September 1985

Report No. STAN-CS-85-1074

Designing New Typefaces with Metafont

by

Richard Southall

Department of Computer Science

Stanford University
Stanford, CA 94305

Abstract

The report discusses issues associated with the symbolic design of new typefaces using programming languages such as Metafont.

A consistent terminology for the subject area is presented. A schema for type production systems is described that lays stress on the importance of communication between the designer of a new typeface and the producer of the fonts that embody it. The methods used for the design of printers' type from the sixteenth century to the present day are surveyed in the context of this schema.

The differences in the designer's task in symbolic and graphic design modes are discussed. A **new** typeface design made with Metafont is presented, and the usefulness of Metafont as a tool for making new designs considered.

Designing new typefaces with Metafont

Richard Southall

Acknowledgments

I owe a special debt to Donald Knuth for getting me involved with Metafont in the first place, and for guiding me through my battles with it. The help, friendship and support I have received from the other members of the Digital Typography Group (Chuck Bigelow, Neenie Billawala, David Fuchs, John Hobby, Cleo Huggins, Dan Mills, Lynn Ruggles, New Wave Dave Siegel and Carol Twombly) is beyond individual acknowledgment. Conversations with Matthew Carter were most valuable in clarifying the problems of both traditional and digital type design. I am grateful as well to John Warnock, Liz Bond, Sumner Stone and members of the font group at Adobe Systems Inc.

During the preparation of this report, as at other times, Johanna Goldschmid and Susie Taylor of the San Francisco Public Library's Special Collections were extremely kind and helpful in giving me access to material in the Robert Grabhorn Collection there. David Levy's criticisms of the early versions of my terminology have yet to bear their full fruit. Rachel Hewson got me through the last bit.

This research was supported in part by the National Science Foundation under grant IST-820-1926, and by the Systems Development Foundation.

Preface

This report is an attempt to summarise the conclusions reached during two years' study of the problems connected with the symbolic design of typefaces.

Most of this work was done with the old and new versions of Metafont, while I was a member of the Digital Typography Group at Stanford from September 1983 to September 1984 and from April to September 1985. In January 1985 I helped to teach an experimental course on Metafont as part of the course on typography and computer science that was organised by the Institut National pour la Recherche en Informatique et Automatique (INRIA) at Rennes (Andre and Southall, 1985). The paper I wrote as part of the supporting material for the course (Southall, 1985) provided me with a valuable opportunity to clarify my thinking about Metafont at that time. Much of the argument in Section 4 is derived from material that originally appeared there.

Between April and September 1985, in addition to my involvement with the Digital Typography Group, I worked at Adobe Systems Inc. on a project concerned with the symbolic description of mathematical characters. This work provided a valuable perspective on the problems of symbolic-mode design that are also discussed in Section 4.

1 Terminology

One of the main aims of this report is to provide a conceptual framework within which the problems of type design and production for present-day document preparation systems can be discussed.

The first step in doing this is to define a useful and consistent terminology. This is especially important in a report such as this one, because the computer science literature on type design has been bedevilled by the misuse of terms carelessly borrowed from the traditional technologies of typography and type manufacture. The word font, in particular, is used indifferently in the literature to refer to three or four fundamentally distinct entities – an unusual departure from the normal precision of computer science terminology.

The terminology proposed here tries to be as generally applicable as possible in the description of documents and their production, rather than being restricted to describing passages of English-language text rendered on computer-driven laser printers and visual displays. As with all specialist terminologies, its purpose is to draw distinctions between concepts that ordinary language does not usually distinguish but that are in fact distinct.

I use slanted type to identify important concepts as they occur.

1.1 Texts and documents

A text is a sequence of semantic elements intended to convey a meaning. [1]

Texts are abstract entities, that are realized as documents. Documents are ***hand*-written** or machine-written.

Documents can be ***virtual*** or ***actual.*** A ***virtual*** document is one that exists in a concrete but non-graphic form: for example, inside a computer memory or as a recording on a magnetic disc. Virtual documents can take several forms (Quint, 1983). Only machine-written documents can exist in virtual form: a handwritten document must be realized as an actual document if it is to exist at all.

An actual document, whether handwritten or machine-written, is an ***assembly of*** marks on a substrate. This assembly of marks realizes the text of the document in a particular graphic ***form.*** The marks in an actual document are grouped into graphic elements that are visible realizations of the semantic elements of the text. The graphic elements of a document can have verbal and ***non-verbal components.***

[1] ***Sequence*** here means ***element optionally followed by one or more elements;*** so that a text may contain only a single element.

Actual documents can be of two kinds: ***originals*** or copies. All handwritten documents are originals: each exists only in a single instance.[2] Actual machine-written documents can be originals or copies.

An original actual machine-written document is one that is produced directly by the action of a mechanical writing system. Machine-written actual documents are different from handwritten documents, in that there can be more than one instance of an original document. We can call these multiple instances of a single document replicated originals or replicates.

A familiar example of a mechanical writing system that produces replicated original documents is a computer-driven airport information system using television displays. Each display realizes an actual document. These documents are not copies of an actual master document that exists somewhere behind the scenes at the airport. They are different objects, because they are in different places and may be of different sizes; but they are all original realizations of a single virtual document in the computer that drives the system. When a change is made to the virtual document, all the actual documents on the displays change simultaneously.

Copies, on the other hand, are machine-written documents that are made by a mechanical reproducing ***system*** from a single original document. This original has to be in existence as an actual document before copies can be made from it. Thus copying is a diachronic process, while replication is a ***synchronic*** one.

Copies, as well as originals, can be replicated. In some television-based airport information systems, there really is a master original document somewhere behind the scenes, with a television camera pointing at it. When the master original is changed, the documents on all the displays change simultaneously: but they are still copies, because a new original, or a new piece to be added to the old one, had to be made before it could be put in front of the camera.

1.2 Typefaces

Actual machine-written documents are assemblies of marks on a substrate, pro duced by a mechanical writing system. The marks that make up the verbal components of the document are character images. Before we go on to discuss how these images are produced, we need to define a few terms that we can use to refer to them.[3]

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 serve to distinguish it from other typefaces.

The visual attributes of a typeface are of two kinds: ***stylistic*** and functional. The stylistic visual attributes of typefaces are the subject matter of typeface classification schemes (DIN 16 518, 1964; Atkins, 1975) as well as of aesthetic criticism of

[2] Someone writing out the same text twice by hand produces two different documents that differ, even if ever so slightly, in their graphic form. A machine-made copy (for example a photocopy) of a handwritten document is a machine-written document that is a copy.

[3] ***Script, typeface*** and ***family of typefaces*** are terms taken from traditional typography. The definitions these terms are given here are more precise than their definitions in traditional usage tended to be.

the traditional kind. The functional visual attributes of typefaces are those that perform the function of typographic differentiation in the rendering of complex text (Walker, 1983; Waller, 1980).

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

1.3 Character images

The character images (as well as the other marks) in an actual document produced by a mechanical writing system are made by a marking device that is part of the system. The character images are actual marks, and as such they have graphic attributes: among these are shape, size and colour.[4] The shape of the mark that constitutes the character image provides a graphic rendering of the visual attributes of the typeface character shape. The other graphic attributes of the character image are added to the visual attributes of the typeface character shape by the marking device to give the character image its appearance.

We now need to consider how the character images in a document get their shapes, and the extent to which the marking device can vary its contribution to the appearance of the shapes it produces.

1.4 Image carriers

Every marking device contains one or more image carriers. Image carriers, like documents, can be actual or virtual. The function of an image carrier is to ***specify the*** shapes of character images to the marking device. The marking device makes output character images by making marks on the document substrate with some kind of marking process, following the specifications on the image carrier.

An actual image carrier contains one or more visible images of character shapes; so that in such an image carrier the specification for the shape of the character image to be produced is itself a shape, and we can think of marking devices that use actual image carriers as producing output character images that are ***prefabricated.***

In marking devices of this kind, the function of the marking process is to transfer the character image from the image carrier to the document substrate. This transfer can be done ***directly*** or ***indirectly.*** If it is done directly – if, for example, the marking device uses the character image on the carrier to effect the transfer of pigment to the document substrate – the marking device cannot perform any transformations on the character image in its passage from document to substrate; so that the size, aspect ratio[5] and slant of the output character image will correspond to the size, aspect ratio and slant of the character image on the carrier. If the transfer is indirect, on the other hand – if, for example, the character image is transferred to the substrate by means of an optical or electro-optical system and the substrate is marked photographically – the part of the marking device that effects the transfer may have the ability to perform transformations on the character image during its passage from the image carrier to the document substrate; so that the output image may differ from the image on the carrier in size, aspect ratio, slant or in all three.

[4] In discussions of type design and type composing techniques we tend to forget about the eventual colour of character images. This is all right if we are dealing with type designs for photocomposing machines and laser printers; it is not necessarily all right if we are dealing with designs that are intended to be seen on visual displays.

[5] The aspect ratio of a shape is the ratio of its height to its width.

Thus with direct-transfer marking processes the number of output character images with different appearances that the marking device can produce at any one time is completely determined by the number of character images on the image carriers that are in the device at that time. With indirect-transfer marking processes, the number is determined both by the number of images on the carriers in the device at that moment and by the transforming ability of the image-transfer process.

Marking devices that use virtual image carriers do not transfer images from carrier to substrate in the same way. The character shape specification contained in a virtual image carrier is not a shape; it is indeed a specification for a shape, that has to be interpreted by the marking device in order for the device to make an output character image on the document substrate. The function of the marking process in devices of this kind is in a sense much more basic than in devices using actual image carriers: it is to make a dot or draw a line on the document substrate. The output character images that such a device produces are ***synthetic,*** built up from elementary units. The positions (and lengths, if they are lines) of the elementary units that make up the output image are determined by the marking device as a result of its interpretation of the character shape specification in the image carrier.

In the present state of the art, the character shape specifications in virtual image carriers come in two main varieties (Bigelow and Day, 1983; Hegron, 1985). For the sake of brevity, we can call them ***bitmap*** and ***outline*** **specifications.**[6] In a bitmap specification, the positions and lengths of the elementary units that make up the output image shape are specified in detail: so that the specification is simply a set of instructions to draw the shape. An outline specification contains the locations of points on the edge of the character shape, and (usually) information about whether they are joined by straight lines or curves: the marking device is left to figure out the positions and lengths of the elementary units it needs to draw in order to make the output image shape.

It is clear that a bitmap specification allows the marking device very much less latitude in interpretation than an outline specification does. If the positions and lengths of the elementary units that make up a character shape are specified in detail, then about all the marking device can do to change the appearance of the output image is to change the coordinate system in which the positions and lengths are specified. Certain digital photocomposing machines do this, producing different output .**images** that are isomorphically or anamorphically scaled, and perhaps slanted as well, from a single bitmap specification. This scaling involves altering both the separation between the lines written on the photosensitive material and the size of the writing spot itself, **and** because of the need to maintain consistent photographic exposure of the output character images the amount of scaling is limited to a factor of about **2:1** along each axis. On present-day commercially available laser printers, neither the separation of the written lines nor the size of the writing spot can be altered, and each combination of scaling and slanting in the output image requires a separate specification in the image carrier.

With outline specifications, the marking device has a great deal of freedom to interpret the specification. In principle, any geometrical transformation whatever can be applied to the shape whose outline is specified, and the limit to what is done in practice is set more by what is desirable than by what is possible. The particular advantage that outline specifications have for commercial applications is that output images of any size within the range that the composing machine

[6] Bitmap specifications, for the purposes of the present discussion, include **run-length-**encoded and other 'packed' versions of bitmaps.

can produce can be obtained from a single image carrier specification;[7] so that compared even with optimally packed bitmap specifications, outline specifications for a wide range of output image sizes are very compact. (This compactness is paid for, of course, in reduced output speed: the marking device has to do much more computing to produce each character image.)

Thus, with virtual image carriers, the use of bitmap specifications in the marking device limits the capability of the mechanical writing system in very much the same way that the use of direct-transfer marking processes does in marking devices that use actual image carriers.

It is important to notice, though, that in a real mechanical writing system, whether it uses actual or virtual image carriers, the actual shapes (and consequently the appearances) of the character images the marking device produces are not completely determined either by the specification contained in the image carrier or by the marking device's interpretation of it. The marking process itself affects to a greater or lesser extent the shapes of the marks that are produced. If the shape of the output character image is to render the visual attributes of the typeface character shape as accurately as possible, then the specification in the image carrier has to take into account the effects that the marking process has on the shape of the output image.

This may be easier to do with an image carrier for a direct-transfer marking process, or with a bitmap specification, than it is with a carrier for an indirect-transfer marking process or an outline specification. For its existence to be worthwhile, the interpreting process in the marking device has to be able to give each of the graphic attributes it adds to the output images a wide range; and an interpretation technique that works well at one end of the range of a particular attribute may work very much less well at the other. This is especially true in going from large to small output image sizes and from small to large image aspect ratios. In the present state of the art it is difficult to find interpreting techniques that are fast enough for interpretation to be done 'on the fly' while composition is in progress, and also clever enough to take into account the effects of a less-than-ideal marking process on the small details of character shape over the whole range of output image sizes and aspect ratios.

It is also worth noticing that the effects of the marking process may not be completely predictable. In most processes that involve the transfer of pigment to a substrate, for example, the amount of pigment available for transfer and the surface characteristics of the substrate, both of which affect the shapes of the marks that are produced, cannot be exactly predicted at the time the character image specifications in the image carrier are designed.

1.5 Glyphs and fonts

The character shape specifications in the image carrier, then, whether they are actual or virtual objects, are the means by which the shapes of output character images are specified. If, for the moment, we introduce the designer as someone who is concerned with the correspondence between the shapes of output character images and the visual attributes of typeface character shapes, then making the specifications in the image carrier is clearly a primary part of the designer's concern. But image carriers are not unique objects made by the designer; they are manufactured objects, made by a production process according to instructions that

[7] Whether or not this is a good thing, in terms of the typographic quality of the output, is a different question (*cf.* Bigelow, 1981).

it is given. Thus the designer, in order to determine the shapes of the output character images produced by a particular marking device, has to make ***instructions to the production*** process that makes the character shape specifications in the image carriers the marking device uses.

We can call a set of instructions to the production process to make a particular character image shape specification on an image carrier intended for a particular type of marking device a ***glyph.*** The set of character shape specifications, on one or more image carriers, that corresponds to the character shapes of ***a*** particular typeface is called a ***font.”***

Notice that a font is not usually the same as a set of glyphs, even though the character shape specifications in the font derive from the instructions in the glyphs. The eventual object of making ***a*** font is to allow a marking device to produce sets of output character images that realize the visual attributes of a typeface. The shapes of these images will be affected by the marking process that produces them: thus the character shape specifications in the font will need to take the characteristics of the marking process into account. The exact content of the character shape specifications in the font may be affected by the process by which the font is produced: thus the instructions in the glyph may need to take the characteristics of the font production process into account. It is only when the font production process has no effect at all on the content of the specifications in the font that the content of a set of glyphs can be the same as the content of the font that derives from them.

Because the terminology presented here tries to be as general as possible, it uses general rather than process-specific terms in its descriptions. A character shape specification can itself be a shape, and a set of instructions to a font production system can be a shape as well: but neither need be a shape, and in some font production processes neither is.

[8] This terminology has the great advantage that it yields definitions of typeface and ***font*** that correspond very closely to the meanings these words had in the traditional terminology. It also distinguishes clearly between them: something that computer science usage often fails to do. In the sense in which Knuth uses the qualifier ***meta-*** in his 1982 paper, a typeface is a meta-font, in that it is something that gives rise to a number of different fonts. What Knuth calls a **‘Meta-Font’** is, according to the definitions given here, a meta-typeface.

2 Type manufacturing systems

We can now look more closely at the connections between typefaces, **glyphs**, fonts and output character images.

2.1 Requirements for **a** font production system

The shape of an output character image provides a rendering of the visual attributes of a character shape in a particular typeface. If this rendering is to be accurate, the character shape specification in the font must take into account the effects on the character image shape of the marking process used by the marking device for which the font is intended.

The character shape specification in the font is produced by **a** manufacturing process that follows the instructions in a glyph. If the content of the specification in the font is to be correct, the instructions in the glyph must take into account the effects on the content of the specification of the manufacturing process by which the font is produced.

Thus making a glyph that will give rise to an output character image with a shape that accurately renders the visual attributes of a typeface character shape involves taking into account both the characteristics of **a** particular marking process and those of **a** particular font manufacturing process.

As well as these system-dependent factors, there are two other factors that need to be considered in defining the content of glyphs. The first of these has to do with the way character images are perceived by the reader. The perceived characteristics of shapes that are seen small are affected by the way the human visual system works: so that, for example, a circle will normally look smaller than a square whose sides are the same length as the diameter of the circle. The visual phenomena involved are illustrated by Legros and Grant, and their effects on the perception of small character shapes are discussed by Harry Carter (in footnotes in his edition of Fournier's ***Manuel typogruphique*** as well as in his 1937 paper) and by T. L. De Vinne (Carter, 1930, 1937; De Vinne, 1900; Legros and Grant, 1916, ch.5).

The existence of these perceptual effects means that the visual attributes that a character shape has when it is seen small are different from those that the same shape has when it is seen large. Thus if an output character image is to be perceived as providing an accurate rendering of the visual attributes of a typeface character shape, its own shape (and consequently the specifications in the image carrier and the instructions in the glyph) must take the relevant perceptual effects into account.

It can be very hard to predict the way in which the appearance of a shape will change when it is seen small. It can also be hard to predict exactly the effects of particular marking processes on certain features of character shapes. Because of this, font manufacturing systems have typically included steps in which the content of the glyphs is developed by a process of iterative testing and modification. This testing has included the making and testing of trial fonts, so that the effect of the whole manufacturing process on the appearance of the output character images is explored. The content of the glyphs has only been finalized, and font manufacture begun, when the output character images have been seen to be satisfactory both as renderings of the typeface character shapes and in terms of their technical quality.

The need for the output character images to be of good technical quality is the other factor that needs to be taken into account in defining the content of glyphs. The character images, as well as realizing the visual attributes of the typeface, have to perform as well as possible in the technical conditions for which they are intended. This means that they should give rise to documents that are as legible as possible in the conditions in which they are mainly intended to be read.

A legible document is one that is easy to read; thus in order to understand how criteria for legibility are arrived at, we need to understand the mechanism of normal reading. Fluent readers do not scan a line of character images from left to right, identifying individual characters one by one. They read groups of characters, words, or groups of words, not necessarily in sequence from left to right, and process the information they receive with mechanisms that rely heavily on the linguistic as well as the visual contexts that surround the part of the text that is being read (Pirozzolo and Wittrock, 1981; Tzeng and Singer, 1981; **Visible *Language,*** 1984).

This means that the most important requirement for a legible document is that the words in it should be easy to identify. The minimum criteria for the identifiability of a word are that the boundaries of the word should be clear, and that the character images in it should be identifiable (so that one can tell which character a particular image is supposed to represent) and discriminable (so that one can distinguish, for example, a c from an o, or a 2 from a *Z*). These minimum conditions for the word-by-word decipherability of a document often seem to be taken by display engineers as sufficient conditions for its legibility (*cf.* Shurtleff, 1980).

When we read **a** document, most of the characters in the words we see are not resolved sharply, because their images do not fall on the parts of the retina that can detect fine detail. This means that in order to be identifiable in normal reading, words must have a clearly articulated visual structure that does not depend for its identity on the small details of character shape. This in turn means that the character images in the document must have harmonious visual relationships with one another: they must not disturb the visual structures of words by diverting the eye to incongruous features of their own shapes. It is also important that the character images should not be too close together or too far apart. If they are too close together they may combine, in parts of the reader's visual field where they are not well resolved, to form ambiguous or unrecognizable shapes; if they are too far apart the space between them may be mistaken for the space between words.

Equally, the character images should not combine in such a way as to disturb the reading process itself. In normal reading, the reader's eyes move in very rapid jumps (saccades) between the fixations during which the words in the document are perceived (Spencer, 1969). This alternation of saccades and fixations should not be disturbed by accidental graphic features in the document. For example, there should not be dark patches within words, caused by character images that are too heavy or too close together, that might cause inappropriate fixations to occur.

Thus legible character images have identifiable, discriminable and visually harmonious shapes, with no inconsistencies of weight or spacing. A legible typeface is one whose character shapes yield legible character images in documents produced by all the marking devices for which the typeface is intended.

2.2 Type designer and font producer

A typeface is a set of character shapes with particular stylistic and functional visual attributes. In real design work the functional visual attributes of a typeface, and many of its stylistic ones, are derived from the specifications for the design. In making the design, the designer has to know how to incorporate the visual attributes the typeface is required to have into a set of shapes that are also identifiable as characters. (Matthew Carter's descriptions of his work are particularly informative about the problems of doing this: Carter, 1985; Cooper Union, 1982.) Type design in this sense is clearly a different kind of business from font production, which involves understanding the effects of font manufacturing processes and marking processes on the shapes of character images, and the effects of the human visual system on the way they are perceived.

Also, because a typeface is a set of shapes, a single typeface can be realized in output character images produced by many different kinds of marking device that use different marking processes and different fonts. Fonts, and the glyphs that give rise to them, are by contrast specific to particular marking processes and very often to particular types of marking device. Thus, if we think of a ***type* manufacturing *system*** as including everything involved in making a new typeface design and producing the fonts that cause it to be realized by a particular type of marking device, such a system will have two distinct parts that correspond to the type designer's and the font producer's roles in the process. The type designer's role is to define shapes for the characters of the typeface that have a particular combination of visual attributes. The font producer's role is to make glyphs that will give rise (via the font production process and the action of the marking device) to output character images that are perceived as having the same visual attributes as the typeface character shapes.

The issue of communication between type designer and font producer is thus of central importance in a type manufacturing system. If the distinctive visual attributes of the typeface are to be properly realized in the output character images that result from the producer's work, the producer must be able to learn from the designer what those attributes are. The producer should also be able to discuss with the designer, during the development of the glyphs for a font, the extent to which the character images the glyphs give rise to in their current state realize the designer's intentions.

Because the distinctive attributes of a typeface are visual, the commonest and most effective mode of communication between designer and producer has been a graphic one. In this mode, the designer makes drawings of the typeface character shapes, and these provide graphic realizations of the visual attributes of the typeface.

It is important, in considering a particular type manufacturing system, to understand the precise role that the designer's drawings play in it. Most often they serve as ***models,*** that illustrate the visual attributes of the typeface to the font producer. In some systems, particularly those in which the characteristics of the font production process and the effects of the marking process are easy to understand, the designer's drawings may be intended as patterns that the font producer can use directly to specify the contents of glyphs. However, because glyphs are specific to particular font production systems and marking processes, drawings that perform well as patterns in one type manufacturing system are unlikely to do so in another.[1]

[1] This point was well illustrated at one stage in the development of direct-photography photocomposition (section 3.3 below). Some type manufacturers took drawings that had been developed as patterns for the production of matrices for hot-

The role of the glyph in font production

In this schema of font production, the role of the glyph is significant. The stage at which the content of the glyph is defined is the last stage at which designer or producer can intervene in controlling the shape of the output character image. The effects of the font production process and the marking process can (and should) be anticipated in the content of the glyph. It should be possible for the effects of the font production process to be controlled by the font producer: but the effects of the marking process cannot be controlled, except perhaps by the eventual user of the font.

2.3 Design systems and drafting systems

Just as the word ***font*** is used in the ordinary computer science literature to refer to any of the concepts identified here as typefaces, glyphs, fonts or sets of output character images, ***so*** the word ***design*** is used to describe both the type designer's and the font producer's role in ***a*** type manufacturing system. It is important, particularly in discussing computer-based type manufacturing systems, to use design in a single sense.

We can make a distinction between ***design systems*** and drafting ***systems.*** The function of a drafting system is essentially to ***translate*** character shape descriptions from one form to another. Such a system always has to have graphic representations of existing character shapes as input, and the first step in using such a system is always to describe the character shapes to it in some way. Drafting systems are typically used at the front end of commercial digital font production systems, where their function is to translate character shape descriptions from graphic to numerical form.

The type designer uses a design system to define the shapes of the characters in a new typeface. The function of a design system is to allow a designer to make new character shapes, that have never existed before and are not derived directly from any shapes that have existed before.[2]

metal type-composing machines, and used them directly as patterns for the production of photographic matrices. The different effects of the font production processes, together with the great difference in the characteristics of the marking processes, meant that the photocomposed character images that resulted were of very poor technical quality. Other manufacturers used as patterns character shapes that they copied from the types produced by hot-metal composing machines, with similar results.

[2] To understand what is meant by 'new' in this context, we need to understand the distinction between the configurations of characters and their shapes. A small p, for example, has a familiar configuration: a descending vertical stroke, with a bowl at the top right. If we make a character shape with a new configuration – by putting the bowl to the left of the stroke, or at the bottom, or giving it a crossbar – we make something that is harder to identify as a small p than the familiar configuration would be, and is therefore less effective as a realization of the character. Type designers do not usually have the task of devising new configurations for characters.

Developing new shapes for characters, on the other hand, is what type designers do for a living.

3 A survey of type design methods

With these ideas in mind we can now make a brief survey of type design methods that have been used in the past, with the aim of studying the interactions between designer and producer in each one.

The fact that a particular font production system or marking device is obsolete in technological terms need not prevent us from considering **a** design method that was used with it. Almost all the typefaces we use (and copy) today were originally designed for composing techniques that are no longer in commercial use. Making character images of good technical quality has always been among the font producer's objectives; and in reading designers' accounts of their work it should be possible to disentangle the design method that was being used from the techniques that served to implement it.

The history of type design and production falls into three periods, before the commercial success of digital composing techniques in the latter part of the 1970s did away with the production of actual objects by the type manufacturers. The first and longest of these was the era of hand punchcutting. The techniques of this craft seem to have been established within fifty years of Gutenberg's invention (Carter, 1969); although it dwindled rapidly in importance **as a** commercial process after L. B. **Benton's** invention of the pantographic punchcutting machine in 1885 (US patent 327855 of **1885),** it lingered on as a production method into the **1930s,** and still survives in a few hands today (Drost , 1985).

The second period was that of mechanical punchcutting and the mass production of matrices for hot-metal typecasting machines. This period began with **Benton's** invention; it came to an end very rapidly at the end of the **1960s,** after electromechanical and electronic photocomposing machines had been introduced. These **ma**chines, which used photographic negatives as image carriers, brought with them the last and shortest period in the production of actual objects as type. By 1977 it was clear that the future of typesetting was with digital photocomposition and virtual image carriers; and about the same time the introduction of xerographic laser printers began to accelerate the crumbling of the boundaries between typesetting proper and the office environment that had begun with the introduction of the IBM Selectric Composer typewriter in 1966 and is now almost complete.

3.1 Hand punchcutting

The technique of type production by hand punchcutting, matrix-making and casting is summarised by Harry Carter in ***A view of early typography*** (Carter, 1969). The classic contemporary accounts are those of Moxon (1683) and Fournier (1764-66). Carter's edition of Fournier is indispensable (Carter, 1930).

A ***punch*** is a steel bar, an inch or two long, on one end of which a character shape is cut with files and gravers. The character shape on the punch is the glyph in this process. During its development, the punchcutter tests the shape of the glyph by making ***smoke-proofs:*** the end of the punch is blackened with soot in a smoky flame and pressed on to a piece of smooth card. When the shape of the glyph is judged to be correct, the punch is hardened by heat treatment and struck into a block of copper to make a ***matrix.*** After being adjusted for width and alignment, the matrix is used with a ***mould*** to cast type' (Nelson, 1985). The pieces ('sorts') of type that result are the image carriers.

Charles Bigelow describes punchcutting ***as*** a ***glyptal*** process, emphasizing its sculptural quality (Bigelow and Day, 1983). From our point of view, type production by hand punchcutting is characterized by two important features that were lacking in subsequent processes for the production of actual image carriers.

The first of these distinctive features is that the character shapes on the punches, which are the glyphs, are the same size (except for the minor contributions of the font production and marking processes) as the eventual character images. This means that in developing the shape of the glyph the punchcutter does not have to take into account the effect of reduction in size on the appearance of a large shape. The glyphs (and hence the character images) can be given their desired appearance directly. In modifying the punch and making smoke-proofs, the punchcutter is working on the ***appearance*** of the character rather than its shape.

The second distinctive feature of hand punchcutting is that during almost the whole history of the technique it was actually impossible for the punchcutter to work from patterns. This is because the technology needed to transfer a shape accurately from a drawing to the face of the punch did not exist until the end of the nineteenth century. (The technique, which involves making a reduced photoengraving of the drawing, is described by Drost.) Moxon indeed gives scale drawings of what he considers to be correct character shapes; but it is quite clear, from looking at the drawings as well as reading his account, that in our sense of the words they were for use as models rather than as patterns (Moxon, pp. 106-109; illus.pp. 124-130).

Fournier's attitude towards patterns is much more forthright than Moxon's: 'As to the best possible shape to give to letters, it is useless to write of it: it is a matter for the taste and discernment of the cutter, and it is in this that he displays his proficiency or his incapacity.'[2]

Designer and producer in hand punchcutting

Fournier, being his own designer, was not troubled by problems of communication in deciding on the subtle combinations of stylistic and functional visual attributes in the typefaces he produced. One can imagine that Bishop Wilkins, in having the characters for his 'philosophical language' cut by Moxon, was more concerned that Moxon should render the characters' shapes correctly than that they should be in any particular style (Moxon, p. 364). For Pierre Didot, on the other hand, the problem of communicating a very precise notion of style to the punchcutter was a primary one. We can see how he solved it from the introduction to his ***Specimen*** of 1819: 'For about ten consecutive years, during which I regularly spent some three hours a day on this work with M. Vibert, certainly one of the most skilled punchcutters we have today, my most frequently repeated corrections, my most exact instructions, even my perhaps whimsical perfectionism, that often led me to

[1] This is the origin of the word ***font:*** something that is cast. The process of casting type is called ***la fonte*** in French.

[2] ***Manuel,*** vol. I, p. 19; Carter's translation. Fournier's well-known criticisms of the Jaugeon commission's designs for the ***nouvelles lettres françaises,*** though amusing to read and perfectly justified (especially in the case of the italic small letters, which are indeed quite extraordinarily clumsy and ugly) may have been prompted by something other than disinterested aesthetic considerations (Jammes, 1961).

remake the same letters two or three times over, could not chill his enthusiasm, or allow me to discover the limits of his patience.'[3]

The difficulties that arise when the collaboration between designer and punchcutter is less intimate, and communication between them consequently breaks down – in particular, when the function of the designer's drawings is understood differently by the parties – are vividly described in John Dreyfus' account of the production of the Cranach Press italic for Count Harry Kessler by the English punchcutter E. P. Prince (Dreyfus, 1966).[4]

3.2 Mechanical punchcutting

Benton's pantographic punchcutting machine introduced the industrial era of type manufacture, and also the separation between the activities of design and production that is characteristic of industrial processes in general. The machine cut punches by copying the character shapes from large metal patterns; these were made by tracing round the outlines on drawings that were larger still (Legros and Grant, p. 213; Dwiggins, 1940).[5] Thus, with Benton's invention, the glyphs in type manufacture suddenly changed: from being three-dimensional objects embodying shapes that were the same size as the output character images, they became two-dimensional objects with shapes that were very much larger; and the activity that produced the shapes, instead of being *making,* became *drawing.*[6]

Benton's machine, because it cut punches, was mainly used in the production of type intended for composing continuous text: 18 point (around 5 mm capital letter height) and below. Very similar techniques were used in the production of display type: matrices were engraved directly using a pantographic engraving machine (Legros and Grant, p. 236), or made by electrotyping from relief masters cut in type metal (ibid., pp. 238-239). These techniques were used in the 1920s and 1930s to produce new typefaces in quantities that have never been equalled since (Southall, 1982).

There are not very many accounts by designers of their work in designing for these production methods, although the technical aspects of seeing a design through the manufacturing process are described and illustrated in publications produced by the Monotype Corporation in England under the aegis of Beatrice Warde (Monotype Corporation, 1956; Morison, 1932). Of the available accounts, Eric Gill's (Gill, 1931) is too quirky in its perspective to be useful from a technical point of view. Jan van Krimpen's ***On* designing *and* devising *type*** does not give a typical picture of designer-producer relationships: almost all his designs were realized by hand punchcutting in the first instance, and he was always more or less dissatisfied with

[3] Didot, 1819; my translation.

[4] This work is a locus classicus for any study of the nature of the relationships between designer and producer – whether in the production of type or anything else.

[5] Charles Bigelow, in an internal discussion paper written for the Digital Typography Group, counts mechanical punchcutting in with hand punchcutting as a glyptal process. This seems to me to miscategorize it. At every stage in the process – whether making the pattern drawing, cutting the pattern or cutting the punch – the operator is tracing round the outline of a shape rather than carving an object. For want of a better word, I would describe the process of mechanical punchcutting *as delineatory.*

[6] This shift from making the object that is required to making a representation of it is of enormous conceptual importance. It corresponds to the shift from concrete to abstract representation of objects or processes that is characteristic of computer technology.

the machine-cut versions (van Krimpen, 1957, 1972). Frederic Goudy's description of his matrix-engraving work, although very informative about his design objectives, is also untypical of the industrial situation, because he was a designer doing his own production (Goudy, 1940). Perhaps the most accessible accounts in English of design for production are Hermann Zapf's ***About alphabets*** for display type design (though he also deals with text type design) and W. A. Dwiggins' ***WAD to RR*** for text type design (Dwiggins, 1940; Zapf, 1960).

Two things emerge from these accounts that are important for the present discussion. The first is how much less direct the process of developing the shapes of the glyphs was in the industrial technique than in hand punchcutting. The designer would make a design and give it to the font producer. Some time later, a proof of the type would appear; the designer would mark corrections on the proof and return it, and some time later still a corrected proof would come back. This long interval between specifying a correction to a shape and seeing its result is not surprising, in view of the complexity of the production process, but it contrasts very unfavourably with the immediacy of punch and smoke-proof (or computer keyboard and visual display) .[7]

The other thing that appears from these accounts is the difficulty that even experienced designers have in anticipating the appearance of character shapes when they are seen small. Zapf: 'All await impatiently the proof of the first letters. They are seemingly transformed on the paper, now ***at lust*** appearing in their native character to the designer. They seem either pure, graceful, true of image, quite as he conceived them; or knavishly grinning, awry, as if out of insolence ready to fall on their faces, or too gaily hopdancing on the line, wilful, unregarding of their neighbours; still others are unduly fat or plump or again too spidery, too meager and wretched in expression. Others on the contrary **have** bumps on their curves, lame arches, lack tension or expression on the line.'" Dwiggins: '***Curves*** do all kinds of queer things when reduced; and the way lines running together make spots is a thing that will surprise you . . . I am beginning to get the drift of it and to foresee from the large drawings what will happen in the type. I ***can modify*** in the large outline, but so far I can't ***originate*** in that medium."

3.3 Photomatrix making for direct-photography photocomposition

With the development of 'second-generation' photocomposing machines in the 1960s (Seybold, 1984, ch. 8) the complicated and forbidding process of font production for hot-metal typecasting machines seemed to give way to a process of transparent simplicity: designers made drawings that were photographed to make fonts.

---

[7] The means that designers used in making corrections to the character shapes on proofs of metal type vividly illustrate the problem of designer-producer communication. The designers could not interact directly with the character images: the type on the proofs was too small for accurate drawn corrections to be made to it. Nor was it easy for them to modify the glyphs: even when the pattern drawings were accessible to them, their graphic nature was quite different from that of their own drawings or the type (Dwiggins, 1940, p. [11]). The consequence was that the designers' corrections were made in a combination of verbal and graphic modes, in a way that clearly relied on a great deal of common understanding between designer and producer (Zapf, 1960, illus. p. 67). My own favourite is **a** remark in a list of corrections to a proof of Rudolf Koch's ***Magere deutsche Schrift*** (in the Klingspor-Museum at Offenbach): ***D: Form nicht gut!***

[8] Zapf, 1960, p.22; my italics.

[9] Dwiggins, 1940, pp. [3]–[4]; emphasis in the original.

Even if font production for direct-photography photocomposition was in reality a good deal more complicated than this, it was still conceptually very simple; and the increased capability of photocomposing machines compared with hot-metal type-casting machines meant that many fewer technical constraints had to be put on the designers' work. This in turn meant that going from design to glyph became much more straightforward: designers could indeed sometimes make glyphs directly (Frutiger, 1980, p. 45). Developing the shapes of glyphs to make technically satisfactory fonts became much simpler too: not only because going from glyph to font was relatively quick, but because the connection between a change in the glyph and the corresponding change in the output character image was easy to understand.[10]

For these reasons, designers' accounts of their work for photocomposition differ strikingly from the accounts of designers working for hot-metal type manufacture, in that the font production process is seen not as an obstacle to the realization of the designer's intentions but as something with which a straightforward accommodation has to be reached (Frutiger, 1970, 1980; Gürtler *et al.*, n.d.).

The effects of these changes in the production process on the kind of designs that came to be made can be illustrated by Adrian Frutiger's Univers. Although its first appearance was as foundry type for hand composition, the Univers family was conceived for the Photon-Lumitype photocomposing machine (Frutiger, 1980, p. 16; Seybold, 1984, p. 78). The way in which the subtle relationships between the visual attributes of the different faces in the family depended for their realization on the advanced capability – for its time, at least – of the Photon machine can be appreciated by anyone who compares the photocomposed settings of Univers in the Lumitype specimen book with specimens of the Monotype hot-metal versions. The lack of uniformity of slope and fit in the Monotype versions of the narrow italics is particularly noticeable: and not surprising, because the original designs of these typefaces call for character width allocation and kerning facilities that the Monotype hot-metal system was incapable of providing.

The relative ease of design and production for direct-photography photocomposition, and the commercial success of Univers, meant that the idea of the typeface family became a natural one for designers. Since the establishment of the International Typeface Corporation in 1971, it is a rare typeface indeed that comes to the market only in the traditional combination of roman, italic and bold versions.

3.4 Digital photocomposition

The early development of digital photocomposition is described by Seybold (ch. 9). Primarily because of the cost of the computer memory needed for storing bitmap fonts, digital photocomposition remained an expensive special-purpose composing technique until the mid-1970s, a decade after its introduction.

---

[10] If it was all as simple as this, why did photocomposition get such a bad name for quality in the late '60s and early '70s? Briefly, because simple processes are not necessarily easy ones. Photographic quality had to be very carefully controlled all the way through the process, from designer's drawing to composed type: this was often hard to do, particularly at the stage of composition, when the type manufacturer's quality standards were at the user's mercy.

Also, because the whole business of photocomposing machine manufacture needed much less capital than hot-metal machine manufacture did, some manufacturers who were ignorant of typographic standards came into the business to turn a quick profit. Not a few of them lost their money and vanished or were swallowed up, leaving a legacy of second-rate machines with poor-quality fonts in the hands of incompetent users.

In the mid-'70s both memory and processing power began to become much cheaper. The Linotron 202, introduced in 1977 and using outline fonts interpreted 'on the fly' by special hardware, was followed in 1979 by the Linotype CRTronic: a desk-top direct-entry machine, which used one of its two microprocessors to interpret fonts of the same type (the other provided the machine's word-processing and text-formatting capability). These two machines, along with the Compugraphic 8600 (using outline fonts) and the Autologic APS-5 (using bitmaps), signalled the end of direct-photography photocomposition in the same way that the Photon 713, the Compugraphic 4961 and the Linotype VIP had signalled the end of metal-type composition a decade before.

In many ways, the type manufacturers must have been glad to see the decline of direct-photography photocomposition. They faced an enormous task in producing numerical descriptions of the character shapes in their typeface libraries; but that could be done once for all, and the subsequent processes of font production immeasurably simplified. Whatever the difficulties of arriving at a satisfactory virtual font, the fact that it could be replicated quickly, at low cost and without error was a tremendous step forward in a commercial sense.

The process of digital font production from an existing typeface library differs in some important ways from earlier font production processes. The input to the font production system is the same realization of a typeface character shape that had been used as a glyph for photomatrix production. The first step in making a digital font is to take this physical object, wich carries a solid shape bounded by straight lines and curves, and make a numerical description of the shape."

This is done either by generating a very-high-resolution bitmap description of the shape with a rotary drum scanner, or by using the shape as input to a drafting system that produces a numerical description of it. The Ikarus system, the PM Digital Spiral and the Camex Letter Input Processor are all examples of drafting systems that are used in the first stage of digital font production (Elsner, 1980; Flowers, 1984; Ruggles, 1983).

It is clear from watching these systems at work that their product is not a simple description of the character shape, but an interpretation of it in terms of the descriptive idiom of the system in question. The systems depend for their successful operation on the interpretive skill of their operators.

A large part of the reason why this interpretation has to take place is that the curves in the input character shape are not curves of any particular mathematical kind: they are curves drawn by the designer, or traced by the staff of the type drawing office that produced the shape. A drafting system, on the other hand, does have a particular mathematical way of describing curves, and it can only describe them in that way. For example, the Camex Letter Input Processor describes curves as successive arcs of circles with differing radii (Flowers, 1984, p.46). In using the Camex system to make a description of a character shape, the operator has to decide where to put the 'knots' on the character outline so that its shape is best approximated to by the curves that the system is capable of producing.

(Another contribution the drafting-system operator makes is to give a degree of consistency to the shape descriptions in their numerical form that they may not have had in their physical form. All the upright strokes of a typeface, for example,

---

[11] The material in the following paragraphs is derived largely from observations made between 1977 and 1984 on visits to D. Stempel AG, Frankfurt; URW, Hamburg; and Bitstream Inc., Cambridge, Mass. It is a pleasure to acknowledge the help of the personnel of these organizations.

can be made exactly vertical, or given the same slope. Part of the operator's skill is to know when to do this and when not to: *cf.* Ryder, 1979, pp. 76-77).

The next step in digital font production, after numerical descriptions of the character shapes have been made, is to use them to make fonts. The nature of the fonts depends on the nature of the interpreting process in the marking device for which they are intended (section 1.4 above): in terms of current technology, they will be either bitmap or outline specifications for character image shapes.

In either case, a further stage of interpretation is required. In making bitmap specifications, the weight and spacing of strokes in the output character images have to be specified in terms of the sizes and positions of the marks the marking device can make. This can be partly done by program (Elsner, 1980; Bigelow, 1981, p. 15) but some human intervention is still required at present, even with advanced systems (Matthew Carter; personal communication). With outline specifications, the weight and spacing of strokes in the output character image are decided by the interpreting program in the marking device. At present, programs that are small enough to go inside marking devices and fast enough to be useful there are not clever enough to make decisions of this sort on their own, and information has to be added by hand to the specifications to help them.

Where character shape descriptions have been produced by drafting systems, the task of interpretation at the font production stage is eased by the drafting-system operators' work in systematizing character shape descriptions, as well as correcting obvious errors and inconsistencies at the input stage. With high-resolution scanning, no interpretation at all takes place at the input stage, and the amount of work to be done at the font production stage is correspondingly increased.

Digital font production is therefore not simply a question of making literal translations of character shapes from graphical to numerical form. The translation that is made has to take into account not only the limited descriptive vocabulary of a drafting system, but also the limited shape-rendering ability of a raster-scan marking device.[12]

Thus, in order to be faithful, the translation process has to bear in mind the spirit of the typeface, rather than just the shapes of its characters: it has to remember that the visual attributes of the typeface are what have to be realized in the output character images. How to do this, particularly in the output from low and medium resolution devices, is a visual rather than a mathematical problem.

Compared to the huge task of translating existing designs into digital form, very little original design for digital composing systems has been done. It is noteworthy that in much of what original work has been done, designers have short-circuited the interpretive stages of digital font production by going to work directly on the pixel grid. Gerard Unger and Hermann Zapf have done this in their work for the Digiset, as have Matthew Carter and Ladislas Mandel in their designs for telephone directory composition (Bigelow, 1982b; Cooper Union, 1982; Unger, 1979).

This makes for an interesting comparison with design for hot-metal typecasting, where some designers at least would have dearly loved to get their hands directly on the the final stage of the font production process if it had been accessible to

[12] However high the resolution of such a device, the fact that the only marks it can make are straight lines of constant direction inevitably limits to some extent its ability to render arbitrarily complex – or rather, in type design, arbitrarily subtle – shapes.

them ***cf.*** van Krimpen, 1972). The digital designers' work has in fact been very closely analogous to the hand punchcutter's, in that working on the pixel grid has been the nearest they could get to working directly on the output character image.

The place of the glyph in digital font production

It is quite difficult to say what the glyphs are in digital font production of this kind. The problem is that because of the necessity for interpretation imposed by the limited shape-rendering capability of the marking device, the production process itself plays a large part in deciding the content of the character shape specifications in the font. (This is particularly true with bitmap fonts.) The character shapes that serve as input to the drafting system at the front end of the process are not the glyphs: the effect of the production process on their shapes cannot be predicted, and hence cannot be allowed for in their shapes; if there are any process-specific features in their shapes they are likely to have been put there to compensate for the effects of photography in the photomatrix production that was their original purpose. If we take the operational definition of the glyph that was suggested in section 2.2, as representing the last stage in the production process at which the designer or the producer could exercise control over the shape of the character image, then the product of the interpreter's work at the font production stage is the glyph, and the analogy between the designer working on the pixel grid and the hand punchcutter becomes closer still.

4 Metafont as a design system

Where does Metafont stand in all this? Because 'doing Metafont' is so completely unlike any activity that type designers have indulged in (or laboured at) in the past, we ought perhaps to begin by seeing how Metafont fits into the picture of type design and font production that has been developed in the previous two sections. Is Metafont a design system? And if not, is it a drafting system?

Of course, these questions are not well posed. 'Old Metafont' – Metafont79 – was a 'system' in that sense; it had very firm ideas about what tools to use for drawing character shapes and how to use them. But, in Knuth's words, 'its use should rapidly fade away' (Knuth, 1986, p. viii) as it is replaced by Metafont proper, which is not a system but a programming language. So the question should be: 'Can we use Metafont to build design systems or drafting systems?'

It would not be hard to use Metafont to build a drafting system like Ikarus, to take character shapes and produce numerical descriptions of them. Very many of the problems that beset the Euler project (Siegel, 1985), in which a drafting system was built with Metafont79, would be avoided in a similar system built with the new Metafont, because of the change of emphasis from pen tracks to outlines in the new version. However, making drafting systems does not address the problem that Metafont was developed to tackle: its author clearly intends it as a tool for building design systems.

4.1 A second look at design systems

A design system allows the designer to define new shapes for the characters of a typeface. We saw in Sections 2 and 3 above that if fonts are to be made that give rise to character images that successfully realize the visual attributes of the new typeface, there has to be easy and rapid communication between the type designer and the font producer.

In traditional type design (which means, in this context, everything that is not Metafont design), this communication between designer and producer has been carried on by means of an exchange of graphic objects. The designer has made drawings that provide graphic realizations of the visual attributes of the new typeface, and shown them to the font producer. The test for acceptability of the character images (apart from their technical quality) has been whether they realize the visual attributes of the typeface character shapes that the designer's drawings show: whether they look like the drawings. We can call a design system of this kind, where the designer makes drawings that realize the typeface character shapes, a graphic-mode design system: and the associated kind of font production system, in which communication between designer and producer is by graphic means, a graphic-mode production system.

It is essential in a graphic-mode design system that the designer should be able to make an exact realization of the character shape that is wanted. It is very important to understand that during almost all of the design process the designer

does not know exactly what ***that shape*** is. The designer will know a great deal about the technical characteristics of the shape: these, and most of its functional visual attributes, are dictated by the design specification for the typeface. If the typeface is to be stylistically coherent, the designer must also know what its stylistic visual attributes are. But knowing the visual attributes of the typeface is not enough: in graphic mode, they have to be realized as character shapes; and the designer does not know, at the beginning of the design task, what the shapes are that will realize the visual attributes of the typeface correctly. In graphic-mode design, the designer makes shapes and changes them until they look correct.'

Seen in this light, the implementation of Metafont that exists at present and is described in ***The Metafontbook*** (Knuth, 1986) is clearly not a graphic-mode design system of the traditional kind. It does not give its user the rapid and easy interaction with the shapes themselves that is essential to designing in the graphic mode.

Notice, though, that it is not a prerequisite for a design system that the definition of character shapes should be done graphically; nor is it essential to the success of a font production system that communication between designer and producer should be in a graphic mode. From the system's point of view, the only important thing about this communication is that it should be ***efficient***: that it should communicate effectively everything that needs to be communicated. Up until now, at least, the most efficient way to communicate information about the visual attributes of typeface character shapes has been for the designer to make graphic realizations of the shapes by drawing them, and show the drawings to the producer.

However, we can imagine another kind of design system: one in which the designer makes not shapes, but specifications for shapes. In such a system, a marking device of some kind makes shapes in response to the specifications the designer produces. The designer goes on making changes to the specifications until the marking device produces a shape that has the desired appearance. In terms of present-day technology, the designer's specifications have to be instructions to a computer, expressed by means of verbal and numerical symbols; so we can call a design system of this kind a symbolic-mode design system.

Such **a** design system could also provide direct symbolic input to a digital font production- system. This would eliminate the need for a drafting system and its attendant operator at the first stage of the production system. Indeed, it might allow the design and production processes to be integrated entirely, so that the

---

[1] The absolute necessity for the designer to be able to form exact judgements about the performance of a character shape in realizing a known set of visual attributes goes a long way towards explaining the doubts that many designers feel about the practicability of designing character shapes directly on the screen of an ordinary graphics workstation. Because the display distorts the shapes it presents, it is very hard to tell by looking at the screen what shape it is that has been made, and hence what the performance of the shape will be.

This is a much more serious difficulty than non-designers understand it to be. Most designers, at the stage of defining the shape of a character, feel that they need to locate the outline of the shape with an accuracy of around 0.2% of the capital letter height (Bigelow, **1982a,** p. 8; personal experience [RS]). Workstation manufacturers are notably coy about giving distortion performance figures for their displays; but in broadcast television practice only the most expensive picture monitors have geometrical distortions of less than about 1%.

results of the designer's work were realized directly *as* fonts A system like this would be a ***symbolic-mode production system.***

The question then is, can a symbolic-mode production system work? Can it produce fonts that give rise to technically satisfactory character images that successfully render the visual attributes of new typefaces?

A symbolic-mode production system will work (given a satisfactory symbolic-mode design system at its front end) if the knowledge and experience that is brought to bear by a human operator on the problem of interpreting a typeface character shape in digitized form can be incorporated in the programs that perform the task of font production in the system. Thus a working symbolic-mode production system requires a working symbolic-mode design system; and making the design system work does not necessarily solve the problem of making the production system work.

It is clear that Metafont79 was conceived as a symbolic-mode production system: one does not have to read far into the lecture in which Knuth first described his idea in order to see this (Knuth, 1979: ***Mathematical typography,*** **pp. 27–29**). However, we should not try to answer our question by looking at **Metafont79's** products. The only tools Metafont79 had for making character shapes were virtual pens of various shapes and sizes: this meant that in a practical sense, and for everyone except the system's creator, it was impossible to produce technically satisfactory character images with Metafont79.[2] The question is not 'Did Metafont79 work?' – the answer to that, in terms of the standards of typographic quality that are accepted outside the computer science community, is clearly 'No' – but 'Can a symbolic-mode production system, or a symbolic-mode design system, work at all?'

Charles Bigelow, in three internal discussion papers written for the Digital Typography Group in late 1983, argues eloquently that the answer to this question, too, is 'No'. It is unfair to the quality of his argument to try to summarize it, but a key point in it is that '. . . the designer thinks with images, not about images'. My contention is a similar one, though not exactly the same: it is that though designers know a great deal about the visual properties of character shapes, their knowledge is not usually available to them in a form in which it can be articulated symbolically. Designers know, for instance, that there are visual interactions between the elements of a character shape that affect the way it is perceived; they know also what the nature of these interactions are, and that they are governed by certain rules. But they cannot formulate these rules otherwise than by making shapes that take the effects of the interactions into account.

The point can be illustrated with a simple example. In a normal typeface, the bowls of small b and small p are visually related to each other; but they are not usually the same shape. They may ***appear to*** be the same shape, but to do so they will actually ***have to*** be different shapes. This is because the perceived shape of each bowl is affected by its visual interaction with the vertical stroke of the character, and the different positions of the vertical in the two characters – above and to the left in one, below and to the left in the other – mean that bowl shapes that are to be perceived as being the same will actually have to be different. A designer, asked to explain the rules that govern the visual interaction between the vertical and the bowl in b and p, will point to the difference in the shapes of the bowls. There are very few designers indeed who, when asked 'But why do the bowl shapes differ ***in***

[2] This was largely because many features, such as flat-bottomed notches at the junctions of strokes, that are needed in typeface character shapes for them to give rise to technically satisfactory output character images were exceedingly difficult to specify with Metafont79.

that particular way?' are able to explain the different consequences of the visual interactions in the two shapes otherwise than by pointing once again to the shapes themselves.

All type designers have had to learn about the visual effects that occur in the perception of small shapes. But the resulting knowledge is 'craft knowledge': it has become part of the intuitive understanding of the person concerned, and is not (and perhaps cannot be) articulated in an explicit form (Jones, 1970). Because this is so, and because until now all technically satisfactory typefaces have been designed with graphic-mode design systems, we do not at present have the theoretical basis for ***predicting*** the shapes of technically satisfactory typeface characters on which a successful symbolic-mode design system could be built. It is true that once the design of a technically satisfactory typeface has been completed, exact definitions of the shapes of all the characters in it do exist: but these definitions are graphic rather than symbolic, and the routes by which they were arrived at cannot be restated explicitly in algorithmic form.

4.2 The problem of meta-design

These doubts about the practicability of symbolic-mode design are reinforced when we consider the problem of meta-design. It has always been an essential part of Knuth's idea that 'by changing [typeface] parameters, we obtain infinitely many styles of type, yet all of them are related and they seem to blend harmoniously with one another' (Knuth, 1979: ***Mathematical typography,*** p. 31). ' . . . Once we have successfully explained how to draw something in a sufficiently general manner, the same explanation will work for different shapes, in related circumstances' (Knuth, 1986, p. 1).

In our terms, a meta-typeface is a typeface design in which the stylistic and functional visual attributes of the design have parameters associated with them. Each setting of the parameters defines a different typeface. The question about the practicability of meta-design then is 'Can parameterized character shape specifications be made such that every combination of parameter settings within a reasonable range defines a typeface that gives rise to technically satisfactory character images that realize visual attributes appropriate to the parameter settings?'

Once again, the fact that meta-typefaces made with Metafont79 do not give rise to technically satisfactory character images should not be allowed to discount the possibility of doing meta-design. The question, as with symbolic-mode design, is whether meta-design can be done at all, not whether it can be done with a particular tool.

In a meta-design for a typeface character, the specification for the character shape incorporates specifications for the changes in the shape that occur as a consequence of changes in the typeface parameters. In a symbolic specification for a character shape, these specifications will be in the form of functions that relate features of the character shape to values of the typeface parameters; and we can describe as ***a priori meta-design*** a design method in which these functions and their coefficients are specified explicitly by the designer.

Doing *a* priori meta-design in a way that ensures the eventual production of technically satisfactory character images for all reasonable combinations of typeface parameter settings requires the same thing that successful symbolic-mode design requires: explicit formulations of the rules that govern the visual interactions between the elements of character shapes.[3]

In contrast to this, one can imagine a way of doing meta-design a ***posteriori.*** In such a method the designer knows, and can express symbolically, the kinds of changes that will occur to the character shapes with changes in the typeface parameters. What the designer does not know initially is the size of the change to a given typeface parameter that is needed to make a particular change in a character shape. The designer therefore sets trial values for the coefficients of the functions that relate character shapes and typeface parameters; makes fonts and uses them to produce output character images; assesses the technical quality of the images and the extent to which they realize the visual attributes appropriate to the parameter settings; makes suitable changes to the coefficients; and so on.

The difficulty of doing ***a posteriori*** meta-design clearly increases with the complexity of the character shape as well as with the number of typeface parameters. It is not too difficult to make a meta-design for $\leq$. Mathematical symbols of this kind do not have stylistic visual attributes at all; the only functional attribute one might want to parameterize is boldness; and the features of the shape that change with boldness are the thickness of the strokes, the angle of the vee, the separation between the vee and the bar, and the alignment of the character as a whole. It is a great deal more difficult to make a meta-design for small ***h,*** where (leaving aside any features of the shape that depend on stylistic attributes) there are at least eight dimensions that vary with the single functional attribute of boldness.[4] Thus, while the present state of our formal knowledge about the visual interactions of shapes makes ***a priori*** meta-design impossible, ***a posteriori*** meta-design looks as if it might be just barely possible, given a great deal of dedication and effort on the designer's part.

Knuth's programs for the Computer Modern family of typefaces (Knuth, 1980) are examples of ***a posteriori*** meta-design in this sense. In writing the programs, the designer had a very clear idea of the way he wanted the character shapes to behave with changes in the typeface parameters. To a great extent, their behaviour was already embodied for him in the character shapes of the Monotype Modern Extended Series 8A he took as a model (Knuth, 1982, p. 7). Thus we can think of the Computer Modern project as being also an example of ***imitative meta-design***, where the objective is to make meta-typefaces that at certain settings of their parameters' realize the visual attributes of typefaces that already exist.'

---

[3] Notice that there appears to be no difficulty in doing ***a priori*** meta-design, or symbolic design, if one is not concerned with (or is unaware of) the technical quality of the result.

[4] The weight of the ascender; the weight of the vertical strokes below the arch; the weight of the arch where it springs from the vertical stroke, and the weight of its horizontal part; the distance from the top of the arch to the baseline; the separation between the left and right vertical strokes in the arch; and the left and right sidebearings.

[5] It seems clear that the primary motivation behind the development of both TEX78 and Metafont79 was to allow Knuth to reproduce the appearance of the volumes of ***The Art of Computer Programming*** that had been set with metal type (Knuth, 1979: ***Mathematical typography,*** pp. 16-17). It is interesting to speculate on the way TEX82 might have turned out if Knuth had worked with typographers during the revision of TEX78 in the same way that he worked with type designers during the revision of Metafont79: if, that is, there had been a shift in the system's objectives from the reproduction of existing typographic configurations to the origination of new ones.

5 Designing with Metafont

In Knuth's design of Computer Modern, roman and italic fonts already existed that had been fully developed to be technically satisfactory as metal type. Character images derived from them could serve as models for some of the typefaces the meta-typeface was to produce. How is the designer to proceed when no models exist: when the objective is to use Metafont to make a new design?

We should distinguish this situation from the one that obtained at the beginning of the Euler project (Siegel, 1985). In that project the designer was Hermann Zapf, and he treated Metafont and the Digital Typography Group as if they were a graphic-mode production system of the traditional kind. He made drawings for all the characters that were required and gave them to the group, who were to make fonts and submit their products for criticism and correction. The project was unexpectedly difficult and long-drawn-out because the font production process was ill-defined at the beginning, passed through a good many hands in its development, and later on made unanticipated contributions of its own to the shapes of the character images.

As a production system for bitmap fonts, Metafont has one great advantage over the more traditional methods described in section 3.4 above. In a Metafont program run, information about the positions of points on the outlines of character shapes is interpreted by the program in terms of the capability of the marking device for which the output of the run is intended (Knuth, 1986, ch. 24). This means that sets of character images derived from fonts produced by properly-written Metafont programs will have dimensional consistency of a kind that is at the very least laborious to achieve with traditional methods of digitization. In addition, this consistency will be present in the output from every marking device that Metafont knows about.

Metafont, with this feature that dimensions in a character shape specification adjust themselves to suit the capability of a given marking device, contrasts with a drafting system like Ikarus in which the the uninterpreted output describes the dimensions of an existing character shape in absolute terms. In a sense, a Metafont program contains instructions to a marking device about ***how to draw*** a character shape, rather than simply information about ***what shape*** the character is. This ability to capture what Knuth calls 'the "intelligence" that lies behind a design' (Knuth, 1985, p. 38) makes Metafont a potentially very valuable tool for the designer working with raster-scan output devices.

Unfortunately, in the present state of Metafont's development, the designer can only exploit this ability by writing programs; by working, that is, in symbolic mode. Given the impossibility of doing symbolic-mode design that was argued in Section 4 above, how should the designer who wants to use Metafont proceed?

The answer seems to be to work in a mixture of graphic and symbolic modes. With the idea of a typeface in mind, the first step is to make one or two character shapes that realize its visual attributes (*cf.* Fournier, vol. I, p. 16); the next, to write Metafont procedures that reproduce the components of these shapes and can be used to make other characters.

5.1 The problem of parameterization

It is clear that the procedures the designer writes need to be parameterized: one should not have to write different procedures to draw the arches in ***h, m, n*** and *u* simply because they are slightly different shapes. This implies not only that the procedures should behave reasonably over reasonable ranges of their own parameters, but also that the procedures' parameters should relate in a reasonable way to the parameters that describe the typeface as a whole. However unconvinced one is of the practicability of *a* priori meta-design, it is impossible to ignore the usefulness of a typeface-wide stroke-weight parameter to which the different weights in the arch of a character shape relate. There is then the problem of finding out what is the correct relationship between the weights in the arch and the stroke weight, and how the arch weights should change when the stroke weight changes.

In this mixed-mode method of working, the natural way in which the notion of **'meta-ness'** can be accommodated in a symbolic-mode design becomes evident straight away: as does the difficulty of implementing a meta-design in practice, The problem of finding out the relationships between the values of the parameters to procedures that function at the character level (like the weight parameters to an arch-drawing procedure) and the values of parameters at the typeface level (like the stroke-weight parameter) occurs over and over again for every character of the design.

The root of this problem is that the consistency of weight, shape and spacing that is essential in technically satisfactory character images (section 2.1 above) is a consistency of appearance, rather than one of actual dimensions. A designer who knows about the effects that the human visual system has on the perception of shapes can add to the Metafont programs parameters that take those effects into account: so that, for example, the program for small ***f*** has a parameter that controls the difference in the weight of the vertical above and below the crossbar. The difficulty is that such parameters have to be given their correct values, and the relationships between procedure parameters and typeface parameters have to be properly defined. These values and relationships can be written into the character programs; but the only way the designer can be assured of the correctness of the program for a character shape is by testing it. This testing has to be repeated over a range of settings for every parameter under consideration, and at every repetition it involves for the designer a change from the familiar graphic mode to the less familiar and less congenial symbolic mode. And even when every character program has been tested, the designer is only assured of the correctness of the design in the particular combinations of output image size and output device resolution in which the testing has been done.

abcdefghijlmnopqrstu

nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnun
oaobocodoeofogohoiojolomonooopoqoorsotouo

hamburgefons

abcdefghijlmnopqrstu

nanbncndnenfngnhninjnlnmnnnonpnqnrnsntnun
oaobocodoeofogohoiojolomonooopoqoorsotouo

hamburgefons

abcdefghijlmnopqrstu

nanbncndnenfngnhninjnlnmnnnonpnqnmsntnun
oaobocodoeofogohoiojolomonooopoqoorsotouo

hamburgefons

An experimental typeface design made with Metafont.

The nominal sizes of the three fonts are approximately 5 pt, 8 pt and 10 pt.

5.2 An example of Metafont design

This approach to doing type design with Metafont is illustrated by the character images shown here. The programs that specify them were developed by the method just described. So far, no attempt has been made to incorporate **'meta-ness'** into the designs, except for the adjustments to character proportions, stroke weights and sidebearings that are made in going from size to size.

These size-dependent adjustments are made in a way that differs from the approach that Knuth adopts in his Computer Modern design. In his design, Knuth is aiming at a known selection of nominal output image sizes, and can provide explicit settings for all the typeface parameters at each size. My approach (influenced perhaps by experience with the outline fonts of the Linotype **CRTronic** photocomposing machine) has been to use a single parameter to specify output image size, and allow all the size-dependent typeface parameters to adjust themselves to suitable values. This is done with a simple implementation of the non-linear interpolation idea, due to John Hobby, that is described in Appendix D of ***The Metafontbook***. The task is made easier because there are relatively few typeface parameters: this is because of the deliberately-chosen simplicity of the character shapes. (The option of choosing simple character shapes was not available to Knuth; because of his objectives, he was obliged in the Computer Modern design to imitate existing shapes, some of whose features are extremely unsuitable for digitization.)

The design shown here has been developed and tested on the Imagen **8/300** printer, for **a** range of capital letter heights between 1.25 mm and 2.5 mm, corresponding to a nominal size range of 5-10 pt (20-40 pixels at 300 spots per inch). Tests of the design on the QMS **Lasergrafix** 1200 printer, which has the same resolution as the Imagen but a different marking device, have vividly illustrated the necessity for taking the characteristics of the eventual marking process into account when developing the content of a glyph.

6 Conclusions

It is possible to derive some conclusions about designing with Metafont from the work that was described in the previous section, as well as from the survey of type design methods that was presented in Section 3.

6.1 The problem of device-independent design

There does seem to be a definite difference of opinion between computer scientists and type designers about the practicability of making typeface designs that are independent of the marking device on which the output is to be produced. Knuth's view in 1985 seems to be essentially the same as it was in 1978: '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 difference between this and the type designers' view is suggested by **Herrman** Zapf's statements in his lecture at the **ATypI** seminar 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. As an example, digitizing my Optima roman presented many difficulties. The well-balanced shape of the stems is contrary to the digital principle, especially in low resolutions, some of which go down to 300 lines per inch. The design must be reduced to a heart-breaking compromise. The answer to this problem is that Optima was never designed for digital storage. If I had been asked, I would have done a new design, used another principle and another name, but would have tailored it to the needs and limitations of today's equipment . . . Using an interactive process and a display screen, the designer will work out the best solution or the best compromise between the original idea and the image of the generated letter' (Zapf, 1985, pp. 29, 33).

My own view is that type design is a visual business; designers work by looking at character images; and it is inconceivable that one could make satisfactory designs for a machine whose output one could not see.

6.2 The problem of programming

Any designer working with Metafont will soon experience the conflict between doing good designing and doing good programming. It is not so much that the two are necessarily opposed, as that either uses up time that would be better spent on the other. Most designers are better at designing than they are at programming, and their first priority is to get a shape to look at. The temptation is then to hack together **a** procedure that will draw the shape in question. Even if a small change in its parameters does not destroy the procedure the next minute, an attack of conscience will probably tear it apart the next day. The designer consequently spends a lot of time in what are essentially literary labours, rewriting programs that might have been better written in the first place if the longing to ***see something*** had not been so strong.

This raises the obvious question whether the designer should work together with a programmer. Knuth evidently thinks so (Knuth, 1986, p.v); I am not so sure. The problem is once again one of communication: this time between designer and programmer. In doing typographic design with TEX (where my own experience suggests that designer-programmer collaboration can be successful) the designer provides the programmer with dimensioned drawings that show the location of graphic elements in a document, and the design is developed by altering the positions of the elements of the document in a relatively simple way. Communication between designer and programmer can therefore be carried on successfully in a numerical mode. In type design, a dimensioned structure exists as well, but it serves simply as a scaffolding on which the character shape is built. The stage in type design at which the character shapes themselves start to be defined is the stage at which numerical communication breaks down: the designer can only describe the shapes that are required by making them.

An alternative mode of designer-programmer communication corresponds to the design method adopted by Pierre Didot (section 3.1 above). The programmer writes a program that makes character shapes, and shows the results to the designer: the designer says 'No, that's not right'. Parts of the Euler project proceeded in this fashion (Siegel, 1985). The problem, already discussed, is that it is much easier for the designer to see that a shape is wrong than to understand what it was in the program that made it wrong.

Another aspect of the problem of programming is the problem of conflicting styles. The programs the designer works with have to correspond with the designer's working methods. The effort that has gone into the development of well-behaved pseudo-pens in the new Metafont (Knuth, 1986, ch. 4, ch. 16; Hobby, 1985) is wasted on designers who (like myself) think about character shapes in terms of outlines rather than pen tracks.[1]

Perhaps the best (at any rate, the most positive) way of looking at the problem of programming is to consider the designer's knowledge of programming as being equivalent to the punchcutter's knowledge of metalworking: something that has to be learned to make possible the real business of getting the character shapes defined. After all, punchcutters have always made their own tools (Drost, 1985, p. 101).

The conflict of aesthetics in programming

The programmer's remark that 'the character shapes must be right, because the programs are right' is not entirely a malicious fabrication by the designers. All one can say in reply is that it is the character shapes, not the programs, that the reader sees. 'Unlike many trades, in which indifferent productions find an employment proportionate to their worth, printing must have the best: not even the second best will serve; for it costs as much to cast and print ill-cut letters as to cast and print the very best.'[2] The whole history of 'ideal' character design, from Cresci

[1] I believe also that it is possible to argue a convincing case that the 'pen model' of character shape construction is badly adapted to the production of technically satisfactory character shape specifications for medium-resolution marking devices.

[2] Fournier, vol. I, p. 3; Carter's translation.

to the present day, demonstrates the failure of elegant constructions to produce satisfactory character shapes (Knuth, 1979: ***Mathematical typography,*** pp. 18-21).

6.3 Is Metafont useful as a design tool?

I think that on balance, and with many reservations and qualifications, the answer in the end has to be 'Yes'. Metafont's ability to do 'intelligent digitization', and the possibility it offers of making size-dependent changes to typeface parameters, are immensely valuable features that are hard to achieve with other systems. Siegel argues that, even in spite of its unfortunate history, the Euler project eventually achieved a productivity that was comparable with that of conventional systems for the production of digital type. Knuth's rate of work in making the programs for Computer Modern has been spectacular.

However, neither in the Euler project nor in the development of Computer Modern was Metafont used as a tool for original design. It has to be said that Metafont is unbelievably difficult to use as a tool for making new typeface designs. The reason for this is its lack of interactivity, and the consequent necessity for continual switching on the designer's part between symbolic and graphic modes. At every stage, the program intervenes between the idea of the character shape and its realization. This is the price that Metafont exacts for its descriptive approach to character image construction.

Kathy Carter at Cambridge, and Eliyezer Kohen at **Zürich**, have built systems that allow designers to make outlines directly on a visual display (Carter, 1984; Kohen, to be published). However, both these systems suffer from the same disadvantage that drafting systems have: the only information the designer can incorporate in the shape is the information the shape itself contains. If Metafont has one overriding advantage, it is that it gives the designer a means of expressing ***why*** a shape is to be made in a particular way. It is hard to see how this can be done at present otherwise than by symbolic specification of character shapes.

References

Andre, J., and Southall, R.
Experiments in teaching Metafont
in Lucarella, D. (ed.), ***TEX for scientific documentation.*** Reading, Mass.: Addison-Wesley, 1985

Atkins, G.
The classification** of **printing types
Oadby, Leicestershire: Apple Barrell, 1975

Bigelow , C.
Aesthetics vs. technology
Seybold Report, vol. 10 no. 24, pp. 3-16 (1981)

Bigelow , C.
The principles of digital type – I
Seybold Report, vol. 11 no. 11, pp. 3-23 (1982)

Bigelow , C.
The principles of digital type – II
Seybold Report, vol. 11 no. 12, pp. 10–19 (1982)

Bigelow, C. and Day, D.
Digital typography
Scientific *American,* vol. 249 no. 2, pp. 106-119 (1983)

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

Carter, H. G.
The optical scale in typefounding
Typography, no. 4, pp. 2-6 (1937)

Carter, H. G.
A view** of **early typography
Oxford: Clarendon Press, 1979

Carter, K. A.
IMP – A system for computer-aided typeface design
in J. J. H. Miller (ed.), ***Proceedings** of **the first international conference on** text **processing systems.*** Dublin: Boole, 1984

Carter, M.
Galliard: a modern revival of the types of Robert Granjon
Visible *Language,* vol. 19 no. 1, pp. 77-97 (1985)

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

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

Didot, P.
Specimen des nouveuux caractères . . .
Paris: Didot, 1819

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

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

Drost, H.
Punchcutting demonstration
Visible Language, vol. 19 no. 1, pp. 99–105 (1985)

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

Elsner, V.
Ikarus: computergesteuerte Vorlagernerstellung für Foto-, CRT- und Lasersatz
Typographische Monutsbldtter, no. 2, pp. 69-79 (1980)

Flowers, J.
Digital type manufacture: an interactive approach
IEEE Computer, pp. 40-48 (May 1984)

Fournier , S.-P.
Manuel typographique
Paris: Fournier, 1764 (vol. I), 1766 (vol. II)

Frutiger, A.
Letterforms in phototypography
Visible Language, vol. 4 no. 4, pp. 327-335 (1970)

Frutiger, A.
Type sign ***symbol***
Zürich: ABC Verlag, 1980

Gill, E.
An essay on typography
London: Sheed & Ward, 1931

Goudy, F. W.
Typologia
Berkeley, Ca.: University of California Press, 1940

Gürtler, A., Mengelt, C. and Gschwind, E.
Von der Helvetica – zur Haas Unica
Miinchenstein, Switzerland: Haas'sche Schriftgiesserei (n.d.)

Hegron, G.
Algorithmes de generation de caractires
in Andre, J., and Sallio, P. (eds.), ***Typogruphie et informutique. Support du cours INRIA, Rennes, 21–25 Junvier 1985.*** Rocquencourt: INRIA, 1985

Hobby, J. D.
Digitized brush trajectories
Stanford Computer Science Department report STAN-CS-85-1070 (1985)

Jammes, A.
La reforme de la typogruphie royale sous Louis XIV: le Grundjean
Paris: Jammes, 1961

Jones, J. C.
Design methods
London: Wiley-Interscience, 1970

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

Knuth, D.E.
The Computer Modern family of typefaces
Stanford Computer Science Department report STAN-CS-80-780 (1980)

Knuth, D. E.
The Concept of a Meta-Font
Visible ***Language,*** vol. 16 no. 1, pp. 3-27 (1982)

Knuth, D. E.
Lessons learned from Metafont
Visible Language, vol. 19 no. **1,** pp. 35-53 (1985)

Knuth, D. E.
The Metafontbook
Reading, Mass.: Addison-Wesley (to be published in 1986)

Kohen, E.
An* interactive *method for middle resolution font design ***on personal workstations***
Zurich: Institut **für** Informatik, ETH (to be published)

Legros, L. A. and Grant, J. C.
Typographical printing-surfaces
London: Longmans, Green, 1916

Monotype Corporation
Monotype Recorder, vol.40 no. 3 (1956)

Morison, S.
The design and cutting of The Times new roman: how matrices are made
Monotype Recorder, vol. 31 no. 246, pp. 12-15 (1932)

Moxon, J .
Mechanick* exercises *on the whole art of printing (1683)
London: Oxford, 1958; New York: Dover, 1978

Nelson, S.
Mould making, matrix fitting and hand casting
Visible ***Language,*** vol. 19 no. 1, pp. 107-120 (1985)

Pirozzolo, F. J., and Wittrock, M. C. (eds.)
Neuropsychologicul and cognitive processes* in *reading
New York: Academic Press, 1981

Quint, V.
Une approche de l'ddition interactive en deux dimensions
in Andre, J. (ed.), ***Actes des journées sur la manipulation de documents,*** *Rennes,* ***4-6 Mai* 1989.** Rocquencourt: INRIA, 1983

Ruggles, L.
Letterform design systems
Stanford Computer Science Department report STAN-CS-83-971 (1983)

Ryder, J.
The case for legibility
New York: Moretus, 1979

Seybold, J. W.
The world** of **digital typesetting
Media, Pa.: Seybold, 1984

Shurtleff, D. A.
How to make displays legible
La Mirada, Ca.: Human Interface Design, 1980

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

Southall, R.
A new index of typefaces
Penrose, vol. 74, pp. 255-258 (1982)

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

Spencer, H.
The* visible *word
New York: Hastings House, 1969

Tzeng, 0. J . L., and Singer, H. (eds.)
Perception** of **print: reading research* in *experimental psychology
Hillsdale, N.J.: Lawrence Erlbaum, 1981

Unger, G.
The design of a typeface
Visible *Language,* vol. 13 no. 2, pp. 134-149 (1979)

van Krimpen, J.
On* designing *and* devising *type
New York: The Typophiles, 1957

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

Visible *Language*
Special issue: Psychological processes in reading
Visible *Language,* vol. 18 no. 4 (1984)

Walker, S. F.
Descriptive techniques* for *studying verbal graphic language
Unpublished thesis: Department of Typography & Graphic Communication, University of Reading, England (1983)

Waller, R. W.
Graphic aspects of complex texts: typography as macro-punctuation
in Kolers, P. A., Wrolstad, M. E. and Bouma, H. (eds.), ***Processing** of **visible language, vol.*** 2. New York: Plenum, 1980

Zapf, H.
***About alphabets: some marginal notes on type* design**
New York: The Typophiles, 1960

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