Prehistory of digital fonts

Jacques André

Abstract

Over the second half of the 20th century, typography moved from physical metal type to the abstractions of digital computing. This revolution did not follow a straight path. We examine here some of the very first attempts to produce printed characters on computers.

In the 1950s, to satisfy the needs of physicists, the first vectorized letters (and numbers, signs, ...) were made on CRT screens and plotters. In the 1960s, the dot matrix concept allowed consideration of characters as surfaces, leading to digital phototypesetting. In the 1970s, thanks to research in computer-aided design, the way was opened to the fundamentals of digital letter outlines. The first font formats occurred in the late 1970s. The innovation of laser printers, around 1985, marked the beginning of the mature rendering of digital fonts, and the beginning of the commercial font wars, where we will leave off.

1 Introduction

Some people think that digital outline fonts were invented by Adobe, others say that they occurred first with phototypesetting, while still others say ...

One reason for this misunderstanding of history is that there have been no detailed and technical historical overviews of this subject[1] (we hope this paper could be a first attempt).

Another, more important, reason is that this story did not follow a straight path, but rather formed a set of rays converging towards the same outcome. At the beginning of the 20th century, book printing was done by experts, both for commercial presses and for institutional documents. In parallel, handwriting became less and less used, while the typewriter industry grew.

During the second world war, a need for a new kind of writing arose: Scientists needed to manipulate drawings and annotate them with letters on brand-new media, such as radar screens. So it was engineers who drew letters as if they were mathematical figures (Bézier, De Casteljau and Karow, for example, were mathematicians or physicists working in the industrial field, and were pioneers in this area, as we'll see). Some scientific developers contacted prominent typographers, for example Higonnet and Moyroud (at Lumitype) worked with Frutiger, while Karow and Knuth worked closely with Zapf. These first research concepts won over the manufacturers, who thus created a new, popular mass market for fonts. I personally think that the success of digital fonts comes from this intimate collaboration of artists and scientists, although it hasn't always been easy!

In this article, we will try to show, without claiming to be exhaustive, many various inventions, even if some turned out to be dead ends. But let us be clear, we do not tell the story of digital typeface designs (even if we happen to cite them), but rather a history of the technological inventions of digital fonts, and the tools for manipulating them by computer. Along the same lines, let's say that this is a story of digital fonts, not of text processing (even if TEX users know that both are related, like METAFONT and TEX).

For lack of space, and also to avoid making this a story of computer science, we have forbidden ourselves to go into many technical details. They can be found notably in the books by Haralambous [57] and Rubinstein [111].

Figure 1 shows the main tools or concepts studied here; it also shows the complexity of this story. We will therefore follow a chronological approach, with interludes to bring together some comparable developments.

2 First computerized characters: Line segments

Long before computer data processing, office operations were performed with equipment such as tabulators and printers that used impact technology as typewriters do. Since 1930 two companies were leaders in this area: IBM and (in Europe) Bull. By

Copyright 2023 Adverbum. This article is a translation of *Histoire de l'écriture Typographique – le XXième siècle, tome II/II – de 1950 à 2000 – Chapitre 5 : Histoire technique des fontes numériques* © 2016 – Adverbum pour les éditions Atelier Perrousseaux – France.

Translated and published with permission of the author and publisher. Translation by Patrick Bideault, with assistance from the author and Charles Bigelow.

Editor's note: Preparing this original book chapter for publication required extensive efforts. We profoundly thank the author for undertaking the project at all, after his writing of the original monumental volumes in French [5, 6] (for more on this series of books, see `https://tug.org/books/#andre`), and Patrick Bideault for the translation into English. We also thank Charles Bigelow for the initial suggestion, and his invaluable advice and assistance along the way. Christina Thiele made useful initial translations to get the project off the ground. Thanks, everyone.

[1] In addition to the many specific studies which we will cite below, let us mention some papers such as Knuth's TEX history [79], a master's thesis by C. Knoth [77], and the historical introductions of books on digital fonts like Haralambous's [57] and Southall's [117]. Two important studies (although on a less general topic) have appeared since the French version of this paper was published: Romano's study of desktop publishing [110] and Bigelow's study on the Font Wars [27].

doi.org/10.47397/tb/44-1/tb136andre-prehistory

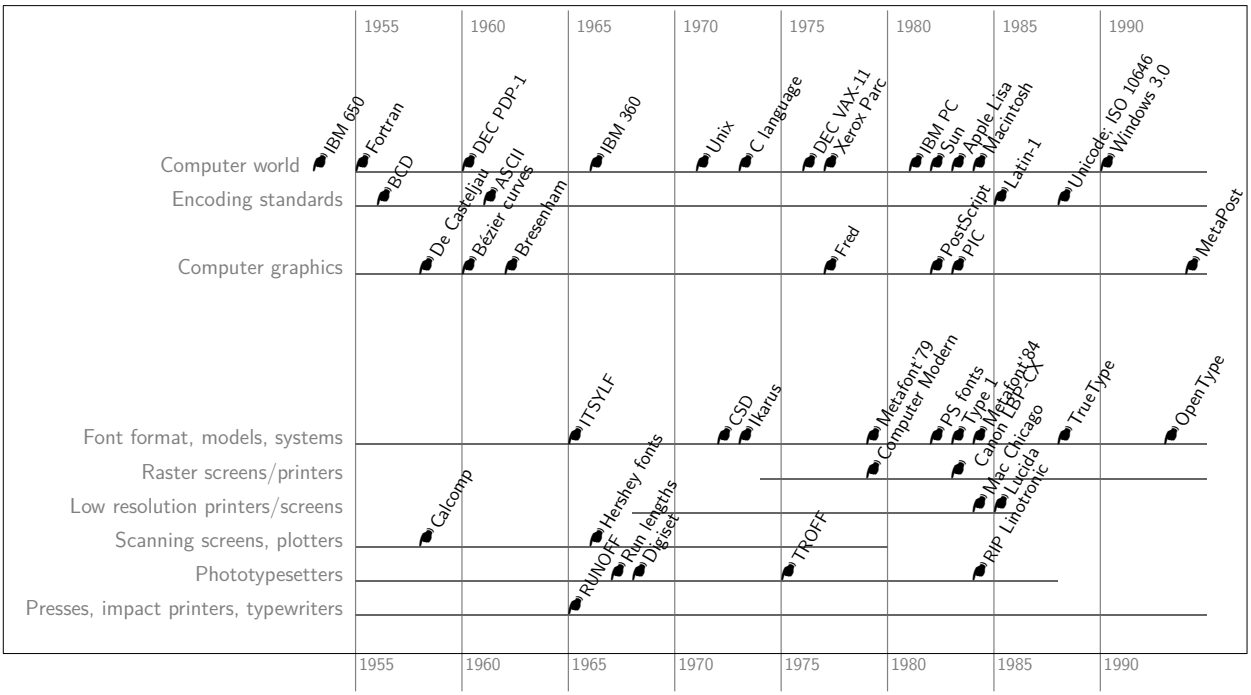


Figure 1: Chronology of the concepts and products that led to the birth of digital fonts during the years 1955–1990.

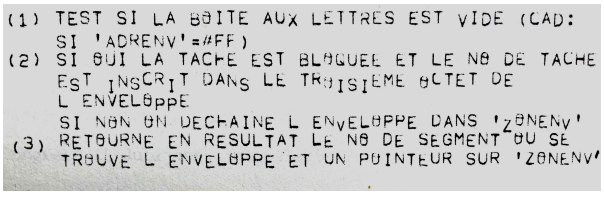


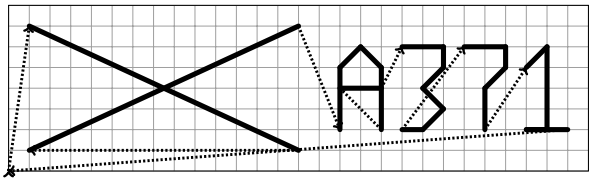
Figure 2: Computer output printed by impact devices were not always of high quality... Here, comments in a program [23], 1975.

1945, the first computer outputs were made with such equipment; impact devices remained in use up to around 1995 (a few even later) though today, their printer output may make us smile (figure 2).

Shortly thereafter, around 1950, cathode ray screens and then plotter devices allowed *drawing* of graphics and letters.

2.1 CRTs and plotters

Invented near the end of the 19th century, cathode ray tubes saw their first applications (oscilloscope, radar, television) in the first half of the 20th century. But it was not until 1946 that they were equipped with a binary memory that allowed drawings and then alphanumeric symbols to be drawn on them (figure 3). Early displays included EDSAC (1949), the IBM 740 CRT (1954), and others at Manchester University, MIT, and General Electric.



```
0  0  0  % start
0  1  7  % X upper left
1 14  1  % bottom right
0  1  1  % bottom left
1 14  7  % upper right
0 16  2  % A bottom left
...
1 27  2  % 1 bottom right
0  0  0  % return and loop
```

Figure 3: Cathode ray screen with XY scanning, and its control program. A spotlight runs along the screen, following the line segment connecting two consecutive points whose coordinates are given. This spot can be lit (thick lines) or switched off (dotted lines). The path, kept in memory, is in a loop which allows the screen to be refreshed (i.e. redisplayed).

It was on these that the first research was done for the basis of what is now called CAD or Computer Aided Design. To control such a screen, it suffices to have a sequence of triplets of the form (e, x, y),

where e is 0 or 1, indicating if the spot is lit, and (x, y) the coordinates of the next point. It is these triplets that we will later find in the run lengths of photocomposition.

Plotters

During the same period, a little after 1950, plotters first appeared, using the same principle of XY plotting as CRT screens. The CalComp 565 plotter, developed in 1958 in California, was the first widely marketed machine and in some ways the archetype of all these products. Other early plotters widely used at that time included the Olivetti XY 600 and the IFELEC 2025 S connected to an IBM 1130 computer.

The CalComp 565 plotter resembles the machine that Nicolas-Jacques Conté had invented in 1800 to engrave the plates of the *Description de l'Égypte* [4, p. 156] (see figure 4) but is electromechanically and computer-controlled. Its operation is analogous to that of the CRT, with the light spot replaced by a pencil that can be lifted or placed on a sheet of paper. This plotter, and all the others, were thus driven by commands sent by the computer according to machine codes specific to each. They operated with only three instructions, quite similar to those of the CRT XY scan (figure 3). To get away from the problem of machine dependency, higher-level languages, such as FORTRAN (notably the PLOT procedure), were soon used.

Plotters were first used in industrial drawing to draw maps for geography, charts for statistics, and so on. These jobs required additional commands, such as "draw a circle with center (x, y) and radius R". This was done by using routines that broke the curves into small line segments. In the years 1960–1980 much research took place on the approximation of curves by line segments (curves of degree one), then by curves of degree two, etc. This led to the creation of data-processing languages dedicated to the drawing of curves such as GPCP (*A General Purpose Contouring Program* of CalComp) then HPGL (*Hewlett Package Graphic Language*) which became ancestors of the Fred system at Xerox and from there to PostScript at Adobe (discussed below).

Figure 4: Two drawing machines. Top: Conté's manual etching machine, 1800 [Courtesy CNAM]; bottom: the Calcomp 565, the first electronic drawing machine, 1958 [Courtesy Wikipedia]. (These and following images are grayscaled for print in *TUGboat*.)

2.2 Drawing letters with lines

Figure 5 shows that the Calcomp had the ability to draw characters, essential in technical drawing for legends and markings of all kinds. Characters are treated as small drawings formed by a series of line segments (right-hand image). To the characters originally provided in the CalComp 565 (capitals, numerals, and "a few special characters") were gradually added the other characters of various six-bit

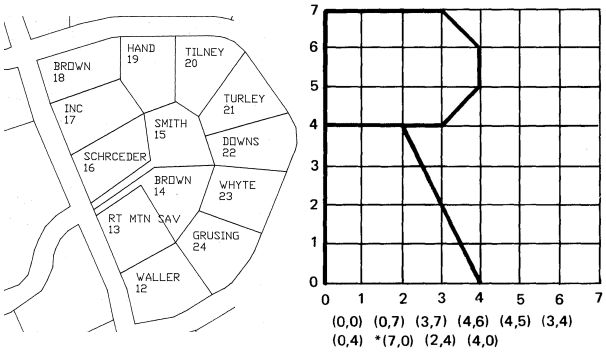


Figure 5: Left: extract of cadastral map drawn and written with a CalComp [Courtesy University of Denver Special Collections and Archives]; right: detail of the drawing of a letter R with line segments by a plotter. Extract from a Calcomp manual [41].

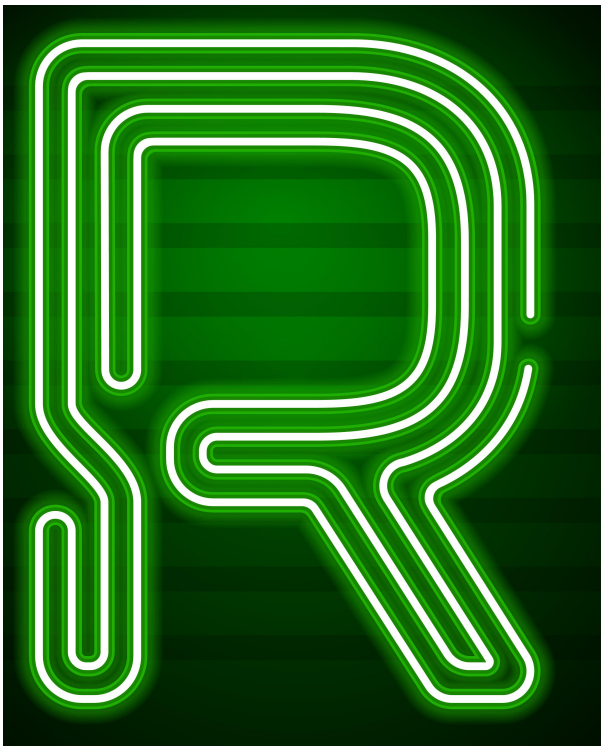


Figure 6: This linear neon tube fills the letter R like a Peano curve. [Courtesy Depositphotos]

binary-coded decimal (BCD) codes of the time, and then of seven-bit ASCII, then in its infancy. Eventually, given the extensive use of these symbols, CalComp "hardwired" the symbol plotting instructions, making them very fast.

Other plotters were soon created. One example is the Perthronic plotter from Aristo (Hamburg), which as early as 1960 was plotting numbers using so-called "stick digits", characters drawn with only straight lines. Today, plotters use standard vector fonts.

Filling characters with strokes. A figure defined by its outline can be filled in by hand with fairly tight strokes. Foundry catalogs from the 1930s show designs such as Prisma by Rudolf Koch (1931). As early as 1925, Fernando Jacopozzi displayed the letters "Citroën" (the famous French car maker) on the Eiffel Tower by electric bulbs that were aligned on wires (not a pixel array). This technique was used extensively for signs with neon tubes (now in the Las Vegas Neon Museum) and some letters could even be filled in with a single tube using Peano's curves (figure 6). At the end of the 1970s, METAFONT79 offers the concept of "double draw" for filling in between curves [80, chapter 6].

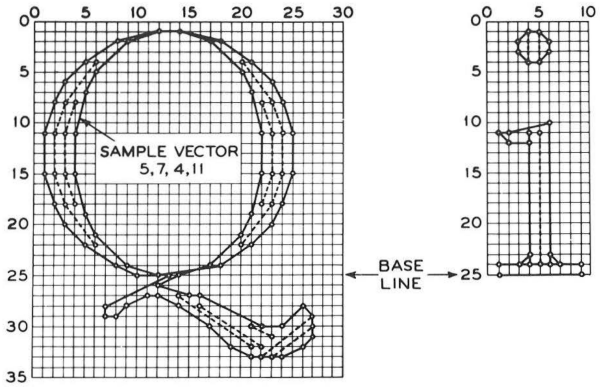


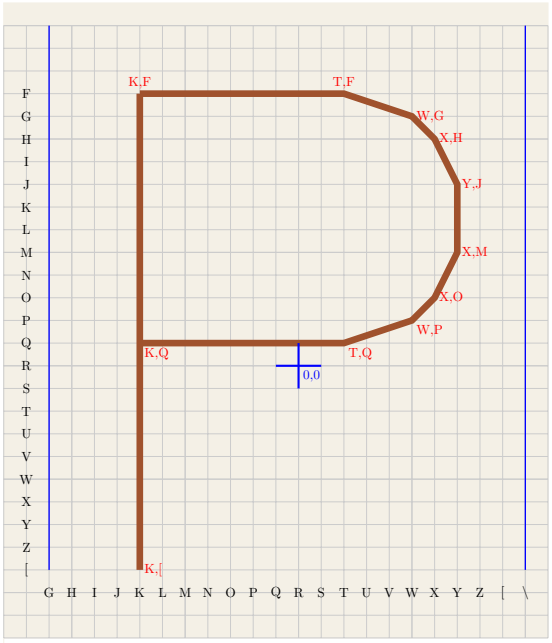
Figure 7: First attempts at filling letters with interior tracings: two letters, inspired by Baskerville, from the Bell system, 1967 [98]. [Courtesy Visible Language]

Bell characters. Under the direction of M. Mathews, a team at Bell Telephone (Murray Hill, USA) studied, shortly after 1965, a character production system for CRTs producing microfilms [98]. Character outlines were defined using line segments with a keyboard input system that allowed for the definition of several character sizes. Because the plotters' strokes were thin, the letters were blackened by drawing "inner outlines", a technique that would be seen again with Allen Hershey's typefaces. Figure 7 shows the principle.

Hershey typefaces. Around 1967, Allen Hershey developed a series of fonts at the Naval Weapons Laboratory (USA) that could initially be used with the Calcomp. Well documented — see [61] and [128] — and virtually copyright-free, they were widely distributed and used in the graphics world for years in their native form; they are still used in vector form today [38].

They were not written directly in the Calcomp language but in their own format, which made it easy to port them to other plotters. For Hershey, a font is a database, whose elements include (in a language called R-code) a glyph number (e.g. 516 for P), the number of points describing the design (14 for P), two "abscissae" to deduce the slopes and the width of the character, and finally the coordinates of the points of the line segments. Each coordinate was given by an alphanumeric sign according to the transliterated ASCII type encoding: `G` $= -11$, `H` $= -10, \ldots,$ `R` $= 0,$ `S` $= 1, \ldots,$ `[` $= 9,$ `\` $= 10$, and so on; recall that at that time available memory was very limited and one had to find tricks to save space.

In general, each character is defined in three modes: simplex (with a single stroke), duplex (two strokes) and triplex (three strokes) simulating three



516 14G\KFK[RKFTFWGXHYJYMXOWPTQKQ

Figure 8: Hershey's P pattern. Above: a detailed plot, with the R-code of this "P 516" below; the coordinates are indicated by the letters FGH. . .
Below: the three Hershey simplex, duplex and triplex P's have different weights, simulated by the presence of one, two or three lines. Drawings created after Hershey's tables [127].

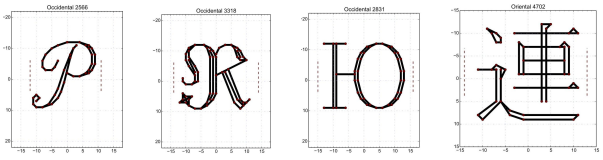


Figure 9: Examples of Hershey's characters: round, blackletter, Cyrillic, CJK — all drawn with straight lines. Based on [113]. [Courtesy Stewart Russel]

```
1 setlinewidth
1 setlinecap % rounded ends
0.5 4.5 moveto 4.25 4.5 lineto % top horizontal
4.5 4.25 moveto 4.5 0.75 lineto % right vertical
4.25 0.5 moveto 0.75 0.5 lineto % bottom hor.
0.5 0.75 moveto 0.5 1.5 lineto % left corner
1.75 0.75 moveto 2.75 2.75 lineto % diagonal
stroke
```

christian DELORME

Figure 10: Left: an 'a' from Delorme;
right: corresponding PostScript instructions [47];
below: Delorme's name in his font.

different weights. In addition to these weight variations, Hershey programmed a series of style variants (cursive letters, blackletter, etc.), and also non-Latin characters (including mathematical [128] and chemical characters, Cyrillic, and Japanese); see figure 9.

Microfilms. The first microfilm systems were equipped with a CRT that also produced text, such as the IBM 228, Alden, Benson, Control Data 280 systems, and others. A special mention to Stromberg-Carlson who, after a 64-character set for XY scanning, offered their 4600 Microfilm Recorder model with 112 characters, also using arcs, thin and thick strokes, thanks to a four-coordinate system in a 4096×3072 raster [105, p. 172]. These systems clearly influenced the third-generation photocomposers (page 28).

2.3 New line-based typefaces

These line-based typefaces have had little influence on digital fonts (except for the microfilm technique), but they have played a vital role in computer science, especially in CAD (Computer Aided Design), and they could not be ignored.

Either for fun, or to simulate old fonts dating back to the first plotters, digital stroked fonts can still be found nowadays, such as VECTOR BATTLE. Others are part of the typographic research of the 1980s.

The Delorme typeface. Christian Delorme designed a typeface composed of cardboard strips, rectangular and rounded at the ends, allowing for a much greater weight than that left by the tip of a pencil (figure 10). The connection of the segments of these thick straight lines gave the corners an illusion of roundness. It was digitized in a PostScript font format using the so-called `PaintType=3`, as used for the initial PostScript Courier (discussed below).

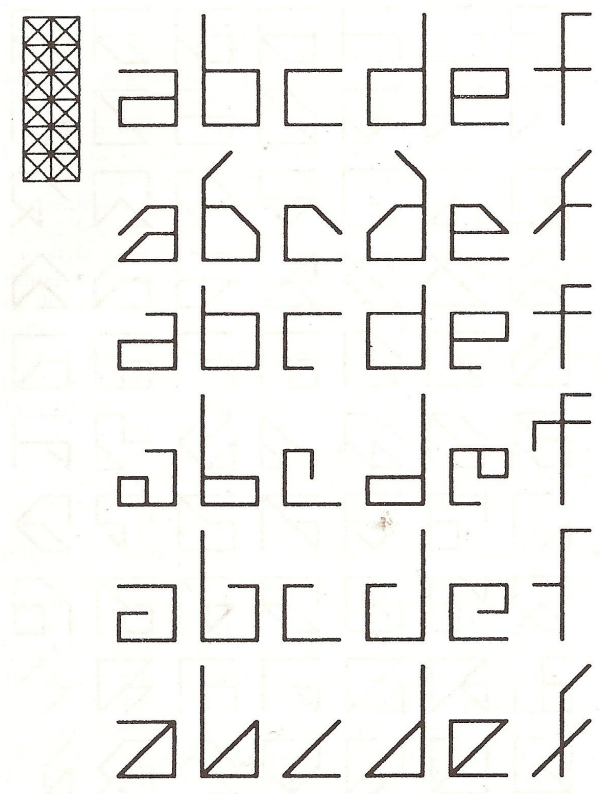


Figure 11: These alphabets composed in 1985 only of horizontal, vertical and diagonal lines were designed by computer, each line being letters "in the same spirit". Excerpt from Douglas Hofstadter, *Metamagical Themas* [65, figure 24-14]. [Courtesy Perseus Books]

Douglas Hofstadter's gridfonts. Douglas Hofstadter is a professor of cognitive science and computer science, with adjunct appointments in philosophy, comparative literature and other departments, at Indiana University in Bloomington, Indiana, USA. Most famous for his book *Gödel, Escher, Bach: An Eternal Golden Braid*, he is also known for his research on letterforms, including a long essay in response to Knuth's "The concept of a meta-font" [82], collected in his book *Metamagical Themas* [65, ch. 13].

In his work, Hofstadter asks himself the question of how to draw automatically (by computer, using artificial intelligence programs) as many 'a's as possible and then create the rest of the alphabets in such a way that all the letters of a single alphabet (which he calls gridfonts) share "the same spirit" (figure 11). His research is more philosophical (what is "the essence of 'A'-ness"?; what does "in the same spirit" mean?; etc.) than technical. But what we note here is that he uses characters composed only of strokes.

```
%FontType=1 PaintType=3 isFixedPitch=true
40 setlinewidth % bold => 80
0 setlinejoin
1 setlinecap
/A{/base currentlinewidth 2 div def
   120 545 moveto 325 545 lineto    % 1
   520 base lineto                  % 2
   280 545 moveto 80 base lineto    % 3
   30 base moveto 200 base lineto   % 4
   400 base moveto 575 base lineto  % 5
   165 210 moveto 440 210 lineto    % 6
   stroke } def %  A
```

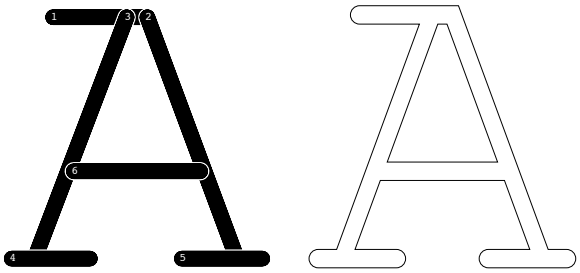


Figure 12: Adobe's Courier font in PostScript. Left: building the capital A with six stroked line segments (the PostScript instructions are shown), as used in the initial release of PostScript.
Right: in subsequent PostScript releases, Adobe used outlines for Courier, as with all other bundled fonts. (Excerpts from [12]).

Adobe's Courier, v1. To enter the CAD market, Adobe included the then-commonly used stroked fonts in its PostScript language. Fonts supported a so-called `PaintType=3` mode where only stroke instructions were used to draw the character, with the `fill` operation having no effect; the thickness of the strokes could be specified with the `linewidth` parameter.

The first version of Adobe's Courier [12], included in the initial release of PostScript, was defined using only thick strokes with rounded ends. In subsequent releases of PostScript, Courier, like all the other included fonts, was defined using outlines. The two are compared in figure 12, while figure 13 shows a clever use of a fixed thickness to simulate the variable thickness of the apostrophe.

3 Initial bitmap concepts

3.1 Screens, bitmaps and scanning

The first screens used XY scanning (page 22) but, with the cost of memory decreasing, since 1950 CRT screens with television scanning were in use. TV scanning consists of filling a matrix of points line by

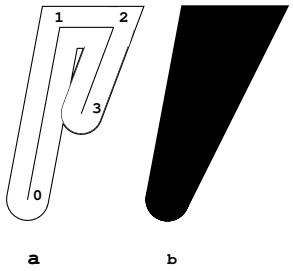


Figure 13: Adobe Courier v1 apostrophe construction (from [12]).

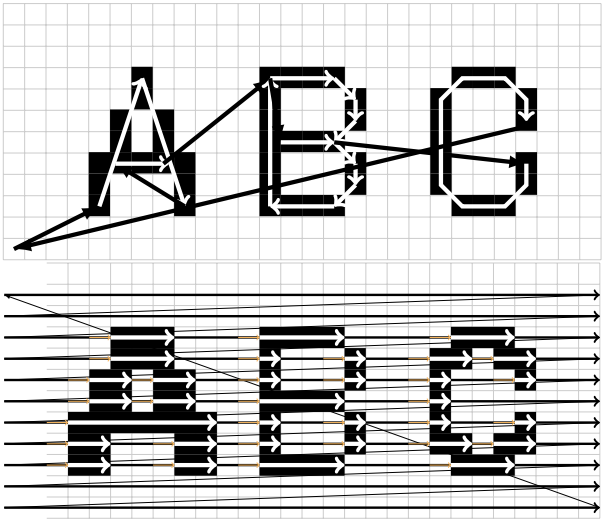


Figure 14: Two scanning methods for screens (television, computer, etc.): above, XY scanning (direct, by vectors); below, television scanning. Along white arrows, the beam blackens the pixels of the screen; with thin black lines, the beam writes nothing; raster returns are indicated by a thinner beam. The pixels are enormously magnified as large squares so as to show the scan.

line (figure 14): the usable surface of the screen is scanned from top to bottom, line by line, each one from left to right, with a step as small as possible. Some screens had a different scanning direction; for example, the Digiset scanned vertically (figure 18).

The "carriage return" of the beam to refresh the screen is called the "frame return", the image of the screen being assimilated to a frame.

3.2 Frame concept

Canvases existed long before computers, as fabrics appearing in the Western world, as early as the Neolithic period. These fabrics, when they are thick and not too tight, define a kind of grid, and are called canvas. Canvas fabric served as the base for needlepoint embroideries and tapestries: a thread

Figure 15: Above, excerpt from *Belle Prérie* by Le Bé, 1601 [coll. J.A.]; below, school exercise in cross-stitch embroidery, late 19th century [Credit Stefano Bianchetti/Les Éditions de l'Amateur].

of wool is passed through this grid, thus defining "points" corresponding to the pixels of our bitmaps. Cross-stitch embroidery began in the Middle Ages.

As early as 1600, Le Bé shows models of letters embroidered with a grid of 10×14 such "pixels". Figure 15 shows that there were already solutions to problems that we will see again with our computer bitmaps: the diagonal of the N is not linear (as in figure 21) and there are white squares at the junctions of the letters; these limitations are used for aesthetic purposes. Around 1750, the Encyclopedia of Diderot and D'Alembert shows very beautiful alphabets on a grid of only 7×7 pixels [50, Suppl. 3, pl. 4].

In the nineteenth century, with the introduction of compulsory schooling, the embroidered alphabet was substantively developed. The teaching of it was abandoned by 1930.

Woven books. It is well known that Joseph Marie Jacquard designed at the beginning of the 19th century the first mechanical loom using punch tapes (based on 18th century inventions) and so the first computer automaton (Charles Babbage was inspired by it to make his Analytical Engine [53]). Recent studies [27, 107, 126] pay attention to the fact that this machine was able not only to design graphics but also texts, considering letters as a special case of graphics (as PostScript and METAPOST would

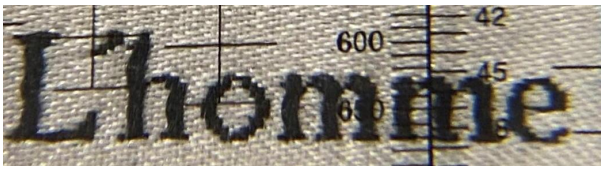


Figure 16: Detail of *Les Laboureurs* by Lamartine, Lyons, 1878; shown through a lens, scaled ≈ ×3.5. [Courtesy RIT Cary Graphic Arts Collection]

do a dozen decades later). In the city of Lyons (France), some manufacturers exhibited their skill by weaving books in silk on a Jacquard loom. Among these books, extremely rare today, let us mention *Les Laboureurs* by Lamartine, woven in 1878, and *Le Livre des Prières*, 1886.

The weft thread behaves, when it is over the warp thread, like a black rectangular pixel and when under, like a white pixel. The succession of over/under allows filling characters as in figures 18 and 54 below. The loom mechanism putting thread over or under the frame was governed by punched tapes, according to a bitmap, called "mise en carte". It is a paper with a grid of 1 cm square, each one divided in 10×10 pixels. It is not clear exactly how this bitmap was "programmed", to use a modern term. But it is conceivable that the letter images were reproduced from templates or pre-digitized models.

This Lamartine text (figure 16) has been composed in body size close to 8 pt. The jewel-like precision of the book type has a digital resolution comparable to laser printer resolutions of a century later. As Bigelow says [27], these books show the true first ancestors of digitized types.

Mosaics. Although the mosaics of the Greek, Roman, early Christian, etc., times often have textual inscriptions (figure 17), they are not true raster letters in the sense that there is no regular raster (neither for the background, nor for the letters). Rather, they are a construction with completely disordered pixels, without being a random raster.

3.3 Photocomposers

The photocomposers of the first two generations used characters photographed on film [6, ch. 1, Photocomposition]. Generally, these typesetters were driven by in-house tools. At the beginning of the 1970s, the Unix group at Bell labs got a Graphic Systems CAT phototypesetter. Joe Ossanna then wrote a version of nroff (a text formatter for typewriters or impact printers) that would drive it. A few years later (around 1975) Brian Kernighan adapted troff to C programming, to any kind of second generation typesetter [74], and even to mathematics (eqn language) [76].

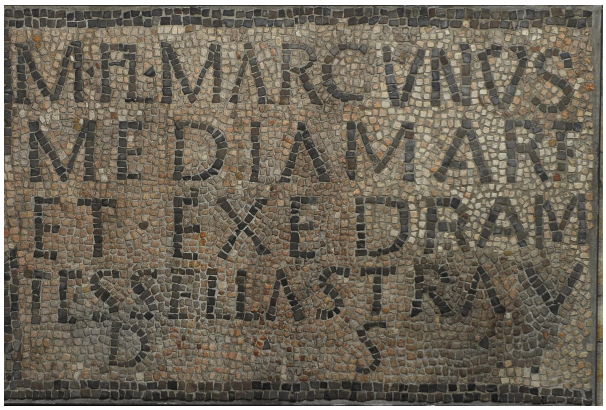


Figure 17: Early Christian inscription (CIL XIII 11479) in mosaic tesserae discovered in 1905 in Avenches/Aventicum, Switzerland. [Courtesy AVENTICVM]

The third generation of photocomposers marks the beginning of the use of digitized characters. The first digital photocomposer was created in Kiel (Germany) by Dr. Rudolf Hell [59, 117] whose company specialized in special equipment, photography and electronics.

Hell was inspired by the technique used for the first microfilm systems (page 25). The image of a typeface is projected onto the screen, with a television-type scan (figure 14, but in this case, a small technical difference, the scan is vertical and not horizontal), then exposed, produced from a matrix drawn by a typographer.

To do this, it drew (figure 18) the desired character on a large layer and marked with 1, or X, the boxes to be blackened, the others with 0, or left them empty. A programmer translated this drawing into commands for the (vertical) scanning: number of the column, number of the first pixel to be filled, number of pixels. This is what we call *run lengths*.

3.4 About bitmaps

The grid of screens can be considered a matrix with each element being 0 or 1. Each such element is called a *pixel* (abbreviation of *picture element*). If each element is a 0.1 inch square, i.e. if there are 10 pixels in an inch, the *resolution* of this grid is said to be 10 dpi (*dots per inch*) (figure 19). The resolution for phototypesetters was very high (often 1200 dpi, sometimes more), resulting in the naked eye seeing very smooth curves and characters. On the other hand, the screens of the first microcomputers or Minitel (see page 33) had a resolution of only

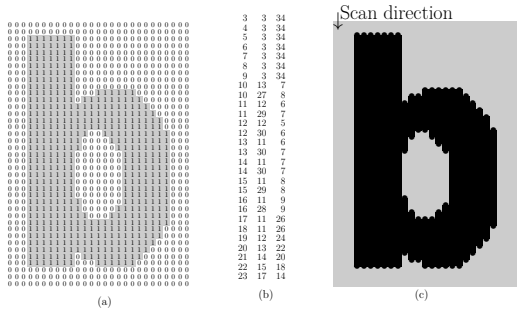


Figure 18: Principle of the Digiset: (a) the binary matrix conceived "by hand" by the typographer, (b) code by range (i.e. run-lengths), (c) image provided by the photocomposer: the bands (here slightly narrowed to distinguish them) are scanned from top to bottom (the returns of screen are not indicated); the gray corresponds to a phase of non-illumination (the screen and the paper are not exposed) and the black with a phase of illumination (thus exposed).

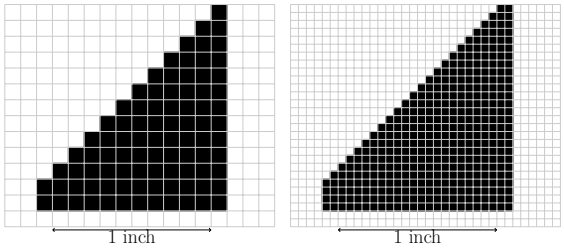


Figure 19: The same triangle rendered as a bitmap at 10 dpi and 20 dpi resolutions.

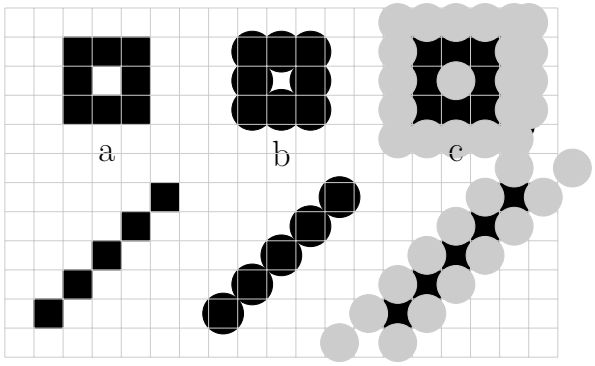


Figure 20: Influence of exposure modes: (a) theoretical form; (b) "white then black" mode; (c) "black then white" mode (here white is gray). After Pierre MacKay [92].

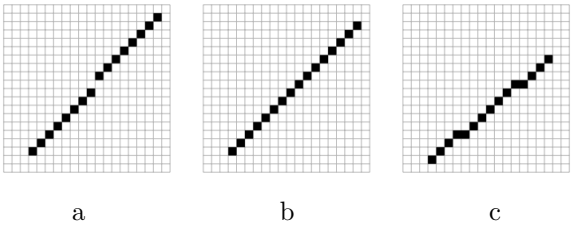


Figure 21: Bresenham's algorithm (1962). (a) A line segment drawn directly using the Cartesian equation $y{=}ax{+}b$ with integers; (b) the same line segment drawn using Bresenham's algorithm: a slight shift allows continuity (no breaks as in a); c) another line segment (with less slant) drawn also using Bresenham's algorithm, still ensuring continuity.

72 dpi, with resulting "pixelated" mosaic-appearing characters.

Such a bitmap is a virtual image to be displayed on screen or printed on paper, resulting in a few differences from the theoretical matrix: the pixels which should be square are often round, like the trace left by rays of light (figure 20); in addition, some output devices (in particular, the LN printers from Digital Equipment Corporation (DEC), the Ricoh printers and some from Xerox, still in use at the end of the 1980s, and some of the black and white screens of the time) did not work by blackening a white zone, but by blackening initially all the paper and by then writing, by sweeping, the white where it is necessary. This gave appreciably different results according to the machine used (figure 20).

The underlying problem with bitmaps is that we go from a continuous world to a discontinuous one. One result in particular is that any slant in relation to the direction of the pixels presents pixelations or so-called staircase effects. Several methods have been used to reduce these effects; they cannot be eliminated (even when using vector fonts, contrary to what we sometimes read in the press). First, and most simply, increase the resolution, i.e. decrease the size of the pixels (figure 19).

Second, use a concept of bitmap not based solely on black and white, but with more subtle possibilities, such as grayscale screens (figure 22) that will appear around 1980, and LCD (Liquid Crystal Displays) at the end of the 1990s (page 52).

Third, and most generally, researchers found ways to reduce the artifacts in bitmaps by using techniques from computer graphics. If we draw a line $y = ax + b$ by writing a loop giving to x the integer values $3, 4, \ldots, 18$ for which we calculate the corresponding integer value y and then blacken the box (x, y), we obtain figure 21a, which is not satisfactory since the slanted line segment is cut in two. In 1962, an IBM engineer, Jack Elton Bresenham, who was working on the first Calcomp plotter using bitmaps, looked at the problem. The Cartesian method doesn't work because the rounding done to take the

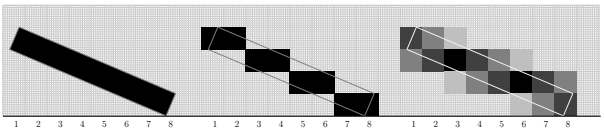


Figure 22: A line segment, left: expected; middle: with Bresenham algorithm; right: with Bresenham algorithm on a grayscale screen at the same resolution.

integer part causes such dropouts. This is correctable, but the correction method known used operations on real numbers, which were very time consuming (especially for computers of that time). Bresenham's algorithm starts from the parametric form of the equation of the line and for each point studies its neighbors [39]. It thus manages to optimize the plot by using only integer operations, which is extremely fast; figure 21b shows the result obtained. This algorithm has been improved to deal with borderline cases and adapted to other curves (including circles) and even to grayscale screens (figure 22). This is, in a way, the archetype of all computer graphics programs used in typography!

3.5 Bitmap fonts

To these technical problems, type designers brought an artistic solution: circumvent the problem by using no or few diagonal lines.

Thus, Adrian Frutiger, who experienced "the passage from lead to CRT, so greedy in memory, then to vectorized representations [...] then to Bézier curves" [102, p. 286], says about his Méridien font, which had already been adapted from lead to Lumitype, "When I saw what the digitization gave, with all these small stairs, I was horrified. [...] So I tried to get around the technical deficiencies by drawing. It was necessary to avoid the slight curvatures of the slightly curved solids and the concave serifs, which would have made the pixelation visible ..." He then drew the Breughel font, which takes these adaptations into account (figure 23). His specific recommendations were to avoid the stairstep rendering by the absence of oblique lines, in particular, to flatten the serifs; to increase the curvature of the curved solids, or on the contrary to flatten them (left side and right side of the two stems of the 'n'); to prevent the ink traps of the holes by enlarging them, and so on.

Other type designers for the CRT had the same problem, for one, Ladislas Mandel with his Galfra design (figure 24). Hermann Zapf also studied the digitization of a subtle typeface design he had designed for lead, Optima, and refrained from completing this work for printers with less than 300 dpi [129, p. 103] (figure 25).

Despite these shortcomings, and the heavy workload involved, many fonts were designed character by character for the three generations of phototypesetters (figure 26). Many fonts were marketed for or sold with photocomposers.

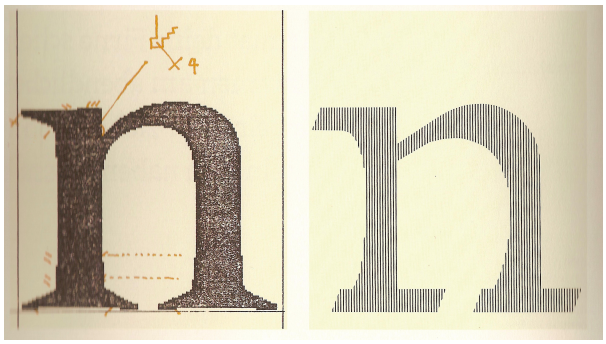


Figure 23: The defects of the Frutiger Meridian scan (left) were corrected by hand, resulting in the Breughel design (right) [102, p. 290].

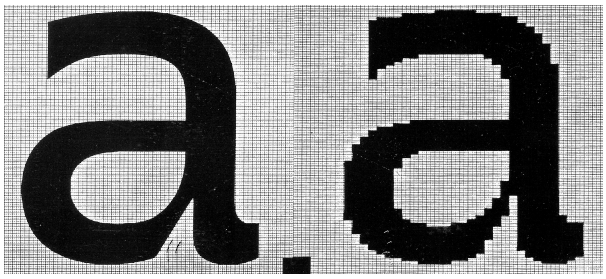


Figure 24: The 'a' in Ladislas Mandel's Galfra typeface (~1978): left, hand-drawn; right, pre-digitized. [94]

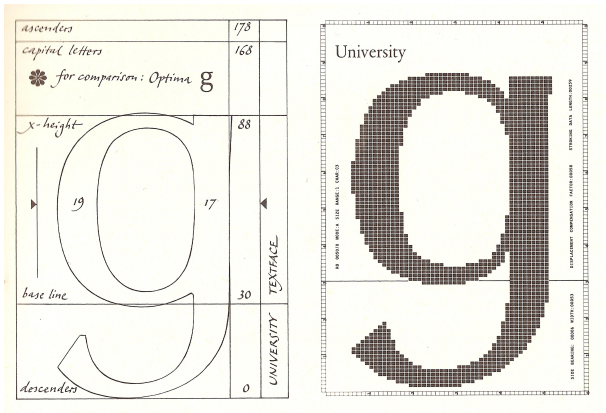


Figure 25: Hermann Zapf preferred not to finish this draft of his Optima typeface for a printer at less than 300 dpi, which could not render the design well [129].

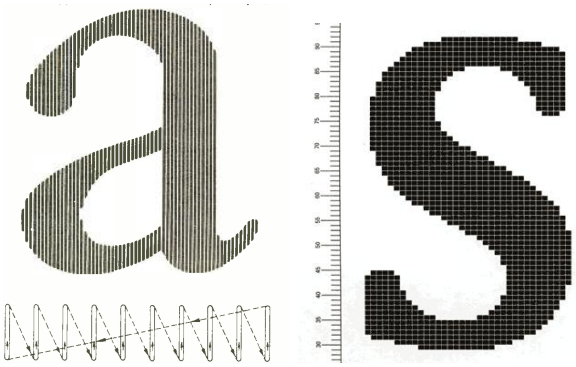


Figure 26: Left: 'a' of the Videocomp composer (1967) with scan and frame return diagram, from a newspaper, 1971. Right: one of the first fonts for Hell's Digiset, Demos by Gerard Unger, 1975 [Courtesy Gerard Unger].

3.6 Research of new typographies and imitations of pixelated characters

While designers like Frutiger and Mandel sought to free themselves from technical contingencies, others used them as a means of expression.

Crouwel's New Alphabet. The Dutchman Wim Crouwel [18, 45] was attracted to digital fonts, which he discovered thanks to Rudolf Hell. One of the first fonts he designed, directly in bitmaps for CRTs, is New Alphabet, in 1967 (figure 27). Since he could not solve the problem of curves or even diagonals, which are always stairstepped in digital rendering, he decided not to use them. So he adapted his fonts to the machine, using only horizontals and verticals, with slightly diagonalized junctions. But then some characters became unconventional (the A for example!). His typeface was not without criticism at the time. In 1983, Charles Bigelow wrote: "A letter is something other than a collection of bits. All curves are eliminated. All shapes are simplified. From a purely technical point of view, the result is an undeniable success: each letter reproduces itself impeccably, even through the largest frames. This new alphabet has only one disadvantage: it is unreadable. It is unacceptable. Without legibility, there is no communication [...] Who can distinguish letters from numbers?" [24]. We also owe to Crouwel the somewhat more customary typeface Claes Oldenburgh (figure 28).

In addition, the look of bitmapped mosaic characters inspired graphic designers. In 1982, Michel Olyff took up the concept of bitmap by drawing pixels, by hand, one by one for his famous poster (figure 29), inspired by a doily embroidered during the first world war.

Figure 27: New Alphabet is a typeface designed by Wim Crouwel in 1967, banning all curves and diagonals; there are no upper/lowercase distinctions. Above, the digitized version [courtesy The Foundry]. Below, the original 'A' (reprogrammed from [18]).

Figure 28: The Foundry's Architype Catalogue font, digitized in 2003 from the Claes Oldenburg font designed in 1970 by Wim Crouwel, which was clearly inspired by bitmap drawings like those in figure 20. [Courtesy The Foundry]

For its part, the Honeywell-Bull computer company used very pixelated characters for its poster of the SICOB computer show in 1982 (figure 30).

3.7 Matrix printers

The principle of drawing characters by a matrix of dots has existed since the end of the 19th century; e.g. the *pin points* of punches, characters for Smith typewriters in 1910, and then characters written at the top of IBM-026 punched cards. It was taken up by *dot matrix* printers in Japan in 1968 and spread to the USA via the LA30 and then LA36 from DEC.

The basic principle is as follows: a print head comprises a number of vertically-aligned pins allowing the spontaneous generation of a column of points. They are propelled by electromagnets and, through a carbon ribbon, print the points of the character images. The characters are defined by a matrix of points, often 5×7 but which, by shifting, in fact defined 9 lines — figure 31 shows the principle.

A 5×7 matrix does not give good results, so manufacturers improved the quality of the characters, while often providing two modes: one of draft quality,

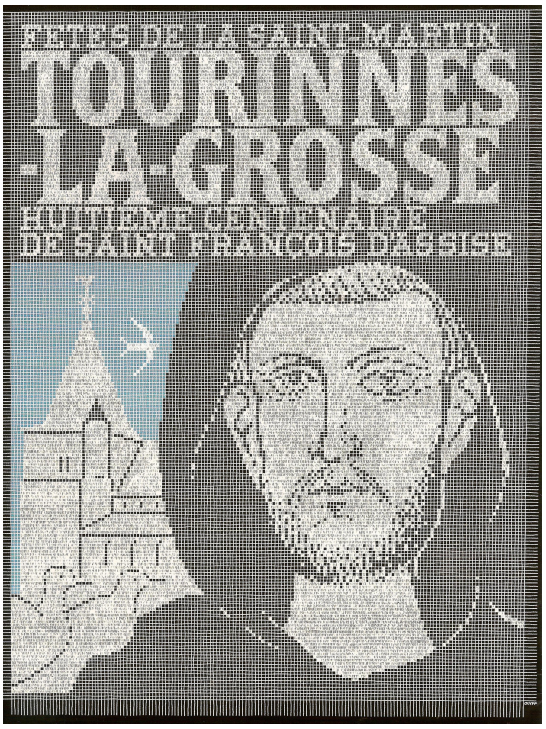


Figure 29: Poster by Michel Olyff (1982) inspired by the pixels of embroidery and printers. [Courtesy Michel Olyff]

Figure 30: Poster of the Honeywell-Bull Company for the SICOB computer show, 1982.

the other of "mail quality". This higher quality was generally obtained either by the simultaneous passage of the reading head with a slight shift in x and y giving more continuity to the final design, or by using a larger number of pins, or both (figure 32).

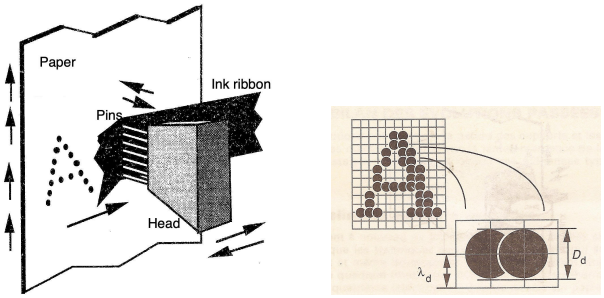


Figure 31: Principle of dot matrix printing and construction of characters by overprinting [52].

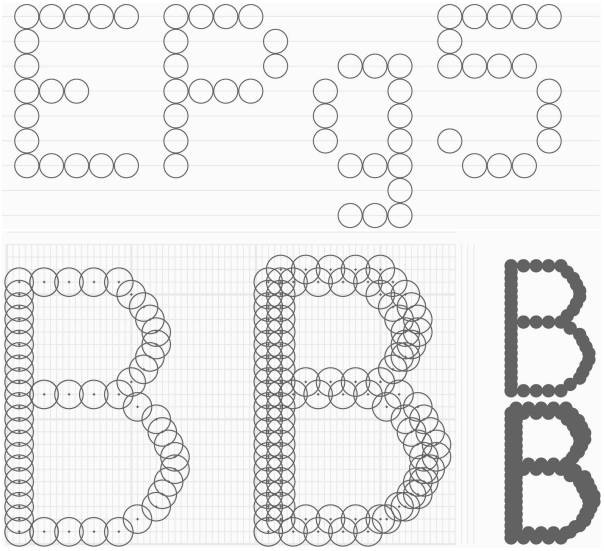


Figure 32: Above: classic character printing of a 5×7 dot matrix with a 9-pin matrix head. Bottom left: construction of bold on a dot printer using a superposition of two images slightly offset in x and y. Right: result of printing a normal B and a bold B. [From a Sanders commercial brochure, circa 1980]

Similarly, italics could be simulated by a slant but also by a more adequate design. Unlike daisy wheel printing, it was then possible to use several character styles, weights, etc. in one line without manual operation. Of course, all these dies were drawn by hand (today's character displays on LED-type lamp panels are made by filling in characters defined by their outlines).

Dot matrix printers were used extensively from the 1970s–1990s as the printers distributed with the early personal computers. The emblematic example remains the 9-pin LaserWriter of the original Macintosh (figure 36). They were dethroned in the mid-1980s by the arrival of laser printers.

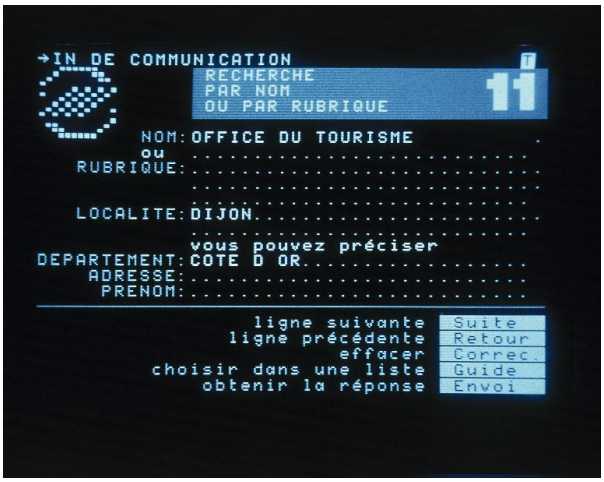


Figure 33: A typical Minitel screen, the home page of the French electronic white pages.

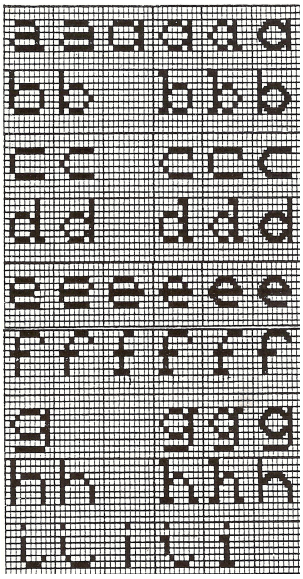


Figure 34: The Minitel characters had been tested with several variants, the choice having been made following readability studies [35, p. 56].

3.8 Three historical cases

The screens appearing with the first mass-market computers had low resolution which, given the enormous size of the pixels forming the characters, gave typography by computer a bad reputation.

Around 1980, two machines were created which used these bitmapped characters (for screen only or with printer), emblematic of this time: in France, the Minitel and in the United States, the Macintosh. We add here a third example, that of a font which was probably the first one designed for such highly "mosaic" characters, Lucida.

3.8.1 The Minitel

Studies on Minitel started in 1979 at CCETT[2] in Rennes (France). The initial goal was to launch a videotex network accessible by a low-cost terminal and then to make an "Electronic Directory" available to all telephone subscribers, which was a commercial and long-lasting success — it was not completely ended until 2012 [97, 93].

The Minitel was a passive computer terminal, consisting only of a keyboard and a screen. The screen (figure 33) was a text matrix with a size of 25 lines by 40 columns. A line of text could thus receive 40 characters, of fixed size as for a typewriter; this is around 1980.

Each character was formed on a grid of 7 pixels in width by 10 pixels in height (a little larger than the 5×7 of matrix printers). These typefaces were much criticized at the time. For many people (especially typographers), it was the first contact with computerized typefaces. However, they had been the subject of extensive legibility studies (in the spirit of the work done since Javal, see [26]). Figure 34 shows some of the typeface models used for testing at CCETT.

[2] *Centre commun d'études de télévision et télécommunications*: Joint Center for Television and Telecommunications Studies

3.8.2 The "original" Macintosh

The Macintosh was the first mass-market personal computer launched by Apple Computer, in January 1984. The project was started at the end of 1978 by Jef Raskin, who wanted to create a computer that was easy to use and inexpensive, and therefore accessible to average consumers. He joined forces with Burrell Smith and then, in 1980, with Steve Jobs who introduced the mouse (which he had seen working at PARC, page 47). In some aspects, notably the graphical user interface, the Macintosh followed the Apple Lisa computer, released a year earlier.

The Macintosh's display device was a 1-bit CRT screen (black and white) with a resolution of 512×342 pixels. The desktop processing (DTP) standard (which we still find for web images), corresponding to 72 dpi, would come from there. This value is not insignificant. It is close to the 72.27 typographic points per inch of the Americans: 8 points (pixels) of the screen measured thus 8 points (typographic); a character of 12 typographic points was drawn with 12 pixels, including the slope. To this computer, it was also possible to connect a printer with 9 pins, the ImageWriter, designed by the Japanese company Itoh and already in use at Apple. This printer had

Figure 35: The first Macs were noted for the quality of their fonts; for the general public, these were the first computerized fonts seen! At left, a menu (composed in Chicago) allowing the user to choose a font, its variant (bold, italic, etc.) and its body; at bottom, demonstration of font combinations [Images from the *Guide Marabout du Macintosh*, 1984; courtesy Susan Kare]. At right, the Font Mover icon, by Susan Kare for the original Macintosh (1982).

a resolution of 144 dpi. A printed text thus had the same size as its image displayed on the screen, but with twice the resolution (one screen pixel corresponding to four pixels on paper).

This first Mac came with four fonts, all designed by Susan Kare, to whom we also owe nearly all the Mac icons (such as the Font Mover icon, figure 35). These fonts were named Chicago, Geneva, New York and Monaco (figure 36). Chicago was a special case: it was the "system" font used to display the Mac's commands and which existed only in size 12, deliberately a little bold. These fonts were drawn directly on screen by Susan Kare, in a grid, using a small editor designed by Andy Hertzfeld, letter by letter, size by size. Hertzfeld also wrote small programs to distort these characters to make several variants such as bold, italic, shaded, raised, etc. (figure 35), each of which can be combined.

These fonts were usable by the Mac's word processing system, MacWrite, which was one of the first mainstream WYSIWYG applications. Professional typographers, who obviously didn't take this new "typography" seriously, found it hard to accept that a few years later this same Mac would offer typefaces printed with a quality bordering on their tradition.

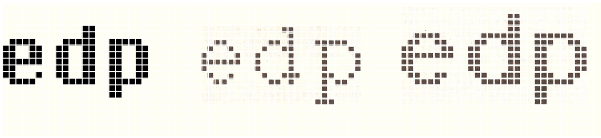


Figure 36: Three of the original Macintosh fonts: Chicago (12pt), New York (12pt) and Geneva (14pt). [Courtesy Dafont and Susan Kare]

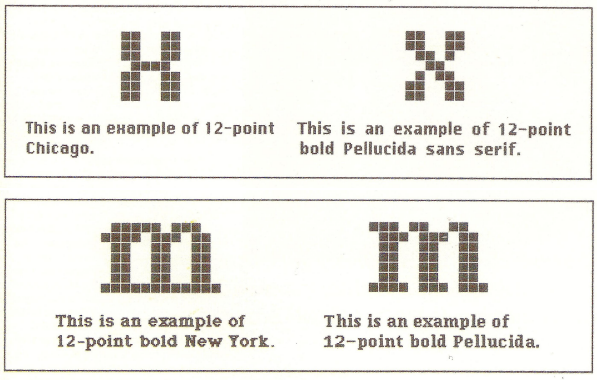


Figure 37: Bigelow & Holmes showed, with their Pellucida font initially conceived for the VAXstation, that even with the low screen resolution of the original Mac, one could improve the readability of the characters. [*Macworld*, 1985]

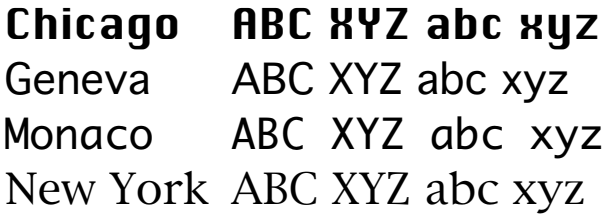


Figure 38: The four main fonts of the original Macintosh, redesigned in TrueType by Bigelow & Holmes using Ikarus, from the original bitmaps by Susan Kare [29, 66]. The nominal size shown here is 32 pt. [Courtesy Bigelow & Holmes]

Let's anticipate the rest of this article a bit... These first bitmapped fonts were redrawn in TrueType by Charles Bigelow and Kris Holmes (figure 38), who used a beta version of the IkarusM software, using only line segments and circular arcs for the curves (like Renaissance drawings, page 35); these were converted into quadratic splines. TrueType Chicago was designed to render bit-for-bit the same (except for a few symbols) as the original Chicago bitmap font, at the system size; the other TrueType designs diverge further from the originals [29].

Figure 39: Lucida, first released in 1984, was the first typeface design designed for low-resolution printers. The image shows the Lucida seriffed lowercase 'a' at three resolutions corresponding to 8, 10, and 24 point fonts on a 300 dpi laser printer. The effects of undersampling (insufficient resolution) are evident. At left, the lowest resolution shows a strongly aliased image with "jaggies" that disrupt the curved and diagonal letter elements. At right, the highest resolution still shows noise along the contours. When these idealized bitmaps are reconstructed as actual images by a laser printer, the sharp images of the stairsteps are smoothed, but some distortion of the forms remain. (Text and images from [30].)

3.8.3 Lucida

Although it is a font rather than a computer system, and although it was released later than the previous examples, let us point out that Lucida by Bigelow & Holmes was the first font designed specifically for (not adapted to) low-resolution printers, such as the 300 dpi laser printers of the time (figure 39 and [125]). They also created (by hand) a companion screen font, Pellucida (figure 37).

4 Mathematical character models

It was immediately tempting to have the computer do the tedious work of preparing the *run lengths* previously mentioned, and as early as 1965 computer scientists began to write such systems. The basic idea, used almost universally, was to consider that a character is a mathematical surface (defined by its contours) which is projected onto a bitmap and which must be filled. We will first recall that these contours have been known since antiquity, then we will see how they have been improved and adapted to the needs of typography.

In general, these models were based on the existence of an already-drawn character that was to be scanned (Ikarus for example), but some went further by proposing tools to prepare these outlines (e.g. CSD with a modular approach, or METAFONT by using the ductus of calligraphers). Figure 1 showed the chronological evolution of these systems and tools.

4.1 Models in antiquity

It is known that during Roman antiquity, since at least the beginning of our era, patterns of lettering

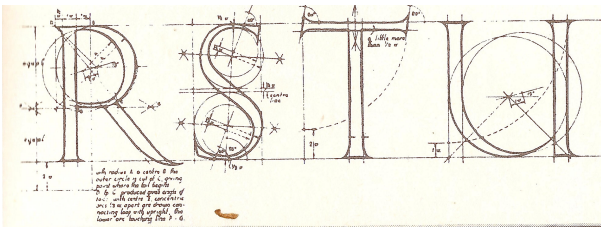


Figure 40: The Roman capitals on the Trajan column were drawn with a ruler and compass. Study by Edward Catich, 1968 [42].

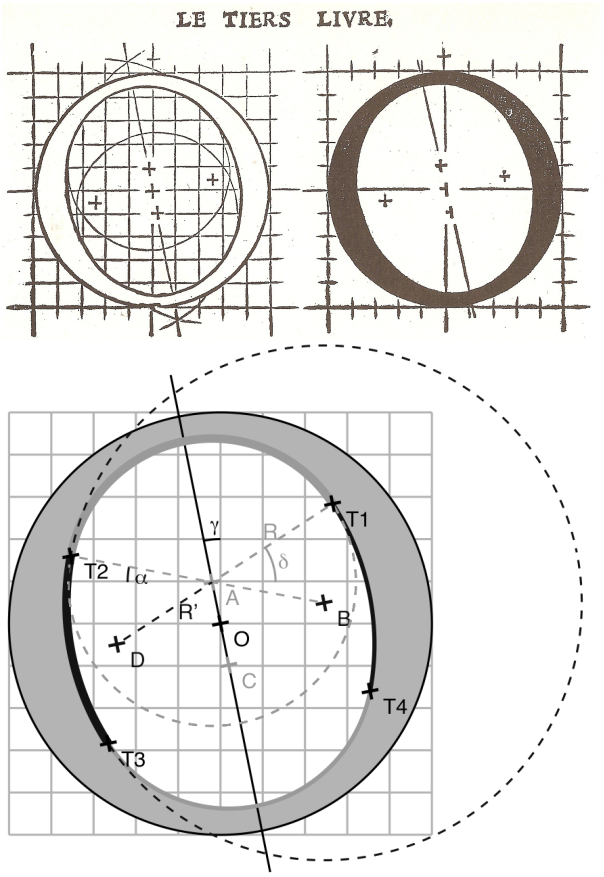


Figure 41: Above, the O capital as seen by Tory, *Champfleury*, 1529; below, a trigonometric interpretation [13].

were already being used with the ruler and compass. Thus, the capitals of the famous Trajan column (dedicated in 113 CE) have been shown to be rigorously based on straight line segments and arcs of a circle (figure 40).

During the Renaissance, various authors proposed models for the letters engraved on the pediments of public or religious buildings based on the Roman capitals and using the only constructions

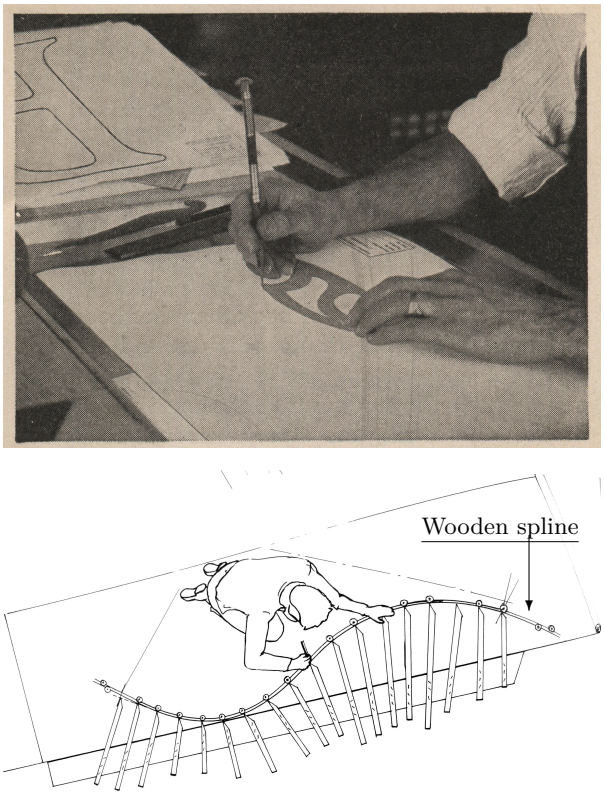


Figure 42: Historical tools for complex curve drawing. Top: French curve used when preparing dies for Linotype, around 1930; bottom: wooden spline used in naval carpentry, around 1990 [54].

then known, the ruler and the compass. These models are those of Damiano de Moile, Felice Feliciano, Luca Pacioli, Luca Orfei, etc. They were frequently quoted by typographers [1, 36, 101, 121, 130] or analyzed mathematically [13, 81]. It was Dürer in Germany and then Tory (figure 41) in France, in the sixteenth century, who made the first models substantively applied to typefaces for printed texts, with not only capitals but also lowercase letters. This way of modelling typefaces with a ruler and compass was long-lasting, since it is found in the eighteenth century (e.g. the Romain du roi [15]) and in preparatory drawings by Eric Gill in 1927.

Many professions (carpenters, marine carpenters, architects, industrial designers, road engineers, boilermakers, for automobiles and airplane wings, etc.) have had the problem of drawing harmonious curves passing through a certain number of points but which could not be drawn with the compass in a simple way (that is, without using many arcs of circles). This was solved manually by using tools (figure 42) such as the French curve or the spline (a word that will soon be found again in this article!).

4.2 Curves, mathematics and approximation

Mathematical studies on curves were initiated by the Greeks and the Romans and developed at length by their successors, in particular at the end of the 17th and the beginning of the 18th century (Euler, Monge, Cauchy, Legendre, etc.), with the theory developed further in the 20th century (Hermite, Bernstein, ...).

When the equation of a curve is not known, it can be approximated by simpler pieces of curves.

Lines and circles. The simplest curves are line segments. This is what plotters did, where the curved body of an R is replaced by five straight line segments (figure 5, right). Less simple curves are circles. This is what Tory did (figure 41), replacing the vaguely elliptical lower curve of his O by four arcs of circles [13].

Conics. For a long time, mathematicians looked for curves more complex than straight lines and circles. It turned out that circles belong, together with the parabola, the hyperbola and the ellipse, to the class of conics. It is therefore natural that some font models use conics and in particular parabolas; for example, Coueignoux (page 38 and [43]) and TrueType (figure 70).

Superellipses and spirals. Some ellipses have a more rectangular shape, or even the shape of a rectangle with rounded corners: the superellipses (called super eggs by the Danish poet–designer Piet Hein, 1905–1996). Typographers have used them (figure 43). But, what interests us most here is that these curves have also been used to draw pieces of type outlines, for example in the Itsylf (page 38), CSD (page 38) and METAFONT [80] systems.

The kinematic study of road layouts, then of railroads and highways, led to the use of special curves for the connection between two straight segments (change of direction of a road for example). The most "comfortable" trajectory is not a circular arc but a clothoid arc (or Cornu spiral or Euler spiral). These spirals were used in typography by Purdy for the Varityper (figure 44) and some researchers currently recommend the use of such clothoids [90].

Bézier quadratics and cubics. Shortly after 1950, when computer graphics started developing, engineers needed to define curves to calculate profiles of, for example, automobile body panels. This simulated what marine carpenters used to do, i.e. to use, as in figure 42, physical splines, which first meant to cut these large curves (now called splines) into smaller pieces. Of course, the small pieces had to joined together while keeping the curve smooth. This is how Pierre Bézier, an engineer at Renault (where

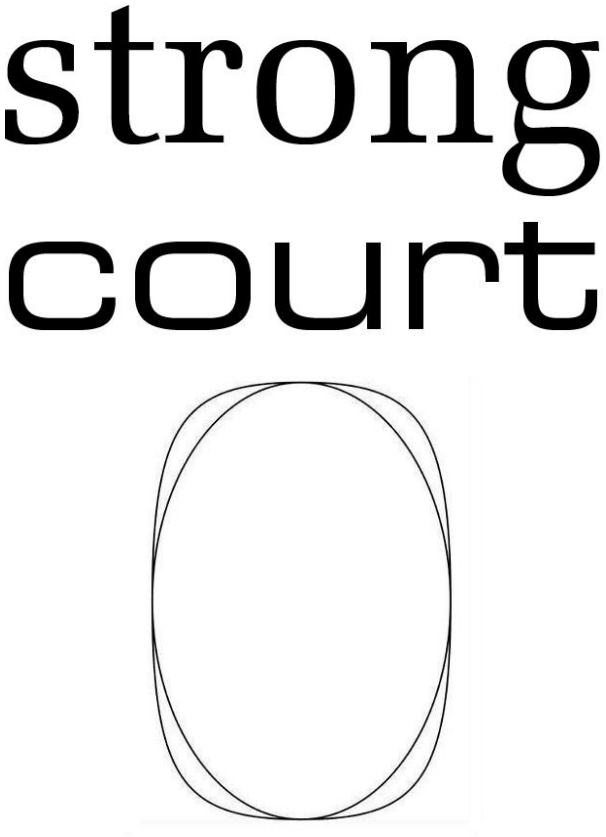


Figure 43: Superellipses have been used in type design, notably for 'O'. Top: Melior by Hermann Zapf (1952); middle: Eurostile by Aldo Novarese (1962) [Courtesy Peter Karow]; bottom: Lucida Grande Mono DK by Bigelow & Holmes (2014) with, inside, an ellipse with the same axes [Courtesy Charles Bigelow].

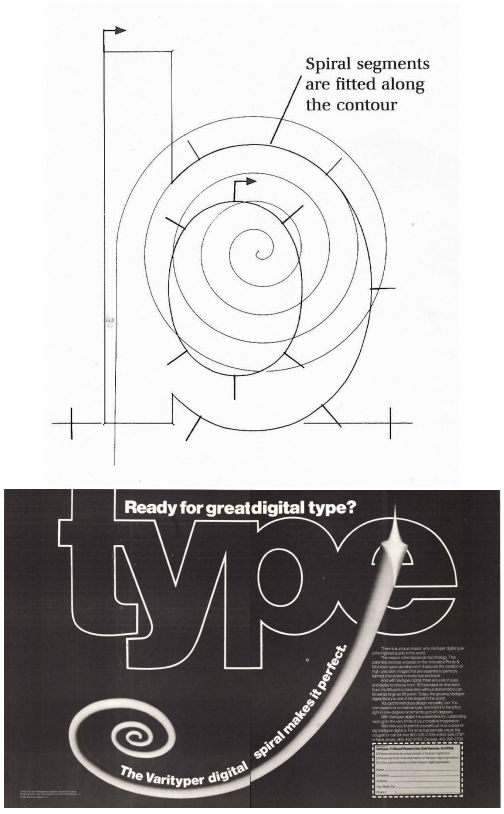


Figure 44: Approximation of a character via pieces of spirals. Top: an illustration of the principle (from [70], with permission of Peter Karow);
bottom: an advertisement by Varityper (appeared in *U&lc*, Aug. 1984).

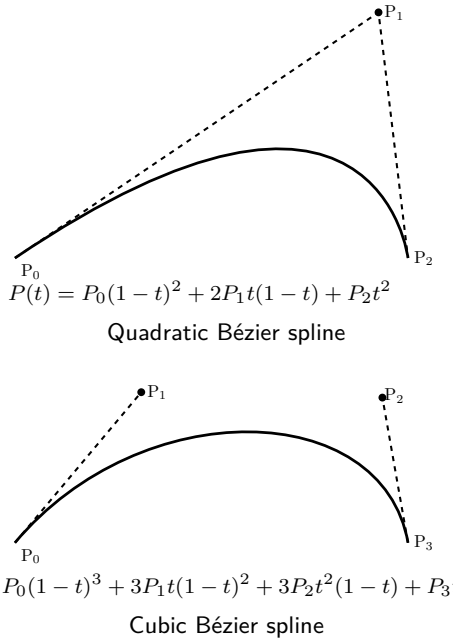


Figure 45: Diagram and formulas of two Bézier curves frequently used in digital typography.

he had already created Unisurf, the archetypal CAD software), studied the curves that now bear his name (Bézier curves or B-splines), which are in fact special cases of Hermite and Bernstein polynomials. To be practically usable, these curves had to be easily and quickly computable. The French mathematician De Casteljau (at Citroën) discovered a very efficient algorithm for plotting based on binary divisions.

Let's just show here the Bézier curves used in typography and in particular the two most common models, the quadratic and cubic splines (figure 45). Their names derive from "quad" (square, therefore two) and cube (three), terms with which, since the Renaissance, mathematicians named the powers of two and three.

The cubic Bézier curves (figure 45, bottom) are defined by four points P0, P1, P2 and P3, i.e. by the two tangents P0–P1 and P3–P2. The curve passes through the points P0 and P3 and is included in the parallelogram P0–P1–P2–P3, which gives a first

approximation. Introduced in the typographic world by Xerox, and made widely known by Adobe, they are used by many font formats.

Quadratics (figure 45, top) are defined by the triangle P0–P1–P2 and are in fact pieces of parabolas. They are also used in some font formats, notably TrueType (see figure 70). They ensure the continuity of tangents at junctions, but not those of curves, and therefore need to be divided into smaller segments than cubic curves. Thus, although each quadratic piece inherently takes fewer points to define than a corresponding cubic, more pieces are needed. When converting from quadratics to cubics, the curves are not quite equivalent.

5 First contour-based fonts

The trial and error period of the 1960s is not well known. But, thanks to Lynn Ruggles [112], we can mention a few names.

ITSYLF, the first font generation system, 1968. Ruggles considers this to be one of the very first systems specifically dedicated to type design, although it was never made operational. This system, ITSYLF [100], developed by Mergler and Vargo, had two novel features compared to the few existing systems. First, it used conics, more precisely superellipses (or Lamé curves), rather than arcs of circles, to approximate the curves. The arcs of curves are defined by superellipses of the form $(X/A)^F+(Y/B)^F = 1$. For two values of A and B, we can vary F and obtain a series of more or lesser curved lines passing through these two points A and B.

Second, it does not define a font, but a family, thanks to a skeleton and parameters allowing to refine the final shape. These parameters are defined for the E, and then adapted automatically to the other letters. Figure 46 shows the skeleton of a C and variations calculated automatically for this C by varying the parameters. The whole font would thus always be homogeneous.

This is not far from what will be possible a few years later with METAFONT (page 41).

CSD, FRANCE and Coueignoux' works, 1973. The Frenchman Philippe Coueignoux developed, in 1973 at MIT (USA), what Knuth considers "the first use of sophisticated mathematics to describe letterforms by computer" [82]. His work was mainly published in his two theses, *Compression of Type Faces by Contour Coding* in 1973 and *Generation of Roman printed fonts* [43] in 1975, the latter being widely cited: even though his "academic" research did not lead directly to industrial developments, his ideas are found in many later systems.

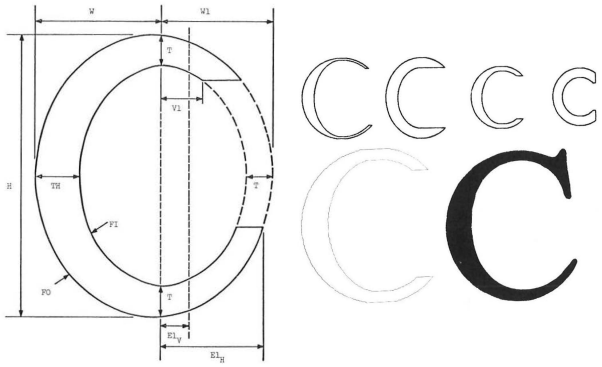


Figure 46: The ITSYLF system of Mergler and Vargo (1968). On the left, the schematic of the C, with indications of the parameters (W, W1, T, V1, ...). On the right, top: letters printed by varying these parameters; bottom: from the basic C, the C of Times Roman can be defined. [Courtesy *Visible Language*]

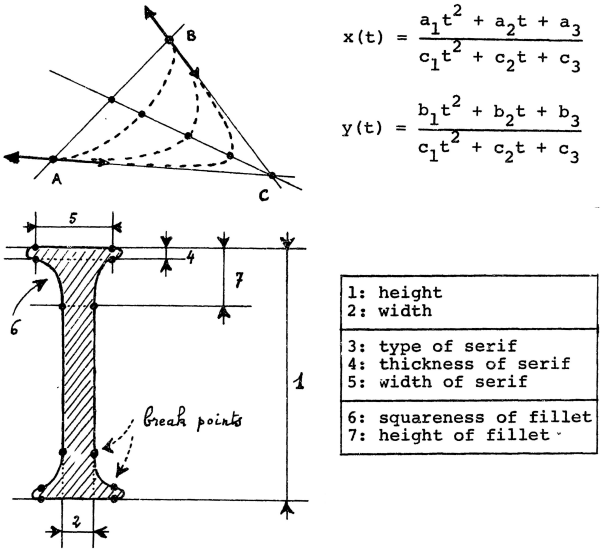


Figure 47: Coueignoux's use (in 1973) of conics: the curves (in dotted lines) are defined by the four coordinates of the points A and B and by the distance from C to the curve. Bottom, examples of parameters to define a character. [43]

Coueignoux [43] describes a model for encoding character outlines, which he developed apparently without knowing the concurrent work of Karow (which we'll discuss subsequently). As in ITSYLF (though without knowing it either), Coueignoux uses line segments for stems, bars, etc., and conics for the arcs of terminals and superellipses for bowls. Similarly, characters are defined using parameters.

What is new is that Coueignoux uses a structured grammar (like those of Chomsky, well known to academic linguists and computer scientists since

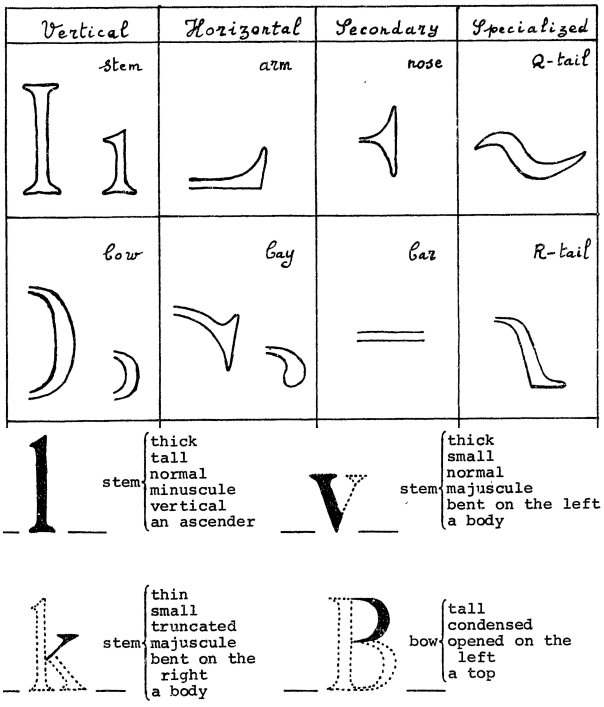


Figure 48: Some basic CSD primitives and extracts from the generic grammar. [43]

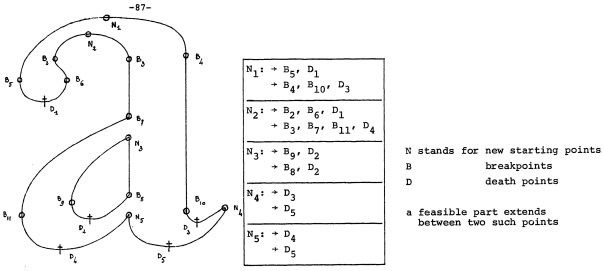


Figure 49: The FRANCE software, by Coueignoux in 1975, makes it possible to find the breakpoints of splines and segments. [43]

about 1965) to define his characters. *Primitives* form the basic elements allowing to define characters. This is an incremental definition of characters, of which we find new attempts since the beginning of our third millennium [67]. This generic method is associated with a font production system, CSD (*Character Simulated Design*), as shown in figure 48, bottom.

Moreover, the FRANCE system (*Font Retrieval: A Natural Coding of Edges*) is a program that does what is now called *autotracing* (like Ikarus, as we will see): starting from a character known by its representation in the form of a matrix of points, it deduces an algebraic description, using splines (figure 49).

Figure 50: The first Ikarus hardware, 1973. [73]

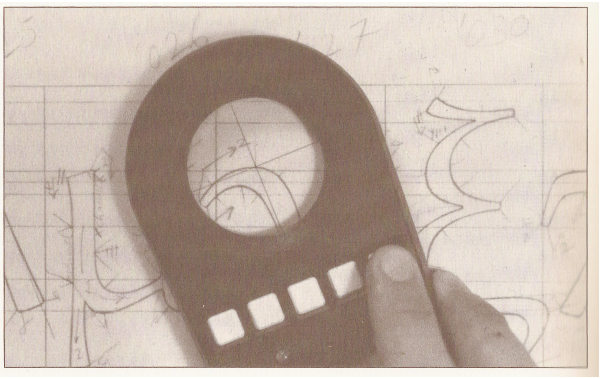


Figure 51: Using the Ikarus "mouse"; here in the context of the Euler project at Stanford (1983), led by Bigelow, Knuth, and Southall, with characters designed by Zapf. [119]

6 Ikarus

URW (named after its first two founders, Rubow and Weber) was established in 1971 in Hamburg, Germany. Peter Karow joined the company in 1972 and was responsible for the automation of the production of fonts for photocomposers. See his background and his research and development in [68, 71, 72, 73].

Peter Karow's first "customer" was Walter Brendel [73], a type designer who was responsible for fonts such as Lingwood and Volkswagen and who was then starting a digital type library for photocomposers (this was only around 1970; it became the basis of the "TypeShop Collection" of Elsner+Flake). His customers wanted modifications to his fonts, such as "blacker", "spaced out more", "shaded", etc., which could not be done on the bitmaps and had to be done from the drawings themselves. Karow first made tests with characters cut from 15 cm high vinyl plates (quite similar to Frutiger's scratch cards) and then understood the interest in digitizing the characters to work with reusable formats.

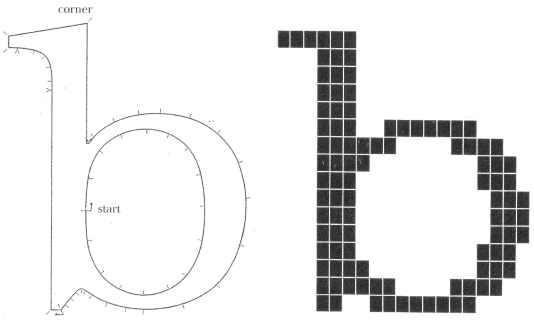


Figure 52: Left: outline of a 'b' with its guide points for Ikarus; right: the same after translation from IK format to bitmaps, here at low resolution. [70]

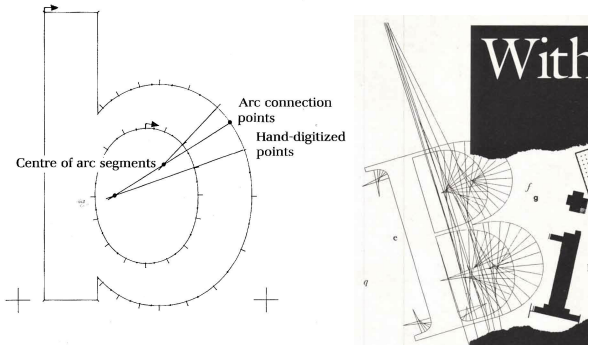


Figure 53: Left: approximation of a 'b' by arcs of circles [56]; right: Bitstream advertisement based on the construction of a 'b' by arcs of circles [*U&LC*, vol. 12, no. 4, Feb. 1986].

Karow substantially started his project at the end of 1972, in collaboration with Aristo, a CAD company from Hamburg, who was responsible in 1960 for the Perthronic table which drew stick letters (page 24), and later for the Aristogrid tablet with cursor. He named it Ikarus in May 1973. The commercialized system included (figure 50) a digitizing tablet controlled by a cursor, cousin of the mouse (figure 51), and a computer with CRT and alphanumeric screen, keyboard, etc. Ikarus was written in Fortran and was quickly adapted to the VAX and Sun minicomputers on which many professionals used it. (These workstations were called "mini" in comparison to the large computers of the time, but were still almost entirely used only by companies and had nothing to do with personal computers.)

Ikarus was, first, a system for digitizing character drawings. By clicking — with the cursor box moved over the enlarged character on the tablet — on a starting point, angles (or corners), points distributed *on the curve* (and in particular the ends of the curves and the points of inflection) and tangents (with the help of two points allowing measurement of the angle) for breaks in continuity, one obtained a set of coordinates, stored in a format named IK [68]. The system analyzes these points and deduces a mathematical description of the contour of this character using cubic splines. The first applications of this system were to calculate directly the bitmaps, or more precisely the run lengths, making it possible to control a photocomposer and in particular the Digiset (page 28).

The next applications of this system were for plotters that only worked with line segments and arcs (page 23). When using this format to draw the outline of such a character, the IK format was then transformed into another format, DI, describing the outline with vectors and circular arcs. The center and radius coordinates were calculated by Ikarus based on the points of the IK format [68].

Early Ikarus customers also used this DI format, either locally (e.g. for arches, Bigelow, page 34) or systematically for all curves (this was the case for the early Bitstream fonts, figure 53 and [6, chap. 6]).

In the 1980s, URW played an important role as a font vendor and, following the advent of PostScript, improved Ikarus.

7 Filling, rendering and hinting

7.1 Bitmaps and filling

Filling of characters which are defined by their contours requires calculating the zones of the bitmap which will be scanned, line by line, by the laser beam (which comes back to calculating the run lengths of the photocomposition, see page 28), taking into account the theoretical contour. We reuse the scan conversion techniques developed since the 1960s for computer graphics. The principle is to follow the curve line by line and mark the pixels whose centers are inside the contour delimited by the curve (figure 54). This method was adapted to characters in the 1980s by researchers such as Ackland, Bétrisey, Gonczarowski, Hersch and Pavlidis [60, 109]. Similar methods are used as well with METAFONT84 and described in [57, appendix F.1.4] and [83, chapter 24].

The difficulty is to determine the angles (e.g. vertex of an A) and not to fill in extra or omit some pixels (dropout) because of singularities of the curve. Bad detection of such points explains why some printers of the 1980s erroneously drew horizontal lines across the whole page, the point of a V, for example, having been badly detected.

These filling operations are not independent of those of rendering and hinting (discussed below)

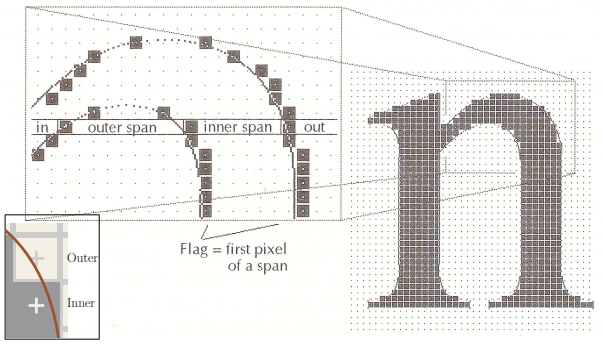


Figure 54: Principle of the filling of a character by marking the limits of the internal zones (inner span) and external (outer span) of the zones to blacken in a bitmap on which one projects the theoretical curve of the character [60]. [Courtesy Roger Hersch]

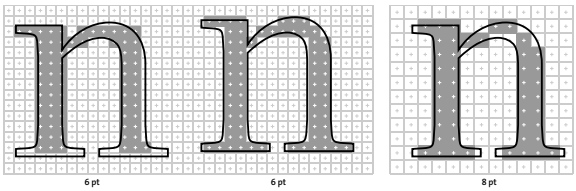


Figure 55: For the same glyph (with thick black lines), the blackened pixels depend on the definition (here, pixel size of 6 pt and 8 pt) and on the precise position of this glyph in the grid.

which allow more precisely refining the choice of the pixels to blacken.

7.2 Rendering improvements, hinting

The filling algorithm does its job well, but the results depend on the resolution (see figure 55 for cases of 6 pt or 8 pt pixels) and also on the position in the grid (again figure 55, the two 6 pt cases, and figure 56). Many rendering defects appear in this way, for example unevenness of descenders, disappearance of thin parts (serifs, ties, swashes), appearance of holes, etc.

This phenomenon is only noticeable for small and medium sizes and low resolutions (< 200 dpi, which is (commonly) the case for screens and for the first laser printers). To limit these phenomena, many fonts have been designed specifically for specific screen sizes (e.g. sizes 10 to 12). Let us cite Verdana by Matthew Carter (1996), the typefaces Base 9 and Base 12 from Zuzana Licko (1995) and Hachette Multimédia by Olivier Nineuil (1996).

The Ikarus system, and those that we will see later on (METAFONT, Fred, etc.), had all the information to prepare the bitmaps; the basic idea being to make (very slight) shifts in the theoretical curve

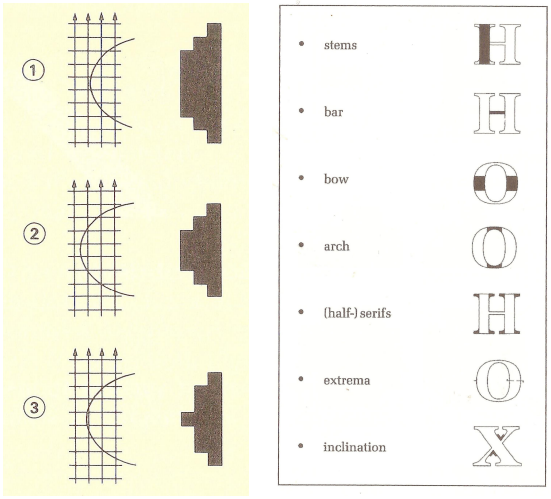


Figure 56: Examples of needed rendering improvements (in 1 and 3 the top of the curve is too close to a pixel border: the result is bad) and, right, some of the cases studied by Karow since 1981. [68]

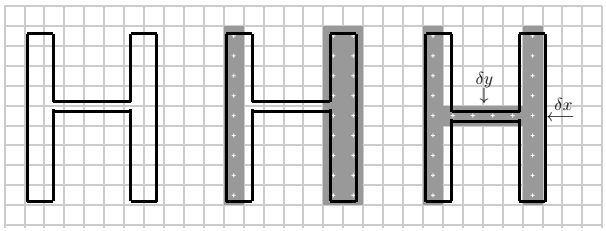


Figure 57: Simple example of hinting. Left: the outline of an H; center: how it would normally be translated into pixels; right: displacements, downwards for the bar and backwards for the right stem, give a better rendering.

so that the filling would be more in the spirit of what is expected (figure 57). But with the appearance of PostScript, or rather raster image processors (RIP), we will see that it is more complicated.

There is no perfect solution to this problem even today, but many approaches have been made. Descriptions of these methods can be found for example in [57, 60, 70, 109] and, recently, some manufacturers' websites explain their methods. The difficulty lies in the way to express what one wants to do in a language (or by a method) accessible to the type designer, who is the only one competent for these drawing problems. Hinting methods are thus characteristic of font systems, whether Ikarus, Type 1, TrueType or OpenType.

8 TEX and METAFONT

Around 1975, the American mathematician Donald Knuth was revising a volume in his magnum

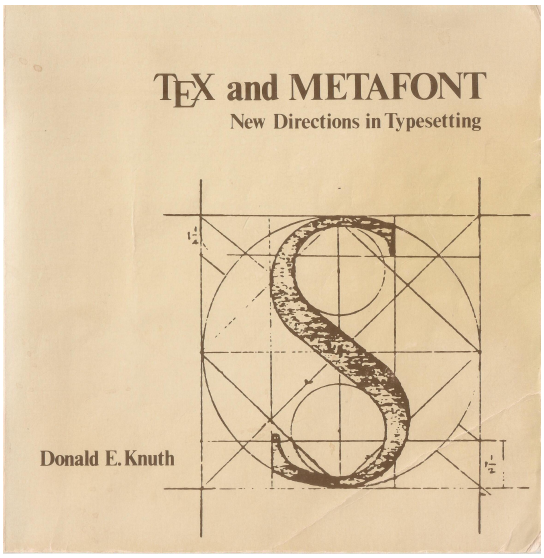


Figure 58: The book in which Donald Knuth first presented (1979) both TEX and METAFONT. [80]

opus, *The Art of Computer Programming* (in brief, *TAOCP*), and realized that the typesetting and printing methods that had become prevalent — not only the composition of mathematics but also that of text — were falling far short of the quality of his earlier editions. He then embarked on the adventure of building himself a typesetting system using digital fonts for which he also defined a construction tool. This is the TEX+METAFONT "couple", with Knuth's first publications in 1978–79 [78, 80] (figure 58), which *TUGboat* readers know well!

In the 1970s, there were few text editors (see e.g. [55, 108, 124]) and they were devoted to dedicated devices with hardwired fonts: typewriter terminals, line printers, and, rarely, second generation phototypesetters (only n/troff, the Unix documentation language from Bell Labs, can be cited). Furthermore, researchers were in the early stages of mathematically defining digital fonts (new tools such as CSD and Ikarus were still confidential). So Knuth felt obliged to create something new! For a general view of this story, see [22], and let's mention right away that Donald Knuth has published extensively about TEX and METAFONT (see in particular [80], [82] and [86]). Many practical aspects can also be learned in *Fonts & Encodings* by Yannis Haralambous [57].

For his TEX system, originally intended for works including significant mathematics, Knuth also needed a font for mathematics. He decided to design such a font himself and soon realized that he needed a program to design not just characters, but entire fonts and even font families. This is how METAFONT was born, around 1979.

Since the primary purpose of TEX and METAFONT was to typeset *TAOCP*, it was necessary to test METAFONT output with something better than the impact printers then in use. Fortunately, Stanford University had a copy of the first raster printer, Xerox's XGP (see section 9.3). As early as 1977, Knuth was able to use it, thanks to drivers written by Frank Liang and Michael Plass (two of his Ph.D. students). Around 1980, David Fuchs (another of his students) implemented the new concept of device-independent drivers (DVI), including one for the new Versatec chemical printer used then in the TEX project. This was before the first laser printer (see page 48).

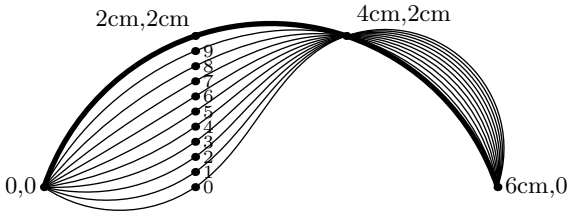
8.1 Basic principles of METAFONT

Knuth embarked on an historical study of typography. He learned that, unlike the hot metal types made with a Benton pantograph (and also unlike phototypesetter fonts with optical lenses), traditional movable types did not use geometric scaling. For example, the glyph of an A at body size 16 pt is not just two times larger than the same A at body size 8 pt. Reasons can be due to human vision, printing details (such as ink spreading), etc. That means that a model for font has to use mathematical variables to be general. His study of old type models [81, 86] (as well as the references cited earlier, page 35) showed that characters may be mathematically defined by curves. Furthermore, from studying calligraphy, he discovered the importance of the *ductus* (the shape and order of the strokes used to compose letters) to draw characters, and that a letter can be defined as the trace of a pen moving (along the ductus).

Unlike Ikarus and other products that proposed tools to digitize existing types (at least as sketches), Knuth wanted METAFONT to enable creating a character family from scratch, and without the user having to know any underlying mathematics. Typically, a user says "I would like a nice curve crossing points `(0,0), ..., (6,0)`" and METAFONT chooses the best spline (figure 59). This is why METAFONT is categorized as a declarative programming language (about which we will say nothing further, as beyond our scope here).

8.2 METAFONT79, and experimentation

Here we appended "79" to METAFONT since Knuth published the first version of METAFONT in 1979. (He first produced machine-drawn letters in 1977, writing directly in the SAIL language.) He developed METAFONT79 concurrently with the first version




```
% thick line:
draw (0,0)..(2cm,2cm)..(4cm,2cm).. (6cm,0);
% bunch:
for a=9  step -1 until 0:
 draw (0,0)..(2cm,a*.2cm)..(4cm,2cm).. (6cm,0);
endfor
```

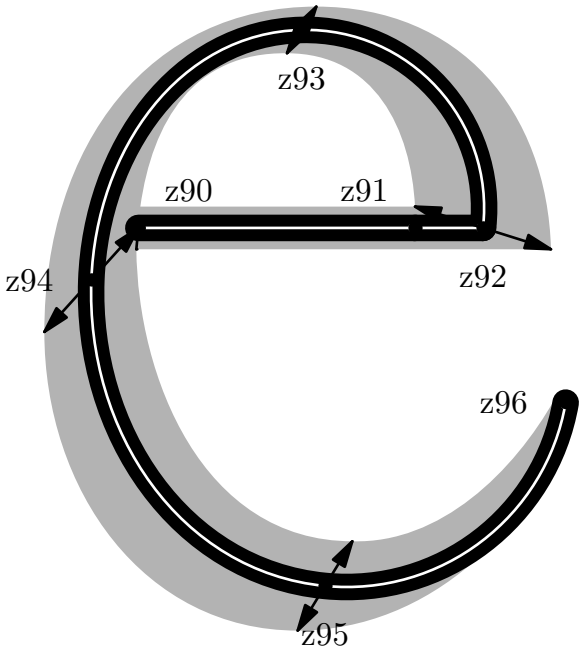
Figure 59: This bunch of curves (top) was produced by the given METAFONT79[4] program (below). (Coordinates are added for readability.)

of his Computer Modern typeface (January 1980) with the help of renowned calligraphers and typographers, such as Charles Bigelow, Matthew Carter, Kris Holmes, Richard Southall, and Hermann Zapf. Their remarks, as well as those of his computer science students (especially John Hobby, who designed the key algorithms of METAFONT [63]), led Knuth to completely revise the METAFONT language and program, which he first released in 1984.

During those years, 1980–1984, all of the above, along with many others such as Vaughan Pratt, Lynn Ruggles, John Seybold, Gerard Unger, et al., were involved with the Stanford Digital Typography Program of that time, a set of lectures, seminars, workshops, etc., dedicated to a mix of mathematicians or designers. They strongly influenced METAFONT (see [22, p. 89]). When it's necessary to distinguish between METAFONT's two major incarnations, we specify METAFONT79 or METAFONT84 below.

The main goal of METAFONT79 was to produce all the bitmaps of each font of a family, at all expected body sizes. That implies the strong use of parameters and variables. In figure 59 you can see that the abscissa of the second point is defined by using a parameter, `a`, as a scaling factor `(2cm, a*.2cm)`, i.e. this abscissa is not a numeric constant but a variable. In practice, these variables may represent the body size, boldness, or any other dependencies (for practical examples of such parameters, see figure 63 and figure 64). This is immediately clear for programmers; however, if you look at any glyph description in graphical tools such as Fontographer, Fontforge, Fontlab, and Glyphs, you'll see that coordinates of glyphs are always constant.

[4] In this figure, and in forthcoming ones, METAPOST has been used instead of METAFONT79 or METAFONT84: these languages are, in that case, almost equivalent.




```
epen ...; %grey:
draw z90--z91--z92..z93..z94..z95..z96;
cpen 20; %black
draw z90--z91--z92..z93..z94..z95..z96;
cpen 2;   %white
draw z90--z91--z92..z93..z94..z95..z96;
```

Figure 60: Three stacked e characters designed with METAFONT, using the same ductus (the path `z90 .. z96`). The black one and the white one are each painted with a circular pen (a round brush) with diameters of, respectively, 20 and 2; the grey one uses a more complex pen (marked here with black double arrows): its length and orientation depend on the position along the path.

One of the characteristics of METAFONT is that variables may be defined with geometrical equations. For example, the intent in a design that the three stems of an 'm' are equally spaced horizontally might be expressed as

$$x_2 - x_1 = x_3 - x_2$$

if points 1, 2, and 3 are at the bottom ends of the three stems; whereas the intent that they all end on the same vertical position would be

$$y_1 = y_2 = y_3.$$

The principal objects handled by METAFONT are the splines that the user may define with a `draw` instruction, and the points of the plane where the spline goes; see the example in figure 60.

Unlike almost all other digital font systems, METAFONT79 does not offer the user a way to describe the characters by their outlines but used only a pen metaphor for drawing glyphs: it assumes their definition via the ductus of a polygonal or elliptical pen, as done by calligraphers and the early printers. Figure 60 shows a nib (white line) which starts from point z90, goes straight to point z92 and arrives at point z96 after having drawn a Garamond-like e.

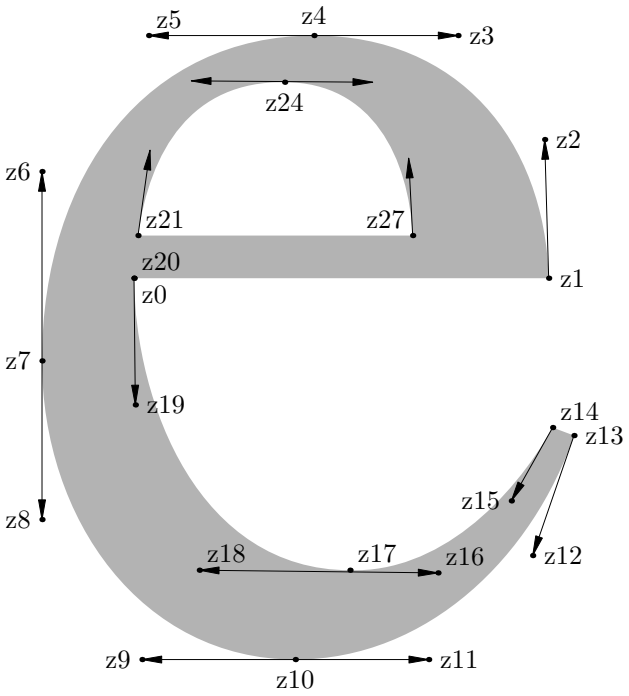
METAFONT79 allows defining curves more precisely, e.g. by defining angles at some points. Various kinds of pens are supported in METAFONT79: circular with various diameters (e.g. the white and black curves in figure 60), elliptical pens, and more general pens that have to be mathematically defined. Erasers are special pens that erase some part of a previous painted area.

The tools we have just mentioned (curves, pens) are those from the user's point of view. Internally, it was a different situation altogether. Knuth and his students used sophisticated mathematics to determine the curves finally drawn or painted. Let's summarize by saying they used polynomial curves, including cubics (Knuth does not use the word "Bézier" in [80] — probably because it was not fashionable at that time!).

8.3 METAFONT84

As we mentioned earlier, the people testing METAFONT79 found it was difficult to be used as a design tool by non-programmers, and Knuth completely redefined METAFONT [82], notably with the help of John Hobby [63]. Among the new added concepts, Bézier curves are now intensively used, both internally and from the user's point of view. Characters may be defined by their outlines (described with control points) and related instructions to fill the surface they define. This can be explicitly used by type designers as in PostScript (see below, page 49): figure 61 shows the same e of figure 60, but defined with Bézier control points (you may compare the syntax with that used by Ti*k*Z, `ctan.org/pkg/tikz`.)

However, METAFONT always focuses on the use of pens to draw characters, the creation of Computer Modern being the primary goal. Thanks to new procedures (`pickup, penstroke, penpos`, etc.), it is possible to define new types of pens, their local positioning, their paths, etc. The variation of the pens' marks, together with the use of parameters, makes it possible to draw a whole family of characters at once. For example, figure 62 shows that the arches of an 'n' are defined (without any reference to outlines) by variable pen positions, here `penpos` *i*.




```
z0=...; ...  z27=...;
fill z0--z1 ..controls z2 and z3 ..z4
     ..controls z5 and z6 ..z7
     ..controls z8 and z9 ..z10
     ..controls z11 and z12 ..z13--z14
     ..controls z15 and z16 ..z17
     ..controls z18 and z19 ..z20
     --cycle withcolor .7white;
unfill z21 ..controls z22 and z23 ..z24
     ..controls z25 and z26 ..z27--cycle;
```

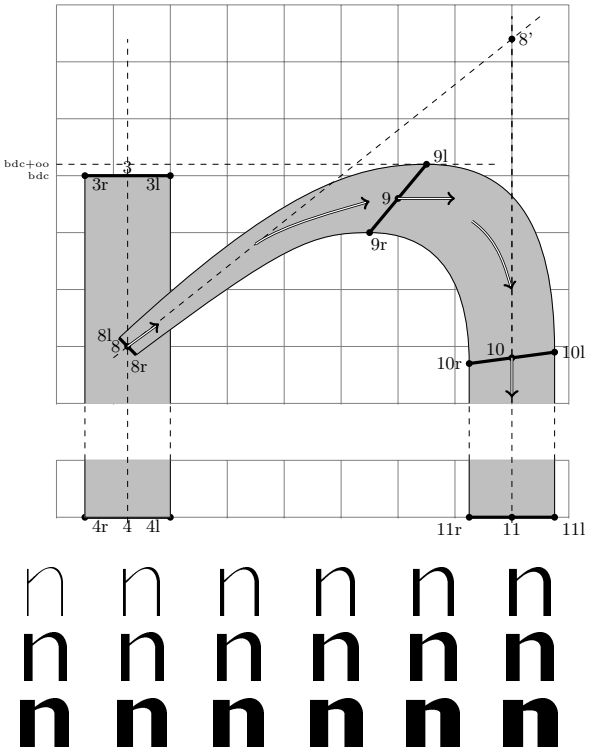
Figure 61: METAFONT84 allows painting a character from outlines described as Bézier curves. Compare with figure 60. (Labelled dots and tangents are added for convenience.)

More complex examples are in the final Computer Modern fonts (see section 8.4).

As a programming language, METAFONT offers many possibilities; let's just quote here the fact that "definitions" (also called "macros") allow, for example, making serifs compatible to each character of a font, like the incremental primitives of CSD (figure 48) or like, today, making a serif font with Glyphs using "corner components".

8.4 Computer Modern and others

The first large typeface family defined using METAFONT was Computer Modern (also called "cm"); the design was based on Monotype Modern 8A. It was created by Donald Knuth himself, with advice and assistance from Hermann Zapf, Charles Bigelow and Richard Southall. It was in fact the first "total




```
weight=(i/63);       % weight = function of i
loose=...
z3=...; z4=...;              % left stem ends
penpos4(weight,180);         % pen # 4
penstroke z3e..{down}z4e;    % stroke left stroke
z11=...;
penpos11(weight,180);              % pen 11
z8=...; penpos8(loose,angle(...)); % pen 8
y9l=bdc+oo; x9=.68[x8,x10];        % point 9
penpos9(.87[loose,weight],-136);   % pen 9
%
penstroke z8e..z9e..z10e---z11e; % arch and stem
```

Figure 62: METAFONT allows the construction of a typeface family based on the ductus alone. Above, the principle; below, a selection of 18 n's with different thicknesses and, below, an extract from the METAFONT program. After Haralambous [57], with kind permission.

```
cmchar "The letter e";
beginchar("e",7.25u#+max(.75u#,.5curve#),x_height#,0);
italcorr .5[bar_height#,x_height#]*slant+.5min(curve#-
1.5u#,0);
adjust_fit(if monospace: .25u#,.5u# else: 0,0 fi);
numeric left_curve,right_curve;
left_curve=right_curve+6stem_corr=curve if not serifs: -
3stem_corr fi;
...
path testpath; testpath=super_arc.r(2,3) & super_arc.r(3,4);
y1'r=y0r=y0l+.6[thin_join,vair]; y1'l=y0l; x1'l=x1'r=x1;
forsuffixes $=l,r:
 x0$=xpart(((0,y0$)--(x1,y0$)) intersectionpoint testpath);
endfor
fill stroke z0e--z1'e; % crossbar
penlabels(0,1,2,3,4,5); endchar;
```

Figure 63: Definition of the Computer Modern character e in METAFONT84 [84]. The variables such as `bar_height, x_height, monospace` are defined in a driver file, given values as desired for a particular font.

typography pack" [37], since it includes not only roman, italic and bold combined, but also variants of (real) small capitals, serif and sans serif typefaces, fixed width typefaces, and more. Not to mention a very large number of mathematical symbols [84].

All these typefaces have a family resemblance, and for good reason: they are defined by a single METAFONT program with many parameters. Knuth defined about sixty parameters (figure 64) to generate all these fonts; the entire family is completely described in a whole book, *Computer Modern Typefaces* [84]. Figure 63 shows a part of the definition of the model for the e's.

Although METAFONT is reputed to have been little used, hundreds of fonts have been created with it (an "incredible list" has been compiled by Luc Devroye [49]), especially for languages with non-Latin alphabets (Unicode did not exist for quite a few years after TEX and METAFONT), and in particular for ancient languages (including full accented Greek).

Richard Southall used METAFONT again to create Colorado (figure 65) by Ladislas Mandel [116]. This font, intended for the composition of telephone directories, required character to remain very readable even at very small body sizes.

8.5 METAFONT and type design

As we mentioned (section 8.2), many type designers have evaluated METAFONT, both during its development, and after. We previously discussed Hofstadter's answer [65] to Knuth's "Concept of a metafont" [82]; see also, for example, [32, 31]. Here are some highlights.

Family of fonts, parametrization One of METAFONT's strong ideas is to draw a whole family of fonts and not just one font. To express these instructions, parameters, etc., the designer must express them in the METAFONT language, which is in fact a programming language. And this is the problem

> Some type designers, such as Gerard Unger [122, 123], did not fail to say "Besides being a designer, I have no objection to acting as a system operator; but I don't want to become a programmer — let alone a parameterizer."

WYSIWYG or not Not all designers are ready to program, as they are used to working on character images and not on how to obtain these images. At a low level, let us say that it is easier for them to drag a dot on a screen and see what happens to some outline than to change the coordinates of this point in some program, experiment with the effect, running the program each

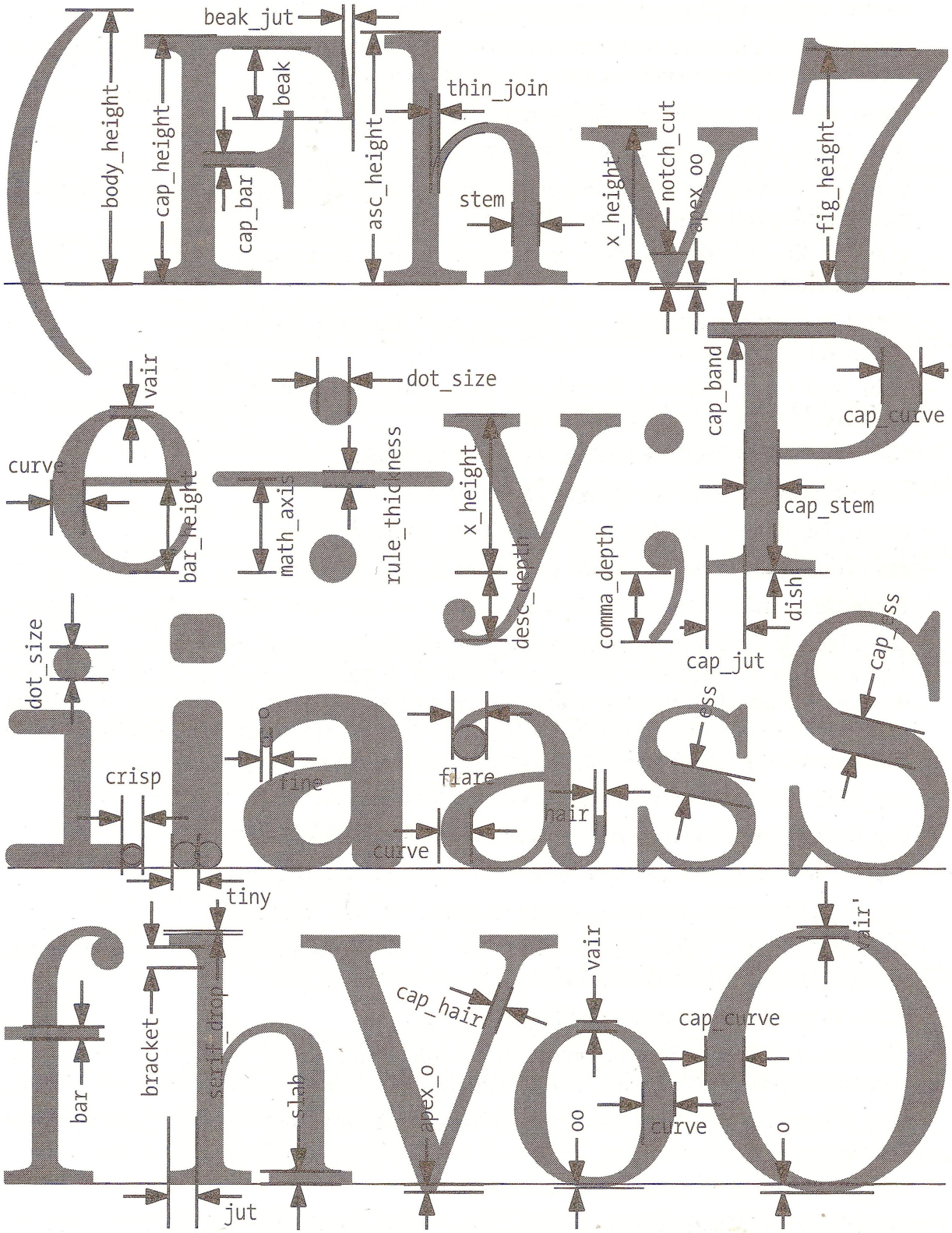


Figure 64: The 62 parameters that define Computer Modern, shown via selected characters [57].

Figure 65: Specimen of Colorado, a font for telephone directories by Mandel, designed with METAFONT by Richard Southall [117]. Real size; the first four lines are body size 6 pt, with 0.5 point leading. [Courtesy Kris Holmes]

time, until achieving the desired outline. Richard Southall [115, 117] and Dave Crossland [44] have studied this issue extensively.

Curves METAFONT84 uses Bézier curves, namely cubic splines. It seems Knuth considered these splines to be, mathematically, nicer than others (including conics). However some designers (at least ones used to Ikarus and later TrueType fonts) prefer to use quadratic splines as they provide (require) more points to control, and these points are closer to the expected outline, so more controllable. See figure 70.

Ductus model The ductus model (and the related concept of pen in METAFONT) is surely good for calligraphy or Oriental scripts based on separate strokes. However, this concept was largely abandoned for type design since the time of Aldus Manutius (around 1500), this abandonment being the precise differentiation between calligraphic writings and typographical ones. Since that time, typographers see types as surfaces, for which outlines are everywhere. Alas, the surfaces are less suitable for parameterizing, for example, the boldness of the arches.

9 Xerox PARC

The American company Xerox was founded in 1940 to exploit a new photocopying process, xerography, and quickly became the world leader in the very profitable market of photocopiers. Just before 1970, the company embarked on an emerging discipline, that of information technology applied to the "office of the future" (by "office", we mean not only the work of secretaries, but also the administration of companies and workshops and research centers with their technical drawings), a discipline that would come to be called Office Automation.

At the end of the 1960s, we were still in the era of heavy computing, the computer market being held by IBM, Bull-Honeywell, Control Data, etc. Minicomputers were also beginning to appear (notably the PDP series from Digital Equipment Corporation), especially in the industrial world, and systems such as Unix (1971), primarily in the academic and research world. But, discreetly, a very different kind of computing was born, whose pioneers included Vannevar Bush, who had the first conception of hypertext; Douglas Engelbart, inventor of many of the modern concepts in the human–machine interface, including the mouse; without forgetting the American military, which funded significant research, including the ARPAnet, from which the Internet developed.

In 1970, Xerox created a research center in addition to its headquarters in Rochester (New York), with a strong focus on physics and chemistry, the Xerox Palo Alto Research Center (PARC; for its history, see [88] and [99]), located in California in what would later be called Silicon Valley. The mission of Xerox PARC is simple, at least in its statement: "Invent the office of the future."

9.1 Alto

Thanks to the proficiency of the researchers (many from Stanford and Berkeley), successes came very quickly: invention of the concept of the personal computer, and a prototype (Alto) with a screen and user interface with windows, icons, etc., manipulated by a mouse (it had been invented some years earlier, but this was the first use of it), invention of prototype printers, xerographic, chemical (like a modified Versatec) and laser; all were bitmapped, with resolutions between 300 and 400 dpi.

For the Alto, PARC developed a large variety of software, mostly office tools. These include: Bravo, the first WYSIWYG (What You See Is What You Get) text editor; drawing software including Draw, which allowed curves (actually cubic splines), hand drawings, and text elements to be integrated into a figure; Press, a "universal" (or portable, i.e. printer-independent) page description language (PDL), developed in 1975 for a complete description of documents (text and figures). This last would be followed by

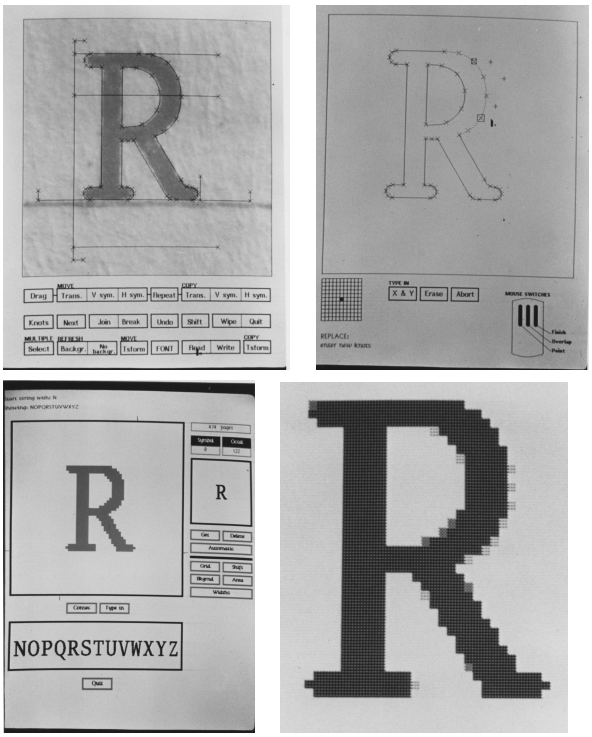


Figure 66: Screen images of the creation of fonts on the Alto (around 1975). With Fred: placing control points on a scan and improvements; with Prepress: a filled and improved bitmap (shades of gray indicate corrections, additions or deletions of pixels). [Courtesy Patrick Baudelaire [19] and Amelia Hugill-Fontanel (RIT Cary Graphic Art Collection)]

InterPress by Charles Geschke and John Warnock in 1980, and both would serve as a model for PostScript (see page 49).

9.2 The Alto font model

All Alto software used the same font model, developed by Bob Sproull and Patrick Baudelaire [20, 118]. The font creation system (figure 66) included an interactive spline editor and a filling program [19]. The first one, Fred, was written by Patrick Baudelaire (who had already designed the graphic language Draw, with procedures to draw splines). Fred projected onto a screen the image of a character to be digitized and, thanks to the mouse, one could draw the control points using B-splines (figure 66).

This Alto system was never commercialized, but was frequently presented and published in conferences and expert meetings and thus served as a catalyst. The Alto spawned many important successors, two of which are especially noteworthy:

- In 1978 Apple launched its personal computer project that would lead to the Macintosh (see page 33). In 1979, Steve Jobs bought from Xerox the right to exploit the research done at PARC, and Apple would benefit greatly from these revolutionary concepts.
- In 1982, Charles Geschke and John Warnock, two members of the PARC team (where they had developed InterPress based on Fred+Prepress), created Adobe, a company specialized in electronic publishing.

9.3 Birth of laser printers

Xerox was a leader in photocopying, and by 1972 had produced the first raster printer (XGP, the Xerox Graphics Printer, had a resolution of 192 dpi) to gain substantial use by computer scientists (at Carnegie-Mellon, Stanford, MIT, Caltech, and the University of Toronto).

Another PARC team worked on laser printers, led by Gary Starkweather. The first laser printer, EARS, was built and used with the Alto in 1973–76. In 1976 came the Dover printer, and in 1977 a color prototype. The first commercial product resulting from the work of Xerox PARC was the Xerox 9700 laser printer, inspired by the EARS prototype for the laser technology (at 300 dpi). It was distributed starting in 1977 and was a huge success worldwide.

IBM (which had been working on the problem of replacing impact printers since the 1960s) released an equivalent machine, the IBM 3800, in 1976. The Japanese company Canon also tackled the problem in the early 1970s and joined forces with Hewlett-Packard to produce a "big" laser printer, the HP 2680. It was not until 1983 that the first desktop printer, Canon's 300 dpi LBP-CX, was marketed by HP as the HP LaserJet. The same LBP-CX engine would be used in the Apple LaserWriter, the first commercial printer supporting PostScript.

10 Dissemination of the digital fonts concept

Through the end of the 1970s, research and development (technical and commercial) on digital typography was mainly carried out by small companies and university research laboratories. After 1980, thanks to laser printers, the concept of digital fonts spread at a very high speed and we have seen the birth and growth of foundries and development companies to the point where we are sometimes convinced that digital fonts were invented by Apple, Adobe, etc. It should be remembered that all the research, both earlier and today, could only be spread thanks to the community of researchers, scientific conferences and publications.

Typesetting with photocomposers was followed in particular by the ATypI conferences, but also revealed by experts like John W. Seybold, who created a consulting company for the graphic industries in 1963. In 1971, he launched, with his son Jonathan, the bi-monthly magazine *The Seybold Report* which remained for many years the canonical reference magazine for typesetting developments.

On the other hand, knowledge of digital fonts (apart from photocomposers) spread, initially, more in university circles and private research laboratories. This dissemination was made energetically and enthusiastically thanks to people interested and capable in two skills, at first sight mismatched: typography and data processing. The first attempts probably have been the Stanford classes, workshop and conferences organized at the end of the 1970s by Donald Knuth and his team around METAFONT (see page 42).

Let us quote too, in general: the ACM's Special Interest Group on Computer Graphics (SIGGRAPH), the magazine *Visible Language* (with articles by Vargo [100], Unger [122], and many others), the proceedings of the August 1983 conference of the ATypI at Stanford University [32, 31], the publication in 1983 of a "popular science" article by Bigelow (typographer) and Day (specialist in computer image processing) [28], the conferences *Electronic Publishing* and *Raster Imaging and Digital Typography* [109], and much more. All these communications are worthwhile above all because of the contacts that were made possible. Let us not forget also the publication of several books accessible by both communities, typographers and scientists, such as those by Seybold [114], Rubinstein [111], Alison Black [33] then Jorge De Buen [46], and Yannis Haralambous [57]. Finally, it is also worth mentioning magazines specialized in typography (*U&lc*, *Eye*, *PRINT*), etc. or not (like *TUGboat*!).

11 Adobe and PostScript

As we have seen, John Warnock and Charles Geschke worked at Xerox PARC and developed the InterPress page description language there. Because of the lack of interest by Xerox to commercially develop its revolutionary products, they left Xerox and founded Adobe in 1982.

There, Warnock and Geschke started to develop PostScript and looked for a desktop printer to market this language. For his part, Steve Jobs, at Apple, was looking to replace the ImageWriter (a dot matrix printer) of the first Macintosh, and discovered the LBP-CX Canon printer. Jobs then convinced Warnock to license PostScript to Apple for the LaserWriter, which would be marketed by Apple. A third partner then intervenes: Jonathan Seybold (son of John W. Seybold) had introduced to Apple another former PARC staff member, Paul Brainerd, who then founded the Aldus company, which developed PageMaker, and was another early PostScript licensee. It was the beginning of the success of PostScript and the success of Adobe.

In addition, Seybold's involvement had implications for the world of photocomposition — Aldus also entered this market. Linotype would be the first typesetter company to discover the importance of PostScript, and in 1984 released a PostScript raster image processor on its Linotronic 101 photocomposers, and then on the following model, the Linotronic 300. These photocomposers with 1270, 2540 and even 3300 dpi showed the very high quality of the PostScript fonts, which one could only imagine until then because of the relatively low 300 dpi resolution of the LaserWriter.

PostScript, conceived at the beginning for office automation, thus became the universal language (at least a portable language) in the world of pre-press and traditional printing.

Along with the commercial dissemination of PostScript — the customers being OEM (*Original Equipment Manufacturer*) companies in data processing and in particular the manufacturers of printers, photocomposers and computers — Adobe made efforts to spread knowledge of PostScript programming, which went far to expand its use. The best known is the publication of three manuals, respectively red (a reference manual), blue (a tutorial book of "recipes") and green (more oriented toward documents); see [7] to [10]. A fourth book (black, on the Type 1 format) would be published during the font wars.

We also find efforts to spread knowledge of PostScript in the booklets, sometimes luxurious, distributed by the branches like Adobe-France, in particular with font specimens.

11.1 The PostScript language

Based on graphic languages like Draw from PARC, and even more on InterPress, PostScript is a page markup language and not a document formatting language: it is up to the formatter to compose text, to hyphenate a word at the end of a line, etc. PostScript is designed to be the output language of typesetting programs; it is therefore analogous, in a general way, to the DVI language output by the original TEX.

Graphics supported in PostScript include line drawings, formed by line segments and Bézier curves (also supported in PARC's Fred software). Characters are only drawings, so in PostScript they become

```
/HH 100 def /H HH 2 div def
newpath
0 H moveto HH  0 rlineto 0 HH rlineto
HH neg 0 rlineto
closepath
.5 setgray fill % gray
HH HH moveto
H neg  H rlineto
H neg H 0 HH H H  rcurveto
H H HH 0 H H neg rcurveto closepath
1 0 0 setrgbcolor fill % red
0 setgray              % black
/Helvetica findfont 50 scalefont setfont
25 HH moveto (TYPO) show
```

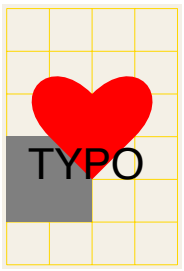


Figure 67: Example (typical around 1985) of PostScript program using Bézier curves (by the instruction `rcurveto`) and its result (the yellow grid, with a step of 50 points, is added).

procedures (routines) to draw them. A font is a dictionary of procedures for drawing characters, and their use is limited to two groups of operations: the selection of a font (with global metric properties and geometric properties depending on the graphic state) and the writing of a sequence of characters in this font. This simplicity of definition hides, however, a whole typographic machinery to which we will return.

The PostScript machinery. While PostScript is innovative in term of structures and possibilities, its most important feature is undoubtedly the way in which it functions. Until now, the drawing or font software computed bitmaps and sent them to the printer, but PostScript is more incremental [7]: the generation of bitmaps is done *inside* the printer; or, in front of the photocomposers, in a device called a *Raster Image Processor* (RIP).

Without going into details, we give two examples of PostScript programs. The first, figure 10, shows a letter 'P' with line segments used to approximate the curve of the bowl; we can observe that the instructions there resemble those of plotters and screens (figure 3). The other, figure 67, uses more concepts.

11.2 Type 3 and PostScript fonts

The font model defined using only the PostScript language [7] is what is now called *Type 3*. But in 1984, this number (3) only corresponded to a completely general method of generating characters, unlike the later-known Type 1 format, a less general method but additionally provided with hinting procedures.

PostScript characters are thus procedures (programs) describing their contours using line segments and Bézier curves. Their definition is independent of any particular size: PostScript uses geometric scaling, i.e. computes the coordinates scaled according to the desired body height, purely mathematically.

The PostScript font machinery is a special case of the general PostScript machinery, see [7] and [14]. It is important to remember that bitmaps are computed at the time of use; however, a caching mechanism allows these computations to be done only once per page for a given character, in a given font, at a given body height, etc. (unless this mechanism is disabled, see below).

Glyphs and encoding tables. Internally, a PostScript font defines its characters by their name: the character 'e' corresponds a PostScript procedure named `/e`. In fact, a PostScript font mainly consists of glyph-drawing procedures in the sense well-known today, but practically unknown then, in the 1980s. At the time of selecting a font, the PostScript interpreter builds an access table to at most 256 characters of the font, and an encoding vector to access this table, similar to character access via the `inputenc` package of LaTeX (in the years before Unicode). Also, just as TeX uses TFM (*TeX Font Metric*) information for typesetting, Adobe defined AFM (A for Adobe) metric information to accompany PostScript fonts.

With PostScript level 2, in 1991, it would become possible to call a character by its name (e.g. '`/ffi glyphshow`') even when it is not in the 256 character table. The list of these glyphs gradually became a standard for numeric fonts (via the so-called *glyphlist* file) and equivalences with Unicode names were made shortly before 2000.

Dynamic fonts. PostScript, like METAFONT, allows variables to be used in writing plot instructions, including characters. If these variables are given, for example, random values in METAFONT, since the bitmaps are calculated once when generating the fonts, before any final print, these values are not re-evaluated. A typical example is the Punk font [85] where each occurrence of 'E' is the same, though its form was generated randomly by METAFONT. That is, if we ran METAFONT again before typesetting the text, the 'E' would be different (figure 68, top).

By default, it is the same with PostScript since a cache mechanism avoids recalculating the bitmaps, for efficiency. However, with Type 3 fonts, the cache mechanism can be disabled. As a result, using a Type 3 version of Punk [103], we can get "real" random characters (before Beowolf by Erik van Blokland and Just van Rossum, for example) as shown in figure 68, bottom.

Beyond this playful aspect, this mechanism allows generating characters depending on the context (body, neighborhood, ...) or on, say, the time of day. However, few fonts have been developed in Type 3

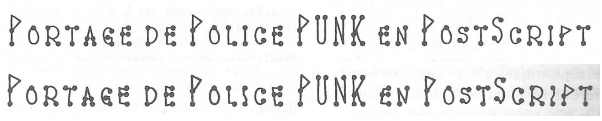


Figure 68: The same text written with Knuth's punk font: above with METAFONT [85], below in a PostScript Type 3 version [103]. In the first line, all E's are identical, which is not the case in the second.

format (although see [48, Type 3 Software]), and even fewer commercialized. The model was too general and required training to be able to interact with the font. Thirty years later, this concept has recently been rediscovered under the name of *variable fonts* and is very fashionable.

11.3 Type 1 fonts

Those who used PostScript at its beginning with the Type 3 font model with its cache mechanism thought that Adobe had hidden a "magic bullet" for fonts, and they were not wrong. Adobe kept more or less jealously secret a font model allowing faster and better rendering: Type 1. It was provided to certain foundries, under restrictive conditions, only after March 1985, and was not made public until 1990, with the publication of the "black book" at the time of the font wars [10].

Type 1 PostScript fonts work in essentially the same way as Type 3 fonts, but differ from Type 3 in that they are more professional or commercial by nature and, above all, by the possibility of programmable hinting.

Professional aspects. Although Adobe did not provide any typeface production software, third party software, such as Fontographer, saved designers from programming in raw PostScript. Type 1 fonts use the same Bézier curve contour description procedures as Type 3, and PostScript in general, but have a few other lower-level procedures that are particularly well suited to typography (vertical stems, horizontal lines, junctions, etc.) and are much more efficient than the general *lineto* and *curveto* operators of Type 3.

Type 1 fonts also offer a series of controls related to font secrecy (copyright, identification number) etc., and elaborate encryption methods! The encryption algorithm was disclosed by Adobe in the Type 1 book.

Hinting. More important typographically, hinting instructions in Type 1 allow giving instructions to the bitmap rendering procedures to improve the final drawing of the characters (see earlier discussion and examples, page 41). We have seen that Ikarus proposed them as early as 1983 (figure 56). Earlier,

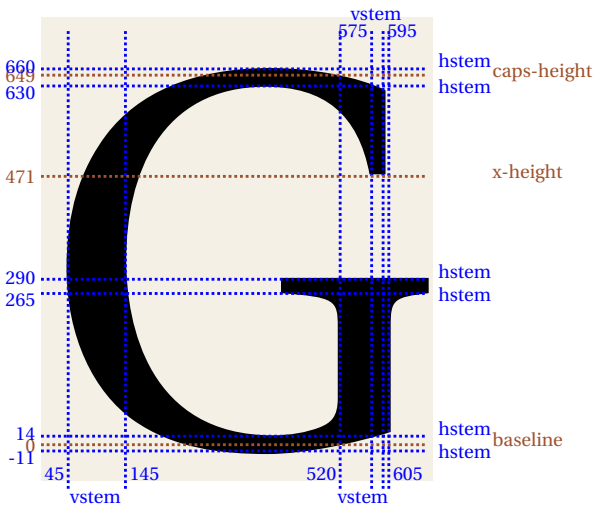


Figure 69: Hinting in Type 1 PostScript fonts is done through "blue lines". Here they define various horizontal (`hstem`) and vertical (`vstem`) thicknesses. Following [10].

Fred and METAFONT did not need explicit hinting because they generated the bitmaps themselves.

The Type 1 hinting operators are complicated. and are intended primarily to help render fonts at relatively high (printer) resolutions, not for screens. The method chosen by Adobe was that of so-called "blue lines", entered by the designer to specify constraints (figure 69).

11.4 The Adobe font library

As mentioned above, Adobe did not initially offer any tools for writing fonts. Instead, Adobe entered, if not revolutionized, the font market by digitizing a huge number of fonts and implementing them in PostScript. According to Peter Karow [71, p. 269], the first 250 outline-based fonts distributed by Adobe were purchased from URW.

12 As a matter of conclusion

With the advent of PostScript and laser printers, the prehistory of the digital fonts ends. Let us say simply in a few words what happened next ...

- The widespread adoption of PostScript with its cubic Bézier curves, including for printing, had immediate consequences for other software in the field, which adapted to the current tastes. For example, TEX quickly supported PostScript with a new DVI conversion program, `dvips`; METAFONT itself was not changed, but a companion program METAPOST [64] was developed to output encapsulated PostScript instead of bitmaps;

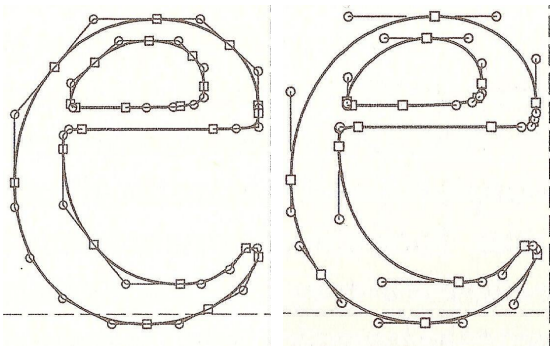


Figure 70: The same 'e' implemented, on the left, in TrueType (quadratic splines) and, on the right, in Type 1 (cubic splines). After Bringhurst [40, p. 184].

in addition, programs such as Metafog were developed to extract the curves from inside METAFONT and output Type 1 or Type 3 fonts. Troff also had PostScript output early. Ikarus kept its IK format but made a BE version where the original conics are translated into cubic Béziers (the translation is straightforward).

Other conic spline formats were tried. Let us mention in particular the F3 format of Vaughan Pratt for Sun [106, pp. 144, 331].

- But Adobe kept to itself the rights to the Type 1 format and machinery, so other manufacturers, notably Apple, later joined by Microsoft, developed an alternative. This led to the birth of the TrueType format and to the release by Adobe to the public of the Type 1 format, in 1989. Each model, Type 1 and TrueType, has its own partisans defending the superiority of their hints or their particular splines, according to the needs of type designers (figure 70 has a comparison).
- The various font formats and new encodings, in particular Unicode, provided for portability of fonts, and their use for languages with different character sets.
- In the late 1970s, screens were developed with pixels that are not black or white, but grayscale. At the end of the 1990s, liquid crystal displays appeared where pixels could be divided into subpixels. In the 2000s, three sub-pixel renderers were in use: Adobe's CoolType, Apple's ATSUI (*Type Services for Unicode Imaging*, using the Quartz2D engine), and Microsoft's ClearType. All have been significantly enhanced and/or replaced in the years since.
- A whole collection of font creation systems were developed (FontForge, Fontlab, Fontographer, FontStudio, Glyphs, etc.). Many of them were developed by individuals (such as Von Ehr, Yuri Yarmola, and George Williams) before being taken over by larger companies.
- The most important point of this period is what is called "the font wars" opposing TrueType and Type 1 supporters (see Bigelow [27]). Although the background was technical (choice of conics or cubics and especially method of hinting), it played out mainly with commercial connotations. TrueType and Type 1 eventually converged with OpenType (1993).
- At the time of the original "movable types", a cast was a set of classified types, possibly with the mechanical composition (e.g., the Monotype machines). With the second generation phototypesetters we see appear side information: the width tables. These then also existed for digital Hershey fonts. In the Ikarus formats, TEX and METAFONT, and PostScript fonts accompanied these width tables with information on ligatures and kerning (TFM and AFM respectively). OpenType has taken over and considerably increased this mechanism of side tables (`gpos`, etc.). Its strength is thus based on the experience of all the preceding work.
- It seems to us personally that the evolution of the future fonts will be based on the side tables by increasing their content (in particular by the use of variables) but also by using these tables not just at the time of their loading, but also during composition (these will then be real variable fonts).

Finally, we would like to point out that all the tools (except tentatively METAFONT) that we have shown are more manufacturing tools — the drawing of a character that already exists, even if only in the form of a sketch, than creation (from scratch). As yet, we have no answer to the philosophical questions of Douglas Hofstadter (What is the essence of a-ness?) or Richard Southall (Are the shape and the appearance of a character identical?).

By way of final words, I'd like to conclude with an homage to Southall by quoting these words by Gerry Leonidas [89]:

> Richard's ideas about "models" and "patterns" in type design are the definitive starting point for any discussion of typemaking, and — with some adjustments for terminology — absolutely essential in any review of typeface design processes with digital tools. In fact, the growth of rendered instances of typeforms across many devices make his ideas more relevant than ever, and prove that his approach

provides the key ideas for discussing typeface design across type-making technologies. Together with some texts by Robin Kinross, his writings [for a list, see [16]], are amongst the very few indispensable texts for any theoretical discussion of typeface design.

Acknowledgments

The author would like to thank the many people who provided illustrations for this article, and also the people who helped him in the drafting of this text and then its revision, in particular Patrick Baudelaire, Charles Bigelow, Yannis Haralambous, Roger Hersch, Vania Joloboff, Peter Karow, Christian Laucou, and Alan Marshall.

For this publication in *TUGboat*, he would also like to thank Patrick Bideault for his English translation, and Barbara Beeton and Karl Berry for their deep proofreading and remarkable work as editors.

References

[1] *Histoire de l'écriture typographique – De Gutenberg au XVII^e siècle*, by Yves Perrousseaux. Atelier Perrousseaux éd./Adverbum, 2004.

[2] *Histoire de l'écriture typographique – Le XVIII^e siècle*, tome I/II, by Yves Perrousseaux. Atelier Perrousseaux éd./Adverbum, 2010.

[3] *Histoire de l'écriture typographique – Le XVIII^e siècle*, tome II/II, by Yves Perrousseaux. Atelier Perrousseaux éd./Adverbum, 2010.

[4] *Histoire de l'écriture typographique – Le XIX^e siècle français*, by Jacques André and Christian Laucou. Atelier Perrousseaux éd./Adverbum, 2013.

[5] *Histoire de l'écriture typographique – Le XX^e siècle, de 1900 à 1950*, collective work under the direction of Jacques André. Atelier Perrousseaux éd./Adverbum, 2016.

[6] *Histoire de l'écriture typographique – Le XX^e siècle, de 1950 à 2000*, collective work under the direction of Jacques André. Atelier Perrousseaux éd./Adverbum, 2016.

[7] Adobe Systems Inc., *PostScript Language Reference Manual*, first edition, Reading, MA: Addison-Wesley, 1985; second edition, 1991.

[8] Adobe Systems Inc., *PostScript Language Tutorial and Cookbook*, Reading, MA: Addison-Wesley, 1985.

[9] Adobe Systems Inc., *PostScript Language Program Design*, Reading, MA: Addison-Wesley, 1988.

[10] Adobe Systems Inc., *The Type 1 Format Specification*, Reading, MA: Addison-Wesley, 1990.

[11] Jacques André, *Création de fontes en typographie numérique*, Mémoire d'HDR, Université Rennes I, 29 sept. 1993, 124 pp. theses.hal.science/tel-00011218/file/andre.pdf

[12] Jacques André, *Courier – Histoire d'un caractère – De la machine à écrire aux fontes numériques*, éd. du Jobet, 1993. jacques-andre.fr/fontex/courier.pdf

[13] Jacques André, De Pacioli à Truchet : trois siècles de géométrie pour les caractères, *4 000 ans d'histoire des mathématiques : les mathématiques dans la longue durée – 13^e colloque Inter-IREM d'épistémologie et histoire des mathématiques*, IREM-Rennes, mai 2000, pp. 1–38. hal.inria.fr/inria-00000956

[14] Jacques André and Justin Bur, Métrique des fontes PostScript, *Cahiers Gutenberg*, n° 8 (1991), pp. 29–50. http://numdam.org/item/CG_1991___8_29_0/

[15] Jacques André and Denis Girou, Father Truchet, the typographic point, the Romain du roi, and tilings, *TUGboat*, Vol. 20 (1999), No. 1, pp. 8–14. tug.org/TUGboat/tb20-1/tb62andr.pdf

[16] Jacques André and Alan Marshall, Richard Southall: 1937–2015, *TUGboat*, Vol. 36 (2015), No. 2, pp. 100–102. tug.org/tugboat/tb36-2/tb113southall.pdf

[17] Jacques André and Moncef Mlouka (eds.), *Workshop on Font Design Systems*, INRIA-Sophia, May 1987. See also [109], 1989.

[18] Augustin, *Wim Crouwel*, Index Grafik, 7 avril 2014. http://indexgrafik.fr/wim-crouwel/

[19] Patrick Baudelaire, *The Fred User's Manual*, Internal Report, Xerox Palo Alto Research Center, Palo Alto, California, 1976.

[20] Patrick Baudelaire, The Xerox Alto Font Design System, in [31].

[21] Patrick Baudelaire and M. Stone, Techniques for Interactive Raster Graphics, SIGGRAPH 80 Proceedings, *Computer Graphics*, Vol. 14, No. 3, 1980.

[22] Barbara Beeton, Karl Berry and David Walden, TEX: A Branch in Desktop Publishing Evolution, Part 1, *IEEE Annals of the History of Computing*, Vol. 40, No. 3, Jul./Sept. 2018, pp. 78–93. ieeexplore.ieee.org/document/8509554/

[23] Yves Bekkers, Daniel Herman, and Michel Raynal, *Conception et réalisation d'une machine-langage de haut niveau adaptée à l'écriture de systèmes*, Ph.D. thesis, Rennes University, 24 sept. 1975.

[24] Charles Bigelow, Les caractères rationalisés, in *La manipulation de documents* (Jacques André, ed.), INRIA-Centre de Rennes, mai 1983, pp. 15-27. jacques-andre.fr/japublis/manip83.pdf

[25] Charles Bigelow, Review and Summaries of *The History of Typographic Writing — The 20th century*. Originally published in three parts in *TUGboat* Vol. 38 (2017); combined: tug.org/books/reviews/tv38bigelow.pdf

[26] Charles Bigelow, Typeface Features and Legibility Research, *Vision Research*, Vol. 165, Dec. 2019, pp. 162–172. doi.org/10.1016/j.visres.2019.05.003

[27] Charles Bigelow, The Font Wars, Parts 1 and 2, *IEEE Annals of the History of Computing* (Special issue: History of Desktop Publishing), Vol. 42, No. 1, Jan./Mar. 2020, pp. 7–40. doi.org/10.1109/MAHC.2020.2971202 doi.org/10.1109/MAHC.2020.2971745

[28] Charles Bigelow and Donald Day, Digital Typography, *Scientific American*, Vol. 249, No. 2, pp. 106–119, Aug. 1983.

[29] Charles Bigelow and Kris Holmes, Notes on Apple 4 Fonts, *Electronic Publishing*, Vol. 4, No. 3, Sept. 1991, pp. 171–181. http://cajun.cs.nott.ac.uk/compsci/epo/papers/volume4/issue3/ep050cb.pdf

[30] Charles Bigelow and Kris Holmes, The Design of Lucida: An Integrated Family of Types for Electronic Literacy, in *Text Processing and Document Manipulation* (J.C. van Vliet, ed.), Cambridge University Press, 1986.

[31] Charles Bigelow and Kevin Larson (eds.), *Visible Language* (Special issue: Reflecting on 50 Years of Typography, Vol. 50, No. 2, Aug. 2016. journals.uc.edu/index.php/vl/issue/view/461

[32] Charles Bigelow and Lynn Ruggles (eds.), *Visible Language* (Special issue: The Computer and the Hand in Type Design), Vol. 19, No. 1, Winter 1985. journals.uc.edu/index.php/vl/issue/view/369

[33] Alison Black, *Typefaces for Desktop Publishing: A User Guide*, London: Architecture Design and Technology Press, 1990.

[34] Lewis Blackwell, *20th-Century Types*, Lawrence King Publishing, 2004.

[35] Gérard Blanchard (coordinated by), *L'écriture télématique, années zéro*, Les Cahiers de Lure, 1985.

[36] Gérard Blanchard, *L'eredita Gutenberg*, Gianfranco Altieri Editore, 1989.

[37] Gérard Blanchard, *Aide au choix de la typo-graphie – Cours supérieur*, Atelier Perrousseaux éd., 1998.

[38] Paul Bourke, *Hershey Vector Font based on the Hershey character set*, Oct. 1977. http://paulbourke.net/dataformats/hershey/

[39] Jack E. Bresenham, Algorithm for computer control of a digital plotter, *IBM Systems Journal*, Vol. 4, No. 1, Jan. 1965, pp. 25–30. doi.org/10.1147/sj.41.0025

[40] Robert Bringhurst, *The Elements of Typographic Style*, Hartley & Marks publishers, 4th edition, 2015.

[41] *CalComp Software Reference Manual*, California Computer Products Inc., Oct. 1976. archive.org/details/bitsavers_calcompCalceManualOct76_6872751

[42] Edward M. Catich, *The Origin of the Serif: Brush Writing and Roman Letters*, Davenport, IA: The Catfish Press, 1968.

[43] Philippe J.M. Coueignoux, *Generation of Roman Printed Fonts*, Ph.D. thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1975. dspace.mit.edu/bitstream/handle/1721.1/27408/02149218-MIT.pdf

[44] Dave Crossland, "Why didn't METAFONT catch on?", *TUGboat*, Vol. 29 (2008), No. 3, pp. 418-420. tug.org/TUGboat/tb29-3/tb93crossland.pdf

[45] *Typographic Architectures typographiques*, texts by Wim Hendrik Crouwel, Catherine de Smet and Emmanuel Bérard, Editions fsept F7, Paris, 2007.

[46] Jorge De Buen, *Manual de diseño editorial*, Santilano, Mexico, 2000,

[47] Christian Delorme and Jacques André, Le Delorme, un caractère modulaire et dépendant du contexte, *Communication et langages*, Vol. 86 (1990), pp. 64–76. www.persee.fr/web/revues/home/prescript/article/colan_0336-1500_1990_num_86_1_2261

[48] Jean-Luc Devroye, *Type Design, Typography, Typefaces and Fonts: An encyclopedic treatment of type design, typefaces and fonts*. Web page closed on May 6, 2022. http://luc.devroye.org/fonts.html

[49] Jean-Luc Devroye, METAFONT links, in [48]. http://luc.devroye.org/metafont.html

[50] Diderot & D'Alembert, *Encyclopédie, ou Dictionnaire Raisonné des Sciences, des Arts et des Métiers*, 1751–1772. encyclopedie.uchicago.edu

[51] Albrecht Dürer, Of the just shaping of letters. www.zigzaganimal.be/elements/just_shaping_scan.pdf

[52] Jean-Jacques Eltgen, Techniques d'impression d'images numérisées, *Techniques de l'ingénieur*, art. E-5-670, 1992.

[53] James Essinger, *Jacquard's Web: How a hand loom led to the birth of the information age*. Oxford, U.K.: Oxford University Press, 2004.

[54] Bernard Ficatier and Hugues Roche, *Concevoir, relever et dessiner des plans de voiliers classiques et traditionnels*, Douarnenez: Chasse-marée, 2004.

[55] Richard Furuta, Jeffrey Scofield, and Alan Shaw, Document Formatting Systems: Survey, Concepts, and Issues, *Computing Surveys*, Vol. 14, No. 3, Sept. 1982, pp. 417–472. doi.org/10.1145/356887.356891

[56] Pierre-Marie Gallois, *Quand Paris Était Ville-Lumière*, L'Âge D'homme, 2001.

[57] Yannis Haralambous, *Fonts & Encodings*, O'Reilly, 2007.

[58] Tamir Hassan, Changyuan Hu, and Roger D. Hersch, Next Generation Typeface Representations: Revisiting Parametric Fonts, *ACM DocEng 2010 conference*, Sept. 2010, pp. 181–184. lspwww.epfl.ch/publications/typography/ngtrrpf_10.pdf

[59] Rudolf Hell, Le Digiset, composeuse binaire électronique, *Caractère*, vol. 16, nº 11, 1965, pp. 5–16.

[60] Roger Hersch (ed.), *The Visual and Technical Aspects of Type*, Cambridge University Press, 1993. lspwww.epfl.ch/publications/typography/vataot.html

[61] Allen V. Hershey, *Calligraphy for Computers*, U.S. Naval Weapons Laboratory, 1967, 500pp. archive.org/details/hershey-calligraphy_for_computers

[62] Allen V. Hershey, A Computer System for Scientific Typography, *Computer Graphics and Image Processing*, Vol. 1 (1972), pp. 273–385.

[63] John D. Hobby, *Digitized Brush Trajectories*, Ph.D. thesis, Stanford University, Aug. 1985. tug.org/docs/hobby/hobby-thesis.pdf

[64] John D. Hobby, *A User's Manual for METAPOST*. AT&T Bell Laboratories Computing Science Technical Report 162, 1992. With updates: tug.org/docs/metapost/mpman.pdf

[65] Douglas Hofstadter, *Metamagical Themas*, Basic Books, 1985. archive.org/details/MetamagicalThemas

[66] Kris Holmes, *Dossier — Calligraphy, Lettering, Signage and Graphic Design, Filmmaking and Articles*, Keepsake for the Frederic Goudy Award, Rochester Institute of Technology, 2012.

[67] Changyuan Hu and Roger D. Hersch, Parameterizable Fonts Based on Shape Components, *IEEE Computer Graphics and Applications*, Vol. 21, No. 3, May/June 2001, pp. 70–85. lspwww.epfl.ch/publications/typography/pfbosc.pdf

[68] Peter Karow, *Digital Formats for Typefaces*, URW Verlag, Hamburg, 1987. archive.org/details/digitalformatsfo0000karo

[69] Peter Karow, Digital punch cutting, *Electronic Publishing*, Vol. 4, No. 3, Sept. 1991, pp. 151–170. http://cajun.cs.nott.ac.uk/compsci/epo/papers/volume4/issue3/ep044pk.pdf

[70] Peter Karow, *Digital Typefaces: Description and Fprmats*, Springer-Verlag, 1994. books.google.com/books?id=oomrCAAAQBAJ

[71] Peter Karow, Two decades of typographic research at URW: A retrospective, in [109, pp. 265–304 (1998)].

[72] Peter Karow, *Font Technology: Methods and Tools*, Springer Science, 2012.

[73] Peter Karow, Digital Typography & Artificial Intelligence, On the occasion of the presentation of the third *Dr. Peter Karow Award for Font Technology & Digital Typography* to Dr. Donald E. Knuth at the ATypI Amsterdam 2013 conference, Adobe and Dutch Type Library, 2013.

[74] Brian W. Kernighan, *A Typesetter-independent TROFF*, Computing Science Technical Report No. 97, Bell Labs, revised, Mar. 1982. archive.org/details/typesetter-independent-troff

[75] Brian W. Kernighan, PIC — a language for typesetting graphics, *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, June 1981, pp. 92–98. doi.org/10.1145/800209.806459 Revised publication, May 1991: archive.org/details/pic-graphics-language

[76] Brian W. Kernighan and Lorinda L. Cherry, A System for Typesetting Mathematics, *Communications of the ACM*, Vol. 18, No. 3, Mar. 1975, pp. 151–157. dl.acm.org/doi/10.1145/360680.360684

[77] Christopher Knoth, *Computed Type Design*, Master Art Direction, ECAL Lausanne, 2011. christoph-knoth.com/shared/computed_type_-_christoph_knoth.pdf

[78] Donald E. Knuth, "TAU EPSILON CHI: A System for Technical Text", STAN-CS-78-675.1, Computer Science Department, Stanford University, Stanford, California, Nov. 1978. purl.stanford.edu/jy605yq4819

[79] Donald E. Knuth, Mathematical Typography, *Bulletin (N.S.) of the American Mathematical Society*, Vol. 1, No. 2, 1979, pp. 337–372. doi.org/10.1090/S0273-0979-1979-14598-1

[80] Donald E. Knuth, *TEX and Metafont — New directions in typesetting*, Digital Press and American Mathematical Society, 1979.

[81] Donald E. Knuth, The Letter S, *The Mathematical Intelligencer*, Vol. 2 (1980), pp. 114–122.

[82] Donald E. Knuth, The Concept of a Meta-Font, *Visible Language*, Vol. 16, No. 1, Jan. 1982, pp. 3-27. journals.uc.edu/index.php/vl/article/view/5329

[83] Donald E. Knuth, *The METAFONTbook*, Computers & Typesetting, Reading, MA: Addison-Wesley, 1986.

[84] Donald E. Knuth, *Computer Modern Typefaces*, Reading, MA: Addison-Wesley, 1986.

[85] Donald E. Knuth, A Punk Meta-Font, *TUGboat*, Vol. 9 (1988), No. 2, pp. 152–168. tug.org/TUGboat/tb09-2/tb21knut.pdf

[86] Donald E. Knuth, *Digital Typography*, xvi+685pp. CSLI Lecture Notes, no. 78, Stanford, California, 1999.

[87] Eliyezer Kohen, A simple and efficient way to design middle resolution fonts, in [17, pp. 3–19] and [109, pp. 22–33 (1989)].

[88] Sacha Krakowiak, Xerox PARC et la naissance de l'informatique contemporaine, *Interstices* (revue Inria en ligne), 2012. interstices.info/jcms/int_64091/xerox-parc-et-la-naissance-de-l-informatique-contemporaine

[89] Gerry Leonidas, Farewell, Richard Southall, 17 June 2015. leonidas.net/2015/06/17/farewell-richard-southall/

[90] Raph Levien and Carlo H. Séquin, Interpolating Splines: Which is the fairest of them all?, *Computer-Aided Design & Applications*, Vol. 6, No. 1, 2009, pp. 91–102. http://graphics.berkeley.edu/papers/Levien-IIS-2009-06/

[91] Pierre MacKay, The KATIB System, a revolutionary advancement in Arabic script typesetting by means of the computer, *Scholarly Publishing* Vol. 8, No. 2, 1977, pp. 142–150.

[92] Pierre A. MacKay, Looking at the Pixels. Quality Control for 300 dpi Laser Printer Fonts, Especially METAFONTs, in [109, pp. 205–217 (1991)].

[93] Julien Mailland and Kevin Driscoll, *Minitel: The Online World France Built Before the Web*, 20 June 2017. spectrum.ieee.org/minitel-the-online-world-france-built-before-the-web

[94] Ladislas Mandel, Un caractère pour annuaires téléphoniques, *Communication et langages*, n° 39, 1979, pp. 51–61.

[95] Ladislas Mandel, Naissance d'une écriture – Réflexions sur la typographie et la télématique, dans *L'écriture télématique, années zéro* [35, pp. 41–49].

[96] Ladislas Mandel, *Du pouvoir de l'écriture*, Atelier Perrousseaux éd., 1998.

[97] Marie Marchand, *La Grande Aventure du Minitel*, Librairie Larousse, 1987.

[98] M. V. Mathews, Carol Lochbaum, and Judith A. Moss, Array: Three Fonts of Computer-drawn Letters, *The Journal of Typographic Research*, Vol. 1, No. 4, Oct. 1967, pp. 345–356. journals.uc.edu/index.php/vl/article/view/5008

[99] Paul McJones, Xerox Alto file system archive, Computer History Museum, last revised 9 Nov. 2017. xeroxalto.computerhistory.org

[100] H.W. Mergler and P.M. Vargo, One Approach to Computer Assisted Letter Design, *The Journal of Typographic Research*, Vol. 2, No. 4, Oct. 1968, pp. 299–322. journals.uc.edu/index.php/vl/article/view/5032

[101] Stanley Morison, On Some Italian Scripts of the XV and XVI Centuries, in *Letter forms, typographic and scriptorial: Two essays on their classification, history, and bibliography*, Typophiles, pp. 95–129, 1968.

[102] Heidrun Osterer and Philipp Stamm, *Adrian Frutiger – Caractères: L'œuvre Complète*, Walter de Gruyter, Switzerland, 2012.

[103] Victor Ostromoukhov and Jacques André, Punk : de METAFONT à PostScript, *Cahiers GUTenberg*, n° 4 (1989), pp. 23–28. http://numdam.org/item/CG_1989___4_23_0/

[104] Scott Pakin, *The Comprehensive LaTeX Symbol List*, 2021. ctan.org/pkg/comprehensive

[105] Arthur Phillips, *Computer Peripherals & Typesetting*, London, Her Majesty's Stationery Office, 1968.

[106] Vaughan Pratt, Techniques for conic splines, *ACM SIGGRAPH Computer Graphics*, Vol. 19, No. 3, July 1985, pp. 151–159. doi.org/10.1145/325165.325225

[107] Lilian M.C. Randall, A Nineteenth Century 'Medieval' Prayerbook Woven in Lyon, in *Art the Ape of Nature: Studies in Honor of H.W. Janson*, Moshe Barasch, Lucy F. Sandler (eds), New York, NY: Harry N. Abrams, 1981, pp. 651–668.

[108] Brian K. Reid and David Hanson, An annotated bibliography of background material on text manipulation, *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, ACM SIGPLAN Notices, Vol. 16, No. 6, June 1981, pp. 157–160. doi.org/10.1145/800209.806467

[109] RIDT, conference series proceedings:

- *Raster Imaging and Digital Typography*, Lausanne, Oct. 1989 (Jacques André and Roger Hersch, eds.), Cambridge University Press, 1989. books.google.com/books?id=mjO9AAAAIAAJ
- *Raster Imaging and Digital Typography II*, Boston, Oct. 1991 (Robert A. Morris and Jacques André, eds.), Cambridge University Press, 1991. books.google.com/books?id=Q9KtGcpfNgUC
- *Raster Imaging and Digital Typography*, special issue of *Electronic Publishing Origination Dissemination and Design*, (Jacques André, Jakob Gonczarowski, and Richard Southall, eds.), Wiley, 1994. books.google.com/books?id=gJcVAQAAIAAJ
- *Electronic Publishing, Artistic Imaging, and Digital Typography* (Roger Hersch,

Jacques André, and Heather Brown, eds.), Lecture Notes in Computer Science #1375, Springer-Verlag, 1998. books.google.com/books?id=bo453EDNBp4C

[110] Frank Romano (with Miranda Mitrano), *History of Desktop Publishing*, Oak Knoll Press, 2019.

[111] Richard Rubinstein, *Digital Typography — an introduction to type and composition for computer system design*, Reading, MA: Addison-Wesley, 1988.

[112] Lynn Ruggles, *Letterform Design Systems*, Stanford University Technical Report STAN-CS-83-971, 1973. http://i.stanford.edu/pub/cstr/reports/cs/tr/83/971/CS-TR-83-971.pdf

[113] Stewart C. Russell, *Hershey Font Outlines*, May 2014. scruss.com/wordpress/wp-content/uploads/2014/05/hershey_samples.pdf

[114] John Seybold and Fritz Dressler, *Publishing From the Desktop*, New York, NY: Bantam Books, 1987. archive.org/details/publishingfromde0000seyb

[115] Richard Southall, Interfaces between the designer and the document, in *Structured documents* (Jacques André, Richard Furuta, and Vincent Quint, eds.), Cambridge University Press, 1989, pp. 119–131. dl.acm.org/doi/10.5555/73173.73179

[116] Richard Southall, METAFONT in the Rockies: The Colorado Typemaking Project, in *EP'98* [109, 167–180 (1998)]; link.springer.com/chapter/10.1007%2FBFb0053270. Republished in *Computers and Typography 2* (Rosemary Sassoon, ed.), Intellect Books, 2002; books.google.com?id=wdYmvQD5C8IC

[117] Richard Southall, *Printer's Type in the Twentieth Century — Manufacturing and Design Methods*, The British Library/Oak Knoll Press, 2005.

[118] Bob Sproull, *Font Representations and Formats*, Internal note, Xerox PARC, Mar. 1977. xeroxparcarchive.computerhistory.org/indigo/printingdocs/.FONTFORMATS.PRESS!1.pdf

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

[120] David Sudweeks, Type Trends: Superelliptical Type, *FontShop Typographic Trends*, Nov. 2012. fontshopblog.wordpress.com/2012/11/22/type-trends-superelliptical-type

[121] Edward Tufte, *Visual Explanations — Images and Quantities, Evidence and Narrative*, Cheshire, CT: Graphic Press, 1997.

[122] Gerard Unger, The Design of a Typeface, *Visible Language*, Vol. 13, No. 2, Apr. 1979, pp. 134–149. journals.uc.edu/index.php/vl/article/view/5266

[123] Gerard Unger, *in* Other Replies to Donald E. Knuth's article "The Concept of a Meta-Font", *Visible Language*, Vol. 16, No. 4, Oct. 1982, pp. 353–356. journals.uc.edu/index.php/vl/issue/view/360

[124] Andries Van Dam and Eric E. Rice, On-line Text Editing: A Survey, *Computing Surveys*, Vol. 3, No. 3, Sept. 1971, pp. 93–114. doi.org/10.1145/356589.356591

[125] Yue Wang, Interview with Charles Bigelow, *TUGboat*, Vol. 34 (2013), No. 2, pp. 136–167. tug.org/TUGboat/tb34-2/tb107bigelow-wang.pdf

[126] Matthew Westerby, *The Woven Prayer Book: Cocoon to Codex*, Satellite Series. Paris, France & Chicago, IL, USA: Les Enluminures, 2019.

[127] Norman M. Wolcott and Joseph Hilsenrath, *A Contribution to Computer Typesetting Techniques: Tables of coordinates for Hershey's Repertory of Occidental Type Fonts and Graphic Symbols*, National Bureau of Standards, NBS Special Publication 424, Apr. 1976. scruss.com/wordpress/wp-content/uploads/2014/04/tables_of_coordinates_for_hersheys_repertory_of_occidental_type_fonts-wolcott_and_hilsenrath.pdf

[128] Norman M. Wolcott, FORTRAN IV Enhanced Character Graphics, National Bureau of Standards, Institute for Computer Sciences and Technology, NBS Special Publication 500-32, Apr. 1978, 64 pp. archive.org/details/fortranivenhance5003wolc

[129] Hermann Zapf, *Hermann Zapf and His Design Philosophy*, Chicago, IL: Society of Typographic Arts, 1987. Introduction by Carl Zahn.

[130] Herman Zapf, Vom Formgesetz der Renaissance-Antiqua, *Der Polygraph*, Heft 21.

◇ Jacques André
https://jacques-andre.fr