Typesetting Concrete Mathematics

Donald E. Knuth
Stanford University

During 1987 and 1988 I prepared a textbook entitled *Concrete Mathematics* [1], written with co-authors Ron Graham and Oren Patashnik. I tried my best to make the book mathematically interesting, but I also knew that it would be typographically interesting — because it would be the first major use of a new typeface by Hermann Zapf, commissioned by the American Mathematical Society. This typeface, called AMS Euler, had been carefully digitized and put into METAFONT form by Stanford's digital typography students [8]; but it had not yet been "tuned up" for real applications. My new book served as an ideal test case, because (1) it involved a great variety of mathematical formulas; (2) I was highly motivated to make the book readable and attractive; (3) my experiences with tuning up Computer Modern gave me insights into how to set the mysterious font parameters used by TEX in math mode; and (4) the book was in fact being dedicated to Leonhard Euler, the great mathematician after whom the typeface was named.

The underlying philosophy of Zapf's Euler design was to capture the flavor of mathematics as it might be written by a mathematician with excellent handwriting. For example, one of the earmarks of AMS Euler is its zero, '0', which is slightly pointed at the top because a handwritten zero rarely closes together smoothly when the curve returns to its starting point. A handwritten rather than mechanical style is appropriate for mathematics because people generally create math with pen, pencil, or chalk. The Euler letters are upright, not italic, so that there is a general consistency with mathematical symbols like plus signs and parentheses, and so that built-up formulas fit together comfortably. AMS Euler includes seven alphabets: Uppercase Roman (ABC through XYZ), lowercase Roman (abc through xyz), uppercase Greek (ABΓ through XΨΩ), lowercase Greek ($\alpha\beta\gamma$ through $\chi\psi\omega$), uppercase Fraktur ($\mathfrak{ABC}$ through $\mathfrak{XY3}$), lowercase Fraktur ($\mathfrak{abc}$ through $\mathfrak{xy3}$), and uppercase Script ($\mathcal{ABC}$ through $\mathcal{XYZ}$). It also includes two sets of digits (0123456789 and $o123456789$), as well as special characters like $\wp$, $\aleph$, and some punctuation marks. Details about its design are discussed in another article [7].

To refine the digitized characters for mathematical use, I began by correcting the way they appeared in their "boxes," from TEX's viewpoint. For this purpose I used the \math tests of the standard testfont routine [5, Appendix H]; these tests put the characters through their basic paces by typesetting formulas such as $|A|+|B|+|C|+\cdots+|Z|$, $a^2+b^2+c^2+\cdots+z^2$, $a_2+b_2+c_2+\cdots+z_2$, and $\widehat{A}+\widehat{B}+\widehat{C}+\cdots+\widehat{Z}$. I noticed among other things that the Fraktur characters had all been placed too high above the baseline, and that more blank space was needed at the left and right of the characters in subscript/superscript sizes. After fixing such problems I also needed to set the italic corrections so that subscripts and superscripts would have proper offsets; and I needed to define suitable kerns with a \skewchar so that accents would appear visually centered. AMS Euler contains more than 400 characters, and Hermann had made them beautiful; my job was to find the right adjustments to the spacing so that they would fit properly into mathematics.

The next step was to design a set of TEX macros so that AMS Euler could be used conveniently in math mode. This meant adding new "families" to the conventions I had defined in previous formats. Plain TEX typesets mathematics with four basic font families (namely, \fam0 for text, \fam1 for math italic, \fam2 for symbols, and \fam3 for large delimiters), plus a few others that are used less frequently (\fam4 for text italic, \fam5 for slanted text, \fam6 for boldface roman, and \fam7 for typewriter style). I added \fam8 for AMS Euler Script and \fam9 for AMS Euler Fraktur; the AMS Euler Greek and Roman went into the old position of math italic, \fam1. Hermann had designed new parentheses and brackets, which were bundled together with the Fraktur fonts; therefore my macro file changed plain TEX's conventions by defining, e.g., \mathcode`(="4928 and \delcode`(="928300. Similarly, Euler has the symbol '$\leqslant$' as an alternative to '$\leq$', packaged with the Script alphabet; to make TEX recognize this substitution I said \mathchardef\leq="3814 \let\le=\leq.

With such additions to the plain TEX macros I could type formulas like \$\tan(x+y)\$ as usual and get not '$\tan(x+y)$' but 'tan(x+y)'. There was, however, one significant difference between typing the manuscripts for *Concrete Mathematics* and for *The Art of Computer Programming*, caused by the fact that the Euler numerals 0123456789 are distinctly different from the numerals 0123456789 in ordinary text. In previous work, I used to "optimize" my typing by saying, e.g.,

```
$x$ is either 1 or $-1$,
```

thereby omitting $'s around a mathematical constant unless I needed them to get a minus sign instead of a hyphen. After all, I reasoned, those extra $'s just make TEX work harder and the result looks the same; so why should I be logical? But in *Concrete Mathematics* I needed to type

```
$x$ is either $1$ or $-1$,
```

to keep x from being 'either 1 or −1'. The early drafts of my manuscript had been prepared in the old way; therefore I needed to spend several hours laboriously hunting down and correcting all instances where the new convention was necessary. This experience proved to be worthwhile, because it taught me that there is a useful and meaningful distinction between text numerals and mathematical numerals. Text numerals are used in contexts like '1776' and 'Chapter 5' and '41 ways', where the numbers are essentially part of the English language; mathematical numerals, by contrast, are used in contexts like 'the greatest common divisor of 12 and 18 is 6', where the numbers are part of the mathematics. (Authors of technical texts in languages like Japanese, where Arabic numerals are used in formulas but not in ordinary text, have always been well aware of this distinction; now I had a chance to learn it too.)

As I was tooling up to begin using AMS Euler, my publishers were simultaneously showing the preliminary manuscript of *Concrete Mathematics* to a book designer, Roy Howard Brown. I had sent Roy a copy of the first Euler report [8] so that he could see examples of the typeface we planned to use for mathematics. Our original intention, based on Zapf's original plans when he began the design in 1980, was to use Computer Modern Roman for the text and AMS Euler for the mathematics. But Roy noticed that AMS Euler was somewhat darker in color than a traditional mathematical italic, so he decided that the text face should be correspondingly heavier. He sent me several samples of typefaces with more suitable weights, so that I could prepare a special font compatible with AMS Euler. (One of my basic premises when I had developed the Computer Modern meta-font was that it should be readily adaptable to new situations like this.) When I saw Roy's samples, I decided to pursue something that I'd wanted an excuse to do for several years, namely to find settings for the parameters of Computer Modern that would produce an "Egyptian" (square-serif) style.

The cover designs for *Computers & Typesetting*, Volumes A–E, show a gradual transition of the respective letter pairs Aa, Bb, Cc, Dd, and Ee from the style of standard Computer Modern Roman to an Egyptian style. I had made these cover designs just for fun, at the suggestion of Marshall Hendricks, but I had never had time to experiment with a complete text face in that style. Now I had a good reason to indulge that whim, and after a pleasant afternoon of experiments I found a combination of parameters that looked reasonably attractive, at least when I examined samples produced by our laser printer. (I magnified the fonts and viewed them from a distance, to overcome the effects of 300 dpi resolution.) Then I made more elaborate samples of text and printed them on Stanford's APS phototypesetter, to see if the new fonts would really pass muster. Some characters needed to be adjusted (e.g., the 'w' was too dark), but I was happy with the result and so was Roy.

I decided to call the resulting font Concrete Roman, because of its general solid appearance and because it was first used in the book *Concrete Mathematics*. (In case you haven't guessed, the text you are now reading is set in Concrete Roman.) *There also is Concrete Italic, a companion face that is used for emphasis in the book.* EVEN STRONGER EMPHASIS IS OCCASIONALLY ACHIEVED BY USING A CONCRETE ROMAN CAPS AND SMALL CAPS FONT. Anybody who has the METAFONT sources for Computer Modern can make the Concrete fonts rather easily by preparing parameter files such as `ccr10.mf`, analogous to `cmr10.mf`; you just need to change certain parameter values as shown in the accompanying table.

Here are samples of Concrete Roman in the 9 pt, 8 pt, 7 pt, 6 pt, and 5 pt sizes:

Mathematics books and journals do not look as beautiful as they used to. It is not that their mathematical content is unsatisfactory, rather that the old and well-developed traditions of typesetting have become too expensive. Fortunately, it now appears that mathematics itself can be used to solve this problem.

A first step in the solution is to devise a method for unambiguously specifying mathematical manuscripts in such a way that they can easily be manipulated by machines. Such languages, when properly designed, can be learned quickly by authors and their typists, yet manuscripts in this form will lead directly to high quality plates for the printer with little or no human intervention.

A second step in the solution makes use of classical mathematics to design the shapes of the letters and symbols themselves. It is possible to give a rigorous definition of the exact shape of the letter "a", for example, in such a way that infinitely many styles (bold, extended, sans-serif, italic, etc.) are obtained from a single definition by changing only a few parameters. When the same is done for the

> other letters and symbols, we obtain a mathematical definition of type fonts, a definition that can be used on all machines both now and in the future. The main significance of this approach is that new symbols can readily be added in such a way that they are automatically consistent with the old ones.
>
> Of course it is necessary that the mathematically-defined letters be beautiful according to traditional notions of æsthetics. Given a sequence of points in the plane, what is the most pleasing curve that connects them? This question leads to interesting mathematics, and one solution based on a novel family of spline curves has produced excellent [sic] fonts of type in the author's preliminary experiments.
>
> We may conclude that a mathematical approach to the design of alphabets does not eliminate the artists who have been doing the job for so many years; on the contrary, it gives them an exciting new medium to work with. [2]

Heavier weight makes the type more resilient to xeroxing and easier to read in a poorly lighted library, so these new typefaces may help solve some of the legibility problems we all know too well. But a typeface that is too bold can also make a book tiresome to read. To avoid this problem, Roy decided to use a `\baselineskip` of 13 pt with 10 pt type. This gives an additional advantage for mathematical work, because it prevents formulas like '$\sum_{0\leq k<n} a_k^d$' in the body of the text from interfering with each other; the normal 12 pt baselineskip used in most mathematics books can get uncomfortably tight. Of course, the increased space between lines also increases the number of pages by about 8%; this seems a reasonable price to pay for increased readability.

Table of parameter values for Concrete fonts, using the conventions found on pages 12–31 of [6]:

name	cmr10	ccr10	ccr9	ccr8	ccr7	ccr6	ccr5	ccslc9	ccti10	ccmi10	cccsc10	*lower*
font_ident	CMR	CCR	CCR	CCR	CCR	CCR	CCR	CCSLC	CCTI	CCMI	CCCSC	
serif_fit	0	1	1	1	1	1	1	0	1	1	1	
cap_serif_fit	5	3	2.8	2.6	2.4	2.2	2	2	3	3	3	2
x_height	155	165	148.5	132	115.5	99	82.5	155	165	165	155	116
bar_height	87	92	78.3	69.6	60.9	52.2	43.5	85	92	92	87	65
tiny	8	11	10	9	8	7	6	9	11	11	11	
fine	7	6	6	6	6	6	5	6	6	6	6	
thin_join	7	17	17	15	13	12	11	13	17	17	17	
hair	9	21	20	19	17	15	14	16	21	21	21	
stem	25	25	24	22	20	18	16	22	24	25	25	23
curve	30	27	26	24	21.5	19	17	23	26	27	27	
ess	27	25	24	22	20	17	12	25	24	25	25	
flare	33	29	26	24	23	20	18	28	28	29	29	22
cap_hair	11	21	20	19	17	15	14	16	21	21	21	21
cap_stem	32	27	26	24	21.5	19	17	23	26	27	27	24
cap_curve	37	28	27	25	22.5	20	18	24	27	28	28	26
cap_ess	35	27	24	22	21.5	19	14	23	26	27	27	24
bracket	20	5	5	4	4	3	3	5	5	5	5	
jut	28	30	27	24	21	19	17	15	30	30	30	
cap_jut	37	32	29	26	23	20	18	16	32	32	32	24
vair	8	21	20	19	17	15	14	15	21	21	21	
notch_cut	10	5/6	3/4	2/3	7/12	1/2	5/12	3/4	5/6	5/6	5/6	
bar, etc.*	11	21	20	19	17	15	14	15	21	21	21	21
cap_notch_cut	10	1	.9	.8	.7	.6	.5	.9	1	1	1	3/4
serif_drop	4	5	3.6	3.2	2.8	2.4	2	3.6	5	5	5	
o	8	4	4	3	3	3	3	4	4	4	4	3
apex_o	8	3	3	3	3	3	2	3	3	3	3	3
other values from		cmr10	cmr9	cmr8	cmr7	cmr6	cmr5	cmsl9	cmti10	cmmi10	cmcsc10	

*The measurements for *bar* apply also to *slab*, *cap_bar*, and *cap_band*. All of the Concrete fonts have *dish* = 0, *fudge* = .95, *superness* = 8/11, *superpull* = 1/15, and *beak_darkness* = 11/30. Parameters not mentioned here are inherited from the corresponding cm fonts, except that cccsc10 has *lower.fudge* = .93; ccti10 has the *u* value 20 not 18.4, and the *crisp* value 11 not 8; ccmi10 has the *crisp* value 0 not 8.

Is the extra weight of Concrete Roman really necessary for compatibility with AMS Euler? Here is a small sample that uses ordinary cmr10 as the text font, so that readers can judge this question for themselves:

> The set S is, by definition, all points that can be written as $\sum_{k\geq 1} a_k(i-1)^k$, for an infinite sequence $a_1, a_2, a_3, \ldots$ of zeros and ones. Figure 1 shows that S can be decomposed into 256 pieces congruent to $\frac{1}{16}S$; notice that if the diagram is rotated counterclockwise by 135°, we obtain two adjacent sets congruent to $(1/\sqrt{2})S$, since $(i-1)S = S \cup (S+1)$. [3]

And now let's replay the same text again, using Concrete Roman and \baselineskip=13pt:

> The set S is, by definition, all points that can be written as $\sum_{k\geq 1} a_k(i-1)^k$, for an infinite sequence $a_1, a_2, a_3, \ldots$ of zeros and ones. Figure 1 shows that S can be decomposed into 256 pieces congruent to $\frac{1}{16}S$; notice that if the diagram is rotated counterclockwise by 135°, we obtain two adjacent sets congruent to $(1/\sqrt{2})S$, since $(i-1)S = S \cup (S+1)$. [3]

Equation numbers presented us with one of the most perplexing design questions. Should those numbers be typeset in Euler or cast in Concrete? After several experiments we hit on a solution that must be right, because it seems so obvious in retrospect: We decided to set equation numbers in an "oldstyle" variant of Concrete Roman, using the digits '0123456789'. The result—e.g., '(3.14)'—was surprisingly effective.

After I had been using AMS Euler for several months and was totally conditioned to "upright mathematics," I began to work on a chapter of the book where integral signs appear frequently. It suddenly struck me that the traditional integral sign is visually incompatible with AMS Euler, because it slopes like an italic letter. Such a slope was now quite out of character with the rest of the formulas. So I designed a new, upright integral sign to match the spirit of the new fonts. Then I could typeset

$$\int_a^b f(x)\,dx \;-\; \frac{1}{2\pi i}\oint_{|z|=r} \frac{g(z)\,dz}{z^n}$$

and $\int_{-\infty}^{\infty} \cos x^2\,dx$, instead of

$$\int_a^b f(x)\,dx \;-\; \frac{1}{2\pi i}\oint_{|z|=r} \frac{g(z)\,dz}{z^n}$$

and $\int_{-\infty}^{\infty} \cos x^2\,dx$. The new integral signs went into a new font called euex10, which became \fam10 in math mode; I told TEX to get integral signs from the new font by simply saying

```
\mathchardef\intop="1A52
\mathchardef\ointop="1A48
```

in my macro file. Later I noticed that the infinity sign '∞' of Computer Modern was too light to be a good match for the Euler alphabets, so I created a darker version '∞' and put it into euex10 with the new integral signs.

Hermann Zapf was helping to advise me during all this time. For example, he approved a draft of Chapter 1 that had been phototypeset in Concrete Roman and AMS Euler, while I was tuning things up. Later, when he received a copy of the first printing of the actual book, he saw Chapter 2 and the other chapters for the first time; and this led him to suggest several improvements that he could not have anticipated from Chapter 1.

Chapter 2 is about summation, and I had used the sign $\sum$ from Computer Modern's cmex10 font, together with its displaystyle counterpart

$$\sum_{k=0}^{n} f(k)\,,$$

to typeset hundreds of formulas that involve summation. Hermann pointed out that the capital \Sigma of Euler looks quite different—it is 'Σ', without beaks—so he suggested changing my summation signs to look more like the Σ of Euler. I did this in the second printing of the book, using $\sum$ in text formulas and

$$\sum_{k=0}^{n} f(k)$$

in displays. Hermann also asked me to make the product symbols less narrow, more like Euler's 'Π'; so I changed them

from $\prod$ and $\displaystyle\prod$ to $\prod$ and $\displaystyle\prod$.

Moreover, he wanted the arrows to have longer and darker arrowheads: '$\rightarrow$', not '$\rightarrow$'. And he wanted curly braces to be lighter, so that

$$\left\{\begin{matrix} a \\ b \\ c \\ d \end{matrix}\right\} \quad \text{would become} \quad \left\{\begin{matrix} a \\ b \\ c \\ d \end{matrix}\right\}.$$

All of these new characters were easy to design, using the conventions of Computer Modern [6], so I added them to euex10 and used them in the second printing.

Readers of *Concrete Mathematics* will immediately notice one novel feature: There are "graffiti" printed in the margins. My co-authors and I asked students who were testing the preliminary book drafts to write informal comments that might be printed with the text, thereby giving the book a friendly-contemporary-lifelike flavor. We weren't sure how such "remarks from the peanut gallery" should be typeset, but we knew that we did want to include them; in fact, we collected almost 500 marginal notes. Roy hit on the idea of putting them in the *inner* (gutter) margin, where they would not have too much prominence. He also sent a sample of a suitably informal typeface, on which I modeled "Concrete Roman Slanted Condensed" type.

This 9 pt typeface proved to work very nicely for marginal graffiti, where it is typeset ragged right, 6 picas wide, with 10 pt between baselines.

To typeset such graffiti, I introduced a \g macro into my TEX format file, so that it was possible to type simply '\g Text of a graffito.\g' on whatever line of text I wanted the marginal comment to begin. I did a bit of positioning by hand to ensure that no two comments would overlap; but my \g macro did most of the work. For example, it automatically decided whether to put graffiti into the left margin or the right margin, based on an auxiliary 'grf' file that recorded the choice that would have been appropriate on the previous TEX run.

My macros for *Concrete Mathematics* cause TEX to produce not only the usual dvi file and log file corresponding to the input, but also the grf file just mentioned and four other auxiliary files:

- The ans file contains the text of any answers to exercises that appeared in the material just typeset; such answers will be \input at an appropriate later time. (Page 422 of *The TEXbook* discusses a similar idea. The only difference between *Concrete Mathematics* and *The TEXbook* in this regard was that I used one file per chapter in *Concrete Mathematics*, while *The TEXbook* was typeset from one long file.)
- The inx file contains raw material for preparing the index. After everything but the index was ready, I put all the inx files together, sorted them, and edited the results by hand. (See pages 423–425 of *The TEXbook*, where I describe similar index macros and explain why I don't believe in fully automatic index preparation.)
- There's also a ref file, which contains the symbolic names of equations, tables, and exercises that may be needed for cross references. (A ref file is analogous to an aux file in LATEX.)
- Finally, a bnx file records the page numbers of each bibliographic reference, so that I can include such information as a sort of index to the bibliography. That index was done automatically.

I wouldn't want to deal with so many auxiliary files if I were producing a simpler book or a system for more general consumption. But for the one-shot purposes of *Concrete Mathematics*, this multiple-file approach was most convenient.

We decided to use a nonstandard numbering system for tables, based on the way superhighway exits are numbered in some states: Table 244, for example, refers to the table on page 244. (The idea wasn't original with us, but I don't remember who suggested it.) Macros to accommodate this convention, and to update the cross-references when the page numbers change, were not difficult to devise.

All of the macros I wrote for *Concrete Mathematics* appear in an 814-line file called gkpmac.tex, which (I'm sorry to admit) includes very little documentation because it was intended only to do a single job. Macro writers may like to look at this file as a source of ideas, so I've made it publicly accessible; for example, it is <tex.doc>gkpmac.tex at score.stanford.edu on the Arpanet. But people who attempt to use these macros should be aware that I have not pretended to make them complete or extremely general. For example, I implemented a subset of LATEX's picture environment, and used it to prepare all but one of the illustrations in the book; but I didn't include everything that LATEX makes available. Moreover, I didn't need boldface type in the mathematical formulas of *Concrete Mathematics* (except for the Q on page 143 of the second printing); so I didn't include macros for accessing any of the bold fonts of AMS Euler in math mode. In this respect, the book was not a perfect test, because almost half of the Euler characters are boldface and therefore still untried. Macros for bold mathematics would be easy to add, following the pattern already established in gkpmac; but I must leave such tasks to others, as I return to my long-delayed project of writing the remaining volumes of *The Art of Computer Programming*.

References

[1] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, *Concrete Mathematics.* Addison-Wesley, 1989.

[2] Donald E. Knuth, "Mathematical typography," *Bulletin of the American Mathematical Society* (new series) 1 (1979), 337–372.

[3] Donald E. Knuth, *Seminumerical Algorithms*, second edition. Addison-Wesley, 1981.

[4] Donald E. Knuth, *The TEXbook*, Volume A of *Computers & Typesetting.* Addison-Wesley, 1984 and 1986.

[5] Donald E. Knuth, *The METAFONTbook*, Volume C of *Computers & Typesetting.* Addison-Wesley, 1986.

[6] Donald E. Knuth, *Computer Modern Typefaces*, Volume E of *Computers & Typesetting.* Addison-Wesley, 1986.

[7] Donald E. Knuth and Hermann Zapf, "AMS Euler — A new typeface for mathematics," *Scholarly Publishing*, to appear.

[8] David R Siegel, *The Euler Project at Stanford.* Computer Science Department, Stanford University, 1985.

Outline fonts with METAFONT

Doug Henderson

Lately I've been feeling like no one else out there is having any fun with METAFONT. I sure am. Recently, I came across a small gem in the rough in the METAFONTbook and decided to polish it up some. In chapter thirteen there is an interesting exercise (13.23, page 121) which calls for the user to replace a character by its "outline". Since it was an idea that appealed to me (double dangerous bend kind of fun) I set about applying the solution in various places in the METAFONT source code to see the affects. What I finally settled on was hooking it in as a definition extension of the `endchar` macro. I reasoned that for each Computer Modern character METAFONT produces in a font, there exists both a `beginchar` statement, with which you relate things like the height, width and depth, and an `endchar` statement, which tells the program, among other things, to create a grid box for proof characters and to ship the character off to the Generic Format file (GF file). So, the `endchar` definition seemed like a good spot to tell METAFONT to take whatever character image had been created and convert it to an outline, since it is called once per character, right before it's shipped out.

Here is the macro definition I used to create outline fonts with:

```
message"Loading the font outline macros.";
boolean outlining;
% only outline when told to
outlining:=false;

def outline =
 if outlining:
  cull currentpicture keeping (1,infinity);
  picture v; v:=currentpicture;
  cull currentpicture keeping (1,1)
   withweight 3;
  addto currentpicture also v - v
   shifted right -v shifted left - v
   shifted up - v shifted down;
  cull currentpicture keeping (1,4);
  if (pixels_per_inch >= 600) :
    addto currentpicture also currentpicture
     shifted left;
    addto currentpicture also currentpicture
     shifted up;
  fi
  showit;
 fi
enddef;

extra_endchar:=extra_endchar & "outline";
```

The first statement declares a `boolean` variable named `outlining`. The next line initializes `outlining` to be `false`, so we don't create outline fonts by default. The definition of `outline` includes the line `if outlining:` which tests to see whether the outlining feature is desired. If so, the macro proceeds to punch out the pixels on the inside of our character (leaving more than one pixel for the outline if using a high resolution printer or phototypesetter) and show the results, and, if not, ends the `if` statement with the `fi` statement. The last statement is interesting since it shows a nifty way to tack on new features when creating your characters. Instead of redefining the definition of the `endchar` macro with your special effects (in this case a character outline), just add to the definition of `endchar` with the `extra_endchar` statement. Some similar "hooks" exist for the `beginchar` and