Designing a new typeface with METAFONT

Richard Southall
Laboratoire de typographie informatique
Université Louis-Pasteur
67084 Strasbourg, France

Abstract

This paper summarizes the conclusions reached as a consequence of two years' work on the design of an original typeface using METAFONT. While being in one sense unsuccessful, in that the design of the typeface is still far from complete, the experience has been instructive in pointing up a number of discrepancies between the underlying assumptions about the design process that professional type designers bring to their work and those current in the TEX world.

These discrepancies, and what appear to be the reasons for them, are discussed.

1 Terminology

The computer science literature on document preparation has traditionally been bedevilled by the misuse of terms carelessly borrowed from the existing technologies of typography and type manufacture. The word *font*, for example, is used indifferently in the literature to refer to three or four fundamentally distinct entities: this leads to a good deal of confusion in discussions about typefaces, type design and similar subjects. So the first thing we need to do is to define a useful and consistent terminology[1].

1.1 Typefaces and character images

A *document* is an assembly of mechanically-produced marks on a substrate. The marks that make up the verbal components of the document are *character images*, and it is these images that the reader sees. We need to define terms that allow us

[1] This terminology is a shortened form of the fully-worked-out version proposed in *Designing new typefaces with METAFONT* (Southall, 1985b).

to discuss how character images are produced, and what part the type designer plays in deciding on their appearance.

A *script* is a set of characters used to write one or more languages.

A *typeface* is a set of distinctive, visually related shapes that represent some or all of the characters of a script and are intended for mechanical reproduction.

Each of the character shapes in a typeface has an *identity*, which is that of the character it represents. The typeface as a whole – the set of shapes with different identities – has a number of *visual attributes*. It is the visual attributes of a typeface that distinguish it from other typefaces.

The visual attributes of a typeface are of two kinds: *stylistic* and *functional*. Stylistic visual attributes are such things as *seriffedness* and *cursiveness*; functional visual attributes are such things as *boldness* and *condensedness*. Typeface classification schemes such as DIN 16 518 : 1964, and aesthetic criticism of the traditional kind, deal with the stylistic visual attributes of typefaces. The functional visual attributes of typefaces are those that make different typefaces useful in differentiating the components of complex text.

A *family of typefaces* is a set of typefaces with similar stylistic visual attributes and differing functional visual attributes[2].

The character images in a document have *graphic attributes* that give them their *appearance*. The appearance of character images realizes the visual attributes of the typeface the images represent.

1.2 Fonts

The character images (as well as the other marks) in a mechanically-produced document are made by a *marking device*. This contains a *marking engine*[3] that uses a *marking process*. Thus the character images in the document that constituted the camera-ready copy for this paper were made by a Canon CX-2000 laser marking engine. This was inside a Canon LBP-8 A1 laser printer driven by a Telmat SM90 minicomputer: these two machines together made up the marking device. The marking process was the familiar electrographic one, in which a laser writes on the surface of an electrically-charged semiconducting drum which then transfers toner to the surface of plain paper.

---

[2] On this definition, the Computer Modern family of typefaces (Knuth, 1980) is a long way from being the traditional nuclear family of roman, italic and bold. The stylistic attributes that relate the sanserif or the `typewriter face` to Computer Modern Roman are not at all easy to recognize, except in some features of certain characters.

[3] The term *marking engine*, like much else that has helped to clarify thinking in this field, is due to Brian Reid.

An engine like the Canon CX-2000 has no knowledge of its own about the shapes of character images. Somewhere inside the marking device there need to be sets of instructions to the marking engine that tell it how to make the images that realize the character shapes of a typeface in a particular size or range of sizes. These sets of instructions are the *fonts*.

Thus, in the TEX–METAFONT system, there is the Computer Modern family of typefaces, of which `cmr10` is a member. The character shapes of `cmr10` are described in `cmr10.mf` and its associated METAFONT programs. Running these programs with `mode=imagen` and `mag=1` produces a 'generic font file' `cmr10.300gf`: this is not a font, but an intermediate in the font production process. Running `gftopxl` on `cmr10.300gf` produces the pixel file `cmr10.1500pxl`. This *is* a font: it contains instructions to the Canon printer (which uses the same marking engine as the Imagen printer for which the font was developed) about how to produce character images that realize the visual attributes of the typeface `cmr10`.

Notice that the font is *device-specific*: not only is it made for a particular writing resolution (300 dots per inch in this case: the fact that the extension of the font file name is `.1500pxl` is a peculiarity of the system) but the `mode` information that the programs read assigns values to the METAFONT variables `blacker`, `fillin` and `o_correction` that are appropriate for the Canon engine. These variables would be assigned different values if the font were being made for the Xerox 1200 marking engine, which has the same writing resolution of 300 dots per inch as the Canon but very different marking characteristics.

2 What is design?

In order to be able to evaluate METAFONT's usefulness as a typeface design tool, we need to find out what is involved in designing a typeface. Before doing this, it is important to get a clear idea of what the word 'design' implies in this context.

Design, like *font*, is a word whose meaning in the vocabulary of computer science tends to be rather different from the meaning it carries in everyday life. In normal use, at least among designers, *designing* means making something new: in the case of a typeface, a new set of shapes for the characters of a script, that are not the same as any set of shapes for the same characters that have existed before. Used in relation to computer-produced documents, the word often appears to mean more or less exactly the opposite: the design activity seems to be intended to produce something that resembles an existing object or set of objects as closely as possible.

In this paper, I use *adaptation* for the second of these two meanings, and keep *design* for the first. Thus I would consider Computer Modern Roman to be an

adaptation of the Lanston Monotype Company's Modern Series 8A (Knuth, 1980, p. 2), and the letters in the METAFONT logo to be an original *design.*

The important difference between design and adaptation, in the context of a discussion about type production, is that in the second case the typeface already exists: the adapter does not have to decide what shapes the characters ought to be. Nor is this all: character images exist as well, that arise from fonts that have been fully developed to be technically satisfactory, even if for use in another medium. The results of all the decision-taking that went into the design of the original typeface and the production of the original fonts are available in the material from which the adapter works. It is not surprising that both the nature and the amount of the work involved in type design and font production is misapprehended, if the difference between design and adaptation is misunderstood.

3 Designing a typeface

The end product of type manufacture is a series of fonts.

To be worth making, these fonts must be of good technical quality – that is, they must give rise to sets of character images that are of good technical quality. To be useful, the fonts must give rise to character images that realize the visual attributes of a particular typeface. The type manufacturing process thus has two parts: one in which the visual attributes of the typeface are decided upon, and another in which the fonts that cause these attributes to be realized in character images are produced. What goes on in the first part of the process is *type design*; what goes on in the second part is *font production.*

In traditional type manufacture (which means, in the context of the current discussion, in the era before METAFONT), these two parts of the manufacturing process have usually been the responsibilities of different people. The *type designer* defines the visual attributes the typeface is to have; the *font producer* makes fonts that give rise to character images whose graphic attributes gives them an appearance that realizes the visual attributes of the typeface.

3.1 Type designer and font producer

It is the type designer's task to decide on the appearance of the typeface, and to characterize its appearance by defining its visual attributes. The designer's decisions are almost always expressed in the form of drawings of characters, whose shapes realize the visual attributes the typeface is intended to have.

It is very important to understand the role that the designer's drawings play in the type production process. Except in a very few cases, they are not *patterns*

that tell the font producer what shapes the character images ought to *be*: they are *models* that show the producer what the images are intended to *look like*.

This is because the font producer does more than copy the designer's drawings. The font producer's objective is to make fonts that give rise to images of good technical quality that realize the visual attributes of the characters on the designer's drawings. The instructions contained in the font have to take into account the effects of the human visual system on the way the character images are perceived, as well as the effects of the marking process on the shapes of the images themselves.

The effects of the visual system

It is easy to state the criteria for good technical quality in a set of character images: they should be consistent in apparent size, weight and spacing. This consistency is in the *appearance* of the images, not necessarily in their *shape*; in order to appear to be consistent, the actual shapes of the images should allow for the visual effects that occur when they are perceived.

The essential feature of the character images in text is that they are small: one way of looking at the history of type manufacture is to see it as the development of techniques for the rapid multiplication of accurately-defined small shapes. In the perception of small shapes, the human visual system has an effect on what is perceived that is quite different from the effect it has in the perception of large shapes: adjacent parts of a small shape interact with each other on their way through the system. Thus, for example, a square looks larger than a circle whose diameter is the same as the side of the square; parallel-sided strokes that meet at an acute angle look as if they get wider as they approach each other. The effects of these visual phenomena on the perception of small character shapes are discussed by Harry Carter (in footnotes in his edition of Fournier's *Manuel typographique* as well as in his 1937 paper) and by T. L. De Vinne (Carter, 1930, 1937; De Vinne, 1900).

The effects of the marking process

It often seems to be supposed that the effects of the marking process ceased to be important when digital type-composing techniques were introduced. This is far from being the case. One has only to compare output from the Canon CX-2000 and the Xerox 1200 electrographic marking engines, already mentioned, to see how much the differences in present-day marking processes can affect the shapes of the character images they produce. The simple picture that Knuth described in his Gibbs lecture in 1978, of the page of a book as 'a huge matrix of 0's and 1's' (Knuth, 1979: *Mathematical typography*, p.16), omits an important consideration: neighbouring 1's interact with each other, on the physical as well as on the perceptual level[4].

[4] Knuth has subsequently advanced from this elementary view (Knuth, 1985).

The font producer's task

Because the marking process has an effect on the shapes of the character images the font gives rise to, the shapes of the images are different from the shapes that are represented in the font. Equally, because they have to take into account the effects of the human visual system, the shapes of the character images (and, *a fortiori*, the shapes represented in the font) are not the same as the shapes of the characters on the designer's drawings. Because the effects that enter into the perception of small and large shapes are different, large shapes change their appearance when they are reduced: a small shape that has the same visual attributes as a large shape will not be simply a smaller version of it. Thus the production of technically satisfactory fonts involves the font producer in *interpreting* the designer's drawings, rather than simply *reproducing* them; and because different marking processes affect the shapes of character images in different ways, this interpretation has to be done differently for each marking process for which fonts are being produced.

The designer's drawings

The shapes on the drawings that the designer gives to the font producer are not made in a single pass, but are themselves the subject of considerable development work (Dwiggins, 1940). It is not until they are consistent in appearance, and express a clear intention on the part of the designer, that the designer's drawings are useful to the producer. (The kind of problems that arise when drawings are given to the font producer before this clarity of intention has been achieved are vividly described in John Dreyfus' account of the production of the Cranach Press italic: Dreyfus, 1966).

In the past, the typeface character shapes on the drawings have been developed empirically, by a process that is also an essential part of font production: iterative testing and modification. Since the shapes are modified by working over them by hand, and the testing usually done in the early stages by putting the drawings up on the wall and looking at them, the designer's activity tends not to look like a process in the usual sense of the word; consequently, the fact that it *is* a process, with certain characteristics that are important in making it effective, is easy to overlook. These characteristics are a high degree of interactivity in the modification phase, and a short cycle time in the testing phase.

3.2 The importance of empirical testing in type manufacture

The knowledge that type designers and font producers have had, both about the effects of marking processes and the characteristics of the human visual system, has mostly been intuitive and qualitative rather than explicit and quantitative. Even though the ways in which marking processes affect the shapes of character images are fairly easy to understand, the extent to which a particular process

will affect the shape of a particular image is very hard to predict. Similarly, the kinds of visual effect that occur in the perception of character images are well known, but the way in which they will affect the appearance of a particular shape is difficult to tell without making the shape and looking at it. The development of technically satisfactory fonts for a new typeface, like the development of the designer's drawings, has therefore traditionally been carried out by iterative modification of the font and empirical testing of the character images it produces; and what is tested is essentially the appearance of the images rather than their shapes.

It needs to be made clear, perhaps especially to an audience from the computer science community, that the empirical nature of the type manufacturers' working methods is not due to technical backwardness, or to an anti-technological attitude, on their part. The fact is that not enough is yet known about the human visual system to enable us to construct a theory that will predict exactly, from the shape of a character image, what its appearance will be. Nor is our knowledge of the characteristics of any marking process sufficiently detailed to allow us to predict exactly what shape a particular set of instructions in a font will give rise to. In these circumstances, empirical methods are the only ones that are effective for font development, if the visual quality of the product is to be maintained.

3.3 What does a type manufacturing system need?

The picture we get of type design and font production in the pre-METAFONT era, then, is one in which the subject-matter of the work is the appearance of character images; where communication between designer and producer is carried on by means of exchanges of graphic objects; and in which the desired appearance of both drawings and character images is arrived at by empirical testing and modification, using processes that are at their best when they are most interactive[5].

Since this picture is so unlike the one that METAFONT offers us, we need to ask which of its features are essential to a successful type manufacturing system: that is, one that produces fonts of good technical quality that give rise to character images having the required appearance. It is hard to answer this question by looking at unsuccessful type manufacturing systems from the past, because such systems have not usually survived or been described in published work[6].

What we can say is that type manufacturing systems have produced good results, and have been congenial to the designers working with them, to the extent that the designers have felt themselves to be in control of the appearance of the character images that were the final product of the system. The reservations expressed by the Dutch designer Jan van Krimpen about the success of his work for the

[5] A more extensive justification of this picture is in Section 3 of *Designing new typefaces with METAFONT* (Southall, *op. cit.*).

[6] Oddly enough, the situation with unsuccessful type-composing systems is quite different: these have an extensive and easily accessible literature.

Monotype Corporation are significant in this respect (van Krimpen, 1972). In this respect, too, it is interesting that recently, where electronic systems have allowed designers to work directly on the pixel grid (that is, in our terms, to be their own font producers), the designers have grasped the opportunity with enthusiasm. Examples of the success of this working method are the fonts used for telephone directory composition in France and the United States of America, designed by Ladislas Mandel and Matthew Carter respectively (Cooper Union, 1982).

4 METAFONT as a design tool

4.1 Making type with METAFONT

In a type manufacturing system that uses METAFONT, the shapes of characters are described by programs written in the METAFONT language. The METAFONT interpreter reads these programs and produces a 'generic font file': this is essentially a set of run-length encoded descriptions of character bitmaps at a particular resolution. For each run, the interpreter needs to know the writing resolution of the device for which the output of the run is intended, as well as other information about the device: it finds this out by reading preloaded `mode` information. Fonts are made from the generic font files by the `gftopxl` or `gftopk` programs, which are part of the 'METAFONTware' software.

Knuth's view of his objectives for the METAFONT system seems to have remained essentially unchanged between 1978 and 1985. 'One of my main motivations was the knowledge that the problem would be solved once for all, if I could find a purely mathematical way to define the letter shapes and convert them to discrete raster patterns ... although the precision of the raster may change, the letter shapes can stay the same forever, once they are defined in a machine-independent form' (Knuth, 1979: *Mathematical typography*, p. 17). 'We now have the ability to give a completely precise definition of letter shapes that will produce essentially equivalent results on all raster-based machines' (Knuth, 1986, p. v). The new implementation of METAFONT has solved most of the rasterizing problems that troubled the old system: the technical quality of the new Computer Modern on medium and low resolution marking devices is enormously improved.

In making Computer Modern, METAFONT is playing the role of font producer. Each size of each typeface in the Computer Modern family has a driver program (`cmr10.mf`, for example). This program sets the values of a large number of dimensional parameters for the typeface, and then reads a series of programs (`roman.mf`, `romanu.mf`, `romanl.mf` and so on) that describe the character shapes in terms of the parameters that have been set by the driver program.

This is where the 'meta-ness' comes in: the same programs, read by different driver files, produce characters of very different appearance (from five point roman to **ten point sanserif bold extended**, `typewriter face` and so on) according to the settings of the parameters.

There are a few parameters (`blacker`, `fillin` and `o_correction`) that go some way towards characterizing the marking process used by the marking engine for which the font is intended. The values for these parameters are part of the `mode` information that is read at the beginning of the run.

Knuth does not exclude the possibility of interactive modification of the fonts that METAFONT produces, but sees this only as a 'tidying-up' expedient (Knuth, 1986, p. 195).

4.2 Designing with METAFONT

Knuth's Computer Modern is an example, extraordinarily fully worked out, of the adaptation of a set of existing typeface designs to the METAFONT system. Designs that have been produced with the old version of METAFONT by other workers (Tom Hickey's CHEL, for example, or Georgia Tobin's MF Roman) are also adaptations. To find out about METAFONT's useability as a design tool, we need to find instances of its use for original design.

The designer working with METAFONT has two options. One is to make a set of fully-developed drawings of character shapes in the traditional way; write programs, or cause programs to be written, that describe the shapes on the drawings; and then alter the programs or cause them to be altered until the output they produce is acceptable. While not being exactly the same as the adaptation of an existing design (because fonts derived from the drawings do not yet exist) this way of working is something of a halfway house between design and adaptation. The task of the programmer, or of the designer in the programming phase of the work, is to express in METAFONT's terms the character shapes that already exist on a set of drawings, rather than to develop the shapes using METAFONT itself.

The second option before the designer working with METAFONT is to exploit the characteristics of the system: to use METAFONT itself to help develop the character shapes.

The Euler project

The first of these two options is the one that Hermann Zapf adopted in his design of the Euler typeface for the American Mathematical Society. Zapf made the drawings, and the Digital Typography Group at Stanford wrote the METAFONT programs. In the Euler project, Zapf was treating the Digital Typography Group as a font production team of the traditional kind; the problem that faced the programmers in the group was essentially to teach METAFONT how to do a competent job in its role as font producer.

This turned out not to be particularly easy. The new version of the language had not been conceived when the Euler project began, and the initial phase of describing the character shapes on Zapf's drawings in terms of the virtual pens of the old METAFONT presented great difficulties. The results of the first attempts were rejected by the designer; a program was then developed that allowed the characters' outlines to be described to the computer by means of a digitizing tablet. Writing such a program was a great deal more difficult with the old METAFONT than it would have been with the new, because of the change of emphasis from pen tracks to character outlines in the new version.

The story of the Euler project is told by David Siegel, a member of the Digital Typography Group (Siegel, 1985).

The nmt design

In making the nmt design with METAFONT, I decided to take the second of the two options described above, and work as closely as possible with the computer. This was partly due to the fact that I had just come from using a computer-based font design tool that had many of the virtues (in speed and interactivity) and all the defects (in terms of the lack of generality of the product) of font design systems in general; partly because the studio facilities that would have been needed to make fully-developed character drawings were not easily available in the Computer Science Department at Stanford.

The design was begun in November 1983, using the old version of METAFONT. This early work was abandoned in February 1984; the overall form that the new version of the language would have was becoming clear by that time, and there was evidently no point in struggling to define the outlines of character shapes by means of the tracks of virtual pens whose centres were offset from the outlines, when it would soon be possible to define the outlines directly.

Work on the design was begun again in the summer of 1984, during the METAFONT course at Stanford (Knuth, 1984); suspended between September 1984 and April 1985, while I was in Europe; continued, with a very much developed version of the language, between April and October 1985 at Stanford, and then at Strasbourg until the beginning of June 1986.

It can be seen from this chronology that nmt and the new version of METAFONT have grown up together. This has had its disadvantages. Getting on with the design has tended to take priority over updating the low-level METAFONT routines in the character programs to take advantage of improvements to the language. The consequence is that my programs, on the whole, neglect facilities that have been added to the language and to `plain.mf` since the beginning of August 1985. (It was not until the beginning of June 1986 that version 1.0 of METAFONT was installed at Strasbourg.)

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

abcdefghijlmnopqrstuvw
nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnunvnwn
oaobocodoeofogohoiojolomonooopoqorosotouovowo
hamburgefons

Figure 1: nmt *in nominal sizes from 3 to 10 pt, reduced from* `\magstep2`.

It should also be pointed out that the nmt programs do not use the pseudo-pens of new METAFONT. Some of the problems I had with getting the stroke-drawing routines to behave well at low resolution would probably have been avoided if I had used these pens; on the other hand, I cannot see at present how to use them to make what I consider to be technically important features of certain characters, particularly small v and w.

The main motive behind the development of nmt has been to make METAFONT do as good a job as possible of font production for actual marking devices. There are two main reasons for this. In the first place, the technical quality of the Computer Modern fonts that were available for medium-resolution devices in 1983, when I began work on the design, left a great deal to be desired. I felt it was important to find out whether it was possible to make a technically satisfactory medium-resolution font with METAFONT, since no-one had succeeded in doing so at that time.

In the second place, I felt that a type design method that was aimed at an ideal marking device ran the risk of giving every user of a real marking device more or less of a bad deal. It seemed to me that the interests of users, in particular those of users of medium-resolution marking devices, were being neglected in favour of an approach to design that ignored the characteristics of marking processes that did exist while aiming at an ideal process that did not exist.

I have also taken the view that one should not impose too high a lower limit on the writing resolution of the devices for which a design is intended. The resolution of a 'high-performance' cathode-ray tube display, in terms of the number of addressable points along the line, is after all no more than that of an Epson FX80 printer in graphics mode. What characters there are in nmt perform reasonably well down to 7 pixels x-height: this corresponds to a nominal size of 2.88 pt or 1.012 mm on the Canon printer, and 5.78 pt or 2.031 mm on the Numelec bitmap terminal.

4.3 Font optimization and 'device-independent' design

In making a technically satisfactory font for a medium-resolution raster-scan marking device, every feature of the character shapes has to be conceived of in terms of the characteristics of the marking process the device uses and the size and shape of the the pixels it produces. The device's pixels are used as building blocks for the character shapes.

With this approach, it is relatively easy to achieve the evenness of apparent weight and spacing of the character images that are important in a technically satisfactory design. On the other hand, though, the character widths become completely device-dependent: there is no device-independent '`tfm` width' that expresses the width of a character in absolute terms.

It is hard to see how device-independent widths for the characters of an original design can be arrived at, if the objective in making the design is to optimize the performance of the fonts it produces. In `plain.mf`, the `beginchar` macro expects the device-independent width and height of a character to be known at the beginning of the program that describes it (Knuth, 1986, p. 275). This seems to me to reflect the conceptual confusion between adaptation and design that I have already mentioned. In adapting an existing typeface, the absolute dimensions of the characters are indeed known before the adaptation is begun, and there is no problem in incorporating them in the programs. In making an original design, on the other hand, where the objective is to optimize font performance, the only way to develop the character shapes is to look at the marks made by particular marking devices. In doing this, it is difficult to describe the characters' dimensions otherwise than in terms of the dimensional units that those devices use. The dimensions of each character become consequences of the way the character is constructed for the marking device in question, and hence are not known at the beginning of the character program.

The question whether or not it is possible to make technically satisfactory fonts for use on medium-resolution devices from designs whose character widths are defined in device-independent terms is an interesting one for the TEX community.

The design of every such font has to begin with the definition of a set of device-dependent widths for the characters of the typeface. These widths need to be allocated in such a way that the cumulated differences in character positioning with respect to the device-independent widths amount, over the length of an average word, to less than about a quarter of the average interword space. If this can be done, the characters within a word can be properly spaced without the interword spacing becoming too irregular.

The difficulty is, of course, that both the length of the average word and its composition, in terms of the frequencies of occurrence of different characters, vary between one language and another. A set of character width allocations that produce good results in French text will not work so well in English or in German text. In the German text, also, because the words tend to be longer, the cumulated differences in character positioning will be greater and there will be fewer interword spaces to absorb them.

I cannot see any way out of this difficulty at present.

4.4 Meta-ness in nmt

In the Computer Modern family of typefaces, the specification of each size of each typeface in the family begins with the assignment of explicit values, most of them absolute dimensions, to a large number of variables in the character programs (sixty-two in `cmr10.mf`). This approach is entirely appropriate for a situation of

he eighteenth centur was also something more it was and
above all in rance the nurser of the modern world deas
and social forces the seeds of which were doubtless sown much
earlier can be seen now pushing above the surface not in the
neatl arranged rows of the careful gardener but in the haphaard
tangle of nature et the can be seen and distinguished the
field is no longer a seedbed but it is not et a jungle and a
pattern is discernible

he eighteenth centur was also something more it was and
above all in rance the nurser of the modern world deas
and social forces the seeds of which were doubtless sown much
earlier can be seen now pushing above the surface not in the
neatl arranged rows of the careful gardener but in the haphaard
tangle of nature et the can be seen and distinguished the
field is no longer a seedbed but it is not et a jungle and a
pattern is discernible

he eighteenth centur was also something more it was and
above all in rance the nurser of the modern world deas
and social forces the seeds of which were doubtless sown much
earlier can be seen now pushing above the surface not in the
neatl arranged rows of the careful gardener but in the haphaard
tangle of nature et the can be seen and distinguished the
field is no longer a seedbed but it is not et a jungle and a
pattern is discernible

he eighteenth centur was also something more it was and
above all in rance the nurser of the modern world deas
and social forces the seeds of which were doubtless sown much
earlier can be seen now pushing above the surface not in the
neatl arranged rows of the careful gardener but in the haphaard
tangle of nature et the can be seen and distinguished the
field is no longer a seedbed but it is not et a jungle and a
pattern is discernible

Figure 2: nmt *in 4, 5, 6 and 7 pt, reduced from* `\magstep2`. *The change in 'colour' between the 4 and 5 pt fonts is marked: between 5 and 6 pt or 6 and 7 pt less marked, but still visible.*

a posteriori meta-design, in which the appearance of a large number of existing fonts is to be matched by the output from a single set of METAFONT programs.

I have expressed elsewhere my reservations about the practicability of original meta-design (Southall, 1985a, 1985b). Further experience with nmt has given me little reason to modify my earlier views. The problem is still one of defining, and then testing, the relationships between the parameters that operate at the character level to change the shapes of character images and the typeface-wide parameters that affect the size or the appearance of the typeface.

The nmt programs have three parameters: size, boldness and expansion (the two latter as yet more or less untested). These parameters are intended to be continuously variable between limits. Thus, for example, users can set any character size they like, between upper and lower limits that depend on the writing resolution of the eventual marking device, in increments that likewise depend only on the device resolution. The sort of problems that this approach gives rise to, and that remain to be resolved, are demonstrated by the abrupt steps in typographic 'colour' that occur in the small sizes of nmt when the vertical stroke weight changes by one pixel.

The fact that defects of this kind have survived so long into the development of the design provides a further illustration of my contention about the impracticability of original meta-design. The design of every font involves the designer in a great number of decisions. With METAFONT, these decisions cannot be made and implemented visually, as they can with a font design tool, but have to be incorporated into the character programs. Because there is no theory that allows the correctness or otherwise of such decisions to be predicted in advance of seeing their results, each of the programs that embodies them has to be tested, by the production of actual fonts, over the whole range of parameter settings for the design, for each device for which the design is intended.

Because METAFONT provides no simple way of adding new character shape specifications to an existing font, the whole set of character programs has to be reprocessed every time the effects of a change to one of them need to be assessed. This makes the process of font testing a great deal more time-consuming than it might be.

4.5 METAFONT as a tool for original design

The system described in *The METAFONTbook*, and implemented by METAFONT and the present version of `plain.mf`, embodies a particular model of the type design process. In this model, a set of character shapes are described by METAFONT programs. Character dimensions are expressed in these programs in device-independent units of measurement. Font intermediates (the `.gf` files) are produced by processing the character programs with the METAFONT interpreter, which is given information about the marking device for which the fonts are intended.

The technical quality of the character images the fonts give rise to is assured by correct programming, using the facilities the METAFONT language provides for positioning important parts of characters correctly on the pixel grid.

This model seems to me to reflect a situation like the one that obtained with Computer Modern or Euler, in which fully-developed character shape definitions already existed when the work of METAFONT programming was started. As a model for a process of original design, in which the designer develops a series of technically satisfactory fonts by writing METAFONT programs, it has two serious drawbacks.

The first of these is that the separate problems of defining the typeface character shapes and developing the METAFONT programs that describe them are confounded. It cannot be too strongly emphasised that at the beginning of a design the designer *does not know* what shapes the characters are going to be. The designer necessarily has a clear idea of the appearance of the intended typeface: what is not known is the details of the character shapes that will cause that appearance to be realized. Finding this out, by making sketches, is the first stage in making the design. It is difficult to write a program to describe a shape, if one does not know exactly what that shape is.

The approach to this problem that *The METAFONTbook* seems to recommend is to write a program that produces a shape whose overall features are more or less correct, and then modify the program until the shape it produces is the right one. The difficulty with this approach is that it removes from the shape-development process what to the designer is its most important feature: direct interaction with the shape itself. As Charles Bigelow has said[7] '... the designer thinks with images, not about images'.

The second serious drawback of *The METAFONTbook*'s model of the design process is that it assumes that the technical quality of the fonts the character programs produce can be assured by correct programming. I have argued in section 3.2 above that the theoretical knowledge required to ensure the correctness of the character programs in this respect is not yet available. In the absence of such knowledge, the programs have to be developed by iterative testing and modification. The testing is done by using the programs to make fonts, and looking at the character images the fonts give rise to: as I have suggested, the way the present METAFONT system is configured makes this process a great deal more difficult than it need be. It consequently suffers, even more than the process by which the character shapes are developed, from a lack of the responsiveness that is important to its success.

[7] In an internal discussion paper written for the Digital Typography Group at Stanford in late 1983.

Designer-programmer collaboration in METAFONT design

Knuth suggests that designers and programmers should collaborate in making typefaces with METAFONT (Knuth, 1986, p. v). While perhaps being practicable when programs are being written to describe existing character shapes (though the history of the Euler project suggests some of the possible pitfalls) this approach leaves out of account the fact that in original design work the designer does not know at the time the work is started what shapes the characters should be. The first stage in making a new design is sketching: the responsiveness of the sketching tool to the designer's thoughts, already seriously prejudiced in METAFONT design by the need for shapes to be described as programs, would be further vitiated if the characteristics of the shapes in the sketches had to be explained to another person.

5 Conclusion: font design and the computer science community

In traditional type manufacture, right up to the present day, every advance in marking technology has been followed by the production of new fonts that optimize the rendering of existing typeface designs with the new techniques. This continual revision of their production processes involves the type manufacturers in huge amounts of work. They are obliged to undertake it, because the maintenance of technical quality is the only way to ensure a continued market for their products: bad type won't sell.

The attitude of the computer science community to developments in marking technology is epitomised by Knuth's statement that 'We now have the ability to give a completely precise definition of letter shapes that will produce essentially equivalent results on all raster-based machines' (Knuth, 1986, p. v) and to font design by Eliyezer Kohen's comments in the introduction to his description of the Picor type design system: 'The systems for this purpose are ... too tedious (bitmap editing takes a lot of time)' (Kohen, 1985).

Because of the idealist attitude towards real marking processes expressed in Knuth's statement, consideration of the problems of font design is out of the main current of computer-science thinking: the consequence is that no good computer-based font design tools have yet been built[8]. This in turn has the consequence that the community's attitude towards font design constitutes a self-fulfilling prophecy: it *is* too tedious, because no tools have been built to make it easier, because it is too tedious.

[8] Picor is intended to be a *type* design system, not a *font* design system. The outline description stage that Picor implements is followed by a bitmap editing stage.

In this respect, the approach that the computer science community adopts to the problem of producing fonts (and hence documents) of good technical quality is exactly opposite to the approach adopted by experienced type designers. The designers' approach is exemplified by Hermann Zapf's remarks at the 1983 working seminar of the Association Typographique Internationale at Stanford: 'Today offset printing and electrostatic processes offer some new possibilities in the transfer of letterforms to paper and may automatically require new design solutions. Digitized alphabets therefore should be designed for the bitmap' (Zapf, 1985).

One has to say that the consequence of this difference in views is evident in the appearance of documents produced by computer-based systems. To acknowledge that such documents are much better than they used to be is not to say that they are as good as they should be. The objective of workers in the field should be to produce results with the new technology that are equivalent in quality to those the old technology produced as a matter of course.

References

Carter, H. G.
Fournier on punchcutting: the text of the Manuel Typographique
London: Soncino Press, 1930; New York: Burt Franklin, 1970

—

The optical scale in typefounding
Typography, no. 4, pp. 2–6 (1937)

Matthew Carter: Bell Centennial (Type and technology monograph no. 1)
New York: Cooper Union, 1982

de Vinne, T. L.
Plain printing types
New York: Century, 1900

DIN 16 518
Klassifikation der Schriften
Berlin: Deutschen Normenausschuß, 1964

Dreyfus, J.
Italic quartet
Cambridge: Cambridge University Press (privately printed), 1966

Dwiggins, W.A.
WAD to RR: a letter about designing type
Cambridge, Mass.: Harvard College Library, 1940

Knuth, D. E.
TEX and METAFONT: new directions in typesetting
Bedford, Mass.: Digital, 1979

—
The Computer Modern family of typefaces
Stanford Computer Science Department report STAN-CS-80-780 (1980)

—
A Course on METAFONT Programming
TUGboat, vol. 5 no. 2, pp. 105–118 (1984)

—
Lessons learned from METAFONT
Visible Language, vol. 19 no. 1, pp. 35–53 (1985)

—
The METAFONTbook
Reading, Mass.: Addison-Wesley, 1986

Kohen, E.
An interactive method for middle resolution font design on personal workstations
in Bucci, G., and Valle, G. (eds.), *Computing 85: a broad perspective of current developments.* Amsterdam: North-Holland, 1985

Siegel, D.
The Euler project at Stanford
Stanford, Ca.: Stanford University Department of Computer Science, 1985

Southall, R.
Metafont and the problems of type design
in André, J., and Sallio, P. (eds.), *Typographie et informatique: support du cours INRIA, Rennes, 21–25 Janvier 1985.* Rocquencourt: INRIA, 1985

—
Designing new typefaces with Metafont
Stanford Computer Science Department report STAN-CS-85-1074 (1985)

van Krimpen, J.
A letter to Philip Hofer on certain problems connected with the mechanical cutting of punches
Cambridge, Mass.: Harvard College Library, 1972

Zapf, H.
Future tendencies in type design
Visible Language, vol. 19 no. 1, pp. 23–33 (1985)