The METAFONTbook

The fine print in the upper right-hand
corner of each page is a draft of intended
index entries; it won't appear in the real book.
Some index entries will be in `typewriter type`
and/or enclosed in ⟨...⟩, etc;
such typographic distinctions aren't shown here.
An index entry often extends for several pages;
the actual scope will be determined later.
Please note things that should be indexed but aren't.

Apology: The xeroxed illustrations are often hard to see;
they will be done professionally in the real book.

The METAFONTbook

DONALD E. KNUTH *Stanford University*

Knuth, Donald Ervin
Bibby, Duane Robert

Illustrations by
DUANE BIBBY

ADDISON–WESLEY PUBLISHING COMPANY

Reading, Massachusetts
Menlo Park, California
New York
Don Mills, Ontario
Wokingham, England
Amsterdam · Bonn
Sydney · Singapore · Tokyo
Madrid · San Juan

Palais
Wilkins
Tobin
Knuth, Donald Ervin

This manual describes METAFONT Version 2.0. Some of the advanced features mentioned here are absent from earlier versions.

The joke on page 8 is due to Richard S. Palais.

The Wilkins quotation on page 283 was suggested by Georgia K. M. Tobin.

Library of Congress cataloging in publication data

```
Knuth, Donald Ervin, 1938-
   The METAFONTbook.

   (Computers & Typesetting ; C)
   Includes index.
   1. METAFONT (Computer system).  2. Type and type-
founding--Data processing.  I. Title.  II. Series:
Knuth, Donald Ervin, 1938-    .  Computers &
typesetting ; C.
Z250.8.M46K58  1986         686.2'24      85-28675
ISBN 0-201-13445-4
ISBN 0-201-13444-6 (soft)
```

Incorporates the final corrections made in 1995.

Internet page `http://www-cs-faculty.stanford.edu/~knuth/abcde.html` contains current information about this book and related books.

ISBN 0-201-13445-4
6 7 8 9 10 11 12–CRS–9998979695

Zapf, Hermann

To Hermann Zapf:
Whose strokes are the best

Preface

beauty

GENERATION OF LETTERFORMS by mathematical means was first tried in the fifteenth century; it became popular in the sixteenth and seventeenth centuries; and it was abandoned (for good reasons) during the eighteenth century. Perhaps the twentieth century will turn out to be the right time for this idea to make a comeback, now that mathematics has advanced and computers are able to do the calculations.

Modern printing equipment based on raster lines—in which metal "type" has been replaced by purely combinatorial patterns of zeroes and ones that specify the desired position of ink in a discrete way—makes mathematics and computer science increasingly relevant to printing. 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. Moreover, the shapes can be defined in terms of variable parameters; computers can "draw" new fonts of characters in seconds, making it possible for designers to perform valuable experiments that were previously unthinkable.

METAFONT is a system for the design of alphabets suited to raster-based devices that print or display text. The characters that you are reading were all designed with METAFONT, in a completely precise way; and they were developed rather hastily by the author of the system, who is a rank amateur at such things. It seems clear that further work with METAFONT has the potential of producing typefaces of real beauty. This manual has been written for people who would like to help advance the art of mathematical type design.

A top-notch designer of typefaces needs to have an unusually good eye and a highly developed sensitivity to the nuances of shapes. A top-notch user of computer languages needs to have an unusual talent for abstract reasoning and a highly developed ability to express intuitive ideas in formal terms. Very few people have both of these unusual combinations of skills; hence the best products of METAFONT will probably be collaborative efforts between two people who complement each other's abilities. Indeed, this situation isn't very different from the way types have been created for many generations, except that the rôle of "punch-cutter" is now being played by skilled computer specialists instead of by skilled metalworkers.

A METAFONT user writes a "program" for each letter or symbol of a typeface. These programs are different from ordinary computer programs, because they are essentially *declarative* rather than imperative. In the METAFONT language you explain where the major components of a desired shape are

to be located, and how they relate to each other, but you don't have to work out the details of exactly where the lines cross, etc.; the computer takes over the work of solving equations as it deduces the consequences of your specifications. One of the advantages of METAFONT is that it provides a discipline according to which the principles of a particular alphabet design can be stated precisely. The underlying intelligence does not remain hidden in the mind of the designer; it is spelled out in the programs. Thus consistency can readily be obtained where consistency is desirable, and a font can readily be extended to new symbols that are compatible with the existing ones.

It would be nice if a system like METAFONT were to simplify the task of type design to the point where beautiful new alphabets could be created in a few hours. This, alas, is impossible; an enormous amount of subtlety lies behind the seemingly simple letter shapes that we see every day, and the designers of high-quality typefaces have done their work so well that we don't notice the underlying complexity. One of the disadvantages of METAFONT is that a person can easily use it to produce poor alphabets, cheaply and in great quantity. Let us hope that such experiments will have educational value as they reveal why the subtle tricks of the trade are important, but let us also hope that they will not cause bad workmanship to proliferate. Anybody can now produce a book in which all of the type is home-made, but a person or team of persons should expect to spend a year or more on the project if the type is actually supposed to look right. METAFONT won't put today's type designers out of work; on the contrary, it will tend to make them heroes and heroines, as more and more people come to appreciate their skills.

Although there is no royal road to type design, there are some things that can, in fact, be done well with METAFONT in an afternoon. Geometric designs are rather easy; and it doesn't take long to make modifications to letters or symbols that have previously been expressed in METAFONT form. Thus, although comparatively few users of METAFONT will have the courage to do an entire alphabet from scratch, there will be many who will enjoy customizing someone else's design.

This book is not a text about mathematics or about computers. But if you know the rudiments of those subjects (namely, contemporary high school mathematics, together with the knowledge of how to use the text editing or word processing facilities on your computing machine), you should be able to use METAFONT with little difficulty after reading what follows. Some parts

dangerous bend

of the exposition in the text are more obscure than others, however, since the author has tried to satisfy experienced METAFONTers as well as beginners and casual users with a single manual. Therefore a special symbol has been used to warn about esoterica: When you see the sign

at the beginning of a paragraph, watch out for a "dangerous bend" in the train of thought—don't read such a paragraph unless you need to. You will be able to use METAFONT reasonably well, even to design characters like the dangerous-bend symbol itself, without reading the fine print in such advanced sections.

Some of the paragraphs in this manual are so far out that they are rated

 ;

everything that was said about single dangerous-bend signs goes double for these. You should probably have at least a month's experience with METAFONT before you attempt to fathom such doubly dangerous depths of the system; in fact, most people will never need to know METAFONT in this much detail, even if they use it every day. After all, it's possible to fry an egg without knowing anything about biochemistry. Yet the whole story is here in case you're curious. (About METAFONT, not eggs.)

The reason for such different levels of complexity is that people change as they grow accustomed to any powerful tool. When you first try to use METAFONT, you'll find that some parts of it are very easy, while other things will take some getting used to. At first you'll probably try to control the shapes too rigidly, by overspecifying data that has been copied from some other medium. But later, after you have begun to get a feeling for what the machine can do well, you'll be a different person, and you'll be willing to let METAFONT help contribute to your designs as they are being developed. As you gain more and more experience working with this unusual apprentice, your perspective will continue to change and you will run into different sorts of challenges. That's the way it is with any powerful tool: There's always more to learn, and there are always better ways to do what you've done before. At every stage in the development you'll want a slightly different sort of manual. You may even want to write one yourself. By paying attention to the dangerous bend signs in this book you'll be better able to focus on the level that interests you at a particular time.

JOKES
truth
EXERCISES
MF79
Hobby

Computer system manuals usually make dull reading, but take heart: This one contains JOKES every once in a while. You might actually enjoy reading it. (However, most of the jokes can only be appreciated properly if you understand a technical point that is being made—so read *carefully*.)

Another noteworthy characteristic of this book is that it doesn't always tell the truth. When certain concepts of METAFONT are introduced informally, general rules will be stated; afterwards you will find that the rules aren't strictly true. In general, the later chapters contain more reliable information than the earlier ones do. The author feels that this technique of deliberate lying will actually make it easier for you to learn the ideas. Once you understand a simple but false rule, it will not be hard to supplement that rule with its exceptions.

In order to help you internalize what you're reading, EXERCISES are sprinkled through this manual. It is generally intended that every reader should try every exercise, except for questions that appear in the "dangerous bend" areas. If you can't solve a problem, you can always look up the answer. But please, try first to solve it by yourself; then you'll learn more and you'll learn faster. Furthermore, if you think you do know the solution, you should turn to Appendix A and check it out, just to make sure.

> WARNING: Type design can be hazardous to your other interests. Once you get hooked, you will develop intense feelings about letterforms; the medium will intrude on the messages that you read. And you will perpetually be thinking of improvements to the fonts that you see everywhere, especially those of your own design.

The METAFONT language described here has very little in common with the author's previous attempt at a language for alphabet design, because five years of experience with the old system has made it clear that a completely different approach is preferable. Both languages have been called METAFONT; but henceforth the old language should be called METAFONT79, and its use should rapidly fade away. Let's keep the name METAFONT for the language described here, since it is so much better, and since it will never change again.

I wish to thank the hundreds of people who have helped me to formulate this "definitive edition" of METAFONT, based on their experiences with preliminary versions of the system. In particular, John Hobby discovered many of

National Science Foundation
Office of Naval Research
IBM Corporation
System Development Foundation
American Mathematical Society
TUGboat
Knuth, Jill
Knuth, Don
BIERCE
MORISON

the algorithms that have made the new language possible. My work at Stanford has been generously supported by the National Science Foundation, the Office of Naval Research, the IBM Corporation, and the System Development Foundation. I also wish to thank the American Mathematical Society for its encouragement and for publishing the *TUGboat* newsletter (see Appendix J). Above all, I deeply thank my wife, Jill, for the inspiration, understanding, comfort, and support she has given me for more than 25 years, especially during the eight years that I have been working intensively on mathematical typography.

Stanford, California — D. E. K.
September 1985

It is hoped that Divine Justice may find
some suitable affliction for the malefactors
who invent variations upon the alphabet of our fathers. . . .
The type-founder, worthy mechanic, has asserted himself
with an overshadowing individuality,
defacing with his monstrous creations and revivals
every publication in the land.

— AMBROSE BIERCE, *The Opinionator. Alphabêtes* (1911)

Can the new process yield a result that, say,
a Club of Bibliophiles would recognise as a work of art
comparable to the choice books they have in their cabinets?

— STANLEY MORISON, *Typographic Design in Relation to Photographic Composition* (1958)

Contents

Contents of this manual, table

1

The Name of the Game

TeX
METAFONT, the name
meta-font

This is a book about a computer system called METAFONT, just as *The TEXbook* is about TEX. METAFONT and TEX are good friends who intend to live together for a long time. Between them they take care of the two most fundamental tasks of typesetting: TEX puts characters into the proper positions on a page, while METAFONT determines the shapes of the characters themselves.

Why is the system called METAFONT? The '-FONT' part is easy to understand, because sets of related characters that are used in typesetting are traditionally known as fonts of type. The 'META-' part is more interesting: It indicates that we are interested in making high-level descriptions that transcend any of the individual fonts being described.

Newly coined words beginning with 'meta-' generally reflect our contemporary inclination to view things from outside or above, at a more abstract level than before, with what we feel is a more mature understanding. We now have metapsychology (the study of how the mind relates to its containing body), metahistory (the study of principles that control the course of events), metamathematics (the study of mathematical reasoning), metafiction (literary works that explicitly acknowledge their own forms), and so on. A metamathematician proves metatheorems (theorems about theorems); a computer scientist often works with metalanguages (languages for describing languages). Similarly, a meta-font is a schematic description of the shapes in a family of related fonts; the letterforms change appropriately as their underlying parameters change.

Meta-design is much more difficult than design; it's easier to draw something than to explain how to draw it. One of the problems is that different sets of potential specifications can't easily be envisioned all at once. Another is that a computer has to be told absolutely everything. However, once we have successfully explained how to draw something in a sufficiently general manner, the same explanation will work for related shapes, in different circumstances; so the time spent in formulating a precise explanation turns out to be worth it.

Typefaces intended for text are normally seen small, and our eyes can read them best when the letters have been designed specifically for the size at which they are actually used. Although it is tempting to get 7-point fonts by simply making a 70% reduction from the 10-point size, this shortcut leads to a serious degradation of quality. Much better results can be obtained by incorporating parametric variations into a meta-design. In fact, there are advantages to built-in variability even when you want to produce only one font of type in a single size, because it allows you to postpone making decisions about many aspects of your design. If you leave certain things undefined, treating them as parameters instead of "freezing" the specifications at an early stage, the computer will be able to draw lots of examples with different settings of the parameters, and you will be able to see the results of all those experiments at the final size. This will greatly increase your ability to edit and fine-tune the font.

If meta-fonts are so much better than plain old ordinary fonts, why weren't they developed long ago? The main reason is that computers did not exist until recently. People find it difficult and dull to carry out calculations

with a multiplicity of parameters, while today's machines do such tasks with ease. The introduction of parameters is a natural outgrowth of automation.

interpolate
recipe

OK, let's grant that meta-fonts sound good, at least in theory. There's still the practical problem about how to achieve them. How can we actually specify shapes that depend on unspecified parameters?

If only one parameter is varying, it's fairly easy to solve the problem in a visual way, by overlaying a series of drawings that show graphically how the shape changes. For example, if the parameter varies from 0 to 1, we might prepare five sketches, corresponding to the parameter values 0, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, and 1. If these sketches follow a consistent pattern, we can readily interpolate to find the shape for a value like $\frac{2}{3}$ that lies between two of the given ones. We might even try extrapolating to parameter values like $1\frac{1}{4}$.

But if there are two or more independent parameters, a purely visual solution becomes too cumbersome. We must go to a verbal approach, using some sort of language to describe the desired drawings. Let's imagine, for example, that we want to explain the shape of a certain letter 'a' to a friend in a distant country, using only a telephone for communication; our friend is supposed to be able to reconstruct exactly the shape we have in mind. Once we figure out a sufficiently natural way to do that, for a particular fixed shape, it isn't much of a trick to go further and make our verbal description more general, by including variable parameters instead of restricting ourselves to constants.

An analogy to cooking might make this point clearer. Suppose you have just baked a delicious berry pie, and your friends ask you to tell them the recipe so that they can bake one too. If you have developed your cooking skills entirely by intuition, you might find it difficult to record exactly what you did. But there is a traditional language of recipes in which you could communicate the steps you followed; and if you take careful measurements, you might find that you used, say, $1\frac{1}{4}$ cups of sugar. The next step, if you were instructing a computer-controlled cooking machine, would be to go to a meta-recipe in which you use, say, $.25x$ cups of sugar for x cups of berries; or $.3x + .2y$ cups for x cups of boysenberries and y cups of blackberries.

In other words, going from design to meta-design is essentially like going from arithmetic to elementary algebra. Numbers are replaced by simple formulas that involve unknown quantities. We will see many examples of this.

A METAFONT definition of a complete typeface generally consists of three main parts. First there is a rather mundane set of subroutines that take care of necessary administrative details, such as assigning code numbers to individual characters; each character must also be positioned properly inside an invisible "box," so that typesetting systems will produce the correct spacing. Next comes a more interesting collection of subroutines, designed to draw the basic strokes characteristic of the typeface (e.g., the serifs, bowls, arms, arches, and so on). These subroutines will typically be described in terms of their own special parameters, so that they can produce a variety of related strokes; a serif subroutine will, for example, be able to draw serifs of different lengths, although all of

Hobby
Gu
Chinese characters
Kandinsky
Herbin
HULTÉN
Tinguely
KNUTH

the serifs it draws should have the same "feeling." Finally, there are routines for each of the characters. If the subroutines in the first and second parts have been chosen well, the routines of the third part will be fairly high-level descriptions that don't concern themselves unnecessarily with details; for example, it may be possible to substitute a different serif-drawing subroutine without changing any of the programs that use that subroutine, thereby obtaining a typeface of quite a different flavor. [A particularly striking example of this approach has been worked out by John D. Hobby and Gu Guoan in "A Chinese Meta-Font," *TUGboat* **5** (1984), 119–136. By changing a set of 13 basic stroke subroutines, they were able to draw 128 sample Chinese characters in three different styles (Song, Long Song, and Bold), using the same programs for the characters.]

A well-written METAFONT program will express the designer's intentions more clearly than mere drawings ever can, because the language of algebra has simple "idioms" that make it possible to elucidate many visual relationships. Thus, METAFONT programs can be used to communicate knowledge about type design, just as recipes convey the expertise of a chef. But algebraic formulas are not easy to understand in isolation; METAFONT descriptions are meant to be read with an accompanying illustration, just as the constructions in geometry textbooks are accompanied by diagrams. Nobody is ever expected to read the text of a METAFONT program and say, "Ah, what a beautiful letter!" But with one or more enlarged pictures of the letter, based on one or more settings of the parameters, a reader of the METAFONT program should be able to say, "Ah, I understand how this beautiful letter was drawn!" We shall see that the METAFONT system makes it fairly easy to obtain annotated proof drawings that you can hold in your hand as you are working with a program.

Although METAFONT is intended to provide a relatively painless way to describe meta-fonts, you can, of course, use it also to describe unvarying shapes that have no "meta-ness" at all. Indeed, you need not even use it to produce fonts; the system will happily draw geometric designs that have no relation to the characters or glyphs of any alphabet or script. The author occasionally uses METAFONT simply as a pocket calculator, to do elementary arithmetic in an interactive way. A computer doesn't mind if its programs are put to purposes that don't match their names.

[Tinguely] made some large, brightly coloured open reliefs,
juxtaposing stationary and mobile shapes.
He later gave them names like Meta-Kandinsky *and* Meta-Herbin,
to clarify the ideas and attitudes that lay at the root of their conception.
— K. G. PONTUS HULTÉN, *Jean Tinguely: Méta* (1972)

The idea of a meta-font should now be clear. But what good is it?
The ability to manipulate lots of parameters may be interesting and fun,
but does anybody really need a $6\frac{1}{7}$*-point font*
that is one fourth of the way between Baskerville and Helvetica?
— DONALD E. KNUTH, *The Concept of a Meta-Font* (1982)

2

Coordinates

If we want to tell a computer how to draw a particular shape, we need a way to explain where the key points of that shape are supposed to be. METAFONT uses standard *Cartesian coordinates* for this purpose: The location of a point is defined by specifying its x coordinate, which is the number of units to the right of some reference point, and its y coordinate, which is the number of units upward from the reference point. First we determine the horizontal (left/right) component of a point's position, then we determine the vertical (up/down) component. METAFONT's world is two-dimensional, so two coordinates are enough.

Cartesian coordinates
x coordinate
y coordinate
graph paper
raster
grid
pixels
pel, see pixel

For example, let's consider the following six points:

(Figure 2a will be inserted here; too bad you can't see it now.)

METAFONT's names for the positions of these points are

$$(x_1, y_1) = (0, 100); \quad (x_2, y_2) = (100, 100); \quad (x_3, y_3) = (200, 100);$$
$$(x_4, y_4) = (0, \ \ 0); \quad (x_5, y_5) = (100, \ \ 0); \quad (x_6, y_6) = (200, \ \ 0).$$

Point 4 is the same as the reference point, since both of its coordinates are zero; to get to point $3 = (200, 100)$, you start at the reference point and go 200 steps right and 100 up; and so on.

▸ **EXERCISE 2.1**
Which of the six example points is closest to the point $(60, 30)$?

▸ **EXERCISE 2.2**
True or false: All points that lie on a given horizontal straight line have the same x coordinate.

▸ **EXERCISE 2.3**
Explain where the point $(-5, 15)$ is located.

▸ **EXERCISE 2.4**
What are the coordinates of a point that lies exactly 60 units below point 6 in the diagram above? ("Below" means "down the page," not "under the page.")

In a typical application of METAFONT, you prepare a rough sketch of the shape you plan to define, on a piece of graph paper, and you label important points on that sketch with any convenient numbers. Then you write a METAFONT program that explains (i) the coordinates of those key points, and (ii) the lines or curves that are supposed to go between them.

METAFONT has its own internal graph paper, which forms a so-called raster or grid consisting of square "pixels." The output of METAFONT will specify that certain of the pixels are "black" and that the others are "white"; thus, the computer essentially converts shapes into binary patterns like the designs a person can make when doing needlepoint with two colors of yarn.

resolution
Cartesian
Descartes

Coordinates are lengths, but we haven't discussed yet what the units of length actually are. It's important to choose convenient units, and METAFONT's coordinates are given in units of pixels. The little squares illustrated on the previous page, which correspond to differences of 10 units in an x coordinate or a y coordinate, therefore represent 10×10 arrays of pixels, and the rectangle enclosed by our six example points contains 20,000 pixels altogether.*

Coordinates don't have to be whole numbers. You can refer, for example, to point $(31.5, 42.5)$, which lies smack in the middle of the pixel whose corners are at $(31, 42)$, $(31, 43)$, $(32, 42)$, and $(32, 43)$. The computer works internally with coordinates that are integer multiples of $\frac{1}{65536} \approx 0.00002$ of the width of a pixel, so it is capable of making very fine distinctions. But METAFONT will never make a pixel half black; it's all or nothing, as far as the output is concerned.

The fineness of a grid is usually called its *resolution*, and resolution is usually expressed in pixel units per inch (in America) or pixel units per millimeter (elsewhere). For example, the type you are now reading was prepared by METAFONT with a resolution of slightly more than 700 pixels to the inch, but with slightly fewer than 30 pixels per mm. For the time being we shall assume that the pixels are so tiny that the operation of rounding to whole pixels is unimportant; later we will consider the important questions that arise when METAFONT is producing low-resolution output.

It's usually desirable to write METAFONT programs that can manufacture fonts at many different resolutions, so that a variety of low-resolution printing devices will be able to make proofs that are compatible with a variety of high-resolution devices. Therefore the key points in METAFONT programs are rarely specified in terms of pure numbers like '100'; we generally make the coordinates relative to some other resolution-dependent quantity, so that changes will be easy to make. For example, it would have been better to use a definition something like the following, for the six points considered earlier:

$$(x_1, y_1) = (0, b); \qquad (x_2, y_2) = (a, b); \qquad (x_3, y_3) = (2a, b);$$
$$(x_4, y_4) = (0, 0); \qquad (x_5, y_5) = (a, 0); \qquad (x_6, y_6) = (2a, 0);$$

then the quantities a and b can be defined in some way appropriate to the desired resolution. We had $a = b = 100$ in our previous example, but such constant values leave us with little or no flexibility.

Notice the quantity '$2a$' in the definitions of x_3 and x_6; METAFONT understands enough algebra to know that this means twice the value of a, whatever a is. We observed in Chapter 1 that simple uses of algebra give METAFONT its meta-ness. Indeed, it is interesting to note from a historical standpoint that Cartesian coordinates are named after René Descartes, not because he invented the idea of coordinates, but because he showed how to get much more out of

* We sometimes use the term "pixel" to mean a square picture element, but sometimes we use it to signify a one-dimensional unit of length. A square pixel is one pixel-unit wide and one pixel-unit tall.

..
z convention
draw
Union Jack
hex symbol

that idea by applying algebraic methods. People had long since been using coordinates for such things as latitudes and longitudes, but Descartes observed that by putting unknown quantities into the coordinates it became possible to describe infinite sets of related points, and to deduce properties of curves that were extremely difficult to work out using geometrical methods alone.

So far we have specified some points, but we haven't actually done anything with them. Let's suppose that we want to draw a straight line from point 1 to point 6, obtaining

---

(Figure 2b will be inserted here; too bad you can't see it now.)

One way to do this with METAFONT is to say

draw (x_1, y_1) .. (x_6, y_6).

The '..' here tells the computer to connect two points.

It turns out that we often want to write formulas like '(x_1, y_1)', so it will be possible to save lots of time if we have a special abbreviation for such things. Henceforth we shall use the notation z_1 to stand for (x_1, y_1); and in general, z_k with an arbitrary subscript will stand for the point (x_k, y_k). The '**draw**' command above can therefore be written more simply as

draw z_1 .. z_6.

Adding two more straight lines by saying, '**draw** z_2 .. z_5' and '**draw** z_3 .. z_4', we obtain a design that is slightly reminiscent of the Union Jack:

(Figure 2c will be inserted here; too bad you can't see it now.)

We shall call this a hex symbol, because it has six endpoints. Notice that the straight lines here have some thickness, and they are rounded at the ends as if they had been drawn with a felt-tip pen having a circular nib. METAFONT provides many ways to control the thicknesses of lines and to vary the terminal shapes, but we shall discuss such things in later chapters because our main concern right now is to learn about coordinates.

If the hex symbol is scaled down so that its height parameter b is exactly equal to the height of the letters in this paragraph, it looks like this: '✳'. Just