexpand\let\expandafter\noexpand\next@
   \expandafter\noexpand\nextii@}%
 \expandafter\tmp@\fi }

\xydef@\xyPSnotimplemented@@#1@@{%
 \xywarning@{This PostScript effect is not implemented yet}}%

\xydef@\xyPSnotfinished@@{%
 \xywarning@{This effect is not reliable yet; box sizes may be wrong}}%
```

The end & Log

```
\endinput

% $Log: xyps-ps.doc,v $
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 03:50:08  ross
```

```
% Maintenance release.
%
% Revision 3.3  1996/12/18  09:55:56  ross
% improvements to the file-loading commands
% more robust installation procedures
% shorter tracing messages
% dictionary no longer loads multiply
%
% Revision 3.2  1995/09/19  18:21:41  ross
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
%
% NEW for version 3.1 by by Ross Moore.
```

4.11.4 xyps-c.doc

Header:

```
%% $Id: xyps-c.doc,v 3.3 1996/12/18 10:05:50 ross Exp $
%%
%% Xy-pic ``Colour extension'' PostScript backend support.
%% Copyright (c) 1993-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
```

```
\expandafter\ifx\csname xyps-cloaded\endcsname\empty
 \message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-cloaded\endcsname=\empty \xycatcodes
```

4.11.5 Colour

The included file `xyps-c.tex` (version 3.3) provides PostScript support for the effects defined in the `color` extension in §2.6.

The PostScript style methods are required, so ensure that they are loaded.

```
\expandafter\ifx \csname xyps-sloaded\endcsname\empty\else
 \xyinputorelse@{xyps-s}{\xyerror@{Could not load xyps-s}{}}%
 \xycatcodes\fi
```

This file is loaded and its effects are activated automatically whenever `\xyoption{color}` is requested and the current ⟨driver⟩ supports colours using PostScript. Should there be any need to turn off this support, the following commands are available; they obey usual TEX groupings.

`\NoPScolor`	remove PostScript support
`\UsePScolor`	reinstate PostScript.

Without PostScript support some drivers may still be able to provide some support for colours. These commands are not guaranteed to work adequately with all drivers. They are provided primarily for testing and trouble-shooting; *e.g.* with ⟨driver⟩ configurations untested by the authors of XY-pic, who should be notified of any difficulties.

```
\xydef@\UsePScolor{\installPScolor@}
\xydef@\NoPScolor{\UnloadColor@}
```

Installation Installation consists of rebinding the macros `\xycolor@` and `\newxycolor@` and of reloading the standard colours to include a description for PostScript. This installation should be delayed till after `color` and `ps` themselves have been installed.

```
\xydef@\installPScolor@{\installPSstyle@
 \let\xymath@=\xyPSmath@ \let\xynomath@=\xyPSnomath@
 \let\checkxyPScolor@=\checkxyPScolor@@
 \let\xycolor@@=\xyPScolor@@
 \xystandardcolors@
 \xyuncatcodes }
```

All effects defined in the `color` extension can be implemented using the PostScript dictionary, loaded by `xyps-ps`. This file provides "generic" PostScript code which is known to work correctly with most drivers. No attempt is made to provide special code for particular drivers; for driver-specific variations, consult the appropriate `xy`⟨driver⟩`.doc` file.

```
\xydef@\xyPSmath@{\hbox\bgroup\dimen@=.55ex \checkxyPScolor@ \xyinside@}
\xydef@\xyPSnomath@{\hbox\bgroup\dimen@=\z@ \checkxyPScolor@ \xyinside@}

\xydef@\checkxyPScolor@@{\xycolor@raw@@{}\let\checkxyPScolor@=\relax}
\xylet@\checkxyPScolor@=\relax

\xydef@\xyPSnewcolor@#1#2#3#4#5{%
 \expandafter\DN@\expandafter{\csname shape [#1]\endcsname}%
```

```
 \DNii@{\relax}\ifx\next@\nextii@\newxycolor{#1}{}\relax\fi
 \DNii@{#4}\ifx\nextii@\empty
  \expandafter\DNii@\expandafter{\csname xyPSshape@#1@\endcsname}%
  \expandafter\def\nextii@{\xyPScolor@{#2 #3}}\else
  \expandafter\DNii@\expandafter{\csname xyPSshape@#1@\endcsname}%
  \expandafter\def\nextii@{\xyPScolor@{#4}}\fi
 \ifx\nextii@\empty\expandafter\def\next@{\xyPSnotimplemented@@`#1'@@}%
 \else
  \edef\tmp@{\noexpand\let\expandafter\noexpand\next@
   \expandafter\noexpand\nextii@}\expandafter\tmp@\fi }

\xydef@\xyPScolor@{\ifPSspecials@\expandafter\xyPScolor@@
 \else\expandafter\eat@\fi}
\xydef@\xyPScolor@@#1{\bgroup \checkXyStyle@
 \def\preStyle@@{\expandafter\addtostyletoks@\expandafter{\xyPSpre@}}%
 \def\postStyle@@{\expandafter\addtostyletoks@\expandafter{\xyPSpost@}}%
 \let\modPSboxz@=\modXYstyle@ \xyPS@color@@{#1}%
 \toks@={\egroup\let\xy@style@=\relax \def\Drop@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\Drop@@}\def\Connect@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\Connect@@}}%
 \the\toks@ }

%\xydef@\xyPS@color@@#1{#1 /xycolor{#1}def }
\xydef@\xyPS@color@@#1{\xyPSsplitPS@{{#1}cc}{}}
\xydef@\xycolor@raw@@#1{\PSraw@@{xyc}}
```

The end & Log

```
\endinput

% $Log: xyps-c.doc,v $
% Revision 3.3  1996/12/18  10:05:50  ross
% minor improvements to file-loading commands
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.9  1994/06/09  14:38:56  ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% Revision 2.9  1994/06/09  14:38:56  ross
% Release 3beta.
```

```
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% NEW for version 2.9 by by Ross Moore.
```

4.11.6 xyps-f.doc

Header:

```
%% $Id: xyps-f.doc,v 3.4 1997/05/18 01:13:24 ross Exp $
%%
%% Xy-pic ``Frame extension'' PostScript backend support.
%% Copyright (c) 1995-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\expandafter\ifx\csname xyps-floaded\endcsname\empty
 \message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-floaded\endcsname=\empty \xycatcodes
```

4.11.7 Frames

The included file `xyps-f.tex` (version 3.4) provides PostScript support for the effects defined in the `frame` extension described in §2.2. It implements some effects otherwise unattainable.

This file is loaded and its effects are activated automatically whenever `\xyoption{frame}` is requested and the current ⟨driver⟩ supports PostScript effects for frames. Should there be any need to turn off this support, the following commands are available; they obey usual TeX groupings.

`\NoPSframes`	remove PostScript support
`\UsePSframes`	reinstate PostScript.

Without PostScript support ellipses may be shown as circles and all filled regions may be represented as black rectangles. These commands are provided primarily for testing and trouble-shooting;

e.g. with ⟨driver⟩ configurations untested by the authors of XY-pic, who should be notified of any difficulties.

Installation The ⟨driver⟩-file must call the following macro to rebind control sequences defined in `xyframe`, thereby allowing the PostScript methods to be used.

```
\xydef@\installPSframes@{%
 \let\framed@body@@=\xyPSframed@body@
 \let\circled@x@@=\xyPScircled@x@
 \let\ellipsed@x@@=\xyPSellipsed@x@
 \installPSfills@
 \xyuncatcodes}
```

```
\xydef@\UsePSframes{\installPSframes@}
\xydef@\NoPSframes{\UnloadFrames@}
```

Frames: solid, dotted and dashed For solid frames use PostScript ovals, otherwise adjust the spacing between dashes or dots according to the lengths, both vertical and horizontal.

```
\xydef@\xyPSframed@body@{\DN@{\framed@body@}%
 \ifx\framehfill\frm@solidh@@
  \DN@{\xyPSframed@oval@\xyPSoval@Special}%
% \else\ifdim\frmradius@@=\z@\else
 \else\ifdim\R@=\z@\else
  \ifx\framehfill\frm@dashh@@
   \DN@{\xyPSframed@oval@\xyPSdashoval@Special}%
  \else\ifx\framehfill\frm@doth@@
   \DN@{\xyPSframed@oval@\xyPSdotoval@Special}%
  \fi\fi
 \fi\fi \next@ }
```

```
\xydef@\xyPSframed@oval@#1{\vglue\U@c
        \hbox{\hglue\L@c\xypsoval@#1\hglue\R@c}\vglue\D@c}
```

```
\xydef@\xyPScircled@x@{\DN@{\xyPScircled@x@@\xyPScirc@Special}%
 \ifx\framehfill\frm@doth@@
  \DN@{\xyPScircled@x@@\xyPSdotcirc@Special}%
 \else\ifx\framehfill\frm@dashh@@
  \DN@{\xyPScircled@x@@\xyPSdashcirc@Special}%
 \fi\fi \next@ }
```

```
\xydef@\xyPScircled@x@@#1{\vglue2\R@
 \hbox{\hglue2\R@\xypscircle@#1\hglue2\R@}%
 \vglue2\R@}
```

In practice the parameters `#1` and `#2` will hold ⟨dimen⟩s.

```
\xydef@\xyPSellipsed@x@#1#2{%
 \DN@{\xyPSellipsed@x@@\xyPScirc@Special}%
 \ifx\framehfill\frm@doth@@
  \DN@{\xyPSellipsed@x@@\xyPSdotcirc@Special}%
 \else\ifx\framehfill\frm@dashh@@
  \DN@{\xyPSellipsed@x@@\xyPSdashcirc@Special}%
 \fi\fi \next@{#1}{#2}}
```

```
\xydef@\xyPSellipsed@x@@#1#2#3{\vglue #3\vglue #3\relax
 \hbox{\L@c = #2\relax \R@c=\L@c \D@c = #3\relax \U@c=\D@c
  \hglue#2\hglue#2\relax\xypsellipse@#1\hglue#2\hglue#2\relax}%
 \vglue #3\vglue #3\relax}
```

These may be used for dashed frames.

```
\xydef@\xyPSdashhfill@{{\SemiDirectionChar=95 \Direction=3072 \xyPSdash@}}
\xydef@\xyPSdashvfill@{{\SemiDirection=31 \Direction=1023 \xyPSdash@}}
```

ovals and circles The width, height, corner-radius are available as `\dimen@i`, `\dimen@ii` and `\R@` respectively; we only need `\R@` here, since the extents hold the real information that we need.

```
\xydef@\xypsoval@#1{%
 \dimen@=\L@c \advance\dimen@\R@c
 \dimen@ii=\U@c \advance\dimen@ii\D@c
 \ifdim\dimen@ii<\dimen@ \relax \dimen@=\dimen@ii\fi \divide\dimen@\tw@
% \R@=\frmradius@@
 \ifdim\R@>\dimen@ \relax \R@=\dimen@ \fi
 \D@c=-\D@c \L@c=-\L@c
 \edef\tmp@{\expandafter\removePT@\the\R@
  \space\expandafter\removePT@\the\R@c
  \space\expandafter\removePT@\the\D@c
  \space\expandafter\removePT@\the\L@c
  \space\expandafter\removePT@\the\U@c }%
 \expandafter#1\expandafter{\tmp@}}

\xydef@\xypscircle@#1{\dimen@=\R@
 \edef\tmp@{\expandafter\removePT@\the\dimen@}%
 \expandafter#1\expandafter{\tmp@}}

\xydef@\xypsellipse@#1{\edef\tmp@{%
  \expandafter\removePT@\the\U@c \space
  \expandafter\removePT@\the\D@c \space
  \expandafter\removePT@\the\L@c \space
  \expandafter\removePT@\the\R@c }%
 \expandafter#1\expandafter{\tmp@}}

\xydef@\xyPSoval@Special#1{\PSmacro@{#1 ov}}
\xydef@\xyPScirc@Special#1{\PSmacro@{#1 ox}}
\xydef@\xyPSdotcirc@Special#1{\PSmacro@{#1 ot}}
\xydef@\xyPSdashcirc@Special#1{\PSmacro@{#1 od}}
\xydef@\xyPSdotoval@Special#1{\PSmacro@{#1 vt}}
\xydef@\xyPSdashoval@Special#1{\PSmacro@{#1 vd}}
```

The POSTSCRIPT operators `ov`, `ox` etc. are defined within `xyps-pro`, having scope limited to `XYdict` and sub-dictionaries.

```
\xydef@\xyPSfilledRegion#1#2{\setboxz@h{\lower\D@c\vbox{%
 \vglue\U@c\hbox{#1#2\empty\hglue\L@c\hglue\R@c}\vglue\D@c}}%
 \ht\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ \boxz@}

\xydef@\filledRectangle@PS{%
 \xyPSfilledRegion{\xypsoval@\xyPSfilloval@Special}}
```

```
\xydef@\filledCircle@PS#1{{\R@c=\R@ \L@c=\R@ \U@c=\R@ \D@c=\R@
 \xyPSfilledRegion{\xypscircle@\xyPSfillcirc@Special}#1}}
\xydef@\filledEllipse@PS{%
 \xyPSfilledRegion{\xypsellipse@\xyPSfillellipse@Special}}
\xydef@\filledOval@PS{%
 \xyPSfilledRegion{\xypsoval@\xyPSfilloval@Special}}

\xydef@\frame@emph@PS#1{\filled@{#1}\relax}

\xydef@\installPSfills@{%
 \let\frame@emph@@=\frame@emph@PS
 \let\filledRectangle@@=\filledRectangle@PS
 \let\filledOval@@=\filledOval@PS
 \let\filledCircle@@=\filledCircle@PS
 \let\filledEllipse@@=\filledEllipse@PS
}
```

Filled regions Rectangles are treated as a special case of ovals. The `#2` parameter signifies whether the outline of the object should also be stroked (`#2=\relax`) after being filled. Such a stroke is always in black.

```
%\xydef@\xyPSfillrect@Special#1#2{%
% \ifx#2\relax\PSmacro@{#1 sfr}%
% \else\PSmacro@{#1 fr}\fi}
\xydef@\xyPSfilloval@Special#1#2{%
 \ifx#2\relax\PSmacro@{#1 sfo}%
 \else\PSmacro@{#1 fo}\fi}
\xydef@\xyPSfillcirc@Special#1#2{%
 \ifx#2\relax\PSmacro@{#1 sfc}%
 \else\PSmacro@{#1 fc}\fi}
\xydef@\xyPSfillellipse@Special#1#2{%
 \ifx#2\relax\PSmacro@{#1 sfe}%
 \else\PSmacro@{#1 fe}\fi}
```

The end & Log

```
\endinput

% $Log: xyps-f.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  10:07:22  ross
% cosmetic changes to macro-names
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
```

```
%
%
% NEW for version 3.0 by by Ross Moore.
```

4.11.8 xyps-l.doc

Header:

```
%% $Id: xyps-l.doc,v 3.4 1997/05/18 01:13:24 ross Exp $
%%
%% Xy-pic ‘‘Line extension’’ PostScript backend support.
%% Copyright (c) 1993-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\expandafter\ifx\csname xyps-lloaded\endcsname\empty
 \message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-lloaded\endcsname=\empty \xycatcodes
```

4.11.9 Line-styles

The included file `xyps-l.tex` (version 3.4) provides POSTSCRIPT support for the effects defined in the `line` extension described in §2.4.

The POSTSCRIPT style methods are required, so ensure that they are loaded.

```
\expandafter\ifx\csname xyps-sloaded\endcsname\empty\else
 \xyinputorelse@{xyps-s}{\xyerror@{Could not load xyps-s}{}}%
 \xycatcodes\fi
```

This file is loaded and its effects are activated automatically whenever `\xyoption{line}` is requested and the current ⟨driver⟩ supports POSTSCRIPT line styles. Should there be any need to turn off this support, the following commands are available; they obey usual TeX groupings.

`\NoPSlines`	remove POSTSCRIPT support

`\UsePSlines` reinstate PostScript.

Without PostScript support lines can be expected to be displayed in the default style, having thickness of .4pt. These commands are provided primarily for testing and trouble-shooting; *e.g.* with ⟨driver⟩ configurations untested by the authors of XY-pic, who should be notified of any difficulties.

Installation The ⟨driver⟩-file must call the following macro to rebind a control sequence defined in `xyline`, thereby allowing the PostScript method to be used. The purpose of the `\xyPSlinew@` hook is to allow the PostScript style sequencing method to be used with other back-ends, changing just the contents of the `\special`s actually placed.

```
\xydef@\installPSline@{\installPSstyle@
 \installPSlinestyles@ \installPSpolylines@ }
```

```
\xydef@\UsePSlines{\installPSline@}
\xydef@\NoPSlines{\UnloadLine@\relax}
```

widths, joins and caps This, and the rebindings below, is all that is needed for the cleaner, more sophisticated way.

```
\xydef@\xyPSlineSpecial@#1{\addtostyletoks@{ #1 lw}}
\xydef@\xyPScapSpecial@#1{\addtostyletoks@{ #1 lc}}
\xydef@\xyPSjoinSpecial@#1{\addtostyletoks@{ #1 lj}}
\xydef@\xyPSmiterSpecial@#1{\addtostyletoks@{ #1 ml}}
```

```
\xydef@\installPSlinestyles@{%
 \let\xylinewidth@@=\xylinewidth@
 \let\transxyline@@=\transxyline@
 \let\xylineSpecial@@=\xyPSlineSpecial@
 \let\resetxyline@@=\resetxyline@i
%
 \let\xy@linecap@@=\xy@linecap@
 \let\xycapSpecial@@=\xyPScapSpecial@
 \let\resetxylinecap@@=\resetxylinecap@i
%
 \let\xy@linejoin@@=\xy@linejoin@
 \let\xyjoinSpecial@@=\xyPSjoinSpecial@
 \let\resetxylinejoin@@=\resetxylinejoin@i
%
 \let\xy@linemiter@@=\xy@linemiter@
 \let\xymiterSpecial@@=\xyPSmiterSpecial@
 \let\resetxylinemiter@@=\resetxylinemiter@i
}
```

The PostScript operators `lw`, `lc`, `lj`, `ml` are defined in `xyps-ps` to set the `linewidth`, `linecap`, `linejoin`, `miterlimit` graphics-state parameters, respectively.

```
\xydef@\xypolylinePS@Special#1{\PSmacro@{#1 pl}}
\xydef@\xypolydotPS@Special#1{\PSmacro@{#1 pt}}
\xydef@\xypolydashPS@Special#1{\PSmacro@{#1 pd}}
\xydef@\xypolyfillPS@Special#1{\PSmacro@{#1 pf}}
\xydef@\xypolyeofillPS@Special#1{\PSmacro@{#1 pe}}
```

```
\xydef@\installPSpolylines@{%
 \let\xypolyline@Special=\xypolylinePS@Special
 \let\xypolydot@Special=\xypolydotPS@Special
 \let\xypolydash@Special=\xypolydashPS@Special
 \let\xypolyfill@Special=\xypolyfillPS@Special
 \let\xypolyeofill@Special=\xypolyeofillPS@Special
 \let\xy@polystyle@@=\xy@polystyle@ }
```

Poly-lines

The end & Log

```
\endinput
```

```
% $Log: xyps-l.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  10:05:50  ross
% minor improvements to file-loading commands
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.9  1994/06/09  14:38:56  ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% Revision 2.9  1994/06/09  14:38:56  ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% NEW for version 2.9 by by Ross Moore.
```

4.11.10 xyps-r.doc

Header:

```
%% $Id: xyps-r.doc,v 3.3 1996/12/18 10:05:50 ross Exp $
%%
%% Xy-pic ``PostScript Rotations and Scaling''.
```

```
%% Copyright (c) 1993-1996 Ross Moore <ross@mpce.mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\expandafter\ifx\csname xyps-rloaded\endcsname\empty
 \message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-rloaded\endcsname=\empty \xycatcodes
```

4.11.11 Rotations and scaling

The included file `xyps-r.tex` (version 3.3) provides PostScript support for the effects defined in the `rotate` extension described in §2.5.

The PostScript style methods are required, so ensure that they are loaded.

```
\expandafter\ifx\csname xyps-sloaded\endcsname\empty\else
 \xyinputorelse@{xyps-s}{\xyerror@{Could not load xyps-s}{}}%
 \xycatcodes\fi
```

This file is loaded and its effects are activated automatically whenever `\xyoption{rotate}` is requested and the current ⟨driver⟩ supports PostScript rotations. Should there be any need to turn off this support, the following commands are available; they obey usual TeX groupings.

`\NoPSrotate`	remove PostScript support
`\UsePSrotate`	reinstate PostScript.

Without PostScript support diagrams can be expected to be displayed unrotated and unscaled. These commands are provided primarily for testing and trouble-shooting; *e.g.* with ⟨driver⟩ configurations untested by the authors of XY-pic, who should be notified of persistent difficulties.

Installation Call the following macro to allow the PostScript mechanisms to be used.

```
\xydef@\installPSrotscale@{\installPSstyle@
 \let\xyRotate@@=\xyPSrotate@@
% \let\xySpecialRotate@@=\xyPSspecialRotate@
 \let\doSpecialRotate@@=\doPSspecialRotate@@
```

```
% \let\xyRotnamed@=\xyPSpsxy@
% \let\xyRot@named@=\xyPSrot@named@@
 \let\xyscale@@=\xyPSscale@@
 \xyPSshapes@true
 \xyuncatcodes }
```

```
\xydef@\UsePSrotate{\installPSrotscale@}
\xydef@\NoPSrotate{\UnloadRotate@\relax}
```

All effects defined in the `rotate` extension can be implemented using a PostScript ⟨driver⟩. However different ⟨driver⟩s need not handle things in the same way; different PostScript code may be required to match the specific environment used by the ⟨driver⟩.

This file provides generic PostScript code which is known to work correctly with most drivers. It is written so as to indicate where modifications may be made to accomodate specific ⟨driver⟩s. For such driver-specific variations, consult the appropriate ⟨driver⟩-file, called `xy`⟨driver⟩`.doc`.

```
\xydef@\xyPSpretransform@{\xyPSrawA@{\preXYtransform@@}}
\xydef@\xyPSposttransform@{\xyPSrawZ@{\postXYtransform@@}}
```

```
\let\preXYtransform@=\xyPSpretransform@
\let\postXYtransform@=\xyPSposttransform@
```

```
\xydef@\xyPSrotSplit@{%
 \expandafter\def\expandafter\preShape@@\expandafter{\xyPSpre@}%
 \expandafter\def\expandafter\postShape@@\expandafter{\xyPSpost@}}
```

Transform lists

```
\xydef@\xyPSscale@@#1#2{%
 \xyPSsplitPS@{\xyscale@start(#1,#2)}{\xyscale@end}%
 \xyPSrotSplit@\xyPSpsxy@ }
```

Rescaling

```
\xydef@\xyPSrotate@@#1{%
 \xyPSsplitPS@{\xyrot@start(#1 xyd)}{\xyrot@end}%
 \xyPSrotSplit@\xyPSpsxy@ }
```

```
\xydef@\xyPSspecialRotate@#1{%
 \xyPSsplitPS@{\xyrot@start(#1)}{\xyrot@end}\xyPSrotSplit@\xyPSpsxy@ }
```

```
\xydef@\doPSspecialRotate@@#1@@{\xyPSspecialRotate@{#1}}
```

Rotations The PostScript operator `xyr` is defined in `xyps-ps.doc`, to store the given parameter as a rotation angle.

Shearing This feature is not implemented yet.

PostScript commands The PostScript codes to start the rotation or scaling are given as functions, `xyrot@start` and `xyscale@start` with arguments to include a specification of the rotation-angle or scaling factors.

```
\xydef@\xyRotScale@@{%
 \def\xyrot@start(##1){{xyt ##1\space xyr}}%
```

```
\def\xyscale@start(##1,##2){{xyt ##1\space##2\space xys}}%
\def\xyrot@end{}\def\xyscale@end{}}
```

These strings are placed using `\xyPSsplitPS@` so as to correctly occur before and after the code for the actual ⟨object⟩ being typeset. This is always followed by `\xyPSpsxy@`, which results in PostScript code to identify the current position, storing it with keys `xyXpos` and `xyYpos` so as to be accessible to the code for the rotation or scaling.

A specific ⟨driver⟩ must either call `\xyRotScale@@` to use this default mechanism, or define its own macro to be called at installation time, which sets alternative expansions to `\xyrot@start(#1)`, `\xyscale@start(#1,#2)`, `\xyrot@end` and `\xyscale@end`.

The end & Log

```
\endinput
```

```
% $Log: xyps-r.doc,v $
% Revision 3.3  1996/12/18  10:05:50  ross
% minor improvements to file-loading commands
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% Revision 2.9  1994/06/09  14:38:56  ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% Revision 2.9  1994/06/09  14:38:56  ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% NEW for version 2.9 by by Ross Moore.
```

4.11.12 xyps-t.doc

Header:

```
%% $Id: xyps-t.doc,v 3.4 1997/05/18 01:13:24 ross Exp $
%%
%% Xy-pic ``Pattern and Tile extension'' PostScript support.
%% Copyright (c) 1993-1997 Ross Moore <ross@mpce.mq.edu.au>
%%
```

```
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\expandafter\ifx\csname xyps-tloaded\endcsname\empty
 \message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-tloaded\endcsname=\empty \xycatcodes
```

4.11.13 Patterns and tiles

The included file `xyps-t.tex` (version 3.4) provides POSTSCRIPT support for the effects defined in the `tile` extension described in §2.7.

The POSTSCRIPT style methods are required, so ensure that they are loaded.

```
\expandafter\ifx\csname xyps-sloaded\endcsname\empty\else
 \xyinputorelse@{xyps-s}{\xyerror@{Could not load xyps-s}{}}%
 \xycatcodes\fi
```

This file is loaded and its effects are activated automatically whenever `\xyoption{tile}` is requested and the current ⟨driver⟩ supports POSTSCRIPT patterns. Should there be any need to turn off this support, the following commands are available; they obey usual TEX groupings.

`\NoPStiles`	remove POSTSCRIPT support
`\UsePStiles`	reinstate POSTSCRIPT.

Without POSTSCRIPT support tile patterns can be expected to be displayed as solid black. These commands are provided primarily for testing and trouble-shooting; *e.g.* with ⟨driver⟩ configurations untested by the authors of XY-pic, who should be notified of any difficulties.

Installation Installation consists of rebinding the macros `\xypattern@` and `\newxypattern@` and of reloading the standard patterns to include a description for POSTSCRIPT. This installation should be delayed till after `pattern` and the POSTSCRIPT ⟨driver⟩ have been installed.

```
\xydef@\installPSpatterns@{\installPSstyle@
 \let\xypattern@=\xyPSpattern@@
 \let\xypatternwarning@=\relax
 \xystandardpatterns@
```

```
 \xyuncatcodes }
\xylet@\UnloadPSpatterns@=\Unloadpattern@
\xydef@\UsePStiles{\installPSpatterns@}
\xydef@\NoPStiles{\UnloadPSpatterns@\relax}
```

All effects defined in the `tile` extension can be implemented using the PostScript dictionary, loaded by `xyps-ps`. This file provides “generic” PostScript code which is known to work correctly with most drivers. No attempt is made to provide special code for particular drivers; for driver-specific variations, consult the appropriate `xy`⟨driver⟩`.doc` file.

```
\xydef@\xyPSnewpattern@#1#2#3#4#5{%
 \expandafter\DN@\expandafter{\csname shape [#1]\endcsname}%
 \DNii@{\relax}\ifx\next@\nextii@\newxypattern{#1}{}\relax\fi
 \DNii@{#4}\ifx\nextii@\empty
  \expandafter\DNii@\expandafter{\csname xyPSshape@#1@\endcsname}%
  \expandafter\def\nextii@{\xyPSpattern@{#2 #3}}\else
  \expandafter\DNii@\expandafter{\csname xyPSshape@#1@\endcsname}%
  \expandafter\def\nextii@{\xyPSpattern@{#4}}\fi
 \ifx\nextii@\empty\expandafter\def\next@{\xyPSnotimplemented@@`#1'@@}%
 \else
  \edef\tmp@{\noexpand\let\expandafter\noexpand\next@
   \expandafter\noexpand\nextii@}\expandafter\tmp@\fi }
\xydef@\xyPSpattern@{\ifPSspecials@\expandafter\xyPSpattern@@
 \else\expandafter\eat@\fi}%\xyPSpattern@@}
\xydef@\xyPSpattern@@#1{\bgroup \checkXyStyle@
 \def\preStyle@@{\expandafter\addtostyletoks@\expandafter{\xyPSpre@}}%
 \def\postStyle@@{\expandafter\addtostyletoks@\expandafter{\xyPSpost@}}%
 \let\modPSboxz@=\modXYstyle@ \xyPS@pattern@@{#1}%
 \toks@={\egroup\let\xy@style@=\relax \def\Drop@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\Drop@@}\def\Connect@@}%
 \expandafter\addtotoks@\expandafter{\expandafter{\Connect@@}}%
 \the\toks@ }
\xydef@\xyPS@pattern@@#1{\xyPSsplitPS@{{#1}sp}{}}
```

The end & Log

```
\endinput
% $Log: xyps-t.doc,v $
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18  10:05:50  ross
% minor improvements to file-loading commands
%
% Revision 3.1  1995/09/05  20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07  20:13:19  ross
% Major release w/new User's Guide!
```

```
%
% Revision 2.13  1995/07/04  15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.0 by by Ross Moore.
```

Chapter 5

Standard Fonts

This chapter presents the sources of the XY-pic standard fonts used by the kernel for computations and as the default way to draw lines and frames in the DVI output.

To Do: Rewrite the METAFONT code to `mft` format with font tables.

5.1 Semidirectional font

These are fonts with symmetric characters in 64 directions created using the driver file `xyd2.mf`. The standard repertoire includes only one such font: `xydash10`.

5.1.1 xyd2.mf

```
% $Id: xyd2.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYD2: generate characters of ‘Xy-pic SemiDirectional’ font.
% Copyright (c) 1992  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
% -------------------------------------------------------------------------
```

```
%
% CONTENTS: Generate ‘semidirectional’ characters from 0..127 by calling
% ‘chartowards(Code, Dx, Dy)’ with
%
% Code:         0       30       31       63       95       127
% Dx,Dy:   31/32,1  1/32,1      0,-1     1,-1     1,0      1,1
%                /      |                                    /
% Direction:    o       o        o        o        o-       o
%                                |         \
%
% where the intervening codes are got by interpolating either Dx or Dy as
% appropriate.
% -------------------------------------------------------------------------
%
% Parameters:
%
font_coding_scheme:="Xy-pic semidirectional";
%
font_x_height   segx#;
font_quad       segl#;
fontdimen 8:    segw#;                  % default_rule_thickness
%
% Characters:
%
for cc =  0 step 1 until  30: chartowards(cc,(31-cc)/32,        1); endfor;
for cc = 31 step 1 until  63: chartowards(cc,(cc-31)/32,       -1); endfor;
for cc = 64 step 1 until 127: chartowards(cc,         1,(cc-95)/32); endfor;
%
% $Log: xyd2.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User’s Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.7  1992/12/14  01:41:26  kris
% Generate all Xy-pic ‘semidirectionsl’ characters.
%
% Extracted from xyline10.mf [Revision 2.1 1992/01/02 14:54:07 kris]
```

5.1.2 xydash10

```
% $Id: xydash10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
```

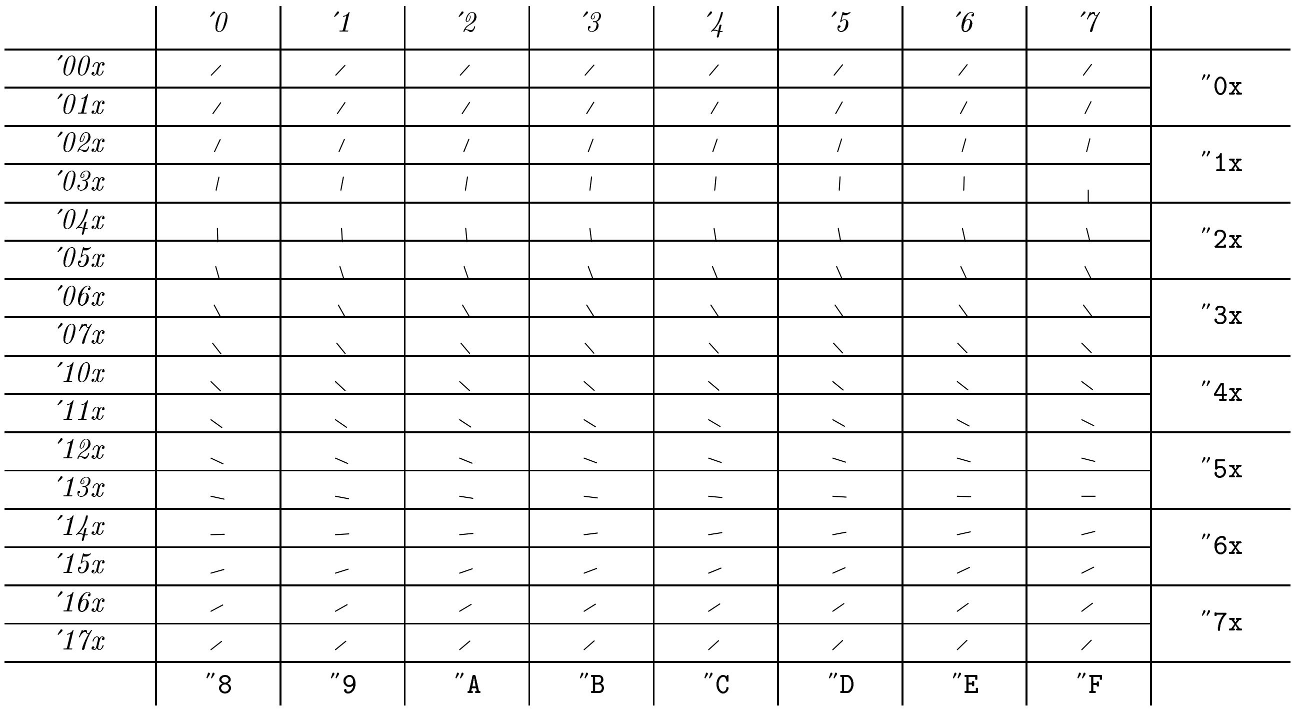


Figure 5.1: Font table for `xydash10`.

```
% XYDASH10: dashes for Xy-pic at 10 point.
% Copyright (c) 1991,1992  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
% ----------------------------------------------------------------------
%
% CONTENTS: Line segments going segl# in all rightward directions...each
```

```
% character has  (width + italic correction) = cos(direction) * segl#.
% ------------------------------------------------------------------------
%
% SETUP:
%
font_identifier "XYDASH"; font_size 10pt#;
mode_setup;

% METANESS...
%
segl# = 1/2 designsize; define_pixels(segl);            % segment length
segx# = 1/5 designsize; define_pixels(segx);            % segment height
segw# = .4pt#; define_whole_blacker_pixels(segw);       % segment thickness
%
pickup pencircle scaled segw; char_pen := savepen;
%
% chartowards makes char cc with length segl line from (0,0) towards (dx,dy)...
%
def chartowards(expr cc,dx,dy) =
 beginchar(cc, abs(dx)*segl#, 0, 0);
  a := angle(dx,dy);
  z0 = (0,0); z1 = z0 + dir a scaled segl;
  pickup char_pen; draw z0--z1;
  charic := abs ((cosd a)*segl#) - charwd;
  penlabels(0,1);
 endchar
enddef;

% TESTING...we redefine openit because the characters extend far to the
% left of the bounding box!
%
def openit = openwindow currentwindow
 from origin to (screen_cols,screen_rows) at (-200,300) enddef;

% GENERATE...
%
input xyd2

bye.
%
% $Log: xydash10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.7  1992/12/14  01:41:26  kris
% Now uses xyd2.mf.
```

```
%
% Based on obsolete xyline10.mf  2.1  1992/01/02  14:54:07  kris
```

5.2 Directional font

These are fonts with characters in 128 directions created using the driver file `xyd.mf`. The standard repertoire includes several such fonts.

5.2.1 xyd.mf

```
% $Id: xyd.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYD: generate characters of `Xy-pic Directional' font.
% Copyright (c) 1992  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
% --------------------------------------------------------------------------
%
% CONTENTS: Generate `directional' characters from 0..127 by calling
% `chartowards(Code, Dx, Dy)' with
%
% Code:         0     15    31    47    63    79    95    111   127
% Dx,Dy:  -15/16,-1  0,-1  1,-1  1,0   1,1   0,1  -1,1  -1,0  -1,-1
%                                         /    |    \
% Direction:    o     o     o     o-    o     o     o     -o     o
%              /      |      \                                  /
%
% where the intervening codes are got by interpolating either Dx or Dy as
% appropriate.
% --------------------------------------------------------------------------
%
% Parameters:
```

```
%
font_coding_scheme:="Xy-pic directional";
%
font_x_height   segx#;
font_quad       segl#;
fontdimen 8:    segw#;                  % default_rule_thickness
%
% Characters:
%
for cc =  0 step 1 until  31: chartowards(cc,(cc-15)/16,       -1); endfor;
for cc = 32 step 1 until  63: chartowards(cc,         1, (cc-47)/16); endfor;
for cc = 64 step 1 until  95: chartowards(cc,(79-cc)/16,        1); endfor;
for cc = 96 step 1 until 127: chartowards(cc,        -1,(111-cc)/16); endfor;
%
% $Log: xyd.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.7  1992/12/14  01:41:26  kris
% Generate all Xy-pic `directional' characters.
%
% Extracted from xyatip10.mf [Revision 2.1 1992/01/02 14:54:07 kris]
```

5.2.2 xyatip10

```
% $Id: xyatip10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYATIP10: upper arrow tips for Xy-pic at 10 point "technical style".
% Copyright (c) 1991-1998  Kristoffer H. Rose  <krisrose@ens-lyon.fr>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
```

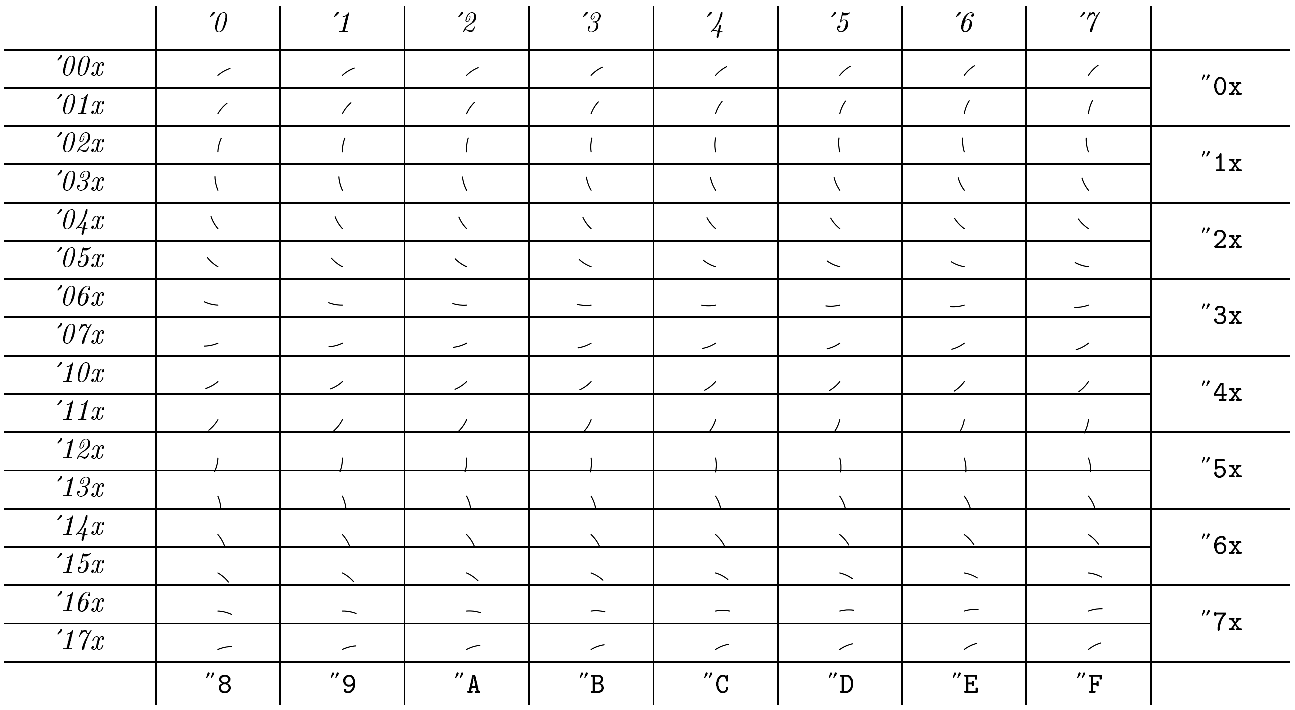


Figure 5.2: Font table for `xyatip10`.

```
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restrictions.  This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYATIP"; font_size 10pt#;
mode_setup;

input xytech
input xyatip

% $Log: xyatip10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.7  1992/12/14  01:41:26  kris
% Now uses xyd.mf.
%
% Revision 2.1  1992/01/02  14:54:07  kris
```

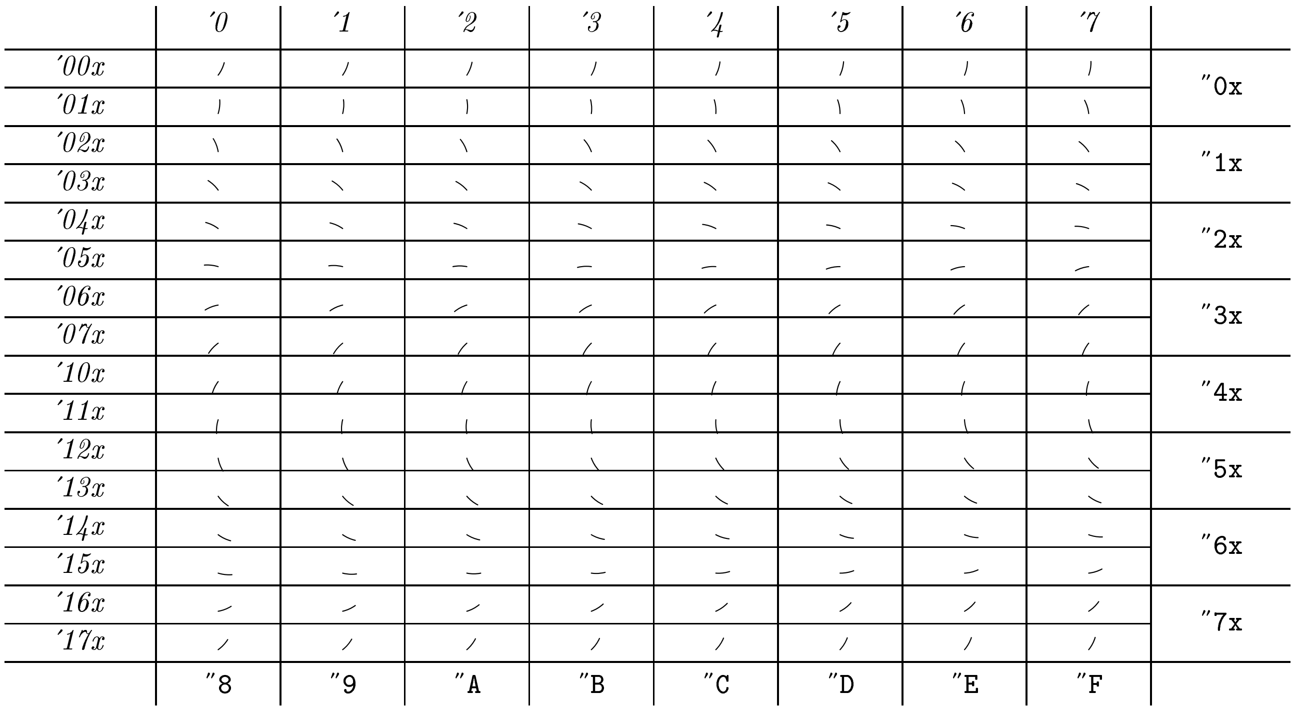


Figure 5.3: Font table for `xybtip10`.

```
% Release version.
%
% Revision 1.1  1991/11/27  06:40:24  kris
% Initial: \beta-test on DIKU.
%
% Based on obsolete xytip10.mf v.1.4.
```

5.2.3 xybtip10

```
% $Id: xybtip10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYBTIP10: lower arrow tips for Xy-pic at 10 point "technical style".
% Copyright (c) 1991,1992  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
```

```
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYBTIP"; font_size 10pt#;
mode_setup;

input xytech
input xybtip

% $Log: xybtip10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.7  1992/12/14  01:41:26  kris
% Now uses xyd.mf.
%
% Revision 2.1  1992/01/02  14:54:07  kris
% Release version.
%
% Revision 1.1  1991/11/27  06:41:03  kris
% Initial: \beta-test on DIKU.
%
% Based on obsolete xytip10.mf v.1.4.
```

5.2.4 xybsql10

```
% $Id: xybsql10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYBSQL10: lower squiggles/quarter circles for Xy-pic at 10 point.
% Copyright (c) 1992  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
```

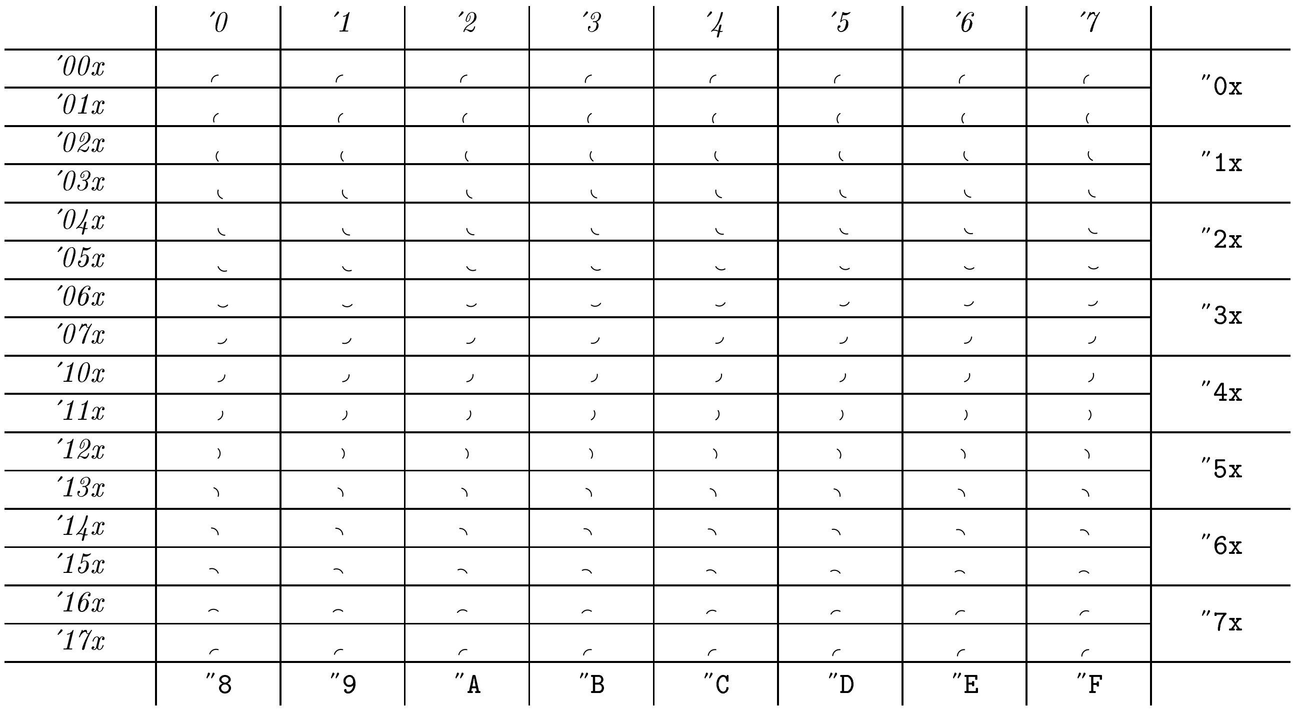


Figure 5.4: Font table for `xybsql10`.

```
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
% -----------------------------------------------------------------------
%
% CONTENTS: Squiggles are quarter circles with secant in all directions.
% Each has zero bounding box around the start point, i.e., from which the
% quarter circle `turns letf' ... this can be illustrated as follows:
%
% Code:    [-1]   15   31   47   63   79   95   111  127
%                                               _
%                                     |    )     |    /\
%             _o     o    o    o    o-   o     o     o   _o
%             |      (    |_   \/                       |
% -----------------------------------------------------------------------
%
font_identifier "XYBSQL"; font_size 10pt#;
```

```
mode_setup;

% METANESS...
%
segl# = sqrt 2 * 1/4 designsize; define_pixels(segl);   % segment length
segx# = (sqrt 2 - 1)*segl#; define_pixels(segx);        % segment height
segw# = .4pt#; define_whole_blacker_pixels(segw);       % segment thickness
%
pickup pencircle scaled segw; char_pen := savepen;
%
% chartowards makes char cc with quarter circle that is secant to line
% from (0,0) to (dx,dy), turning left.
%
def chartowards(expr cc,dx,dy) =
 beginchar(cc,0,0,0);
  a := angle (dx,dy);
  z0 = (0,0); z1 = segl * dir a;
  pickup char_pen; draw z0{dir(a-45)}..{dir(a+45)}z1;
 endchar
enddef;

% GENERATE...
%
input xyd

bye.
%
% $Log: xybsql10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.7  1992/12/14  01:41:26  kris
% Now uses xyd.mf.
%
% Based on obsolete xyqc10.mf [Revision 2.6  1992/06/24  01:23:34  kris]
```

5.3 Special fonts

The kernel circle construction uses the following special font with 1/8 circle segments at various sizes:

5.3.1 xycirc10

```
% $Id: xycirc10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYCIRC10: 1/8 circles with varying radii for Xy-pic at 10 point.
% Copyright (c) 1992,1993  Kristoffer H. Rose  <krisrose@brics.dk>
```

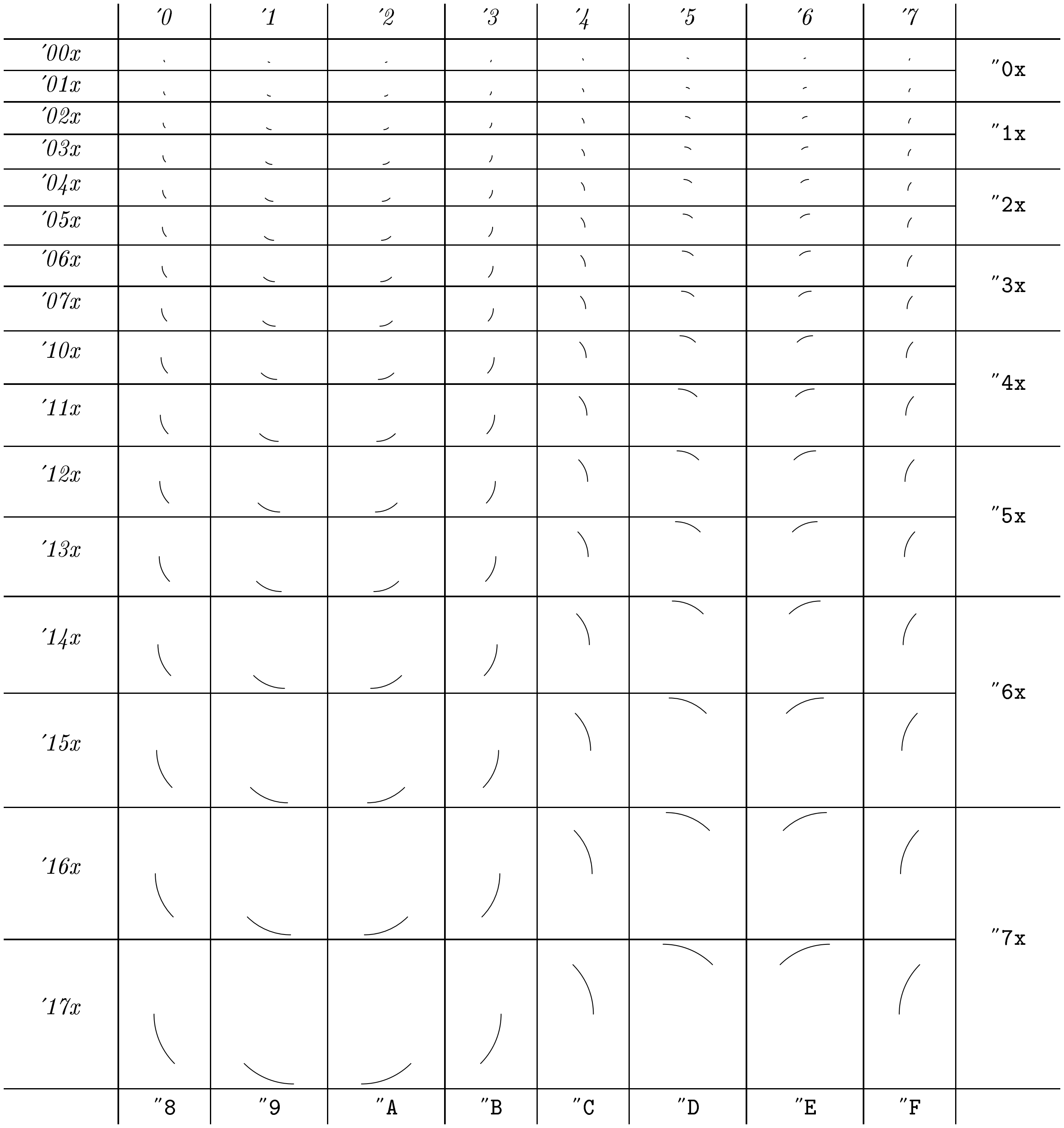


Figure 5.5: Font table for `xycirc10`.

```
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation, Inc.,
% 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
% -------------------------------------------------------------------------
%
% CONTENTS: 1/8 circles with radii from 1 to 32pt dissected into the 1/8
% circle segments shown below
%                                   6     5
%                                   .......         _
%                              7  ..   |   ..  4    |
%                                : |   |   | :      | radius
%                               :  |   |   |  :     |
%       Reference point ---> ___:__|___|___|__:_____|
%                               :  |   |   |  :
%                                : |   |   | :
%                              0  .!   |   !.  3
%                                   ...|...
%                                   1     2
%
% such that each group of 8 characters cc+0 to cc+7 constitute a full circle
% with the given radius when typeset on the same baseline (as shown).
%
% The bounding box of each segment is the vertical slice of the unit square
% around the circle.
%
% The radius is given for each character group g = cc mod 8 by the formula
%
%  radius [pt] = g+1,           if  0 <= g < 8
%                2(g-8) + 10,   if  8 <= g < 12         [= 2g-6]
%                4(g-12) + 20,  if 12 <= g < 16         [= 4g-28]
%
% The bounding box of each segment is the smallest box enclosing it (not
```

```
% including the rule width) with reference point on the horizontal line
% through the center of the circle---so segment 1256 have either height or
% depth negative.
% -------------------------------------------------------------------------
%
font_identifier "XYCIRC"; font_size 10pt#;
mode_setup;

% METANESS...
%
rulew# = .4pt#; define_whole_blacker_pixels(rulew); % line thickness
%
pickup pencircle scaled rulew; char_pen := savepen;

% TESTING...we redefine openit because the characters extend far to the
% left of the bounding box!
%
def openit = %let echar = endchar; def endchar = echar; stop ". " enddef;
 openwindow currentwindow from origin to (1000,800) at (-200,300)
enddef;

% FONT.
%
% Font dimension 8 is the rule thickness (cf. The TeXbook, app.G)
%
font_coding_scheme:="Xy-pic 1/8 circles";
fontdimen 8:    rulew#;                 % default_rule_thickness

% Use the following macro that generates an entire group from character cc
% to cc+7 with radius:
%
def makeg(expr g,radius) =

 minor# := (1 - sqrt 1/2) * radius;
 major# := radius - minor#;
 full#  := radius;
 define_pixels(minor,major,full);

 beginchar(8g  ,minor#,full#,full#); pickup char_pen;
  z0=(0,0); z1=(minor,-major); draw z0{0,-1}..{1,-1}z1;
  penlabels(0,1); endchar;

 beginchar(8g+1,major#,full#,full#); pickup char_pen;
  z0=(0,-major); z1=(major,-full ); draw z0{1,-1}..{1,0}z1;
  penlabels(0,1); endchar;

 beginchar(8g+2,major#,full#,full#); pickup char_pen;
  z0=(0,-full); z1=(major,-major); draw z0{1,0}..{1, 1}z1;
  penlabels(0,1); endchar;

 beginchar(8g+3,minor#,full#,full#); pickup char_pen;
  z0=(0,-major); z1=(minor,0); draw z0{1, 1}..{0, 1}z1;
  penlabels(0,1); endchar;

 beginchar(8g+4,minor#,full#,full#); pickup char_pen;
  z0=(0, major); z1=(minor,0); draw z0{1,-1}..{0,-1}z1;
```

```
  penlabels(0,1); endchar;

 beginchar(8g+5,major#,full#,full#); pickup char_pen;
  z0=(0, full); z1=(major, major); draw z0{1,0}..{1,-1}z1;
  penlabels(0,1); endchar;

 beginchar(8g+6,major#,full#,full#); pickup char_pen;
  z0=(0, major); z1=(major, full); draw z0{1, 1}..{1,0}z1;
  penlabels(0,1); endchar;

 beginchar(8g+7,minor#,full#,full#); pickup char_pen;
  z0=(0,0); z1=(minor, major); draw z0{0, 1}..{1, 1}z1;
  penlabels(0,1); endchar

enddef;

% Make groups:
%
for g =  0 step 1 until  7: makeg(g,  (g+1)*pt#); endfor;
for g =  8 step 1 until 11: makeg(g, (2g-6)*pt#); endfor;
for g = 12 step 1 until 15: makeg(g,(4g-28)*pt#); endfor;

bye.

% $Log: xycirc10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.7  1994/03/08  02:09:27  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% NEW file to go in version 2.7!
% Based on xymisc10.mf [Revision 2.6  1992/06/24  01:23:34  kris]
```

5.4 Optional fonts

The “tips” extension uses the following two fonts by default.

5.4.1 xycmat10

```
% $Id: xycmat10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYCMAT10: upper arrow tips for Xy-pic at 10 point "Computer Modern style".
% Copyright (c) 1994-1996  Kristoffer H. Rose  <krisrose@brics.dk>
%
```

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x									″0x
´01x									
´02x									″1x
´03x									
´04x									″2x
´05x									
´06x									″3x
´07x									
´10x									″4x
´11x									
´12x									″5x
´13x									
´14x									″6x
´15x									
´16x									″7x
´17x									
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure 5.6: Font table for `xycmat10`.

```
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYCMAT"; font_size 10pt#;
mode_setup;

input xycm
input xyatip
```

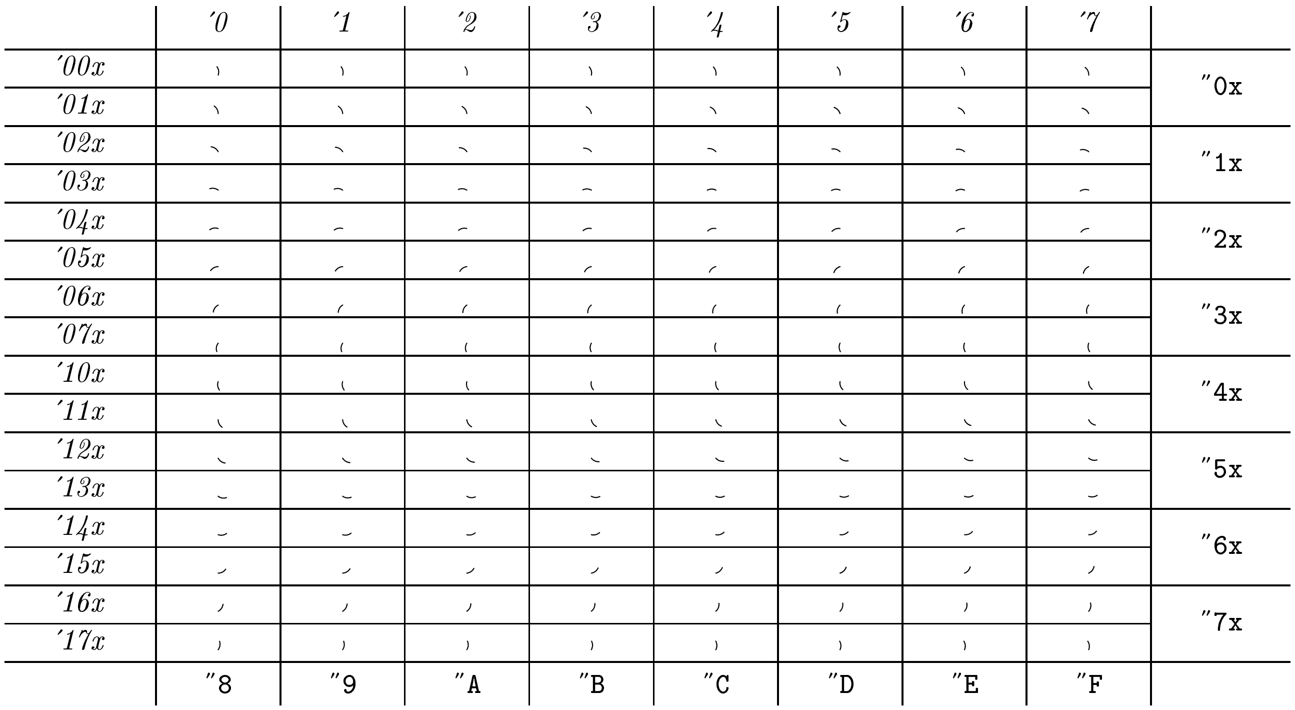


Figure 5.7: Font table for `xycmbt10`.

```
% $Log: xycmat10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Recreated from xyatip10.mf (Revision 2.7 1992/12/14 01:41:26 kris).
%
% Revision 2.6  1993/10/21  21:21:24  kris
% NEW for 2.7...
%
% Based on xyatip10.mf [2.6] and Knuth's SYMBOL.MF.
```

5.4.2 xycmbt10

```
% $Id: xycmbt10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp krisrose $
%
% XYCMBT10: lower arrow tips for Xy-pic at 10 point "Computer Modern style".
```

```
% Copyright (c) 1994-1996  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYCMBT"; font_size 10pt#;
mode_setup;

input xycm
input xybtip

% $Log: xycmbt10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Recreated from xybtip10.mf (Revision 2.7 1992/12/14 01:41:26 kris).
%
% Revision 2.6  1993/10/21  21:21:24  kris
% NEW for 2.7...
%
% Based on xyatip10.mf [2.6] and Knuth's SYMBOL.MF.
```

The following fonts are made to match with the Euler math fonts.

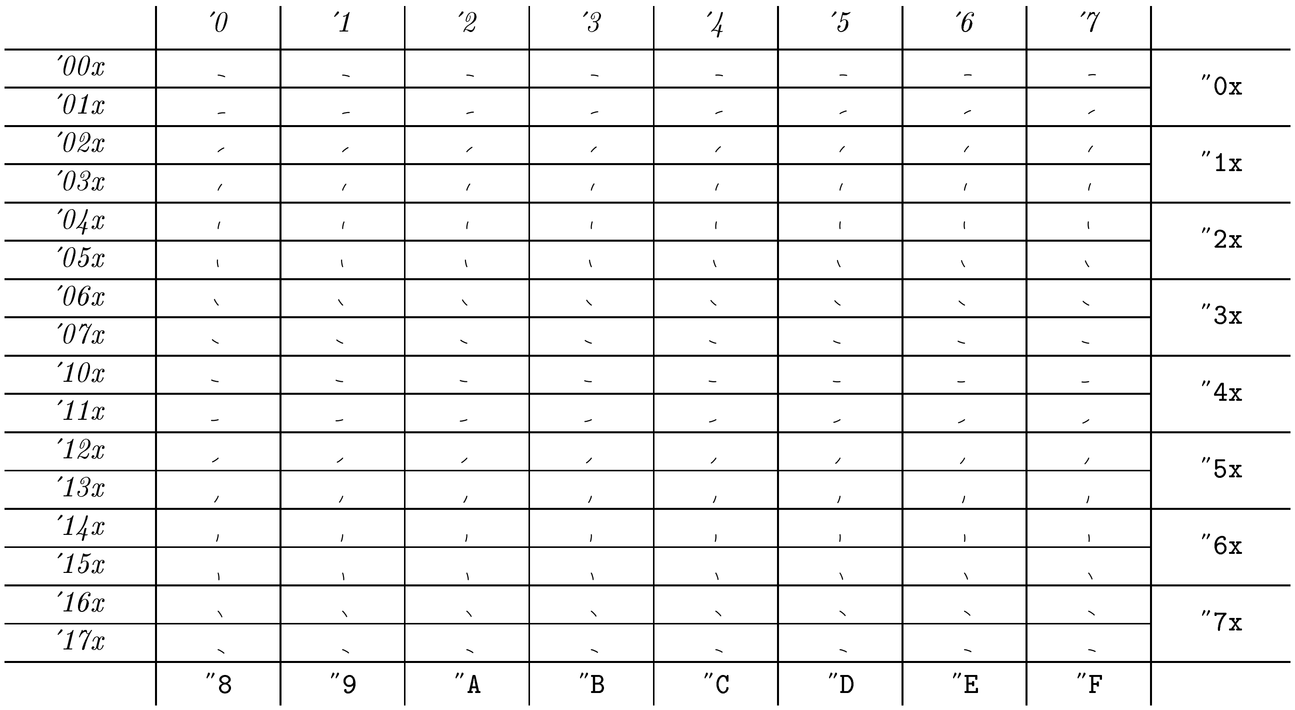


Figure 5.8: Font table for `xyeuat10`.

5.4.3 xyeuat10

```
% $Id: xyeuat10.mf,v 3.3 1996/12/19 04:12:13 krisrose Exp krisrose $
%
% XYEUAT10: upper arrow tips for Xy-pic at 10 point "Euler style".
% Copyright (c) 1994-1996  Kristoffer H. Rose  <kris@diku.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
```

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x									˝0x
´01x									
´02x									˝1x
´03x									
´04x									˝2x
´05x									
´06x									˝3x
´07x									
´10x									˝4x
´11x									
´12x									˝5x
´13x									
´14x									˝6x
´15x									
´16x									˝7x
´17x									
	˝8	˝9	˝A	˝B	˝C	˝D	˝E	˝F	

Figure 5.9: Font table for `xyeubt10`.

```
% version 3.7 of Xy-pic.
%
font_identifier "XYEUAT"; font_size 10pt#;
mode_setup;

input xyeuler
input xyatip

% $Log: xyeuat10.mf,v $
% Revision 3.3  1996/12/19 04:12:13  krisrose
% New for this maintenance release.
%
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp
```

5.4.4 xyeubt10

```
% $Id: xyeubt10.mf,v 3.3 1996/12/19 04:12:13 krisrose Exp krisrose $
%
% XYEUBT10: lower arrow tips for Xy-pic at 10 point "Euler style".
% Copyright (c) 1994-1996  Kristoffer H. Rose  <kris@diku.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
```

```
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation,
% Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYEUBT"; font_size 10pt#;
mode_setup;

input xyeuler
input xybtip

% $Log: xyeubt10.mf,v $
% Revision 3.3  1996/12/19 04:12:13  krisrose
% New for this maintenance release.
%
% Based on xycmbt10.mf,v 3.0 1995/07/07 20:14:21 kris Exp
```

Appendix A

Answers to all exercises

Answer to exercise 1.1 (p.22): In the default setup they are all denote the reference point of the XY-picture but the cartesian coordinate ⟨pos⟩ `(0,0)` denotes the point *origo* that may be changed to something else using the `:` operator.

Answer to exercise 1.2 (p.28): Use the ⟨pos⟩ition `<`X`,`Y`>+"ob"`.

Answer to exercise 1.3 (p.29): It first sets c according to "...". Then it changes c to the point right of c at the same distance from the right edge of c as its width, w, *i.e.*,

$$\underbrace{\text{The}\ \ldots}_{w}\underbrace{\qquad\quad}_{w}\times$$

Answer to exercise 1.4 (p.38): The ⟨coord⟩ "`{"A";"B": "C";"D", x}`" returns the cross point. Here is how the author typeset the diagram in the exercise:

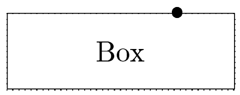
```
\xy
 %
 % set up and mark A, B, C, and D:
 (0,0)="A"  *\cir<1pt>{}*+!DR{A},
 (7,10)="B" *\cir<1pt>{}*+!DR{B},
 (13,8)="C" *\cir<1pt>{}*+!DL{C},
 (15,4)="D" *\cir<1pt>{}*+!DL{D},
 %
 % goto intersection and name+circle it:
 {"A";"B":"C";"D",x} ="I" *\cir<3pt>{},
 %
 % make dotted lines:
 "I";"A"**{} +/1pc/;-/1pc/ **@{..},
 "I";"D"**{} +/1pc/;-/1pc/ **@{..}
 %
\endxy
```

A `?!`... ⟨place⟩ could also have been used.

Answer to exercise 1.5 (p.40): To copy the p value to c, *i.e.*, equivalent to "`p`".

Answer to exercise 1.6 (p.40): When using the kernel connections that are all straight there is no difference, *e.g.*, `**{}?<` and `**{}+E` denote exactly the same position. However, for other connections it is not necessarily the case that the point where the connection enters the current object, denoted by `?<`, and the point where the straight line from p enters the object, denoted by `+E`, coincide.

Answer to exercise 1.7 (p.40): The code typesets the picture



Answer to exercise 1.8 (p.41): This does the job, saving each point to make the previous point available for the next piece:

```
\xy
 @={(0,-10),(10,3),(20,-5)},
 s0="prev" @@{;"prev";**@{-}="prev"}
\endxy
```

Notice how we close the line by first saving `s0`, the last point visited, such that the first point will be connected to it.

Answer to exercise 1.9 (p.44): The author used

```
\xy ={.{+DL(2)}.{+UR(2)}}"dbl",
 *+<3pc,2pc>{+}*\frm{.}, "dbl"*\frm{--}
\endxy
```

to typeset the figure in the exercise.

Answer to exercise 1.10 (p.51): The first typesets "a" centered around `0` and then moves c to the lower right corner, the second typesets "a" above the `0` point and does not change c. With a "+" at `0` they look like this:  and .

Answer to exercise 1.11 (p.52): They have the outlines

 and 

because the first is enlarged by the positive offset to the upper right corner and the second by the negative offset to the lower left corner.

Answer to exercise 1.12 (p.64): The first has no effect since the direction is set to be that of a vector in the current direction, however, the second reverses the current direction.

Answer to exercise 1.13 (p.64): None in both cases.

Answer to exercise 1.14 (p.79): One way is

```
$$\xy
 *{+}; p+(6,3)*{+} **{} ?(1)
 *@{-} *!/-5pt/^\dir{-}
 *^\dir{-} *!/^-5pt/\dir{-}
\endxy$$
```

Thus we first create the two +s as p and c and connect them with the dummy connection `**{}` to setup the direction parameters. Then we move 'on top of c' with `?(1)` and position the four sides of the square using `^` and `_` for local direction changes and `/`⟨dimen⟩`/` for skewing the resulting object by moving its reference point in the opposite direction.

Answer to exercise 1.15 (p.88): One way is to add extra half circles skewed such that they create the illusion of a shade:

```
$$\xy
 *\cir<5pt>{}
 *!<-.2pt,.2pt>\cir<5pt>{dr^ul}
 *!<-.4pt,.4pt>\cir<5pt>{dr^ul}
 *!<-.6pt,.6pt>\cir<5pt>{dr^ul}
\endxy$$
```

Answer to exercise 2.1 (p.121): This is the code that was actually used:

```
\xy (0,20)*[o]+{A};(60,0)*[o]+{B}="B"
**\crv{} \POS?(.4)*_+!UR{0},"B"
**\crv{(30,30)} \POS?*^+!D{1},"B"
**\crv{(20,40)&(40,40)} \POS?*^+!D{2},"B"
**\crv{(10,20)&(30,20)&(50,-20)&(60,-10)}
\POS?*+^!UR{4} \endxy
```

Answer to exercise 2.2 (p.121): This is the code that was used to typeset the picture:

```
\xy (0,20)*+{A};(60,0)*+{B}
**\crv{(10,20)&(30,20)&(50,-20)&(60,-10)}
 ?<*\dir{<} ?>*\dir{>}
 ?(.65)*{\oplus} *!LD!/^-5pt/{x}
 ?(.65)/12pt/*{\oplus} *!LD!/^-5pt/{x'}
 ?(.28)*=0{\otimes}-/40pt/*+{Q}="q"
 +/100pt/*+{P};"q" **\dir{-}
\endxy
```

Answer to exercise 2.3 (p.122): Here is the code that was used to typeset the picture:

```
\def\ssz#1{\hbox{$_{^{#1}}$}}
\xy (0,0)*+{A};(30,-10)*+{B}="B",**\dir{-},
"B"**\crv{(5,20)&(20,25)&(35,20)}
 ?<(0)*\dir{<}="a" ?>(1)*\dir{>}="h"
 ?(.1)*\dir{<}="b" ?(.9)*\dir{>}="i"
 ?(.2)*\dir{<}="c" ?(.8)*\dir{>}="j"
 ?(.3)*\dir{<}="d" ?(.7)*\dir{>}="k"
 ?(.4)*\dir{<}="e" ?(.6)*\dir{>}="l"
 ?(.5)*\dir{|}="f",
 "a"*!RC\txt{\ssz{(\lt)}};
  "h"*!LC\txt{\ssz{\;(\gt)}},**\dir{.},
 "b"*!RD{\ssz{.1}};
  "i"*!L{\ssz{\;.9}},**\dir{-},
```

```
 "c"*!RD{\ssz{.2}};
  "j"*!L{\ssz{\;.8}},**\dir{-},
 "d"*!RD{\ssz{.3}};
  "k"*!L{\ssz{\;.7}},**\dir{-},
 "e"*!RD{\ssz{.4}};
  "l"*!LD{\ssz{.6}},**\dir{-},
 "f"*!D!/^-3pt/{\ssz{.5}}
\endxy
```

Answer to exercise 2.4 (p.200): Here is how:

```
\xy
 (0,0)  *++={A} *\frm{o} ;
 (10,7) *++={B} *\frm{o} **\frm{.}
\endxy
```

Answer to exercise 2.5 (p.203): The `*\cir {}` operation changes c to be round whereas `*\frm {o}` does not change c at all.

Answer to exercise 2.6 (p.206): Here is how:

```
\xy
 (0,0)  *+++{A} ;
 (10,7) *+++{B} **\frm{.}
 **\frm{^\}} ; **\frm{_\}}
\endxy
```

The trick in the last line is to ensure that the reference point of the merged object to be braced is the right one in each case.

Answer to exercise 2.7 (p.233): This is how the author specified the diagram:

```
\UseCrayolaColors
\xy\drop[*1.25]\xybox{\POS
(0,0)*{A};(100,40)*{B}**{}
 ?<<*[@_][red][o]=<5pt>{\heartsuit};
 ?>>>*[@_][Plum][o]=<3pt>{\clubsuit}
 **[|*][|.5pt][thicker]\dir{-},
?(.1)*[left]!RD\txt{label 1}*[red]\frm{.}
?(.2)*[!gsave newpath
  xyXpos xyYpos moveto 50 dup rlineto
  20 setlinewidth 0 0 1 setrgbcolor stroke
  grestore][psxy]{.},
?(.2)*[@]\txt{label 2}*[red]\frm{.},
?(.2)*[BurntOrange]{*},
?(.3)*[halfsize]\txt{label 3}*[red]\frm{.}
?(.375)*[flip]\txt{label 4}*[red]\frm{.}
?(.5)*[dblsize]\txt{label 5}*[red]\frm{.}
?(.5)*[WildStrawberry]{*},
?(.7)*[hflip]\txt{label 6}*[red]\frm{.}
?(.8)*[vflip]\txt{label 7}*[red]\frm{.}
```

```
?(.9)*[right]!LD\txt{label 8}*[red]\frm{.}
?(.5)*[@][*.66667]!/^30pt/
 \txt{special effect: aligned text}
 *[red]\frm{.}
}\endxy
```

Answer to exercise 3.1 (p.305): Here is what the author did:

```
\xy *+{A}*\cir<10pt>{}="me"
 \PATH `ul^ur,"me" "me" |>*:(1,-.25)\dir{>}
\endxy
```

The trick is getting the arrow head right: the `:` modifier to the explicit `\dir` ⟨object⟩ does that.

Answer to exercise 3.2 (p.306): The author did

```
\xy(0,0)
 \ar @{-->} (30,7) ^A="a"
 \POS(10,12)*+\txt{label} \ar "a"
\endxy
```

Answer to exercise 3.3 (p.312): Here is the entire XY-picture of the exercise:

```
\xy ;<1pc,0pc>:
 \POS(0,0)*+{A}
 \ar    +(-2,3)*+{A'}*\cir{}
 \ar @2 +( 0,3)*+{A''}*\cir{}
 \ar @3 +( 2,3)*+{A'''}*\cir{}
 \POS(6,0)*+{B}
 \ar @1{||.>>} +(-2,3)*+{B'}*\cir{}
 \ar @2{||.>>} +( 0,3)*+{B''}*\cir{}
 \ar @3{||.>>} +( 2,3)*+{B'''}*\cir{}
\endxy
```

The first batch use the default `{->}` specification.

Answer to exercise 3.4 (p.312): The author used

```
\newdir{ >}{{}*!/-5pt/\dir{>}}
```

Answer to exercise 3.5 (p.313): The author used

```
\xy
 \ar @{>>*\composite{\dir{x}*\dir{+}}<<}
 (20,7)
\endxy
```

Answer to exercise 3.6 (p.314): The author used

```
\xy *{\circ}="b" \ar@(ur,ul) c
 \ar@{.>}@(dr,ul) (20,0)*{\bullet}
\endxy
```

Note that it is essential that the curving specification comes after the arrow style.

Answer to exercise 3.7 (p.325): Here is the code used to typeset the *pasting diagram* in figure 3.3.

```
\xymatrixrowsep{1.5pc}
\xymatrixcolsep{3pc}
\diagram
 &&\relax\rtwocell<0>^{f_3^{}\;\;}{\omit}
 &\relax\ddtwocell<0>{\omit}
  \drtwocell<0>^{\;\;f_4^{}}{<3>}
  \ddrrtwocell<\omit>{<8>}\\
&&&&\relax\drtwocell<0>^{\;\;f_5^{}}{\omit}\\
A \uurrlowertwocell<-6>{\omit}\relax
\uurrcompositemap<2>_{f_1^{}}^{f_2^{}}{<.5>}
 \drtwocell<0>_{g_1^{}\;}{\omit}
 &&&\relax\urtwocell<0>{\omit}
 &&\relax\rtwocell<0>^{f_6^{}\;}{\omit}
 &\relax\rlowertwocell<-3>_{g_4^{}}{<-1>}
 \rcompositemap<6>_{f_7^{}}^{f_8^{}}{\omit}
 & B \\
 &\relax\urrtwocell<0>{\omit}
 \xcompositemap[-1,4]{}%
  <-4.5>_{g_2^{}}^{g_3^{}}{\omit}\\
\enddiagram
```

For the straight arrows, it would have been simpler to use `\..to` provided `xyarrow` has been loaded. Instead `\..twocell<0>...{\omit }` was used to illustrate the versatility of nudging and `\omit` ; thus `xy2cell` can completely handle a wide range of diagrams, without requiring `xyarrow`. Note also the use of `\relax` at the start of each new cell, to avoid premature expansion of a complicated macro, which can upset the compiling mechanism.

Answer to exercise 3.8 (p.329): Here is the code used by the author to set the first diagram.

```
{\uppercurveobject{{?}}
 \lowercurveobject{{\circ}}
\xymatrixcolsep{5pc}
\xymatrixrowsep{2pc}
\diagram
 \relax\txt{ FUn }\rtwocell<8>{!\&}
 & \relax\txt{ gaMES }
 \enddiagram}
```

Here is the code used for the second diagram.

```
\xymatrixcolsep{2.5pc}
\xymatrixrowsep{4pc}
\diagram
 \relax\txt<1.5cm>{\bf Ground State}
 \rrtwocell<12>~^{+{}~**!/-2.5pt/\dir{>}}
 ~_{++{}~**!/5pt/\dir{<<}}
 ^{<1.5>\txt{\small continuous power}}
 _{<1.5>\txt{\small pulsed emission}}{!}
```

```
& \relax\;\; N\!i\,C\!d\;\; \Circled
& \relax\txt<1.50cm>{\bf Excited State}
\enddiagram
```

Answer to exercise 3.9 (p.357): The author did

```
\xymatrix @!=1pc {
 **[l] A\times B
     \ar[r]^{/A} \ar[d]_{/B}
 & B \ar[d]^{\times A}
\\
 A   \ar[r]_{B\times}
 & **[r] B\times A
}
```

Answer to exercise 3.10 (p.357): Modifiers are used to make all entries round with a frame – the general form is used to ensure that the sequence is well-defined. Finally the matrix is rotated to make it possible to enter it as a simple square:

```
\entrymodifiers={=<1pc>[o][F-]}
\xymatrix @ur {
 A \save[];[r] **\dir{-},
        [];[dr]**\dir{-},
        [];[d] **\dir{-}\restore
   & B \\
 C & D }
```

Answer to exercise 3.11 (p.359): Here is how:

```
\xymatrix @W=3pc @H=1pc @R=0pc @*[F-] {%
 : \save+<-4pc,1pc>*\hbox{\it root}
     \ar[]
   \restore
\\
 {\bullet}
   \save*{}
    \ar`r[dd]+/r4pc/`[dd][dd]
   \restore
\\
 {\bullet}
   \save*{}
    \ar`r[d]+/r3pc/`[d]+/d2pc/
          `[uu]+/l3pc/`[uu][uu]
   \restore
\\
 1 }
```

Answer to exercise 3.12 (p.365): The first A was named to allow reference from the last:

```
\xygraph{
```

```
 []A="A1" :@/^/ [r]A
          :@/^/ [r]A
          :@/^/ "A1" }
```

Answer to exercise 3.13 (p.368): The author did

```
\SelectTips{cm}{}
\objectmargin={1pt}
\xygraph{ !{0;(.77,-.77):0}
 !~:{@{-}|@{>}}
 w (:[r(.6)]{x_1}
   ,:[d]z:[r]y:[u(.6)]{x_2}:"x_1":"z"
    :@( {"w";"z"}, {"y";"z"})"z":"x_2") }
```

It also shows that one *can* use {}s inside delimited arguments *provided* one adds a space to avoid the {}s being shaved off!

Answer to exercise 3.14 (p.371): Here is the code actually used to typeset the `\xypolygon` s, within an `\xygraph` . It illustrates three different ways to place the numbers. Other ways are also possible.

```
\def\objectstyle{\scriptscriptstyle}
\xy \xygraph{!{/r2pc/:}
[] !P3"A"{\bullet}
"A1"!{+U*++!D{1}} "A2"!{+LD*+!RU{2}}
"A3"!{+RD*+!LU{3}} "A0"
[rrr]*{0}*\cir<5pt>{}
!P6"B"{~<-\cir<5pt>{}}
"B1"1 "B2"2 "B3"3 "B4"4 "B5"5 "B6"6 "B0"
[rrr]0 !P9"C"{~*{\xypolynode}}}\endxy
```

Appendix B

Backwards Compatibility

Backwards compatibility with XY-pic version 2 is ensured by the 'v2' option and its support files.

B.1 Version 2 Compatibility

Vers. 3.4 by Kristoffer H. Rose ⟨krisrose@brics.dk⟩

This section describes the special backwards compatibility with XY-pic version 2: diagrams written according to the "Typesetting diagrams with XY-pic: User's Manual" [13] should typeset correctly with this loaded. The compatibility is available either as an XY-option or through the special files `xypic.sty` and `xypic.tex` described below.

There are a few exceptions to the compatibility: the features described in §B.1.1 below are not provided because they are not as useful as the author originally thought and thus virtually never used. And one extra command is provided to speed up typesetting of documents with XY-pic version 2 diagrams by allowing the new compilation functionality with old diagrams.

The remaining sections list all the obsolete commands and suggest ways to achieve the same things using XY-pic 3.6, *i.e.*, without the use of this option. They are grouped as to what part of XY-pic replaces them; the compilation command is described last.

Note: "version 2" is meant to cover all public releases of XY-pic in 1991 and 1992, *i.e.*, version 1.40 and versions 2.1 through 2.6. The published manual cited above (for version 2.6) is the reference in case of variations between these versions, and only things documented in that manual will be supported by this option![1]

Header:

```
%% $Id: xyv2.doc,v 3.4 1997/05/18 01:14:25 krisrose Exp $
%%
%% Xy-pic ``Version 2 compatibility'' feature.
%% Copyright (c) 1993-1995  Kristoffer H. Rose  <kris@diku.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1995  Kristoffer H. Rose  <kris@diku.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
```

[1] In addition a few of the experimental facilities supported in v2.7–2.12 are also supported.

```
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
\ifx\xyloaded\undefined \input xy \fi
```

```
\xyprovide{v2}{Version 2 Compatibility}{\stripRCS$Revision: 3.4 $}%
 {Kristoffer H.~Rose}{krisrose@brics.dk}%
 {BRICS/Computer Science, University of Aarhus, Ny Munkegade, building 540,
  DK--8000 Aarhus~C}
```

Before we start, the old name macro.

```
\xylet@\XY=\Xy
```

B.1.1 Unsupported incompatibilities

Here is a list of known incompatibilities with version 2 even when the `v2` option is loaded.

- Automatic 'shortening' of arrow tails using `|<<` breaks was a bug and has been 'fixed' so it does not work any more. Put a `|<\hole` break before it.

- The version 2.6 `*` position operator is not available. The version 2.6 construction $t_0\texttt{;}t_1\texttt{*(}x\texttt{,}y\texttt{)}$ should be replaced by the rather long but equivalent construction

$$\texttt{\{}\,t_0\,\texttt{;p+/r/:}\,t_1\,\texttt{="1";p+/u/,x;(0,0);:}$$
$$\texttt{"1";p+/r/,y;(0,0);::(}x\texttt{,}y\texttt{)\}}$$

  In most cases $t_0\texttt{;}t_1\texttt{**\{\}?(}x\texttt{)}$, possibly with a trailing `+/^`…`/`, suffices instead.

- Using $t_0\texttt{;}t_1\texttt{:(}x\texttt{,}y\texttt{)}$ as the target of an arrow command does not work. Enclose it in braces, *i.e.*, write

$$\texttt{\{}\,t_0\,\texttt{;}\,t_1\,\texttt{:(}x\texttt{,}y\texttt{)\}}$$

- The older `\pit`, `\apit`, and `\bpit` commands are not defined. Use `\dir{>}` (or `\tip`) with variants and rotation.

- The even older notation where an argument in braces to `\rto` and the others was automatically taken to be a 'tail' is not supported. Use the supported `|<`… notation.

If you do not use these features then your version 2 (and earlier) diagrams should typeset the same with this option loaded except that sometimes the spacing with version 3 is slightly different from that of version 2.6 which had some spacing bugs.

B.1.2 Obsolete kernel features

The following things are added to the kernel by this option and described here: idioms, obsolete positions, obsolete connections, and obsolete objects. For each we show the suggested way of doing the same thing without this option:

Removed AMS-TeX idioms

Some idioms from AMS-TeX are no longer used by Xy-pic: the definition commands `\define` and `\redefine`, and the size commands `\dsize`, `\tsize`, `\ssize`, and `\sssize`. Please use the commands recommended for your format—for plain TeX these are `\def` for the first two and `\displaystyle`, `\textstyle`, `\scriptstyle`, and `\scriptscriptstyle` for the rest. The `v2` option ensures that they are available anyway.

```
\ifx\redefine\undefined \let\redefine=\def \fi
\ifx\define\undefined \let\define=\xydef@ \fi
```

```
\ifx\dsize\undefined \let\dsize=\displaystyle \fi
\ifx\tsize\undefined \let\dsize=\textstyle \fi
\ifx\ssize\undefined \let\ssize=\scriptstyle \fi
\ifx\sssize\undefined \let\sssize=\scriptscriptstyle \fi
```

Version also 2 used the AMS-TeX `\text` and a (non-object) box construction `\Text` which are emulated—`\text` is only defined if not already defined, however, using the native one (of AMS-TeX or AMS-LaTeX or whatever) if possible. Please use the `\txt` object construction described in §1.6.3 directly since it is more general and much more efficient!

```
\ifx\text\undefined
 \def\text{\relax\textC}%
 \xydef@\textC#1{\relax
  \ifmmode\mathchoice
     {\hbox{\the\textfont0\relax#1}}%
     {\hbox{\the\textfont0\relax#1}}%
     {\hbox{\the\scriptfont0\relax#1}}%
     {\hbox{\the\scriptscriptfont0\relax#1}}%
  \else{\relax#1}\fi}
\fi
```

```
\xydef@\Text{\relax\xyFN@\Text@}
```

```
\xydef@\TextC{\relax\textC}%
```

```
\xydef@\Text@{%
 \addLT@\ifx\next \addGT@{\addLT@\DN@##1}{\A@=##1\Text@i}%
 \else \DN@{\A@=\maxdimen \Text@i}\fi \next@}
```

```
\xydef@\Text@i#1{%
 \ifmmode \mathchoice
   {\expandafter\Text@ii\the\textfont0{#1}}%
   {\expandafter\Text@ii\the\textfont0{#1}}%
   {\expandafter\Text@ii\the\scriptfont0{#1}}%
   {\expandafter\Text@ii\the\scriptscriptfont0{#1}}%
 \else
   \leavevmode \expandafter\Text@ii\the\textfont0{#1}%
 \fi\ignorespaces}
```

```
\xydef@\Text@ii{\hbox\bgroup \txt@i}
```

Obsolete state

Upto version 2.6 users could access the state variables `\cL`, `\cR`, `\cH`, and `\cD`, which are defined.

```
\xylet@\cL=\L@c
\xylet@\cR=\R@c
\xylet@\cD=\D@c
\xylet@\cH=\U@c
```

From v2.7 to 2.12 users could use the names of the state `\dimen` registers `\Xmin`, `\Xmax`, `\Ymin`, and `\Ymax`; `\Xp`, `\Yp` `\Dp`, `\Up`, `\Lp`, and `\Rp`; `\Xc`, `\Yc` `\Dc`, `\Uc`, `\Lc`, and `\Rc`; `\Xorigin`, `\Yorigin`, `\Xxbase`, `\Yxbase`, `\Xybase`, and `\Yybase`. Now the same effect can be achieved using ⟨corner⟩s but `v2` defines the aliases.

```
\xylet@\Xc=\X@c
\xylet@\Yc=\Y@c
\xylet@\Uc=\U@c
\xylet@\Dc=\D@c
\xylet@\Lc=\L@c
\xylet@\Rc=\R@c
```

```
\xylet@\Xp=\X@p
\xylet@\Yp=\Y@p
\xylet@\Up=\U@p
\xylet@\Dp=\D@p
\xylet@\Lp=\L@p
\xylet@\Rp=\R@p
```

```
\xylet@\dX=\d@X
\xylet@\dY=\d@Y
```

```
\xylet@\Xorigin=\X@origin
\xylet@\Yorigin=\Y@origin
\xylet@\Xxbase =\X@xbase
\xylet@\Yxbase =\Y@xbase
\xylet@\Xybase =\X@ybase
\xylet@\Yybase =\Y@ybase
```

```
\xylet@\Xmin=\X@min
\xylet@\Ymin=\Y@min
\xylet@\Xmax=\X@max
\xylet@\Ymax=\Y@max
```

Obsolete position manipulation

In version 2 many things were done using individual ⟨decor⟩ control sequences that are now done using ⟨pos⟩ operators.

Version 2 positioning	Replacement
`\go`⟨pos⟩	`\POS;p,`⟨pos⟩
`\aftergo{`⟨decor⟩`}`⟨pos⟩	`\afterPOS{`⟨decor⟩`};p,`⟨pos⟩

`\merge`	`\POS.p\relax`
`\swap`	`\POS;\relax`
`\Drop{`⟨text⟩`}`	`\drop+{`⟨text⟩`}`

The code is basically that of `xy.doc` 2.6...

```
\xydef@\go{\aftergo{}}
\xydef@\aftergo#1{\xy@\aftergo{\enter@\pfromthec@}\afterPOS{\xy@@\leave@#1}}
\xydef@\merge{\POS.p\relax}
\xydef@\swap{\xy@\swap\swap@}
\xydef@\Drop{\expandafter\drop\entrymodifiers@}
```

Obsolete connections

These connections are now implemented using directionals.

Version 2 connection	Replacement
`\none`	`\connect h\dir{}`
`\solid`	`\connect h\dir{-}`
`\Solid`	`\connect h\dir2{-}`
`\Ssolid`	`\connect h\dir3{-}`
`\dashed`	`\connect h\dir{--}`
`\Dashed`	`\connect h\dir2{--}`
`\Ddashed`	`\connect h\dir3{--}`
`\dotted`	`\connect h\dir{.}`
`\Dotted`	`\connect h\dir2{.}`
`\Ddotted`	`\connect h\dir3{.}`
`\dottedwith{`⟨text⟩`}`	`\connect h{`⟨text⟩`}`

Note how the 'hidden' specifier `h` should be used because version 2 connections did not affect the size of diagrams.

```
\xydef@\none{\xy@\none{}\ignorespaces}
\xydef@\solid{\connect h\dir{-}}
\xydef@\Solid{\connect h\dir2{-}}
\xydef@\Ssolid{\connect h\dir3{-}}
\xydef@\dashed{\connect h\dir{--}}
\xydef@\Dashed{\connect h\dir2{--}}
\xydef@\Ddashed{\connect h\dir3{--}}
\xydef@\dotted{\connect h\dir{.}}
\xydef@\Dotted{\connect h\dir2{.}}
\xydef@\Ddotted{\connect h\dir3{.}}
\xydef@\dottedwith#1{\connect h{#1}}
```

Obsolete tips

These objects all have `\dir`-names now:

Version 2 tip	Replacement
`\notip`	`\dir{}`
`\stop`	`\dir{|}`
`\astop`	`\dir^{|}`
`\bstop`	`\dir_{|}`
`\tip`	`\dir{>}`
`\atip`	`\dir^{>}`
`\btip`	`\dir_{>}`
`\Tip`	`\dir2{>}`
`\aTip`	`\object=<5pt>:(32,-1)\dir^{>}`
`\bTip`	`\object=<5pt>:(32,+1)\dir_{>}`
`\Ttip`	`\dir3{>}`
`\ahook`	`\dir^{(}`
`\bhook`	`\dir_{(}`
`\aturn`	`\dir^{'}`
`\bturn`	`\dir_{'}`

```
\xydef@\notip{\dir{}}

\xydef@\stop{\dir{|}}
\xydef@\astop{\dir^{|}}
\xydef@\bstop{\dir_{|}}

\xydef@\tip{\dir{>}}
\xydef@\atip{\dir^{>}}
\xydef@\btip{\dir_{>}}

\xydef@\Tip{\dir2{>}}

\xywarnifdefined\aTip
\xywarnifdefined\bTip
{\xyuncatcodes
 \gdef\aTip{\object=<5pt>:(32,-1)\dir^{>}}
 \gdef\bTip{\object=<5pt>:(32,+1)\dir_{>}}}

\xydef@\Ttip{\dir3{>}}

\xydef@\ahook{\dir^{(}}
\xydef@\bhook{\dir_{(}}

\xydef@\aturn{\dir^{'}}
\xydef@\bturn{\dir_{'}}
```

The older commands `\pit`, `\apit`, and `\bpit`, are not provided.

Obsolete object constructions

The following object construction macros are made obsolete by the enriched ⟨object⟩ format:

Version 2 object	Replacement
`\rotate(`⟨factor⟩`)`⟨tip⟩	

	`\object:(`⟨factor⟩`,`⟨factor⟩`){`⟨tip⟩`}`
`\hole`	`\object+{}`
`\squash`⟨tip⟩	`\object=0{`⟨tip⟩`}`
`\grow`⟨tip⟩	`\object+{`⟨tip⟩`}`
`\grow<`⟨dimen⟩`>`⟨tip⟩	`\object+<`⟨dimen⟩`>{`⟨tip⟩`}`
`\squarify{`⟨text⟩`}`	`\object+={`⟨text⟩`}`
`\squarify<`⟨dimen⟩`>{`⟨text⟩`}`	
	`\object+=<`⟨dimen⟩`>{`⟨text⟩`}`

where rotation is done in a slightly different manner in version 3.6 (it was never accurate in version 2).

These are mostly just the unmodified version 2.6 code:

```
\xydef@\rotate{\hbox\bgroup
 \DN@{\ifx (\next %)
   \DN@(####1){\rotate@ii{####1}}%
  \else \DN@{\rotate@i}\fi\next@}\xyFN@\next@}

\xydef@\rotate@i#1{\reverseDirection@ #1\OBJECT@x}

\xydef@\rotate@ii#1#2{\dimen@=#1\p@
 \ifdim     \dimen@=-\p@ \aboveDirection@\xydashl@
 \else\ifdim\dimen@= \p@ \belowDirection@\xydashl@
 \else\ifdim\dimen@<-1.5\p@ \advance\dimen@-2\p@
  \edef\next@{\expandafter\removePT@\the\dimen@}\vDirection@(-1,\next@){1pc}%
 \else\ifdim\dimen@<-.5\p@ \vDirection@(#1,-1){1pc}%
 \else\ifdim\dimen@< .5\p@ \vDirection@( 1,#1){1pc}%
 \else\ifdim\dimen@<1.5\p@ \vDirection@(#1, 1){1pc}%
 \else \dimen@=-\dimen@ \advance\dimen@2\p@
  \edef\next@{\expandafter\removePT@\the\dimen@}\vDirection@(-1,\next@){1pc}%
 \fi\fi\fi\fi\fi\fi
 #2\OBJECT@x}

\xydef@\squash#1{%
 \ifmmode\setboxz@h{\m@th${\mathstrut}#1{\mathstrut}$}\else\setboxz@h{#1}\fi
 \setboxz@h{\kern-.5\wdz@ \dimen@=.5\ht\z@ \advance\dimen@-.5\dp\z@
  \lower\dimen@\boxz@}%
 \wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}

\xydef@\grow{%
 \DN@{\ifx <\next \DN@<####1>{\dimen@=####1\grow@\empty}%
  \else \DN@{\dimen@=2\p@ \grow@\empty}\fi\next@}\xyFN@\next@}

\xydef@\grow@#1#{\grow@@{#1}}
\xydef@\grow@@#1#2{\relax
 \ifmmode \setboxz@h{\kern\dimen@ \m@th$#2$\kern\dimen@}%
 \else    \setboxz@h{\kern\dimen@ {#2}\kern\dimen@}\fi
 \dimen@ii=\ht\z@ \advance\dimen@ii\dimen@ \ht\z@=\dimen@ii
 \dimen@ii=\dp\z@ \advance\dimen@ii\dimen@ \dp\z@=\dimen@ii \boxz@}

\xydef@\squarify{\addEQ@{\addPLUS@\object}}
```

We also reimplement the version 2 undocumented `\qc` quarter circle which is now called `\corn@@` of the frame extension.

```
\xydef@\qc{\corn@@}
```

B.1.3 Obsolete extensions & features

Version 2 had commutative diagram functionality corresponding to the `frames` extension and parts of the `matrix` and `arrow` features. These are therefore loaded and some extra definitions added to emulate commands that have disappeared.

Frames

The version 2 frame commands are emulated using the frame extension (as well as the `\dotframed`, `\dashframed`, `\rounddashframed` commands communicated to some users by electronic mail):

Version 2 object	Replacement
`\framed`	`\drop\frm{-}`
`\framed<`⟨dimen⟩`>`	`\drop\frm<`⟨dimen⟩`>{-}`
`\Framed`	`\drop\frm{=}`
`\Framed<`⟨dimen⟩`>`	`\drop\frm<`⟨dimen⟩`>{=}`
`\dotframed`	`\drop\frm{.}`
`\dashframed`	`\drop\frm{--}`
`\rounddashframed`	`\drop\frm{o-}`
`\circled`	`\drop\frm{o}`
`\Circled`	`\drop\frm{oo}`

For each of the above commands (and a few hidden ones) we parse the above and convert to the proper format.

```
\xyrequire{frame}\xycatcodes

\xydef@\framed{\relax\convertframed{-}}
\xydef@\Framed{\relax\convertframed{=}}

\xydef@\circled{\relax\convertframed{o}}
\xydef@\Circled{\relax\convertframed{oo}}

\xydef@\convertframed#1{\DNii@##1@{\drop\frm##1{#1}}\xyFN@\convertframed@i}

\xydef@\convertframed@i{%
 \addLT@\ifx \next
  \addGT@{\addLT@\DN@##1}{\addGT@{\addLT@{\nextii@}##1}@}%
 \else \DN@{\nextii@ @}\fi \next@}

\xydef@\dotframed{\relax\drop\frm{.}}
\xydef@\dashframed{\relax\drop\frm{--}}
\xydef@\rounddashframed{\relax\drop\frm{o-}}

\xydef@\filled{\DN@{%
  \addLT@\ifx \next
   \addGT@{\addLT@\DN@####1}{\xy@@{\frame@fill@@{####1}}}%
  \else \DN@{\xy@@{\frame@fill@@\z@}}%
  \fi\next@}\xyFN@\next@}

\xydef@\Filled{\DN@{%
  \addLT@\ifx \next
   \addGT@{\addLT@\DN@####1}{\xy@@{\frame@emph@@{####1}}}%
  \else \DN@{\xy@@{\frame@emph@@\z@}}%
  \fi\next@}\xyFN@\next@}
```

Matrices

The `\diagram` ⟨rows⟩ `\enddiagram` command is provided as an alias for `\xymatrix{` ⟨rows⟩ `}` centered in math mode and `\LaTeXdiagrams` changes it to use `\begin` ... `\end` syntax. `v2` sets a special internal 'old matrix' flag such that trailing `\\` are ignored and entries starting with `*` are safe.

```
\xyrequire{matrix}\xycatcodes
```

```
\xydef@\diagram#1\enddiagram{\relax\diagram@\diagram@@{#1}}
```

```
\xydef@\diagram@{\relax
 \ifmmode \DN@##1##2{\vcenter{\oldxymatrix@true \xy##1{\xymatrix{##2}}\endxy}}%
 \else\DN@##1##2{{\oldxymatrix@true \xy/u.8ex/##1{\xymatrix{##2}}\endxy}}%
 \fi \next@}
```

```
\xydef@\LaTeXdiagrams{\def\diagram{\ltxdiagram@\diagram@@}%
 \def\diagramnocompile{\ltxdiagram@\literal@}% Declared in Compiling
 \def\diagramcompileto##1{\ltxdiagram@{\NoCompileMatrices\xycompileto{##1}}}}
```

```
\xydef@\ltxdiagram@#1#2\end#3{\relax
 \DN@{#3}\DNii@{diagram}\ifx\next@\nextii@\diagram@{#1}{#2}\end{diagram}%
 \else\xyerr@{Put \string{\string} around \string\begin...\string\end\space
        within diagrams}\fi}
```

```
\xylet@\diagram@@=\literal@
```

```
\xydef@\NoisyDiagrams{}
```

`\NoisyDiagrams` is ignored because the matrix feature always outputs progress messages.

Finally the version 2 `\spreaddiagramrows`, `\spreaddiagramcolumns` spacing commands are emulated using `\xymatrixrowsep` and `\xymatrixcolsep`:

```
\xydef@\spreaddiagramrows#1{\addPLUS@\xymatrixrowsep{#1}}
\xydef@\spreaddiagramcolumns#1{\addPLUS@\xymatrixcolsep{#1}}
```

Arrows

The main arrow commands of version 2 were the `\morphism` and `\definemorphism` commands which now have been replaced by the `\ar` command.

`v2` provides them as well as uses them to define the version 2 commands `\xto`, `\xline`, `\xdashed`, `\xdotted`, `\xdouble`, and all the derived commands `\dto`, `\urto`, ...; the `\arrow` commands of the β-releases of v3 is also provided.

Instead of commands like `\rrto` and `\uldouble` you should use the arrow feature replacements `\ar[rr]` and `\ar@{=}[ul]`.

The predefined turning solid arrows `\lltou`, ..., `\tord` are defined as well; these are now easy to do with ⟨turn⟩s.

```
\xyrequire{arrow}\xycatcodes
\xylet@\arrow=\ar
```

`\morphism` is a reduced version of the `\ar` command.

```
%\xyuncatcodes \catcode35=6
% \def\next#1#2#3{\relax \PATH ~={#1} ~>{\POS?>*{#2}\relax} ~<{\POS;?<*{#3};\relax}}
%\xycatcodes
%\xylet@\morphism=\next
```

```
{\xyuncatcodes
```

```
 \gdef\morPHISM#1#2#3{\PATH~={\oldconnect{#1}}~<{|<*{#3}}~>{|>*{#2}}}}

\xydef@\morphism{\relax\morPHISM}

\xylet@\connect@iii=\connect
\xydef@\oldconnect#1{%
 \let\PATHlabelabove@@=\PATHlabelabove@ %KLUDGE!
 \let\PATHlabelbelow@@=\PATHlabelbelow@
 \let\connect=\preconnect#1\let\connect=\connect@iii}
```

To Do: Unpack this to what `\PATH` does. . .

`\definemorphism` is essentially unchanged except we now use the privacy checking commands.

```
\xydef@\definemorphism#1#2#3#4{%
 \xydefcsname@{x#1}{\morphism{#2}{#3}{#4}}%
 \xydefcsname@{u#1}{\morphism{#2}{#3}{#4}[u]}%
 \xydefcsname@{d#1}{\morphism{#2}{#3}{#4}[d]}%
 \xydefcsname@{l#1}{\morphism{#2}{#3}{#4}[l]}%
 \xydefcsname@{r#1}{\morphism{#2}{#3}{#4}[r]}%
 \xydefcsname@{uu#1}{\morphism{#2}{#3}{#4}[uu]}%
 \xydefcsname@{dd#1}{\morphism{#2}{#3}{#4}[dd]}%
 \xydefcsname@{ll#1}{\morphism{#2}{#3}{#4}[ll]}%
 \xydefcsname@{rr#1}{\morphism{#2}{#3}{#4}[rr]}%
 \xydefcsname@{ur#1}{\morphism{#2}{#3}{#4}[ur]}%
 \xydefcsname@{ul#1}{\morphism{#2}{#3}{#4}[ul]}%
 \xydefcsname@{dr#1}{\morphism{#2}{#3}{#4}[dr]}%
 \xydefcsname@{dl#1}{\morphism{#2}{#3}{#4}[dl]}%
 \xydefcsname@{uul#1}{\morphism{#2}{#3}{#4}[uul]}%
 \xydefcsname@{uur#1}{\morphism{#2}{#3}{#4}[uur]}%
 \xydefcsname@{ull#1}{\morphism{#2}{#3}{#4}[ull]}%
 \xydefcsname@{urr#1}{\morphism{#2}{#3}{#4}[urr]}%
 \xydefcsname@{ddl#1}{\morphism{#2}{#3}{#4}[ddl]}%
 \xydefcsname@{ddr#1}{\morphism{#2}{#3}{#4}[ddr]}%
 \xydefcsname@{dll#1}{\morphism{#2}{#3}{#4}[dll]}%
 \xydefcsname@{drr#1}{\morphism{#2}{#3}{#4}[drr]}%
 \xydefcsname@{uull#1}{\morphism{#2}{#3}{#4}[uull]}%
 \xydefcsname@{uurr#1}{\morphism{#2}{#3}{#4}[uurr]}%
 \xydefcsname@{ddll#1}{\morphism{#2}{#3}{#4}[ddll]}%
 \xydefcsname@{ddrr#1}{\morphism{#2}{#3}{#4}[ddrr]}}

\definemorphism{to}\solid\tip\notip
\definemorphism{line}\solid\notip\notip
\definemorphism{dashed}\dashed\notip\notip
\definemorphism{dotted}\dotted\notip\notip
\definemorphism{double}\Solid\notip\notip
```

To Do: These could be *hugely* optimised by using the `arrow` feature directly. . .

Next the predefined bent morphisms of version 2. These required some hacking to work; I hope this doesn't mean that the ⟨path⟩ semantics has changed to much. . . ☹

```
\xywarnifdefined\xyviitoi
\xywarnifdefined\xyviitoii

\xydef@\XXtoY@#1#2#3#4#5{\save;[#3]\relax
 \xy@@{%
```

```
  \dimen@=\turnradius@ \ifdim#1<#2\relax \dimen@ii=#2\else\dimen@ii=#1\fi
  \advance\dimen@\ifdim\dimen@>\dimen@ii +\dimen@\else +\dimen@ii\fi
  \edef\xyviitoi{\the\dimen@}\leave@}%
 \xto ~{[#3]} `#4[#3]#5/#4\xyviitoi/ `[#3]}
```

```
\xydef@\toXY@#1#2#3#4#5{\xy@@{\idfromc@{@}%
  \dimen@=\turnradius@ \dimen@ii=#1\relax
  \advance\dimen@\ifdim\dimen@>\dimen@ii +\dimen@\else +\dimen@ii\fi
  \edef\xyviitoi{\the\dimen@}%
  \dimen@=\turnradius@ \dimen@ii=#2\relax
  \advance\dimen@\ifdim\dimen@>\dimen@ii +\dimen@\else +\dimen@ii\fi
  \edef\xyviitoii{\the\dimen@}}%
 \morphism\solid{\object #4:(-1,0)\dir{>}}\notip
  ~{`"@" "@"} `#3"@"#5/#3\xyviitoi/#5/#4\p@/ `"@"#5/#4\xyviitoii/}
```

```
\xyuncatcodes \catcode64=11 %
```

```
\xydef@\lltod{\XXtoY@\D@c\D@p{ll}{d}+}
\xydef@\lltou{\XXtoY@\U@c\U@p{ll}{u}+}
\xydef@\rrtod{\XXtoY@\D@c\D@p{rr}{d}+}
\xydef@\rrtou{\XXtoY@\U@c\U@p{rr}{u}+}
\xydef@\ddtol{\XXtoY@\L@c\L@p{dd}{l}+}
\xydef@\ddtor{\XXtoY@\R@c\R@p{dd}{r}+}
\xydef@\uutol{\XXtoY@\L@c\L@p{uu}{l}+}
\xydef@\uutor{\XXtoY@\R@c\R@p{uu}{r}+}
```

```
\xydef@\told{\toXY@\L@c\D@c{l}{d}+}
\xydef@\tolu{\toXY@\L@c\U@c{l}{u}+}
\xydef@\tord{\toXY@\R@c\D@c{r}{d}+}
\xydef@\toru{\toXY@\R@c\U@c{r}{u}+}
\xydef@\todl{\toXY@\D@c\L@c{d}{l}+}
\xydef@\todr{\toXY@\D@c\R@c{d}{r}+}
\xydef@\toul{\toXY@\U@c\L@c{u}{l}+}
\xydef@\tour{\toXY@\U@c\R@c{u}{r}+}
```

```
\xycatcodes
```

Finally the version 2 arrow radius default command that actually never worked but some people might have hoped. . .

```
\xydef@\TurnRadius{\addEQ@\turnradius}
```

B.1.4 Obsolete loading

The v2 User's Manual says that you can load XY-pic with the command `\input xypic` and as a LaTeX 2.09 'style option' `[xypic]`. This is made synonymous with loading this option by the files `xypic.tex` and `xypic.sty` distributed with the `v2` option.

xypic.doc

This file (version 3.3) just loads the `v2` feature.

```
%% $Id: xypic.doc,v 3.3 1996/12/19 03:31:56 krisrose Exp $
%%
%% Xy-pic v2 compatibility loading.
```

```
%% Copyright (c) 1993-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-1996 Kristoffer H. Rose <krisrose@brics.dk>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, write to the Free Software Foundation, Inc.,
%% 675 Mass Ave, Cambridge, MA 02139, USA.
%%
```

Here it is...

```
\input xyv2
\endinput
```

The log reveals that this was originally the main file.

```
% $Log: xypic.doc,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.10  1994/06/15  12:55:07  kris
% Second 3beta release: bug fixes.
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% This file NEW for version 2.7 for backwards compatibility.
% Original xypic.doc renamed to xy.doc.
```

xypic.sty

```
%% $Id: xypic.sty,v 3.3 1996/12/19 03:31:56 krisrose Exp $ -*-tex-*-
%%
%% LaTeX style option to load Xy-pic in version 2 compatibility mode.
%% by Kristoffer H. Rose  <krisrose@brics.dk>
%%
%% This file is in the public domain.  It is distributed in the hope that it
%% will be useful, but WITHOUT ANY WARRANTY; without even the implied
%% warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
%%
\input{xy.sty}
\xyoption{v2}
\catcode`\@=\xystycatcode
\endinput
```

```
% $Log: xypic.sty,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.8  1994/04/08  04:30:00  kris
% Second (bug fix) 3alpha release.
%
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
```

B.1.5 Compiling v2-diagrams

In order to make it possible to use the new compilation features even on documents written with XY-pic v2, the following command was added in v2.12:

`\diagramcompileto{` ⟨name⟩ `}` ... `\enddiagram`

---

which is like the ordinary diagram command except the result is compiled (see note 1.5e). Note that compilation is not quite safe in all cases!

```
\xydef@\diagramcompileto#1#2\enddiagram{\relax
 \diagram@{\NoCompileMatrices\xycompileto{#1}}{#2}}
```

There is also the following command that switches on *automatic compilation* of all diagrams created with the v2 `\diagram` ... `\enddiagram` command:

`\CompileAllDiagrams {` ⟨prefix⟩ `}`
`\NoCompileAllDiagrams`
`\ReCompileAllDiagrams`

---

will apply `\xycompileto{`⟨prefix⟩n`}{`...`}` to each diagram with n a sequence number starting from 1. Use `\CompileMatrices` and `\CompilePrefix` instead!

If for some reason a diagram does not work when compiled then replace the `\diagram` command with `\diagramnocompile` (or in case you are using the LaTeX form, `\begin{diagramnocompile}`).

```
\xydef@\diagramnocompile#1\enddiagram{\relax\diagram@\literal@{#1}}
```

```
\xydef@\CompileAllDiagrams#1{\CompilePrefix{#1}\CompileMatrices}
\xydef@\NoCompileAllDiagrams{\NoCompileMatrices}
\xydef@\ReCompileAllDiagrams{\CompileMatrices}
```

End & log

```
\xyendinput
```

```
% $Log: xyv2.doc,v $
% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.
%
% Revision 3.3  1996/12/19 14:43:55  krisrose
% Maintenance release.
%
% Revision 3.2  1995/09/19  18:22:27  kris
% Bug fix release.
%
% Revision 3.1  1995/09/05  20:31:32  kris
% Releasing!
%
% Revision 3.0  1995/07/07  20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.14  1995/07/05  22:10:12  kris
% Buglets...
%
% Revision 2.13  1995/07/04  15:11:17  kris
% Ready to release v3?
%
% Revision 2.12  1994/10/25  11:34:25  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Revision 2.11  1994/07/05  10:37:32  kris
% Third 3beta release [bug fixes].
% Experimental graph feature included (for ECCT-94 presentation).
%
% Revision 2.10  1994/06/15  12:55:07  kris
% Second 3beta release: bug fixes.
%
% Revision 2.9  1994/06/09  14:59:19  kris
% Release 3beta.
%
% Revision 2.8  1994/04/08  04:30:00  kris
% Second (bug fix) 3alpha release.
%
```

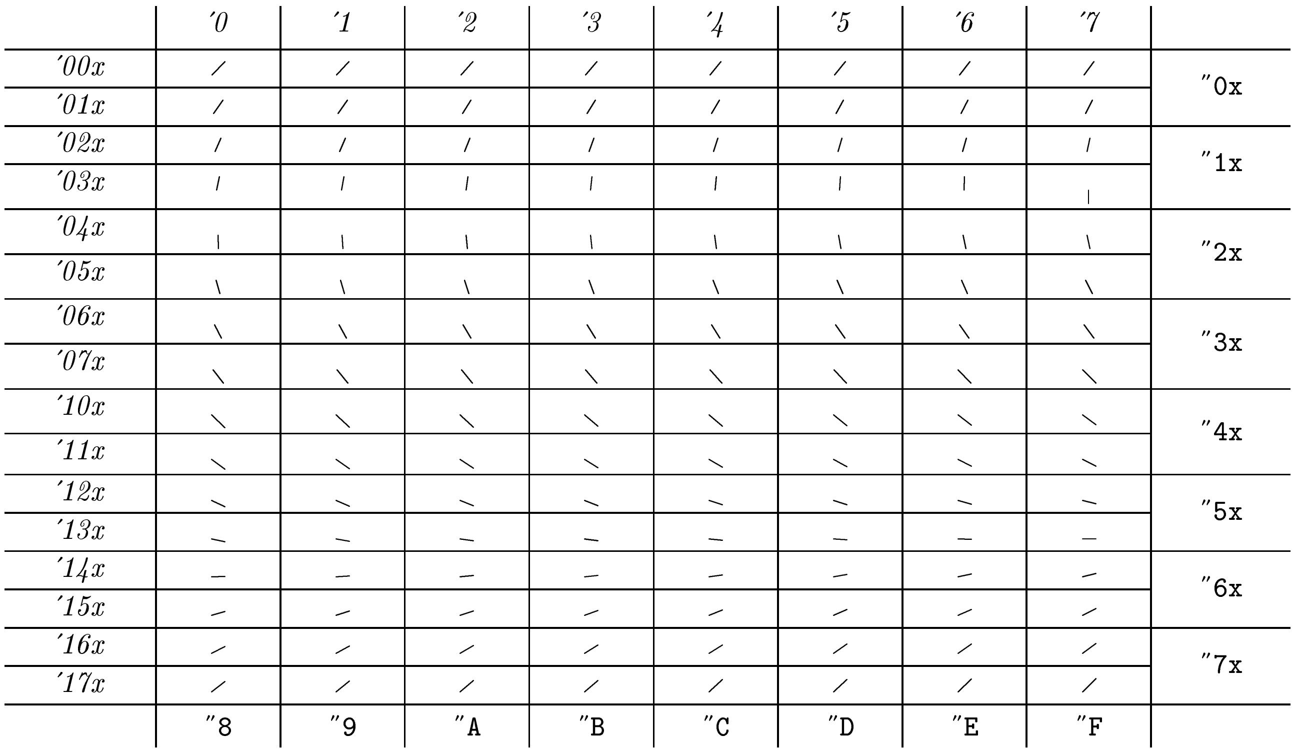


Figure B.1: Font table for `xyline10`.

```
% Revision 2.7  1994/03/08  02:06:01  kris
% Release 3alpha.
%
% Revision 2.6.9.1  1994/03/07  04:22:46  kris
% Last internal 3alpha and pre-2.7 release.
%
% NEW file to go in version 2.7 (extracted from xy.doc 2.6.1.1).
```

B.2 Obsolete fonts

These fonts are obsolete since v2.7 but remain part of the distribution to make it possible to print DVI files created with version 2.6 and earlier versions.

B.2.1 xyline10

```
% $Id: xyline10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp $ -*-tex-*-
%
% XYLINE10: line segments for XY mode at 10 point.
% Copyright (c) 1991,1992  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the XY-pic macro package.
%
% The XY-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
```

```
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The XY-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation, Inc.,
% 675 Mass Ave, Cambridge, MA 02139, USA.
%
% CONTENTS: Line segments going either segl# down or to the right.  Codes
% range from 0 to 127, distributed evenly in each of the
% intervals between the ‘purely’ horizontal/
% vertical/diagonal characters with code 31,        ..../
% 63, 95, and 127 (code -1 would be  pure as        :  /:
% well if it existed):                              : / :
%                                                   :/  :
%       ..../  ..../   |   o   o....  o....  o----  o...:
%       :  /:  :  /:   |   |   :\  :  :\  :
%       : / :  : / :   |   |   : \ :  : \ :
%       :/  :  :/  :   |   |   :  \:  :  \:
%       o...:  o...:   o   |   :...\  :...\
%
% Code: [-1]     0    30  31     63     64     95     127
%
font_identifier "XYLINE"; font_size 10pt#;
font_coding_scheme:="XY line segments";
mode_setup;

% METANESS...
%
segl# = 1/2 designsize; define_pixels(segl);    % line segment length
rulew# = .4pt#; define_whole_blacker_pixels(rulew); % line thickness
%
% drawsegment draws a line from (0,0) to the argument point...
def drawsegment expr endpoint =
 z0 = (0,0); z1 = endpoint;
 pickup pencircle scaled rulew rotated (angle (z1-z0) - 90);
 draw z0--z1; penlabels(0,1) enddef;

% TESTING...we redefine openit because the characters extend far to the
% left of the bounding box!
%
def openit = openwindow currentwindow
 from origin to (screen_cols,screen_rows) at (-200,300) enddef;

% FONT.
%
% Font dimension 8 is the rule thickness (cf. The TeXbook, app.G)
```

```
fontdimen 8: rulew#;
%
% The characters follow...
%
for cc = 0 step 1 until 30:
 beginchar(cc,(31-cc)/32*segl#,segl#,0); drawsegment (w,h); endchar;
endfor;
%
for cc = 31 step 1 until 63:
 beginchar(cc,(cc-31)/32*segl#,0,segl#); drawsegment (w,-d); endchar;
endfor;
%
for cc = 64 step 1 until 94:
 beginchar(cc,segl#,0,(95-cc)/32*segl#); drawsegment (w,-d); endchar;
endfor;
%
for cc = 95 step 1 until 126:
 beginchar(cc,segl#,(cc-95)/32*segl#,0); drawsegment (w,h); endchar;
endfor;
%
beginchar(127,segl#,segl#,0); drawsegment (w,h); endchar;

bye.
%
% $Log: xyline10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.1  1992/01/02  14:54:07  kris
% Release version.
%
% Revision 1.7  1991/12/17  04:51:16  kris
% Version distributed with 'final draft' on Usenet.
%
% Revision 1.6  1991/11/27  06:54:21  kris
% \beta-test on DIKU.
%
% Revision 1.5  1991/10/21  23:19:08  kris
% Version described in DIKU student report 91-7-10.
%
% Revision 1.4  1991/07/28  22:16:18  kris
% Inverted char0..30 to avoid negative widths.
%
% Revision 1.3  1991/07/26  01:22:28  kris
% Set bounding box completely!
%
% Revision 1.2  91/06/09  21:59:49  kris
```

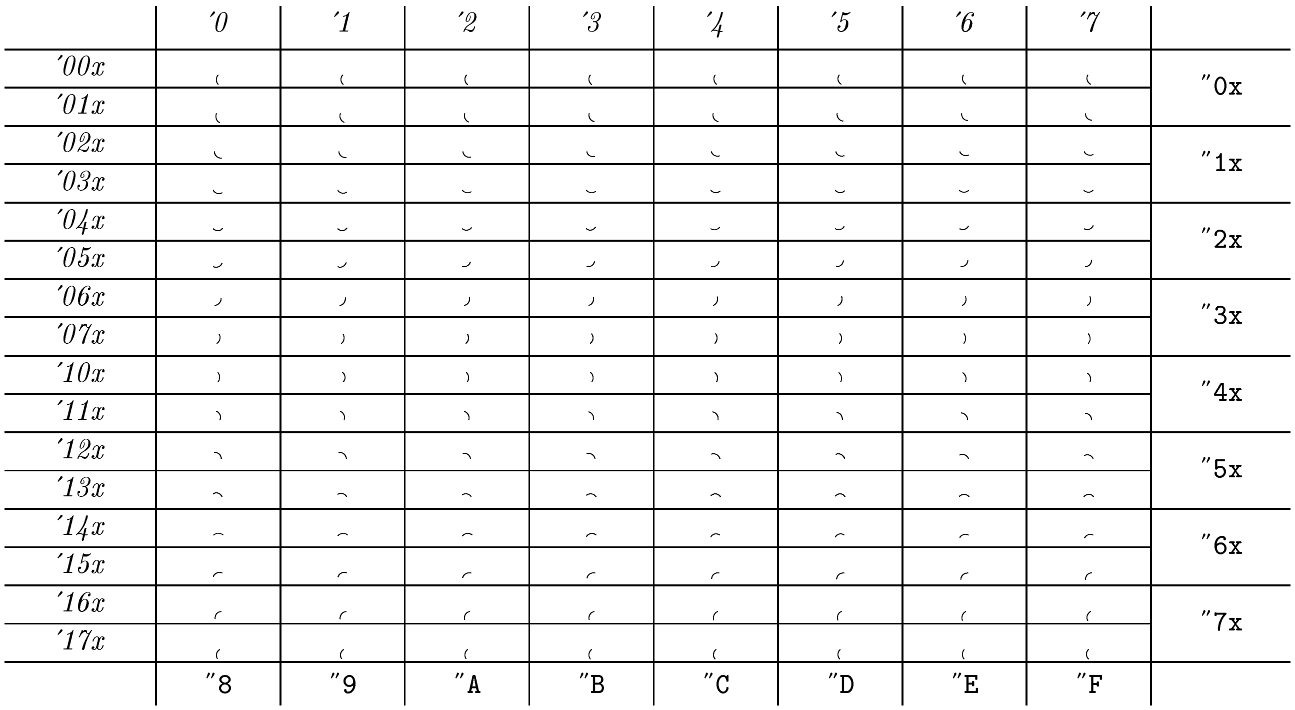


Figure B.2: Font table for `xyqc10`.

```
% separated METANESS and CHARACTERS
%
% Revision 1.1  91/06/06  22:52:31  kris
% furst succesful attempt
```

B.2.2 xyqc10

```
% $Id: xyqc10.mf,v 3.0 1995/07/07 20:14:21 kris Exp $ -*-tex-*-
%
% XYQC10: quarter circles for XY mode at 10 point.
% Copyright (c) 1992  Kristoffer H. Rose  <kris@diku.dk>
%
% This file is part of the XY-pic macro package.
%
% The XY-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The XY-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
```

```
% with this macro package; if not, write to the Free Software Foundation, Inc.,
% 675 Mass Ave, Cambridge, MA 02139, USA.
%
% CONTENTS: Quarter circles in all directions:
%
%                                   -.
%                            |   )   |  /\
%           o    o    o    o-'  o    o    o  .-o    o
%          (     |_    \/                    |     (
%
% Code:   [-1]   15   31   47   65   79   95  111  127
%
% Each character has a cosinoid bounding box :-) with wd = cos I = sin F,
% where I,F are the angles from direction --> to the initial,final direction.
%
font_identifier "XYQC"; font_size 10pt#;
font_coding_scheme:="XY quarter circles";
mode_setup;

% METANESS...
%
qcd# = 1/2 designsize;  % quarter circle diameter
rulew# = .4pt#;         % quarter circle line width
define_pixels(qcd,rulew);
%
% beginqc draws the quarter circle starting in the direction towards endpoint
% and turning left...
def beginqc(expr cc,ex,ey) =
 alpha := angle (ex,ey);
 cw := sqrt(1/2) * abs(sind(alpha + 135));
 beginchar(cc,cw*qcd#,0,0);
 pickup pencircle scaled rulew rotated (alpha - 90);
 if (cc<64): draw quartercircle shifted (-.5,0) rotated (alpha - 90)
        scaled qcd
 else: draw quartercircle shifted (-.5,0) rotated (alpha - 90) shifted (cw,0)
        scaled qcd
 fi
enddef;

% TESTING...we redefine openit because the characters extend far to the
% left of the bounding box!
%
def openit = openwindow currentwindow
 from origin to (screen_cols,screen_rows) at (-200,200) enddef;

% FONT.
%
% Font dimension 8 is the rule thickness (cf. The TeXbook, app.G).
fontdimen 8: rulew#;
%
% Here are the characters...
```

```
%
for cc = 0 step 1 until 31:
 beginqc(cc,cc-15,-16); endchar;
endfor;
%
for cc = 32 step 1 until 63:
 beginqc(cc,16,cc-47); endchar;
endfor;
%
for cc = 64 step 1 until 95:
 beginqc(cc,79-cc,16); endchar;
endfor;
%
for cc = 96 step 1 until 127:
 beginqc(cc,-16,111-cc); endchar;
endfor;

bye.
%
% $Log: xyqc10.mf,v $
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.6  1992/06/24  01:23:34  kris
% Implemented and used for hooks.
```

B.2.3 xymisc10

```
% $Id: xymisc10.mf,v 3.3 1996/12/19 03:31:56 krisrose Exp $ -*-tex-*-
%
% XYMISC10: miscellaneous characters for XY mode at 10 point.
% Copyright (c) 1991,1992  Kristoffer H. Rose  <krisrose@brics.dk>
%
% This file is part of the XY-pic macro package.
%
% The XY-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The XY-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, write to the Free Software Foundation, Inc.,
% 675 Mass Ave, Cambridge, MA 02139, USA.
%
% CONTENTS:
```

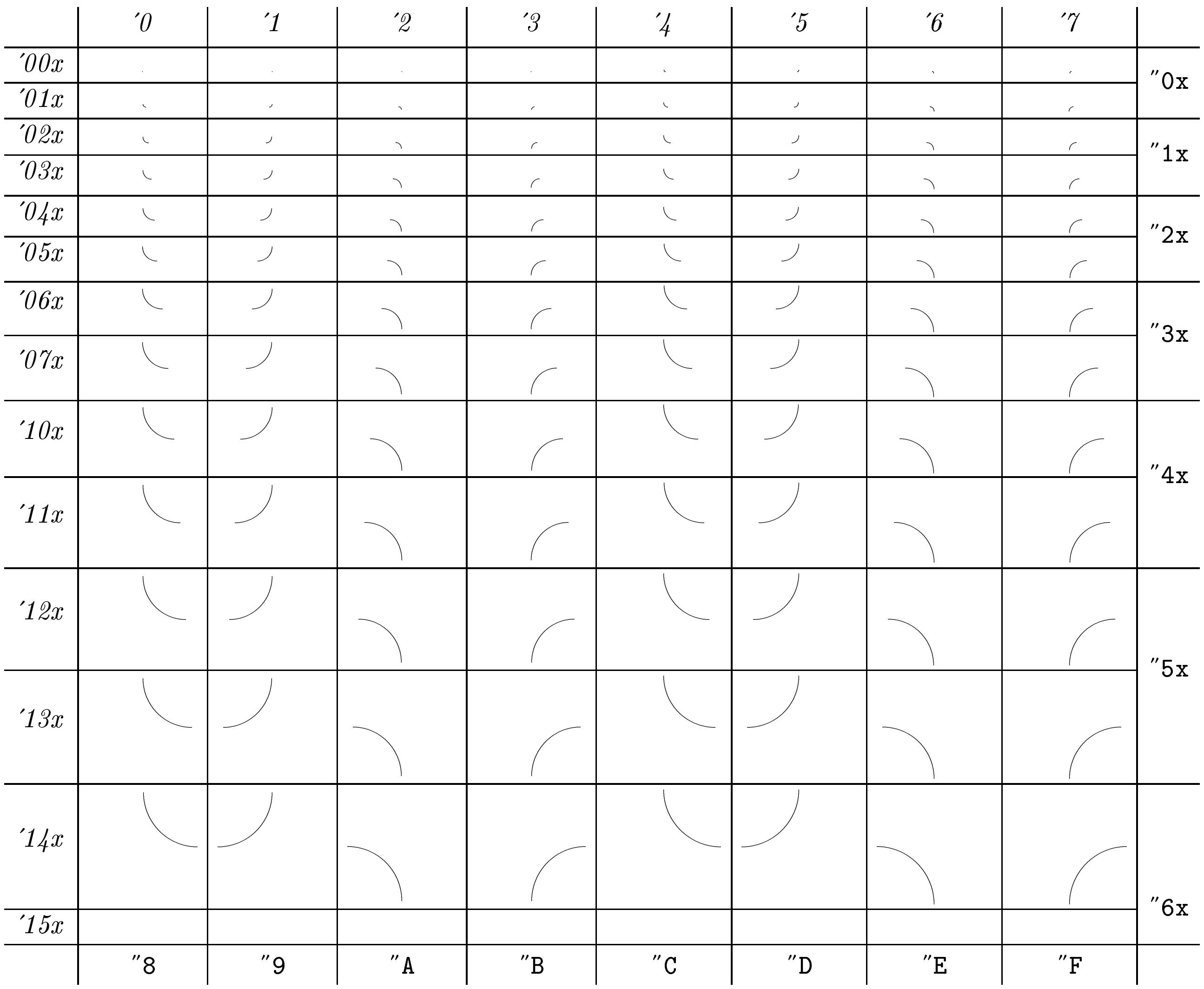


Figure B.3: Font table for `xymisc10 scaled 578`.

```
%   Metaness.
%   Testing...
%   Font.
%     Dot fitting the rule width.
%     Quarter circles with radii from 1 to 10 and 12 to 30 pt; all have
%       height = depth = .5width = radius and are centered such that they
%       enter and leave the box in the middle of the sides.
%
font_identifier "XYMISC"; font_size 10pt#;
font_coding_scheme:="XY miscellaneous";
mode_setup;

% METANESS...
%
rulew# = .4pt#; define_whole_blacker_pixels(rulew); % line thickness
%
def pickuppen = pickup pencircle scaled rulew enddef;

% TESTING...we redefine openit because the characters extend far to the
% left of the bounding box!
%
def openit = %%let echar = endchar; def endchar = echar; stop ". " enddef;
 openwindow currentwindow
  from origin to (1000,800) at (-200,300)
 enddef;

% FONT.
%
% Font dimension 8 is the rule thickness (cf. The TeXbook, app.G)
fontdimen 8: rulew#;

% DOT FOR DOTTED LINES AND DUMMY CIRCLE...
%
beginchar(0,0,0,0); pickuppen; drawdot (0,0); endchar;
beginchar(1,0,0,0); pickuppen; drawdot (0,0); endchar;
beginchar(2,0,0,0); pickuppen; drawdot (0,0); endchar;
beginchar(3,0,0,0); pickuppen; drawdot (0,0); endchar;

% QUARTER CIRCLES...  The four circles with radius N pt, 0 < N <= 10, are
% located at N*4 + QUADRANT, where 1 <= QUADRANT <= 4.  All have width as
% their radius with reference point at the level of the horizontal end of the
% arc and height, depth as the extent of the arc.
%
for radius = 1 step 1 until 10:
 beginchar(radius*4,2radius*pt#,radius*pt#,radius*pt#);
  pickuppen;
  draw quartercircle rotated 180 shifted (1,.5) scaled (radius*2pt);
 endchar;
 beginchar(radius*4+1,2radius*pt#,radius*pt#,radius*pt#);
  pickuppen;
  draw quartercircle rotated -90 shifted (0,.5) scaled (radius*2pt);
 endchar;
 beginchar(radius*4+2,2radius*pt#,radius*pt#,radius*pt#);
```

```
  pickuppen;
  draw quartercircle shifted (0,-.5) scaled (radius*2pt);
 endchar;
 beginchar(radius*4+3,2radius*pt#,radius*pt#,radius*pt#);
  pickuppen;
  draw quartercircle rotated 90 shifted (1,-.5) scaled (radius*2pt);
 endchar;
endfor;
%
% For 12 <= N <= 40 we step in 2pt increments... so for 10 <= N <= 40 the
% quarter circle with radius N is located at (N div 2 + 20 + QUADRANT).
%
for radius = 12 step 2 until 40:
 beginchar(radius*2+20,2radius*pt#,radius*pt#,radius*pt#);
  pickuppen;
  draw quartercircle rotated 180 shifted (1,.5) scaled (radius*2pt);
 endchar;
 beginchar(radius*2+21,2radius*pt#,radius*pt#,radius*pt#);
  pickuppen;
  draw quartercircle rotated -90 shifted (0,.5) scaled (radius*2pt);
 endchar;
 beginchar(radius*2+22,2radius*pt#,radius*pt#,radius*pt#);
  pickuppen;
  draw quartercircle shifted (0,-.5) scaled (radius*2pt);
 endchar;
 beginchar(radius*2+23,2radius*pt#,radius*pt#,radius*pt#);
  pickuppen;
  draw quartercircle rotated 90 shifted (1,-.5) scaled (radius*2pt);
 endchar;
endfor;

bye.
%
% $Log: xymisc10.mf,v $
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User's Guide!
%
% Revision 2.6  1992/06/24  01:23:34  kris
% Cleaned up.
%
% Revision 2.1  1992/01/02  14:54:07  kris
% Release version.
%
% Revision 1.6  1991/11/27  06:54:21  kris
% \beta-test on DIKU.
%
% Revision 1.5  1991/10/21  23:19:08  kris
```

```
% Version described in DIKU student report 91-7-10.
%
% Revision 1.4  1991/08/22  01:07:46  kris
% Now has quarter circles to radius 40pt as required for xy.doc[1.28].
%
% Revision 1.3  1991/07/21  21:04:35  kris
% Tested with xy.doc[1.19].
%
% Revision 1.2  1991/07/19  14:52:26  kris
% Changed quarter circles to fit xy.doc[1.19?].
%
% Revision 1.1  1991/06/24  20:57:37  kris
% Works with xy.doc [1.6] :-)
```

Appendix C

GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and conditions for copying, distribution and modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

    The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

    If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the

Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) 19yy  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix D

Distribution support files

This appendix presents the sources of various support files that are part of the Xy-pic source distribution (except the file `COPYING` that contains the text reproduced in appendix A): the files with ‘hype’ for the package, the installation instructions, and the generation instructions (to make Xy-pic from the sources).

D.1 Hype

These files explain what Xy-pic is as required by the CTAN ‘LaTeX 2_ε bundle’ standard.

D.1.1 CATALOG

```
Name: xypic
Version: 3.6 <1998/03/06>

Description: Xy-pic is a package for typesetting graphs and diagrams.
 It is structured as several modules, each defining a custom notation for
 a particular kind of graphical object or structure. Example objects are
 arrows, curves, frames, and colouring/rotation on drivers that support
 it; these can be organised in matrix, directed graph, path, polygon, knot,
 and 2-cell structure. Xy-pic works with most formats, including LaTeX,
 AMS-LaTeX, AMS-TeX, and plain TeX, and has been used to typeset
 complicated diagrams from many application areas including category
 theory, automata theory, algebra, neural networks, and database theory.

Keywords: diagram, picture, graph, arrow, curve, commutative diagram.

Author: Kristoffer H. Rose
Problems-To: Kristoffer.Rose@ens-lyon.fr
Address: LIP/ENS-Lyon; 46, Allee d'Italie; 69364 Lyon 7; FRANCE

<URL: http://www.ens-lyon.fr/~krisrose/>
```

D.1.2 README

```
-*-text-*- $Id: README,v 3.6 1998/03/06 01:28:05 krisrose Exp krisrose $

        =====================================================
        THE Xy-pic PACKAGE FOR TYPESETTING GRAPHS AND DIAGRAMS
        =====================================================
```

```
Xy-pic is a package for typesetting graphs and diagrams with TeX.

It works with most formats, including LaTeX (both 2.09 and 2e),
AMS-LaTeX, AMS-TeX, and plain TeX.

It has been used to typeset complicated diagrams from many application
areas including category theory, automata theory, algebra, neural
networks, and database theory.

                                 - o -

See below in this file for an overview of the structure and features!
See CATALOG for up to date version information!
See TRAILER for author and accessibility information!
See COPYING for the conditions for redistribution and modification!
See MANIFEST for a complete file list!
See VERSIONS for the exact list of the versions of all source files!
See the various INSTALL files for how to install Xy-pic on your system!
See the W3 <URL: http://www.ens-lyon.fr/~krisrose/Xy-pic.html> for news!

                                 - o -

Xy-pic is structured as a `kernel' and several orthogonal modules called
`options', each defining a custom notation for a particular kind of
graphical object or structure.

These (combinable) structures are available:

* A graph combinator mode where diagrams are specified the way they are
  composed as graphs.

* A matrix-like mode where the dimensions of the drawing are computed by
  aligning diagram entries in rows and columns (this is the `diagram'
  mode Xy-pic version 2 users are used to).

* A polygon mode where diagrams shaped as regular polygons are entered
  in a simple way.

* A mode for typesetting beautiful knots and links.

* A general object-oriented `turtle graphic' drawing language for
  specifying graphs with objects and connections between them in a
  manner independent of orientation.

The following objects may be used:

+ Positions can be given in variety of formats (extendable) including
  user defined coordinates (x,y) and relative to previous positions,
  objects, object edges, and points on connections.

+ Objects may be circular, elliptic, or rectangular (more shapes can be
  added) and adjusted in several ways.

+ Large library of objects with mnemonic names.

+ Objects that orient themself along a connection when placed relative
  to it, e.g., `\dir{|-}' is like \vdash but thus oriented; new such
  objects can be defined in a convenient way.

+ TeX `boxes', i.e., text and mathematical formulae.

+ Includes circle segments and optionally arbitrary elliptical,
```

```
  quadratic, and cubic arcs.

+ Connections are aligned between the reference points of objects but
  start and end on the edges.

+ Any object can be used to build a connection (using `diagonal
  filling'); library objects provide common line types.

+ Flexible notation for drawing arrows and general paths with tail,
  stem, and head built from any object(s).  Special support for arrows
  that cross each other, arrows that `go by' other entries, paired
  arrows (including support for 2-cells), curved arrows, and arrows with
  bends.

+ Library of frames and braces.

+ Special notation for rotation, scaling, colour, and line thickness.
  The correct typesetting of these features requires a backend that
  supports it (i.e., PostScript) but even when this is not available
  Xy-pic tries to approximate what is requested such that at least the
  picture size is stable (and thus page breaks).

+ Output can use PostScript for drawing (several \special formats are
  supported: dvips, Textures, and OzTeX -- more are being added
  continuously: contact Ross Moore <ross@mpce.mq.edu.au> for the current
  list).  Notation for inclusion of literal PostScript is available.

* Other graphics can be imported and then `decorated' using Xy-pic.

All object forms may be used with all structures.

                                - o -

Finally, enjoy Xy-pic!

Sincerely,
               Kristoffer H. Rose      <krisrose@ens-lyon.fr>
               Ross Moore              <ross@mpce.mq.edu.au>
```

D.2 Installation instructions

These files explain how Xy-pic is downloaded and installed, *i.e.*, where the various files of the distribution should reside.

D.2.1 TRAILER

```
-*-text-*- $Id: TRAILER,v 3.6 1998/03/06 01:28:05 krisrose Exp krisrose $

======================================================================
     ANNOUNCING the Xy-pic version 3.7 DIAGRAM TYPESETTING PACKAGE
======================================================================

This is to announce a release of my diagram typesetting package Xy-pic.

With version 3.7 we explicitly grant permission to use the unmodified
Xy-pic fonts and specials in distributed documents.

----------------------------------------------------------------------
                               GENERAL
```

```
----------------------------------------------------------------------

Xy-pic is a package for typesetting a variety of graphs and diagrams
with TeX.  Xy-pic works with most formats (including LaTeX, AMS-LaTeX,
AMS-TeX, and plain TeX), in particular Xy-pic is provided as a LaTeX2e
‘supported package’ (following the ‘CTAN LaTeX2e bundle’ standard).

Further specifics of the package are in the distribution README file.

----------------------------------------------------------------------
                                 NEWS
----------------------------------------------------------------------

The primary purpose of release 3.7 is the granting of special permission
to copy the Xy-pic fonts and graphic specials inside documents, for
example in DVI, PostScript, and PDF files.  This has no practical
significance except making current (reasonable) practice explicitly
legal.

Release 3.7 furthermore fixes a few bugs in release 3.6 and adds a few
features: a "smart turning path" feature and an extension with macro
expansion in kernel expressions and polygon shapes by George C. Necula,
and the beginnings of an emulation of LaTeX picture mode proposed by
Paul Burchard and Bill Richter.

Special thanks to George Necula for the bug fixes and impressive
extensions, to Bill Richter for extending Paul Burchard’s pb-diagram to
use Xy-pic as a backend and for contributing the ‘picture’ feature, to
Berthold Horn of Y&Y Inc. for releasing the copyright on their
PostScript fonts, and to Pieter Donche and Marcelo Cunha dos Santos for
the bug reports.

----------------------------------------------------------------------
                              AVAILABILITY
----------------------------------------------------------------------

Xy-pic can be retrieved through the World Wide Web Xy-pic ‘home page’:

  URL:                 http://www.ens-lyon.fr/~krisrose/Xy-pic.html

as well as by anonymous ftp from

  CTAN:                macros/generic/diagrams/xypic

Check the README file in each location for specifics, in particular
check that you have reached a version 3.7 copy (some archives take a
while to mirror the latest files)!

----------------------------------------------------------------------
                           HISTORY & CREDITS
----------------------------------------------------------------------

The first public release (version 1.40) of Xy-pic was created by
Kristoffer H. Rose, DIKU, U of Copenhagen (now ENS-Lyon, France) and
distributed via Usenet on December 19, 1991.  This quickly became
version 2 of which version 2.6 was the stable for sevarl years.

The thorough rewrite that became version 3 is a continued
collaboration with Ross Moore, Macquarie U, Sydney, initiated through
```

```
a visit to Macquarie (Jan-May 1994 supported by the Australian
Research Council, Macquarie University, and using donated DEC
equipment).  However, full backwards compatibility is maintained
(except for the unavoidable but fully documented obscure cases).

Xy-pic is Copyright (c) 1991-1998 by Kristoffer H. Rose and 1994-1998
by Ross Moore under GNU COPYLEFT (GPL) which means that you can use
the package for any purpose but if you provide the macros or any code
derived from them to a third party then you are obliged to include the
entire Xy-pic package (full details in the file COPYING).  The
PostScript fonts are additionally Copyright (c) 1997 Y&Y Inc.  The
FONTCOPYING file details the special permissions to distribute fonts
without the full requirements of the GPL.

----------------------------------------------------------------------

         This is the end of the announcement.  Enjoy Xy-pic!

----------------------------------------------------------------------
```

D.2.2 INSTALL

```
-*-text-*- $Id: INSTALL,v 3.6 1998/03/06 01:28:05 krisrose Exp $

             ================================================
             INSTALLING THE Xy-pic DIAGRAM TYPESETTING MACROS
             ================================================

This file explains how you install the files in the `run' distribution of
Xy-pic.  See `README' for general information.

Contents:

-1- Copyleft
-2- Overview of files and where they should be installed
-3- Information for specific systems
-4- Updating from version 2.6 (and earlier)
-5- Bugs & comments

--------------------------------------------------------------------------
-1- COPYLEFT
--------------------------------------------------------------------------

Note that the entire Xy-pic package is distributed with copyleft:

  Copyright (c) 1991-1998   Kristoffer H. Rose  <krisrose@ens-lyon.fr>
  Copyright (c) 1994-1998   Ross Moore          <ross@mpce.mq.edu.au>

  The Xy-pic package is free software; you can redistribute it and/or
  modify it under the terms of the GNU General Public License as
  published by the Free Software Foundation; either version 2 of the
  License, or (at your option) any later version.

  The Xy-pic package is distributed in the hope that it will be
  useful, but WITHOUT ANY WARRANTY; without even the implied warranty
  of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  General Public License for more details.
```

```
  You should have received a copy of the GNU General Public License
  along with this package; if not, write to the Free Software
  Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

What this means is that if you install Xy-pic then you should (1) make
sure that the file `COPYING' is accessible to the users, and (2) be
prepared to provide Xy-pic (including sources) to any of your users
that request it (you may charge a fee for this if you are so
inclined).

Note: Some option files may be copyrighted by other people but the
conditions remain the same for any file distributed as part of Xy-pic.

----------------------------------------------------------------------------
-2- OVERVIEW OF FILES AND WHERE THEY SHOULD BE INSTALLED
----------------------------------------------------------------------------

In this section we explain in 11 steps where the different groups of
Xy-pic files in the `run' and `doc' distribution should be installed.
For each file category we indicate the location recommended by TDS,
the TeX Directory Standard (as formed by a TUG working group of that
name).

IMPORTANT
---------
If you have users currently using version 2.6 then make sure you read
section 4!  Please also check the next section for any system specific
considerations.

SPACE REQUIREMENTS
------------------
Installing the Xy-pic package requires almost 1MB with the macros
macros taking up most of the space.  Installing everything, including
sources, documentation in both DVI, DVIPS, and PS, form, as well as
bitmap fonts for several printers and in PostScript Type1 form,
requires more than 12MB.

Step 1. Eliminating Obsolete Files
----------------------------------
The new PostScript Type1 fonts contributed by Y&Y Inc. require that
you REMOVE ALL COPIES of the following files BEFORE installing:

Old font map files should be removed:

  xy.map
  xypic.map

TDS directory: texmf/dvips/xypic

Also be aware that some installations, for example the TeX Live 2
CD-ROM, insert the contents of map files in `psfonts.map' (with TDS
directory texmf/dvips/base); then you must EDIT it to remove all lines
that start with `xy'.

Remove old PostScript Type1 font files:

  xyatip10.pfa
```

```
  xybsql10.pfa
  xybtip10.pfa
  xycirc10.pfa
  xycmat10.pfa
  xycmbt10.pfa
  xydash10.pfa
  xyeuat10.pfa
  xyeubt10.pfa

  xyatip10.pfb
  xybsql10.pfb
  xybtip10.pfb
  xycirc10.pfb
  xycmat10.pfb
  xycmbt10.pfb
  xydash10.pfb
  xyeuat10.pfb
  xyeubt10.pfb

TDS directory: texmf/fonts/type1/public/xypic

Remove old `xy.sty' and `xypic.sty' files that you might have stored
earlier in the TDS directory `texmf/tex/latex/misc'.


[FILES IN xy*run.tar.gz:]


Step 2. Availability & License
------------------------------
These eight files contain general information and should be installed
in a publicly readable place:

  README        generic information on Xy-pic
  CATALOG       version information and summary
  TRAILER       announcement and availability
  INSTALL       how to install (this file)
  MANIFEST      list of all files
  VERSIONS      the RCS version ids of all source files
  COPYING       GNU General Public License

TDS directory: texmf/doc/generic/xypic


Step 3. Macros
--------------
Files `texinputs/*' are TeX and LaTeX macro files that must be
installed in a place which TeX and LaTeX will search (e.g., with un*x
TeX, a directory users include in the TEXINPUTS environment variable).

The files `texinputs/*.ini'

TDS directory: texmf/tex/generic/xypic


Step 4. TeX Font Metric
-----------------------
Files `texfonts/*.tfm' should be installed in a directory that TeX
will search for fonts (e.g., with un*x TeX, a directory users include
```

```
in the TEXFONTS environment variable).

TDS directory: texmf/fonts/tfm/public/xypic


Step 5. METAFONT Sources
------------------------
Files `mfinputs/*.mf' should be installed in some directory where the
METAFONT program will search for them (e.g., with un*x TeX, a
directory included in the MFINPUTS environment variable).

NOTE: The METAFONT source files are only useful on installations where
the printer driver and previewer can utilise them to generate the
required bitmap fonts `on-line'.  If this is possible on your
installation then please use it: it saves substantial amounts of space
and means that you will automatically use fonts optimised for each
printer (provided evrything is correctly set up, of course).

TDS directory: texmf/fonts/source/public/xypic


Step 6. Bitmap Fonts [Optional]
-------------------------------
Files matching the patterns `pkfonts/cx<N>/*.<N>pk' and
`pkfonts/ljfour<N>/*.<N>pk', where <N> is some number, contain bitmap
font files and should be installed where DVI previewers and printer
drivers will find them (e.g., with un*x, the `xdvi' program should see
them through the XDVIFONTS environment variable).  (`cx' is a 300 dpi
(dots per inch) mode and `ljfour' a 600 dpi mode.)

NOTE: This step is optional because the bitmaps are only needed (and
only recommended) for installations that can't use the METAFONT
sources directly but do have a 300 or 600 dpi laserprinter driver that
accepts PK bitmap font files.

TDS directory: texmf/fonts/pk/cx/public/xypic/dpi<N>
          and: texmf/fonts/pk/ljfour/public/xypic<N>


Step 7. PostScript Type1 Fonts
------------------------------
Files `type1/*.pfb' contain versions of the Xy-pic fonts in special
PostScript form that may be useful on some installations.  They should
be installed where the applications using them look for such fonts (if
used with DVIPS then the `ps/xypic.map' file from the next group
should also be installed).

NOTE: Some installations, notably teTeX 0.4, will PREFER using these
fonts over the METAFONT-generated equivalents (IMHO this is a bug in
the MakeTeXPK script).  On those systems we recommend that you ALSO
install the PK fonts to circumvent the problem.

IMPORTANT: Make sure that you do not have any old Xy-pic PostScript
`xy*.pfa' fonts floating around in your system - search for and remove
all files named `xy*.pfa' when you have installed the new fonts.  One
common way old (and wrong) fonts is manifest is when some arrowheads
appear skewed or too thin.
```

```
TDS directory: texmf/fonts/type1/public/xypic

Step 8. Adobe Font Metric Files
-------------------------------
Files `afm/*.afm' are special files for the Adobe Type Manager and
some other software.  Install them if you need them.

TDS directory (usually): texmf/fonts/afm/public/xypic

Step 9. PostScript Font Metric Files
------------------------------------
Files `pfm/*.pfm' are special files for the Windows version of the
Adobe Type Manager and some other software.  Install them if you need
them.

TDS directory (usually): texmf/fonts/pfm/public/xypic

Step 10. PostScript Header Files
--------------------------------
Files `ps/*' contain PostScript header files.  If you plan to use the
PostScript extension option then you should copy these files to a
place where your PostScript DVI driver will find them.

NOTE: For older versions of DVIPS (at least for 5.58f) you should
replace the occurrences of `<<' with `<' in `xypic.map'.

TDS directory (usually): texmf/dvips/xypic

[FILES IN xy*doc.tar.gz:]

Step 11. Manuals
----------------
The typeset User's Guide is distributed as a DVI file which you should
be able to print (or preview) once you have installed the fonts (as
source or bitmaps as applicable for your system):

  doc/xyguide.dvi      User's Guide, typeset using the kernel Xy-pic fonts
  doc/xyguide.dvips    --- using DVIPS specific `specials'
  doc/xyguide.ps       --- in PostScript (with 600dpi bitmap fonts)

Similarly, the reference manual for the kernel and the distributed
options is included in the same format:

  doc/xyrefer.dvi      Reference Manual, using Xy-pic fonts
  doc/xyrefer.dvips    --- using DVIPS specific `specials'
  doc/xyrefer.ps       --- in PostScript (with 600dpi bitmap fonts)

Notice that only the .dvi documents are portable among all TeX
installations.

TDS directory: texmf/doc/generic/xypic

That's all.  If you want to install the Xy-pic sources (to experiment
with development, for example), you need to get the separate
distribution `xy*src.tar.gz'.

---------------------------------------------------------------------------
```

```
-3- INFORMATION FOR SPECIFIC SYSTEMS
------------------------------------------------------------------------


UN*X
----
The above installation procedure (using the TDS recommended places) is
encoded in the script `support/install-tds': the command

  ./support/install-tds <texmf>

where <texmf> is the full path name of the `texmf' directory (the root
of the TDS tree), will install all the Xy-pic runtime files (or more
precisely: it will perform steps 2-5 and 7-10 above).

NOTE: You must MANUALLY remove obsolete files (do step 1 above)!

If you get the source distribution as described in `TRAILER' and
change the configuration section of the provided `src/Makefile' to
reflect your local file system conventions as described in
`src/MAKE', then the command

  make TEXMF=<texmf> install

will also do this (but with the generated files, of course).


VMS
---
(Mon, 20 Mar 1995; thanks to Austin Benson <austin@shapel.ug.eds.com>
and Ralf Gaertner <gaertner@cthulhu.rhein-main.de>):

(1) VMS tfm and pk files are in a funny file format, which means they
    EITHER should be generated from the mf source on a VMS machine, OR
    be converted by a conversion program such as BILF (part of the
    DECUS TeX distribution and the ZOO distribution although the
    latter is reported to only work on VAXen).

    FIX: Install .mf sources and pregenerate .tfm and .pk files
    manually OR convert with BILF or a similar program.

(2) At least some (OLD) versions of TeX for VMS had a `buglet' to do
    with searching for input files: if you go \input <filename> on VMS
    then TeX prepends the filename with a chained logical
    "tex_inputs:" - equivalent to a Unix path. Unfortunately \openin
    does _not_ do this - although on Unix systems \openin will search
    the TEXINPUTS path just as \input does.

    FIX: Change the following line in xy.tex

\xydef@\xyinputorelse@@#1#2{\openin\xyread@=#1

    TO

\xydef@\xyinputorelse@@#1#2{\openin\xyread@=tex_inputs:#1


PC WITH PCTEX
-------------
On PCTeX fonts should be stored in files `C:\PIXEL\DPI<dpi>\<font>.pk'
where <dpi> and <font> are replaced by appropriate values, e.g., `600'
and `xyatip' (thanks to Alain Amiouni <alain_amiouni@msn.com>).
```

```
MAC WITH Textures or OzTeX
--------------------------
See the special files `INSTALL.Textures' and `INSTALL.OzTeX'.

---------------------------------------------------------------------------
-4- UPDATING FROM VERSION 2.6 (AND EARLIER)
---------------------------------------------------------------------------

IMPORTANT: If you have installed release 3.* or 2.7-2.12 then you can
skip this section. If you have not, and use version 2.6, then READ IT!

Xy-pic release 2.7 was almost a complete rewrite of Xy-pic; later
releases are updates to this version.  If you are using version 2.6 (or
some earlier version) then note the following:

* All the fonts have been redone AND renamed (except two that have not
  changed).  Thus installing version 3 will not break any DVI files
  you already have that were typeset using the old fonts.

  NOTE: In fact the distribution still contains the old fonts but only
  as METAFONT sources, TFM font metrics, and PK bitmaps.

* The distribution contains new versions of the files `xypic.tex' and
  `xypic.sty' that load Xy-pic in a special compatibility mode.  Thus
  retypesetting your documents will create DVI files using the new
  fonts.  If you wish to keep the old `xypic.tex' and `xypic.sty' then
  you should rename them before installation; we suggest naming them
  `xypic-2.tex' and `xypic-2.sty' and REMEMBER to change the line
  `\input xypic' in `xypic-2.sty' to `\input xypic-2' !

* There are a few points where the backwards compatibility is not
  perfect -- these are noted in the documentation.

---------------------------------------------------------------------------
-5- BUGS & COMMENTS
---------------------------------------------------------------------------

We will appreciate reports on any problems you may encounter and
opinions you have on how the usefulness of Xy-pic can be improved.
Please report using either

  Electronic mail (Internet): krisrose@ens-lyon.fr        [preferred]

  Paper mail:   Kristoffer H. Rose
                Laboratoire de l'Informatique du Parallélisme
                Ecole Normale Supérieure de Lyon
                46, Allée d'Italie
                F-69364 Lyon cedex 07
                FRANCE

  Fax:          +33 4 7272 8080

  W3            <URL: http://www.ens-lyon.fr/~krisrose/Xy-pic.html>
---------------------------------------------------------------------------

This is the end of INSTALL.  We hope you'll enjoy Xy-pic!
```

```
                                     -- Ross Moore and Kristoffer Rose

---------------------------------------------------------------------------

Kristoffer Høgsbro Rose, Ph.D., prof.associé  <Kristoffer.Rose@ENS-Lyon.FR>
Laboratoire de l'Informatique du Parallélisme  équipe PLUME, bureau LR5-026
Ecole Normale Supérieure de Lyon; 46, Allée d'Italie; F-69364 Lyon 07 cedex
phone: +33(0)4 7272 8642; fax:...8080    <http://www.ens-lyon.fr/~krisrose>

---------------------------------------------------------------------------
```

D.3 Generation

These files are relevant when XY-pic is generated from sources on a un*x system.

D.3.1 MAKE

```
-*-text-*- $Id: MAKE,v 3.5 1997/05/28 22:40:26 krisrose Exp $

           ================================================
           MAKE: Xy-pic diagram typesetting package SOURCES
           ================================================

Welcome to the source distribution of the Xy-pic macros for typesetting
diagrams!  Below you will find the following sections:

  -1- License information
  -2- Making the package and manual from sources
  -3- Bugs & comments
  -4- Distribution log

Please see the file `README' for a summary of features `TRAILER' for how
you can obtain the newest Xy-pic, and the various `INSTALL' files if you
have already retrieved pregenerated macros and fonts.


----------------------------------------------------------------------
  -1- LICENSE INFORMATION
----------------------------------------------------------------------

The Xy-pic package is licensed under `GNU CopyLeft':

  Xy-pic: Typesetting graphs and diagrams with TeX.
  Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@brics.dk>

  The Xy-pic macro package is free software; you can redistribute it
  and/or modify it under the terms of the GNU General Public License as
  published by the Free Software Foundation; either version 2 of the
  License, or (at your option) any later version.

  The Xy-pic macro package is distributed in the hope that it will be
  useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this package; if not, write to the Free Software
  Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

```
(Some options may be copyright by other people but the conditions remain
the same).

The full text of the GNU General Public License is supplied with Xy-pic
in the file ‘COPYING’.


----------------------------------------------------------------------
  -2- MAKING THE PACKAGE AND MANUAL FROM SOURCES
----------------------------------------------------------------------

This section describes how to make the Xy-pic package from sources on
a un*x system.  Making Xy-pic requires use (and some knowledge) of
several standard un*x utilities; furthermore the typesetting og the
reference manual needs a **LARGE** TeX (see below), and full
installation requires both the plain TeX and LaTeX2e formats as well
as METAFONT (with the plain and cm bases.

NOTE FOR VMS USERS: The procedure for making the package from sources
described below does not work on VMS, of course.  You can, however,
run the enclosed ‘support/doc2tex.com’ procedure file contributed by
R.Gaertner to create the .tex files from the .doc files.

NOTE FOR OTHER NON-UN*X USERS: If you do not have a un*x system then
you are best adviced to get the ‘ready to run’ package and install it
as described in ‘INSTALL’.

UN*X GENERATION AND INSTALLATION STEPS:

1. First make sure that you have a separate directory ‘src’ which
   contains all the files of the distribution (including this file).

   The file ‘MANIFEST’ contains a list of the files.

2. Next you should edit the ‘Configuration section’ of ‘Makefile’ to fit
   the configuration of your system by following the directions given
   there.

   When you have configured everything then just do

        make

   which should generate all the files needed to use the package.

3. You can test the newly generated system by typesetting the User’s
   Guide which you probably want for reference anyway: do

        make guide

   which should produce a file ‘xyguide.dvi’ by running LaTeX several
   times and BibTeX once.

4. If possible it is a good idea to try to preview the User’s Guide (in
   whatever way you do on your system) at this point.  This is slightly
   difficult because it uses the Xy-pic fonts which you haven’t
   installed yet, so the previewer should be instructed to look in the
   current directory before looking in the system font directories.  The
   following show how to do this for the most common previewer (on un*x
   systems with X windows) and might work for you:

        env XDVIFONTS=.: xdvi xyguide.dvi
```

```
5. You can then install the package by ensuring that the install
   directories in `Makefile' are setup correctly and do

        make install

   This will replace exisiting `xypic.tex' and `xypic.sty' files with
   the new (compatibility mode) ones; the old ones will remain available
   with an `.orig' suffix (if you want to keep the old version then read
   section 2 of `INSTALL').

6. You can typeset the entire manual with

        make manual

   to create both files `xyguide.dvi' and `xyrefer.dvi'; they can be
   installed in the TEXDOC directory with the command

        make install.manual

   NOTE: this requires a LARGE TeX system: the last typesetting (running
   LaTeX on `xyrefer.man' without any driver) uses more than 600000 TeX
   words, as noted above.  Using Karl Berry's Web2C 7.0 this is
   achieved by including the lines

extra_mem_top = 400000 % extra high memory for chars, tokens, etc.
extra_mem_bot = 400000 % extra low memory for boxes, glue, breakpoints, etc.

   in the texmf.cnf file.

7. You can build format files by the commands

        make xytex.fmt
        make xylatex.fmt

   (the latter requires the new standard LaTeX release aka LaTeX2e).
   These assume that the files `plain.tex' and `latex.ltx' are available
   in the TeX search path; also you have to install the format files in
   the appropriate place yourself.

That is all.


----------------------------------------------------------------------
  -3- BUGS & COMMENTS
----------------------------------------------------------------------

I will appreciate reports on any problems you may encounter and
opinions you have on how the usefulness of Xy-pic can be improved.
Please report using either

  Electronic mail (Internet): krisrose@brics.dk [preferred]

  Paper mail:   Kristoffer H. Rose
                BRICS/Computer Science
                University of Aarhus
                Ny Munkegade, building 540
                DK-8000 Aarhus C
                DENMARK

  Fax:          +45 89423255  att. K.Rose / Karen Møller

You can always get the latest news of Xy-pic from computers connected
```

```
to the Internet by opening the Xy-pic World Wide Web page:

  W3            <URL: http://www.brics.dk/~krisrose/Xy-pic.html>

----------------------------------------------------------------------
  -4- DISTRIBUTION LOG
----------------------------------------------------------------------

The following versions have been distributed publicly:

$Log: MAKE,v $
Revision 3.5  1997/05/28 22:40:26  krisrose
Fixed missing breaks bug.

Revision 3.3  1996/12/19 03:31:56  krisrose
Maintenance release

Revision 3.1  1995/09/05  20:31:32  kris
Releasing!

Revision 3.0  1995/07/07  20:14:21  kris
Major release w/new User's Guide!

Renamed to MAKE for version 3 source description.

Revision 2.12  1994/10/25  11:34:25  kris
Interim release just before v3 [works with AMS-LaTeX 1.2]...

Revision 2.11  1994/07/05  10:37:32  kris
Third 3beta release [bug fixes].
Experimental graph feature included (for ECCT-94 presentation).

Revision 2.10  1994/06/15  12:59:44  kris
Second 3beta release [bug fixes].

Revision 2.8  1994/04/11  09:31:09  kris
Second (bug fix) 3alpha release [corrected].

Revision 2.7  1994/03/08  02:06:01  kris
Release 3alpha.

MAJOR REWRITE of distribution to prepare for Xy-pic 3 functionality.
Curve extension code contributed by Ross Moore <ross@mpce.mq.edu.au>;
support funding from MURG, ARC, and equipment from DEC.

Revision 2.6  1992/06/24  01:23:34  kris
Added `hook' tips using new font xyqc10.
Added new POSitions: * and !.
Added triple lines \Ssolid, \Ddashed, \Ddotted, and companion \Ttip.
Reorganised manual.

Revision 2.5  1992/02/24  03:30:54  kris
Fixed bugs in \Direction calculation logic...
Added (FACTOR) to \rotate to allow arbitrary rotation.
` intermediate points now accept an optional /RADIUS argument.
Added \Tip with wide tip.

Revision 2.4  1992/01/22  02:15:10  kris
Fixed bugs [with thanks]:
\\ works with AMS-LaTeX: don't use \Let@ [Werner Struckmann/Darrel Hankerson]
```

```
No spurious arrow heads with LaTeX: \pit now undefined [Werner Struckmann]
\Solid works: sets \Density [Dave Bowen]
Short diagonal lines work...major rewrite of \connectv@ [Eric Domenjoud]

Revision 2.3  1992/01/10  21:38:27  kris
Several bugs fixed [with thanks]:
The \ddtoX and \uutoX arrows were interchanged [Nico Verwer].
Some diagonal lines were wrong [Eric Domenjoud].
AMS-LaTeX users had name clash problems [Werner Struckmann].
The installation instructions in the Makefile were buggy [Leen Torenvliet].

Revision 2.2  1992/01/09 04:05:40  kris
Patched to fix problem with \text and support AMS-LaTeX.

Revision 2.1  1992/01/02 14:54:07  kris
Release version.

Revision 1.40  1991/12/17 04:53:23 kris
Version distributed as ‘final draft’ on Usenet.

---------------------------------------------------------------------

This is the end of the source overview.  I hope you will enjoy Xy-pic!

---------------------------------------------------------------------

Kristoffer Høgsbro ROSE                           <krisrose@brics.dk>
BRICS                               <URL: http://www.brics.dk/~krisrose/>
Department of Computer Science B3.26, +45 89423193 (fax +45 89423255)
University of Aarhus, Ny Munkegade, build. 540, 8000 Århus C, DENMARK

---------------------------------------------------------------------
```

D.3.2 Makefile

```
# $Id: Makefile,v 3.6 1998/03/06 01:28:05 krisrose Exp krisrose $
#
# Un*x Makefile for Xy-pic macros, fonts, documentation, and distribution.
# Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@ens-lyon.fr>
#
# This file is part of the Xy-pic macro package.
# Copyright (c) 1991-1997  Kristoffer H. Rose  <krisrose@ens-lyon.fr>
# See the README and INSTALL files for further information.
#
# The Xy-pic package is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by the
# Free Software Foundation; either version 2 of the License, or (at your
# option) any later version.
#
# The Xy-pic package is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
# or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
# for more details.
#
# You should have received a copy of the GNU General Public License along
# with this package; if not, write to the Free Software Foundation, Inc.,
```

```
# 675 Mass Ave, Cambridge, MA 02139, USA.

VERSION = 3.7
SHORTVERSION = 37

# CONTENTS:
#
#   Configuration section:
#       Directories.
#       Fonts.
#       Programs.
#
#   Generic targets.
#   Macros.
#   PostScript headers.
#   Fonts.
#   Formats.
#   Manual: User's guide and reference manual.
#   Reference manual.
#   TeXnical documentation.
#   Distribution tree.
#   Ftp'able distribution.
#   WWW publicity.
#   Announcements.
#   Tags.
#   RCS stuff.
#   Development targets.
#
#   Log.

#### CONFIGURATION SECTION ##################################################

# Please configure as instructed below for each of the DIRECTORIES, FONTS,
# and PROGRAMS parts by editing the definitions to fit your system.

#### DIRECTORIES.
#
# The following variables define the directories where the various parts of
# Xy-pic should be installed.
#
# You should *EITHER* set TDSROOT to the root of a TDS-compliant directory
# tree, usually named .../texmf (if you have a TDS [TeX Directory Structure]
# compliant system then that is in fact all you need to do):
#
USR = /usr/local
TDSROOT = $(USR)/share/texmf
#
# *OR* you should set each of the following variables right for you system
# (do this if your setup does not match TDS exactly):
#
TEXINPUTDIR = $(TDSROOT)/tex/generic/xypic
TEXFONTDIR = $(TDSROOT)/fonts/tfm/public/xypic
```

```
TEXDOCDIR = $(TDSROOT)/doc/generic/xypic
MFINPUTDIR = $(TDSROOT)/fonts/source/public/xypic
TYPE1DIR = $(TDSROOT)/fonts/type1/public/xypic
PFMDIR = $(TDSROOT)/fonts/pfm/public/xypic
AFMDIR = $(TDSROOT)/fonts/afm/public/xypic
PSHEADERDIR = $(TDSROOT)/dvips/xypic
#
# FONTDIR is the directory where bitmap fonts are stored and is special
# (If you do not plan to install bitmap fonts because they are automatically
# generated and remember to set the MAGS variable below to `.' then you may
# ignore this step.)
#
# Since the location of individual fonts sometimes depend on properties of
# the font, you can insert
#
#       $${dpi}         for the font resolution in `Dots Per Inch' (NOT the
#                       device resolution---that should be inserted directly)
#
#       $${mode}        for the METAFONT `mode' used to generate the font
#
# ---what these get set to depends on the next section.
#
# For example, if you set FONTDIR=/usr/lib/tex/pk$${dpi} and decide in the
# next section to generate PK fonts at 300 dpi using a generic mode
# (localfont) then you get files named /usr/lib/tex/pk300/xy*.300pk.
#
# If your installation still uses the (inefficient) principle of having all
# PK files in the same directory as the TFM files then you should just set
# FONTDIR=$(TEXFONTDIR).
#
FONTDIR = $(TDSROOT)/fonts/pk/$${mode}/public/xypic/dpi$${dpi}

#### FONTS.
#
# The METAFONT program needs to know the `mode' of your printer in order to
# be able to generate correct fonts for it.  On most installations the mode
# `localfont' is made synonymous with the mode required for the default
# printer; in that case you can just pick the `MODES = localfont' line.  If
# this is not the case -- or you need fonts for several output devices---then
# you will have to consult the local METAFONT guide/guru to inquire what
# modes are used at your site.  A good place to look for the modes used is in
# the MakeTeXPK script found on some sites.
#
# Set MODES to the list of printers you wish to use:
#
MODES = localfont
#MODES = cx ljfour
#
# You should also decide which TeX \magsteps you wish the font to be
# generated at.
```

```
#
# The default is . (dot) which prevents installation of any bitmaps because
# most modern installations generate them automatically.
#
MAGS = .
#MAGS = 0
#MAGS = 0 0.5 1 2

#### PROGRAMS.
#
# You don't have to edit the following lines unless you have a non-standard
# un*x system or un*x TeX installation...
#
SHELL = /bin/sh
#
TEX = TEXFONTS=.:$(TEXFONTDIR):$$TEXFONTS tex
LATEX = TEXFONTS=.:$(TEXFONTDIR):$$TEXFONTS latex
PDFLATEX = TEXFONTS=.:$(TEXFONTDIR):$$TEXFONTS pdflatex
BIBTEX = BIBINPUTS=. bibtex
MAKEINDEX = makeindex
INITEX = TEXFONTS=.:$(TEXFONTDIR):$$TEXFONTS initex
#
# Set MF to a command that will run plain METAFONT on your system.
#
MF = MFINPUTS=.:$(MFINPUTDIR):$$MFINPUTS mf
#
# Set GFTOPK to the command "false" and PK to "gf" if you use GF rather than PK
# bitmap font files.  There is no support for PXL-files [are they still used?].
# (Note: the default setting of MAGS above doesn't use bitmaps at all.)
#
GFTOPK = gftopk
PK=pk
#
#GFTOPK = false
#PK=gf
#
RM = rm -fr
CP = cp -f
MV = mv -f
#
# Set the INSTALLs to cp -f if your system does not have the install command.
#
INSTALL = install -c -m 444
INSTALLW = install -c -m 644
INSTALLX = install -c -m 555
#INSTALL = cp -f
#INSTALLW = cp -f
#INSTALLX = cp -f
#
# Set MKDIR to a mkdir command that creates intermediate directories.
```

```
#
MKDIR = mkdir -p -m 755
#
# DOC2TEX <file>: output efficient .tex file from inefficient .doc <file> with
# inlined documentation [my favourite hack :-].  Removes all DOCMODE lines,
# leaving only things before the first and in DOCMODE(...DOCMODE) brackets;
# also removes all blank lines and comment lines.
#
# If it breaks your sed you can just set DOC2TEX to cat although that will
# make the installed macro files three times larger and the reading of them
# somewhat slower...so in that case you are probably better of getting the
# `run' distribution where this has already been done for you!
#
DOC2TEX = sed \
        -e '/^.DOCMODE(/b Yes' \
        -e '/^.DOCMODE/b No' \
        -e 'x' \
        -e '/%%DONTCOPY%%/b Ignore' \
        -e 'x' \
        -e 's/[         ][      ]*/ /g' \
        -e 's/{%.*$$/{%/' -e 's/}%.*$$/}%/' -e 's/ %.*$$//'  \
        -e '/^ *%$$/d' -e '/^ *$$/d' -e '/^ *%[^%]/d' \
        -e 's/^%% \$$Id/%% $@ from $$Id/' \
        -e 's/REPLACEWITHVERSION/$(VERSION)/' \
        -e 's/REPLACEWITHSHORTVERSION/$(SHORTVERSION)/' \
        -e 'b End' \
        -e ': No' -e 's/.*/%%DONTCOPY%%/' -e 'x' -e 'd' \
        -e ': Yes' -e 's/.*//' -e 'x' -e 'd' \
        -e ': Ignore' -e 'x' -e 'd' \
        -e ': End'
#
# MF2TFM script: make .tfm file and all requested bitmap files from .mf
# file by executing METAFONT with $${mode} and $${dpi} as described above.
#
# NOTE: for use as implicit make rule; do not modify unless you are sure you
# need to!
#
MF2TFM = set -x; for mode in $(MODES); do \
        if [ "$(MAGS)" = "." ]; then \
          $(MF) "\\mode=$$mode; input $<";\
          $(RM) $*.*gf;\
        else \
          for mag in $(MAGS); do \
            $(MF) "\\mode=$$mode; mag=magstep($$mag); input $<"; done;\
          for f in $*.*gf; do if test -f "$$f"; then \
            if $(GFTOPK) ./$$f; then $(RM) $$f; fi; else :; fi; done;\
        fi; done

#### END OF CONFIGURATION SECTION ##########################################
```

```
# GENERIC TARGETS.
#
all:            macros fonts obsoletefonts
.PHONY:         all install clean realclean sterile world
#
install:        all install.macros install.trailer \
                install.fonts install.obsoletefonts \
                install.ps install.psfonts
#
clean:
        $(RM) *.aux *.toc *.lof *.blg *.log *.idx *.ilg *.fmt \
          *.ps *.ppm *.nops *.tmp *.ans *.xyc *.xyd \
          ./#* *~ core *.BAK *JNL xytest.*gf out
#
realclean:      clean
        $(RM) MANIFEST VERSIONS CATALOG
        $(RM) xydoc.back; touch xydoc.back
        $(RM) *.bbl *.glo *.ind \
          *.tfm *.*gf *.*pk *.tex *.dvips *.dvi *.pdf *.last *.pro \
          *.gif *.epsf *.size *.pbm *.ppm \
          *.afm *.pfm *.pfa *.pfb *.asc
        $(RM) xyguide-html index*
        $(RM) pnmrawtopcropwhite
#
sterile:        realclean rcsclean
        $(RM) xydoc.back make.* TAGS
        $(MAKE) Makefile xydoc.sty xydoc.bib
#
world:          realclean rcsclean
        time $(MAKE) all ftp ftpdoc www 2>&1 |tee make.world

# MACROS.
#
TRAILERSOURCES =        TRAILER COPYING CATALOG.doc \
                        MAKE README Makefile $(INSTALLSTAR)

INSTALLSTAR =           INSTALL INSTALL.OzTeX INSTALL.Textures

KERNELSOURCES =         xy.doc xyidioms.doc xyrecat.doc

OPTIONSOURCES =         \
        xyframe.doc xycmtip.doc xytips.doc xycurve.doc \
        xyline.doc xyrotate.doc xycolor.doc xycrayon.doc xytile.doc \
        xyimport.doc xytpic.doc xytp-f.doc \
        xyps.doc xyps-ps.doc xyps-pro.doc xyps-col.doc xyps-c.doc xyps-f.doc \
        xyps-l.doc xyps-r.doc xyps-s.doc xyps-t.doc \
        xypsdict.doc xypspatt.doc xydocps.doc \
        xydummy.doc xyall.doc xymatrix.doc xyarrow.doc xygraph.doc \
        xy2cell.doc xypoly.doc xyarc.doc xyknot.doc xyweb.doc \
        xydvips.doc xytextures.doc xy16textures.doc xyoztex.doc xy17oztex.doc \
        xycmactex.doc xydvitops.doc xyemtex.doc xydvidrv.doc xyxdvi.doc \
        xymovie.doc xyv2.doc xypic.doc \
```

```
        xynecula.doc xysmart.doc

FORMATSOURCES =         xytex.ini xylatex.ini

LATEXSTYLES =           xy.sty xypic.sty movie.cls
OPTIONDATA =            xymacpat.xyp

MACROSOURCES =          $(TRAILERSOURCES) $(KERNELSOURCES) $(LATEXSTYLES) \
                        $(OPTIONSOURCES) $(OPTIONDATA) $(FORMATSOURCES)

TRAILERRUN =            TRAILER COPYING CATALOG README \
                        $(INSTALLSTAR) VERSIONS

KERNELRUN =             xy.tex xyidioms.tex xyrecat.tex

OPTIONRUN =             \
        xyframe.tex xycmtip.tex xytips.tex xycurve.tex \
        xyline.tex xyrotate.tex xycolor.tex xycrayon.tex xytile.tex \
        xyimport.tex xytpic.tex xytp-f.tex \
        xyps.tex xyps-ps.tex xyps-pro.tex xyps-col.tex xyps-c.tex xyps-f.tex \
        xyps-l.tex xyps-r.tex xyps-s.tex xyps-t.tex \
        xypsdict.tex xypspatt.tex $(OPTIONDATA) \
        xydummy.tex xyall.tex xymatrix.tex xyarrow.tex xygraph.tex \
        xy2cell.tex xypoly.tex xyarc.tex xyknot.tex xyweb.tex \
        xydvips.tex xytextures.tex xy16textures.tex xyoztex.tex xy17oztex.tex \
        xycmactex.tex xydvitops.tex xyemtex.tex xydvidrv.tex xyxdvi.tex \
        xymovie.tex xyv2.tex xypic.tex \
        xynecula.tex xysmart.tex

MACRORUN =              $(KERNELRUN) $(OPTIONRUN) $(FORMATSOURCES) \
                        $(LATEXSTYLES)
PSRUN =                 xy$(SHORTVERSION)dict.pro

SUPPORT =               install-tds doc2tex.com \
                        dvitogif89a pnmrawtopcropwhite.c

# Macros have .tex suffix and are generated from the .doc files.
#
.SUFFIXES:      .tex .doc
.doc.tex:;      $(DOC2TEX) $< > $@
#
.PHONY:         macros install.macros
#
macros:         $(MACRORUN)
install.macros: $(MACRORUN)
        -test -d $(TEXINPUTDIR) || $(MKDIR) $(TEXINPUTDIR)
        for f in $(MACRORUN); do $(INSTALL) $$f $(TEXINPUTDIR)/$$f; done
#
# Kernel dependencies:
#
xy.tex:                         xy.doc
xyidioms.tex:                   xyidioms.doc
xyrecat.tex:                    xyrecat.doc
#
# Extension option dependencies:
```

```
#
xyframe.tex:                    xyframe.doc
xytips.tex:                     xytips.doc
xycmtip.tex:                    xycmtip.doc
xycurve.tex:                    xycurve.doc
xyline.tex:                     xyline.doc
xyrotate.tex:                   xyrotate.doc
xycolor.tex:                    xycolor.doc
xycrayon.tex:                   xycrayon.doc
xyimport.tex:                   xyimport.doc
xytile.tex:                     xytile.doc
#
xytpic.tex:                     xytpic.doc
xytp-f.tex:                     xytp-f.doc
#
xyemtex.tex:                    xyemtex.doc
#
xyps.tex:                       xyps.doc
xyps-ps.tex:                    xyps-ps.doc
xyps-pro.tex:                   xyps-pro.doc
xyps-col.tex:                   xyps-col.doc
xyps-c.tex:                     xyps-c.doc
xyps-f.tex:                     xyps-f.doc
xyps-l.tex:                     xyps-l.doc
xyps-r.tex:                     xyps-r.doc
xyps-s.tex:                     xyps-s.doc
xyps-t.tex:                     xyps-t.doc
xypsdict.tex:                   xypsdict.doc
xypspatt.tex:                   xypspatt.doc
#
# Feature option dependencies:
#
xydummy.tex:                    xydummy.doc
xyall.tex:                      xyall.doc
#
xymatrix.tex:                   xymatrix.doc
xyarrow.tex:                    xyarrow.doc
xygraph.tex:                    xygraph.doc
xy2cell.tex:                    xy2cell.doc
xypoly.tex:                     xypoly.doc
xyarc.tex:                      xyarc.doc
xyknot.tex:                     xyknot.doc
xyweb.tex:                      xyweb.doc
#
# Driver option dependencies:
#
xycmactex.tex:                  xycmactex.doc
xydvips.tex:                    xydvips.doc
xydvitops.tex:                  xydvitops.doc
```

```
xydvidrv.tex:                   xydvidrv.doc
xydocps.tex:                    xydocps.doc
xytextures.tex:                 xytextures.doc
xy16textures.tex:               xy16textures.doc
xyoztex.tex:                    xyoztex.doc
xymovie.tex:                    xymovie.doc
xy17oztex.tex:                  xy17oztex.doc
xyxdvi.tex:                     xyxdvi.doc
#
# Backwards compatibility dependencies:
#
xyv2.tex:                       xyv2.doc
xypic.tex:                      xypic.doc
#
# Required documentation files
#
install.trailer:        $(TRAILERRUN)
        -test -d $(TEXDOCDIR) || $(MKDIR) $(TEXDOCDIR)
        for f in $(TRAILERRUN); do $(INSTALL) $$f $(TEXDOCDIR)/$$f; done

# PostScript HEADERS:
#
.PHONY: install.ps
install.ps:             $(PSRUN)
        test -d $(PSHEADERDIR) || $(MKDIR) $(PSHEADERDIR)
        for f in $(PSRUN); do $(INSTALL) $$f $(PSHEADERDIR)/$$f; done

# FONTS.
#
.PHONY:         mfinputs texfonts $(PK)fonts fonts install.fonts \
                psfonts install.psfonts obsoletefonts install.obsoletefonts
#
METAFONTS = \
        xydash10.mf xybsql10.mf xyd2.mf \
        xyatip10.mf xybtip10.mf xytech.mf xyatip.mf xybtip.mf xyd.mf \
        xycirc10.mf \
        xycmat10.mf xycmbt10.mf xycmat11.mf xycmbt11.mf \
         xycmat12.mf xycmbt12.mf xycm.mf \
        xyeuat10.mf xyeubt10.mf xyeuat11.mf xyeubt11.mf \
         xyeuat12.mf xyeubt12.mf xyeuler.mf
#
TEXFONTRUN = \
        xydash10.tfm xybsql10.tfm xyatip10.tfm xybtip10.tfm xycirc10.tfm \
        xycmat10.tfm xycmbt10.tfm xycmat11.tfm xycmbt11.tfm \
         xycmat12.tfm xycmbt12.tfm \
        xyeuat10.tfm xyeubt10.tfm xyeuat11.tfm xyeubt11.tfm \
         xyeuat12.tfm xyeubt12.tfm
#
# Note: we only depend on the tfm-files, so you must remove them to
# regenerate the fonts at all desired magnifications.
```

```
#
.SUFFIXES:      .mf .tfm
.mf.tfm:;       $(MF2TFM)
#
mfinputs:       $(METAFONTS)
#
texfonts $(PK)fonts fonts: $(METAFONTS) $(TEXFONTRUN)
install.fonts:  fonts
        if test -n "$(MFINPUTDIR)"; then \
          test -d $(MFINPUTDIR) || $(MKDIR) $(MFINPUTDIR); \
          for f in $(METAFONTS);do $(INSTALL) $$f $(MFINPUTDIR)/$$f;done; fi
        -test -d $(TEXFONTDIR) || $(MKDIR) $(TEXFONTDIR)
        for f in $(TEXFONTRUN); do $(INSTALL) $$f $(TEXFONTDIR)/$$f; done
        #
        [ "$(MAGS)" = "." ] || \
         for mode in $(MODES); do for mag in $(MAGS); do \
          $(RM) xytest.*gf xytest.*tfm;\
          $(MF) "\\mode=$$mode; mag=magstep($$mag); input xytest";\
          dpi="`expr xytest.*gf : 'xytest\.\([0-9]*\)gf'`";\
          for f in $(TEXFONTRUN); do \
            f="`expr $$f : '\([a-z0-9]*\)\.tfm'`.$${dpi}$(PK)" ;\
            if [ ! -f $$f ]; then echo "ERROR: FONT $$f NOT GENERATED";\
            else test -d "$(FONTDIR)" || $(MKDIR) $(FONTDIR);\
              $(INSTALL) $$f $(FONTDIR)/$$f; fi; done; done; done
#
xydash10.tfm:   xydash10.mf xyd2.mf xytest.mf
xybsql10.tfm:   xybsql10.mf xyd.mf xytest.mf
xycirc10.tfm:   xycirc10.mf xytest.mf
xyatip10.tfm:   xyatip10.mf xytech.mf xyatip.mf xyd.mf xytest.mf
xybtip10.tfm:   xybtip10.mf xytech.mf xybtip.mf xyd.mf xytest.mf
xycmat10.tfm:   xycmat10.mf xyatip.mf xyd.mf xytest.mf
xycmbt10.tfm:   xycmbt10.mf xybtip.mf xyd.mf xytest.mf
xyeuat10.tfm:   xyeuat10.mf xyatip.mf xyd.mf xytest.mf
xyeubt10.tfm:   xyeubt10.mf xybtip.mf xyd.mf xytest.mf
#
# Obsolete fonts (from 2.6 and before).
#
OBSOLETEMETAFONTS = xyline10.mf xyqc10.mf xymisc10.mf
OBSOLETETEXFONTRUN = xyline10.tfm xyqc10.tfm xymisc10.tfm
#
obsoletefonts:  $(OBSOLETEMETAFONTS) $(OBSOLETETEXFONTRUN)
#
install.obsoletefonts:  obsoletefonts
        if test -n "$(MFINPUTDIR)"; then \
          test -d $(MFINPUTDIR) || $(MKDIR) $(MFINPUTDIR); \
          for f in $(OBSOLETEMETAFONTS);do $(INSTALL) $$f $(MFINPUTDIR)/$$f;done; fi
        -test -d $(TEXFONTDIR) || $(MKDIR) $(TEXFONTDIR)
        for f in $(OBSOLETETEXFONTRUN); do \
          $(INSTALL) $$f $(TEXFONTDIR)/$$f; done
```

```
        #
        [ "$(MAGS)" = "." ] || \
         for mode in $(MODES); do for mag in $(MAGS); do \
          $(RM) xytest.*gf xytest.*tfm;\
          $(MF) "\\mode=$$mode; mag=magstep($$mag); input xytest";\
          dpi="`expr xytest.*gf : 'xytest\.\([0-9]*\)gf'`";\
          for f in $(OBSOLETETEXFONTRUN); do \
            f="`expr $$f : '\([a-z0-9]*\)\.tfm'`.$${dpi}$(PK)" ;\
            if [ ! -f $$f ]; then echo "ERROR: FONT $$f NOT GENERATED";\
            else test -d "$(FONTDIR)" || $(MKDIR) $(FONTDIR);\
              $(INSTALL) $$f $(FONTDIR)/$$f; fi; done; done; done
#
xyline10.tfm:   xyline10.mf xytest.mf
xyqc10.tfm:     xyqc10.mf xytest.mf
xymisc10.tfm:   xymisc10.mf xytest.mf

# PostScript fonts w/support files...from Y&Y.
#
TYPE1FONTS = \
        xyatip10.pfb xybsql10.pfb xybtip10.pfb xycirc10.pfb xydash10.pfb \
        xycmat10.pfb xycmbt10.pfb xyeuat10.pfb xyeubt10.pfb
#
PFMFONTS = \
        xyatip10.pfm xybsql10.pfm xybtip10.pfm xycirc10.pfm xydash10.pfm \
        xycmat10.pfm xycmbt10.pfm xyeuat10.pfm xyeubt10.pfm
#
AFMFONTS = \
        xyatip10.afm xybsql10.afm xybtip10.afm xycirc10.afm xydash10.afm \
        xycmat10.afm xycmbt10.afm xyeuat10.afm xyeubt10.afm
#
YANDYFONTS = $(TYPE1FONTS) $(AFMFONTS) $(PFMFONTS)
#
psfonts: $(YANDYFONTS)
#
$(YANDYFONTS): YandY_xy_fonts.zip
        unzip -aoL YandY_xy_fonts.zip
        touch $(YANDYFONTS)
#
install.psfonts: psfonts xypic.map
        test -d $(TYPE1DIR) || $(MKDIR) $(TYPE1DIR)
        test -d $(PFMDIR) || $(MKDIR) $(PFMDIR)
        test -d $(AFMDIR) || $(MKDIR) $(AFMDIR)
        for f in $(TYPE1FONTS); do $(INSTALL) $$f $(TYPE1DIR)/$$f; done
        for f in $(PFMFONTS); do $(INSTALL) $$f $(PFMDIR)/$$f; done
        for f in $(AFMFONTS); do $(INSTALL) $$f $(AFMDIR)/$$f; done
        test -d $(PSHEADERDIR) || $(MKDIR) $(PSHEADERDIR)
        $(INSTALL) xypic.map $(PSHEADERDIR)/xypic.map
#
# All together...
#
```

```
FONTSOURCES = $(METAFONTS) $(OBSOLETEMETAFONTS) xytest.mf \
        YandY_xy_fonts.zip xypic.map

# FORMATS:
#
# Build formats...
#
.SUFFIXES:      .ini .fmt
.ini.fmt:;      $(INITEX) $<
#
xytex.fmt:      xytex.ini macros
xylatex.fmt:    xylatex.ini macros
#xyamslatex.fmt:        xyamslatex.ini macros
#xydoc.fmt:     xydoc.ini macros


# MANUAL: USER'S GUIDE AND REFERENCE MANUAL.
#
# Note: the first generations use the DVIPS backend to run faster.
#
.PHONY:         guide manual install.manual manual.ps
#
MANUALDATA = ross.eps kris.eps import1.eps
MANUALSOURCES =         \
        xyguide.man xydoc.sty latin1.sty xydoc.bib xydoc.back \
        xyrefer.man $(KERNELSOURCES) $(LATEXSTYLES) $(OPTIONSOURCES) \
        $(OPTIONDATA) $(MANUALDATA)
#
MANUALRUN =     xyguide.dvi xyrefer.dvi
#
.SUFFIXES:      .man .dvips .dvi .pdf
.man.dvips:
        $(MAKE) macros fonts
        echo "\\xyoption{dvips}\xyoption{ps}" > xydoc.back
        $(LATEX) $<
        @-$(CP) $<.dvi $*.dvi ; $(CP) $<.aux $*.aux ; $(CP) $<.idx $*.idx
        $(BIBTEX) $*
        @-$(CP) $*.bbl $<.bbl
        [ -f $*.idx ] && $(MAKEINDEX) $* || :
        @-$(CP) $*.ind $<.ind
        echo "\\xyoption{dvips}\xyoption{ps}" > xydoc.back
        $(LATEX) $<
        @-$(CP) $<.dvi $*.dvi ; $(CP) $<.aux $*.aux ; $(CP) $<.idx $*.idx
        [ -f $*.idx ] && makeindex $* || :
        @-$(CP) $*.ind $<.ind
        echo "\\xyoption{dvips}\xyoption{ps}" > xydoc.back
        $(LATEX) $<
        @-$(CP) $<.dvi $*.dvi ; $(CP) $<.aux $*.aux ; $(CP) $<.idx $*.idx
        mv $*.dvi $*.dvips
        touch $*.dvips
#
```

```
.dvips.dvi:
	echo '' > xydoc.back
	$(LATEX) $*.man
	@-$(CP) $*.man.dvi $*.dvi ; $(CP) $*.man.aux $*.aux ; $(CP) $*.man.idx $*.idx
	touch $*.dvi
#
.dvips.pdf:
	echo '' > xydoc.back
	$(PDFLATEX) $*.man
	@-$(CP) $*.man.pdf $*.pdf ; $(CP) $*.man.aux $*.aux ; $(CP) $*.man.idx $*.idx
	touch $*.pdf
#
guide:		xyguide.dvi
#
manual:		$(MANUALRUN)
install.manual: manual
	-test -d $(TEXDOCDIR) || $(MKDIR) $(TEXDOCDIR)
	for f in $(MANUALRUN); do $(INSTALL) $$f $(TEXDOCDIR)/$$f; done
#
xyguide.dvips:  xyguide.man xydoc.sty latin1.sty xydoc.bib version.tex
xyrefer.dvips:  xyrefer.man xydoc.sty latin1.sty xydoc.bib version.tex \
	$(KERNELSOURCES) $(OPTIONSOURCES) $(MANUALDATA)
xytest.dvips:   xytest.man xydoc.sty latin1.sty xydoc.bib version.tex \
	$(KERNELSOURCES) $(OPTIONSOURCES) $(MANUALDATA)
#
xyguide.dvi:    xyguide.dvips version.tex
xyrefer.dvi:    xyrefer.dvips version.tex
xytest.dvi:     xytest.dvips version.tex
#
xyguide.pdf:    xyguide.dvips version.tex
xyrefer.pdf:    xyrefer.dvips version.tex
xytest.pdf:     xytest.dvips version.tex
#
version.tex:    version.doc
	$(LATEX) version.doc
#
.SUFFIXES:      .pro
xy$(SHORTVERSION)dict.pro: xydocps.tex xypsdict.tex xypspatt.tex xyps-pro.tex
	-$(RM) xy$(SHORTVERSION)dict.pro
	$(TEX) xydocps.tex
#
# PostScript targets:
#
# DVIPS should be Tom Rokicki's dvips PostScript DVI driver as a filter.
DVIPS = dvips -D600 -f
#
.SUFFIXES:      .ps
.dvips.ps:;     $(DVIPS) < $< > $@
#
```

```
xyguide.ps:     xyguide.dvips $(PSRUN)
xyrefer.ps:     xyrefer.dvips $(PSRUN)
#
manual.ps:      xyguide.ps xyrefer.ps
```

```
# TeXNICAL DOCUMENTATION.
#
.PHONY:         TeXnical
TeXnical:       xysource.ps xysource.dvi xysource.pdf
#
TeXNICALSOURCES = xysource.man COPYING.patch xydoc.back
#
TeXNICALAUX = \
        TRAILER CATALOG README $(INSTALLSTAR) MAKE MANIFEST \
        COPYING.tex
#
xysource.dvips: xysource.man macros.tmp \
                macros fonts obsoletefonts $(ALLSOURCES) $(TeXNICALAUX)
xysource.ps:    xysource.dvips
xysource.dvi:   xysource.dvips
xysource.pdf:   xysource.dvips
#
xywork.dvips: xywork.man macros.tmp \
                macros fonts obsoletefonts $(ALLSOURCES) $(TeXNICALAUX)
xywork.ps:      xywork.dvips
xywork.dvi:     xywork.dvips
xywork.pdf:     xywork.dvips
#
COPYING.tex: COPYING COPYING.patch
        patch COPYING -o COPYING.tex < COPYING.patch
#
FONTCOPYING.tex: FONTCOPYING FONTCOPYING.patch
        patch FONTCOPYING -o FONTCOPYING.tex < FONTCOPYING.patch
#
# Macro index...requires GNU AWK (gawk) or (in a pinch) NAWK.
#
GAWK = gawk
#
macros.tmp: $(KERNELSOURCES) $(LATEXSTYLES) $(OPTIONSOURCES) $(METAFONTS)
        -$(RM) macros.tmp
        for f in $(KERNELSOURCES) $(LATEXSTYLES) $(OPTIONSOURCES) $(METAFONTS);\
        do $(GAWK) '\
          func err(s) { printf("%s,%d: %s\n",FILENAME,FNR,s) >>"/dev/stderr"};\
          func prt(s,i) { \
           printf("\\filelinedef{%s}{%d}\\index def{%s}\n",FILENAME,FNR,s) };\
          func prtdef(i) { t = substr($$0,i);\
           if (match(t,/[\\|][a-zA-Z@]*|\\.|[^\\|]|{([^{}]|\\{|\\})*}/) != 1) \
            err("Weird TeX token");\
           else prt(substr($$0,i,RLENGTH),i) };\
          func prtcsdef(i) { t = substr($$0,i);\
```

```
           if (match(t,/{([^{}]|\\{|\\})*}|{([^{}]|\\{|\\})*{([^{}]|\\{|\\})*}([^{}]|\\{|\\})*}/) != 1) \
            err("Weird TeX cs name token");\
           else prt(substr($$0,i+1,RLENGTH-2),i) };\
          /^\\let[^A-Za-z@]/            ||\
          /^\\def[^A-Za-z@]/            { prtdef(5); next };\
          /^\\gdef[^A-Za-z@]/           ||\
          /^\\edef[^A-Za-z@]/           ||\
          /^\\xdef[^A-Za-z@]/           { prtdef(6); next };\
          /^\\newif[^A-Za-z@]/          { prtdef(7); next };\
          /^\\xydef@[^A-Za-z@]/         ||\
          /^\\xylet@[^A-Za-z@]/         { prtdef(8); next };\
          /^\\xyfont@[^A-Za-z@]/        { prtdef(9); next };\
          /^\\global\\let[^A-Za-z@]/    { prtdef(12); next };\
          /^\\xywarnifdefined[^A-Za-z@]/ { prtdef(17); next };\
          /^\\ifx\\undefined[^A-Za-z@]/ { prtdef(15); next };\
          /^\\xynew@{[^{}]*}/           { prtdef(index($$0,"}")+1); next };\
          /^\\xydefcsname@{/            { prtcsdef(14); next };\
          /^\\xyletcsnamecsname@{/      { prtcsdef(20); next }' $$f;\
        done | sed -e 's,\\{,\\\\otherebgroup,' -e 's,\\},\\\\otheregroup,' \
        > macros.tmp

###printf("\\macroentry{%s}{%s}{%d}\n",s,FILENAME,FNR) };\

# Xy-pic the MOVIE :)
#
.SUFFIXES: .texmovie .gif
#
.texmovie.gif:
        $(LATEX) '\documentclass[dvips]{movie}' \
                '\usepackage[all]{xy}' \
                '\begin{document}' \
                '\input{$*.texmovie}' \
                '\end{document}'
        @-[ -f texput.dvi ] && $(MV) texput.dvi $*.dvi && $(MV) texput.size $*.size
        @-[ -f movie.cls.dvi ] && $(MV) movie.cls.dvi $*.dvi && $(MV) movie.cls.size $*.size
        @-[ -f movie.dvi ] && $(MV) movie.dvi $*.dvi && $(MV) movie.size $*.size
        ./dvitogif89a $*.dvi
#
.texmovie.dvi:
        $(LATEX) '\documentclass{article}' \
                '\usepackage[movie,all]{xy}' \
                '\begin{document}' \
                '\title{Storyboard for ``$*.''}' \
                '\maketitle' \
                '\input{$*.texmovie}' \
                '\end{document}'
#
dragon.gif:     dragon.texmovie movie.cls dvitogif89a
dragon.dvi:     dragon.texmovie movie.cls
#
```

```
sesam.gif:      sesam.texmovie movie.cls dvitogif89a
sesam.dvi:      sesam.texmovie movie.cls
#
gasket.dvi:     gasket.texmovie movie.cls dvitogif89a
gasket.gif:     gasket.texmovie movie.cls

# DISTRIBUTION TREE.
#
DISTDIR = $(FTPROOT)/xy-$(VERSION)
DISTFONTDIR = $(DISTDIR)/$(PK)fonts/$${mode}$${dpi}
#
# Files:
#
SOURCES = $(MACROSOURCES) $(FONTSOURCES) $(MANUALSOURCES) $(SUPPORT)
#
# Commands:
#
# TAR <files>           output tar archive with <files>
# GZIP                  compression filter
# GZ                    filename extension appropriate for compressed files
#
# TODO: change compression to proper GNU gzip once everyone supports it...
#
TAR = tar cvf -
GZIP = gzip -v9
GZ = .gz
#
# Targets:
#
.PHONY: dist dist-$(VERSION) distsrc distrun distdoc
#
dist dist-$(VERSION):
        cd $(DISTDIR);\
          find . -type f -print | sed 's,^./,,' | sort > MANIFEST
        $(INSTALL) README $(DISTDIR)/README
#
distsrc:        $(DISTDIR)/src
#
distrun:        $(DISTDIR)/CATALOG \
                $(DISTDIR)/TRAILER \
                $(DISTDIR)/README \
                $(DISTDIR)/INSTALL \
                $(DISTDIR)/VERSIONS \
                $(DISTDIR)/texinputs \
                $(DISTDIR)/texfonts \
                $(DISTDIR)/mfinputs \
                $(DISTDIR)/$(PK)fonts \
                $(DISTDIR)/doc/xyguide.dvi \
                $(DISTDIR)/doc/xyrefer.dvi \
                $(DISTDIR)/doc/xysource.dvi \
```

```
		$(DISTDIR)/ps \
		$(DISTDIR)/type1 \
		$(DISTDIR)/pfm \
		$(DISTDIR)/afm \
		$(DISTDIR)/COPYING \
		$(DISTDIR)/support
#
distdoc:	$(DISTDIR)/doc/xyguide.dvips \
		$(DISTDIR)/doc/xyguide.ps \
		$(DISTDIR)/doc/xyguide.pdf \
		$(DISTDIR)/doc/xyrefer.dvips \
		$(DISTDIR)/doc/xyrefer.ps \
		$(DISTDIR)/doc/xyrefer.pdf \
		$(DISTDIR)/doc/xysource.dvips \
		$(DISTDIR)/doc/xysource.ps
#### CURRENTLY BROKEN:
#		$(DISTDIR)/doc/xyguide-html \
#
# Explanation targets:
#
# CATALOG is automatically generated:
#
$(DISTDIR)/CATALOG: CATALOG $(DISTDIR)
	$(INSTALL) CATALOG $(DISTDIR)/CATALOG
#
CATALOG:	CATALOG.doc
	$(LATEX) CATALOG.doc
	mv -f CATALOG.out CATALOG

$(DISTDIR)/TRAILER: TRAILER $(DISTDIR)
	$(INSTALL) TRAILER $(DISTDIR)/TRAILER

$(DISTDIR)/README: README $(DISTDIR)
	$(INSTALL) README $(DISTDIR)/README

$(DISTDIR)/VERSIONS: VERSIONS $(DISTDIR)
	$(INSTALL) VERSIONS $(DISTDIR)/VERSIONS

$(DISTDIR)/INSTALL: $(INSTALLSTAR) $(DISTDIR)
	for f in $(INSTALLSTAR); do if test -f "$$f"; then \
	  $(INSTALL) $$f $(DISTDIR)/$$f; else :; fi; done

$(DISTDIR)/COPYING: COPYING $(DISTDIR)
	$(INSTALL) COPYING $(DISTDIR)/COPYING

$(DISTDIR)/FONTCOPYING: FONTCOPYING $(DISTDIR)
	$(INSTALL) FONTCOPYING $(DISTDIR)/FONTCOPYING
#
# Directory targets:
#
$(DISTDIR):
	-test -d $(DISTDIR) || $(MKDIR) $(DISTDIR)

$(DISTDIR)/src: $(SOURCES) MANIFEST VERSIONS ChangeLog $(DISTDIR)
```

```
	-test -d $(DISTDIR)/src || $(MKDIR) $(DISTDIR)/src
	for f in $(SOURCES); do echo $$f; done | sort -fu |\
	 while read u; do $(INSTALL) $$u $(DISTDIR)/src/$$u; done
	echo '' > xydoc.back
	$(INSTALLW) xydoc.back $(DISTDIR)/src
	$(INSTALLW) MANIFEST VERSIONS ChangeLog $(DISTDIR)/src

xydoc.back:
	echo '' > xydoc.back
#
# The MANIFEST in src is just a list of the distributed source files.
#
MANIFEST:	$(SOURCES)
	ls -alg `for f in $(SOURCES); do echo $$f; done | sort -fu` >MANIFEST
#
# The VERSIONS is just a list of the RCS versions.
#
VERSIONS:	$(SOURCES)
	sed -n 's/^.*\([$$]Id[:][^$$]*[$$]\).*$$/\1/p' \
	 `for f in $(SOURCES); do echo $$f; done | sort -u` >VERSIONS

$(DISTDIR)/texinputs: macros $(DISTDIR)
	-test -d $(DISTDIR)/texinputs \
	 || $(MKDIR) $(DISTDIR)/texinputs
	for f in $(MACRORUN) $(LATEXSTYLES); \
	 do $(INSTALL) $$f $(DISTDIR)/texinputs/$$f; done

$(DISTDIR)/texfonts: texfonts obsoletefonts $(DISTDIR)
	-test -d $(DISTDIR)/texfonts \
	 || $(MKDIR) $(DISTDIR)/texfonts
	for f in $(TEXFONTRUN) $(OBSOLETETEXFONTRUN); \
	 do $(INSTALL) $$f $(DISTDIR)/texfonts/$$f; done

$(DISTDIR)/mfinputs: mfinputs $(OBSOLETEMETAFONTS) $(DISTDIR)
	-test -d $(DISTDIR)/mfinputs || $(MKDIR) $(DISTDIR)/mfinputs
	for f in $(METAFONTS) $(OBSOLETEMETAFONTS); \
	 do $(INSTALL) $$f $(DISTDIR)/mfinputs/$$f; done

$(DISTDIR)/$(PK)fonts: $(PK)fonts $(DISTDIR)
	-test -d $(DISTDIR)/$(PK)fonts || $(MKDIR) $(DISTDIR)/$(PK)fonts
	:
	for mode in $(MODES); do for mag in $(MAGS); do \
	  $(RM) xytest.*gf xytest.*tfm;\
	  $(MF) "\\mode=$$mode; mag=magstep($$mag); input xytest";\
	  dpi="`expr xytest.*gf : 'xytest\.\([0-9]*\)gf'`";\
	  for f in $(TEXFONTRUN); do \
	    f="`expr $$f : '\([a-z0-9]*\)\.tfm'`.$${dpi}$(PK)" ;\
	    if [ ! -f $$f ]; then echo "ERROR: FONT $$f NOT GENERATED";\
	    else test -d "$(DISTFONTDIR)" || $(MKDIR) $(DISTFONTDIR);\
	      $(INSTALL) $$f $(DISTFONTDIR)/$$f; fi; done; done; done

$(DISTDIR)/ps: xy$(SHORTVERSION)dict.pro xypic.map $(DISTDIR)
	-test -d $(DISTDIR)/ps || $(MKDIR) $(DISTDIR)/ps
```

```
	$(INSTALL) xy$(SHORTVERSION)*dict.pro $(DISTDIR)/ps
	$(INSTALL) xypic.map $(DISTDIR)/ps

$(DISTDIR)/type1: $(TYPE1FONTS) $(DISTDIR)
	-test -d $(DISTDIR)/type1 || $(MKDIR) $(DISTDIR)/type1
	for f in $(TYPE1FONTS) ; \
	 do $(INSTALL) $$f $(DISTDIR)/type1/$$f; done

$(DISTDIR)/pfm: $(PFMFONTS) $(DISTDIR)
	-test -d $(DISTDIR)/pfm || $(MKDIR) $(DISTDIR)/pfm
	for f in $(PFMFONTS) ; \
	 do $(INSTALL) $$f $(DISTDIR)/pfm/$$f; done

$(DISTDIR)/afm: $(AFMFONTS) $(DISTDIR)
	-test -d $(DISTDIR)/afm || $(MKDIR) $(DISTDIR)/afm
	for f in $(AFMFONTS) ; \
	 do $(INSTALL) $$f $(DISTDIR)/afm/$$f; done

$(DISTDIR)/support: $(SUPPORT)
	-test -d $(DISTDIR)/support || $(MKDIR) $(DISTDIR)/support
	for f in $(SUPPORT) ; \
	 do $(INSTALLX) $$f $(DISTDIR)/support/$$f; done

# Document targets:
#
$(DISTDIR)/doc/xyguide.dvi:	xyguide.dvi $(DISTDIR)/doc
	$(INSTALL) xyguide.dvi $(DISTDIR)/doc

$(DISTDIR)/doc/xyguide.dvips:	xyguide.dvips $(DISTDIR)/doc
	$(INSTALL) xyguide.dvips $(DISTDIR)/doc

$(DISTDIR)/doc/xyguide.pdf:	xyguide.pdf $(DISTDIR)/doc
	$(INSTALL) xyguide.pdf $(DISTDIR)/doc

$(DISTDIR)/doc/xyguide.ps:	xyguide.ps $(DISTDIR)/doc
	$(INSTALL) xyguide.ps $(DISTDIR)/doc

$(DISTDIR)/doc/xyguide-html:	xyguide-html $(DISTDIR)/doc
	tar cf - xyguide-html | (cd $(DISTDIR)/doc; tar xpvf -)

$(DISTDIR)/doc/xyrefer.dvi:	xyrefer.dvi $(DISTDIR)/doc
	$(INSTALL) xyrefer.dvi $(DISTDIR)/doc

$(DISTDIR)/doc/xyrefer.dvips:	xyrefer.dvips ross.eps kris.eps import1.eps \
				$(DISTDIR)/doc
	$(INSTALL) xyrefer.dvips ross.eps kris.eps import1.eps $(DISTDIR)/doc

$(DISTDIR)/doc/xyrefer.pdf:	xyrefer.pdf ross.eps kris.eps import1.eps \
				$(DISTDIR)/doc
	$(INSTALL) xyrefer.pdf $(DISTDIR)/doc

$(DISTDIR)/doc/xyrefer.ps:	xyrefer.ps $(DISTDIR)/doc
	$(INSTALL) xyrefer.ps $(DISTDIR)/doc

$(DISTDIR)/doc/xyrefer-html:	xyrefer-html $(DISTDIR)/doc
	$(INSTALL) xyrefer-html $(DISTDIR)/doc

$(DISTDIR)/doc/xysource.dvi:	xysource.dvi $(DISTDIR)/doc
	$(INSTALL) xysource.dvi $(DISTDIR)/doc
```

```
$(DISTDIR)/doc/xysource.dvips:  xysource.dvips ross.eps kris.eps import1.eps \
                                $(DISTDIR)/doc
        $(INSTALL) xysource.dvips ross.eps kris.eps import1.eps $(DISTDIR)/doc

$(DISTDIR)/doc/xysource.ps:     xysource.ps $(DISTDIR)/doc
        $(INSTALL) xysource.ps $(DISTDIR)/doc

$(DISTDIR)/doc:
        -test -d $(DISTDIR)/doc || $(MKDIR) $(DISTDIR)/doc


# FTP'ABLE DISTRIBUTION.
#
FTPROOT = $$HOME/public_ftp/TeX
FTPDOCDIR = $(FTPROOT)/xy-$(VERSION)/doc
#
.PHONY:         ftp ftp-$(VERSION) ftpsrc ftprun ftpdoc ftpunpacked
#
ftp ftp-$(VERSION):     ftpsrc ftprun ftpdoc ftpunpacked
        $(INSTALL) TRAILER $(FTPROOT)/xy$(SHORTVERSION).TRAILER
#
ftpsrc $(FTPROOT)/xy$(SHORTVERSION)src.tar$(GZ): distsrc
        cd $(DISTDIR)/.. ;\
         $(TAR) xy-$(VERSION)/src \
         | $(GZIP) > $(FTPROOT)/xy$(SHORTVERSION)src.tar$(GZ)
#
ftpdoc $(FTPROOT)/xy$(SHORTVERSION)doc.tar$(GZ): distrun distdoc
        cd $(DISTDIR)/.. ;\
         $(TAR) xy-$(VERSION)/doc \
         | $(GZIP) > $(FTPROOT)/xy$(SHORTVERSION)doc.tar$(GZ)
        $(GZIP) < xyguide.dvi > $(FTPROOT)/xyguide.dvi$(GZ)
        $(GZIP) < xyguide.dvips > $(FTPROOT)/xyguide.dvips$(GZ)
        $(GZIP) < xyguide.ps    > $(FTPROOT)/xyguide.ps$(GZ)
        $(GZIP) < xyguide.pdf    > $(FTPROOT)/xyguide.pdf$(GZ)
        $(GZIP) < xyrefer.dvi > $(FTPROOT)/xyrefer.dvi$(GZ)
        $(GZIP) < xyrefer.dvips > $(FTPROOT)/xyrefer.dvips$(GZ)
        $(GZIP) < xyrefer.ps    > $(FTPROOT)/xyrefer.ps$(GZ)
        $(GZIP) < xyrefer.pdf    > $(FTPROOT)/xyrefer.pdf$(GZ)
        $(GZIP) < xysource.dvi > $(FTPROOT)/xysource.dvi$(GZ)
        $(GZIP) < xysource.dvips > $(FTPROOT)/xysource.dvips$(GZ)
        $(GZIP) < xysource.ps    > $(FTPROOT)/xysource.ps$(GZ)
#       $(TAR) xyguide-html | $(GZIP) > $(FTPROOT)/xyguide-html.tar.gz
#
ftprun $(FTPROOT)/xy$(SHORTVERSION)run.tar$(GZ): distrun
        cd $(DISTDIR)/.. ;\
         $(TAR) \
          `for f in $(TRAILERRUN); do echo xy-$(VERSION)/$$f; done` \
          xy-$(VERSION)/texinputs xy-$(VERSION)/mfinputs \
          xy-$(VERSION)/texfonts xy-$(VERSION)/$(PK)fonts \
          xy-$(VERSION)/ps xy-$(VERSION)/type1 xy-$(VERSION)/pfm \
          xy-$(VERSION)/afm \
          xy-$(VERSION)/support \
```

```
         | $(GZIP) > $(FTPROOT)/xy$(SHORTVERSION)run.tar$(GZ)
#
ftpunpacked: dist
        test $(DISTDIR) = $(FTPROOT)/xy-$(VERSION) || \
         (cd $(DISTDIR)/.. ; tar cf - xy-$(VERSION) |(cd $(FTPROOT); tar xvf -))
        cd $(FTPROOT); rm -f xy; ln -fs xy-$(VERSION) xy

# WWW/HTML DOCUMENTATION & HYPE.
#
WWWDIR = $$HOME/public_html
#
WWWSOURCES =    Xy-pic.html Xy.doc Xy.xy \
                man2html.gawk man2html-post.gawk .latex2html-init
#
.PHONY:         guide.html www install.www
#
LATEX2HTML =    latex2html
#
LATEX2HTMLICONDIR = $$HOME/src/latex2html-96.1/icons
LATEX2HTMLICONDIR2 = $$HOME/CV
#
man2html =      $(GAWK) -f man2html.gawk
man2htmlpost =  $(GAWK) -f ../man2html-post.gawk
#
guide.html:     xyguide-html
#
xyguide-html.tex: xyguide.dvips xyguide.man $(WWWSOURCES)
        $(man2html) <xyguide.aux >xyguide-html.aux
        $(man2html) <xyguide.bbl >xyguide-html.bbl
        @-$(CP) xyguide.man.ans xyguide.ans
        @-$(CP) xyguide.man.idx xyguide.idx
        @-$(CP) xyguide.man.ind xyguide.ind
        $(man2html) <xyguide.ans >xyguide-html.ans
        $(man2html) <xyguide.idx >xyguide-html.idx
        $(man2html) <xyguide.ind | sed 's/"/|/g' >xyguide-html.ind
        $(man2html) <xyguide.man >xyguide-html.tex
#
xyguide-html-act:       xyguide-html.tex version.tex
        echo "\\xyoption{dvips}\xyoption{ps}" > xydoc.back
        $(LATEX2HTML) xyguide-html.tex
#
xyguide-html:   xyguide-html.tex
        (cd xyguide-html; \
         for f in *.html; do mv $$f $$f~; \
          $(man2htmlpost) <$$f~ \
          | sed -e 's/``/"/g' -e "s/''/\"/g" -e 's/  [>] />/g' >$$f; done; \
         (cd $$HOME/W3; tar cf - COPYING) | tar xpf - )
        :
        cp "$(LATEX2HTMLICONDIR2)/18pt.html" xyguide-html
        cat xyguide-html/*.html \
```

```
        | tr '"' "\n" \
        | sed -n 's;[.][/]\([a-z_]*[.]gif\);\1;p' | sort -u \
        | while read f ;\
          do if [ -f "$(LATEX2HTMLICONDIR2)/$$f" ]; \
             then echo "[$$f]"; cp $(LATEX2HTMLICONDIR2)/$$f xyguide-html ; \
             elif [ -f "$(LATEX2HTMLICONDIR)/$$f" ]; \
             then echo "[$$f]"; cp $(LATEX2HTMLICONDIR)/$$f xyguide-html ; \
             fi ; done
        :
        (cd xyguide-html; \
         $(RM) .I* .O* *.ppm *.xyc *.ans *.aux *.idx *.bbl *.log *.tex *~)
#
Xy.tex: Xy.doc
Xy.dvi: Xy.tex Xy.xy macros; $(LATEX) Xy
Xy.ps:  Xy.dvi; $(DVIPS) Xy.dvi -oXy.ps
#
PNMCROP = ./pnmrawtopcropwhite | pnmcrop
PGMTOGIF = pgmtoppm rgbi:1/1/1 | ppmtogif -interlace -transparent rgbi:1/1/1
#
Xy.gif: Xy.ps pnmrawtopcropwhite
        echo '' | \
         gs -sDEVICE=pbmraw -r300 \
          -sOutputFile='|$(PNMCROP) >Xy.pbm' \
          -q - Xy.ps
        cat Xy.pbm | $(PGMTOGIF) >Xy.gif
#
pnmrawtopcropwhite: pnmrawtopcropwhite.c
#
www:            Xy-pic.html Xy.gif
## guide.html
#
install.www:    www
        $(INSTALL) Xy-pic.html Xy.gif $(WWWDIR)
#       $(RM) $(WWWDIR)/xyguide-html
#       $(MKDIR) $(WWWDIR)/xyguide-html
#       tar -cf - xyguide-html | (cd $(WWWDIR); tar xvpf -)

# ANNOUNCEMENTS.
#
MAILSOURCES = TRAILER Users mail.Users
#
.PHONY:         announce
announce:       $(MAILSOURCES)
        @./mail.Users

# TAGS for Emacs.
#
TAGS:   $(SOURCES)
        -$(RM) TAGS
        for f in `for ff in $(SOURCES); do echo $$ff; done |sort -u`;\
```

```
	do\
	 $(GAWK) '\
	  func err(s) { printf("%s,%d: %s\n",FILENAME,FNR,s) >>"/dev/stderr"};\
	  func prt(s,n) { printf("%s\177%d,%d\n",s,FNR,c+n) };\
	  func prtdef(i) { t = substr($$0,i);\
	   if (match(t,/[\\|][a-zA-Z@]*|\\.|[^\\|]|{([^{}]|\\{|\\})*}/) != 1) \
	    err("Weird TeX token");\
	   else prt(substr($$0,1,i+RLENGTH),i) };\
	  func prtcsdef(i) { t = substr($$0,i);\
	   if (match(t,/{([^{}]|\\{|\\})*}|{([^{}]|\\{|\\})*{([^{}]|\\{|\\})*}([^{}]|\\{|\\})*}/) != 1) \
	    err("Weird TeX cs name token");\
	   else prt(substr($$0,1,i+RLENGTH),i) };\
	  func fin() { c += length($$0) + 1; next };\
	  BEGIN { c = 0 };\
	  /^\\let[^A-Za-z@]/            ||\
	  /^\\def[^A-Za-z@]/            { prtdef(5); fin() };\
	  /^\\gdef[^A-Za-z@]/           ||\
	  /^\\edef[^A-Za-z@]/           ||\
	  /^\\xdef[^A-Za-z@]/           { prtdef(6); fin() };\
	  /^\\newif[^A-Za-z@]/          { prtdef(7); fin() };\
	  /^\\xydef@[^A-Za-z@]/         ||\
	  /^\\xylet@[^A-Za-z@]/         { prtdef(8); fin() };\
	  /^\\xyfont@[^A-Za-z@]/        { prtdef(9); fin() };\
	  /^\\global\\let[^A-Za-z@]/    { prtdef(12); fin() };\
	  /^\\xywarnifdefined[^A-Za-z@]/ { prtdef(17); fin() };\
	  /^\\ifx\\undefined[^A-Za-z@]/ { prtdef(15); fin() };\
	  /^\\xynew@{[^{}]*}/           { prtdef(index($$0,"}")+1); fin() };\
	  /^\\xydefcsname@{/            { prtcsdef(14); fin() };\
	  /^\\xyletcsnamecsname@{/      { prtcsdef(20); fin() };\
	  /\?\?\=\[/    { s = $$0; i = 0;\
	   while (first = index(s, "\?\?=[")) {\
	    s = substr(s,first+4); i += first+4;\
	    if (j = index(s, "]")) prt(substr($$0,1,i+j-1),i);\
	    else printf("Unmatched []s") }; fin() };\
	  /^@[A-Za-z]*{/ { prt($$0,index($$0,"{")+1); fin() };\
	  { fin() }' $$f >TAGS.tmp ;\
	 wc -c TAGS.tmp|$(GAWK) '{printf "\f\n%s,%s\n","'"$$f"'",$$1}'>>TAGS;\
	 cat TAGS.tmp >>TAGS ;\
	done
	$(RM) TAGS.tmp

# RCS STUFF:
#
.PHONY:		ci ci-$(VERSION) cici rcscheck \
		rcslist rcslockedlist rcsunlockedlist \
		rcscleanlist rcsuncleanlist rcsclean rcsdep
#
ALLSOURCES = $(TRAILERSOURCES) $(SOURCES) $(MANUALSOURCES) $(TeXNICALSOURCES) \
	$(WWWSOURCES) $(MAILSOURCES) $(OBSOLETEMETAFONTS) $(TYPE1FONTS) \
	$(SUPPORT) xypic.map
```

```
#
ROSSSOURCES = INSTALL.OzTeX INSTALL.Textures import1.eps ross.eps \
        xy16textures.doc xy17oztex.doc xy2cell.doc xyarc.doc \
        xyatip10.pfa xybsql10.pfa xybtip10.pfa xycirc10.pfa xycmat10.pfa \
        xycmbt10.pfa xyeuat10.pfa xyeubt10.pfa \
        xycolor.doc xycrayon.doc xycurve.doc xydash10.pfa \
        xydvips.doc xyimport.doc xyknot.doc xyline.doc \
        xymacpat.xyp xyoztex.doc xypoly.doc xyps-c.doc xyps-col.doc \
        xyps-f.doc xyps-l.doc xyps-pro.doc xyps-ps.doc xyps-r.doc \
        xyps-s.doc xyps-t.doc xyps.doc xypsdict.doc xypspatt.doc xyrotate.doc \
        xytextures.doc xytile.doc xytpic.doc xyemtex.doc xyweb.doc \
        xydvidrv.doc xyxdvi.doc xydvitops.doc xytp-f.doc xycmactex.doc
#
ci ci-$(VERSION): $(ALLSOURCES) cici Makefile
cici:
        -rcs -l `for f in $(ALLSOURCES); do echo $$f; done | sort -u`
        -ci -u$(VERSION) `$(MAKE) rcslockedlist`
#
rcscheck:       $(ALLSOURCES)
        @locked="`$(MAKE) rcslockedlist`"; \
         unlocked="`$(MAKE) rcsunlockedlist`"; \
         echo "Missing write permission on locked files:"; \
         for f in $$locked; do [ -w "$$f" ] || echo ' '$$f; done; \
         echo "Wrong write permission on unlocked files:"; \
         for f in $$unlocked; do [ -w "$$f" ] && echo ' '$$f; done; \
         echo "Wrong/missing locks on ross's files:"; \
         for f in $(ROSSSOURCES); \
         do lock=; \
            for ff in $$locked; \
            do if [ "$$f" = "$$ff" ]; then lock=' ross'; fi; done; \
            fgrep -s " ross Exp$$lock "'$$' $$f \
             ||(/bin/echo ' '$$f': \c' && grep 'Id[:].* Exp ' $$f || echo '?'); \
         done; \
         echo "Wrong/missing locks on krisrose's files:"; \
         for f in `$(MAKE) rcslist`; \
         do for ff in $(ROSSSOURCES); \
            do if [ "$$f" = "$$ff" ]; then continue 2; fi; done; \
            lock=; \
            for ff in $$locked; \
            do if [ "$$f" = "$$ff" ]; then lock=' krisrose'; fi; done; \
             fgrep -s " krisrose Exp$$lock "'$$' $$f \
             || (/bin/echo ' '$$f' \c' && grep 'Id[:].* Exp ' $$f || echo ''); \
         done
#
rcslist:
        -@rlog -R `for f in $(ALLSOURCES); do echo $$f; done | sort -u` \
          2>/dev/null | sed 's;RCS/\(.*\),v;\1;'
#
rcslockedlist:
```

```
	-@rlog -R -L `for f in $(ALLSOURCES); do echo $$f; done | sort -u` \
	  2>/dev/null | sed 's;RCS/\(.*\),v;\1;'
#
rcsunlockedlist:
	-@locked="`$(MAKE) rcslockedlist`"; \
	  $(MAKE) rcslist | while read f; \
	  do echo=echo; for ff in $$locked; \
			 do if [ "$$f" = "$$ff" ]; then echo=:; fi; done; \
	     $$echo $$f; done
#
rcscleanlist:
	-@$(MAKE) rcslist \
	 | while read f; do if test -w "$$f" -o "$$f" = Makefile; then :; \
			     else echo $$f; fi; done
#
rcsuncleanlist:
	-@$(MAKE) rcslist \
	 | while read f; do if test -w "$$f" -o "$$f" = Makefile; \
			     then echo $$f; fi; done
#
rcsclean:
	-@$(MAKE) rcscleanlist | while read f; do $(RM) $$f; done
#
# Dependencies...
#
rcsdep:
	$(MAKE) rcslist | while read f; do \
	  echo "$$f: RCS/$$f,v; test -d RCS && co $$f || :"; done

.late2html-init:; test -d RCS -a -f RCS/.late2html-init,v && co .late2html-init || :
CATALOG.doc:; test -d RCS -a -f RCS/CATALOG.doc,v && co CATALOG.doc || :
COPYING.patch:; test -d RCS -a -f RCS/COPYING.patch,v && co COPYING.patch || :
COPYING:; test -d RCS -a -f RCS/COPYING,v && co COPYING || :
FONTCOPYING.patch:; test -d RCS -a -f RCS/FONTCOPYING.patch,v && co FONTCOPYING.patch || :
FONTCOPYING:; test -d RCS -a -f RCS/FONTCOPYING,v && co FONTCOPYING || :
INSTALL.OzTeX:; test -d RCS -a -f RCS/INSTALL.OzTeX,v && co INSTALL.OzTeX || :
INSTALL.Textures:; test -d RCS -a -f RCS/INSTALL.Textures,v && co INSTALL.Textures || :
INSTALL:; test -d RCS -a -f RCS/INSTALL,v && co INSTALL || :
MAKE:; test -d RCS -a -f RCS/MAKE,v && co MAKE || :
Makefile:; test -d RCS -a -f RCS/Makefile,v && co Makefile || :
README:; test -d RCS -a -f RCS/README,v && co README || :
TRAILER:; test -d RCS -a -f RCS/TRAILER,v && co TRAILER || :
Users:; test -d RCS -a -f RCS/Users,v && co Users || :
Xy-pic.blurb:; test -d RCS -a -f RCS/Xy-pic.blurb,v && co Xy-pic.blurb || :
Xy-pic.html:; test -d RCS -a -f RCS/Xy-pic.html,v && co Xy-pic.html || :
Xy-pic.xbm:; test -d RCS -a -f RCS/Xy-pic.xbm,v && co Xy-pic.xbm || :
Xy.doc:; test -d RCS -a -f RCS/Xy.doc,v && co Xy.doc || :
Xy.xy:; test -d RCS -a -f RCS/Xy.xy,v && co Xy.xy || :
YandY_xy_fonts.zip:; test -d RCS -a -f RCS/YandY_xy_fonts.zip,v && co YandY_xy_fonts.zip || :
doc2tex.com:; test -d RCS -a -f RCS/doc2tex.com,v && co doc2tex.com || :
```

```
dvitogif89a:; test -d RCS -a -f RCS/dvitogif89a,v && co dvitogif89a || :
gasket.texmovie:; test -d RCS -a -f RCS/gasket.texmovie,v && co gasket.texmovie || :
import1.eps:; test -d RCS -a -f RCS/import1.eps,v && co import1.eps || :
install-tds:; test -d RCS -a -f RCS/install-tds,v && co install-tds || :
kris.eps:; test -d RCS -a -f RCS/kris.eps,v && co kris.eps || :
latin1.sty:; test -d RCS -a -f RCS/latin1.sty,v && co latin1.sty || :
mail.Users:; test -d RCS -a -f RCS/mail.Users,v && co mail.Users || :
man2html-post.gawk:; test -d RCS -a -f RCS/man2html-post.gawk,v && co man2html-post.gawk || :
man2html.gawk:; test -d RCS -a -f RCS/man2html.gawk,v && co man2html.gawk || :
movie.cls:; test -d RCS -a -f RCS/movie.cls,v && co movie.cls || :
pnmrawtopcropwhite.c:; test -d RCS -a -f RCS/pnmrawtopcropwhite.c,v && co pnmrawtopcropwhite.c || :
ross.eps:; test -d RCS -a -f RCS/ross.eps,v && co ross.eps || :
version.doc:; test -d RCS -a -f RCS/version.doc,v && co version.doc || :
xy.doc:; test -d RCS -a -f RCS/xy.doc,v && co xy.doc || :
xy.sty:; test -d RCS -a -f RCS/xy.sty,v && co xy.sty || :
xy16textures.doc:; test -d RCS -a -f RCS/xy16textures.doc,v && co xy16textures.doc || :
xy17oztex.doc:; test -d RCS -a -f RCS/xy17oztex.doc,v && co xy17oztex.doc || :
xy2cell.doc:; test -d RCS -a -f RCS/xy2cell.doc,v && co xy2cell.doc || :
xyall.doc:; test -d RCS -a -f RCS/xyall.doc,v && co xyall.doc || :
xyamslatex.ini:; test -d RCS -a -f RCS/xyamslatex.ini,v && co xyamslatex.ini || :
xyarc.doc:; test -d RCS -a -f RCS/xyarc.doc,v && co xyarc.doc || :
xyarrow.doc:; test -d RCS -a -f RCS/xyarrow.doc,v && co xyarrow.doc || :
xyatip10.mf:; test -d RCS -a -f RCS/xyatip10.mf,v && co xyatip10.mf || :
xybsql10.mf:; test -d RCS -a -f RCS/xybsql10.mf,v && co xybsql10.mf || :
xybtip10.mf:; test -d RCS -a -f RCS/xybtip10.mf,v && co xybtip10.mf || :
xycirc10.mf:; test -d RCS -a -f RCS/xycirc10.mf,v && co xycirc10.mf || :
xycm.mf:; test -d RCS -a -f RCS/xycm.mf,v && co xycm.mf || :
xycmactex.doc:; test -d RCS -a -f RCS/xycmactex.doc,v && co xycmactex.doc || :
xycmat10.mf:; test -d RCS -a -f RCS/xycmat10.mf,v && co xycmat10.mf || :
xycmat11.mf:; test -d RCS -a -f RCS/xycmat11.mf,v && co xycmat11.mf || :
xycmat12.mf:; test -d RCS -a -f RCS/xycmat12.mf,v && co xycmat12.mf || :
xycmbt10.mf:; test -d RCS -a -f RCS/xycmbt10.mf,v && co xycmbt10.mf || :
xycmbt11.mf:; test -d RCS -a -f RCS/xycmbt11.mf,v && co xycmbt11.mf || :
xycmbt12.mf:; test -d RCS -a -f RCS/xycmbt12.mf,v && co xycmbt12.mf || :
xycmtip.doc:; test -d RCS -a -f RCS/xycmtip.doc,v && co xycmtip.doc || :
xycolor.doc:; test -d RCS -a -f RCS/xycolor.doc,v && co xycolor.doc || :
xycrayon.doc:; test -d RCS -a -f RCS/xycrayon.doc,v && co xycrayon.doc || :
xycurve.doc:; test -d RCS -a -f RCS/xycurve.doc,v && co xycurve.doc || :
xyd.mf:; test -d RCS -a -f RCS/xyd.mf,v && co xyd.mf || :
xyd2.mf:; test -d RCS -a -f RCS/xyd2.mf,v && co xyd2.mf || :
xydash10.mf:; test -d RCS -a -f RCS/xydash10.mf,v && co xydash10.mf || :
xydoc.bib:; test -d RCS -a -f RCS/xydoc.bib,v && co xydoc.bib || :
xydoc.sty:; test -d RCS -a -f RCS/xydoc.sty,v && co xydoc.sty || :
xydocps.doc:; test -d RCS -a -f RCS/xydocps.doc,v && co xydocps.doc || :
xydriver.doc:; test -d RCS -a -f RCS/xydriver.doc,v && co xydriver.doc || :
xydummy.doc:; test -d RCS -a -f RCS/xydummy.doc,v && co xydummy.doc || :
xydvidrv.doc:; test -d RCS -a -f RCS/xydvidrv.doc,v && co xydvidrv.doc || :
xydvips.doc:; test -d RCS -a -f RCS/xydvips.doc,v && co xydvips.doc || :
xydvitops.doc:; test -d RCS -a -f RCS/xydvitops.doc,v && co xydvitops.doc || :
```

```
xyemtex.doc:; test -d RCS -a -f RCS/xyemtex.doc,v && co xyemtex.doc || :
xyeuat10.mf:; test -d RCS -a -f RCS/xyeuat10.mf,v && co xyeuat10.mf || :
xyeuat11.mf:; test -d RCS -a -f RCS/xyeuat11.mf,v && co xyeuat11.mf || :
xyeuat12.mf:; test -d RCS -a -f RCS/xyeuat12.mf,v && co xyeuat12.mf || :
xyeubt10.mf:; test -d RCS -a -f RCS/xyeubt10.mf,v && co xyeubt10.mf || :
xyeubt11.mf:; test -d RCS -a -f RCS/xyeubt11.mf,v && co xyeubt11.mf || :
xyeubt12.mf:; test -d RCS -a -f RCS/xyeubt12.mf,v && co xyeubt12.mf || :
xyeuler.doc:; test -d RCS -a -f RCS/xyeuler.doc,v && co xyeuler.doc || :
xyframe.doc:; test -d RCS -a -f RCS/xyframe.doc,v && co xyframe.doc || :
xygraph.doc:; test -d RCS -a -f RCS/xygraph.doc,v && co xygraph.doc || :
xyguide.man:; test -d RCS -a -f RCS/xyguide.man,v && co xyguide.man || :
xyidioms.doc:; test -d RCS -a -f RCS/xyidioms.doc,v && co xyidioms.doc || :
xyimport.doc:; test -d RCS -a -f RCS/xyimport.doc,v && co xyimport.doc || :
xyknot.doc:; test -d RCS -a -f RCS/xyknot.doc,v && co xyknot.doc || :
xylatex.ini:; test -d RCS -a -f RCS/xylatex.ini,v && co xylatex.ini || :
xyline.doc:; test -d RCS -a -f RCS/xyline.doc,v && co xyline.doc || :
xyline10.mf:; test -d RCS -a -f RCS/xyline10.mf,v && co xyline10.mf || :
xymacpat.xyp:; test -d RCS -a -f RCS/xymacpat.xyp,v && co xymacpat.xyp || :
xymatrix.doc:; test -d RCS -a -f RCS/xymatrix.doc,v && co xymatrix.doc || :
xymisc10.mf:; test -d RCS -a -f RCS/xymisc10.mf,v && co xymisc10.mf || :
xymovie.doc:; test -d RCS -a -f RCS/xymovie.doc,v && co xymovie.doc || :
xyoztex.doc:; test -d RCS -a -f RCS/xyoztex.doc,v && co xyoztex.doc || :
xypic.doc:; test -d RCS -a -f RCS/xypic.doc,v && co xypic.doc || :
xypic.map:; test -d RCS -a -f RCS/xypic.map,v && co xypic.map || :
xypic.sty:; test -d RCS -a -f RCS/xypic.sty,v && co xypic.sty || :
xypoly.doc:; test -d RCS -a -f RCS/xypoly.doc,v && co xypoly.doc || :
xyps-c.doc:; test -d RCS -a -f RCS/xyps-c.doc,v && co xyps-c.doc || :
xyps-col.doc:; test -d RCS -a -f RCS/xyps-col.doc,v && co xyps-col.doc || :
xyps-dto.doc:; test -d RCS -a -f RCS/xyps-dto.doc,v && co xyps-dto.doc || :
xyps-dvi.doc:; test -d RCS -a -f RCS/xyps-dvi.doc,v && co xyps-dvi.doc || :
xyps-f.doc:; test -d RCS -a -f RCS/xyps-f.doc,v && co xyps-f.doc || :
xyps-l.doc:; test -d RCS -a -f RCS/xyps-l.doc,v && co xyps-l.doc || :
xyps-one.doc:; test -d RCS -a -f RCS/xyps-one.doc,v && co xyps-one.doc || :
xyps-oz.doc:; test -d RCS -a -f RCS/xyps-oz.doc,v && co xyps-oz.doc || :
xyps-pro.doc:; test -d RCS -a -f RCS/xyps-pro.doc,v && co xyps-pro.doc || :
xyps-ps.doc:; test -d RCS -a -f RCS/xyps-ps.doc,v && co xyps-ps.doc || :
xyps-pub.doc:; test -d RCS -a -f RCS/xyps-pub.doc,v && co xyps-pub.doc || :
xyps-r.doc:; test -d RCS -a -f RCS/xyps-r.doc,v && co xyps-r.doc || :
xyps-s.doc:; test -d RCS -a -f RCS/xyps-s.doc,v && co xyps-s.doc || :
xyps-t.doc:; test -d RCS -a -f RCS/xyps-t.doc,v && co xyps-t.doc || :
xyps-txo.doc:; test -d RCS -a -f RCS/xyps-txo.doc,v && co xyps-txo.doc || :
xyps-txt.doc:; test -d RCS -a -f RCS/xyps-txt.doc,v && co xyps-txt.doc || :
xyps-wdo.doc:; test -d RCS -a -f RCS/xyps-wdo.doc,v && co xyps-wdo.doc || :
xyps.doc:; test -d RCS -a -f RCS/xyps.doc,v && co xyps.doc || :
xypsdict.doc:; test -d RCS -a -f RCS/xypsdict.doc,v && co xypsdict.doc || :
xypspatt.doc:; test -d RCS -a -f RCS/xypspatt.doc,v && co xypspatt.doc || :
xyqc10.mf:; test -d RCS -a -f RCS/xyqc10.mf,v && co xyqc10.mf || :
xyrecat.doc:; test -d RCS -a -f RCS/xyrecat.doc,v && co xyrecat.doc || :
xyrefer.man:; test -d RCS -a -f RCS/xyrefer.man,v && co xyrefer.man || :
```

```
xyrotate.doc:; test -d RCS -a -f RCS/xyrotate.doc,v && co xyrotate.doc || :
xysmart.doc:; test -d RCS -a -f RCS/xysmart.doc,v && co xysmart.doc || :
xysource.man:; test -d RCS -a -f RCS/xysource.man,v && co xysource.man || :
xystyle.doc:; test -d RCS -a -f RCS/xystyle.doc,v && co xystyle.doc || :
xytech.mf:; test -d RCS -a -f RCS/xytech.mf,v && co xytech.mf || :
xytest.mf:; test -d RCS -a -f RCS/xytest.mf,v && co xytest.mf || :
xytex.ini:; test -d RCS -a -f RCS/xytex.ini,v && co xytex.ini || :
xytextures.doc:; test -d RCS -a -f RCS/xytextures.doc,v && co xytextures.doc || :
xytile.doc:; test -d RCS -a -f RCS/xytile.doc,v && co xytile.doc || :
xytips.doc:; test -d RCS -a -f RCS/xytips.doc,v && co xytips.doc || :
xytp-f.doc:; test -d RCS -a -f RCS/xytp-f.doc,v && co xytp-f.doc || :
xytpic.doc:; test -d RCS -a -f RCS/xytpic.doc,v && co xytpic.doc || :
xyv2.doc:; test -d RCS -a -f RCS/xyv2.doc,v && co xyv2.doc || :
xyweb.doc:; test -d RCS -a -f RCS/xyweb.doc,v && co xyweb.doc || :
xyxdvi.doc:; test -d RCS -a -f RCS/xyxdvi.doc,v && co xyxdvi.doc || :

# DEVELOPMENT TARGETS:
#
.PHONY:		Install.macros Install Ftpsrc Ftpdoc Ftp World Minor

Install.macros:
	$(MAKE)		  USR=$$HOME/pub			install.macros

Install:
	$(MAKE)		  USR=$$HOME/pub			install

Ftpsrc:
	$(MAKE)		MODES='ljfour cx'		\
			 MAGS='0 0.5 1 2'		\
		      FTPROOT=$$HOME/public_ftp/TeX	ftpsrc

Ftpdoc:
	$(MAKE)	      MODES='ljfour cx'			\
		       MAGS='0 0.5 1 2'			\
		    FTPROOT=$$HOME/public_ftp/TeX	ftpdoc

Ftp:
	$(MAKE)	      MODES='ljfour cx'			\
		       MAGS='0 0.5 1 2'			\
		    FTPROOT=$$HOME/public_ftp/TeX	ftp

Www:
	$(MAKE)	    WWWROOT=$$HOME/public_html		install.www

World:
	$(MAKE)	      LATEX='time latex'		\
		      MODES='ljfour cx'			\
		       MAGS='0 0.5 1 2'			world
	mv make.world make.World

TMPFTPROOT = $$HOME/tmp/new-ftp-root

Minor:
	$(MKDIR) $(TMPFTPROOT)
	$(MAKE)	      LATEX='time latex'		\
```

```
                     MODES='ljfour cx'                    \
                      MAGS='0 0.5 1 2'                    \
                   FTPROOT=$(TMPFTPROOT)                  world
        mv make.world make.Minor
#
        @echo "Making patch:";\
         cd $(TMPFTPROOT); \
         diff -urN $(FTPROOT)/xy-$(VERSION) xy-$(VERSION) > FIX; \
         n=`cd $(FTPROOT);ls xy$(SHORTVERSION)fix* | wc -w 2>/dev/null`; \
         n=`expr $$n + 1`; \
         fix=$(FTPROOT)/xy$(SHORTVERSION)fix`printf '%.2d' $$n`; \
         cp -vf FIX $$fix; gzip -v9 $$fix
#
        @echo "Updating real ftp archive $(FTPROOT):";\
         cd $(TMPFTPROOT); \
         find xy* -type f -print \
         | while read f;\
           do cmp -s $$f $(FTPROOT)/$$f\
              || { echo $$f; $(MKDIR) `dirname $(FTPROOT)/$$f`;\
                   if [ -w $$f ]; then $(INSTALLW) $$f $(FTPROOT)/$$f;\
                   else                $(INSTALL)  $$f $(FTPROOT)/$$f;\
                   fi; };\
           done
        $(RM) $(TMPFTPROOT)

# LOG
#
# $Log: Makefile,v $
# Revision 3.6  1998/03/06 01:28:05  krisrose
# Releasing (with Y&Y fonts).
#
# Revision 3.5  1997/05/28 13:05:01  krisrose
# Fixed missing breaks bug.
#
# Revision 3.4  1997/05/18 03:04:44  krisrose
# Essential bugfixes.
# And movies now included!
#
# Revision 3.3  1996/12/19 17:05:13  krisrose
# Maintenance release!
#
# Revision 3.2  1995/09/19  18:22:27  kris
# Bug fix release.
#
# Revision 3.1  1995/09/05  20:31:32  kris
# Releasing!
#
# Revision 3.0  1995/07/07  20:14:21  kris
# Major release w/new User's Guide!
#
```

```
# Revision 2.14  1995/07/06  02:56:02  kris
# Buglets...
#
# Revision 2.13  1995/07/04  15:10:01  kris
# Ready to release v3?
#
# Revision 2.12  1994/10/25  11:46:25  kris
# Interim release just before v3 [works with AMS-LaTeX 1.2]...
#
# Revision 2.11  1994/07/05  10:37:32  kris
# Third 3beta release [bug fixes].
# Experimental graph feature included (for ECCT-94 presentation).
#
# Revision 2.10  1994/06/15  13:35:19  kris
# Second 3beta release [bug fixes].
#
# Revision 2.9  1994/06/09  14:53:07  kris
# Release 3beta.
#
# Revision 2.8  1994/04/08  04:30:00  kris
# Second (bug fix) 3alpha release.
#
# Revision 2.7  1994/03/08  02:06:01  kris
# Release 3alpha.
#
# Revision 2.6.9.1  1994/03/07  04:22:46  kris
# Last internal 3alpha and pre-2.7 release.
#
# MAJOR REORGANISATION for version 2.7...
#
# Revision 2.6  1992/06/24  01:23:34  kris
# Ready to release v.2.6.
#
# Revision 2.5  1992/02/24  03:30:54  kris
# Default magsteps now just 0, 0.5, 1, and 2.
#
# Revision 2.3  1992/01/13  02:12:28  kris
# Fixed installation instructions and other details.
#
# Revision 2.1  1992/01/02  14:54:07  kris
# Release version.
#
# Revision 1.6  1991/12/17  04:51:16  kris
# Version distributed with ‘final draft’ on Usenet.

# Tell Emacs that this is a Makefile and how it is formatted:
# Local Variables:
# mode:fundamental
# fill-prefix:"# "
# fill-column:77
```

```
# End:
```

Bibliography

[1] Adobe Systems Incorporated. *PostScript Language Reference Manual*, second edition, 1990.

[2] American Mathematical Society. *AMS-LaTeX Version 1.1 User's Guide*, 1.1 edition, 1991.

[3] Karl Berry. *Expanded plain TeX*, version 2.6 edition, May 1994. Available for anonymous ftp from CTAN in `macros/eplain/doc`.

[4] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, 1994.

[5] Brian W. Kernighan. PIC—a language for typesetting graphics. *Software Practice and Experience*, 12(1):1–21, 1982.

[6] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1984.

[7] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, 1986.

[8] Donald E. Knuth. *Computer Modern Typefaces*, volume A of *Computers & Typesetting*. Addison-Wesley, 1986.

[9] Leslie Lamport. *LaTeX—A Document Preparation System*. Addison-Wesley, 1986.

[10] Leslie Lamport. *LaTeX—A Document Preparation System*. Addison-Wesley, 2nd edition, 1994.

[11] Tomas Rokicki. *DVIPS: A TeX Driver*. Distributed with the dvips program found on CTAN archives.

[12] Kristoffer H. Rose. How to typeset pretty diagram arrows with TeX—design decisions used in XY-pic. In Jiří Zlatuška, editor, *EuroTeX '92—Proceedings of the 7th European TeX Conference*, pages 183–190, Prague, Czechoslovakia, September 1992. Czechoslovak TeX Users Group.

[13] Kristoffer H. Rose. Typesetting diagrams with XY-pic: User's manual. In Jiří Zlatuška, editor, *EuroTeX '92—Proceedings of the 7th European TeX Conference*, pages 273–292, Prague, Czechoslovakia, September 1992. Czechoslovak TeX Users Group.

[14] Kristoffer H. Rose. *XY-pic User's Guide*. DIKU, University of Copenhagen, Universitetsparken 1, DK–2100 København Ø, 3.0 edition, June 1995.

[15] Kristoffer H. Rose and Ross R. Moore. XY-pic complete sources with TeXnical commentary. not yet available, June 1995.

[16] Kristoffer H. Rose and Ross R. Moore. *XY-pic Reference Manual*. DIKU, University of Copenhagen, Universitetsparken 1, DK–2100 København Ø, 3.0 edition, June 1995.

[17] Michael D. Spivak. *The Joy of TeX—A Gourmet Guide to Typesetting with the AMS-TeX Macro Package*. American Mathematical Society, second edition, 1990.

[18] TUG Working Group TWG-TDS. A directory structure for TeX files version 0.98. URL, May 1995. Available with URL ftp://jasper.ora.com/pub/twg-tds/.

Index

!, 23
&, 342
', 295
(), 23
(0), 23
(0,0), 545
(1), 23
*, 23, 295, 306, 313, 343, 357, 500, 554
**, 23, 305, 357
*stylechar@P@, xynecula.doc:106
*stylechar@|@, xyline.doc:114
+, 23
,, 23, 302
-, 23, 295, 298
., 23
.xyd, 69
/, 295, 299, 304, 313
//, 23
:, 23, 46
::, 23
;, 23
<, 23, 295, 300, 305
<>, 23
<>(.5), 295, 298
=, 23, 295, 297, 299, 300, 305, 306
>, 23, 295, 300, 305
?, 23, 305
@, 23, 295, 354, 455
@!, 355, 455
@!0, 355
@!=, 355
@!C, 355
@!R, 355
@(, 41, 314, 455
@), 41
@*, 357, 455
@+, 41
@-, 41
@/, 314, 455
@1, 359
@<, 315, 455
@=, 41
@?, 315, 455
@@, 41
@C, 354
@H, 355
@L, 355
@M, 355
@R, 354
@W, 355
@`, 455
@i, 41
\[, xy.doc:2538
[.], 54
[=, 46, 59, 238, 245
[P:, 280
[], 54, 55
[c], 54
[d], 54
[dvips], 11
[l], 54
[o], 54
[r], 54
[textures], 11
[u], 54
\\, 342
^, 295, 301–303, 305, 455
_, 295, 301–303, 305, 455
`, 295, 301
`s, 445
{\styletoks@}, xy.doc:3894
{\xypatread@}, xytile.doc:264
|, 295, 455
|<<, 554
|DOCMODE, xy.doc:70
~, 295

0, 20, 23, 455
1, 455
2, 455
3, 455
10, 215
11, 215

12, 215
A, 8
\a, xy.doc:2062
a, 23
\A@, xyidioms.doc:85
\ABfromdiag@, xyarrow.doc:849
\aboveDirection@, xy.doc:7265
\activatedriver@, xy.doc:6705
\activatedriversupport@, xy.doc:6708
\activateEM@, xyemtex.doc:132
\activatetips@, xytips.doc:128
\activateTPIC@, xytpic.doc:222
active characters, 4
\ad, xy.doc:2063
⟨add op⟩, 46
\add@m@keys, movie.cls:95
\addAND@, xy.doc:296
\addAT@, xy.doc:293
\addbeforeDrop@@, xyframe.doc:113
\addDASH@, xy.doc:299
\addDOLL@, xy.doc:295
\addDOT@, xy.doc:300
\addEQ@, xy.doc:302
\addFalse@, xy.doc:318
\addGT@, xy.doc:303
\addHASH@, xy.doc:294
\addLQ@, xy.doc:304
\addLT@, xy.doc:301
\ADDOP@, xy.doc:3595
\Addop@, xy.doc:3631
\ADDOP@minus, xy.doc:3612
\ADDOP@plus, xy.doc:3605
\addPLUS@, xy.doc:298
\addq@, xymatrix.doc:183
\addq@@, xymatrix.doc:188
\addq@i, xymatrix.doc:190
\addq@ii, xymatrix.doc:193
\addRQ@, xy.doc:297
\addtoarinit@, xyarrow.doc:1264
\addtoarlabels@, xyarrow.doc:1267
\addtocrv@, xycurve.doc:1931
\addtocrvpts@, xycurve.doc:2032
\addtoDrop@@, xyframe.doc:111
\addtostyletoks@, xy.doc:3895
\addtotoks@, xy.doc:724
\addTrue@, xy.doc:319
\adjust@forsegments@, xycurve.doc:4114
\adjustBaseOrient@, xyarc.doc:1388
\adjustbsplineedges@, xycurve.doc:4132
\adjustLR@, xy.doc:3142
\adjustmaxmin@, xycurve.doc:3744
\adjustMinMaxcirc@, xycurve.doc:4479
\adjustUD@, xy.doc:3148
\afterADDOP, xy.doc:3591
\afterADDOP , 53
\afterADDOP@, xy.doc:3593
\afteraliases@@, xyarrow.doc:1058
\afterar@@, xyarrow.doc:1218
\AfterBeginDocument, xyps-ps.doc:175
\afterCIR@, xy.doc:6070
\afterCIRDIAG@, xy.doc:6071
\afterCIRorDIAG, xy.doc:6068
\afterCOORD, xy.doc:1377
\afterCOORD , 24
\afterCOORD@, xy.doc:1382
\afterCURVE, xycurve.doc:1978
\afterCURVEaction@, xycurve.doc:274
\afterDIAG, xy.doc:4297
\afterDIRECTION@, xy.doc:4239
\afterDIRECTIONorEMPTY, xy.doc:4233
\aftergo, xyv2.doc:266
\aftergo , 556
\afterMORPHISM, xy2cell.doc:161
\afterNODE, xygraph.doc:213
\afterNODE@@, xygraph.doc:211
\afternudge, xy2cell.doc:827
\afterPATH, xyarrow.doc:187
\afterPATH , 295
\afterPATH@, xyarrow.doc:185
\afterPLACE, xy.doc:2274
\afterPLACE@, xy.doc:2291
\afterPOS, xy.doc:1370
\afterPOS , 24, 65, 301
\afterPOS@, xy.doc:1375
\aftertwo@CELL, xy2cell.doc:1225
\aftertwoCELL, xy2cell.doc:1226
\afterVECTOR@, xy.doc:1391
\afterVECTORorEMPTY, xy.doc:1384
\afterVECTORorEMPTY , 24
\ahook, xyv2.doc:366
\ahook , 558
\ahook@, xy.doc:5730
\ahook@@, xy.doc:5731
\AliasPattern, xytile.doc:286
\aliaspattern@@, xytile.doc:316
allocation, 15
\almostz@, xy.doc:659
\alphanum, xypoly.doc:428

$\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX, 7, 67
$\mathcal{A}\mathcal{M}\mathcal{S}$-TeX, 7
\apit , 554
\appendtoholder@, xyknot.doc:1092
\appendtoholder@@, xyknot.doc:1096
\appendtoholder@@@, xyknot.doc:1100
\appendtoholer@, xyknot.doc:1115
\applyFIFOstyle@, xy.doc:4162
\applyLIFOstyle@, xy.doc:4190
\ar, xyarrow.doc:1226
\ar , 59, 299, 307, 445
\ar@, xyarrow.doc:1248
\ar@anchor, xyarrow.doc:1270
\ar@curve, xyarrow.doc:1736
\ar@curve@, xyarrow.doc:1747
\ar@curveinout, xyarrow.doc:1712
\ar@curveinout@, xyarrow.doc:1710
\ar@curveslash, xyarrow.doc:1673
\ar@error, xyarrow.doc:1324
\ar@form, xyarrow.doc:1287
\ar@i, xyarrow.doc:1346
\ar@ii, xyarrow.doc:1354
\ar@iii, xyarrow.doc:1359
\ar@iv, xyarrow.doc:1365
\ar@modifiers, xyarrow.doc:1781
\ar@PATH, xyarrow.doc:1406
\ar@slashing, xyarrow.doc:1675
\ar@slide, xyarrow.doc:1796
\ar@style, xyarrow.doc:1332
\ar@upsidedown, xyarrow.doc:1807
\ar@x, xyarrow.doc:1377
\arafterPOS@, xyarrow.doc:1408
\arafterPOS@@, xyarrow.doc:1211
\arc@, xyarc.doc:91
\arc@i, xyarc.doc:123
\arcbisector, xyarc.doc:1013
\arcCenter@, xyarc.doc:1020
\arcCenter@i, xyarc.doc:1026
\arcmagic, xyarc.doc:890
\arcmagic@@, xyarc.doc:889
\arcomponent@, xyarrow.doc:1424
\arcomponent@@, xyarrow.doc:1215
\arcomponent@i, xyarrow.doc:1438
\arcomponent@ii, xyarrow.doc:1440
\arcomponent@x, xyarrow.doc:1468
\arcomponenttype@@, xyarrow.doc:1216
\arconn@, xyarrow.doc:1457
\arcScale@@, xyarc.doc:252
\arcScale@dim, xyarc.doc:262
\ARCturn@, xyarc.doc:961
\ARCturn@@, xyarc.doc:964
\ARCturn@@@, xyarc.doc:970
\arexit@@, xyarrow.doc:1213
\arhead@@, xyarrow.doc:1208
\arinit@@, xyarrow.doc:1212
\arlabels@@, xyarrow.doc:1210
\armodifiers@@, xyarrow.doc:1209
array, 361
\arraystretch, xy.doc:6152
\arrow, xyv2.doc:599
arrow head, 312
arrow stem, 312
arrow tail, 312
\arrowobject, xy2cell.doc:207
\Arrowobject@, xy2cell.doc:208
\Arrowtok@, xy2cell.doc:1447
\arSAFE, xyarrow.doc:1228
\arsavedPATHafterPOS@@, xyarrow.doc:1412
\arstem@@, xyarrow.doc:1207
\arstemprefix@@, xyarrow.doc:1205
\artail@@, xyarrow.doc:1206
\artip@, xyarrow.doc:1442
\arvariant@@, xyarrow.doc:1204
\astop, xyv2.doc:349
\astop , 558
\AtBeginDvi, xyidioms.doc:175
\aTip, xyv2.doc:358
\atip, xyv2.doc:353
\aTip , 558
\atip , 558
\atip@, xy.doc:5621
\atip@@, xy.doc:5630
\atip@cm, xytips.doc:125
\atip@xy, xytips.doc:49
\aturn, xyv2.doc:369
\aturn , 558
\aturn@, xy.doc:5760
\aturn@@, xy.doc:5761
\AveryUNLIKELYc@ntr@lSEQUENCE@@, xyidioms.doc:98

B, 8
\B@, xyidioms.doc:86
\backwardSectors@, xyarc.doc:1255
banner, 13
\basefromthebase@, xy.doc:1879
\baseOrient@, xyarc.doc:325
\bblacked@, xyframe.doc:469
\belowDirection@, xy.doc:7282
\bendPLACE@, xycurve.doc:1378

\bhook, xyv2.doc:367
\bhook , 558
\bhook@, xy.doc:5738
\bhook@@, xy.doc:5739
\blacked@, xyframe.doc:461
\blacked@@, xyframe.doc:474
\blacktriangle, xypoly.doc:587
⟨body of the option⟩, 91
\boxz@, xyidioms.doc:114
\bpit , 554
\bprevPLACE@, xycurve.doc:1379
\break@end, xycurve.doc:2933
\break@start, xycurve.doc:2869
\breakendcontinue@, xycurve.doc:2888
\breakendcontinue@i, xycurve.doc:2895
\breakendfound@, xycurve.doc:2923
\breakstartcontinue@, xycurve.doc:2802
\breakstartcontinue@i, xycurve.doc:2808
\breakstartfound@, xycurve.doc:2821
\bsegment@@i, xycurve.doc:4194
\bsegment@@ii@i, xycurve.doc:4227
\bsegment@@ii@iii, xycurve.doc:4178
\bsegment@@iv, xycurve.doc:4204
\bsegment@@y@iv, xycurve.doc:4237
\bsegment@i, xycurve.doc:4260
\bsegment@i@iv, xycurve.doc:4276
\bsegment@z, xycurve.doc:4298
\bsegment@z@i, xycurve.doc:4314
\bsplinecancel@, xycurve.doc:3452
\bsplineconnect@, xycurve.doc:4012
\bsplined@, xycurve.doc:4110
\bsplined@@, xycurve.doc:4002
\bsplineiiiconnect@, xycurve.doc:3960
\bsplineivconnect@, xycurve.doc:3968
\bsplinesegment@, xycurve.doc:4328
\bsplinesegment@iii, xycurve.doc:4351
\bsplinesegment@iv, xycurve.doc:4377
\bstartPLACE@, xycurve.doc:1377
\bstop, xyv2.doc:350
\bstop , 558
\bthisPLACE@, xycurve.doc:1380
\bTip, xyv2.doc:359
\btip, xyv2.doc:354
\bTip , 558
\btip , 558
\btip@, xy.doc:5622
\btip@@, xy.doc:5631
\btip@cm, xytips.doc:126
\btip@xy, xytips.doc:50
\bturn, xyv2.doc:370
\bturn , 558
\bturn@, xy.doc:5766
\bturn@@, xy.doc:5767
\buildcircle@, xycurve.doc:4604

C, 23, 40
c, 21
c, 23
\c@poly@count, xynecula.doc:107
\Calong@@, xy.doc:7930
\cancelPSdict@, xyps-ps.doc:485
cartesian coordinate system, 21
category code, 4
\Cbreak@@, xy.doc:7932
\CCintercept@@, xycurve.doc:1664
CD, 23
\cD, xyv2.doc:199
\cD , 556
\centerobject@, xy.doc:3731
\cfromcontrols@, xyarc.doc:508
\cfromid@, xy.doc:2896
\cfromp@, xy.doc:1629
\cfroms@, xy.doc:2681
\cfromthec@, xy.doc:1660
\cfromthep@, xy.doc:1664
\cH, xyv2.doc:200
\cH , 556
\change@oxy@, xy.doc:860
\changedriver@single, xy.doc:6690
\checkafterCURVE@, xycurve.doc:710
\checkcrvtilde@, xycurve.doc:947
\checkendcurve@, xycurve.doc:680
\checkEqOpen@, xyarc.doc:1128
\checkfoundEbreak@, xycurve.doc:2850
\checkfoundSbreak@, xycurve.doc:2831
\checkgroupbreak@, xyknot.doc:1069
\checkmodmap@, xy2cell.doc:494
\checkoverlap@@, xy.doc:8044
\checkpolytok@, xypoly.doc:519
\checkRadOpen@, xyarc.doc:215
\checkslidemore@, xycurve.doc:1188
\checksplinefind@, xycurve.doc:1315
\checkstartDirection@, xyarc.doc:1303
\checkXy@rules@, xy.doc:5172
\checkxyPScolor@, xyps-c.doc:105
\checkxyPScolor@@, xyps-c.doc:104
\checkXyStyle@, xy.doc:3970
\checkZeroEdge@, xy.doc:3154
\chooseangleinterval@, xy.doc:2038

\Cintercept@@, xy.doc:7931
⟨cir⟩, 85, 88
\cir, xy.doc:5986
\cir , 88, 549
\CIR@, xy.doc:6073
\cir@, xy.doc:5989
\CIR@@, xy.doc:6075
\cir@cir, xy.doc:6000
\cir@diag, xy.doc:6012
\cir@i, xy.doc:6023
\CIRacw@, xy.doc:6273
\cirbuild@, xy.doc:6135
\circhar@, xy.doc:6141
\circhar@@, xy.doc:6145
\circleArc@, xyarc.doc:991
\circleCentre@, xy.doc:7441
\circlecentredEdge@, xy.doc:7477
\circlecentredInner@, xy.doc:7491
\circlecentredOuter@, xy.doc:7495
\circlecentredUnder@, xy.doc:7481
\circleCorner@, xycurve.doc:4715
\Circled, xyv2.doc:495
\circled, xyv2.doc:494
\Circled , 560
\circled , 560
\circled@, xyframe.doc:518
\circled@x@, xyframe.doc:526
\circled@x@@, xyframe.doc:528
\circledobjects@, xycurve.doc:4557
\circleEdge, xy.doc:7420
\circleEdge@, xy.doc:7425
\circleInner@, xy.doc:7455
\circleOuter@, xy.doc:7461
\circleProp@, xy.doc:7431
circles, 14
\circlescale@, xy.doc:7489
\circletemplate@, xycurve.doc:4646
\circleUnder@, xy.doc:7437
\Circmagic, xycurve.doc:4445
\Circmagic@@, xycurve.doc:4444
\CIRcw@, xy.doc:6284
\CIRfull@, xy.doc:6125
\CIRhi@@, xy.doc:6113
\CIRin@@, xy.doc:6064
\CIRlo@@, xy.doc:6112
\CIRorient@@, xy.doc:6066
\CIRout@@, xy.doc:6065
\cirrestrict@@, xy.doc:6168
\CIRtest@@, xy.doc:6111
\CIRtest@inside, xy.doc:6295
\CIRtest@outside, xy.doc:6298
CL, 23
\cL, xyv2.doc:197
\cL , 556
\Clast@@, xy.doc:7933
\clearcurve, xycurve.doc:1847
\clearq@, xymatrix.doc:179
\clearsearchflag@, xycurve.doc:716
cm, 215
\cmerge@, xy.doc:1785
\cmergethec@, xy.doc:1782
\codeof, xy.doc:226
\codeof , 4
\codeof@, xy.doc:227
\Col, xymatrix.doc:160
\Col , 359
\colorxy, xyps-ps.doc:267
column spacing, 354
\comma@@, xy.doc:1591
⟨command⟩, 65
\CompileAllDiagrams, xyv2.doc:797
\CompileFixPoint, xy.doc:4821
\CompileFixPoint , 67
\CompileMatrices, xymatrix.doc:283
\compilename@@, xy.doc:4743
\CompilePrefix, xy.doc:4734
\CompilePrefix , 67
compiling, 67
⟨composite⟩, 46
\composite, xy.doc:3319
\composite , 46, 50, 82
\composite@, xy.doc:3321
\composite@i, xy.doc:3328
\composite@ii, xy.doc:3335
\composite@iii, xy.doc:3337
\composite@iv, xy.doc:3347
\composite@x, xy.doc:3352
\compositemap, xy2cell.doc:1249
\compositemap@, xy2cell.doc:273
\compositemapp, xy2cell.doc:1263
\computeLeftUpness@, xy.doc:3360
\connect, xy.doc:4534
connect, 20
\connect , 65
\connect@, xy.doc:2140
\Connect@@, xy.doc:3226
\connect@iii, xyv2.doc:615
\connect@Twocell, xy2cell.doc:1411

\connectDrop@@, xy.doc:2154
connection methods, 111
\connectobjectbox@@, xy.doc:2153
\connectpostXY@style@, xy.doc:2156
\connectpreXY@style@, xy.doc:2155
\connectRest@@, xy.doc:2158
\connectRestore@, xy.doc:2171
\connectStore@, xy.doc:2160
context, 17
\controlpts, xycurve.doc:1961, xycurve.doc:1989
\convertframed, xyv2.doc:497
\convertframed@i, xyv2.doc:499
\convexhull, xycurve.doc:1962, xycurve.doc:1991
\Conway, xyknot.doc:1231
⟨coord⟩, 23, 280
\COORD@, xy.doc:1513, xynecula.doc:74
copyleft, i
copyright, i
\corn@@, xyframe.doc:380
\corn@x@, xyframe.doc:374
\corn@x@@, xyframe.doc:378
⟨corner⟩, 23
\CORNER@, xy.doc:1443
\CORNER@i, xy.doc:1450
\CORNER@ii, xy.doc:1467
\cosDirection, xy.doc:685
\cosFivePIon, xypoly.doc:697
\cosFourPIon, xypoly.doc:688
\cosPIon, xypoly.doc:663
\cosThreePIon, xypoly.doc:679
\cosTwoPIon, xypoly.doc:670
\count@@, xyidioms.doc:197, xyidioms.doc:200
\count@@@, xyidioms.doc:198, xyidioms.doc:201
\cplusthec@, xy.doc:1694
CR, 23
\cR, xyv2.doc:198
\cR , 556
\Creset@@, xy.doc:7925
\croplattice, xyweb.doc:190
\crv, xycurve.doc:264
\crv@, xycurve.doc:267
\crv@cnt@, xycurve.doc:84
\crv@getStyles@, xycurve.doc:781
\crv@noobject, xycurve.doc:354
\crv@normaltemplate, xycurve.doc:314
\crv@othertemplate, xycurve.doc:315
\crv@saveStyles@, xycurve.doc:777
\crv@special, xycurve.doc:391
\crv@specialtemplate@, xycurve.doc:390
\crv@specialtemplate@@, xycurve.doc:316
\crvconnect@@, xycurve.doc:968
\crvi, xycurve.doc:324
\crvi , 314
\crvobjects@, xycurve.doc:605
\crvpts@, xycurve.doc:86
\crvresetbreaks@, xycurve.doc:1366
\crvs, xycurve.doc:323
\crvs@, xycurve.doc:326
\crvSTYLE@@, xycurve.doc:352
\crvXY@post@, xycurve.doc:775
\crvXY@pre@, xycurve.doc:774
\Cshavec@@, xy.doc:7927
\Cshavep@@, xy.doc:7926
\cskew@, xy.doc:1740
\cskewthec@, xy.doc:1737
\Cslidec@@, xy.doc:7929
\Cslidep@@, xy.doc:7928
\csname , 4
\csp@, xy.doc:700
\ctipEdge@, xy.doc:5595
CU, 23
\cubic@start@, xycurve.doc:3926
\cubicARC@, xyarc.doc:925
\cubicARC@i, xyarc.doc:914
\cubicARCcontrols@@, xyarc.doc:405
\cubicCircleControls@@, xycurve.doc:4496
\cubiccoarsedir@, xycurve.doc:3868
\cubicconnect@, xycurve.doc:3718
\cubicDadvance@@@, xycurve.doc:3823
\cubicdecast@, xycurve.doc:3795
\cubiced@, xycurve.doc:3939
\cubiced@@, xycurve.doc:3832
\cubicfinedir@, xycurve.doc:3851
\cubicIadvance@, xycurve.doc:1519
\cubicinfo@, xycurve.doc:3894
\cubicinsidehull@, xycurve.doc:1796
\cubicIpt@, xycurve.doc:1517
\cubicoutsidehull@, xycurve.doc:1789
\cubicoutsidehull@i, xycurve.doc:1800
\cubicoutsidehull@ii, xycurve.doc:1807
\cubicplot@maybe, xycurve.doc:3825
\cubicplot@maybeii, xycurve.doc:3827
\cubicrec@, xycurve.doc:3819
\cubicreverse@, xycurve.doc:3906
\cubicreverse@@, xycurve.doc:3909
\cubicsegment@, xycurve.doc:3721
\cubicsetparams@, xycurve.doc:3933
\cubictestcvxhull@, xycurve.doc:1612

\cubictestcvxhull@i, xycurve.doc:1618
current object style, 46, 56
\curve, xycurve.doc:256
\curve@check, xyarrow.doc:1759
\curveobject, xy2cell.doc:1087
\curveSTYLE@, xycurve.doc:273
\cuspidalARC@, xyarc.doc:931
\cv@end, xycurve.doc:1919
\cv@restore, xycurve.doc:1920
\cv@start, xycurve.doc:1918
\czeroEdge@, xy.doc:1123

D, 20
D, 23, 40
\D@c, xy.doc:614
\D@p, xy.doc:622
\d@X, xy.doc:673
\d@Y, xy.doc:674
D_c, 21
D_p, 21
\darken@, movie.cls:271
\dash@, xy.doc:5498
\dasharstem@, xyarrow.doc:1320
\dashcorner@, xycurve.doc:4712
\dashcorner@@, xyframe.doc:390
\dashcorntemp@@, xycurve.doc:4706
\Dashed, xyv2.doc:307
\dashed, xyv2.doc:306
\Dashed , 557
\dashed , 557
\dashedcircle@, xycurve.doc:4665
\dashedSpread@, xy.doc:5514
dashes, 14
\dashframed, xyv2.doc:505
\dashframed , 560
\dashhfillchar, xyframe.doc:413
\dashsquiggle@, xy.doc:5529
\dashsquiggledSpread@, xy.doc:5540
\dashvfillchar, xyframe.doc:414
\dbldottedcircle@, xycurve.doc:4669
\dbraced, xyframe.doc:724
\Dc, xyv2.doc:215
\Dc , 556
\dcurveObject@, xy2cell.doc:1091
\Ddashed, xyv2.doc:308
\Ddashed , 557
\dDirection@, xy.doc:7073
\Ddotted, xyv2.doc:312
\Ddotted , 557
\ddtol, xyv2.doc:696
\ddtor, xyv2.doc:697
⟨decor⟩, 65
decoration, 21
\def , 3
\defarstem@, xyarrow.doc:1313
default entry, 357
\defaultinstallps@, xyps-ps.doc:206
\defaultlatticebody, xyweb.doc:267
\defaultpattfile@, xytile.doc:265
\define , 555
define ⟨shape⟩, 36
\definemorphism, xyv2.doc:628
\definemorphism , 561
\definesupermorphism, xy2cell.doc:163
\deflatticebody@, xyweb.doc:266
⟨diag⟩, 46
\DIAG@, xy.doc:4299
\DIAG@@, xy.doc:4307
\DIAG@@@, xy.doc:4313
\diagram, xyv2.doc:536
\diagram , 561
\diagram@, xyv2.doc:538
\diagram@@, xyv2.doc:552
\diagramcompileto, xyv2.doc:769
\diagramnocompile, xyv2.doc:793
\dimen@half, xycurve.doc:2251
\dimen@nth, xycurve.doc:2256
dimension registers, 7
\dir, xy.doc:5035
\dir , 70, 72, 82, 312–314
dir0{(}, xy.doc:5713
dir0{)}, xy.doc:5719
dir0{*}, xy.doc:5912
dir0{+}, xy.doc:5896
dir0{--}, xy.doc:5494
dir0{-}, xy.doc:5128
dir0{.}, xy.doc:5326
dir0{//}, xy.doc:5907
dir0{/}, xy.doc:5903
dir0{<<}, xy.doc:5837
dir0{<}, xy.doc:5617
dir0{>>|}, xy.doc:5862
dir0{>>}, xy.doc:5829
dir0{>|}, xy.doc:5856
dir0{>}, xy.doc:5611
dir0{o}, xy.doc:5917
dir0{x}, xy.doc:5900
dir0{|-}, xy.doc:5890
dir0{|<<}, xy.doc:5877

dir0{|<}, xy.doc:5868
dir0{||}, xy.doc:5850
dir0{|}, xy.doc:5690
dir0{}, xy.doc:5072
dir0{~}, xy.doc:5399
dir0{~~}, xy.doc:5526
dir1{(}, xy.doc:5710
dir1{)}, xy.doc:5716
dir1{--}, xy.doc:5491
dir1{-}, xy.doc:5125
dir1{..}, xy.doc:5547
dir1{.}, xy.doc:5323
dir1{<<}, xy.doc:5832
dir1{<}, xy.doc:5614
dir1{>>|}, xy.doc:5859
dir1{>>}, xy.doc:5824
dir1{>|}, xy.doc:5853
dir1{>}, xy.doc:5608
dir1{|-}, xy.doc:5880
dir1{|<<}, xy.doc:5871
dir1{|<}, xy.doc:5865
dir1{||}, xy.doc:5840
dir1{|}, xy.doc:5682
dir1{}, xy.doc:5073
dir1{~}, xy.doc:5396
dir1{~~}, xy.doc:5523
dir2{--}, xy.doc:5492
dir2{-}, xy.doc:5126
dir2{..}, xy.doc:5548
dir2{.}, xy.doc:5324
dir2{<<}, xy.doc:5835
dir2{<}, xy.doc:5642
dir2{>>|}, xy.doc:5860
dir2{>>}, xy.doc:5827
dir2{>|}, xy.doc:5854
dir2{>}, xy.doc:5641
dir2{|-}, xy.doc:5886
dir2{|<<}, xy.doc:5873
dir2{|<}, xy.doc:5866
dir2{||}, xy.doc:5846
dir2{|}, xy.doc:5687
dir2{}, xy.doc:5076
dir2{~}, xy.doc:5397
dir2{~~}, xy.doc:5524
dir3{--}, xy.doc:5493
dir3{-}, xy.doc:5127
dir3{..}, xy.doc:5549
dir3{.}, xy.doc:5325
dir3{<<}, xy.doc:5836
dir3{<}, xy.doc:5658
dir3{>>|}, xy.doc:5861
dir3{>>}, xy.doc:5828
dir3{>|}, xy.doc:5855
dir3{>}, xy.doc:5657
dir3{|-}, xy.doc:5888
dir3{|<<}, xy.doc:5875
dir3{|<}, xy.doc:5867
dir3{||}, xy.doc:5848
dir3{|}, xy.doc:5688
dir3{}, xy.doc:5077
dir3{~}, xy.doc:5398
dir3{~~}, xy.doc:5525
\dir@, xy.doc:5042
\dir@i, xy.doc:5037
\dir@ii, xy.doc:5040
dir^{'}, xy.doc:5754
dir^{(}, xy.doc:5711
dir^{)}, xy.doc:5717
dir^{<<}, xy.doc:5833
dir^{<}, xy.doc:5615
dir^{>>}, xy.doc:5825
dir^{>}, xy.doc:5609
dir^{`}, xy.doc:5757
dir^{|-}, xy.doc:5882
dir^{||}, xy.doc:5842
dir^{|}, xy.doc:5683
dir^{}, xy.doc:5074
dir_{'}, xy.doc:5755
dir_{(}, xy.doc:5712
dir_{)}, xy.doc:5718
dir_{<<}, xy.doc:5834
dir_{<}, xy.doc:5616
dir_{>>}, xy.doc:5826
dir_{>}, xy.doc:5610
dir_{`}, xy.doc:5758
dir_{|-}, xy.doc:5884
dir_{||}, xy.doc:5844
dir_{|}, xy.doc:5685
dir_{}, xy.doc:5075
\Direction, xy.doc:681
⟨direction⟩, 46
direction, 16
\Direction , 81, 82
\DIRECTION@, xy.doc:4242
\DIRECTION@group, xy.doc:4381
\DIRECTION@i, xy.doc:4414
\DIRECTION@open, xy.doc:4253
\DIRECTION@v, xy.doc:4356

directional, 15
\DirectionChar, xy.doc:688
\DirectionChar , 80
\Directionfromdiag@, xy.doc:4324
\DirectionfromtheDirection@, xy.doc:6997
\Directiontest@@, xy.doc:7021
dir{ }, xy.doc:5079
dir{(}, xy.doc:5714
dir{)}, xy.doc:5720
dir{*}, xy.doc:5909
dir{+}, xy.doc:5894
dir{--}, xy.doc:5495
dir{-}, xy.doc:5129
dir{..}, xy.doc:5550
dir{.}, xy.doc:5327
dir{//}, xy.doc:5905
dir{/}, xy.doc:5902
dir{::}, xy.doc:5551
dir{:}, xy.doc:5328
dir{<<}, xy.doc:5838
dir{<}, xy.doc:5618
dir{==}, xy.doc:5496
dir{=}, xy.doc:5130
dir{>>|}, xy.doc:5863
dir{>>}, xy.doc:5830
dir{>|}, xy.doc:5857
dir{>}, xy.doc:5612
dir{o}, xy.doc:5914
dir{x}, xy.doc:5898
dir{|-}, xy.doc:5891
dir{|<<}, xy.doc:5878
dir{|<}, xy.doc:5869
dir{|=}, xy.doc:5892
dir{||}, xy.doc:5851
dir{|}, xy.doc:5691
dir{}, xy.doc:5070
dir{~}, xy.doc:5400
dir{~~}, xy.doc:5527
\dlDirection@, xy.doc:7068
\DN@, xyidioms.doc:109
\DNii@, xyidioms.doc:110
\do, xy.doc:6837
\dobspline@, xycurve.doc:4017
\docecho@, xy.doc:93
\docfinish@, xy.doc:94
\doCircleArc@@, xyarc.doc:437
\doCircleArc@@@, xyarc.doc:447
\doCircleArcs@, xyarc.doc:548
\doCircleQuadrant@@, xycurve.doc:4512
\doCircleQuadrant@@@, xycurve.doc:4522
\doCircleSectors@, xyarc.doc:1199
\docm@, xy.doc:81
\docm@i, xy.doc:82
\docm@ii, xy.doc:83
\docm@iii, xy.doc:85
\docm@iv, xy.doc:90
DOCMODE, 1, 7
\documentclass , 11
\doEllipseSectors@, xyarc.doc:1284
\doii, xy.doc:6842
\doMergeBoth@@, xyarc.doc:489
\doMergeBothArc@, xyarc.doc:483
\doMergeEnd@@@, xyarc.doc:458
\doMergeEndArc@, xyarc.doc:500
\doMergeEndArc@@, xyarc.doc:452
\doMergeStart@@, xyarc.doc:474
\doMergeStartArc@, xyarc.doc:468
\dontleave@, xy.doc:773
\doPSspecialRotate@@, xyps-r.doc:139
\doShow, xy.doc:6741
\doskewEllipse@, xyarc.doc:1333
\dosolidcircle@, xycurve.doc:4535
\dosolidcircle@@, xycurve.doc:4543
\doSpecialRotate@, xyrotate.doc:359
\dosplineplotpt@, xycurve.doc:3429
\doSplineSegment@, xycurve.doc:4848
\doSplineStraight@, xycurve.doc:4844
\doStraightArc@@, xyarc.doc:504
\dotcorner@, xycurve.doc:4710
\dotcorner@@, xyframe.doc:389
\dotcorntemp@@, xycurve.doc:4708
\dotframed, xyv2.doc:504
\dotframed , 560
\Dotted, xyv2.doc:311
\dotted, xyv2.doc:310
\Dotted , 557
\dotted , 557
\dottedcircle@, xycurve.doc:4659
\dottedSpread@, xy.doc:5366
\dottedSpread@i, xy.doc:5374
\dottedwith, xyv2.doc:314
\dottedwith , 557
\double@, xy.doc:5466
\doublecircle@, xycurve.doc:4647
\Dp, xyv2.doc:222
\Dp , 556
\dparenthesized, xyframe.doc:821
\draw@polyframe, xynecula.doc:146

\draw@polygon, xynecula.doc:213
\drDirection@, xy.doc:7079
⟨driver⟩, 92
\driverextensioncomplain@, xy.doc:6920
\Drop, xyv2.doc:272
\drop, xy.doc:4531
\Drop , 557
\drop , 65
\drop@, xy.doc:2104
\Drop@@, xy.doc:3225
\drop@Twocell, xy2cell.doc:1409
\dropentry@, xymatrix.doc:659
\droplabel@, xyarrow.doc:944
\Droprule@, xy.doc:5303
\droptwocelldrop@, xy2cell.doc:376
\droptwocelllabel@, xy2cell.doc:367
\droptwocelllabel@@, xy2cell.doc:368
\droptwocelltext@, xy2cell.doc:374
\dsize , 555
\dumpPSdict, xyps.doc:659
\dumpPSdict {<filename>}, 260
\dumpPSdict@@, xyps-ps.doc:154
\dumpsizefile, movie.cls:284
\dvidrv@@, xydvidrv.doc:42
\dvidrvEmTeX@, xydvidrv.doc:59
\dvidrvEmTeX@@, xydvidrv.doc:58
\dvidrvLine@, xydvidrv.doc:67
\dvidrvLine@@, xydvidrv.doc:66
\dvimessage@, xy.doc:6929
\dvips@, xyps.doc:266
\dvips@@, xydvips.doc:42
\dvips@Color@, xydvips.doc:71
\dvipsColor@@, xydvips.doc:85
\dvipsCrayola@, xydvips.doc:119
\dvipsCrayola@@, xydvips.doc:117
\dvipsCurrpt@, xydvips.doc:200
\dvipsCurrpt@@, xydvips.doc:201
\dvipsDict@, xydvips.doc:169
\dvipsEM@, xydvips.doc:290
\dvipsEM@@, xydvips.doc:289
\dvipsFrames@, xydvips.doc:257
\dvipsFrames@@, xydvips.doc:255
\dvipsInclude@, xydvips.doc:170
\dvipsLine@, xydvips.doc:244
\dvipsLine@@, xydvips.doc:242
\dvipsMacro@, xydvips.doc:168
\dvipsone@, xyps.doc:270
\dvipsPS@, xydvips.doc:212
\dvipsPS@@, xydvips.doc:138
\dvipsPS@unload, xydvips.doc:140
\dvipsPStypes@, xydvips.doc:151
\dvipsRaw@, xydvips.doc:171
\dvipsRotate@, xydvips.doc:224
\dvipsRotate@@, xydvips.doc:222
\dvipsRotScale@, xydvips.doc:231
\dvipsSpecial@, xydvips.doc:167
\dvipsTiles@, xydvips.doc:270
\dvipsTiles@@, xydvips.doc:268
\dvipsTPIC@, xydvips.doc:282
\dvipsTPIC@@, xydvips.doc:281
\DVITOPS, xydvitops.doc:32
\dvitops@, xyps.doc:269
\dvitops@@, xydvitops.doc:41
\dvitops@Color@, xydvitops.doc:74
\dvitopsBegin@, xydvitops.doc:179
\dvitopsColor@@, xydvitops.doc:72
\dvitopsColor@x, xydvitops.doc:183
\dvitopsCrayon@, xydvitops.doc:136
\dvitopsCrayon@@, xydvitops.doc:135
\dvitopsCurrpt@, xydvitops.doc:208
\dvitopsDict@, xydvitops.doc:175
\dvitopsEnd@, xydvitops.doc:180
\dvitopsFrames@, xydvitops.doc:286
\dvitopsFrames@@, xydvitops.doc:284
\dvitopsInclude@, xydvitops.doc:176
\dvitopsLine@, xydvitops.doc:273
\dvitopsLine@@, xydvitops.doc:271
\dvitopsMacro@, xydvitops.doc:174
\dvitopsOrigin@, xydvitops.doc:178
\dvitopsPS@, xydvitops.doc:216
\dvitopsPS@@, xydvitops.doc:143
\dvitopsPS@unload, xydvitops.doc:145
\dvitopsPStypes@, xydvitops.doc:157
\dvitopsRaw@, xydvitops.doc:177
\dvitopsRotate@, xydvitops.doc:230
\dvitopsRotate@@, xydvitops.doc:228
\dvitopsRotate@x, xydvitops.doc:181
\dvitopsRotScale@, xydvitops.doc:237
\dvitopsSpecial@, xydvitops.doc:173
\dvitopsTiles@, xydvitops.doc:297
\dvitopsTPIC@, xydvitops.doc:310
\dvitopsTPIC@@, xydvitops.doc:308
\dvitopsTransform@x, xydvitops.doc:182
\dvitype@, xyps-ps.doc:92
\dX, xyv2.doc:226
\dY, xyv2.doc:227

E, 23, 40, 103
\eat@, xyidioms.doc:106

\edef , 22
Edge, 20
$Edge_c$, 21
$Edge_p$, 21
\Edge@c, xy.doc:617
\Edge@p, xy.doc:625
\EdgefromtheEdge@, xyframe.doc:104
\ellipA, xyimport.doc:56
\ellipse, xyarc.doc:62
\ellipse@whichdblfrm@, xyframe.doc:589
\ellipse@whichfrm@, xyframe.doc:580
\ellipseArc@, xyarc.doc:393
\ellipsecentredEdge@, xy.doc:7506
\ellipsecentredInner@, xy.doc:7543
\ellipsecentredOuter@, xy.doc:7534
\ellipsecentredUnder@, xy.doc:7522
\ellipsed@, xyframe.doc:641
\ellipsed@e@, xyframe.doc:633
\ellipsed@ei@, xyframe.doc:615
\ellipsed@i, xyframe.doc:649
\ellipsed@i@, xyframe.doc:598
\ellipsed@x@, xyframe.doc:652
\ellipsed@x@@, xyframe.doc:654
\ellipsedashframed@, xyframe.doc:660
\ellipsedotframed@, xyframe.doc:658
\ellipseEdge@, xy.doc:7603
\ellipseInner@, xy.doc:7671
\ellipseOuter@, xy.doc:7669
\ellipseScale@, xyarc.doc:281
\ellipseScale@dim, xyarc.doc:277
\ellipsesolidframed@, xyframe.doc:656
\ellipticARC@, xyarc.doc:922
\ellsave@, xyarc.doc:1158
\EMalternative@, xyemtex.doc:105
\emptyspline@, xycurve.doc:3365
\endcrv@, xycurve.doc:742
\endcrv@@, xycurve.doc:276
\endcsname , 4
\endcurve, xycurve.doc:1982
\enddiagram , 561
\endellipse@@, xyarc.doc:65
\endgraph, xygraph.doc:172
\endpatread@, xytile.doc:301
\endxy, xy.doc:1142
\endxy , 19–21, 64
\endxycurve@, xycurve.doc:1979
\endXYdict@, xyps-ps.doc:466
enter, 17
\enter@, xy.doc:767
entry height, 355
entry margin, 355
entry width, 355
\entry@, xymatrix.doc:607
\entry@@, xymatrix.doc:614
\entry@@i, xymatrix.doc:621
\entry@@norm, xymatrix.doc:644
\entry@@star, xymatrix.doc:625
\entry@@star@i, xymatrix.doc:628
\entry@@star@ii, xymatrix.doc:635
\entrybox, xymatrix.doc:1192
\entrymodifiers, xymatrix.doc:1187
\entrymodifiers , 357
\entrymodifiers@, xymatrix.doc:1190
Error, 14
⟨escape⟩, 457
eu, 215
\everyentry, xymatrix.doc:1243
\everyentry , 359
\everyentry@, xymatrix.doc:1246
\everyxy, xy.doc:1127
\everyxy@@, xy.doc:1130
execution, 19
\expandbeforenext@, xynecula.doc:52
extension, 92
extents, 20
\extractparameterm@, xymovie.doc:64
e|, 280

\F, movie.cls:238, xymovie.doc:80
\F@, movie.cls:241
file header, 1
\filepattern@@, xytile.doc:310
\Filled, xyv2.doc:514
\filled, xyv2.doc:508
\filled@, xyframe.doc:945
\filled@Circle@, xyframe.doc:991
\filled@Ellipse@, xyframe.doc:992
\filled@Oval@, xyframe.doc:990
\filled@Rectangle@, xyframe.doc:989
\filled@Region@, xyframe.doc:981
\filledCircle@@, xyframe.doc:971
\filledCircle@PS, xyps-f.doc:191
\filledCircle@TPIC@@, xytp-f.doc:242
\filledEllipse@@, xyframe.doc:972
\filledEllipse@PS, xyps-f.doc:193
\filledEllipse@TPIC@@, xytp-f.doc:244
\filledOval@@, xyframe.doc:973
\filledOval@PS, xyps-f.doc:195
\filledOval@TPIC@@, xytp-f.doc:245

\filledRectangle@@, xyframe.doc:970
\filledRectangle@PS, xyps-f.doc:189
\filledRectangle@TPIC@@, xytp-f.doc:238
\findbreakwarning@, xycurve.doc:2797
\findIsegment@, xycurve.doc:1528
\findIsegment@i, xycurve.doc:1537
\findIsegment@ii, xycurve.doc:1541
\findIsegment@x, xycurve.doc:1545
\findIsegment@y, xycurve.doc:1550
\finishq@, xymatrix.doc:197
\firstbspline@, xycurve.doc:4024
\firstPS@, xyps-ps.doc:133
\firstPS@@, xyps-ps.doc:135
\firstsplinepoint@, xycurve.doc:2405
fixed grid, 355
\fixedDirection@, xy.doc:7112
\fixedRadiusArc@, xyarc.doc:228
\fixedRadiusArc@i, xyarc.doc:232
\flagwarn@, xycurve.doc:717
fonts, 14
\forceload@XYdict, xyps-ps.doc:609
format dependencies, 7
formats, 7
\forwardSectors@, xyarc.doc:1226
\foundpattern@, xytile.doc:314
\Fr, movie.cls:239, xymovie.doc:81
fraction computation, 18
\frame@compute, movie.cls:216
\frame@emph@, xyframe.doc:940
\frame@emph@@, xyframe.doc:943
\frame@emph@PS, xyps-f.doc:198
\frame@fill@, xyframe.doc:939
\frame@fill@@, xyframe.doc:942
\Framed, xyv2.doc:492
\framed, xyv2.doc:491
\Framed , 560
\framed , 560
\framed@, xyframe.doc:284
\framed@@, xyframe.doc:330
\framed@body@, xyframe.doc:343
\framed@body@@, xyframe.doc:358
\framesetthick@, xyframe.doc:287
\framesetthick@line, xyframe.doc:305
\framezerodot@@, xyframe.doc:265
french.sty, 4
\frm, xyframe.doc:63
\frm , 40, 197
\frm {**}, 209
\frm {*}, 209
frm2{-e}, xyframe.doc:573
frm2{.e}, xyframe.doc:572
\frm@, xyframe.doc:86
\frm@dashh@@, xyframe.doc:406
\frm@dashv@@, xyframe.doc:409
\frm@doth@@, xyframe.doc:258
\frm@dotv@@, xyframe.doc:260
\frm@generic, xyframe.doc:92
\frm@i, xyframe.doc:76
\frm@rddashh@@, xyframe.doc:428
\frm@rddashv@@, xyframe.doc:430
\frm@solidh@@, xyframe.doc:289
\frm@solidv@@, xyframe.doc:290
\frm@thickc@@, xyframe.doc:307
frm[]{--}, xyframe.doc:404
frm[]{-}, xyframe.doc:282
frm[]{.}, xyframe.doc:256
frm[]{=}, xyframe.doc:298
frm[]{}, xyframe.doc:207
frm[o]{--}, xyframe.doc:578
frm[o]{-}, xyframe.doc:575
frm[o]{.}, xyframe.doc:577
frm[o]{=}, xyframe.doc:576
frm[o]{}, xyframe.doc:208
frm[P]{-}, xynecula.doc:141
frm[P]{.}, xynecula.doc:142
frm[P]{=}, xynecula.doc:143
\frmDrop@, xyframe.doc:99
\frmradius@@, xyframe.doc:74
frm{(}, xyframe.doc:811
frm{)}, xyframe.doc:812
frm{**}, xyframe.doc:934
frm{*}, xyframe.doc:929
frm{,}, xyframe.doc:452
frm{-,}, xyframe.doc:459
frm{--}, xyframe.doc:399
frm{-e}, xyframe.doc:569
frm{-o}, xyframe.doc:543
frm{-}, xyframe.doc:281
frm{..}, xyframe.doc:263
frm{.e}, xyframe.doc:568
frm{.o}, xyframe.doc:536
frm{.}, xyframe.doc:252
frm{=}, xyframe.doc:292
frm{\\otherebgroup}, xyframe.doc:714
frm{\\otheregroup}, xyframe.doc:715
frm{^)}, xyframe.doc:814
frm{^\\otheregroup}, xyframe.doc:717
frm{_)}, xyframe.doc:815

frm{_\\otheregroup}, xyframe.doc:718
frm{ee}, xyframe.doc:571
frm{e}, xyframe.doc:567
frm{o-}, xyframe.doc:421
frm{oo}, xyframe.doc:507
frm{o}, xyframe.doc:497
frm{}, xyframe.doc:206
\Fshape@, xyframe.doc:1018, xynecula.doc:550
\Fshape@i, xyframe.doc:1025
\Fshape@ii, xyframe.doc:1030
\Fshape@iii, xyframe.doc:1034
\Fshape@whichframe, xynecula.doc:554
\Fshape@x, xyframe.doc:1037
\fullCircle@, xyarc.doc:191
\fullEllipse@, xyarc.doc:377
\fullPi@, xycurve.doc:4447

german.sty, 4
\getarcExtents@, xyarc.doc:1075
\getarcExtents@ii, xyarc.doc:1082
\getarcExtents@iv, xyarc.doc:1086
\getarcOrientation@, xyarc.doc:1092
\getarcOrientation@i, xyarc.doc:1098
\getarcparams@, xyarc.doc:204
\getbsplineparams@, xycurve.doc:4015
\getcrvsegsnum@, xycurve.doc:2581
\getcubicparams@, xycurve.doc:3890
\getEndDirection@, xyarc.doc:1108
\getEndDirection@i, xyarc.doc:1133
\getEndDirection@x, xyarc.doc:1138
\getEqDirection@, xyarc.doc:1119
\getSpecialRotate@, xyrotate.doc:339
\getsplineedges@, xycurve.doc:2213
\getsplineedges@safe, xycurve.doc:2197
\getsplineedges@safe@, xycurve.doc:2201
\getsquineparams@, xycurve.doc:3629
\gettwocelledges@, xy2cell.doc:1781
\gettwocelledges@@, xy2cell.doc:1786
\getxyDirection@, xy.doc:7235
\getxypattern@, xytile.doc:305
\getxypattern@@, xytile.doc:306
\global , 8
GNU Copyleft, i
\go, xyv2.doc:265
\go , 556
\GRAPH, xygraph.doc:174
⟨graph⟩, 457
\graph, xygraph.doc:164
graph!E, xygraph.doc:500
graph!M, xygraph.doc:455
graph!P, xygraph.doc:495
\GRAPH@, xygraph.doc:176
\GRAPHafterar@, xygraph.doc:206
\GRAPHar@, xygraph.doc:203
\GRAPHeveryarrow@@, xygraph.doc:530
\GRAPHeveryentry, xygraph.doc:464
\GRAPHeveryline@@, xygraph.doc:533
\GRAPHnodemodifiers@@, xygraph.doc:312
\GRAPHsaved@afterar@@, xygraph.doc:201
\GRAPHsaved@PATHafterPOS, xygraph.doc:200
\graphsetup@, xygraph.doc:535
\graphsetup@i, xygraph.doc:537
\grow, xyv2.doc:431
\grow , 559
\grow@, xyv2.doc:435
\grow@@, xyv2.doc:436

h, 46
\H@max, xymatrix.doc:521
\H@maxin, xymatrix.doc:505
\H@maxout, xymatrix.doc:506
\halfEllipse@, xyarc.doc:1349
\halfPi@, xycurve.doc:4448
\halfrootthree, xy.doc:1851
\halfrootthree , 30
\halfroottwo, xy.doc:1849
\halfroottwo , 30
\halign , 361
\halve@dimen, xycurve.doc:2254
\hbox , 46
\Hcap, xyknot.doc:1975
\hcap, xyknot.doc:1545
\Hcross, xyknot.doc:1967
\hcross, xyknot.doc:385
\hcrossneg, xyknot.doc:393
height, 251
\hidePSmessages, xyps-ps.doc:445
\hloop, xyknot.doc:1608
\hole, xyarrow.doc:1003
\hole , 306, 559
\holesize, xyknot.doc:1801
\hollowpoint@, xy.doc:5915
\hook@, xy.doc:5722
\hook@@, xy.doc:5723
hooks, 14
\hover, xyknot.doc:479
\Hrow@c@@, xymatrix.doc:504
\Hrow@HWmax, xymatrix.doc:543
\Hrow@in, xymatrix.doc:502
\Hrow@max, xymatrix.doc:540

\Hrow@out, xymatrix.doc:503
\Hrow@R, xymatrix.doc:537
\htwist, xyknot.doc:658
\htwistneg, xyknot.doc:666
\huncross, xyknot.doc:401
\hunder, xyknot.doc:489
\hunover, xyknot.doc:499
\huntwist, xyknot.doc:674
\HW@max, xymatrix.doc:523
\HWmax@max, xymatrix.doc:546
\HWmax@set, xymatrix.doc:549

i, 46
\idfrombase@, xy.doc:2865
\idfromc@, xy.doc:2862
\idfromcxy@, xy.doc:2893
\idfroms@, xy.doc:2729
\idfromstack@, xy.doc:2868
\idfromxy@, xy.doc:2890
idioms, 7
\ifDIRECTIONempty@, xy.doc:4240
\ifEMspecials@, xyemtex.doc:59
\ifHidden@, xy.doc:3220
\ifInside@, xy.doc:7398
\ifInvisible@, xy.doc:3219
\ifjusthvtest@, xy.doc:5212
\iflastframe@, movie.cls:167, xymovie.doc:89
\ifm@caption, movie.cls:40
\ifm@count, movie.cls:46
\ifm@time, movie.cls:52
\ifnoisymatrices@, xymatrix.doc:219
\ifoldxymatrix@, xymatrix.doc:422
\ifoutlineread@, xy.doc:4887
\ifPATHomitslide@@, xyarrow.doc:313
\ifPATHsingle@, xyarrow.doc:209
\ifpoly@badinters, xynecula.doc:261
\ifpoly@closedrange, xynecula.doc:409
\ifsilentxy@, xy.doc:1538
\ifsm@acceptable, xysmart.doc:297
\ifsm@firstseg, xysmart.doc:296
\ifsplinefar@, xycurve.doc:2170
\ifsplineplot@, xycurve.doc:2171
\ifsupportwarnings@, xy.doc:6901
\iftmp@, xy.doc:662
\ifTPICspecials@, xytpic.doc:61
\ifunsupportwarnings@, xy.doc:6882
\ifVECTORempty@, xy.doc:1390
\ifxydrivernoloads@, xy.doc:6931
\ifxyPSshapes@, xyps-ps.doc:632
\ifxysaving@, xy.doc:4781
\imposeDirection@, xy.doc:7210
\imposeDirection@i, xy.doc:7221
\includePSmessage@, xyps-ps.doc:469
\includePSmessage@@, xyps-ps.doc:468
\includeXYPSdict@, xyps-ps.doc:464
\includeXYPSdict@@, xyps-ps.doc:460
\includeXYPSdict@r, xyps-ps.doc:462
\initbreaks@, xycurve.doc:1368
\initshape@, xycurve.doc:795
\innercroplatticeloop@, xyweb.doc:212
\innerlatticeloop@, xyweb.doc:173
\input xy, 1
\input xypic, 563
\insertmatrixingraph@, xygraph.doc:468
\installCrayolaColors@, xycrayon.doc:46
\installDVITOPSrotate@, xydvitops.doc:245
\installEMline@, xyemtex.doc:202
\installPScolor@, xyps-c.doc:83
\installPSfills@, xyps-f.doc:200
\installPSframes@, xyps-f.doc:63
\installPSline@, xyps-l.doc:79
\installPSlinestyles@, xyps-l.doc:99
\installPSpatterns@, xyps-t.doc:78
\installPSpolylines@, xyps-l.doc:134
\installPSrotscale@, xyps-r.doc:75
\installPSstyle@, xyps-s.doc:86
\installTPICcurves@, xytpic.doc:413
\installTPICfills@, xytp-f.doc:247
\installTPICframes@, xytp-f.doc:55
\installTPICline@, xytpic.doc:318
\installxyps@, xyps-ps.doc:156
\installxyps@@, xyps-ps.doc:160
\installxyps@orig, xyps-ps.doc:195
\installxyps@x, xyps-ps.doc:164
\intercepthull@, xycurve.doc:1715
\interceptinsidehull@, xycurve.doc:1735
\interceptoutsidehull@, xycurve.doc:1761
\interpARC@, xyarc.doc:934
\interpolateinterval@, xy.doc:2043
\interpolatepoint@, xy.doc:2050
\intersect@, xy.doc:2450
\invisbreaks@, xycurve.doc:1365
\inxy@, xy.doc:869
\inxypoly@, xypoly.doc:245
italic corrections, 15
\iterate@, xy.doc:837

\jot , 36
\jumpMOVE@, xygraph.doc:241
\jumpMOVE@@, xygraph.doc:240

\K@, xy.doc:678
\K@dXdY, xy.doc:682
\K@dYdX, xy.doc:683
\khole, xyknot.doc:1052
\KK@, xy.doc:679
\knothole, xyknot.doc:1051
\knothole@, xyknot.doc:1048
\knotholesize, xyknot.doc:1800
\knotholesize@, xyknot.doc:1798
\KNOTstyle, xyknot.doc:1790
\knotSTYLE, xyknot.doc:1781
\knotstyle, xyknot.doc:1784
\knotstyles, xyknot.doc:1787
\knottips, xyknot.doc:1803

L, 20
L, 23, 40
\L@, xycurve.doc:4461
\L@c, xy.doc:615
\L@p, xy.doc:623
L_c, 21
L_p, 21
label separation, 355
\labelbox, xyarrow.doc:1025
\labelbox , 306
\labelmargin, xyarrow.doc:987
\labelmargin , 306
\labelmargin@, xyarrow.doc:985
\labelstyle, xyarrow.doc:1024
\labelstyle , 306
\lastbreak@@, xy.doc:7935
\lastbreaks@, xycurve.doc:1373
\lastbspline@, xycurve.doc:4069
\lastbspline@@, xycurve.doc:4056
\lastobject , 16
\lastobjectbox@, xy.doc:646
\lastprentry@@, xymatrix.doc:380
\lastsplinepoint@, xycurve.doc:2406
LaTeX, 7, 11
\LaTeXdiagrams, xyv2.doc:543
\LaTeXdiagrams , 561
LaTeX 2$_\varepsilon$, 11
\lattice@A, xyweb.doc:50
\lattice@B, xyweb.doc:51
\latticeA, xyweb.doc:52
\latticeB, xyweb.doc:53
\latticebody, xyweb.doc:131, xyweb.doc:278, xyweb.doc:[illegible]
\lbraced, xyframe.doc:765
\Lc, xyv2.doc:216
\Lc , 556
LD, 23
\lDirection@, xy.doc:7106
leave, 17
\leave@, xy.doc:777
\Leftness@, xy.doc:3222
\let , 3, 4
\lighten@, movie.cls:270
\line@, xy.doc:5132
\line@@, xy.doc:5155
⟨list⟩, 457
\literal@, xyidioms.doc:105
\lltod, xyv2.doc:692
\lltou, xyv2.doc:693
\LoadAllPatterns, xytile.doc:280
loading, 1, 3, 4, 11
\LoadPattern, xytile.doc:283
\loadPSdriver@, xyps.doc:262
\loadtipfonts@, xytips.doc:119
\loadxystyle@, xy.doc:3984
logo, 13
\loop , 18
\loop@, xy.doc:836
\loop@@, movie.cls:58
\lowercurveobject, xy2cell.doc:1089
\lowertwocell, xy2cell.doc:1248
\lowertwocell@, xy2cell.doc:271
\lowertwocelll, xy2cell.doc:1259
\Lp, xyv2.doc:223
\Lp , 556
\lparenthesized, xyframe.doc:846
\ltxdiagram@, xyv2.doc:547
LU, 23

\m@caption, movie.cls:143
\m@height, xymovie.doc:61
\m@keys, movie.cls:133, movie.cls:94
\m@nicetotal, movie.cls:225
\m@scene, xymovie.doc:90
\m@width, xymovie.doc:62
\MacPatterns, xytile.doc:60
\makeatletter , 4
\makeatother , 4
\MakeOutlines, xy.doc:4869
\MakeOutlines , 69
math mode, 20
\mathbf, xy2cell.doc:882
\mathsurround , 21
⟨matrix⟩, 343
matrix, 342
matrix orientation, 354

matrix spacing, 354
\matrixsize@, xymatrix.doc:218
\maxcol@, xymatrix.doc:163
\maxcolrow@, xymatrix.doc:164
\maxrow@, xymatrix.doc:162
\maxTPICpoints, xytpic.doc:81
\maybeunraise@, xy.doc:1155
\merge, xyv2.doc:268
\merge , 557
\mergecropextents@, xyweb.doc:227
messages, 13
\middlebspline@, xycurve.doc:4102
\middlebspline@@, xycurve.doc:4071
\mkHmax@, xymatrix.doc:527
\mkHrow@, xymatrix.doc:525
\mkHWdefaults@, xymatrix.doc:531
\mkHWmax@, xymatrix.doc:529
\mkWcol@, xymatrix.doc:526
\mkWmax@, xymatrix.doc:528
\moddefs@, xy2cell.doc:1022
⟨modifier⟩, 46, 280
\modifystyle@, xy.doc:4114
\modmap@, xy2cell.doc:1010
\modmapobject, xy2cell.doc:1054
\modmapobject@, xy2cell.doc:1055
\modPSboxz@, xyps-ps.doc:408
\modPSconnect@, xyps-ps.doc:423
\modPSdrop@, xyps-ps.doc:413
\modXYstyle@, xy.doc:3953
\modXYstyle@@, xy.doc:3959
\modXYtransform@, xyrotate.doc:205
\morepatread@, xytile.doc:302
\morphism, xyv2.doc:613
\morphism , 561
⟨move⟩, 457
\MOVE@, xygraph.doc:243
\MOVE@hop, xygraph.doc:267
\MOVE@letter, xygraph.doc:248
\MOVE@place, xygraph.doc:272
\MOVE@x, xygraph.doc:256
movie, 250
\MovieSetup, movie.cls:138, xymovie.doc:68
\MovieSetup , 251
\MultipleDrivers, xy.doc:6699
\MultipleDrivers , 93
\myPOS, xy2cell.doc:1286

\newdir, xy.doc:5795
\newdir , 82, 312
\newdir@, xy.doc:5797
\newdriver, xy.doc:6808
\newdriver , 95
\newgraphescape, xygraph.doc:598
\newgraphescape , 367
\newgraphescape@, xygraph.doc:607
\newlinechar , 14
\newxyColor, xycolor.doc:175
\newxycolor, xycolor.doc:174
\newxycolor , 235
\newxycolor@, xycolor.doc:193
\newxyPattern, xytile.doc:117
\newxypattern, xytile.doc:114
\newxypattern , 242
\newxypattern@, xytile.doc:123
\newxypattern@i, xytile.doc:115
\newxyPSshape, xyps-ps.doc:679
\newxyshape, xyps-ps.doc:657
\newxystyle, xy.doc:4088
\next, movie.cls:131, movie.cls:30, movie.cls:31, xy.doc:417, xy.doc:420, xy.doc:6836, xy.doc:6841, xy.sty:43, xyknot.doc:1403, xyknot.doc:332
\next , 3
\nextbreak@@, xy.doc:8091
\nextcptlabel, xycurve.doc:1859
\no@, xy.doc:5081
\no@@, xy.doc:7986
\noCalong@@, xy.doc:7952
\noCbreak@@, xy.doc:7973
\noCintercept@@, xy.doc:7960
\noClast@@, xy.doc:7974
\NoCompileAllDiagrams, xyv2.doc:798
\NoCompileMatrices, xymatrix.doc:284
\NoComputerModernTips, xycmtip.doc:52
\noCshavec@@, xy.doc:7942
\noCshavep@@, xy.doc:7937
\noCslidec@@, xy.doc:7948
\noCslidep@@, xy.doc:7944
⟨node⟩, 457
\NODE@, xygraph.doc:220
\NODE@arc, xygraph.doc:501
\NODE@arc@, xygraph.doc:503
\NODE@escape, xygraph.doc:430
\NODE@i, xygraph.doc:274
\NODE@it, xygraph.doc:309
\NODE@matrix, xygraph.doc:457
\NODE@matrix@, xygraph.doc:459
\NODE@poly, xygraph.doc:496
\NODE@poly@, xygraph.doc:497
\NODE@x, xygraph.doc:307

\NODEit@, xygraph.doc:314
\NODEit@i, xygraph.doc:319
\noDictmessage@, xyps-ps.doc:488
\NoDVIPScolor@, xydvips.doc:87
\NoDVIPScolordvi@, xydvips.doc:102
\NoDVITOPScolor@, xydvitops.doc:89
\NoEMlinewidth, xyemtex.doc:212
\NoEMspecials, xyemtex.doc:61
\noinsert@, xy.doc:8002
\nointerceptwarning@, xycurve.doc:1712
\NoisyDiagrams, xyv2.doc:554
\NoisyDiagrams , 561
\NoisyEMTeX, xyemtex.doc:86
\NoisyMatrices, xymatrix.doc:222
\NoisyTPIC, xytpic.doc:130
\none, xyv2.doc:300
\none , 557
\NooldOzTeXColor@, xy17oztex.doc:92
\NooldTexturesColor@, xy16textures.doc:86
\NoOutlines, xy.doc:4876
\NoOutlines , 69
\NoOzTeXColor@, xyoztex.doc:97
\NoPScolor, xyps-c.doc:70
\NoPSframes, xyps-f.doc:71
\NoPSlines, xyps-l.doc:83
\NoPSrotate, xyps-r.doc:86
\NoPSspecials, xyps.doc:308
\NoPSspecials , 254
\NoPStiles, xyps-t.doc:87
\NoResizing, xyrotate.doc:120
\NoRules, xy.doc:5214
\NoRules , 74
\notempty, xyidioms.doc:120
\NoTexturesColor@, xytextures.doc:95
\notip, xyv2.doc:346
\notip , 558
\NoTips, xytips.doc:176
\NoTips , 216
\NoTPICframes, xytp-f.doc:63
\NoTPICspecials, xytpic.doc:63
\notrelaxorelse@, xy.doc:1430
\NoXDVIcolor@, xyxdvi.doc:104
\NoXDVIcolordvi@, xyxdvi.doc:119
\nter@, xy.doc:770
\nudgepos@, xy2cell.doc:828
\nudgepos@@, xy2cell.doc:845
\null@xy@ps, xyps-ps.doc:210
\numcontrolpts, xycurve.doc:1881
\obeyDVIPSlines@, xydvips.doc:182, xydvitops.doc:201
\obeyoldOzTeXlines@, xy17oztex.doc:155
\obeyoldTextureslines@, xy16textures.doc:165
\obeyoutlines@, xyps-ps.doc:562
\obeyOzTeXlines@, xyoztex.doc:160
\obeyTextureslines@, xytextures.doc:170
\obeyXDVIlines@, xyxdvi.doc:195
⟨object⟩, 46
\object, xy.doc:3292
object, 20
\object , 45, 46, 72
object methods, 48
\OBJECT@, xy.doc:3074, xynecula.doc:64
\object@, xy.doc:3294
\OBJECT@@, xy.doc:3125
\OBJECT@change, xy.doc:3522
\OBJECT@change@, xy.doc:3543
\OBJECT@direction, xy.doc:4223
\OBJECT@letter, xy.doc:3079
\OBJECT@other, xy.doc:3092
\OBJECT@set, xy.doc:3532
\OBJECT@shape, xy.doc:3749
\OBJECT@shapei, xy.doc:3779
\OBJECT@shift, xy.doc:3410
\OBJECT@shift@, xy.doc:3416
\OBJECT@x, xy.doc:3176
⟨objectbox⟩, 46
\objectbox, xy.doc:3239, xy.doc:3253
\objectbox , 45, 47, 49, 306
\objectEdge, xy.doc:3698
\objectheight, xy.doc:3490
\objectheight , 52
\objectheight@, xy.doc:3486
\objectmargin, xy.doc:3488
\objectmargin , 52, 305, 306
\objectmargin@, xy.doc:3484
\objectstyle, xy.doc:3241, xy.doc:3255, xy.doc:3668, xypoly.doc:135, xypoly.doc:588
\objectwidth, xy.doc:3489
\objectwidth , 52
\objectwidth@, xy.doc:3485
\old@OzTeX@PS@, xy17oztex.doc:190
\oldconnect, xyv2.doc:616
\oldoztexColor@@, xy17oztex.doc:78
\oldOzTeXColours@, xy17oztex.doc:80
\oldOzTeXCrayon@@, xy17oztex.doc:111
\oldOzTeXCurrpt@, xy17oztex.doc:166
\oldOzTeXCurrpt@@, xy17oztex.doc:167
\oldOzTeXDict@, xy17oztex.doc:134
\oldOzTeXFrames@, xy17oztex.doc:219

\oldoztexFrames@@, xy17oztex.doc:217
\oldOzTeXinclude@, xy17oztex.doc:135
\oldoztexLine@, xy17oztex.doc:206
\oldoztexLine@@, xy17oztex.doc:204
\oldOzTeXMacro@, xy17oztex.doc:133
\oldoztexPS@@, xy17oztex.doc:121, xy17oztex.doc:17
\oldOzTeXPStypes@, xy17oztex.doc:138
\oldoztexPSunload@, xy17oztex.doc:123
\oldOzTeXraw@, xy17oztex.doc:136
\oldOzTeXRotate@, xy17oztex.doc:254
\oldoztexRotate@@, xy17oztex.doc:252
\oldOzTeXSpecial@, xy17oztex.doc:132
\oldOzTeXTiles@, xy17oztex.doc:232
\oldoztexTiles@@, xy17oztex.doc:230
\oldTextures@@, xy16textures.doc:43
\oldtexturesColor@@, xy16textures.doc:84
\oldTexturesColours@, xy16textures.doc:71
\oldtexturesCrayon@, xy16textures.doc:106
\oldtexturesCrayon@@, xy16textures.doc:105
\oldtexturesCurrpt@, xy16textures.doc:172
\oldtexturesCurrpt@@, xy16textures.doc:173
\oldtexturesFrames@, xy16textures.doc:231
\oldtexturesFrames@@, xy16textures.doc:229
\oldTexturesInclude@, xy16textures.doc:145
\oldtexturesLine@, xy16textures.doc:217
\oldtexturesLine@@, xy16textures.doc:215
\oldTexturesMacro@, xy16textures.doc:144
\oldtexturesPS@, xy16textures.doc:190
\oldtexturesPS@@, xy16textures.doc:113
\oldtexturesPStypes@, xy16textures.doc:148
\oldtexturesPSunload@, xy16textures.doc:115
\oldTexturesRaw@, xy16textures.doc:146
\oldtexturesRotate@, xy16textures.doc:203
\oldtexturesRotate@@, xy16textures.doc:201
\oldTexturesSpecial@, xy16textures.doc:143
\oldtexturesTiles@, xy16textures.doc:245
\oldtexturesTiles@@, xy16textures.doc:243
\omitarrows@, xy2cell.doc:508
\omittingTokens@, xyarc.doc:1142
\OnlyOutlines, xy.doc:4871
\OnlyOutlines , 69
\openpattfile@, xytile.doc:273
⟨orient⟩, 85
orientation, 354
\origPATHturn@, xysmart.doc:108
\outlines@@, xy.doc:4865
\outlines@make, xy.doc:4862
\outlines@no, xy.doc:4861
\outlines@only, xy.doc:4863
\outofranges@, xy.doc:2690
\outputstyle@, xy.doc:4119
\outputxyline@, xyline.doc:599
\outputxylinecap@, xyline.doc:319
\outputxylinejoin@, xyline.doc:344
\outputxylinemiter@, xyline.doc:369
\oxy@, xy.doc:858
\OzTeX@, xyps.doc:268
\OzTeX@@, xy17oztex.doc:53, xyoztex.doc:59
\oztexColor@@, xyoztex.doc:83
\OzTeXColours@, xyoztex.doc:85
\OzTeXCrayon@@, xyoztex.doc:116
\OzTeXCurrpt@, xyoztex.doc:171
\OzTeXCurrpt@@, xyoztex.doc:172
\OzTeXDict@, xyoztex.doc:139
\OzTeXFrames@, xyoztex.doc:217
\oztexFrames@@, xyoztex.doc:215
\OzTeXinclude@, xyoztex.doc:140
\oztexLine@, xyoztex.doc:204
\oztexLine@@, xyoztex.doc:202
\OzTeXMacro@, xyoztex.doc:138
\oztexPS@, xyoztex.doc:182
\oztexPS@@, xyoztex.doc:126
\OzTeXPStypes@, xyoztex.doc:143
\oztexPSunload@, xyoztex.doc:128
\OzTeXraw@, xyoztex.doc:141
\OzTeXRotate@, xyoztex.doc:252
\oztexRotate@@, xyoztex.doc:250
\OzTeXRotScale@, xyoztex.doc:265, xyoztex.doc:276
\OzTeXSpecial@, xyoztex.doc:137
\OzTeXTiles@, xyoztex.doc:230
\oztexTiles@@, xyoztex.doc:228

P, 23, 40, 103
p, 21
p, 23
package, 11
\parallelDirection@, xyarc.doc:1372
\parEllipse@, xyarc.doc:1374
\parseCURVE@, xycurve.doc:498
\parsecurve@, xycurve.doc:586
\parseCURVE@@, xycurve.doc:500
\parsecurve@@, xycurve.doc:587
\parsepattern@@, xytile.doc:309
\partroottwo, xy.doc:1850
\partroottwo , 30
\PATH, xyarrow.doc:183
\PATH , 295, 307
\PATH@, xyarrow.doc:211
\PATH@normal, xyarrow.doc:216

\PATHaction, xyarrow.doc:455
\PATHaction , 299
\PATHaction@, xyarrow.doc:462
\PATHaction@default, xyarrow.doc:454
\PATHafter@@, xyarrow.doc:237
\PATHafterPOS, xyarrow.doc:590
\PATHafterPOS , 301
\PATHafterPOS@default, xyarrow.doc:589
\PATHanchor@, xyarrow.doc:377
\PATHanchor@i, xyarrow.doc:379
\PATHanchor@toks, xyarrow.doc:390
\PATHbefore@@, xyarrow.doc:236
\PATHcontinue@@, xyarrow.doc:278
\PATHextra@@, xyarrow.doc:276
\PATHfail@@, xyarrow.doc:234
\PATHfinal@, xyarrow.doc:287
\PATHfinishsegments@, xyarrow.doc:414
\PATHfinishsegments@i, xyarrow.doc:419
\PATHinit@@, xyarrow.doc:274
\PATHinitstraight@, xyarrow.doc:297
\PATHit@, xyarrow.doc:394
\PATHit@i, xyarrow.doc:396
\PATHitshapes@@, xyarrow.doc:392
\PATHlabelabove@, xyarrow.doc:935
\PATHlabelabove@@, xyarrow.doc:941
\PATHlabelalias@, xyarrow.doc:1052
\PATHlabelbelow@, xyarrow.doc:938
\PATHlabelbelow@@, xyarrow.doc:942
\PATHlabelbreak@, xyarrow.doc:959
\PATHlabelit@@, xyarrow.doc:362
\PATHlabels@, xyarrow.doc:364
\PATHlabelsevery@@, xyarrow.doc:239
\PATHlabelsextralast@, xyarrow.doc:294
\PATHlabelslast@@, xyarrow.doc:241
\PATHlabelsnext@@, xyarrow.doc:240
\PATHlastout@@, xyarrow.doc:661
\PATHpost@@, xyarrow.doc:277
\PATHpostpos@@, xyarrow.doc:771
\PATHsegment@, xyarrow.doc:315
\PATHsegment@@, xyarrow.doc:346
\PATHsetting@, xyarrow.doc:243
\PATHslide@@, xyarrow.doc:312
\PATHsmartturn@cir, xysmart.doc:122
\PATHsmartturn@i, xysmart.doc:128
\PATHstraight@, xyarrow.doc:280
\PATHturn@, xyarrow.doc:667, xysmart.doc:109
\PATHturn@cir, xyarrow.doc:677
\PATHturn@diag, xyarrow.doc:704
\PATHturn@i, xyarrow.doc:773
\PATHturn@ii, xyarrow.doc:834
\pfromc@, xy.doc:1632
\pfromthec@, xy.doc:1672
\pfromthep@, xy.doc:1668
\pit , 554
⟨place⟩, 23
\PLACE@, xy.doc:2293
\PLACE@@, xy.doc:2324
\PLACE@intercept, xy.doc:2307
\PLACE@intercept@, xy.doc:2316, xycurve.doc:1650
\placeAllQuadrants@, xycurve.doc:4578
\placeCircObject@, xycurve.doc:4599
\placeCircQuadrant@, xycurve.doc:4590
\PLACEedgec@, xy.doc:2289
\PLACEedgec@@, xy.doc:2286
\PLACEedgep@, xy.doc:2288
\PLACEedgep@@, xy.doc:2285
\PLACEf@, xy.doc:2283
placement state, 21
\placeQuadrants@, xycurve.doc:4572
plainTEX, 7
\plainxy@, xy.doc:892
\point@, xy.doc:5330
\pointlike@, xy.doc:5332
\poly@append, xynecula.doc:204
\poly@append@, xynecula.doc:205
\poly@cachehit, xynecula.doc:279
\poly@cachehitdisable, xynecula.doc:283
\poly@close, xynecula.doc:208
\poly@close@, xynecula.doc:209
\poly@drawseg, xynecula.doc:234
\poly@empty, xynecula.doc:169
\poly@findextent, xynecula.doc:536
\poly@getinterspoints, xynecula.doc:517
\poly@intersdoit, xynecula.doc:307
\poly@interseg, xynecula.doc:357
\poly@intersfinish, xynecula.doc:336
\poly@intersprobecache, xynecula.doc:289
\poly@isinrange, xynecula.doc:410
\poly@isonray, xynecula.doc:447
\poly@isonseg, xynecula.doc:437
\poly@map, xynecula.doc:170
\poly@map@stop, xynecula.doc:178
\poly@mapExpand, xynecula.doc:180
\poly@parse, xynecula.doc:186
\poly@setEdge, xynecula.doc:135
\poly@setEdge@, xynecula.doc:136
\poly@setorigin, xynecula.doc:345
\poly@setorigin@, xynecula.doc:346

\poly@setp, xynecula.doc:229
\poly@setp@, xynecula.doc:230
\polygonEdge, xynecula.doc:243
\polygonEdge@Dist, xynecula.doc:402
\polygonEdge@Inner, xynecula.doc:483
\polygonEdge@Inters, xynecula.doc:263
\polygonEdge@Outer, xynecula.doc:524
\polygonEdge@Under, xynecula.doc:457
\POS, xy.doc:1368
⟨pos⟩, 23
\POS , 24, 65, 299
\POS@, xy.doc:1593
\POS@colon, xy.doc:1603
\POS@star, xy.doc:1608
positions, 20
\postShape@@, xyrotate.doc:194
\postspline@, xycurve.doc:2683
\postStyle@@, xy.doc:3981
\postXY@style@, xy.doc:3854
\postxyescape@, xyps-s.doc:70
\postxyline@, xyline.doc:595
\postxyline@@, xyline.doc:592
\postxylinecap@, xyline.doc:317
\postxylinecap@@, xyline.doc:314
\postxylinejoin@, xyline.doc:342
\postxylinejoin@@, xyline.doc:339
\postxylinemiter@, xyline.doc:367
\postxylinemiter@@, xyline.doc:364
\postXYtransform@, xyps-r.doc:110, xyrotate.doc:186
\postXYtransform@@, xyrotate.doc:189
\pre@emit, xymatrix.doc:444
\pre@emit@, xymatrix.doc:481
\pre@emit@i, xymatrix.doc:485
\pre@emit@ii, xymatrix.doc:492
\preARC@, xyarc.doc:69
\preconnect, xy.doc:4537
\preconnect@, xy.doc:2144
\prentry@, xymatrix.doc:365
\prentry@@, xymatrix.doc:384
\prentry@@@, xymatrix.doc:424
\prentry@@@i, xymatrix.doc:429
\prentry@@norm, xymatrix.doc:405
\prentry@@star, xymatrix.doc:389
\prentry@@star@i, xymatrix.doc:392
\prentry@@star@ii, xymatrix.doc:396
\prentry@@x, xymatrix.doc:410
\preShape@@, xyrotate.doc:193
\preStyle@@, xy.doc:3980
\prevEdge@@, xy.doc:3123
\prevEdgefromtheEdge@, xyframe.doc:107
\prevhowclose@, xycurve.doc:1748
\prevXypostStyle@@, xy.doc:4065
\prevXypreStyle@@, xy.doc:4064
\preXY@style@, xy.doc:3853
\prexyescape@, xyps-s.doc:69
\prexyline@, xyline.doc:594
\prexyline@@, xyline.doc:590
\prexylinecap@, xyline.doc:316
\prexylinecap@@, xyline.doc:312
\prexylinejoin@, xyline.doc:341
\prexylinejoin@@, xyline.doc:337
\prexylinemiter@, xyline.doc:366
\prexylinemiter@@, xyline.doc:362
\preXYstyle@@, xy.doc:3982
\preXYtransform@, xyps-r.doc:109, xyrotate.doc:185
\preXYtransform@@, xyrotate.doc:188
privacy, 3
\processArcDirections@, xyarc.doc:1150
\processArcDirections@@, xyarc.doc:1160
\processDirections@@, xyarc.doc:1146
\PSdict@@, xyps-ps.doc:59
\Pshape@, xynecula.doc:109
\PSignore@, xyps-ps.doc:443
\PSinclude@@, xyps-ps.doc:62
\PSincrease@, xyps.doc:496
\PSmacro@, xyps-ps.doc:449
\PSmacro@@, xyps-ps.doc:58
\PSmessage@, xyps-ps.doc:442
\PSraw@@, xyps-ps.doc:63
\PSread@@, xyps-ps.doc:61
\PSspecial@@, xyps-ps.doc:60
\PSspecialdict@@, xyps-ps.doc:65
\PSspecials@, xyps.doc:444
\PSspecials@@, xyps.doc:449
\PStracing, xyps-ps.doc:446
\pure@crv, xycurve.doc:277

\q, xypoly.doc:609
q, 46
\qartPi@, xytpic.doc:348
\qc, xyv2.doc:449
\qc , 559
\qcount@, xymatrix.doc:176
\qcount@@, xymatrix.doc:177
\qspline, xycurve.doc:4773
\quadARC@, xyarc.doc:928
queue, 17
\queue@, xymatrix.doc:173
\queue@@, xymatrix.doc:174

\quotesmash, xy.doc:6093, xy.doc:6192, xy.doc:6212
\quotient@, xy.doc:812
\quotient@@, xy.doc:824
\quotPTK@, xy.doc:701
\quotsign@@, xy.doc:808

R, 8, 20
R, 23, 40
\R@, xyidioms.doc:87
\R@c, xy.doc:616
\R@p, xy.doc:624
R_c, 21
R_p, 21
⟨radius⟩, 85
\rbraced, xyframe.doc:781
\Rc, xyv2.doc:217
\Rc , 556
RD, 23
\rDirection@, xy.doc:7084
\readoutline@, xy.doc:4889
\readoutline@@, xy.doc:4867
\readsplineparams@, xycurve.doc:2193
\readxycurve, xycurve.doc:1922
\readxycurve@, xycurve.doc:1921
\realinstallxyps@x, xyps-ps.doc:174
\ReCompileAllDiagrams, xyv2.doc:799
\recovercurve@, xycurve.doc:342
\recovercv@end, xycurve.doc:349
\recoverXyStyle@, xy.doc:4072
\rectangleDist@, xy.doc:7787
\rectangleDist@i, xy.doc:7796
\rectangleEdge, xy.doc:7697
\rectangleEdge@, xy.doc:7731
\rectangleEdge@i, xy.doc:7737
\rectangleEdge@ii, xy.doc:7744
\rectangleEdge@iii, xy.doc:7750
\rectangleProp@, xy.doc:7819
\rectangleUnder@, xy.doc:7769
\redefine , 555
redefined, 3
\relax , 3, 4, 65
\reloadxyEM@, xyemtex.doc:138
\reloadxyTPIC@, xytpic.doc:228
\removePT@, xy.doc:809
\resetbreaks@, xycurve.doc:1361
\resetcrvobjects@, xycurve.doc:606
\resetDirection@, xy.doc:7314
\resetPS@, xyps.doc:310
\resetStyle@, xy.doc:3856
\resetstyle@@, xy.doc:4118
\resetTransform@, xyrotate.doc:226
\resetupDirection@, xy.doc:7321
\resetvariant@, xyarrow.doc:1339
\resetxyline@, xyline.doc:597
\resetxyline@@, xyline.doc:601
\resetxyline@i, xyline.doc:598
\resetxylinecap@, xyline.doc:320
\resetxylinecap@@, xyline.doc:322
\resetxylinecap@i, xyline.doc:321
\resetxylinejoin@, xyline.doc:345
\resetxylinejoin@@, xyline.doc:347
\resetxylinejoin@i, xyline.doc:346
\resetxylinemiter@, xyline.doc:370
\resetxylinemiter@@, xyline.doc:372
\resetxylinemiter@i, xyline.doc:371
\restore, xy.doc:4581
\restore , 65
\restoreframestyles@, xyframe.doc:1056
\restR@max, xyframe.doc:958
\reverseDirection@, xy.doc:7253
\reversesplineval@, xycurve.doc:3135
\rotate, xyv2.doc:405
\rotate , 558
\rotate@i, xyv2.doc:410
\rotate@ii, xyv2.doc:412
\rounddashframed, xyv2.doc:506
\rounddashframed , 560
\Row, xymatrix.doc:159
\Row , 359
row spacing, 354
\Rp, xyv2.doc:224
\Rp , 556
\rparenthesized, xyframe.doc:860
\rrtod, xyv2.doc:694
\rrtou, xyv2.doc:695
RU, 23
\rungraphescape@, xygraph.doc:614
\runs@, xy.doc:2688
\runxywith@, xy.doc:6445

s, 23
\s@bot, xy.doc:2673
\s@top, xy.doc:2674
\samecurve, xycurve.doc:614
\save, xy.doc:4576
\save , 65
\save@, xy.doc:4579
\savealias@, xyarrow.doc:1061
\saveC, xy.doc:4577
\savecrvobjects@, xycurve.doc:1963

\savecrvparams@, xycurve.doc:815
\savectrlptsnum@, xycurve.doc:2042
\savecurve, xycurve.doc:613
\saveframestyles@, xyframe.doc:1052
\saveid@, xy.doc:2843
\saveid@COORD, xy.doc:2882
\saveid@COORD@@, xy.doc:2856
\saveid@COORDi, xy.doc:2887
\saveid@COORDii, xy.doc:2888
\savesame@, xycurve.doc:341
\savescope@, xycurve.doc:2036
\saveshape@, xycurve.doc:804
\savespline@, xycurve.doc:2106
\savesplineparams@, xycurve.doc:2119
\savesplinerefs@, xycurve.doc:2122
\savetwocellobjects, xy2cell.doc:1103
\saveXyStyle@, xy.doc:4067
\sbendh, xyknot.doc:1652
\sbendv, xyknot.doc:1648
\scanbspline@, xycurve.doc:4106
\scene, xymovie.doc:92
\scene , 250
\scene@, xymovie.doc:97
\scene@frame, movie.cls:196, xymovie.doc:117
scratch register conventions, 8
\sd@X, xy.doc:675
\sd@Y, xy.doc:676
\searchBreakEsegment@, xycurve.doc:2854
\searchBreakSsegment@, xycurve.doc:2835
\searchflag@, xycurve.doc:714
\searchforEndsegment@, xycurve.doc:2710
\searchforStartsegment@, xycurve.doc:2650
\searchnextpt@, xycurve.doc:695
\selectdriver@, xy.doc:6684
\selectdriver@multiple, xy.doc:6693
\selectdriver@single, xy.doc:6687
\SelectTips, xytips.doc:77
\SelectTips , 215
semidirectional, 15
\SemiDirectionChar, xy.doc:689
\sempty@, xy.doc:2752
\senter@, xy.doc:2741
\setafterCURVEaction@, xycurve.doc:551
\setafterCURVEaction@@, xycurve.doc:562
\setbase@, xy.doc:1869
\setbase@@, xy.doc:1875
\setboxz@h, xyidioms.doc:112
\setcrvobjects@, xycurve.doc:2339
\setcubicdir@, xycurve.doc:3847
\setcubicknotdir@, xycurve.doc:3874
\setcubiczerodir@, xycurve.doc:3878
\setcurve@, xycurve.doc:102
\setcurvearexit@, xyarrow.doc:1757
\setcurvearinit@, xyarrow.doc:1753
\setcurvearinit@i, xyarrow.doc:1752
\setcurveSTYLE@, xycurve.doc:528
\setcurveSTYLE@@, xycurve.doc:529
\setm@caption, movie.cls:177
\setsearchflag@, xycurve.doc:715
\setsplinetest@, xycurve.doc:3225
\setsplineundertest@, xycurve.doc:3203
\setsquinedir@, xycurve.doc:3606
\setsquineknotdir@, xycurve.doc:3615
\setsquinezerodir@, xycurve.doc:3619
\setstdsplinedrop@, xycurve.doc:2317
\settomin@, xy.doc:7757
\setTPICcircstyle@, xytp-f.doc:197
\setupcubic@, xycurve.doc:3737
\setupDirection@, xy.doc:7023
\setupDirection@i, xy.doc:7027
\setupDirection@ii, xy.doc:7170
\setupsquine@, xycurve.doc:3534
\setupxyEMTeX@, xyemtex.doc:118
\setupxyPS@, xyps.doc:493
\setupxyPS@@, xyps-ps.doc:161
\setupxyPSlevelA@, xyps.doc:706
\setupxyPSlevelB@, xyps.doc:724
\setupxyPSlevelC@, xyps.doc:730
\setupxyPSlevelD@, xyps.doc:740
\setupxyPSlevelO@, xyps.doc:483
\setupxyTPIC@, xytpic.doc:203
\setxyPSlevel@, xyps.doc:485
\shaded@, xyframe.doc:455
shape [!...], xyps-ps.doc:227
shape [*...], xyrotate.doc:160
shape [.], xy.doc:3764
shape [<...], xyrotate.doc:159
shape [=...], xy.doc:4040
shape [@...], xyrotate.doc:158
shape [], xy.doc:3759
shape [c], xy.doc:3729
shape [d], xy.doc:3725
shape [dblsize], xyrotate.doc:166
shape [F...], xyframe.doc:1013
shape [flip], xyrotate.doc:164
shape [halfsize], xyrotate.doc:165
shape [hflip], xyrotate.doc:168
shape [Inner], xy.doc:3761

shape [l], xy.doc:3717
shape [left], xyrotate.doc:162
shape [o], xy.doc:3762
shape [Outer], xy.doc:3760
shape [psxy], xyps-ps.doc:231
shape [r], xy.doc:3713
shape [right], xyrotate.doc:163
shape [u], xy.doc:3721
shape [vflip], xyrotate.doc:167
\shiftdir@, xy.doc:5810
\ShowOutlines, xy.doc:4874
\ShowOutlines , 69
\showPSmessages, xyps-ps.doc:444
\silencexy@, xy.doc:1540
\SilentMatrices, xymatrix.doc:221
\SilentMatrices , 343
\sinDirection, xy.doc:686
\sinFivePIon, xypoly.doc:701
\sinFourPIon, xypoly.doc:692
\sinit@, xy.doc:2739
\sinPIon, xypoly.doc:666
\sinThreePIon, xypoly.doc:683
\sinTwoPIon, xypoly.doc:674
⟨size⟩, 46
\skewCircle@, xyarc.doc:1323
\skewEllB@, xyarc.doc:1377
\skewEllipse@, xyarc.doc:1318
\skipspecials@, xy.doc:73, xy.doc:74
\sleave@, xy.doc:2746
⟨slide⟩, 23
\SloppyCurves, xycurve.doc:2290
\sloppyendcrv@, xycurve.doc:729
\sm@accumext, xysmart.doc:426
\sm@advancecount@, xysmart.doc:267
\sm@computeext, xysmart.doc:392
\sm@conn, xysmart.doc:165
\sm@conndraw, xysmart.doc:447
\sm@connect@, xysmart.doc:509
\sm@drawseg, xysmart.doc:462
\sm@drawseglist, xysmart.doc:454
\sm@drop@, xysmart.doc:512
\sm@maxcost, xysmart.doc:162
\sm@nil, xysmart.doc:160
\sm@nnil, xysmart.doc:161
\sm@roundcount@, xysmart.doc:284
\sm@showext, xysmart.doc:361
\sm@straight, xysmart.doc:491
\sm@stri, xysmart.doc:505
\sm@trycirclelist, xysmart.doc:301
\sm@trycirclelist@i, xysmart.doc:367
\sm@trycircles, xysmart.doc:221
\sm@tryclcar, xysmart.doc:373
\smap@, xy.doc:2761
\smapp@@, xy.doc:2769
\smapxy@@, xy.doc:2767
\smapxy@i, xy.doc:2771
\smconn, xysmart.doc:519
\Solid, xyv2.doc:303
\solid, xyv2.doc:302
\Solid , 557
\solid , 557
\solid@, xy.doc:5166
\solidcircle@, xycurve.doc:4542
\solidcorner@, xyframe.doc:362
\solidhrule@, xy.doc:5262
\solidhrule@pre, xy.doc:5265
\solidhrule@typeset, xy.doc:5284
\solidhrule@typeset@, xy.doc:5291
\solidpoint@, xy.doc:5910
\solidSpread@, xy.doc:5186
\solidvrule@, xy.doc:5223
\solidvrule@pre, xy.doc:5226
\solidvrule@typeset, xy.doc:5245
\solidvrule@typeset@, xy.doc:5252
\space@, xyidioms.doc:118
spacing, 354
\spline@@find, xycurve.doc:3141
\spline@@knot, xycurve.doc:3178
\spline@end@, xycurve.doc:3071
\spline@end@@, xycurve.doc:3074
\spline@find, xycurve.doc:3115
\splineadvance@@, xycurve.doc:2355
\splinealong@, xycurve.doc:1238
\splinealong@@, xycurve.doc:1265
\splineBErec@, xycurve.doc:2762
\splinebox@, xycurve.doc:2136
\splinebreak@, xycurve.doc:1386
\splinebreakcancel@, xycurve.doc:3451
\splineBreakEndtest@, xycurve.doc:2440
\splineBreakStarttest@, xycurve.doc:2437
\splineBrec@, xycurve.doc:2736
\splineBSrec@, xycurve.doc:2746
\splinecancel@, xycurve.doc:3443
\splineclosest@, xycurve.doc:1595
\splineconn@, xycurve.doc:2331
\splineConnect@, xycurve.doc:4836
\splineconnect@, xycurve.doc:1064
\splinecontinue, xycurve.doc:2391

\splinecontinue@, xycurve.doc:2385
\splinecontinue@@, xycurve.doc:2388
\splined@@@, xycurve.doc:2417
\splineDadvance@, xycurve.doc:3059
\splineDadvance@@, xycurve.doc:3061
\splinedashed@, xycurve.doc:3306
\splinedbldotted@, xycurve.doc:3313
\splinedefaulttol@, xycurve.doc:2283
\splinedepth@, xycurve.doc:2147
\splinedotted@, xycurve.doc:3310
\splinedoubled@, xycurve.doc:3297
\splineDrop@, xycurve.doc:4832
\splinedrop@, xycurve.doc:2316
\splineEdge, xycurve.doc:822
\splineendScan@, xycurve.doc:2527
\splineEndtest, xycurve.doc:2445
\splineEndtest@, xycurve.doc:2434
\splineErec@, xycurve.doc:2685
\splineextra@@, xycurve.doc:2340
\splinef@breakpt, xycurve.doc:2779
\splinef@end, xycurve.doc:3211
\splinef@pt, xycurve.doc:3212
\splinef@pt@@, xycurve.doc:2786
\splinefcontinue@, xycurve.doc:3186
\splinefindBEnd@, xycurve.doc:2946
\splinefindBStart@, xycurve.doc:2884
\splinefindEnd@, xycurve.doc:2696
\splinefindEnd@@, xycurve.doc:2665
\splinefindStart@, xycurve.doc:2632
\splinefindStart@@, xycurve.doc:2624
\splinefindtest@, xycurve.doc:3198
\splinefinish@, xycurve.doc:3461
\splinefinish@@, xycurve.doc:3441
\splinefinishf@, xycurve.doc:3094
\splinefocus@, xycurve.doc:3236
\splinefrec@, xycurve.doc:3081
\splineglue@, xycurve.doc:3425
\splinegoal@, xycurve.doc:3455
\splineHadvance@, xycurve.doc:1821
\splineHadvance@i, xycurve.doc:1822
\splineHadvance@x, xycurve.doc:1825
\splineHnear@test, xycurve.doc:1769
\splineignore@, xycurve.doc:2178
\splineinsidehull@, xycurve.doc:1731
\splineintercept@, xycurve.doc:1470
\splineintercept@i, xycurve.doc:1472
\splineinterceptTest@, xycurve.doc:1582
\splineIplace@, xycurve.doc:1495
\splineIrec@, xycurve.doc:1509
\splineIsegment@, xycurve.doc:1523
\splineknotpoint@, xycurve.doc:2400
\splineknotpoint@@, xycurve.doc:2402
\splinelast@, xycurve.doc:1403
\splinelength@, xycurve.doc:2158
\splinenear@, xycurve.doc:2344
\splineNointercept@, xycurve.doc:1456
\splinenostart@, xycurve.doc:2523
\splineoutsidehull@, xycurve.doc:1750
\splineoutsidehull@@, xycurve.doc:1753
\splineplot@maybe, xycurve.doc:2451
\splineplot@maybe@@, xycurve.doc:2452
\splineplotpt@, xycurve.doc:3417
\splinepoint, xycurve.doc:2383
\splinepoint@, xycurve.doc:2381
\splinepoint@@, xycurve.doc:2377
\splineRecordBreakEValue, xycurve.doc:3021
\splineRecordBreakValue@, xycurve.doc:2980
\splineRecordEBreakValue@, xycurve.doc:2908
\splineRecordEndBreakValue@, xycurve.doc:2993
\splineRecordEndValue@, xycurve.doc:2969
\splineRecordSBreakValue@, xycurve.doc:2916
\splineRecordValue@, xycurve.doc:2958
\splinereset@, xycurve.doc:1110
\splinereverse@@, xycurve.doc:2676
\splinereverse@@@, xycurve.doc:2677
\splineribboned@, xycurve.doc:3300
\splinescan@, xycurve.doc:2465
\splinescanbreak@, xycurve.doc:2544
\splinescanEndtest, xycurve.doc:2729
\splinescanStarttest, xycurve.doc:2724
\splinesegment@, xycurve.doc:3239
\splineset@, xycurve.doc:3252
\splineset@@, xycurve.doc:3325
\splinesetting@, xycurve.doc:3402
\splinesetting@@, xycurve.doc:3404
\splineshavec@, xycurve.doc:1138
\splineshavep@, xycurve.doc:1129
\splineSintercept@, xycurve.doc:1665
\splineslidec@, xycurve.doc:1167
\splineslidec@@, xycurve.doc:1219
\splineslidec@@@, xycurve.doc:1172
\splineslidemore@, xycurve.doc:1198
\splineslidep@, xycurve.doc:1149
\splineslidep@@@, xycurve.doc:1153
\splineslowEnd@, xycurve.doc:2517
\splineSlowEndtest, xycurve.doc:2449
\splineslowScan@, xycurve.doc:2511
\splineSlowStarttest, xycurve.doc:2447

\splinesolid@, xycurve.doc:3294
\splineSrec@, xycurve.doc:2614
\splinestartScan@, xycurve.doc:2493
\splineStarttest, xycurve.doc:2444
\splineStarttest@, xycurve.doc:2432
\splinetest@, xycurve.doc:3073
\splinetestcvxhull@, xycurve.doc:1726
\splinetol@, xycurve.doc:2152
\splinetolerance, xycurve.doc:2285
\splinetolerance@, xycurve.doc:2279
\splinetooclose@, xycurve.doc:2365
\splinetooclose@@, xycurve.doc:2366
\splinetrace@, xycurve.doc:2179
\splinetracing, xycurve.doc:2177
\splinetrebled@, xycurve.doc:3303
\splineTrec@, xycurve.doc:3380
\splineval@, xycurve.doc:2142
\splinewhich, xycurve.doc:3462
\splinewideend, xycurve.doc:2605
\splinewidestart, xycurve.doc:2597
\splinezeroend, xycurve.doc:2590
\splinezerostart, xycurve.doc:2584
\splitRadius@, xyarc.doc:302
\splitRadius@dim, xyarc.doc:293
\spop@, xy.doc:2735
\Spread@@, xy.doc:8043
\spreaddiagramcolumns, xyv2.doc:566
\spreaddiagramcolumns , 561
\spreaddiagramrows, xyv2.doc:565
\spreaddiagramrows , 561
\spushc@, xy.doc:2716
\spushid@, xy.doc:2720
\squarify, xyv2.doc:442
\squarify , 559
\squash, xyv2.doc:425
\squash , 559
\squiggle@, xy.doc:5402
\squiggle@@, xy.doc:5413
\squiggledSpread@, xy.doc:5429
squiggles, 14
\squine@start@, xycurve.doc:3659
\squineconnect@, xycurve.doc:3515
\squined@, xycurve.doc:3669
\squined@@, xycurve.doc:3673
\squineDadvance@@@, xycurve.doc:3594
\squinedecast@, xycurve.doc:3579
\squineDwhich@, xycurve.doc:3472
\squineIadvance@, xycurve.doc:1513
\squineinfo@, xycurve.doc:3633
\squineIpt@, xycurve.doc:1511
\squineplot@maybe, xycurve.doc:3596
\squinerec@, xycurve.doc:3590
\squinereverse@, xycurve.doc:3643
\squinesegment@, xycurve.doc:3518
\squinesetparams@, xycurve.doc:3665
\squinetestcvxhull@, xycurve.doc:1605
\squinewhich@, xycurve.doc:3464
\ssize , 555
\Ssolid, xyv2.doc:304
\Ssolid , 557
\sssize , 555
\ssz, xycurve.doc:203
\STACK@, xy.doc:2698
\STACK@load, xy.doc:2711
\startxycurve@, xycurve.doc:1916
\startxycurve@@, xycurve.doc:1910
state, 21
\stdsplineconn@, xycurve.doc:2332
⟨step⟩, 457
\stop, xyv2.doc:348
\stop , 558
\stopper@, xy.doc:5693
\stopper@@, xy.doc:5695
\straight@, xy.doc:8046
\straight@typeset, xy.doc:8071
\straightArc@, xyarc.doc:1310
\straightCbreak@, xy.doc:8093
\straightcheckoverlap@, xy.doc:8067
\straightClast@, xy.doc:8103
\straighth@, xy.doc:8149
\straightv@, xy.doc:8176
\straitalong@, xycurve.doc:1022
\straitbreak@, xycurve.doc:1057
\straitconnect@, xycurve.doc:984
\straitintercept@, xycurve.doc:1056
\straitlast@, xycurve.doc:1058
\straitreset@, xycurve.doc:1000
\straitshavec@, xycurve.doc:1013
\straitshavep@, xycurve.doc:1006
\straitslidec@, xycurve.doc:1004
\straitslidep@, xycurve.doc:1003
\stripRCS, xy.doc:412
\stripRCS , 13
\stripRCS@, xy.doc:413
\stripRCS@@, xy.doc:415
style, 56
style [bevel], xyline.doc:251
style [butt], xyline.doc:255

style [miter], xyline.doc:253
style [projcap], xyline.doc:257
style [roundcap], xyline.doc:256
style [roundjoin], xyline.doc:252
style [thicker], xyline.doc:119
style [thinner], xyline.doc:118
style option, 11
\styledboxz@, xy.doc:3908
\supermorphism, xy2cell.doc:160
\swap, xyv2.doc:270
\swap , 557
\swap@, xy.doc:1639
\swapdimen@, xy.doc:1635
\swaptoks@@, xy.doc:1637
system dependencies, 7

\tartPi@, xytpic.doc:349
\testAND@, xy.doc:316
\testbreakedges@, xycurve.doc:3012
\tests@, xy.doc:2683
\testsplineedges@, xycurve.doc:2484
TeX reference point, 20
\Text, xyv2.doc:168
\Text , 555
\text , 555
\Text@, xyv2.doc:172
\Text@i, xyv2.doc:176
\Text@ii, xyv2.doc:186
\TextC, xyv2.doc:170
\Textures@, xyps.doc:267
\Textures@@, xytextures.doc:47
\texturesColor@@, xytextures.doc:93
\TexturesColor@push, xytextures.doc:76
\TexturesColours@, xytextures.doc:78
\texturesCrayon@, xytextures.doc:115
\texturesCrayon@@, xytextures.doc:114
\texturesCurrpt@, xytextures.doc:177
\texturesCurrpt@@, xytextures.doc:178
\TexturesDict@, xytextures.doc:162
\texturesFrames@, xytextures.doc:233
\texturesFrames@@, xytextures.doc:231
\TexturesInclude@, xytextures.doc:164
\texturesLine@, xytextures.doc:219
\texturesLine@@, xytextures.doc:217
\TexturesMacro@, xytextures.doc:161
\texturesPS@, xytextures.doc:193
\texturesPS@@, xytextures.doc:123
\texturesPStypes@, xytextures.doc:147
\texturesPSunload@, xytextures.doc:124
\TexturesRaw@, xytextures.doc:165
\texturesRotate@, xytextures.doc:205
\texturesRotate@@, xytextures.doc:203
\TexturesSpecial@, xytextures.doc:160
\TexturesSpecialDict@, xytextures.doc:163
\texturesTiles@, xytextures.doc:247
\texturesTiles@@, xytextures.doc:245
\theF, movie.cls:158, xymovie.doc:78
\theFr, movie.cls:159, xymovie.doc:79
\theFrame, movie.cls:155
\thelineno@, xy.doc:471
\thematrixCOORD@@, xymatrix.doc:804
\thematrixprefix@@, xymatrix.doc:805
\theMOVE@@, xygraph.doc:218
\theScene, movie.cls:156, xymovie.doc:77
\tildeARC@, xyarc.doc:904
\Tip, xyv2.doc:356
\tip, xyv2.doc:352
\Tip , 558
\tip , 558
\Tip@, xy.doc:5644
\tip@, xy.doc:5620
\Tip@@, xy.doc:5651
\tip@@, xy.doc:5629
\Tip@@eu, xytips.doc:146
\tip@cm, xytips.doc:124
\Tip@eu, xytips.doc:140
\tip@x, xy.doc:5624
\Tip@xy, xytips.doc:52
\tip@xy, xytips.doc:48
tipfamily cm, xytips.doc:115
tipfamily eu, xytips.doc:136
\tipfamily@@, xytips.doc:74
\tipjot@, xy.doc:5822
\tipjot@xy, xytips.doc:55
\TIPS, xytips.doc:88
tips, 14
\tipsize@@, xytips.doc:75
\todl, xyv2.doc:705
\todr, xyv2.doc:706
\toks@, 17
\toks@ii, xyidioms.doc:188, xyidioms.doc:208
\told, xyv2.doc:701
\tolu, xyv2.doc:702
\ToPOS, xy.doc:4554
\toradians@, xy.doc:2048
\tord, xyv2.doc:703
\toru, xyv2.doc:704
\toul, xyv2.doc:707
\tour, xyv2.doc:708

\toXY@, xyv2.doc:680
\TPIC, xytpic.doc:32
\TPICalternative@, xytpic.doc:190
\transformConnect@, xyrotate.doc:222
\transformDrop@, xyrotate.doc:218
\transformstyle@, xy.doc:4122
\transformstyle@@, xy.doc:4121
\transxyline@, xyline.doc:608
\transxyline@@, xyline.doc:610
\TrefoilA, xyknot.doc:1466
\TrefoilB, xyknot.doc:1476
\TrefoilC, xyknot.doc:1486
\triple@, xy.doc:5474
\triplecircle@, xycurve.doc:4650
\trpldottedcircle@, xycurve.doc:4672
\tsize , 555
\Ttip, xyv2.doc:364
\Ttip , 558
\Ttip@, xy.doc:5660
\Ttip@@, xy.doc:5667
\Ttip@@eu, xytips.doc:159
\Ttip@eu, xytips.doc:152
\Ttip@xy, xytips.doc:53
⟨turn⟩, 445
\TurnRadius, xyv2.doc:717
\turnradius, xyarrow.doc:872
\turnradius , 305
\turnradius@, xyarrow.doc:870
\tw@@cell@, xy2cell.doc:467
\tw@@cell@@, xy2cell.doc:691
\tw@cell@, xy2cell.doc:463
\tw@cell@@, xy2cell.doc:693
\tw@cell@@@, xy2cell.doc:281
\twocell, xy2cell.doc:1246
\twocell@, xy2cell.doc:267
\twocell@@, xy2cell.doc:1449
\twocell@@@@, xy2cell.doc:1232
\twocell@DONE, xy2cell.doc:1384
\twocell@path, xy2cell.doc:1193
\twocellhead, xy2cell.doc:453
\twocellhead@, xy2cell.doc:455
\twocelll, xy2cell.doc:1251
\twocellmod@, xy2cell.doc:462
\twocellPATH, xy2cell.doc:1224
\twocellpath, xy2cell.doc:1207
\twocellstyle, xy2cell.doc:365
\twocelltail, xy2cell.doc:454
\twocelltail@, xy2cell.doc:456
\twodecimals@, movie.cls:228
\twoPi@, xycurve.doc:4446
\txt, xy.doc:6331
\txt , 89
\txt@, xy.doc:6333
\txt@i, xy.doc:6337
\txt@ii, xy.doc:6354
\txtline@, xy.doc:6343
\txtline@@, xy.doc:6342
\typesetoutline@, xy.doc:4895

U, 20
U, 23, 40
\U@c, xy.doc:613
\U@p, xy.doc:621
U_c, 21
U_p, 21
\ubraced, xyframe.doc:736
\Uc, xyv2.doc:214
\Uc , 556
\ucurveObject@, xy2cell.doc:1090
\uDirection@, xy.doc:7095
\ulDirection@, xy.doc:7101
\unenter@, xy.doc:775
\uninstallEMline@, xyemtex.doc:207
\UninstallPolylines@, xyline.doc:529
\uninstallTPICcurves@, xytpic.doc:417
\uninstallTPICframes@, xytpic.doc:435
\uninstallTPICline@, xytpic.doc:323
\UnloadColor@, xycolor.doc:108
\UnloadColor@@, xycolor.doc:113
\UnloadDVITOPSrotate@, xydvitops.doc:252
\UnloadFrames@, xyframe.doc:1130
\UnloadLine@, xyline.doc:561
\Unloadpattern@, xytile.doc:221
\UnloadpolySpecials@, xyline.doc:470
\UnloadPS@, xyps-ps.doc:83
\UnloadPS@@, xyps-ps.doc:86
\UnloadPSpatterns@, xyps-t.doc:84
\UnloadPSps@, xyps-ps.doc:67
\UnloadRotate@, xyrotate.doc:362
\Unloadstyle@, xy.doc:3994
\unsetupDirection@, xy.doc:7327
\unsilencexy@, xy.doc:1546
\Up, xyv2.doc:221
\Up , 556
\uparenthesized, xyframe.doc:831
\Upness@, xy.doc:3223
\uppercurveobject, xy2cell.doc:1088
\uppertwocell, xy2cell.doc:1247
\uppertwocell@, xy2cell.doc:269

\uppertwocelll, xy2cell.doc:1255
\urDirection@, xy.doc:7090
\UseAllTwocells, xy2cell.doc:142
\UseCompositeMaps, xy2cell.doc:138
\UseComputerModernTips, xycmtip.doc:51
\UseCrayolaColors, xycolor.doc:269
\UseCrayolaColors , 235
\UseCurveFrames, xyframe.doc:1117
\UseCurveFrames@, xyframe.doc:1114
\UseDVIPSspecials, xyps.doc:231
\UseDVITOPSspecials, xyps.doc:236
\UseEMlinewidth, xyemtex.doc:211
\UseEMspecials, xyemtex.doc:60
\UseFontFrames, xyframe.doc:1122
\UseFontFrames@, xyframe.doc:1119
\UseHalfTwocells, xy2cell.doc:130
\UseOzTeXspecials, xyps.doc:235
\usepackage , 11
\UsePatternFile, xytile.doc:278
\UsePostScriptSpecials, xyps.doc:234
\UsePS@specials, xyps.doc:182
\UsePScolor, xyps-c.doc:69
\UsePSframes, xyps-f.doc:70
\UsePSheader {<filename>}, 260
\UsePSheader {}, 260
\UsePSheader@, xyps-ps.doc:505, xyps.doc:630
\UsePSlines, xyps-l.doc:82
\UsePSrotate, xyps-r.doc:85
\UsePSspecials, xyps.doc:167
\UsePSspecials {}, 255
\UsePSspecials@, xyps.doc:286
\UsePSspecials@i, xyps.doc:168
\UsePSspecials@x, xyps.doc:311
\UsePStiles, xyps-t.doc:86
\UseResizing, xyrotate.doc:121
\UseRules, xy.doc:5215
\UseRules , 74
\UseSingleDriver, xy.doc:6698
\UseSingleDriver , 93
\UseTexturesPSspecials, xyps.doc:232
\UseTexturesSpecials, xyps.doc:233
\UseTips, xytips.doc:174
\UseTips , 216
\UseTPICframes, xytp-f.doc:62
\UseTPICspecials, xytpic.doc:62
\UseTwocells, xy2cell.doc:126
\uutol, xyv2.doc:698
\uutor, xyv2.doc:699

v, 46
\Vcap, xyknot.doc:1983
\vcap, xyknot.doc:1549
\Vcross, xyknot.doc:1953
\vcross, xyknot.doc:381
\vcrossneg, xyknot.doc:389
\vDirection@, xy.doc:7299
⟨vector⟩, 23
\VECTOR@, xy.doc:1420
\VECTOR@a, xy.doc:1435
\VECTOR@letter, xy.doc:1432
\VECTOR@other, xy.doc:1477
\VECTOR@other@open, xy.doc:1498
version, 7, 13
\vfromabsolute@, xy.doc:1647
\vfromabsolute@@, xy.doc:1649
\vfromcartesian@, xy.doc:1863
\vfromcartesian@@, xy.doc:1865
\vfromcartesianangle@, xy.doc:1955
\vfromslide@, xy.doc:2514
\vfromslide@i, xy.doc:2517
\vloop, xyknot.doc:1612
\vover, xyknot.doc:474
\vtwist, xyknot.doc:654
\vtwistneg, xyknot.doc:662
\vuncross, xyknot.doc:397
\vunder, xyknot.doc:484
\vunover, xyknot.doc:494
\vuntwist, xyknot.doc:670

\W@, xyidioms.doc:116
\W@max, xymatrix.doc:522
\W@maxin, xymatrix.doc:499
\W@maxout, xymatrix.doc:500
Warning, 14
warning messages, 3
\Wcol@C, xymatrix.doc:538
\Wcol@c@@, xymatrix.doc:498
\Wcol@HWmax, xymatrix.doc:544
\Wcol@in, xymatrix.doc:496
\Wcol@max, xymatrix.doc:541
\Wcol@out, xymatrix.doc:497
\wdz@, xyidioms.doc:113
\whichCurveObject@, xy2cell.doc:480
\whichframe@@, xyframe.doc:1015
\whichinfo, xy2cell.doc:826
\whichnudge, xy2cell.doc:825
\whichoptions@@, xyframe.doc:1016
\whichPSspecials@, xyps.doc:306
width, 251
\writeoutline@, xy.doc:4878

\writeoutline@@, xy.doc:4866
\writePSdict@@, xyps-ps.doc:529

X, 20
\X, xyframe.doc:898
x, 23, 38
\X@c, xy.doc:611
\X@max, xy.doc:636
\X@min, xy.doc:634
\X@origin, xy.doc:627
\X@p, xy.doc:619
\X@xbase, xy.doc:629
\X@ybase, xy.doc:631
X_{origin}, 21
X_{xbase}, 21
X_{ybase}, 21
X_c, 21
X_p, 21
$X_{\max}$, 20
$X_{\min}$, 20
\xbendd, xyknot.doc:1674
\xbendl, xyknot.doc:1666
\xbendr, xyknot.doc:1662
\xbendu, xyknot.doc:1670
\Xc, xyv2.doc:212
\Xc , 556
\Xcap, xyknot.doc:1895
\xcaph, xyknot.doc:1557
\xcapv, xyknot.doc:1553
\xdashed , 561
\xdotted , 561
\xdouble , 561
\xdvi@@, xyxdvi.doc:56
\xdvi@Color@, xyxdvi.doc:87
\xdviColor@@, xyxdvi.doc:102
\xdviCrayola@, xyxdvi.doc:136
\xdviCrayola@@, xyxdvi.doc:134
\xdviCurrpt@, xyxdvi.doc:213
\xdviCurrpt@@, xyxdvi.doc:214
\xdviDict@, xyxdvi.doc:182
\xdviFrames@, xyxdvi.doc:270
\xdviFrames@@, xyxdvi.doc:268
\xdviInclude@, xyxdvi.doc:183
\xdviLine@, xyxdvi.doc:257
\xdviLine@@, xyxdvi.doc:255
\xdviMacro@, xyxdvi.doc:181
\xdviPS@, xyxdvi.doc:225
\xdviPS@@, xyxdvi.doc:153
\xdviPS@unload, xyxdvi.doc:155
\xdviPStypes@, xyxdvi.doc:166
\xdviRaw@, xyxdvi.doc:184
\xdviRotate@, xyxdvi.doc:237
\xdviRotate@@, xyxdvi.doc:235
\xdviRotScale@, xyxdvi.doc:244
\xdviSpecial@, xyxdvi.doc:180
\xdviTiles@, xyxdvi.doc:283
\xdviTiles@@, xyxdvi.doc:281
\xdviTPIC@, xyxdvi.doc:296
\xdviTPIC@@, xyxdvi.doc:294
\xline , 561
\Xmax, xyv2.doc:238
\Xmax , 556
\Xmin, xyv2.doc:236
\Xmin , 556
\Xorigin, xyv2.doc:229
\Xorigin , 556
\Xover, xyknot.doc:1870
\xoverh, xyknot.doc:515
\xoverv, xyknot.doc:505
\Xp, xyv2.doc:219
\Xp , 556
\xto , 561
\Xtwist, xyknot.doc:1845
\xunderh, xyknot.doc:520
\xunderv, xyknot.doc:510
\Xunover, xyknot.doc:1882
\xunover, xyknot.doc:525
\xunoverh, xyknot.doc:535
\xunoverv, xyknot.doc:530
\Xuntwist, xyknot.doc:1855
\Xxbase, xyv2.doc:231
\Xxbase , 556
\XXover, xyknot.doc:1916
\XXtoY@, xyv2.doc:673
\XXtwist, xyknot.doc:1930
\XY, xyv2.doc:67
\Xy, xy.doc:451
\xy, xy.doc:1109
xy, 215
\Xy , 13
\xy , 19–22, 64
XY-pic, 13
XY-picture state, 21
xy.sty, 11
\xy@, xy.doc:857
\xy@@, xy.doc:886
\xy@@atfont, xytips.doc:61
\xy@@btfont, xytips.doc:62
\xy@@crvaddstack, xycurve.doc:940

\xy@@crvaddstack@, xycurve.doc:945
\xy@@crvmods@, xycurve.doc:650
\xy@@crvmods@@, xycurve.doc:661
\xy@@crvmods@@@, xycurve.doc:662
\xy@@samecurve@, xycurve.doc:340
\xy@begindvi, xyidioms.doc:165
\xy@cosabs@@, xymatrix.doc:301
\xy@cossign@@, xymatrix.doc:299
\xy@cotabs@@, xymatrix.doc:304
\xy@crv@mods@@, xycurve.doc:655
\xy@crvmods@, xycurve.doc:637
\xy@crvmods@@, xycurve.doc:643
\xy@crvmods@@@, xycurve.doc:656
\xy@crvmods@@@@, xycurve.doc:641
\xy@curve@, xycurve.doc:599
\xy@curve@@, xycurve.doc:600
\xy@curve@@@, xycurve.doc:601
\xy@end, xy.doc:1167
\xy@linecap@, xyline.doc:307
\xy@linecap@@, xyline.doc:557
\xy@linecap@i, xyline.doc:299
\xy@linejoin@, xyline.doc:330
\xy@linejoin@@, xyline.doc:558
\xy@linejoin@i, xyline.doc:300
\xy@linemiter@, xyline.doc:355
\xy@linemiter@@, xyline.doc:559
\xy@linemiter@i, xyline.doc:301
\xy@maxX@@, xymatrix.doc:440
\xy@maxY@@, xymatrix.doc:442
\xy@minX@@, xymatrix.doc:439
\xy@minY@@, xymatrix.doc:441
\xy@newCOORD@, xynecula.doc:67
\xy@newOBJECT@, xynecula.doc:57
\xy@oldCOORD@, xynecula.doc:66
\xy@oldOBJECT@, xynecula.doc:56
\xy@polystyle@, xyline.doc:478
\xy@polystyle@@, xyline.doc:527
\xy@sinabs@@, xymatrix.doc:302
\xy@sinsign@@, xymatrix.doc:300
\xy@spline@, xycurve.doc:4775
\xy@spline@g, xycurve.doc:4804
\xy@spline@i, xycurve.doc:4784
\xy@spline@ii, xycurve.doc:4788
\xy@spline@ii@@, xycurve.doc:4798
\xy@spline@iii, xycurve.doc:4800
\xy@spline@iv, xycurve.doc:4813
\xy@spline@v, xycurve.doc:4822
\xy@spline@x, xycurve.doc:4827
\xy@style@, xy.doc:3969
\xy@tanabs@@, xymatrix.doc:303
\xy@transform@, xyrotate.doc:192
\xyactive@Warning, xycurve.doc:516
\xyadddriversupport@, xy.doc:6831
\xyaddsupport, xy.doc:6828
\xyaddsupport , 95
\xyaddunsupport, xy.doc:6829
\xyaddunsupport , 95
\xyadjustBREAK@i, xyknot.doc:1196
\xyadjustJOIN@i, xyknot.doc:1189
\xyadjustknotPLACE@, xyknot.doc:1161
\xyadjustknotPLACE@@, xyknot.doc:1169
\xyadjustknotPLACE@i, xyknot.doc:1155
xyatip10, 14
\xyatipfont, xy.doc:567
\xyatipfont , 14, 15, 80
\Xybase, xyv2.doc:233
\Xybase , 556
\xyBEcolorcheck@, xycolor.doc:295
\xybox, xy.doc:3390
\xybox , 46
\xybspline@, xycurve.doc:3975
\xybspline@iii, xycurve.doc:3955
\xybspline@iv, xycurve.doc:3963
\xybsplinemethods@, xycurve.doc:3983
xybsql10, 14
\xybsqlfont, xy.doc:570
\xybsqlfont , 14, 15, 77, 81, 82
\xybsqlh@, xy.doc:572
\xybsqll@, xy.doc:571
\xybsqlw@, xy.doc:573
xybtip10, 14
\xybtipfont, xy.doc:568
\xybtipfont , 14, 15, 80
\xybug@, xy.doc:489
\xyc@trlpts@, xycurve.doc:1950
\xyc@trlpts@@, xycurve.doc:1952
\xyc@vxhull@, xycurve.doc:1955
\xyc@vxhull@@, xycurve.doc:1958
\xycapSpecial@@, xyline.doc:324
\xycatcodes, xy.doc:58
\xycatcodes , 4, 92
\xycheck@end, xy.doc:1165
\xycheck@end@, xy.doc:1166
xycirc10, 14
\xycircfont, xy.doc:575
\xycircfont , 14, 15, 87
\xycircle, xycurve.doc:4463
\xycircle@, xycurve.doc:4466

\xycircle@i, xycurve.doc:4473
\xycircleframe@x@, xyframe.doc:1098
\xycircleframe@x@i, xyframe.doc:1101
\xycircleframe@x@x, xyframe.doc:1111
\xyclearcrv@, xycurve.doc:1843
\xyclosedown@@, xyidioms.doc:155
\xycolor@, xycolor.doc:105
\xycolor@@, xycolor.doc:102
\xycolor@begin, xydvitops.doc:121
\xycolor@end, xydvitops.doc:122
\xycolor@pop, xycolor.doc:159
\xycolor@push, xycolor.doc:160
\xycolor@push@@, xycolor.doc:158
\xycolor@raw@@, xyps-c.doc:135
\xycolormessage@, xycolor.doc:83
\xycolorwarning@, xycolor.doc:81
\xycolorwarning@@, xycolor.doc:77
\xycomment@, xy.doc:4789
\xycompile, xy.doc:4738
\xycompile , 65, 67
\xycompile@@, xy.doc:4731
\xycompiled, xy.doc:4805
\xycompileno@@, xy.doc:4732
\xycompileto, xy.doc:4746
\xycompileto , 65
\xycontrolpt, xycurve.doc:1895
\xycontrolpt@, xycurve.doc:1883
\xyCOORD@, xy.doc:1508
\xyCOORD@letter, xy.doc:1515
\xyCOORD@other, xy.doc:1528
\xyCOORD@other@open, xy.doc:1548
\xycroplattice@, xyweb.doc:193
\xycrvconn@, xycurve.doc:1948
\xycrvdrop@, xycurve.doc:1947
\xycrvmods@, xycurve.doc:635
\xyctrlpts, xycurve.doc:2019
\xyctrlpts@, xycurve.doc:2011
\xycubicbezier@, xycurve.doc:3691
\xycubicmethods@, xycurve.doc:3697
\xycurve@, xycurve.doc:595
\xycurve@@, xycurve.doc:597
\xycvxhull, xycurve.doc:2085
\xycvxhull@, xycurve.doc:2070
\xycvxhulldrop@, xycurve.doc:2087
xyd.mf, 15
xyd2.mf, 15
xydash10, 14
\xydashfont, xy.doc:562
\xydashfont , 14, 15, 72, 74, 81
\xydashh@, xy.doc:564
\xydashl@, xy.doc:563
\xydashw@, xy.doc:565
\xydate, xy.doc:421
\xydate , 13
\xydecagon@@, xypoly.doc:1021
\xydef@, xy.doc:39, 3
\xydefaultdriver@, xyps-ps.doc:192
\xydefcsname@, xy.doc:201
\xydelayimport@, xyimport.doc:114
\XYdict@, xyps-ps.doc:595
\XYdict@@, xyps-ps.doc:596
\xydist@, xycurve.doc:3045
\xydodecagon@@, xypoly.doc:1109
\xydopoly@style, xyline.doc:491
\xydopoststyles@, xy.doc:3942
\xydopoststyles@@, xy.doc:3940
\xydoprestyles@, xy.doc:3941
\xydoprestyles@@, xy.doc:3939
\xydoTPICcircle@, xytpic.doc:335
\xydriver@unload@support@@, xy.doc:6668
\xydriverfail@, xy.doc:6935
\xydrivernoload@, xy.doc:6932
\xydriversloaded@@, xy.doc:6655
\xydriversselected@@, xy.doc:6656
\xydropRotated@, xyrotate.doc:400
\xydropRotated@@, xyrotate.doc:398
\xydummyrequires@@, xydummy.doc:46
\xyDVITOPScolor@, xydvitops.doc:99
\xyDVITOPScolor@i, xydvitops.doc:103
\xyeat@, xy.doc:4679
\xyeatall@, xymatrix.doc:382
\xyecho, xy.doc:4619
\xyecho , 66
\xyecho@, xy.doc:4622
\xyellipseframe@x@, xyframe.doc:1099
\xyEM@noisy, xyemtex.doc:84
\xyEM@special, xyemtex.doc:80
\xyEMlinewidth@, xyemtex.doc:200
\xyEMstraight@, xyemtex.doc:151
\xyemtex@line, xyemtex.doc:71
\xyemtex@lineto, xyemtex.doc:69
\xyemtex@moveto, xyemtex.doc:70
\xyemtex@pen, xyemtex.doc:73
\xyemtex@point, xyemtex.doc:72
\xyemtex@solid@, xyemtex.doc:146
\xyemtex@splinesolid@, xyemtex.doc:189
\xyendcompiled, xy.doc:4815
\xyenddocmode@, xy.doc:6586

\xyendinput, xy.doc:6575
\xyendinput , 91, 92, 95
\xyerror@, xy.doc:479
\xyerror@RC, xymatrix.doc:867
\xyescape@@, xyps-s.doc:66
\xyescapeSpecial@, xyps-s.doc:72, xyps-s.doc:73
\xyevery@, xy.doc:6483
\xyeveryrequest, xy.doc:6489
\xyeveryrequest , 91
\xyeverywithoption, xy.doc:6490
\xyeverywithoption , 91
\xyexportwarning@, xyimport.doc:127
\xyextern@, xyimport.doc:117
\xyextern@i, xyimport.doc:119
\xyextern@ii, xyimport.doc:123
\xyextern@x, xyimport.doc:130
\xyFN@, xyidioms.doc:108
\xyfont@, xy.doc:559
\xygetfilepatterns@, xytile.doc:267
\xygetknotPLACE@@, xyknot.doc:1181
\xygetpolyprefix@, xypoly.doc:274
\xygetpostBREAK@, xyknot.doc:1202
\xygraph, xygraph.doc:162
\xygraph , 361
\Xygreet@, xy.doc:423
\xyhcap, xyknot.doc:1567
\xyhcross, xyknot.doc:413
\xyhcrossneg, xyknot.doc:419
\xyheptagon@@, xypoly.doc:919
\xyhexagon@@, xypoly.doc:893
\xyhloop, xyknot.doc:1618
\xyhover, xyknot.doc:547
\xyhtwist, xyknot.doc:692
\xyhtwistneg, xyknot.doc:698
\xyhuncross, xyknot.doc:425
\xyhunder, xyknot.doc:553
\xyhunover, xyknot.doc:559
\xyhuntwist, xyknot.doc:695
xyidioms.doc, 7
\xyidiomsloaded, xyidioms.doc:25
\xyignore, xy.doc:4675
\xyignore , 65
\xyignore@, xy.doc:4677
\xyignorescale@, xyrotate.doc:291
\xyignoreSizeMod@, xyrotate.doc:402
\xyimport, xyimport.doc:112
\xyimport@, xyimport.doc:113
\xyimportdefault@, xyimport.doc:126
\xyinitial@, xy.doc:854
\xyinputorelse@, xy.doc:735
\xyinputorelse@@, xy.doc:741
\xyinside@, xy.doc:1113
\xyjoinSpecial@@, xyline.doc:349
\xyk@@, xyknot.doc:1138
\xyk@joincontrol@, xyknot.doc:1508
\xykbacktemp@, xyknot.doc:1044
\xykbacktips@, xyknot.doc:1038
\xykbreak@, xyknot.doc:1056
\xykcaph, xyknot.doc:1586
\xykcapv, xyknot.doc:1579
\xykcheckscale@, xyknot.doc:857
\xykcheckTIPS@, xyknot.doc:1272
\xykcross@, xyknot.doc:365
\xykcrossh, xyknot.doc:446
\xykcrossv, xyknot.doc:438
\xykflexbase, xyknot.doc:1833
\xykflexbase@, xyknot.doc:1832
\xykflexiscale@, xyknot.doc:866
\xykflexiscale@@, xyknot.doc:883
\xykforetemp@, xyknot.doc:1041
\xykforetips@, xyknot.doc:1035
\xykhadjust@, xyknot.doc:906
\xykhobject@, xyknot.doc:800
\xykholesize, xyknot.doc:1799
\xykholetemp@, xyknot.doc:1047
\xykhscale@@, xyknot.doc:897
\xykhuadjust@, xyknot.doc:918
\xykhuobject@, xyknot.doc:815
\xykhvadjust@, xyknot.doc:914
\xykhvobject@, xyknot.doc:810
\xykillstyles@, xyps.doc:1038
\xykillstyles@@, xyps-s.doc:46
\xykjoin@, xyknot.doc:1529
\xykjoincontrol, xyknot.doc:1512
\xykjoincontrol@, xyknot.doc:1514
\xykjoincontrol@@, xyknot.doc:1507
\xykjoincontrol@i, xyknot.doc:1520
\xykkeepscale@, xyknot.doc:843
\xykblooph, xyknot.doc:1634
\xykloopv, xyknot.doc:1628
\xykmidbreak@, xyknot.doc:953
\xykmidbreak@@, xyknot.doc:1812
\xykmidPLACE@, xyknot.doc:1816
\xyknegflexscale@, xyknot.doc:893
\xyknegflexscale@@, xyknot.doc:892
\xyknoflexscale@, xyknot.doc:888
\xyknoflexscale@@, xyknot.doc:887
\xyknotTIPS@, xyknot.doc:1804

\xyknottips@, xyknot.doc:1286
\xyknotTIPS@@, xyknot.doc:1805
\xyknulltemp@, xyknot.doc:1055
\xykoverh, xyknot.doc:601
\xykoverstring@, xyknot.doc:368
\xykoverv, xyknot.doc:594
\xykparsebreak@, xyknot.doc:1008
\xykparsecross@, xyknot.doc:324
\xykparsecross@@, xyknot.doc:342
\xykparsejoin@, xyknot.doc:1391
\xykparsejoin@@, xyknot.doc:1424
\xykparsescale@, xyknot.doc:754
\xykparsetips@, xyknot.doc:1243
\xykparsetips@@, xyknot.doc:1248
\xykPLACE@, xyknot.doc:1817
\xykpostbreak@, xyknot.doc:949
\xykpostbreak@@, xyknot.doc:1811
\xykpostPLACE@, xyknot.doc:1815
\xykprebreak@, xyknot.doc:945
\xykprebreak@@, xyknot.doc:1810
\xykprePLACE@, xyknot.doc:1814
\xykrescale@, xyknot.doc:855
\xykrescale@@, xyknot.doc:870
\xyksavescale@, xyknot.doc:822
\xyksbendh, xyknot.doc:1718
\xyksbendv, xyknot.doc:1711
\xykSCALE@@, xyknot.doc:1807
\xykscaleerror@, xyknot.doc:767
\xyksetjoin@, xyknot.doc:1532
\xyktwisth, xyknot.doc:727
\xyktwistv, xyknot.doc:711
\xykuncrossh, xyknot.doc:462
\xykuncrossv, xyknot.doc:454
\xykunderstring@, xyknot.doc:369
\xykunoverh, xyknot.doc:617
\xykunoverv, xyknot.doc:609
\xykuntwisth, xyknot.doc:734
\xykuntwistv, xyknot.doc:719
\xykvadjust@, xyknot.doc:910
\xykvobject@, xyknot.doc:805
\xykvscale@@, xyknot.doc:901
\xykxbendd, xyknot.doc:1750
\xykxbendl, xyknot.doc:1738
\xykxbendr, xyknot.doc:1732
\xykxbendu, xyknot.doc:1744
\xykxcaph, xyknot.doc:1598
\xykxcapv, xyknot.doc:1592
\xykxoverh, xyknot.doc:629
\xykxoverv, xyknot.doc:623
\xykxunoverh, xyknot.doc:642
\xykxunoverv, xyknot.doc:635
\xykz, xyknot.doc:1136
\xykzbendh, xyknot.doc:1704
\xykzbendv, xyknot.doc:1725
\xylargePolygon@, xypoly.doc:1161
\xylattice, xyweb.doc:161
\xylattice@, xyweb.doc:163
\xylet@, xy.doc:40, 3
\xyletcsnamecsname@, xy.doc:211
\xylinecap@, xyline.doc:221
\xylinejoin@, xyline.doc:222
\xylinemiter@, xyline.doc:223
\xylineSpecial@@, xyline.doc:611
\xylinethick@, xyline.doc:132
\xylinewidth@, xyline.doc:587
\xylinewidth@@, xyline.doc:556
\xylinewidth@i, xyline.doc:137
\xyLoadDrivers@, xy.doc:6701
\xyloaded, xy.doc:24
\xylocalColor@, xycolor.doc:106
\xylocalpattern@, xytile.doc:213
\xylowtolerance@, xycurve.doc:2292
\xymakeADD@, xy.doc:306
\xymath@, xy.doc:1110
\xymatrix, xymatrix.doc:238
\xymatrix , 59, 342
\xymatrix@addop, xymatrix.doc:982
\xymatrix@addop@, xymatrix.doc:983
\xymatrix@addop@@, xymatrix.doc:942
\xymatrix@addop@x, xymatrix.doc:989
\xymatrix@at, xymatrix.doc:944
\xymatrix@C, xymatrix.doc:993
\xymatrix@change, xymatrix.doc:1157
\xymatrix@fix, xymatrix.doc:965
\xymatrix@fix@, xymatrix.doc:973
\xymatrix@fix@x, xymatrix.doc:979
\xymatrix@H, xymatrix.doc:998
\xymatrix@ii, xymatrix.doc:255
\xymatrix@insert, xymatrix.doc:693
\xymatrix@insert@, xymatrix.doc:697
\xymatrix@L, xymatrix.doc:996
\xymatrix@M, xymatrix.doc:995
\xymatrix@measureit, xymatrix.doc:552
\xymatrix@measureit@@, xymatrix.doc:568
\xymatrix@mods, xymatrix.doc:1143
\xymatrix@mods@error, xymatrix.doc:1184
\xymatrix@mods@x, xymatrix.doc:1165
\xymatrix@mods@xx, xymatrix.doc:1177

\xymatrix@prefix, xymatrix.doc:242
\xymatrix@pretypeset, xymatrix.doc:348
\xymatrix@R, xymatrix.doc:992
\xymatrix@rotation, xymatrix.doc:306
\xymatrix@set, xymatrix.doc:1153
\xymatrix@setup, xymatrix.doc:247
\xymatrix@typeset, xymatrix.doc:586
\xymatrix@W, xymatrix.doc:999
\xymatrixcolsep, xymatrix.doc:1018
\xymatrixcolsep@, xymatrix.doc:1015
\xymatrixcompile, xymatrix.doc:280
\xymatrixcompile , 343
\xymatrixcompile@, xymatrix.doc:281
\xymatrixCOORD@, xymatrix.doc:807
\xymatrixCOORD@after, xymatrix.doc:863
\xymatrixCOORD@error, xymatrix.doc:835
\xymatrixCOORD@i, xymatrix.doc:813
\xymatrixCOORD@i@d, xymatrix.doc:824
\xymatrixCOORD@i@l, xymatrix.doc:825
\xymatrixCOORD@i@r, xymatrix.doc:826
\xymatrixCOORD@i@u, xymatrix.doc:823
\xymatrixCOORD@ii, xymatrix.doc:828
\xymatrixCOORD@other, xymatrix.doc:839
\xymatrixCOORD@x, xymatrix.doc:847
\xymatrixCOORD@xx, xymatrix.doc:855
\xymatrixnocompile, xymatrix.doc:224
\xymatrixnocompile , 343
\xymatrixprefix@@, xymatrix.doc:240
\xymatrixrowsep, xymatrix.doc:1017
\xymatrixrowsep@, xymatrix.doc:1014
\xymatrixsavedCOORD@, xymatrix.doc:346
\xymidcross@, xyknot.doc:356
\xymiterSpecial@@, xyline.doc:374
\xynamestyle@, xy.doc:4042
\xynamestyle@@, xy.doc:4048
\xynametransform@, xyrotate.doc:255
\xynew@, xy.doc:41, 3
\xyNoColor@, xycolor.doc:101
\xyNoColor@@, xycolor.doc:103
\xynoedgespline@, xycurve.doc:2491
\xynolinecap@, xyline.doc:546
\xynolinejoin@, xyline.doc:547
\xynolinemiter@, xyline.doc:548
\xynolinewidth@, xyline.doc:544
\xynomath@, xy.doc:1111
\xynonagon@@, xypoly.doc:983
\xyNopattern@, xytile.doc:208
\xyNopattern@@, xytile.doc:210
\xynopolystyle@, xyline.doc:492
\xyNoResizing@, xyrotate.doc:123
\xyNoResizing@@, xyrotate.doc:130
\xynumctrlpts@, xycurve.doc:85
\xyoctagon@@, xypoly.doc:949
\xyoldOzTeXheader, xy17oztex.doc:194
\xyoption, xy.doc:6404
\xyoption , 11, 90, 91
\xyoption@@, xy.doc:6402
\xyoption@truncated, xy.doc:6412
\xyOzTeXheader, xyoztex.doc:192
\xypattern@, xytile.doc:212
\xypattern@@, xytile.doc:209
\xypatternmessage@, xytile.doc:106
\xypatternwarning@, xytile.doc:104
\xypatternwarning@@, xytile.doc:100
\xypentagon@@, xypoly.doc:871
xypic.sty, 563
xypic.tex, 563
\xypoly@, xypoly.doc:224
\xypoly@@, xypoly.doc:316
\xypoly@@@, xypoly.doc:751
\xypoly@@@x, xypoly.doc:771
\xypoly@@i, xypoly.doc:305
\xypoly@i, xypoly.doc:262
\xypoly@n, xypoly.doc:252
\xypoly@SIDES@, xypoly.doc:489
\xypoly@SPOKES@, xypoly.doc:477
\xypoly@x, xypoly.doc:246
\xypolybody@, xypoly.doc:1172
\xypolydash@Special, xyline.doc:468
\xypolydashPS@Special, xyps-l.doc:130
\xypolydefaults@, xypoly.doc:320
\xypolydot@Special, xyline.doc:467
\xypolydotPS@Special, xyps-l.doc:129
\xypolyeofill@Special, xyline.doc:466
\xypolyeofillPS@Special, xyps-l.doc:132
\xypolyfill@Special, xyline.doc:465
\xypolyfillPS@Special, xyps-l.doc:131
\xypolygon, xypoly.doc:222
\xypolygon@@, xypoly.doc:1183
\xyPolygon@@@, xypoly.doc:784
\xypolyiterate@, xypoly.doc:1173
\xypolyline, xyline.doc:389
\xypolyline@, xyline.doc:391
\xypolyline@Special, xyline.doc:464
\xypolylinePS@Special, xyps-l.doc:128
\xypolyloop@, xypoly.doc:1174
\xypolyname, xypoly.doc:272
\xypolynode, xypoly.doc:212

\xypolynode@, xypoly.doc:211
\xypolynospec@, xyline.doc:501
\xypolynullvertex@, xypoly.doc:353
\xypolynum, xypoly.doc:215
\xypolyNUMSIDES@, xypoly.doc:214
\xypolyparams@, xypoly.doc:361
\xypolyPREFIX@, xypoly.doc:270
\xypolyPREFIX@@, xypoly.doc:271
\xypolyprefix@@, xypoly.doc:285
\xypolyrepeat@, xypoly.doc:1171
\xypolystraight@, xyline.doc:516
\xypolytoosmall@, xypoly.doc:773
\xypolyvertex@, xypoly.doc:356
\xypolyVERTEXerror@, xypoly.doc:377
\xypolywhich@, xypoly.doc:463
\xypolywidevertex@, xypoly.doc:359
\xyPOS@, xy.doc:1560
\xypost@Style@, xy.doc:3932
\xypostcross@, xyknot.doc:351
\xypostcurve@, xycurve.doc:3248
\xypre@skipStyle@, xy.doc:3936
\xypre@Style@, xy.doc:3923
\xyprecross@, xyknot.doc:346
\xyprecurve@, xycurve.doc:3247
\XYpredict@, xyps-ps.doc:593
\XYpredict@@, xyps-ps.doc:584
\XYpredict@r, xyps-ps.doc:592
\xyprovide, xy.doc:6558
\xyprovide , 91, 92
\xyPS@@, xyps-ps.doc:523
\xyPS@@@, xyps-ps.doc:524
\xyPS@color@@, xyps-c.doc:134
\xyps@options, xyps.doc:303
\xyPS@pattern@@, xyps-t.doc:125
\xyPSabtip@, xyps.doc:843
\xyPSabtip@i, xyps.doc:856
\xyPSahook@, xyps.doc:872
\xyPSalternative@, xyps.doc:691
\xyPSatip@, xyps.doc:839
\xyPSatip@i, xyps.doc:852
\xyPSaturn@, xyps.doc:879
\xyPSbhook@, xyps.doc:873
\xyPSbtip@, xyps.doc:841
\xyPSbtip@i, xyps.doc:854
\xyPSbturn@, xyps.doc:880
\xyPScapSpecial@, xyps-l.doc:95
\xyPSchar@, xyps.doc:768
\xyPScharmessage@, xyps-ps.doc:639
\xyPScharwarning@@, xyps-ps.doc:634
\xyPScirbuild@, xyps.doc:984
\xyPScirc@, xyps.doc:973
\xyPScirc@Special, xyps-f.doc:170
\xyPScirchar@, xyps.doc:933
\xyPScirchar@@, xyps.doc:951
\xyPScirchar@@@, xyps.doc:963
\xypscircle@, xyps-f.doc:158
\xyPScircled@x@, xyps-f.doc:98
\xyPScircled@x@@, xyps-f.doc:105
\xyPSCIRfull@, xyps.doc:981
\xyPScirrest@, xyps.doc:945
\xyPScm@, xyps.doc:816
\xyPScm@@, xyps.doc:817
\xyPScmatip@, xyps.doc:825
\xyPScmbtip@, xyps.doc:826
\xyPScmtip@, xyps.doc:824
\xyPScolor@, xyps-c.doc:122
\xyPScolor@@, xyps-c.doc:124
\xyPScubic@, xyps.doc:1113
\xyPScurrpt@, xyps-ps.doc:321
\xyPScurrpt@@, xyps-ps.doc:332
\xyPScurrpt@@@, xyps-ps.doc:331
\xyPSdash@, xyps.doc:911
\xyPSdashcirc@Special, xyps-f.doc:172
\xyPSdashed@, xyps.doc:1052
\xyPSdashedSpread@, xyps.doc:1053
\xyPSdashhfill@, xyps-f.doc:131
\xyPSdashoval@Special, xyps-f.doc:174
\xyPSdashvfill@, xyps-f.doc:132
\xyPSdefaultdict, xyps-ps.doc:499, xyps.doc:624
\xyPSdefaultdict , 260
\xyPSdictknown@, xyps-ps.doc:471
\xyPSdotcirc@Special, xyps-f.doc:171
\xyPSdotoval@Special, xyps-f.doc:173
\xyPSdottedSpread@, xyps.doc:1050
\xyPSdriver, xyps-ps.doc:193
\xypsellipse@, xyps-f.doc:162
\xyPSellipsed@x@, xyps-f.doc:113
\xyPSellipsed@x@@, xyps-f.doc:121
\xyPSescapestyle@, xyps-s.doc:56
\xyPSfillcirc@Special, xyps-f.doc:221
\xyPSfilledRegion, xyps-f.doc:185
\xyPSfillellipse@Special, xyps-f.doc:224
\xyPSfilloval@Special, xyps-f.doc:218
\xyPSfliptrue@, xyps.doc:909
\xyPSfont@, xyps.doc:780
\xyPSframed@body@, xyps-f.doc:82
\xyPSframed@oval@, xyps-f.doc:94
\xyPShook@, xyps.doc:871

\xyPSjoinSpecial@, xyps-l.doc:96
\xyPSlineSpecial@, xyps-l.doc:94
\xyPSmath@, xyps-c.doc:101
\xyPSmiterSpecial@, xyps-l.doc:97
\xyPSnewcolor@, xyps-c.doc:109
\xyPSnewpattern@, xyps-t.doc:100
\xyPSnomath@, xyps-c.doc:102
\xyPSnotfinished@@, xyps-ps.doc:693
\xyPSnotimplemented@@, xyps-ps.doc:690
\xyPSobeylines@@, xyps-ps.doc:64
\xypsoval@, xyps-f.doc:144
\xyPSoval@Special, xyps-f.doc:169
\xyPSpattern@, xyps-t.doc:113
\xyPSpattern@@, xyps-t.doc:116
\xyPSpoint@, xyps.doc:1079
\xyPSpost@, xyps-ps.doc:434
\xyPSposttransform@, xyps-r.doc:107
\xyPSpre@, xyps-ps.doc:433
\xyPSpretransform@, xyps-r.doc:106
\xyPSpsxy@, xyps-ps.doc:316
\xyPSrawA@, xyps-ps.doc:397
\xyPSrawAA@, xyps-ps.doc:398
\xyPSrawstyleA@, xyps-s.doc:94
\xyPSrawZ@, xyps-ps.doc:400
\xyPSrotate@@, xyps-r.doc:132
\xyPSrotSplit@, xyps-r.doc:112
\xyPSscale@@, xyps-r.doc:122
\xyPSsemichar@, xyps.doc:771
\xyPSsemifont@, xyps.doc:781
\xyPSshapechar@, xyps-ps.doc:387
\xyPSshipout@, xyps-ps.doc:106
\xyPSshipout@i, xyps-ps.doc:108
\xyPSshipout@ii, xyps-ps.doc:109
\xyPSshipout@PSii, xyps-ps.doc:121
\xyPSsolid@, xyps.doc:1047
\xyPSsolidSpread@, xyps.doc:1048
\xyPSspecialRotate@, xyps-r.doc:136
\xyPSsplinedashed@, xyps.doc:1128
\xyPSsplinedotted@, xyps.doc:1131
\xyPSsplinesolid@, xyps.doc:1125
\xyPSsplitescape@, xyps-s.doc:57
\xyPSsplitPS@, xyps-s.doc:63
\xyPSsplitshape@, xyps-ps.doc:388
\xyPSspread@, xyps.doc:1056
\xyPSsquiggle@, xyps.doc:889
\xyPSsquine@, xyps.doc:1118
\xyPSstopper@, xyps.doc:906
\xyPSstraight@, xyps.doc:1010
\xyPSTip@, xyps.doc:798
\xyPStip@, xyps.doc:795
\xyPSTip@eu@, xyps.doc:802
\xyPStips@, xyps.doc:833
\xyPSTtip@, xyps.doc:799
\xyPSTtip@eu@, xyps.doc:803
\xyPSTtips@, xyps.doc:859
\xyPSwarning@, xyps.doc:313
\xyPSwhichabtip@, xyps.doc:850
\xyPSwhichatip@, xyps.doc:846
\xyPSwhichbtip@, xyps.doc:848
\xyPSzerodot@, xyps.doc:1087
\xyq@, xymatrix.doc:675
\xyq@@toksix@, xymatrix.doc:677
\xyq@@toksix@i, xymatrix.doc:680
\xyqall@, xymatrix.doc:670
\xyqs@, xymatrix.doc:684
\xyquadbezier@, xycurve.doc:3488
\xyquadmethods@, xycurve.doc:3495
\xyquiet, xy.doc:4659
\xyquiet , 65, 66
\xyread@, xy.doc:697
\xyreadpatfile@, xytile.doc:294
\xyrealshipout@, xyps-ps.doc:105
\xyrecat, xy.doc:310
xyrecat.tex, 5
\xyrecat@, xy.doc:309
\xyrecompile@, xy.doc:4764
\xyrecompile@@, xy.doc:4744
\xyReloadDrivers, xy.doc:6703
\xyReloadDrivers , 93, 94
\xyReloadDrivers@, xy.doc:6702
\xyrequire, xy.doc:6421
\xyrequire , 90
\xyrescale@, xyrotate.doc:281
\xyresetcatcodes, xy.doc:311
\xyreuncatcodes, xy.doc:47
\xyRotate@, xyrotate.doc:357
\xyrotatechar@, xyrotate.doc:325
\xyrotateSizeMod@, xyrotate.doc:379
\xyRotScale@@, xyps-r.doc:160
\xysave@, xy.doc:4783
\xysave@@toksix@, xy.doc:4786
\xysaveMinMax@, xy2cell.doc:1215
\xysaving@, xy.doc:4778
\xysbendh, xyknot.doc:1686
\xysbendv, xyknot.doc:1685
\xyscale@, xyrotate.doc:360
\xyscalechar@, xyrotate.doc:269
\xyscalechars@, xyrotate.doc:271

\xyselectoptionsupport@, xy.doc:6721
\xysetjoincontrol@, xyknot.doc:1206
\xysetknotPLACE@, xyknot.doc:1172
\xysetknotPLACE@@, xyknot.doc:1176
\xysetlinecap@, xyline.doc:269
\xysetlinecap@@, xyline.doc:294
\xysetlinejoin@, xyline.doc:275
\xysetlinejoin@@, xyline.doc:295
\xysetmiterlimit@, xyline.doc:281
\xysetmiterlimit@@, xyline.doc:296
\xysetpostbreak@, xyknot.doc:1205
\xysetup@@, xyidioms.doc:133
\xysetup@dummy, xyidioms.doc:132
\xyshape@bevel@, xyline.doc:267
\xyshape@butt@, xyline.doc:261
\xyshape@dblsize@, xyrotate.doc:301
\xyshape@flip@, xyrotate.doc:302
\xyshape@halfsize@, xyrotate.doc:300
\xyshape@hflip@, xyrotate.doc:304
\xyshape@left@, xyrotate.doc:408
\xyshape@miter@, xyline.doc:265
\xyshape@projcap@, xyline.doc:263
\xyshape@rdcap@, xyline.doc:262
\xyshape@rdjoin@, xyline.doc:266
\xyshape@right@, xyrotate.doc:409
\xyshape@thicker@, xyline.doc:134
\xyshape@thinner@, xyline.doc:135
\xyshape@vflip@, xyrotate.doc:303
\xyshearchar@, xyrotate.doc:424
\xyShowDrivers, xy.doc:6735
\xyShowDrivers , 94
\xysidesofhull@, xycurve.doc:1624
\xysidesofinterval@, xycurve.doc:1555
\xySpecialRotate@, xyrotate.doc:358
\xySpecialRotate@i, xyrotate.doc:328
\xysplinespecialcases@, xycurve.doc:3270
\xysquare@@, xypoly.doc:853
\xystackcolor@, xycolor.doc:143
\xystackcolor@i, xycolor.doc:147
\xystandardcolors@, xycolor.doc:230
\xystandardpatterns@, xytile.doc:254
\xystatus, xy2cell.doc:1245
\xystatus@, xy.doc:4640
\xystatus@c, xy.doc:4655
\xystatus@cp, xy.doc:4648
\xysupportwarning@, xy.doc:6903
\xysupportwarnings@@, xy.doc:6917
\xytestforsupport@, xy.doc:6855
\xytilde@Warning, xycurve.doc:518
\xytotoks@, xy.doc:2758
\xytotoks@@toksix@, xy.doc:2759
\xytpfloaded, xytp-f.doc:25
\xyTPIC@accumulate, xytpic.doc:399
\xytpic@ar, xytpic.doc:119
\xytpic@bk, xytpic.doc:126
\xytpic@circhar@, xytpic.doc:355
\xytpic@cnt, xytpic.doc:79
\xytpic@da, xytpic.doc:116
\xytpic@dad, xytpic.doc:170
\xytpic@dashed@, xytpic.doc:241
\xytpic@dashed@@, xytpic.doc:168
\xytpic@dot@@, xytpic.doc:157
\xytpic@dotted@@, xytpic.doc:169
\xytpic@dt, xytpic.doc:117
\xytpic@dtd, xytpic.doc:172
\xytpic@fp, xytpic.doc:114
\xytpic@ia, xytpic.doc:120
\xytpic@in, xytpic.doc:149
\xytpic@ip, xytpic.doc:115
\xytpic@line, xytpic.doc:162
\xytpic@max, xytpic.doc:80
\xyTPIC@noisy, xytpic.doc:129
\xytpic@pa, xytpic.doc:109
\xytpic@pn, xytpic.doc:108
\xytpic@point@, xytpic.doc:244
\xytpic@polydashed@, xytpic.doc:176
\xytpic@polydotted@, xytpic.doc:177
\xytpic@polysolid@, xytpic.doc:175
\xytpic@postpoly@, xytpic.doc:178
\xytpic@pt, xytpic.doc:146
\xytpic@reset@, xytpic.doc:100
\xytpic@set@, xytpic.doc:102
\xytpic@sh, xytpic.doc:121
\xytpic@solid@, xytpic.doc:236
\xytpic@solid@@, xytpic.doc:167
\xytpic@sp, xytpic.doc:118
\xyTPIC@special, xytpic.doc:128
\xyTPIC@spline@ii@, xytpic.doc:387
\xytpic@splinedashed@, xytpic.doc:295
\xytpic@splinedotted@, xytpic.doc:297
\xytpic@splinesolid@, xytpic.doc:294
\xytpic@style@@, xytpic.doc:99
\xytpic@wh, xytpic.doc:125
\xytpic@width@, xytpic.doc:158
\xyTPICcirc@Special, xytp-f.doc:222
\xyTPICcirchar@i, xytpic.doc:381
\xyTPICcircle@, xytp-f.doc:200
\xyTPICcircled@x@, xytp-f.doc:92

\xyTPICcircled@xx, xytp-f.doc:97
\xyTPICdashcirc@, xytp-f.doc:102
\xyTPICdashcirc@Special, xytp-f.doc:224
\xyTPICdashellip@, xytp-f.doc:105
\xyTPICdashrect@Special, xytp-f.doc:221
\xyTPICdotcirc@, xytp-f.doc:100
\xyTPICdotcirc@Special, xytp-f.doc:223
\xyTPICdotellip@, xytp-f.doc:104
\xyTPICdotrect@Special, xytp-f.doc:220
\xyTPICellipse@, xytp-f.doc:214
\xyTPICellipsed@x@, xytp-f.doc:111
\xyTPICellipsed@xx, xytp-f.doc:116
\xyTPICfillcirc@Special, xytp-f.doc:267
\xyTPICfilledoval@, xytp-f.doc:163
\xyTPICfilledRegion, xytp-f.doc:235
\xyTPICfillellipse@Special, xytp-f.doc:270
\xyTPICfilloval@Special, xytp-f.doc:264
\xyTPICfillrect@Special, xytp-f.doc:261
\xyTPICframed@body@, xytp-f.doc:71
\xyTPICframed@rect@i, xytp-f.doc:81
\xyTPICfullcircle@, xytpic.doc:341
\xyTPIClinewidth@, xytpic.doc:316
\xyTPICmaxwarning@, xytpic.doc:84
\xyTPICoval@, xytp-f.doc:150
\xyTPICoval@i, xytp-f.doc:161
\xyTPICpartcircle@, xytpic.doc:343
\xyTPICqspline@, xytpic.doc:403
\xyTPICrect@, xytp-f.doc:126
\xyTPICrect@Special, xytp-f.doc:219
\xyTPICrect@x, xytp-f.doc:145
\xyTPICspline@, xytpic.doc:300
\xyTPICspline@dot, xytpic.doc:306
\xyTPICsplit@, xytpic.doc:137
\xyTPICstraight@, xytpic.doc:249
\xytracelineno@, xy.doc:474
\xytracing, xy.doc:4633
\xytracing , 65, 66
\xytracing@, xy.doc:4637
\xytriangle@@, xypoly.doc:839
\xytrigerror@, xypoly.doc:661
\xyuncatcodes, xy.doc:46
\xyuncatcodes , 4
\xyundecagon@@, xypoly.doc:1063
\xyundefinedEffect@, xyps-ps.doc:667
\xyundefinedLine@, xyline.doc:541
\xyundefinedRotate@, xyrotate.doc:53
\xyundefinedStyle@, xy.doc:4098
\xyunload@, xy.doc:6937
\xyunloadEM@, xyemtex.doc:125
\xyunloadTPIC@, xytpic.doc:215
\xyunsupportwarning@, xy.doc:6884
\xyunsupportwarnings@@, xy.doc:6898
\xyUseResizing@, xyrotate.doc:135
\xyUseResizing@@, xyrotate.doc:136
\xyvcap, xyknot.doc:1566
\xyvcross, xyknot.doc:410
\xyvcrossneg, xyknot.doc:416
\xyVECTOR@, xy.doc:1415
\xyverbose, xy.doc:4625
\xyverbose , 65, 66
\xyverbose@, xy.doc:4629
\xyversion, xy.doc:418
\xyversion , 13
\xyviitoi, xyv2.doc:670
\xyviitoii, xyv2.doc:671
\xyvloop, xyknot.doc:1617
\xyvover, xyknot.doc:544
\xyvtwist, xyknot.doc:683
\xyvtwistneg, xyknot.doc:686
\xyvuncross, xyknot.doc:422
\xyvunder, xyknot.doc:550
\xyvunover, xyknot.doc:556
\xyvuntwist, xyknot.doc:689
\xywarnifdefined, xy.doc:36
\xywarnifdefined , 3
\xywarning@, xy.doc:476
\xywidthchar@, xyline.doc:144
\xywith@@, xy.doc:6443
\xywithoption, xy.doc:6447
\xywithoption , 8, 90, 91
\xywithrun@, xy.doc:6457
\xywithtest@, xy.doc:6454
\xywrite@, xy.doc:698
\xyxbendd, xyknot.doc:1692
\xyxbendl, xyknot.doc:1690
\xyxbendr, xyknot.doc:1689
\xyxbendu, xyknot.doc:1691
\xyxcaph, xyknot.doc:1569
\xyxcapv, xyknot.doc:1568
\xyxoverh, xyknot.doc:569
\xyxoverv, xyknot.doc:563
\xyxunderh, xyknot.doc:572
\xyxunderv, xyknot.doc:566
\xyxunover, xyknot.doc:575
\xyxunoverh, xyknot.doc:581
\xyxunoverv, xyknot.doc:578
\xyxy@, xy.doc:1125
\xyxy@@ix@, xy.doc:879

\xyzbendh, xyknot.doc:1684
\xyzbendv, xyknot.doc:1687

Y, 20
y, 23, 38
\Y@c, xy.doc:612
\Y@max, xy.doc:637
\Y@min, xy.doc:635
\Y@origin, xy.doc:628
\Y@p, xy.doc:620
\Y@xbase, xy.doc:630
\Y@ybase, xy.doc:632
Y_{origin}, 21
Y_{xbase}, 21
Y_{ybase}, 21
Y_c, 21
Y_p, 21
$Y_{\max}$, 20
$Y_{\min}$, 20
\Yc, xyv2.doc:213
\Yc , 556
\Ymax, xyv2.doc:239
\Ymax , 556
\Ymin, xyv2.doc:237
\Ymin , 556
\Yorigin, xyv2.doc:230
\Yorigin , 556
\Yp, xyv2.doc:220
\Yp , 556
\Yxbase, xyv2.doc:232
\Yxbase , 556
\Yybase, xyv2.doc:234
\Yybase , 556

\zbendh, xyknot.doc:1644
\zbendv, xyknot.doc:1656
zero position, 20
\zerocorner, xyframe.doc:360
\zerodash@i, xycurve.doc:4683
\zeroDivide@, xy.doc:2470
\zeroDivide@@, xy.doc:2468
\zeroDivide@message, xy.doc:2469
\zeroDivideLimit, xy.doc:2478
\zeroDivideLimit@, xy.doc:2476
\zeroDivideLimit@@, xy.doc:2472
\zerodot, xy.doc:5321
\zerodot , 16, 76
\zerodot@i, xycurve.doc:4678
\zerodotbox@, xy.doc:654
\zeroEdge, xy.doc:7406
\zeroEdge , 48
\zeroit@, xy.doc:1448
\Zoom@, movie.cls:252
\zz@, xy.doc:660, 16