# Computers and Typesetting

Donald E. Knuth

1999

With a suitable setting of the parameters, we can even imitate a typewriter with its fixed width letters, as shown in Figure 13e. There is also a provision to slant the letters as in Figure 13f; here the pen position is varied, but the actual shape of the pen is not being slanted, so circular dots remain circles.

---

Finally, Figure 13i illustrates the variations you can get by giving weirder settings to the parameters. **In other words, there is a unique tangent at every point of the curve.** *This chapter and the next are for readers who share my fascination with original source documents.* ***The draft report I wrote during the first third of May was entitled TEXDR.AFT***

# **1.** First part

Before getting into a technical discussion, I should probably mention why I started worrying about such things in the first place. The central reason is that today's printing technology is essentially based on discrete mathematics and computer science, not on the properties of metals or of movable type. The task of making a plate for a printed page is now essentially that of constructing a gigantic matrix of 0s and 1s, where the 0s specify white space and the 1s specify ink. I wanted the second edition of one of my books to look like the first edition, although the first edition had been typeset with the old hot-lead technology; and when I realized that this problem could be solved by using appropriate techniques of discrete mathematics and computer science, I couldn't resist trying to find my own solution.

## **1.2.** Sub-sect

By studying this example we can get some idea of the problems in- volved in specifying a proper S shape. However, I was actually seeking the solution to a more general problem than the one Torniello faced: Instead of specifying only one particular S, I needed many different vari- ations, including bold face S's that are much darker than the normal text.I discussed this recently with Alan Perlis, who pointed out that a central issue arising whenever we try to automate something properly is what he calls "the art of making constant things variable."

In the case of letter design, we don't merely want to take a particular drawing and come up with some mathematics to describe it; we really want to find the principles underlying the drawing, so that we can generate infinitely many drawings (including the given one) as a function of appropriate parameters.

After looking at these Renaissance constructions and a lot of mod- ern S shapes, I came to the conclusion that the main stroke of the general S curve I sought would be analogous to the curve in Figure 6; each boundary curve was to be an ellipse followed by a straight line followed by another ellipse. This led me to pose the following problem: What ellipse has its topmost point at (xt, yt) and its leftmost point at (xl, yl) for some yl, and is tangent to the straight line of slope  that passes through (xc, yc), given the values of xt, yt, xl,  , xc, and yc? (The ellipse in question is supposed to have the coordinate axes as its major and minor axes; in other words, it should have left-right sym- metry. See Figure 7 on the next page.) The reason for my posing this problem should be fairly clear from our previous discussions: We know a point that is supposed to be the top of the S curve, and we also know how far the curve should extend to the left; furthermore we have a straight line in mind that will form the middle link of the stroke.

# **2.** Second part

After looking at these Renaissance constructions and a lot of mod- ern S shapes, I came to the conclusion that the main stroke of the general S curve I sought would be analogous to the curve in Figure 6; each boundary curve was to be an ellipse followed by a straight line followed by another ellipse. This led me to pose the following problem: What ellipse has its topmost point at (xt, yt) and its leftmost point at (xl, yl) for some yl, and is tangent to the straight line of slope that passes through (xc, yc), given the values of xt, yt, xl, , xc, and yc? (The ellipse in question is supposed to have the coordinate axes as its major and minor axes; in other words, it should have left-right sym- metry. See Figure 7 on the next page.) The reason for my posing this problem should be fairly clear from our previous discussions: We know a point that is supposed to be the top of the S curve, and we also know how far the curve should extend to the left; furthermore we have a straight line in mind that will form the middle link of the stroke.

The problem stated in the preceding paragraph is interesting to me for several reasons. In the first place, it has a nice answer (as we will see). In the second place, the answer does in fact lead to satisfactory S curves. In the third place, the answer isn't completely trivial; during a period of two years or so I came across this problem four different times and each time I was unable to find my notes about how to solve it, so I spent several hours deriving and rederiving the formulas whenever I needed them. Finally I decided to write this paper so that I wouldn't have to derive the answer again.

These four simultaneous linear equations in x, y, , and are easily solved; and in fact METAFONT will automatically solve simultaneous linear equations, so it is easy to compute the intersection of lines in METAFONT programs.

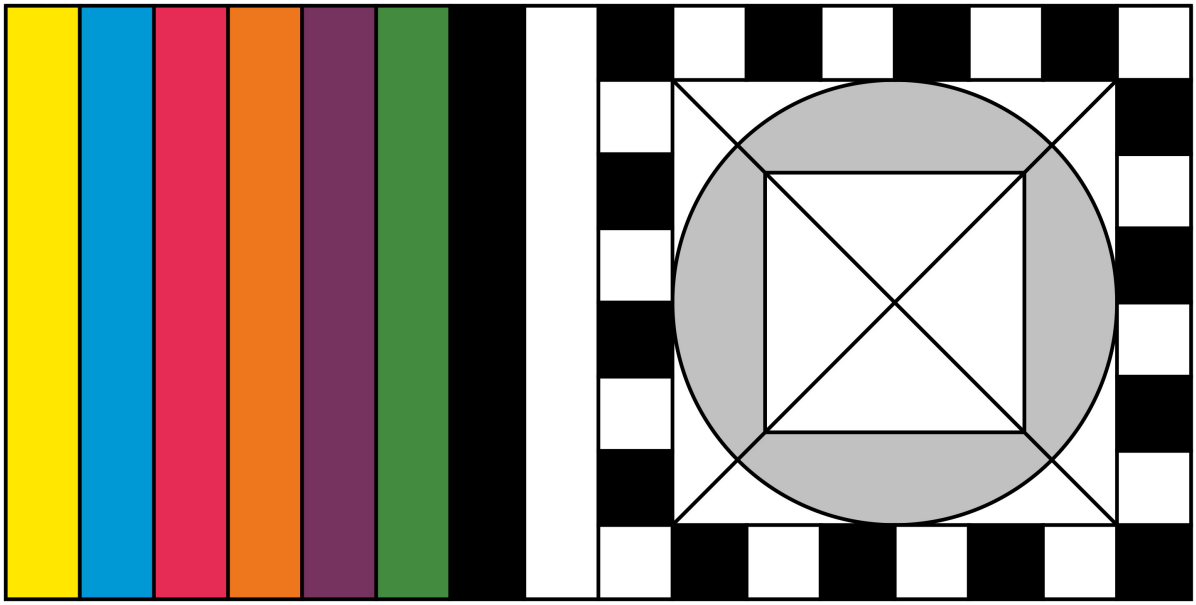

Figure 1: Look at this "calibration image"!!!!

It turned out that it was not hard to achieve this level of quality with respect to the formatting of text, after about two years of work.

1. A list item, WOW
2. A second thing
3. We are up to threeee
4. This is the last one :(

When I looked at them, I was sure that there must be a bug either in IDEAL or in the troff processor that typeset the IDEAL output, be- cause the long shafts of the arrows did not properly bisect the angle made by the two short lines of the arrowheads. The shafts seemed to be drawn one

pixel too high or too low. Chris spent many hours to- gether with Brian Kernighan trying to find out what was wrong, but no errors could be pinned down. Eventually his thesis was printed on a high-resolution phototypesetter, and the problem became much less noticeable than it had been on the laserprinted proofs. There still was a glitch, but I decided not to hold up Chris’s graduation for the sake of a misplaced pixel.

| Some text, like a really long amount of text. This is just to force a linebreak. | Some more text | Even more |
|---|---|---|
| Some text 2 | Some more text 2, there is quite a lot of \| text \| | Even more 2 |
| Some text 3 | Some more text 3 | Even more 3 |

Laboratory Memo AIM-332 (September 1979), 105 pp.; then it was reprinted on a Penguin, with minor corrections, in the Digital Press book mentioned earlier.

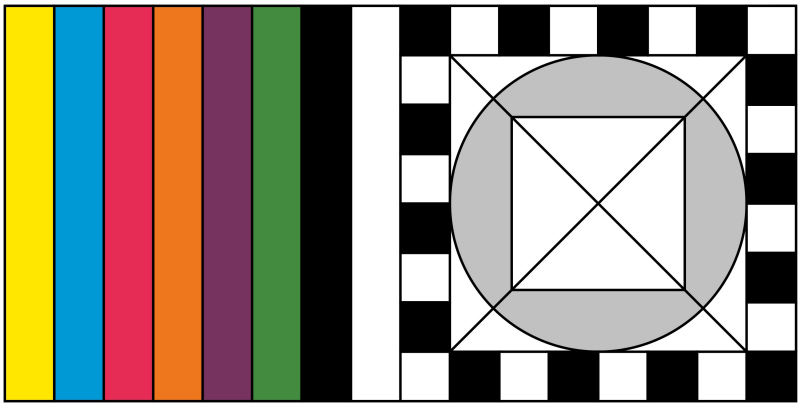

Figure 2: Look at this "calibration image"!!!!

Let me state at the outset that I do not foresee the problem ever becoming simple. Indeed, when I ponder what lessons I have learned so far, the main lesson turns out to be that the problem is quite difficult! In a way, this is not surprising. For more than thirty years, computers have been oversold by salesmen who claim that computing machines are easy to use, while the truth is quite the opposite: Computer program- ming requires more attention to detail than anything else that human beings have ever done. Moreover, the problems of letterform design are extremely subtle, much more complex than most people think, because

our machines and our eyes interact with the shapes in complicated ways. I am convinced that digital alphabet design is an extremely challenging problem, and that it is significant enough to deserve the attention of our best scientific minds and our best artistic skills and sensitivities. Fur- thermore, I believe that the world will be a better place to live in after we learn more about the subject.

```
typesetDocument :: ValidatedDocument -> ResourceMap -> LoadedFonts -> FilePath
-> Bool -> IO ()
typesetDocument (ValidatedDocument cfg meta cnt) res fonts outPath dbg = do
let rConfig = toRenderConfig cfg

let pageRect@(PDFRect _ _ px py) = pageSizeToRect (fromJust $ cfgPageSize cfg)
let (Pt topMargin) = rcVertMargin rConfig
let startY = py - topMargin

-- Get bottom margin based on page numbering.
let bottomMarginRaw = topMargin
let bottomMargin = bottomMarginRaw + case rcPageNumbering rConfig of
NumberingNone -> 0.0
_ -> 10.0
```