Type's Trajectory: from Ikarus to Acrobat

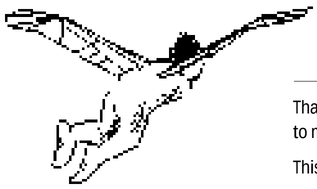
Conrad Taylor

I prepared these notes to support my workshop at the conference **Glasgow Style**, 14–16 September 1993, organised by the British Computer Society Electronic Publishing Specialist Group (BCS–EPSG).

The conference proper was preceded by an optional afternoon workshop with two sessions. Rob Waller conducted his as an introduction to design. My presentation, which followed Rob's, gave a similar overview of modern type technology.

The topic is divided into broadly three sections: developments in digital type formats; problems to do with character sets, especially very large ones; and document interchange strategies.

About my title: **Ikarus** was the first generalized tool for digitizing fonts, now nearly 20 years old; **Acrobat** is Adobe's new technology for exchanging documents electronically, overcoming the hurdle of font interchange.



Thanks to Rachel Kellett and Bruno Maag for helping with research for this paper; also for their contribution to my understanding of type to Erik Spiekermann, Peter Karow at URW, and Louise Domenitz at Bitstream.

This paper is dedicated to the memory of Stella Jane Reekie.

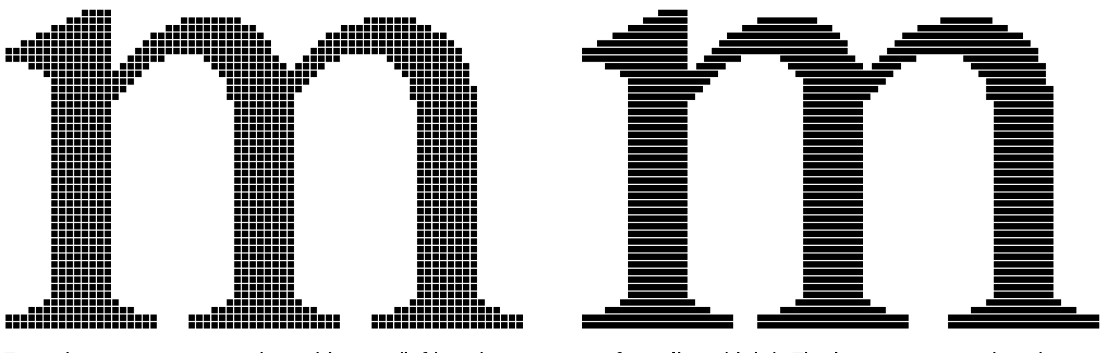
DIGITAL FORMATS *for* TYPE

The type fonts for the first photosetting machines were photographic negatives, through which light was projected to form type images on the film or paper. But in the mid 1960s, CRT photosetters made their debut, to be followed by laser photosetters, and type went digital, ultimately being output as a **bit-map**.* Some way to store these type images inside the typesetting system was required.

Bit-mapped font digitization

The first first method used was to store the bit-maps directly. This minimised the computational requirement, which was appropriate for the slow computers of the time. On the minus side, each size of type had to be stored as a separate entity. Intermediate sizes could not be produced, type could not be rotated, and if large sizes of many fonts were to be kept on line, considerable space was consumed on the hard disk.

One way of minimizing the storage requirement for large type sizes was **run-length encoding**, similar to the compression scheme used by CCITT Group 3 fax machines. Rather than storing every pixel, this method divides the character matrix into horizontal or vertical scanlines; only the points of transition between black and white are recorded rather than the state of each individual pixel



Type character represented as a bit-map (left) and as an array of scanlines (right). The latter corresponds to how the character is imaged in a laser photosetter. Recording the points on the scanline where the beam is switched on or off (run-length encoding) reduces the storage requirement for bit-mapped type.

* Many technical terms are included in the Glossary to this paper.

Vector formats for type

The alternative to bit-map encoding of type is **vector encoding**. The word ‘vector’ comes from a Latin word for a messenger, someone sent point-to-point, and is used in mathematics to describe a line with distance and direction.

When type forms are encoded as vectors, their outlines are defined as line segments running between points on an underlying co-ordinate grid. The advantage of vector encoding over bit-map encoding is that by multiplying the co-ordinates for these points by a scaling factor, a single outline description can be used to produce a wide range of type sizes on request.

Using vector formats—’rasterization’

The only kinds of output device which use vector definitions of type directly are **pen plotters** and other **numerically-controlled machines** as used for cutting vinyl signs, or for cutting, engraving or routing metal or stone; and non-pixel-based CRT displays (‘vectorscopes’).

Typesetting machines, laser printers and modern computer displays are inherently bit-mapped output devices, so the vector-defined type has to be converted into a bit-map form for use. This process of conversion is known as **rasterization**.

It would be silly to perform rasterization calculations each and every time the same letter was required for output. Therefore typesetting systems rasterize all or part of an entire character set* the first time it is requested at a given size, and store the bit-maps in a fast area of temporary memory from which they are retrieved for actual output. This facility is known as the **font cache**, and may be stored in volatile memory (RAM) or written to disk.

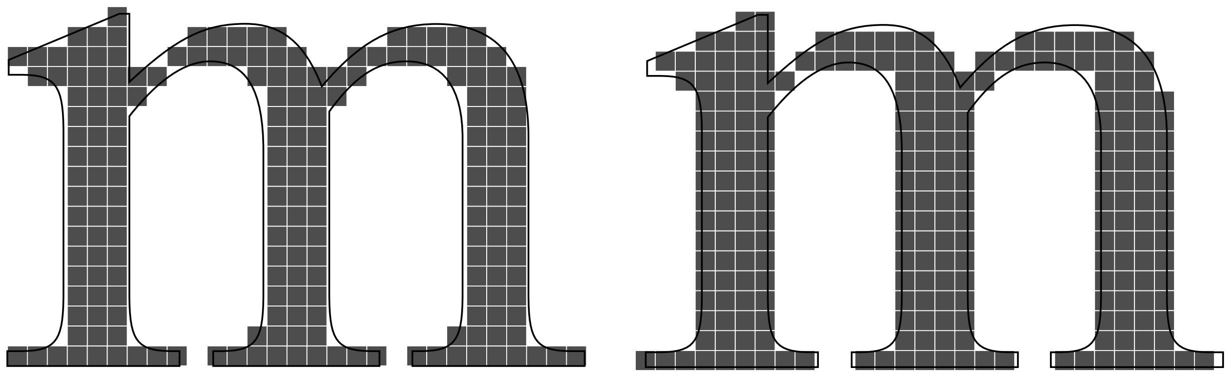
The first vector formats: straight lines

Early typesetting systems using vector formats for storage defined type outlines using lots of short straight lines. For the low-powered computers of the time, these were easy to convert into bitmaps, but the quality of type could deteriorate at large sizes. I recall seeing an IBM in-house magazine, typeset by the local Greenock newspaper publishers, in which the 4 cm headline type looked as it it had been clipped out with scissors!

* For reasons of speed and economy of memory, it is common initially to rasterize only the unaccented Latin upper and lower case letters, numerals and punctuation. If further characters are required, the rest of the character set is rasterized and sent to the font cache.

Rasterization and the aliasing problem

In 1987 when Jan White and I visited Bitstream in Cambridge, Ma., Bitstream workers were engaged in 'fine-tuning' small sizes of bit-mapped fonts. They explained that type designs are curved and subtly detailed; but when such a form is to be rendered as a bit-map, the curves cross the pixel grid and a decision has to be made for each pixel whether it shall be 'on' or 'off'. This leads to **aliasing problems**.



How to represent the curved forms of type with spots? It's not good enough to say that a pixel is 'on' if most of it is inside the outline. Here, the same outline is set down differently on the pixel grid, resulting in different 'on' and 'off' pixels. So should the vertical stems be represented with 3 pixels, or with 4?

At high output resolutions, as in typesetting, there is no problem because the compromises are too small to be perceived. But when the available display matrix is small, as it is in medium-resolution laser printers and still more on computer screens, hard decisions have to be made about which pixels shall be 'on' and which shall be 'off'.

(Bitstream staff had the opportunity to correct these defects by hand. But when type is rasterized 'on the fly' from outline descriptions, programs known as **hints** have to correct these aliasing problems.)

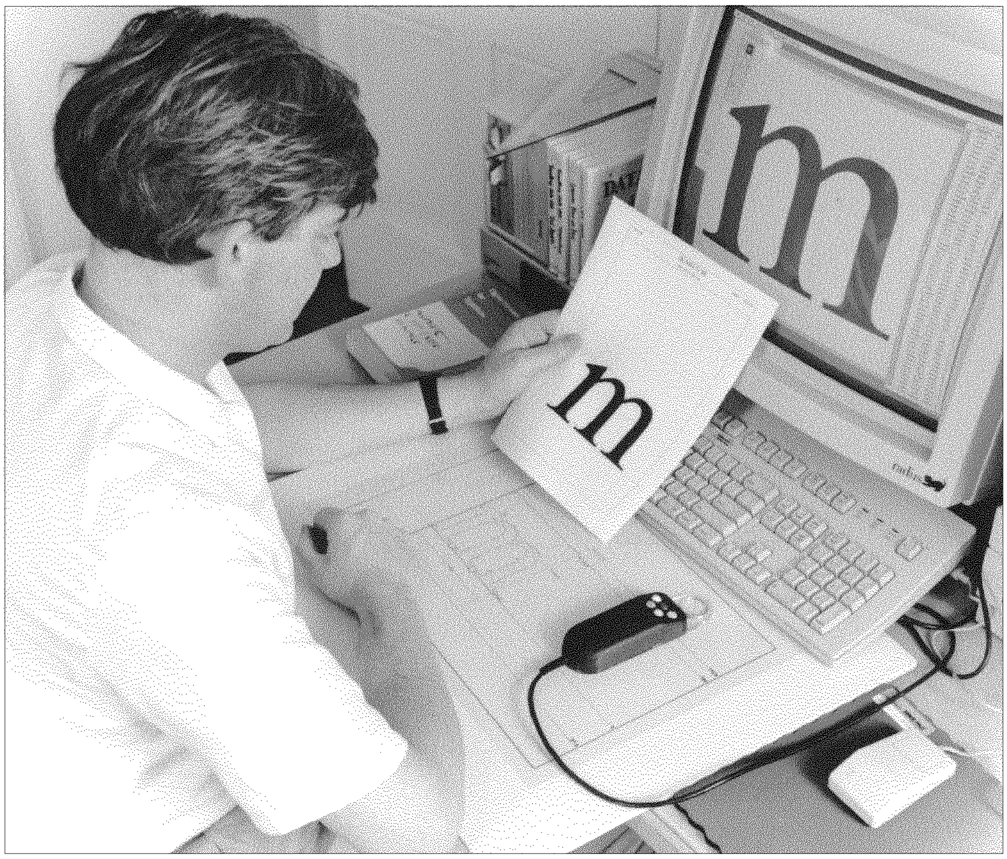
Some typical aliasing problems include:

- ◆ Vertical and horizontal lines, intended to to be equally thick, may be rasterized unequally. The usual solution is to shift them slightly, sacrificing accurate representation of larger white spaces (counters).
- ◆ Delicately curved lines, such as vertical stems in Optima, or the feet of certain old-style serif typefaces, may be rendered as straight lines—except for one or two pixels growing on the stem like a wart, or bitten out of it. **Channeling** controls in hinting force these stems to be rendered straight at low resolutions.

An enlightening discussion of aliasing problems and how to correct for them may be found in Section 7.2 'Manual corrections' of *Digital Formats for Typefaces* by Peter Karow.

Ikarus M

Shown right, the Ikarus M system for font digitization. The type drawings are placed on a graphics tablet and a digitizing puck is used to mark strategic points along the outline of the letterform.



Outline methods for font digitization: the Ikarus system

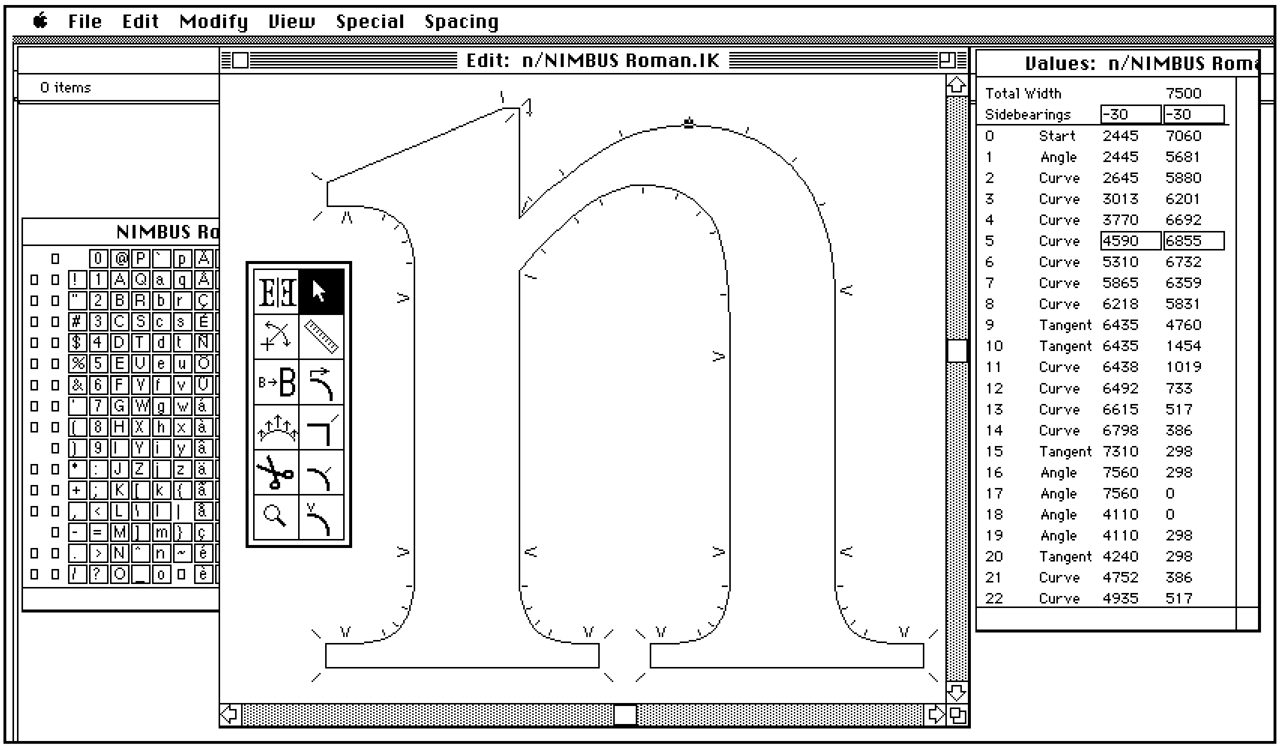
When the firm of Dr. Ing. Rudolf Hell GmbH introduced their pioneering CRT photosetter, the **Hell Digiset**, they realised that an efficient means must be found for converting existing type libraries into digital formats. The firm of URW Unternehmensberatung in Hamburg was contacted to help solve the problem.

Dr. Peter Karow of URW realised it was pointless to re-digitize type every time typesetting technology changed: a high-level format must be devised, from which actual digital fonts in a variety of machine formats could be manufactured on demand.

The result was the **Ikarus** system. (Dr. Karow says it got that name because at first it crashed so often!)* Herman Zapf's *Marconi* was the first typeface to be digitized on Ikarus, by Peter Käpernick in 1977.

The Ikarus system defines type outlines as straight lines and simple arcs. You may think simple arcs (circle segments) are too crude to use for defining the subtly changing curves of type characters, but Ikarus satisfies the requirements of most type designers by describing such curves as a sequence of arcs linked end to end continuously.

* Not because of problems with the Sun… Ikarus runs on a Vax.



Values: n/NIMBUS Rom

Total Width			7500
Sidebearings		-30	-30
0	Start	2445	7060
1	Angle	2445	5681
2	Curve	2645	5880
3	Curve	3013	6201
4	Curve	3770	6692
5	Curve	4590	6855
6	Curve	5310	6732
7	Curve	5865	6359
8	Curve	6218	5831
9	Tangent	6435	4760
10	Tangent	6435	1454
11	Curve	6438	1019
12	Curve	6492	733
13	Curve	6615	517
14	Curve	6798	386
15	Tangent	7310	298
16	Angle	7560	298
17	Angle	7560	0
18	Angle	4110	0
19	Angle	4110	298
20	Tangent	4240	298
21	Curve	4752	386
22	Curve	4935	517

Screen-dump from Ikarus M. The letter-form is defined by the placement of points along its contour. There are curve-points, angle-points, tangent-points where curved segments turn into straight ones; and each contour has a start-point which is simultaneously one of the other kinds of point. The digitization worker uses four different buttons on the puck of the graphics tablet to define each chosen location as being one of these four kinds of point. The curves are calculated automatically by the program. Afterwards, this IK-format can be converted to other formats for use—PostScript Type 1, for example.

In the Ikarus system, arcs are not defined explicitly by the digitization worker, but are calculated automatically to make a smooth path that runs between the explicitly defined points, as shown above.

How good is Ikarus?

Ikarus is poorly suited to type *design*; it is however an efficient way to *digitize* type designs which already exist as drawings or Rubyliths. It has been widely adopted by type design houses such as Monotype and Letraset.

One area of controversy about Ikarus is the alleged 'ease of use'. Its authors comment that it is easier in comparison with the Bézier curves used to define PostScript type, because the control points for Bézier curves do not lie directly on the curve. Dr. Karow says,

> *The trial and error method required to select good Bézier knots and control points is too time consuming.*

For someone really used to using Bézier-curve drawing software such as *Adobe Illustrator*, however, this does not ring true. I can choose Bézier 'knots and control points' quickly and accurately on the basis of an almost instinctive appreciation of the relationship between these points and the resulting curve, which I've gained through experience. And at least one is directly in control of the curve shape, rather than relying on automatic arc interpolations.

Here we come to a central problem of Ikarus, from a user's point of view. You cannot select *any* set of co-ordinates around the contour of a letterform and get a good representation of your original drawing. According to where you place points—curve and tangent points in particular—you get arc interpolations which fit the curve brilliantly, or others where arc interpolations wander all over the place so that only at the digitized points have you any guarantee of congruence.

Ikarus users have to develop an instinctual appreciation of how what I call 'the interpolation engine' works. It surprised me on my own visit to URW *how little* their own type designers understood how this engine works. I kept being told, 'Ah, you must ask Dr. Karow!' Eventually I hypothesised a description of how this engine works, and Peter Karow confirmed that I had it more-or-less right.

How Ikarus arc interpolation works

Ikarus M, with its fast screen updating, offers a unique opportunity to see how the placement of each new curve point along a long curved path *also* affects the interpolation of arcs between *previously* digitized curve points. If you look at the screen just at the moment you press the button on the puck, you see arc adjustments ripple back between as much as *six* previously set points.

Each Ikarus curve point is the meeting point of two arcs. The arcs may have different curvatures (radii), but they meet end-to-end in a perfect tangent. However, between two Ikarus curve points there is not one arc, but two; so somewhere between these points is another point where the two arcs meet tangentially. This point is not explicitly digitised, and may move in accordance with the need of the 'interpolation engine' to ensure that all arc segments butt up end-to-end, all the way along the curve back to the last corner or tangent point.

Therefore, when each new curve point is entered, the interpolation of arcs between that point and the previous one cannot help but affect the shape of the one before it, and the one before that, and the one before that…

The importance of choosing the right points

In helping to draft the text of the English manual for Ikarus M, I commented:

> *Learning to use Ikarus is largely a matter of learning the best locations for points… Experienced type designers who use Ikarus comment that the best method is to produce the very best artwork you can, mark it intelligently, and digitize it with care.*

Following interviews with several workers at URW, who were most helpful, I then drafted the 10 pages or so of the Ikarus M manual which explain to users how to mark up their artwork intelligently to prepare it for digitization.

METAFONT

The energetic and whimsical Professor Donald Knuth of Stanford University was so disenchanted with systems for typesetting mathematics that he created his own—TEX (pronounced *tech*). Initially implemented on large shared computer systems, TEX is available for a range of computers, for example PCTEX and LaTEX, and the Macintosh program TEXtures.

Knuth also devised a way of creating resolution-independent digital typefaces, called METAFONT.

METAFONT's interest for anyone outside the TEX-using community is fairly academic (as for the most part is the TEX-using community), but it is worth a mention because pioneering concepts in METAFONT have inspired other type technologies.

The METAFONT pen-and-ductus model

Knuth wanted his vector-encoded font format to permit more variation than mere scaling. It should be possible to make type bolder or thinner, vary the x-height, even switch serifs on or off.

Knuth demonstrated this, and made large claims for METAFONT, in an illustration for an article in *Visible Language*.* He provided a setting of a psalm, in the course of which the font changed itself gradually—the fashionable term now would be 'morphed'—from a serif font with vertical emphasis and a small x-height, into a sans-serif font with monoweight strokes and a large x-height.

METAFONT was supposed to allow the creation of 'fonts with control knobs on', twiddling which could create the font of your choice.

To make this possible, Knuth eschewed defining the type's outlines explicitly. Instead, a METAFONT character is thought of as being drawn in a number of strokes, each stroke along a vector-described path corresponding to what in calligraphy would be called the **ductus**. Each stroke could be made using one of a set of defined 'pens' of various shapes, sizes and orientations, loaded with either black or white digital ink.

The result of passing this sequence of virtual pens over a matrix of pixels produces the font bitmaps to be cached and used for output.

* *The Concept of a Meta-font* in *Visible Language* 16, no. 1 (Winter 1982): 3–27. Some of Knuth's claims for the 'meta-fontness' of METAFONT were criticised in the reply article *Metafont, Metamathematics, and Metaphysics* by Douglas Hofstadter, which was reprinted in Hofstadter's book *Metamagical Themas.*

See also *TEX and Metafont: new directions in Typesetting* by Donald Knuth; Digital Press 1979; also published, I believe, by Addison-Wesley.

Some comments about METAFONT

- The idea of a mutable font 'with knobs on' has recently been reborn in the form of Adobe's **Multiple Master** font format, although with a less radical agenda than METAFONT, and using a method of interpolation between defined outlines, rather than building out from a central spine.
- METAFONT was rather a boffin's way of making type. Peter Karow comments that:

  Metafont is not very well suited for exact and cost-effective reproduction of existing types. Furthermore, one has to write a rather long program for each individual character.

  (Knuth contrasted METAFONT with the children's programme *Sesame Street*; if the latter can announce 'This program is brought to you by the letter A', Knuth would say, 'This letter A is brought to you by a program'!)
- In discussing various kinds of curve which might prove useful for defining paths, Knuth recommended spline curves such as those which may be described using quadratic and cubic equations. Cubic splines have been adoped (as Bézier curves) for defining font outlines for PostScript, and quadratic splines for TrueType.

PostScript's cubic curves

PostScript, from Adobe Systems, is a Page Description Language (PDL), but also can be used to define the fonts used on those pages, which means that a PostScript font can be downloaded to a printer in the body of the page description itself.

The type characters in a PostScript font are in almost all cases vector-encoded by describing their outlines.* The curves used to define character shapes are **cubic splines**,† that is to say, curves which may be defined by a cubic equation.

The kind of cubic curves used are known as **Bézier curves** after the French mathematician who devised a way of describing them, not with a cubic equation directly, but by reference to locations in co-ordinate space. Each Bézier curve segment may be defined by just four x–y co-ordinates, of which two are on the curve itself, at the start-point and the end-point of the curve segment, and the other two are not on the curve at all,‡ but help to define its shape.

* Exceptions: the PostScript Type 3 format can support font definitions as bitmaps. Also, it is possible to define characters by an 'inline' method, as a collection of lines or paths having defined thickness and endcap styles. The early Adobe versions of Courier, for instance, were inline rather than outline fonts.

† The term 'spline' originally described the long thin planks running from bow to stern in boat construction, then by extension was applied to the flexible rulers used by planners and engineers to help map the paths of railway lines and roads, then by further extension to any curve the curvature of which varies along its length.

‡ To be pedantic, not necessarily on the curve. For example, a straight line could perversely be described as a Bézier curve in which the control point associated with each knot is directly on top of that knot.

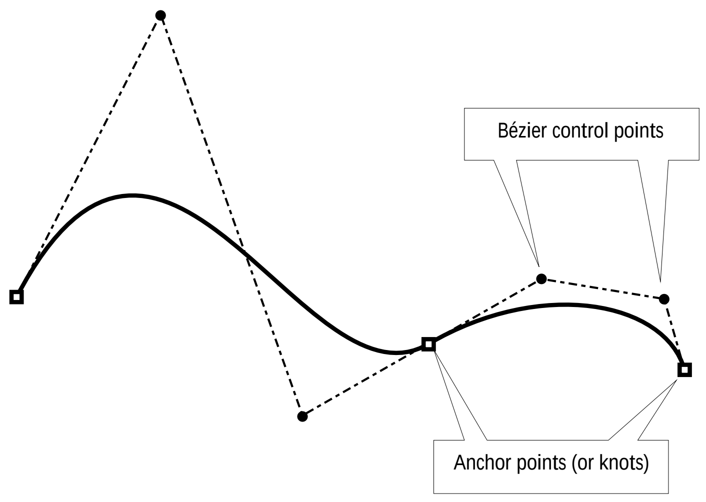
By convention, the points which start and end the curve segment are known as **anchor points** (or sometimes 'knots'), and the other points which influence the shape are called **Bézier control points** — or just control points, or BCPs.

Bézier curves

Four points in co-ordinate space are enough to describe these elegant curves, which can turn back on themselves and which are well suited to describing the outlines of type characters.

This diagram shows anchor points and control points in relation to two curve segments.

In drawing programs like *CorelDRAW!* or *Illustrator*, or in type design programs like *Fontographer*, the control points show up on screen and can be manipulated like handles to adjust the curve to the required shape.



How Bézier curves are rendered

Not only for printing purposes, but also for the display of Bézier curves on screen to facilitate editing, those four defined co-ordinate points must be rasterized. Fortunately this is easy, involving two stages: an iterative process which creates a very fine polygonal path (an approximation to the curve using short straight lines) followed by a procedure which compares this path against the matrix of display pixels and determines which are 'on' and which are 'off'.

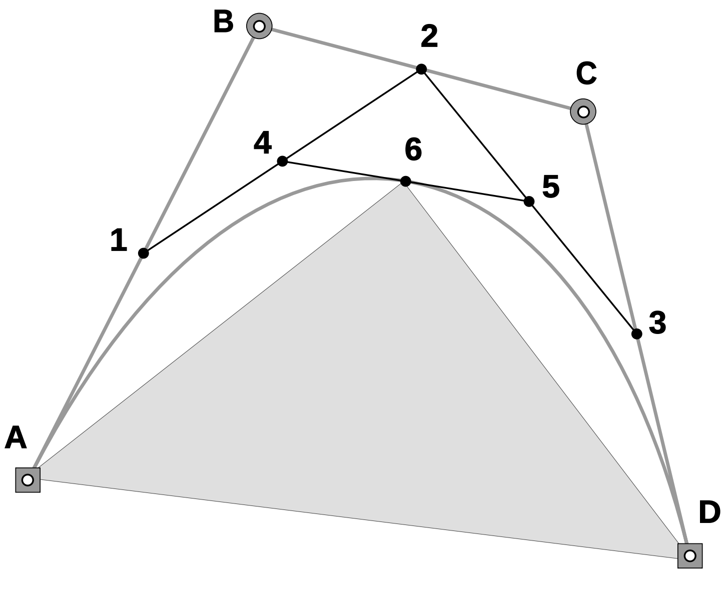
Bézier to polygon: first approximation

At first, only the four defining co-ordinates exist, shown here as anchor points **A** and **D** and Bézier control points **B** and **C**.

Averaging x and y co-ordinates of A and B produces a midway point; the same is done between B and C and between C and D. Three intermediate points are produced (**1**, **2**, **3**).

Repeating this procedure for two more iterations produces all the intermediate points shown on the diagram as small black spots.

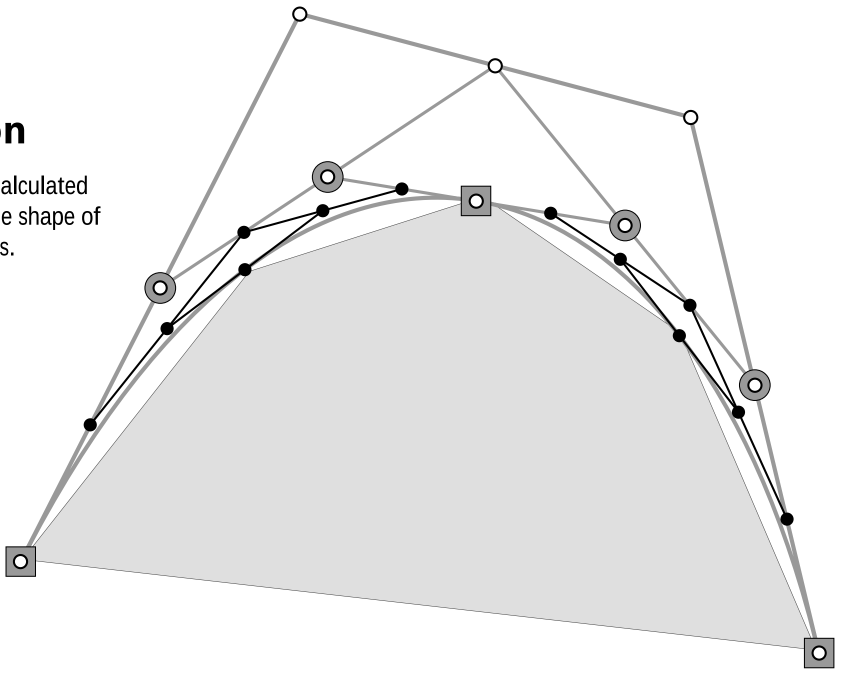
The Bézier curve is now divided into two. **6** is the new intermediate anchor point and **1** & **4**, **5** & **3** are now control points. The shaded area shows the first approximation to the required curve.



Simply by averaging the co-ordinate locations and producing intermediate points as shown above, the first approximation to a Bézier curve display is produced. As shown in the next diagram, performing the same operations using the intermediate points results in a second approximation. The iteration is allowed to 'bottom out' at a point where the granularity of the polygon is at or close to the pixel level.

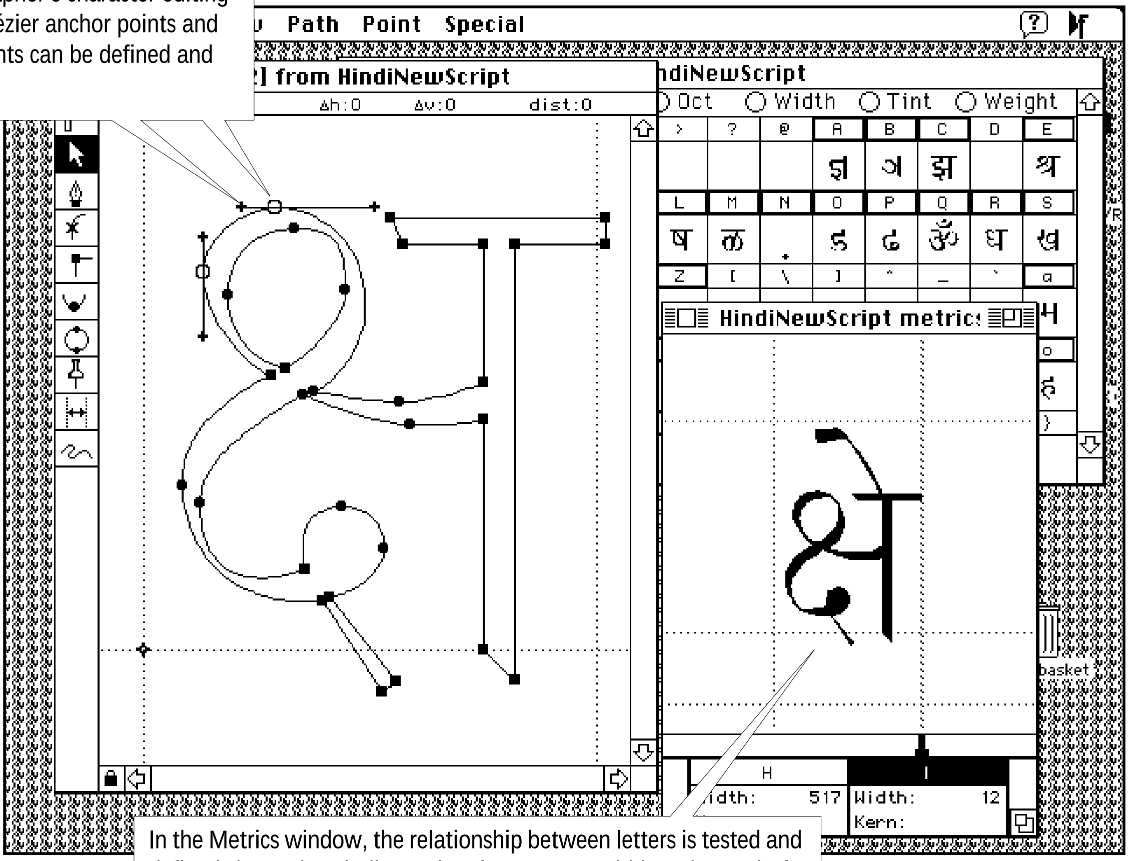
Bézier to polygon: second approximation

More and more intermediate points are calculated until a very fine polygon approximates the shape of the curve defined by the initial four points.



At this point the second part of the rendering process can procede as the outline is tested against a matrix of output pixels and rasterized into a bitmap. However, for the production of characters for lower resolutions, it is helpful if a hinting procedure can intervene to avoid the kind of aliasing problems described on page 3.

Making your own type fonts in Fontographer



In Fontographer's character editing window, Bézier anchor points and control points can be defined and edited.

In the Metrics window, the relationship between letters is tested and defined. (Note the Hindi vowel '**e**' is near zero-width and negatively offset, so it appears superimposed on top of the base consonant.)

Creating PostScript fonts

Programs exist for end users to create their own PostScript typefaces using Bézier curve tools. On the Macintosh, for example, there are *Fontographer, FontStudio* and *Kingsley ATF.*

It is also possible, and often more comfortable, to create character outlines in a drawing program with Bézier curve drawing tools, then bring them together in a font-editing program.* (Erik Spiekermann told me that when he worked at Adobe, *Illustrator* was the preferred tool for defining font outlines.)

Some workers in type design straight to screen; most probably prefer to do at least tentative sketches on paper, scanning them in to serve as reference images over which Bézier outlines are constructed.

* The Macintosh Clipboard does not directly support PostScript objects, but Bézier-curve outlines can be copied as 'PICT with Postscript comments' to the Clipboard by holding down the Option key while choosing Copy from the Edit menu. From the Clipboard they can then be pasted into a character window of the font editor.

Think about 'extreme points'

Just as Ikarus demands forethought in placing digitization points, type designers should observe rules for optimum placement of Bézier anchor and control points—rules which do not apply to people using these curves to illustrate. This is because **hinting processes** in font rasterization look for positional information from *expressly* digitized locations, and cannot efficiently infer them from elsewhere.

So, although it is possible to set Bézier anchor and control points in all sorts of places, it is sensible to place control points at the places which in Ikarus are called '**extreme points**' — the points where a curve reaches its highest, lowest, leftmost or rightmost point. Also:

- A Bézier control point for a top or bottom extreme point should have the same *y* (vertical) co-ordinate as its anchor point
- A Bézier control point for a left or right extreme point should have the same *x* (horizontal) co-ordinate as its anchor point.

This explicit digitization of extremes helps the font-creating program apply the hinting rules correctly.

Note: these rules have been observed for the two selected points of the Devanagari character shown on page 11 (the point top left, with Bézier control 'arms' displayed). However, the rules have not been observed consistently. Can you identify two locations where additional Bézier knots are required?

Type 1 & Type 3; hints and secrets

By the time Adobe launched their first implementation, for the Apple LaserWriter, two alternative PostScript type formats had been defined: Type 1 and Type 3.

Type 3 fonts are also known by Adobe as 'user-defined fonts'. They are unencrypted; they may have complex character shapes; but they do not have provisions for optimizing appearance at lower output resolutions (hinting). The Type 3 format was made public at an early stage, and programs like *Fontographer* came onto the market to let users—and other type vendors—make their own fonts.

Type 1 fonts are what most people think of as 'PostScript fonts'. They support only relatively simple character outlines, but have an effective way of defining hinting parameters. Adobe initially kept the definition of Type 1 fonts secret. They are stored in an encrypted format; and early versions were copy-protected so that they could be installed only for a single PostScript RIP.

Although PostScript was advertised as an 'open' standard, the rival type vendors such as Bitstream and Monotype resented the secrecy surrounding Type 1. It meant they could produce only Type 3 fonts, which worked well on imagesetters, but poorly at 300 dpi.

‘Type Wars’ — Display PostScript, TrueType and ATM

Meanwhile Adobe were exploring PostScript as a model for imaging not only on printers, but also on computer monitors. The benefits of this **Display PostScript** system would be a closer match between what you see on screen and what is printed. NeXT Inc., founded by Steve Jobs, adopted Display PostScript for their computers, and Adobe lobbied to get it adopted at Apple Computer too.

In the process Adobe irked a faction at Apple who considered it would be strategically unwise for Apple to abandon their QuickDraw display model in favour of someone else’s technology, and who also wanted an alternative outline font format developed in competition with PostScript Type 1. Some talked of Apple abandoning PostScript altogether. This ambition led to the ‘Royal’ project, which resulted in a new scaleable font format, **TrueType**.

Apple and Microsoft collaborated to establish TrueType, and were later to build TrueType rasterization facilities into Macintosh System 7 and Windows 3 and 3.1. Following the dramatic announcement of this Apple–Microsoft alliance against Adobe at the Seybold San Francisco conference, Adobe were forced both to compromise and to innovate:

- ◆ Adobe instantly promised to release full details of the Type 1 font format, and soon any other type vendor (indeed, anyone with a copy of *Fontographer*) was able to make Type 1 fonts.
- ◆ Adobe developed **Adobe Type Manager** (ATM) software for Macintosh and Windows PCs, bringing many of the benefits of the Display PostScript system to Mac and Windows users.

At the time, the computer press were describing these manœvres as ‘Type Wars’. But a few years later, compromise and co-operation is once again the order of the day. The enormous support for ATM and Type 1 fonts in the graphic arts and DTP world caused Apple to rethink its attitude to Adobe’s technology; Adobe have been forced to be more open and to price their products more competitively. Adobe is supporting TrueType; Apple is supporting Type 1.

The TrueType font format

Just what is TrueType? Like PostScript Type 1, it is an outline font format, with a hinting mechanism to improve rasterization at low resolutions.

To use a TrueType font on a Macintosh or a Windows PC, a single file only need be installed. Thereafter, you can specify any size of type permitted by the application used, and the type is rasterized from the outline font description to provide accurate representation on screen. In contrast, to get the same facilities with Type 1 fonts, font bitmaps *and* outlines must be installed, and Adobe Type Manager must also be present. (It is worth stressing that this is not a ‘fault’ of Type 1, but a result of Apple’s and Microsoft’s decision to support TrueType more directly than Type 1 fonts.)

Some characteristics of TrueType fonts

- ◆ Whereas PostScript describes type outlines with cubic splines, TrueType uses **quadratic splines**. More digitization points are required to describe the same shape. However, proponents of TrueType argue that type described with quadratic curves can be rasterized more quickly.
- ◆ Like Type 1 fonts used with ATM, TrueType works with the Macintosh (QuickDraw) and Windows system-level printer drivers to rasterize type for *printed* output as well as for screen. This is important, because relatively few output devices have internal support for TrueType rasterization, in contrast with widespread support for PostScript.
- ◆ The hinting mechanism in TrueType is defined fully within each font, whereas Adobe’s strategy is to include hinting *parameters* in a Type 1 font but define the hinting *algorithm* in the PostScript interpreter of the output device or of ATM.

  (There are proponents for both approaches. Adobe argues that their scheme allows any improvements in hinting technology implemented in future versions of PostScript interpreters to apply retrospectively to all pre-existing Type 1 fonts.)

I have found it relatively difficult to gain a direct appreciation of the structure of TrueType fonts. This is because font editing tools often use other models for defining the type outlines (IK-format, Béziers) and generate the quadratic curves for TrueType from those.

Now, thanks to Desktop Publishing and digital type technology, typography has been democratized. Millions of computer users worldwide are now in a sense ‘typographers’, making choices about fonts, sizes, leading and so on.

Of course, we all recognise that having fine typefaces to work with does not make a person into a real typographer. Design education must also be democratized.

However, my focus for the rest of this paper is different: a more critical evaluation of the type technologies themselves. What are their continuing shortcomings? What technical problems occur in their use? And what is being done to overcome these problems?

CHARACTER SETS: *requiem for 26 lead soldiers*

It is not enough to have type characters nicely designed and in a format appropriate for use on your computer. You must have a way of accessing the ones you want, and exchanging them reliably between applications and users.

In my heading above , I recall one of those pompous phrases in which the old compositors and printers used to glorify their trade, often in settings for type sample sheets:*With twenty-six soldiers of lead, I shall conquer the world!*

Of course, a character set of twenty-six letters was *never* enough for typographers. Not only was there always a need for numerals, upper case letters and marks for punctuation; to match accepted standards in manuscript books, Gutenberg had to manufacture a huge number of ligatures and other special letterforms.

Limitations of the machine

There were no finite limits to the number of characters in foundry type, so long as you had enough compartments to keep the little ‘lead soldiers’ in and could remember who lived where.

Mechanized typesetting imposed limits, however. The matrix case of the Monotype caster, a frame holding the moulds (matrices) for all characters available to the type casting machine at any one time, had slots for 16 × 17 characters, 272 in total. (*This sounds impressive, until you realise that if italics were required they would have to be accommodated within the same matrix case*.)

The digitization of type made it possible to hold many fonts on line at once and use them in combination. Typesetters had access to extended type families, and large character sets. In a professional typesetting system, a font might have over 500 characters.

As long as typesetting happened on specialist machines, it was easy for manufacturers to implement their own ways of managing large character sets. Then desktop publishing came along. Type vendors had to adjust their product to a new machine environment: one already defined by industrial standards not really appropriate for handling large character sets.

Bytes, ASCII and all that

Computers manage character sets by binary numerical codes, standardized to make it easy to exchange information between different kinds of computer. The importance of the American market makes the dominant standard **ASCII**, the *American Standard Code for Information Interchange*, defined in ANSI standard X3.4–1986.

The ASCII character set is almost identical to that defined in the ISO standard **ISO 646.**

ASCII allows one **byte**, or 8-bit binary number, to be used to refer to each character in a character set. If you are allowed to use all eight digits in a byte, you can number up to 256 entities.

At the time these standards were being hammered out, the eighth bit in a byte was often used as a ‘parity bit’ for error checking, leaving seven binary digits available to enumerate only 128 entities. Furthermore, 33 of these were reserved for communication control codes such as DEL, CR, ESC, CAN and so on, leaving 94 printable characters as shown below, plus the space character.

7-bit ASCII character set

Omitting the 33 control characters, there are 94 printable characters, plus the space character.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890 ?!.,;:-'"*`
@$%&+=^~_/|\  <([{}])>
```

7-bit security

Users of electronic mail systems such as JANET and the Internet are often limited to the use of this 7-bit character set, because it is the only one which can guarantee compatibility across a wide range of computer systems. In travelling across networks, the eighth bit may be stripped as the file passes through some gateway.

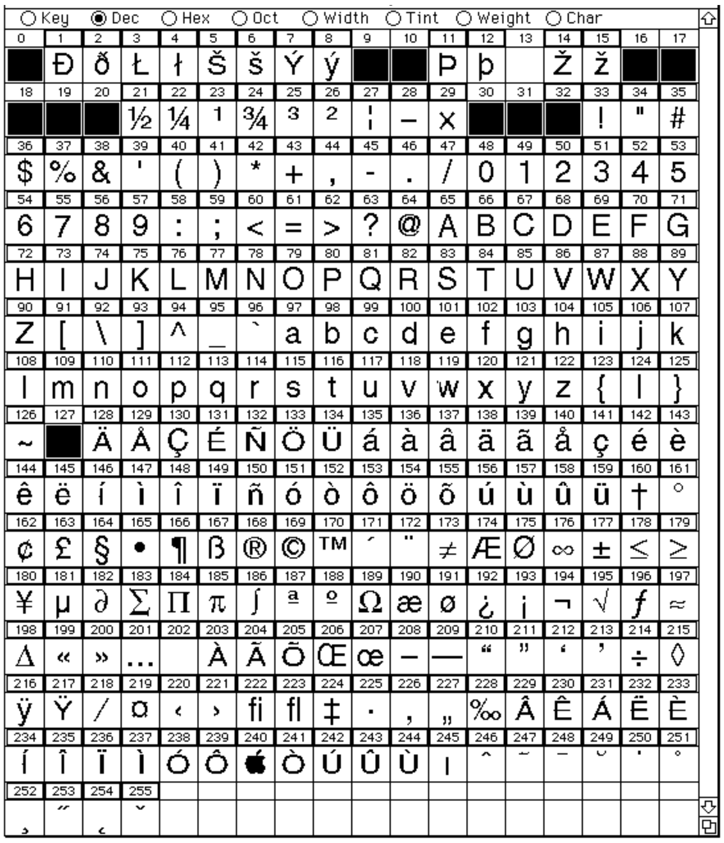
When complex character sets may have to be transferred across wide area networks, they often must be converted into some 7-bit form of expression. Examples of 7-bit coding of wider character sets may be found in TEX markup, PostScript files, PDF files and SGML markup, and in FrameMaker’s Maker Interchange Format.

Adobe Standard Character Encoding

Most Adobe fonts and other PostScript fonts for desktop publishing follow **Adobe Standard Character Encoding**. Not all of the characters in a font can necessarily be accessed, however: note the characters stored in the low ASCII values (1–32) which are inaccessible to users of the Macintosh UK operating system, as are some of the floating accents at the very highest values.

This display window has been created by opening the Helvetica font in Fontographer. Empty locations have been painted black on the screen dump.

Not all characters encoded using this scheme are actually stored in the font themselves. About a dozen, mostly mathematical, are 'referenced' from the Symbol character set and do not show up when a font is opened in Fontographer in this way. They have been restored for the purposes of this display by pasting them into the appropriate character locations.



Key [✓ Dec] Hex Oct Width Tint Weight Char

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
■	Đ	ð	Ł	ł	Š	š	Ý	ý	■	■	Þ	þ		Ž	ž	■	■
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
■	■	■	½	¼	¹	¾	³	²	¦	−	×	■	■	■	!	"	#
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5
54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G
72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107
Z	[	\	]	^	_	`	a	b	c	d	e	f	g	h	i	j	k
108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}
126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
~	■	Ä	Å	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ã	å	ç	é	è
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161
ê	ë	í	ì	î	ï	ñ	ó	ò	ô	ö	õ	ú	ù	û	ü	†	°
162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179
¢	£	§	•	¶	ß	®	©	™	´	¨	≠	Æ	Ø	∞	±	≤	≥
180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197
¥	µ	∂	∑	∏	π	∫	ª	º	Ω	æ	ø	¿	¡	¬	√	ƒ	≈
198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215
∆	«	»	…		À	Ã	Õ	Œ	œ	–	—	“	”	‘	’	÷	◊
216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233
ÿ	Ÿ	⁄	¤	‹	›	ﬁ	ﬂ	‡	·	‚	„	‰	Â	Ê	Á	Ë	È
234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251
Í	Î	Ï	Ì	Ó	Ô		Ò	Ú	Û	Ù	ı	ˆ	˜	¯	˘	˙	˚
252	253	254	255														
¸	˝	˛	ˇ														

8-bit character sets: international chaos

If all the digits in a byte may be used, 256 entities can be enumerated. But which? There are many national character sets to accommodate, and a number of different 8-bit standards have arisen. Furthermore, vendors such as Apple, Microsoft and Adobe have defined their own character encoding schemes, or in some cases several.

- **ISO 8859** defines several 8-bit character sets built on top of 7-bit ASCII: four alternative Latin sets, plus Greek, Hebrew, Arabic and Cyrillic set.
- **Adobe** have defined four encoding schemes for their fonts. Two are alphanumeric: **StandardEncoding** and **ISOLatin1Encoding**. There is also encoding for Symbol and Zapf Dingbats.
- **Apple Macintosh:** Apple associates one character set with each of its script systems. There is a Roman set which is an extension of ASCII and this same encoding is used to organise use of Symbol and Zapf Dingbats. Amongst the other 8-bit sets there are two Greek sets, a Hebrew, an Arabic, a Devanagari and a Thai set.
- **Microsoft Windows** has 8-bit character sets for ANSI, Turkish, Eastern European, Cyrillic, Greek, Arabic and Hebrew.
- **PC Code Page** mappingsprovide ten alternative encodings for Latin scripts, plus modern Greek, Arabic and Cyrillic, and Katakana (Japanese) and Hangul (Korean) syllabaries.

Extending the repertoire with 'expert sets'

Typographers working in languages with latin character sets at first sight appear well served by Adobe Standard Character Encoding. But demand for true small capitals, case fractions, ligatures and other niceties of traditional typography shows this is not true. An intermediate solution is for the type vendor to supply a matched 'expert set' font. Below compare the contents of standard **Monotype Plantin** with the corresponding weight of **Plantin Expert**.

Plantin

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
	abcdefghijlmnopqrstuvwxyz
Currency marks, numerals	£$¢¥¤ & 0123456789
Accented characters, etc	ÄÅÀÃÂÁ ÉÊËÈ ÍÎÏÌ ØÖÕÓÔÒ ÜÚÛÙ Ÿ
	áàâäãå ç éèêë íìîïı ñ óòôöõ ß úùûü ÿ
Ligatures, etc.	ﬁ ﬂ Æ Œ æ œ
Punctuation	.,;: … ' " ‘ ’ “ ” ‚ „ ‹ › « » !?¡¿
Brackets, dashes, ordinals	< > ( ) [ ] { } /⁄|\ - – — _ ª º
	* # % ‰ • ° ^ ~ † ‡ § ¶ ƒ
These characters may be supplied by Plantin, or referenced from Symbol	+ ÷ ± = ® © ™ ¬
These characters are referenced at all times from the Symbol font	≠ ≈ ∞ ≤ ≥ µ ∂ ∑ ∏ π Ω ∆ ∫ √ ◊
Floating accents	` ´ ¨ ˆ ˜ ¯ ˘ ˙ ˚ ¸ ˝ ˛ ˇ

Plantin Expert

True small capitals	ABCDEFGHIJLMNOPQRSTUVWXYZ
Accented true small capitals, etc	ÁÀÂÄÃÅ Ç ÉÈÊË ÍÌÎÏ Ł Ñ ÓÒÔÖÕ ÚÙÛÜ Š ÐÞ Ý Ž
Ligatures and punctutation	ﬀ ﬁ ﬂ ﬃ ﬄ ÆŒ . , ; : - – — ! ¡ ? ¿ . ..
Non-lining numerals	1234567890
True case fractions	¼ ½ ¾ ⅛ ⅜ ⅝ ⅞ ⅓ ⅔
Superscript and subscript numerals, co-ordinated punctuation etc., and ordinal letters	1234567890 $., () 1234567890 abedilmnrst $., ()° -
Currency marks, floating accents	$ ¢ ¢ ₡ Rp 1 & ˆ - ` ¨ ˇ ¸ ˛ ˘ ¯ ˙

Are expert sets the answer?

Expert sets are obviously *an* answer, and people buy them. But there is cause not to be contented with this answer…

- ◆ **Expert sets are pernickety things to use**. This paper is set in Plantin and Plantin Expert. Every time I want to use the expert set, which I do for numerals and acronyms, I must remember to change font, then change back again. Fortunately for my sanity I use FrameMaker DTP software, and have set up the character formats 'Expert' and 'Acronym' to assist me.*

  However, I cannot be bothered to format every ﬂ, ﬃ and ﬄ ligature in this document using the expert set! I suppose I could do so with search-and-replace, though.

- ◆ **Expert sets differ in their organization**. This is inevitable, since they differ in content. Sometimes this is because a type vendor may notice the need for a character which others overlooked; for example, the **fj** ligature in Carter & Cone Galliard. Sometimes this is because different typefaces require different extensions—Helvetica neither needs not desires a swash terminal **e**.

- ◆ **There is no clear (or sensible) standard encoding**. Adobe have a semi-standard encoding for expert sets, and Plantin Expert appears to abide by this, but it is impossible to enforce for the reason just stated. Non-lining numerals and small capitals are logically ordered so that converting lining numerals and lower-case letters in the base font will produce them. But by what logic is the ﬄ ligature in the location of the letter Z?

- ◆ **Expert sets provide typographical, but not linguistic extension**. But of course, that's what expert sets were invented for. They still do not include an Eastern European latin character set.

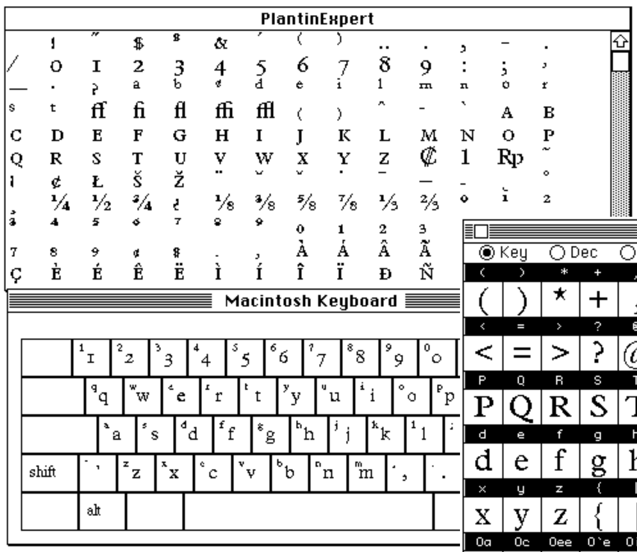
  It's nice to have the Icelandic/Faroese/Old English letters ᴅ (*eth*) and ᴘ (*thorn*) at D and π respectively, but not when you can't get the ordinary upper and lower case *eth* and *thorn* in the base Adobe Standard Character Encoding.

- ◆ **Expert sets cause text processing anomalies**. Use a ligature or swash form, and your spelling checker won't work on the word, nor will [Find…] find it. If you export text for use which involves conversion to other fonts, strange characters will appear.

  (To get technical, **glyph substitution** can be effected only by changing the character identity in the **backing store**.)†

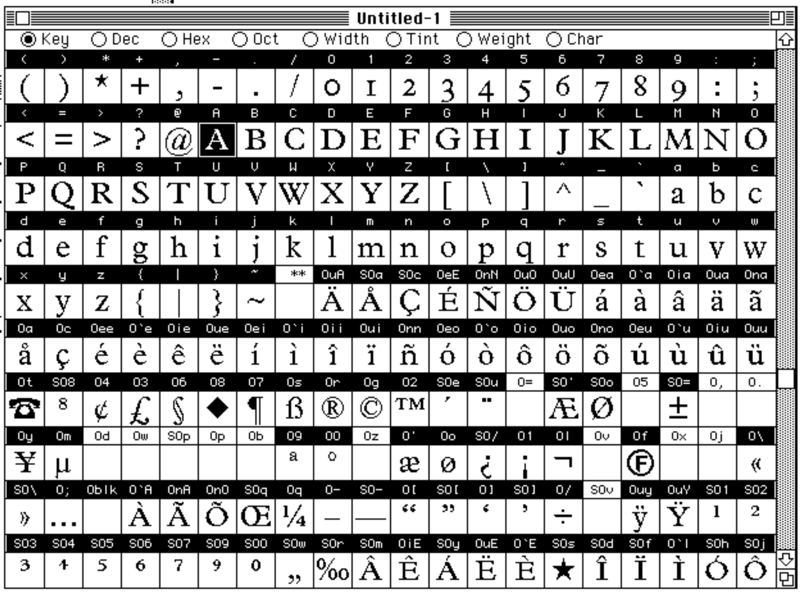
* FrameMaker appears unique amongst DTP programs in having a Character Formats catalogue, a style palette for formatting text *below* the paragraph level.

† In contrast, FrameMaker and QuarkXPress can be set to *display* and *output* the fl and fi ligatures in an Adobe Standard Character Encoded font automatically, without changing the backing store. See also QuickDraw GX below.



FontMixer

Left: Monotype's application lets you mix characters from a number of PostScript Type 1 fonts, assign them to the keys of your choice and save the result as a new font.



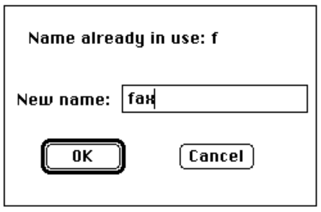
Right: a display of the character array within Fontographer shows the result of the font-mixing experiment described below on this page.

FontMixer— a DIY solution?

Monotype have released *FontMixer*, an application which lets users create a single 8-bit font containing a range of characters of their choice from a number of PostScript fonts, and to assign those characters to whichever keys they wish.

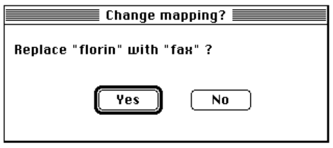
As an example, I can take the character set from Monotype Plantin, but perform the following changes:

- replace the lining numerals 0123456789 with the non-lining numerals 0123456789 from Plantin Expert;
- transfer from Plantin Expert the ½ and ¼ characters and place them mnemonically at [Option–h] and [Option–q], and transfer all the Plantin Expert superior numerals to the [Option–Shift] numeral keys;
- and finally add some dingbats from Zapf Dingbats, plus a 'fax' ideogram of my own design.


Name already in use: f

New name: fax

OK Cancel


Change mapping?

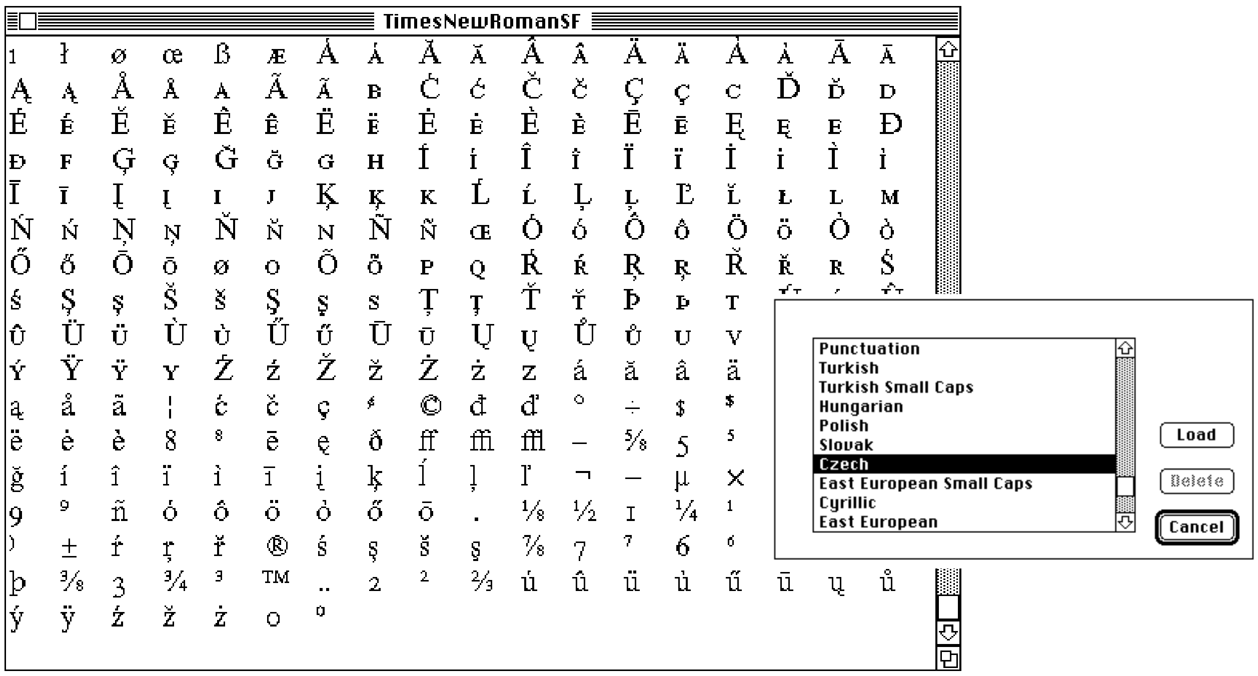
Replace "florin" with "fax" ?

Yes No

I generate a new PostScript Type 1 font called 'IdeoGraphy Plantin', install it, and the result is:

Í a 3 ˙ –inch disk from Monotype Typography⁄°
† 0737 765959
f 0737 769243

…all done without changing font.



Above, a partial display of the 440 characters in the extended latin character set of **Monotype Times New Roman Superfont**, which is supplied with FontMixer. Compound accented characters can be assembled using FontMixer to produce a font to suit a particular need in multilingual setting, e.g. for Estonian or Czech. The inset display shows a list of the **Templates** supplied with FontMixer to accelerate loading defined sets of characters into the correct locations. Users can also create their own templates.

Superfonts and character sets

It is perfectly possible to have PostScript Type 1 fonts with hundreds or even thousands of characters; it's just not possible at present to access all these characters.

Monotype are launching a series of **Superfonts**, which deliberately include many more characters than normal, to make it easier for FontMixer users to put together a font to meet their own needs. A wide number of compound vowels with diacritical marks are supplied, as shown above—including small-capitals forms.

So, are custom-mixed Superfonts the answer?

As I commented about expert sets, Superfonts and FontMixer are *an* answer to many troublesome problems, and provide a way around the limitations of standard 8-bit encoding in particular.

But precisely because FontMixer encourages *custom* solutions, it only partially addresses the need for larger character sets and a way to access them. If anything, 'fontmixing' is likely to increase the chaos around character encoding and to make it more difficult for people to share documents using non-standard character sets.

I believe that the solution is to move towards 'multi-byte' methods of encoding character sets, and type fonts,* and in devising easier ways of accessing them. But that in turn brings fresh dilemmas…

* …and therein lies a tale. Character sets and type fonts are not the same thing, as we shall see.

Multi-byte character encoding

If using a single byte to access characters in a computer-resident character set is so restricting, why not use more than one byte? After all, microprocessor chips have progressed from the 8-bit chips of the 1970s, to 16-bits, then 32-bit chips with 16-bit connections, then chips handling a full 32 bits inside and out, and now we have chips like the DEC Alpha which is 64-bit. So why can't character encoding become more capable in like manner?

Adding a second byte to reference each character in a text file does double the storage requirement, but it does not merely double the size of the possible character set. It multiplies it by two *eight times*. With two bytes, you can enumerate 65,536 entities.

Not only is that much larger than any character set in the known galaxy (Chinese has about 16,000 ideograms); it may provide space enough for every character in every language system in the world to have its own *unique* reference number.

In fact, two-byte addressing is already in use to handle the large character sets of Chinese, Japanese and Korean, and Asian national standards for two-byte character sets exist. But is a global system possible? And how would it work?

The Unicode initiative (and ISO DIS 10646)

Currently discussions are under way to harmonize two emergent standards into one. The International Standards Organisation has a committee (ISO JTC1/SC2/WG2) which has been developing a 32-bit (four-byte) character encoding standard, ISO DIS 10646. (The 'DIS' part indicates a *draft* standard.)

Meanwhile, the Unicode Working Group came into being as an informal co-operation between Apple and Xerox engineers who worked on multilingual operating systems. They set about drafting **Unicode** as a 16-bit (two-byte) encoding standard for information interchange. That project has become the Unicode Consortium, incorporated as *Unicode, Inc.*, with representatives from other companies concerned with handling international text files.*

The Unicode Consortium issued drafts in September 1989, October 1990 and December 1990, and *The Unicode Standard: Worldwide Character Encoding Version 1.0, Volume 1* was published in two volumes during 1991.

* The organizations involved include as Adobe, Aldus, Borland, Claris, GO (the PenPoint people), IBM, Lotus, Metaphor, Microsoft, NeXT, Novell, The Research Libraries Group, Sun Microsystems and WordPerfect

During 1991 an attempt at the ISO to approve ISO DIS 10646 as a full standard was voted down; most of the disapproval from ISO member countries reflected a desire to merge ISO DIS 10646 and Unicode into a single standard. This is currently being worked on, within the framework of making Unicode '*a two-byte subset of a canonical 4-byte international standard encoding*'.

Some characteristics of Unicode

- Unicode is intended to be as universal as ASCII.
- Unicode encodes alphabetic and typographic characters from the world's scripts, plus technical symbols in common use, all handled identically.
- A fixed-width encoding of 16 bits is used for all characters; no escape sequences are necessary, making it easier for software to parse encoded texts.
- The usual way of writing the numerical codes for Unicode is hexadecimal, from 0000 (decimal 0) to FFFF (decimal 65,535).
- Character content of the Unicode standard is derived primarily from existing standards.*
- The first edition of *The Unicode Standard* contains over 28,000 characters from the world's scripts, including over 20,000 unique characters defined by the standards of China, Japan, Taiwan and Korea. There are over 30,000 unallocated positions.
- The alphabetic scripts in Unicode 1.0 include—in this order—all latin scripts, general diacritical marks, standard phonetics (IPA), Greek (with Coptic additions) extended Cyrillic, Armenian, Hebrew, Arabic, Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telegu, Kannada, Malayalam, Thai, Lao, Tibetan and Georgian.
- Work is in hand to encode Ethiopian, Burmese, Khmer, Sinhala and Mongolian; other scripts being considered are Inuktikut & Cree, Egyptian Hieroglyphics, Cuneiform, Cherokee, Maldivian, Syriac and Glagolitic.
- A Symbols zone includes punctuation, superscripts and sub-scripts, currency, diacritics, letterlike forms, number forms, arrows, mathematics, miscellaneous technical, form and chart components, geometrical shapes, miscellaneous dingbats and (for compatibility reasons) Zapf Dingbats.

* Sources include: ISO International Register of Character Sets; ISO 8879 (SGML); bibliographical standards; important national standards such as ISCII 1988 (India), GB 2312-1980 (China), JIS X 0208-1990 and JIS X 0212-1990 (Japan), CNS 11643-1986 (Taiwan); draft standards such as ISO DIS 6861.2 on Glagolitic, Old Cyrillic and Romanian Cyrillic; code pages and character sets from Adobe, Apple, IBM, Lotus, Microsoft, WordPerfect, Xerox etc.; and papers contributed to the ISO SC2/WG2 committee on character encoding.

- The punctuation zone is well thought out, including different kinds of space, such as a zero-width wordbreak space to aid word boundary recognition in languages like Thai and Japanese which have no wordspaces. There is explicit encoding for non-breaking spaces and non-breaking hyphens, which currently are recorded only by private encoding schemes inside publishing software. Likewise, unambiguous codes are included for forcing a new line or a new paragraph.
- Where does it end? '*Unicode…does not encode rare, obsolete,*[*] *idiosyncratic, personal, novel, rarely exchanged or private characters, nor does it encode logos or graphics*[†]*… Graphologies unrelated to text, for example, musical and dance notations, are outside the scope of the Unicode standard.*'

Some dilemmas: character ≠ glyph!

You are going to hear the word '**glyph**' used more and more in discussions of typography. This is not because the term 'character' has suddenly become old-fashioned, but because glyphs and characters are not the same thing, nor do they always map to each other in a one-to-one correspondence.

What is the difference?

A **character** is defined in Unicode as '*the smallest component of a written value that has semantic value.*' [‡] The letter **a** is a character, and it is the same character regardless of what font it is rendered in. Typically, someone asked to spell a word in a language which uses an alphabet or syllabary would spell it out character by character, and in a logical character sequence.

A **glyph** is harder to define, for its definition is controversial. It's clear that a glyph is a single rendered character image;[**] several alternative glyphs could represent the same underlying character. But can **a** and **a** be described as different glyphs because one is Helvetica and one Frutiger? That is not helpful; it is more sensible to describe these as being the same glyph in different fonts.

However, it may be sensible to describe **a** and **a** as different glyphs, because they represent essentially different 'topologies': a hooked-top form and a ring-bodied form—and particularly because some fonts have the two as alternatives, as in Monotype's special versions of Gill Sans for young children's books. That's the acid test.

* But note Egyptian Hieroglyphics and Cuneiform are considered important for scholarship, and are likely to be included.

† But Zapf Dingbats sneaked in for compatibility reasons.

‡ From the Glossary of *The Unicode Standard version 1.0*. It continues: 'Character refers to the abstract idea, rather than a specific shape (see also glyph), though in code tables some form of visual representation is essential for the reader's understanding.'

***Single rendered character image* is not the same as *rendered image of a single character*!

Problematic glyph-to-character mappings

◆ The expression ‘5½’ would be conveyed orally as ‘five-and-a-half’; there is a strong case for considering the ½ as denoting a single *character*. Using an expert set, the ½ can be rendered with a single *glyph*. But if an expert set is not available, the single character (in Unicode `half, fraction one` which is character `00BD`) may be rendered with *three* glyphs: a superscript 1, a fraction-slash, and a downsized 2.

◆ What should we make of the word ‘**soufflé**’? Look at it closely. Ask me to spell it and I’ll say ‘*s–o–u–f–f–l–*[*e-acute*]’. How is it rendered? The three *characters* ‘f–f–l’ have become a single **ffl** ligature *glyph*. And what about **é**? Is that one glyph, or the glyph [**e**], plus a non-spacing diacritic acute accent glyph[´] ?

◆ Think *you’ve* got problems? An Arabic consonant can take one of four different glyphic forms depending upon whether it is **initial**, **medial** or **final** in a word, or **isolate**. Need each of these be represented by a separate character in the backing store?[*] Or should the system record only the desired consonant as a child would spell it, and work out for itself how to represent this *character* as one of a number of alternate *glyphs* when the character must be rendered in context?

◆ Consider ligatures again. There are many cases where two or three characters can be represented with the same number of glyphs in a one-to-one mapping, but æsthetic[†] considerations prefer multiple *characters* to be represented with a single *glyph*.

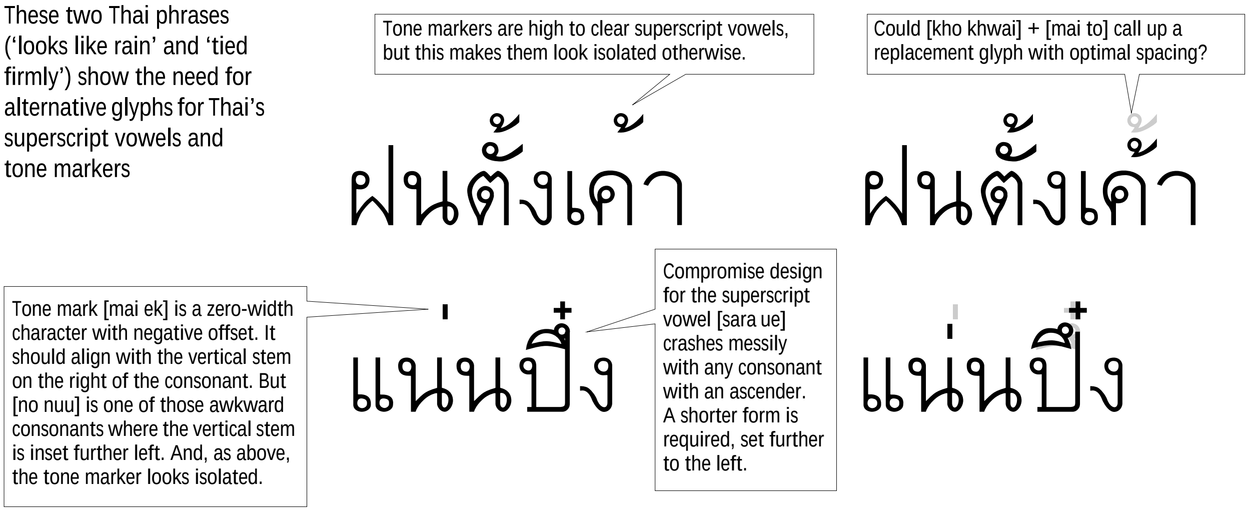
Most Indic scripts add non-spacing vowels (and/or tone marks and other diacritics) above or below base consonantal characters. It isn’t always best to create Indic vowels and tone marks as superimposed non-spacing glyphs, positioned backwards from the cursor insertion point, because of the varying widths and features of the Indic base characters over or under which they are to be placed.

Fussy Thai type-designers may create as many as four alternate versions of each tone mark to fit snugly above consonants or consonant–plus-superscript-vowel combinations. Should Thai typographers continue to have to remember how to call up these alternate tone marker forms? Or should they be able to type the standard Thai character order, registering a procession of consonant, vowel and tone-marker *characters* in the backing store, and have them transformed by the system into optimal compound *glyphs*?

* Backing store: in a computerized text processing system, the backing store records the characters requested in logical order. The relationship between backing store representation and glyphic representation is not necessarily one-to-one.

† Oops! Was that æ a single character, or two characters mapping onto a single glyph?

Glyph replacement therapy?


Tone markers are high to clear superscript vowels, but this makes them look isolated otherwise.
Could [kho khwai] + [mai to] call up a replacement glyph with optimal spacing?
ฝนตั้งเค้า
ฝนตั้งเค้า
Tone mark [mai ek] is a zero-width character with negative offset. It should align with the vertical stem on the right of the consonant. But [no nuu] is one of those awkward consonants where the vertical stem is inset further left. And, as above, the tone marker looks isolated.
Compromise design for the superscript vowel [sara ue] crashes messily with any consonant with an ascender. A shorter form is required, set further to the left.
แน่นปึ๋ง
แน่นปึ๋ง

These two Thai phrases ('looks like rain' and 'tied firmly') show the need for alternative glyphs for Thai's superscript vowels and tone markers

Unicode's dilemmas

This problem is affecting the compilation of the Unicode standard. Unicode hedges its bets by encoding many letters with diacritic marks as characters in their own right…but also encodes the diacritic marks. And despite the insistence that Unicode encodes characters *not* glyphs, the Zapf Dingbats set consists of precise unique glyphs.

Characters, glyphs and keystrokes

A further complication is the keystroke sequence used to call up members of a large character set from a computer keyboard.

Take the Macintosh character set. If you use the UK keyboard resource, the character **ä** is accessed by typing [option-u] + [a]. The keyboard resource directs the computer to intercept the two successive **scan codes** from the keyboard and map them to the single character **ä.** That single character is then output as a single glyph.

However, if you use the Swedish keyboard resource, a single scan code is converted to that character, represented by that glyph.

Some quite complex procedures have to happen behind the scenes for languages with huge character sets. The standard way of typing Japanese is to spell all words using the **katakana** syllable character set, and have the computer convert groups of katakana to a single **kanji** ideographic character where this is the preferred form.

Chinese has no syllabary, and so is often typed using the Pinyin latin transliteration. Thus typing the transliteration *zhong* would give a numbered choice of ideographic characters, from which you could choose 中 (the character meaning 'centre'), which is the first part of *zhongguo* which means 'China' (literally 'centre country').

QuickDraw GX and glyph substitution

Type technology marches on, and soon the Macintosh operating system will acquire a more capable imaging model: **QuickDraw GX**. It is a year since I saw it demonstrated at Seybold San Francisco but it is still relatively difficult to find out exactly how it will work.

QuickDraw GX is an extension of the existing Mac imaging model. It should improve graphics, printer control and typography. Kerning, tracking and justification will now be available at the System level, usable by all applications which make calls to the new tools.

At Seybold, Apple staff demonstrated how QuickDraw GX could do automatic glyph substitution on the fly. Suppose you type '**raffle**'. You type [r + a + f] and they are represented by three glyphs. But as soon as you type another [f], the display changes to '**raff**', showing the two [f] characters with the **ff** ligature. Typing an [l] transforms the display yet again, to the three-glyph form **raffl**. And so on.

Yet in the backing store the six-character sequence spelling 'raffle' will still be there, so you can still select and edit within the ligature (so to speak). Spelling-checking works, and hyphenation rules apply.

TrueType GX, Type 1 GX

To be able to take advantage of glyph substitution, you need more resources in the font; not only replacement glyphs, but also rules for what may substitute for what, and under what circumstances.

Obviously this will be different for each font. For example, the font may include a special swash form of the **e** which may be used only at the ends of words. In such a font, typing the space after 'raffle' will cause the terminal swash form to be expressed. Removing the space will suppress it in favour of the normal form.

Adobe and Apple are working on extended versions of TrueType and PostScript Type 1 fonts called 'TrueType GX' and 'Type 1 GX'.

It seems appropriate for typographers to be both excited and apprehensive of this new technology. Questions which occur to me are:

◆ Would ligature substitution be sensitive to the effects of letterspace adjustment, for example in type justification? Ligatures don't stretch, so they make letterspace-adjusted type look odd.

◆ How would typographers be able to turn clever behavioural features of type fonts on and off?

◆ Will I need a degree in computer science to be a future Adobe Garamond GX user? Will a 175-page manual be suplied with each font?

Perhaps we will also experience 'Gillette economics': the computer will cost £600, the dtp program £700, and each smart font £1000!

MULTIPLE MASTERS: *chameleon type*

No discussion of current type technology would be complete without describing Adobe’s Multiple Master format.

Multiple Master fonts can be transformed from one appearance to another by **interpolation** between outline forms. Each Multiple Master includes at least two sets of outlines. Duplicate characters are digitized with the same number of Bézier anchor and control points, in the same functional location and sequence, but with different co-ordinates. A specific ‘instance’ of the font can be generated with Bézier co-ordinates somewhere *between* these defined extremes.

(Interpolation between outlines is not new: it is used by the Ikarus type production system. But this is the first time interpolation has been put in the hands of end users, to generate new fonts on the fly. Multiple outlines in one PostScript font is not new either. Type 1 format permits **hybrid fonts**. Adobe Optima is an example: one set of outlines is used at low resolution, another at high resolution.)

What do Multiple Masters make possible?

Multiple Master fonts may be described as having ‘axes’. A two-axis font could generate variations from ultralight to ultrabold along one axis, from condensed to expanded along another. A specific instance could then be generated for a moderately condensed semibold font.

Adobe have demonstrated that three-axis fonts can restore to typography the phenomenon of ‘optical scaling’. When type designs are optically scaled, letterforms are adjusted for different sizes to make the design of each size more appropriate. For a contrasty modern serif face like Walbaum, optical scaling might be implemented thus:

- at large sizes, there could be extreme contrasts between thick and thin strokes; at small sizes, contrast would be reduced, and thin strokes strengthened to make them easier to see (and to print);
- x-height may be smaller at large sizes, larger at small sizes;
- at small sizes, serifs may be strengthened;
- setwidths might be subtly adjusted to be wider at small sizes;
- tight counters, as in the upper part of **e** could be made more open at small sizes;
- sidebearing spaces might be small at large sizes to produce tighter letterfit, large at small sizes to aid character recognition.

TYPE IN FLIGHT: *Acrobat & PDF*

In recent years, type portability has become a major obstacle to the electronic exchange of visually rich documents. Adobe's Acrobat technology is an attempt to solve this with the aid of their PostScript language and Multiple Master font format.

What is electronic document interchange?

Firstly, what is **electronic publishing**? It's commonly used to mean the use of computers to prepare documents for publishing *on paper*. Another use of the term implies using the computer itself as the publishing medium, with 'documents' distributed on disk or via networks, and viewed primarily on computer screens.

A 'document' in this sense is not necessarily designed for paper output; it may not even be paginated. Various electronic newsletters are distributed via the Internet, for example, and are simply long text files in 7-bit ASCII split into lines no more than 80 characters wide.

Such files are very information-poor and offer little incentive to save forests! There are several ways in which they could be improved:

- a wider character set should be supported, yet the files must still be transportable through existing networks;
- many benefits could be obtained if the structure of the information could be encoded, perhaps enhanced with hypertext links;
- preserving the typographic formatting and layout of documents would put readers at ease and assist communication;
- graphics should be supported.

How does Acrobat help?

Adobe Acrobat is a family of applications (Reader, Distiller and Exchange) which let a computer user prepare a document for electronic distribution using any application which can produce PostScript output. That PostScript output is converted to a special interchange format called **Portable Document Format** (**PDF**).

The PDF file is written entirely in 7-bit ASCII so that it will pass safely through all of today's data highways.

A recipient of a PDF file who has the Reader or Exchange software can open that file and read it or print from it, and will see it with all of its graphics and formatting. Either the same fonts in the original

The referenced graphic file for this page has been damaged and can neither be viewed nor printed. Another screen dump file will be generated to replace it in the revised edition.

Acrobat Exchange is shown being used to view a PDF file generated from the cover of a training manual for a GeoNet electronic mail service; the manual was edited, formatted and illustrated and partly written by Conrad Taylor. The page design was made in PageMaker and includes four Encapulated PostScript graphic files, all of which have been written into the PDF file and compressed. A recipient needs neither PageMaker, nor the fonts, nor the graphic files to view and print the document.

will be seen, or an adequate representation of them using instances of Multiple Master fonts, *even if the recipient does not have those fonts installed on his or her computer.*

What is PDF?

Portable Document Format is not PostScript, but it is based on it and represents text and graphics in a PDF file using the imaging model of PostScript. PDF is fully described in Adobe’s *Portable Document Format Reference Manual* and will also be discussed further at this conference, so I shall limit my description to a few key points:

- PDF is not as flexible as PostScript, but is optimized for speed of page rendering, document navigation, text searching and so on.
- PDF supports a number of compression filters to keep files small. Colour and greyscale images may be compressed with JPEG[*]

[*] JPEG = Joint Photographic Experts Group. This standard compression algorithm lets the user choose the desired trade-off between compression ratio and the decompressed image’s fidelity to the original.

methods; line art can be run-length encoded or compressed using CCITT Group 3 or Group 4 methods.* The general LZW (Lempel–Ziv–Welch) compression algorithm can be used to compress text as well as graphics.

- ◆ Font independence is achieved by writing a **font descriptor** into the PDF file. In some instances this fully embeds the PostScript definition of the font, which travels with the document; in other instances only the name and metrics of fonts are recorded.

Font descriptors and Multiple Masters

Below can be seen a **font descriptor** and **font resource** retrieved from a PDF file from page 18 of this document. As can be seen, the identity and general metrics of the font are recorded in the font descriptor, and the widths of all the characters in the font resource.

```
8 0 obj
<<
/Type /FontDescriptor
/FontName /Frutiger-Light
/Flags 48
/FontBBox [ -152 -216 1000 911 ]
/MissingWidth 250
/StemV 63.00
/StemH 63.00
/ItalicAngle 0.00
/CapHeight 698
/XHeight 510
/Ascent 750
/Descent -210
/Leading 0
/MaxWidth 0
/AvgWidth 0
>>
endobj
9 0 obj
<<
/Type /Font
/Subtype /Type1
/Name /F9
/BaseFont /Frutiger-Light
/FirstChar 0
/LastChar 255
/Widths [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
278 389 556 556 556 1000 667 278 278 278 556 600 278 333 278 278
556 556 556 556 556 556 556 556 556 556 278 278 600 600 600 500
800 667 556 667 667 500 444 722 667 222 333 611 444 889 667 722
500 722 556 500 500 667 611 944 611 611 500 278 278 278 600 500
222 500 556 444 556 500 333 556 556 222 222 500 222 833 556 556
556 556 333 389 333 556 444 778 444 444 444 278 222 278 600 0
667 667 667 500 667 722 667 500 500 500 500 500 500 444 500 500
500 500 222 222 222 222 556 556 556 556 556 556 556 556 556 556
556 400 556 556 556 500 600 556 800 800 990 222 222 0 889 722
0 600 0 0 556 556 0 0 0 0 0 325 361 0 833 556
500 389 600 0 556 0 0 556 556 1000 0 667 667 722 889 889
500 1000 556 556 278 278 600 0 444 611 167 556 278 278 556 556
556 278 278 556 1000 667 500 667 500 500 222 222 222 222 722 722
0 722 667 667 667 222 222 222 222 222 222 222 222 222 222 222
]
/Encoding /MacRomanEncoding
/FontDescriptor 8 0 R
>>
endobj
```

* Compression methods used in fax transmission.

When the PDF file is read on the recipient's computer, the Acrobat Exchange or Reader software examines these font descriptions. If the font is already installed on the recipient's computer, Acrobat uses that to display the document.

But if the font used in the document is not installed, a specific instance of a Multiple Master font will be created to *simulate* the font used. At least the information provided in the font description will let Acrobat create an instance in which each character is rendered with the correct set-widths, and the weight and the degree of italicization of the font will also approximate to the original so that the key formatting differences and contrasts are preserved.

The types of font which Acrobat can measure and simulate in this way include Type 1 and Type 3 PostScript, TrueType, and specific instances of Multiple Master fonts.

When Acrobat embeds fonts in files

Naturally it would be rather silly to expect a Multiple Master font with a latin character set to simulate a Bengali font!

Acrobat applies a test: it expects that Multiple Master Type 1 fonts will be able to substitute for fonts in a document only if they use **Adobe Roman Standard Character Encoding**. And if fonts in a document have a non-standard encoding? Then Acrobat embeds a full description of the PostScript font in the font descriptor so that it travels with the PDF file.

(Only Type 1 fonts can currently be embedded in PDF files, and the two most commonly used Type 1 fonts with non-standard encoding, Symbol and Zapf Dingbats, are supplied to all purchasers of Acrobat anyway and so are never added.)

PDF: issues for typographers

If PDF and similar interchange technologies for electronic documents really take off, typographers who have hitherto aimed to produce good-looking *paper* documents will have to learn how to choose type for the best display on a monitor, how to use monitor colours with type, and so on.

Well — whatever curve you use to describe it, Type's Trajectory **has certainly been an interesting one since the Digiset crawled out of Hell, and Ikarus tested its wings.**

I hope this Workshop and conference prove that there are still many interesting and turbulent patches of typographic technology to be traversed in future.

Please fasten your safety belts and extinguish your qualms!

GLOSSARY: *some terms in type technology*

Acrobat

An Adobe Systems technology for assisting the electronic interchange of formatted documents, independently of the computer operating system, application software, fonts or graphics used to create the document.

Acrobat converts files into **Portable Document Format**, (PDF) which is based on the PostScript page imaging model but which is optimised for performance and allows users to add annotations, 'bookmarks', hypertext links etc.

PDF files contain **font descriptors** which record fonts used in a document by name plus their character metrics and style information. If the recipient of the file does not have a font used in the creation of the document, the Acrobat Exchange or Reader program simulates its appearance using a font made up as an instance of a **Multiple Master Font**.

(However, this is not true for fonts which do not have Adobe Standard Character Encoding; Acrobat handles these cases by embedding the full PostScript font information in the font descriptor so that it is distributed with the document.)

Contrast the document interchange approach of **SGML**.

Adobe Standard Character Encoding

An 8-bit **character encoding** scheme based on **ASCII**, defined by Adobe Systems and used by most of their latin-script fonts, except for expert-set fonts for which other encoding schemes exists. Many other type manufacturers produce their faces with the same encoding, which has become a de facto standard in desktop publishing.

Aliasing

The process of representing a continuously variable shape in a form consisting of discrete steps, as when the curves of type must be represented as a bit-mapped display.

Type aliasing causes problems at low output resolutions, when the bit-mapped display often compromises observers' expectations of the correct appearance of type. Stems may be of unequal weight, delicate curves may be rendered as jagged stair-steps, rounding errors may make some characters appear too large or too small.

Anti-aliasing:

Any strategy for improving the appearance of the digital display of an analogue form. For instance, instead of drastically rounding a marginal pixel to either black or white, it can be rendered at an appropriate level of grey, or switched rapidly on and off to simulate grey.

Anti-aliasing is not practical for printing devices except film recorders for continuous tone photographic film, and dye sublimation printers; it is sometimes used on computer screens and for TV captions.

ASCII

The American Standard Code for Information Interchange is the US national variant, **ANSI X3.4-1986**, of the ISO character encoding standard **ISO 646**.

ASCII is an agreed convention for representing alphanumeric characters by means of binary numbers. Thus 01001010 stands for J—regardless of the computer used. The existence of ASCII as a standard is important for file transfer and communications.

People talk of '**7-bit ASCII**' and '**8-bit ASCII**'. In fact, only the 7-bit variant (control codes plus 85 alphanumeric and punctuation marks) is truly standard. Using the eighth binary digit in the byte ('8-bit character addressing') doubles the character set, but different computer systems and national industrial standards use the upper-range differently.

(Note: some mainframes, notably IBMs, use another character encoding scheme, EBCDIC. The ISO 646 set is common to both and therefore safest of all.)

ATM, Super ATM

Adobe Type Manager: software for Macintosh and Windows computers which extends the computer system's capabilities so that PostScript Type 1 fonts will be rendered on screen from outline information rather than from the bit-mapped screen fonts which would otherwise be used. ATM also works with printer driver software for non-PostScript printers and faxes, and rasterizes the fonts at the printer resolution.

Super ATM is a development of ATM to work with **Multiple Master Type 1** fonts. Amongst other capabilities, this allows fonts which are missing from the system to be simulated by an instance of a Multiple Master Font.

Backing store

In text processing systems, the characters stored on disk or computer memory, distinct from how they appear on screen or paper. In some non-latin text processing systems, the order in the backing store may differ from the visually presented order.

Bit

Short for 'Binary Digit'—fundamental unit of record-keeping in a computer. The binary numbering system uses base-two notation, so only two kinds of digit are needed, 1 and 0. These can be represented by on and off states in electronic switches.

Bit-map; bit-mapped font

A bit-map is a two-dimensional array of pixels set to either 'on' or 'off' (black/white) and stored as an equivalent array of binary digits in the memory of a computer.

Some typesetting systems store each font character as a bit-map matrix grid. Others store fonts as outlines, and scale and convert them to bit-maps as required. See also **byte-map** for grey-scale and colour displays.

Built fraction

A fraction character created in a type composition program by assembling it from numerals and slash marks; as opposed to a **case fraction**.

Byte

A byte is an eight-digit binary number, such as **00010010** or **10100011**. A byte can represent a block of data such as a alphanumeric character. Memory capacity is measured in multiples of bytes e.g. **kilobyte** and **megabyte**. There are 1024 bytes in a kilobyte, 1024 kilobytes in a megabyte. (The next two levels up are **gigabyte** and **terabyte**.)

Byte-map

As an extension from the term **bit-map**, a byte-map is a matrix array of pixels representing an image or character with more shades of grey than purely black and white. Anti-aliased type characters must be generated as a byte-map.

Cap height

The space occupied vertically by a typeface measured from its baseline to the top of the capital letters.

Case fraction

A fraction which comes 'ready built' in the typesetting system. Rarely available in desktop publishing, where **built fractions** must be used, made of separate components; but Expert Set fonts typically include case fractions.

Character

The smallest component of a written language, e.g. a letter, numeral or punctuation mark. When we talk of a 'character' we mean its abstract identity, such as 'question mark', rather than a particular displayed shape for it (for which see **glyph**).

Character encoding

A scheme which associates a unique number with each character in a set of characters, to facilitate machine processing of text. Many character encoding schemes are national or international standards, e.g. ASCII, Unicode. Some are proprietary, such as Adobe Standard Character Encoding.

In practical use, the distinction between 'characters' and 'glyphs' may be difficult to maintain. Adobe Standard Character Encoding, for instance, encodes the ***ﬁ*** and ***ﬂ*** ligatures as if they were characters, while Unicode does not.

Character and font metrics

Character metrics means the measurements associated with an individual type character, such as its set-width, and left and right sidebearing spaces. Font metrics also include kerning pair tables, touching tables and so on.

Condensed

Describes a version of a type face narrower than the normal or 'medium' version e.g. this type is Frutiger Condensed.

Copyfitting

Calculations to work out how much space a story will take up when turned into a particular style and size of type.

Counters

Those areas of white space wholly or partially contained within letterforms—wholly in the case of **o**, **a**, **g**, **p** for example, and partially in the case of **c** and **u**.

CRT photosetter

A phototypesetting machine in which the type images are generated on a Cathode Ray Tube (like a TV tube). A system of lenses and mirrors focusses this image onto the photosensitive film or paper for output. Examples: Hell Digiset, Linotron 202

Cyrillic

Script derived from Greek and used for writing various Slavic languages, notably Russian and Bulgarian, and attributed to St. Cyril. The Cyrillic character set has been enlarged to write some of the non-Slavic minority languages of the former Soviet Union.

Descender

That part of a letter below the x-height as in **p**, **q**, **y**.

Desktop publishing (DTP)

Defined by Paul Brainerd of Aldus in 1985 as a microcomputer-based method of preparing matter for printed reproduction, in which text and graphics are made up into pages on screen using special software and the result output to a **laser printer** for camera-ready copy or, in some cases, as the finished piece.

The term DTP is often used more broadly to describe more sophisticated levels of electronic page make-up on screen and full page imaging using photosetting equipment.

Diacritic, diacritical mark

A non-spacing mark which is placed above or below another letter, generally a vowel, and which has the function of modifying its sound. Examples: **grave, acute, circumflex, tilde, macron, breve, diæresis, umlaut, ogonek, hungar-umlaut**. See also **tone marks**.

Digital

Any representation of data in the form of numbers; often contrasted with **analogue**.

Digital type

Type forms stored as numbers in a computerized photosetting or desktop publishing system; either **bit-mapped** or **vector-encoded**.

Dingbat

A typographical symbol such as a bullet, box, pointing finger, tick mark etc., residing in a 'dingbat font'.

Em

Unit of type measurement, originally the width of the lower case 'm' body cast in type metal, which was almost always square. An 'em space' or 'quad' was a piece of type metal which from the top appeared perfectly square. It is necessary to qualify em by the point size being used e.g. a 10-point em; but if none is specified, a 12-point em or pica is generally assumed. An em or pica is almost exactly one sixth of an inch (4.23 mm).

Em dash
A long dash often used parenthetically—like this—and in other situations where a pause would be used in speech.

Em space
A typographic horizontal space the width of an em—the same width as the type is high.

En
Originally the body width of a lower case ‘n’ cast in type metal, and hence a unit of typographical measurement which is exactly half an em (see above). Some printers use the en as the unit for calculating characters in typesetting.

En dash
A character (–) somewhat longer than a hyphen (-) used typically to express a range (as in “10–12 months” or “see pages 23–25”, or to juxtapose words opposite each other, as in “Iran–Iraq War” or “Clinton–Yeltsin talks”.

En space
A typographic horizontal space the width of an en—half the width as the height of the type.

End-of-line decisions
Early typesetting systems required the operator to decide which matter to carry over to the next line, and where to hyphenate. These days computers do this automatically (sometimes badly).

Exception dictionary
In typesetting, a dictionary of words which do not follow the normal rules of hyphenation. Computerised typesetting systems which make hyphenation decisions by a set of rules must have an exception dictionary built in.

Expert Set Font
A companion font containing special characters which cannot be accommodated in the normal desktop publishing font because the ASCII encoding scheme does not allow enough ‘slots’ in which to store characters. Example contents could include swash characters, true small capitals, case fractions, superscript and subscript numerals and characters, special currency symbols, and ligatures.

Family
A related group of type fonts which can be called by a single name, such as the plain, bold, italic, bold italic, condensed, expanded etc. of the type family Helvetica.

Film recorder
An output device for a computer that uses a flying beam of light to record onto photographic film. Often used to describe the desktop devices which incorporate a set of filters and create an image on colour transparency film, to generate high-quality 35mm slides for presentation purposes. Term used also to refer to the part of a phototypesetting system which exposes the film or paper.

Fixed space
A space which does not expand or contract in the process of justification of type.

Floating accents
Accents which can be combined with letters to produce accented characters: thus ¨ can be combined with a to make ä, with u to make ü.

Flush (left or right)
Aligning a column of type vertically either to the left (and ragged right) or, more rarely, to the right (and ragged left). Alternative terms used are range (‘range left’) and quad (‘quad left’).

Font, sometimes fount
(Fount is the older English spelling—pronounced font in both cases.) A range of letters, figures, punctuation etc. in one style of a typeface, e.g. italic or bold members of a type family, but not both in one font. The term dates to the time when all type was individually cast in metal at a type foundry, when ‘font’ also implied a single style and size of type.

Font cache
A temporary memory store for bit-mapped type images, used to improve performance in display systems which rasterize type on demand from font outline information. By rasterizing all characters in the font at the requested size and placing them in a font cache, the system can respond quickly to the need to render such characters subsequently.

PostScript printers generate font caches in printer RAM; Adobe Type Manager reserves an area of computer RAM for a cache; FrameMaker writes a font cache to disk which is maintained when the computer is switched off.

Font descriptor
An annotation in a **Portable Document Format** file (see **Acrobat** for definition) which records fonts used in a document by name, plus character metrics and style information. If the recipient of the PDF file does not have a font used in the creation of the document, and the font is not embedded in the file, the Acrobat Exchange or Reader program refers to the font descriptor and makes up an instance of a **Multiple Master Font** for the purposes of font simulation.

Front end system
The input and H-&-J side of a typesetting system, on which text is entered and manipulated and formatted.

Generic coding
The use of codes embedded in word-processed text to indicate headings, minor headings, body copy, footnotes etc., without explicitly defining the typography with which these text entities should be represented.

Individual typesetters may advocate their own coding system; two emerging standards are **SGML** and **ASPIC**.

SGML (q.v.) stands for Standard Generalized Mark-up Language, and is backed by the International Standards Organization; ASPIC stands for Authors’ Symbolic Pre-press Interfacing codes and has the support of the British Printing Industries Federation.

Glyph

The actual shape of the image of a type character, as distinguished from its identity within a character set. The term glyph varies in use. For some authorities, the glyph **g** is different from the glyph **g** because each is in a different font. For others, these are different representations of the same glyph, but the character **g** (the 'spectacles' form of g) is a representation of a distinct glyph.

There is not necessarily a straightforward one-to-one mapping between characters and glyphs, depending on how the character set is defined. A single character may be composed of several glyphs, as in a **built fraction**. Or, a number of characters may be represented by a single glyph, for example a **ligature**.

Glyph substitution

A system for improving the display of type which performs on-the-fly substitution of one glyph for another (or, a number of glyphs for a number of others), without changing the sequence of editable characters held in the backing store.

For instance, glyph substitution may display the swash form of **h** when it appears at the end of the word **fish**, but suppress it when the letters **es** are added to make it **fishes**.

Glyph substitution can be used in Arabic to ensure that a consonantal character is displayed with the form appropriate to its position in the word (different forms are used for consonants at the beginning, middle and end of words.)

Glyph substitution could improve the display of the many Indic languages which have superscript and subscript vowels. These vowels are often set at present as non-spacing marks which overlap the display space of the consonant, but glyph substitution could produce more pleasing forms.

Glyph substitution will be supported by **QuickDraw GX**, the imminent extension to the Macintosh screen imaging model.

H-&-J

Short for 'hyphenation and justification', the process of determining where lines of type shall end, how much space requires to be inserted between words and letters to justify the lines of type, and where words should be hyphenated. Usually now performed by computer.

Han characters

Generic term for ideographic characters of Chinese origin, even though they are also used in Japanese (as **Kanji**) and in Korean (as **Hanja**).

Hangul

The Korean syllabic writing system.

Hanja

The Korean term for ideographic characters of Chinese origin (Han characters) used in written Korean.

Hinting

When outline type is rasterised to a relatively low resolution bitmap grid, 'rounding errors', also known as 'aliasing problems', can cause displeasing and obvious irregularities in the bitmaps thus produced.

Hinting is a way of steering rasterization to produce more pleasing results. Essentially, hinting works by establishing a scale of priorities so distortions in type shape which are less obvious will be chosen over distortions which people will notice. For instance, constant vertical stem weight is more important than constant width of counters.

There are various hinting technologies. PostScript **Type 1** and **TrueType** fonts are both hinted.

Hiragana

Cursive syllabic script used to write Japanese words phonetically, often used to write inflectional endings and to indicate the pronunciation of Japanese words. For each Hiragana syllable there is an equivalent syllable in the **Katakana** syllabary, used for writing Japanese words of Western origin.

Ideographic characters, scripts

In ideographic script systems like Chinese, the written mark or glyph denotes a whole word or concept, rather than a sound. Japanese and Korean also use ideographic characters, though they have phonetic (syllabic) writing systems in addition.

Ikarus

A software system created in the early 1970s by Dr. Peter Karow and associates at URW Unternehmensberatung of Hamburg, for digitizing type in a resolution-independent and system-independent manner.

In Ikarus, type is digitized from drawings by entering points along the type profile using a graphics tablet with a digitizing puck. The type outlines are stored as continuous paths made up of straight lines and tangentially-butted arc segments

Indic scripts

Refers to a broad family of scripts which are all descended from the ancient Brahmi script. There are nine official Indian scripts—Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telegu, Kannada and Malayalam. Each script may be used to write a number of languages e.g. Devanagari is used for Hindi, Sanscrit, Marathi, Nepali and 24 other languages. Other Indic scripts are Sinhala (Sri Lanka), Burmese, Thai, Lao and Khmer, Tibetan and Georgian.

ISO 646

International character set defining 7-bit characters for information interchange. ASCII is the American national variant of ISO 646, differing by only a few bracket characters.

ISO 8859

The ISO standard which defines several 8-bit character sets covering the following script systems: Latin, Greek, Cyrillic, Hebrew and Arabic.

ISO 8879

The ISO standard defining the **Standard Generalized Mark-up Language** (q.v.).

Kanji

Japanese name for **ideographic characters** of Chinese origin, used to express many words in written Japanese.

Katakana

The syllabic writing system used to write Japanese words of Western origin, and also to write out Japanese words for visual emphasis. For each katakana syllable there is an equivalent syllable in **hiragana**, the Japanese cursive syllabary. Katakana sequences are used in Japanese word processing to simplify keyboard entry of words, which are converted for display into **kanji** ideographic symbols.

Kerning

A *kern* was originally (in letterpress type) a part of a character which had to extend beyond the metal 'body' of the type, to rest on the 'shoulder' of a neighbouring character. Kerning now means adjusting letterspace away from the normal—either to improve the fit of two characters to each other ('pair kerning') or to tighten or loosen the appearance of the type ('track kerning')

Laser

The original, largely forgotten acronym LASER stands for **Light Amplification by the Stimulated Emission of Radiation**. Laser light is coherent, capable of being focussed precisely and switched rapidly on and off under computer control. This is why laser light is so important in graphic arts today, e.g. colour and monochrome scanning, phototypesetting, laser printing, plate making and engraving.

Laser printing

A form of printing in which a laser beam 'writes' a pattern of text and graphics onto the electrostatically charged selenium drum under computer control. Some large laser printers can turn out over 130 entirely different double-sided pages every minute; more modest ones are essential output devices for Desktop Publishing systems.

Laser setting

A form of photosetting using a flying laser beam as the light source to image photographic material, i.e. film or paper.

Leading

Retained letterpress term for additional space between lines of type, so called because this spacing out was done originally with strips of lead metal. Measured in points, leading is usually indicated in the text type mark-up as, for example 8/9pt i.e. 8pt, 1pt leaded. See also line feed.

Ligature

Character consisting of two or more letters together, modified to fit together better, such as ***ﬂ*** and ***ﬁ*** instead of ***fl*** and ***fi***.

Mark-up

To mark up a typescript is to add comments which instruct a typesetter. Software may be described as using a mark-up method if the text has visible or invisible codes embedded to command the machine to change font, type size etc.

Metafont

A method of defining type fonts created by Professor Donald Knuth of Stanford University, the inventor of the $\TeX$ typesetting language.

Metafont typefaces may have a large character set, are scaleable and resolution-independent, and are defined as a series of strokes made by imaginary pens and erasers along a set of paths in co-ordinate space.

By reprogramming parameters controlling pen shape, stroke weight, path curve and so on, the fonts can be transformed in many ways, dramatically illustrated by Knuth in an edition of Visible Language in which he rendered a psalm in a font which transformed itself by degrees from a serif font with a small x-height into a sans-serif font with a large x-height.

Metrics

See **Character metrics**.

Monospaced font

A type font in which all characters occupy the same width, as in most typewriter faces.

Multiple Master Fonts

A PostScript font technology defined by Adobe Systems, Inc. which defines fonts as existing between extremes on several axes, with specific font "instances" being interpolated on demand from between them. For instance, a two-axis Multiple Master font could range between ultralight and ultraheavy on one axis, and condensed and extended on the other. Adobe have already demonstrated that an axis can also be used to implement **optical scaling** of type.

Non-latin font

A font for setting in a language other than those which use the latin alphabet. For example Greek, Cyrillic (Russian & Bulgarian), Arabic, Korean, Hindi. Some non-latin fonts are **ideographic**, q.v.

Non-spacing mark

A character which has a character width (advance width) of zero, so that when it is typed after a base character it adds its shape to the base character to form the appearance of a composed character. A typesetting system may set **diacritical marks** as floating non-spacing marks so that **a** plus ¨ will display as **ä**.

In most Indic languages there are superscript and subscript vowels (and in Thai, also tone markers) which are typically set as non-spacing marks.

Optical scaling

A system for generating fonts in which the shape of letterforms is adjusted to compensate optically for their displayed size. Thus the small sizes of an optically-scaled font may have stouter serifs, more open counters, a large relative x-height, loose sidebearing spaces and relatively little contrast between thin and thick stems, while the larger sizes of the same face may have more contrast, finer features, tighter sidebearings and a relatively smaller x-height.

Orphan
A word remaining alone on a line at the end of a paragraph. Some consider that this puts excessive whiteness between those paragraphs, and try to avoid orphans.

Pagination
Making up into pages, and/or assigning page breaks and page numbers.

PDL, Page Description Language
A computer language with the specialized function of describing a page to a digital output device such as a typesetting machine or laser printer. Electronic publishing systems and programs use special printer drivers to compile page description language files, which are then sent to the printing device to control its output. Examples of PDLs: Interpress (Xerox), DDL (Hewlett-Packard), PostScript (Adobe Systems, widely supported).

Phototypesetting
A method of typesetting which sets type images onto photographic material such as paper or film.

Pica
A typographical measurement used in the UK and USA, very nearly equal to one sixth of an inch. (Exactly one sixth of an inch in the modified definition used in PostScript, q.v.).

There are twelve points in a pica. See also **Points system.**

Pixel
An individual 'picture cell' which is the smallest visible part of a bit-mapped digital image, e.g. on a computer screen.

Points system
System of typographic measurement adopted in 1886 in the USA and in 1898 in the UK, in which type was measured with reference to the size of the metal base on which each type letter was cast.

In this Anglo-American system, there are slightly more than 72 points to an inch (72 points = 0.9962") and 12 of these points to a pica; though desktop publishing systems treat the point as being exactly 1/72".

The original point system is the continental 'Didot' system, based on that devised by Fournier in 1737, in which the unit is the didot point, 12 of which make up a cicero.

Portable Document Format
An Adobe Systems document interchange file format based on the PostScript page imaging model, optimised for performance and allowing users to add annotations, 'bookmarks', hypertext links etc. See **Acrobat**.

PostScript
Proprietary name for a Page Description Language created for the graphic arts by John Warnock and Charles Geschke, and marketed by their company Adobe Systems. The predominant language for interfacing desktop publishing systems to high-performance laser printers and photosetting machines, PostScript is notable for its flexible vector-encoded type descriptions.

Quad
A horizontal unit of space equivalent to the type size in use.

QWERTY
The standard typewriter keyboard layout devised by American inventor Scholes to slow the operator to the working speed of his original inefficient mechanism. Name derived from the first six keys in the top row of letters.

Ragged (left/right)
Unjustified lines of type which align on one side only, the other being ragged. Most columns are ragged right for the sake of readability as ragged left setting would be difficult to read. The latter is best restricted to short pieces of text e.g. captions.

RAM
Random Access Memory—the volatile memory of the computer used for working storage of programmes and data. Contents of RAM are erased when the computer is switched off (unlike contents of ROM, q.v.).

Raster
A representation of an image as a matrix of pixels set to either black or white. Most digital typesetting systems produce type by running a light source such as a laser beam across the photographic paper in fine parallel lines, switching it on and off rapidly to build up the image.

Raster Image Processor
Equipment which converts the information in a Page Description Language or typesetter driver programme into machine-specific instructions necessary to create the raster image (see above).

Rasterization
The process of converting an outline type form or image to a raster.

Refresh rate
The number of times per second that an image is renewed on a computer display. A high refresh rate makes for a more stable image which causes less eye strain.

RIP
short for ***Raster Image Processor***.

River
A vertical white chasm down the middle of a column of justified type caused by the accidental alignment vertically of enlarged wordspaces; one of the dangers of justified setting in narrow measures.

ROM
Read Only Memory, a permanent and unalterable area of memory resident on ROM chips. This must include certain basic elements of the operating system, enough so that the machine can 'boot' from a system disk. ROM could also contain programmes and resources—thus the LaserWriter ROM chips include the PostScript interpreter programme and some of the fonts required.

Roman
The font within a type family which is taken to be the 'normal' font, from which the italic and bold versions are derived..

Run length encoding
A strategy for reducing computer memory needed for storing bit-mapped images. including characters. On the assumption that the image is created by a sweeping laser beam switched on and off, only distances between switchings (between on and off) are recorded rather than all states of all of the pixels in a line.

Sans serif
A typeface without serifs, such as Helvetica or Univers; often referred to simply as a sans type.

Scan codes
Computer keyboards communicate with the computer by sending a distinct numerical code—the **scan code**—for each key pressed. The computer **keyboard layout** function maps these keys to particular characters. Thus it is possible for the same keyboard to be used for the English QWERTY layout and for the French AZERTY layout, for example.

In some multilingual computer operating systems, for example the Arabic operating system for Macintosh, keyboard layouts are changed on the fly using a hot key as the operator wishes to switch between English and Arabic.

Serif
Cross lines at the end of the main strokes of letters in certain type faces. Serifs can be curved and flowing naturally from the stroke (bracketed) or straight with a 'stuck on' appearance (slab or hairline, depending on thickness).

Set solid
Lines of type set without leading or additional line feed, so that the baseline to baseline measurement exactly equals the point size of the type in use.

Set width
The width of an individual character of type.

SGML
Standard Generalized Mark-up Language; a document interchange standard based on **generic coding**. SGML has been adopted by the ISO as ISO 8879 and is a key part of the publishing strategies of organisations as diverse as the US Department of Defense, IBM, and Oxford University Press.

In SGML, embedded mark-up is used to encode the structure of documents by defining elements in the text as different types of 'entity' (e.g. chapter, chapter head, bibliographic reference) without any definition of how these should be displayed or printed. SGML mark-up can aid the transition between on-line document databases and printed documents, but easy-to-use SGML tools are still scarce.

(Contrast this approach with Adobe Acrobat, which encodes appearance rather than structure.)

Sidebearing space
The defined space to the left or right of a character in a type font. Each space between a pair of letters is the sum of a left and a right sidebearing space (which in practice could be modified by the typesetting system by either a proportional or a linear amount).

Sloped roman
A roman typeface which has been electronically distorted on a photosetting system to produce a directly-derived 'italic'. This generally only works for sans-serif faces and even then the font is typographically inferior to a true italic designed as such; but it saves storage in the computer system.

Small caps
Capital letters which have been reduced in height so that they are the same height as the x-height of the rest of the setting. True small caps are designed as part of the font, and have the same weight of strokes as the other letters. False small caps are synthesized by setting type a little smaller, but then the line widths also weaken in proportion.

Sort
A single character of type. When a compositor in olden times suddenly found that he had run out of the letter **d**, for instance, he was certainly 'out of sorts'!

Superscript, subscript
Characters, usually smaller than the rest of the type, which are raised or lowered from the normal baseline position, usually to indicate references or for scientific purposes, e.g. H_2SO_4, $e=mc^2$.

Supershift
The use of a keyboard with 'option' or 'alternative' key depressed to access a further range of type characters such as ™ ∞ • µ.

Swash characters
Alternate versions of letterforms with decorative flourishes; in desktop publishing, sometimes available in a companion Expert Set font.

Tabulate
To organize type into columns, usually of figures, e.g. statistics, timetables.

TEX
A pioneering programme allowing a computer user to do elaborate typesetting 'front end' work, devised by Professor Donald Knuth for mathematical typesetting for his books. While Knuth holds copywrite on TEX, he has allowed companies to create commercial implementations for various machines: e.g. PCTEX and LATEX for the IBM PC, and TEXtures for the Macintosh. Unix and mainframe versions also exist.

TEX is noteworthy for its superb hyphenation algorithm and ability to access an extended character set, including mathematical symbols. The fonts for many TEX systems are created in Knuth's **Metafont** format (q.v.).

Tone marks

Some Far Eastern languages are tonal in that semantically distinct words may differ in pronunciation only by being pronounced with e.g. a high, low, falling or rising tone. **Vietnamese** is written with an extended Latin character set and the tones are explicitly indicated with non-spacing tone marks above the vowels. Each **Thai** syllable has inherent tone, but this may be modified, also with one of four non-spacing tone marks.

Touching tables

Part of the character metrics information in a font which defines at what point two adjacent characters will appear to touch each other.

TrueType fonts

The TrueType project originated at Apple Computer as an attempt to provide a competitor to Adobe Systems' Post-Script font format. TrueType fonts are scaleable fonts for both screen and printer, and work with Macintosh System 7.x and Microsoft Windows 3.x software.

TrueType fonts are defined using **quadratic spline curves**, said to require more storage space but to be faster to render into bitmaps for display and printing. TrueType fonts support **hinting**, and the hinting mechanism differs from that of PostScript Type 1 in that it is fully programmed into the font.

Two-byte character addressing

…or, "16-bit" character addressing. Any scheme which uses two computer bytes together to address a character within a font. Whereas a single byte can encode only 256 discrete entiries, two bytes can encode character sets in excess of 16,000 characters (enough even for Chinese).

The **Unicode** standard (see below) is a proposed international standard for two-byte addressing.

Type 1 font

A PostScript font format defined by Adobe Systems; the current standard for fonts in the graphic arts.

Type 1 fonts were originally copy-protected and encrypted, and Adobe refused to publish details of how to make them. In response to Apple and Microsoft's plan to develop **TrueType** (q.v.) as a competing font format, Adobe opened the format and published the specifications.

Type 1 format is suitable for fonts of relatively simple outlines, defined as **Bézier curves**. **Hinting** may be associated with Type 1 fonts to improve their display at low and medium resolutions. Type 1 fonts are compatible with Adobe Type Manager software for on-screen font rendering.

Type 3 fonts

Another PostScript font format defined by Adobe Systems. Until Adobe published the Type 1 format details, these were the only kinds of Postscript font which users and other vendors could make for themselves.

Type 3 fonts are not now much used, as they do not support hinting and cannot be rendered from outlines with Adobe Type Manager. However, they support more complex outlines for characters (e.g. complex logotypes), and also bitmapped type.

Unicode

An emerging standard for two-byte (16-bit) character encoding, promoted by The Unicode Consortium, and in the process of being amended and merged with the ISO's 32-bit character encoding standard, **ISO DIS 10646**.

The aim of Unicode is to give each character in the world's written languages a unique numbered reference, to assist multilingual file compatibility and information interchange and to permit the use of very large character sets.

Unicode developed as an initiative of workers at Apple and Xerox, and now also has representation from Adobe, Aldus, Borland, GO, IBM, Lotus, Metaphor, Microsoft, NeXT, Novell, Sun and WordPerfect. Version 1.0 of The Unicode Standard is currently in print.

Upper case

The CAPITAL letters in a font, so called because they are used less frequently and so in the days of hand-set type kept in the upper of the two cases of type in use.

Vector-mapped, vector-encoded

Strategy for describing a shape in computer terms as an path made up of lines between points in co-ordinate space. Vector-mapped type and images can be re-sized freely without degrading in image quality, unlike bit-mapped images which perform adequately only at the size and resolution for which they were created.

Widow

A line of type at the end of a paragraph, which has been carried to the top of the next column or page. May be worth getting rid of to achieve a tidy text appearance or reduce confusion.

WYSIWYG

What You See Is What You Get. An optimistic term which describes desktop publishing systems that try hard to give an on-screen simulation of the appearance your document will have when printed.

x-height

The height of the lower case letters without their ascenders, or those that have none at all such as c, s and of course x, which is used as the standard because it has flat tops and bottoms and so is free of the slight extensions above and below the x-height which are necessary to correct the optical balance of the other letters.