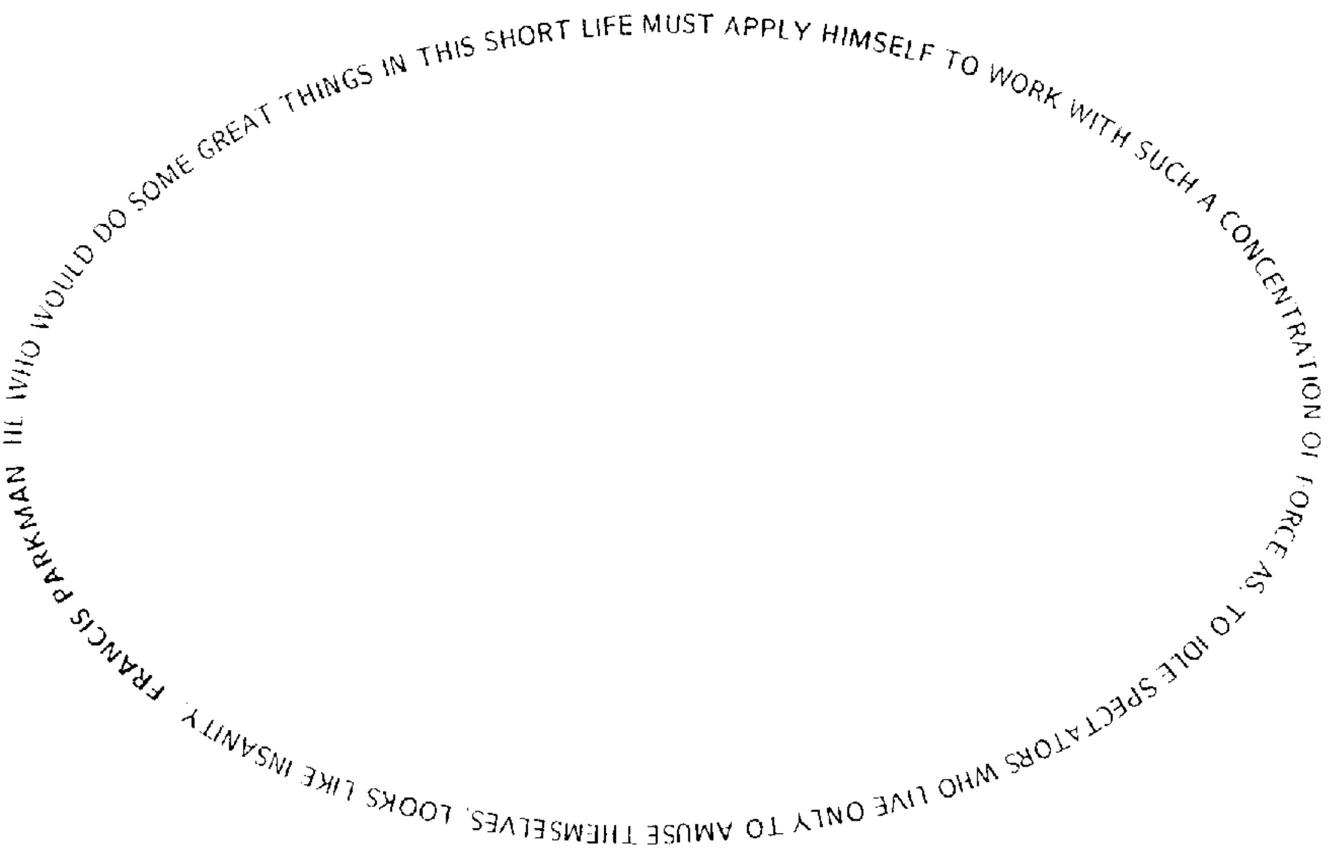

Figure 25: *Typesetting along a golden ellipse.*

Metafont rotates the type by modifying an underlying transformation called

```
currenttransform
```

which is a hook that Knuth provided for the purpose of doing last minute things to letters. It is helpful of thinking of a transformation as something that changes the shape, position, or orientation of a curve in certain allowable ways. You won't be surprised to learn that we can shift a curve up or down in position, and that we can rotate it, but we may also do a few other things such as

- uniformly magnify a curve;
- magnify a curve in the x- or y-directions separately;
- skew a figure; and
- perform any number of these all at once.

Transformations

- uniformly magnify a curve;
- magnify a curve in the x- or y-directions separately;
- skew a figure; and
- perform any number of these all at once.

Figure 26: *Legal transformations in* Metafont.

As long as we are rotating letters, perhaps we can apply other of these transformations to the letters of our text. For example, in figure 27 we have requested that Metafont shrink each letter at the same time it applies the rotation.

In the next figure, figure 28, you see what happens when we apply a transformation that does *not* include a rotation. Each letter has been skewed up or down so it conforms to the profile of the 'sunrise' curve. Actually, the base of each letter is straight, and coincides with an inscribed chord of the curve. (Transformations which apply curvature like that are generally not allowed in Metafont.) As long as the individual characters are so much smaller than the dimensions of the curve, viewers don't seem to mind this innocent deception.

5.6 Non-Linear Transformations in Metafont

It may be that Metafont is capable of making some limited non-linear transformations after all. I'd like to digress yet again to discuss this possibility.

Figure 27: *A new representation for* π.

A Metafont transformation is a collection of six constants which Metafont uses in specified ways to make the transformation. In figure 30, which I'll show presently, I needed to specify a transformation whose six constants were not constant but depended instead on the horizontal distance. For any given column of pixels, that meant that the transform was constant for all the pixels in that column, but would change a bit when we proceeded to the next column.

Figure 28: *Along a* Metafont *sunrise.*

I stored the original, untransformed object as a Metafont picture and I made the transformations on a pixel-by-pixel level. The transformed image is stored in yet another picture variable.

---

1. Update transformation for new column.
2. Pixel by pixel, find out where 'ink' starts and stops.
3. Find the transforms for 'start' and 'stop' values.
4. Continue on that column.
5. Next column.

Figure 29: *Performing non-linear transformations.*

1. Prepare for examining a new column by updating the transformation for that column.
2. Start examining the column from the bottom up. Look for the first pixel in that column that is black. Keep examining pixels until the pixels are clear. At this point, we have a pair of numbers, the first of which records the point at which the pixels are turned on, and the second the point at which they are turned off.
3. Find the transforms of those values. In the image picture, turn on the pixels for that column between these two values.
4. Continue analyzing that column.
5. Continue on to the next columns.

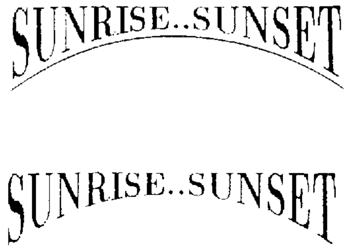

Figure 30: *Nonlinear* Metafont *transformations.*

Figure 30 shows the result of one non-linear transformation. Metafont is not happy at working at the pixel level; it takes about five minutes to generate these images on a 386 PC. (Normally, Metafont could create comparable text in a matter of seconds.)

If you look carefully, you'll see that the boundaries of the transformed letters are more jagged than you expect. This is work in progress, and I hope that this embarrassing state of affairs can be easily corrected.

5.7 An Application for Curved Typesetting

The only real application I can think of for curvilinear typesetting is for setting text around the circumference of a circular university or institutional seal. Despite the simplicity of the path—a mere circle, compared to some of the non-standard curves we have looked at—there are a number of interesting problems that need to be solved in connection with this application.

---

1. Need to center text.
2. Need to alter kern values.
3. Need to set bottom part along a concave path!

Figure 31: *Problems for seals.*

- The text needs to be *centered* with respect to the circumference. That is, it must be properly positioned along the seal.
- Unlike "standard" curvilinear typesetting, we explicitly *do* need to include the kerns between characters. The text of the seal might not look right otherwise. Furthermore, we may need to be able to alter some of the kerns between characters (because