Section Two

Typographic options for Fabula

Typographic options for Fabula

The purpose of this section is to examine criteria for the choice of type fonts suitable for use in Fabula, explain some technical aspects of multilingual computing, and suggest the text layout rules which should control how Fabula texts should be presented.

How will Fabula use type?

The most obvious use of type within Fabula, and that which has received most discussion until now within the Fabula consortium, is the presentation of Fabula storybook texts on screen. However, the whole Fabula package will involve many other uses of type, and we should remember them all.

A Variable, user-authored content

Storybook texts on screen — This refers to the text contents of the two language text panels, as illustrated in the companion report *Screen layout options for the Fabula viewer.*

Word-list texts on screen — The bilingual word-lists prepared by authors to support Fabula stories are potentially quite demanding in their need for the typographic signalling of word function.

B Interface content

Pull-down menu items — Although pull-down menus will be largely unnecessary when Fabula is used for viewing storybooks, they will need to be accessed extensively while authoring.

Dialogue and alert boxes — Dialogue boxes will see some use in the Fabula viewer, for instance when opening a new story, or changing user preferences; and will certainly be required extensively while authoring. Alert boxes (e.g. to advise that settings in a dialogue box are incomplete or impossible) will probably be needed in both contexts of use.

Text within buttons — It may be that storybook interaction buttons will be required to carry text, even if it is only a single numeral. Other screen interface contexts, such as the word-list viewer or on-line help, will also have command buttons with text labels.

On-line help — This is being considered as an option for delivery with Fabula, mostly to advise on authoring tasks. (The *storybook viewer* should be so simple that continuous access to on-line help would be unnecessary, though 'Tool Tips' might be useful as a reminder of the function of tools.[1])

1 By 'Tool Tips' I mean the behaviour whereby, if the cursor is positioned over a tool icon and pauses there for more than about half a second, a small text box with a plain coloured background appears containing a definition of the tool's function. This interface feature appears to have originated on the Windows platform, but has travelled since.

C Printed output

Print-outs of Fabula storybook contents — Whether (and to what degree, and in what form) users should be able to print out storybook content has been much debated within the Fabula consortium and a final decision is far from being made. For the present study, it is important to flag that any choice to allow printing of Fabula content, such as story texts or word-lists, would have typographic consequences. Note that it is not automatically the case that type used to display texts on screen should be the same as those used for printing.

Printed and printable documentation — We should also note that the documentation suite available for teachers to print off and read will require making typographic choices too – though it has been decided to leave this application of typography outside the focus of the current study.

Font choices: three main criteria

It seems sensible even at this early point in the discussion to summarise three main criteria to bear in mind when selecting type fonts for use with Fabula, even though for now many of the terms will remain vague and not yet well defined.

Character set — It is absolutely essential to Fabula that the type fonts used to display ***storybook texts*** and ***word-lists*** (the 'variable, user-authored' content of Fabula, as listed above) should have a range of characters that will display all of the current Fabula languages in a satisfactory manner.

Those languages are presently defined as Basque, Catalan, Dutch, English, French, Frisian, Irish, Spanish and Welsh. In practice, each language will be one of a pair in a defined context of bilingual use, and we have defined six of these language pairs:

Basque	–	French
Basque	–	Spanish
Catalan	–	Spanish
Frisian	–	Dutch
Irish	–	English
Welsh	–	English

Additionally, we have felt it important to consider whether [a] the software's user interface items such as menus and dialogue boxes should be available in all of the project languages and [b] whether the display system should *in principal* be extensible in future versions of Fabula, to service additional language pairs such as Turkish–German.[2]

2 Within the Fabula project team, especially in Britain which has a particularly rich mix of bilingualism thanks to historical relationships with Africa and Asia, there is a strong desire that the mechanisms employed in Fabula software should not make it impossible in future to support such other left-to-right alphabet-based languages as *e.g.* Yoruba, Bengali, Amharic or Việtnamese – though naturally, this would require developing additional or expanded fonts. Supporting right-to-left scripts such as Hebrew and Arabic, and the large character sets of the East Asian ideographic languages, is a major additional complexity which Fabula may never be able to cope with in any future version.

Availability — There are several good arguments, which will be presented below, why the fonts used to display storybook texts and word-lists should be available in identical form to all Fabula users. But this has some large consequences: there is no pre-installed font currently available in completely identical form on both Macintosh and Windows platforms, and the use of additional commercial fonts which are equivalent on both platforms will bring us up against issues of intellectual property.

As this study concludes, there seems to be a reasonably strong case for distributing one or more free fonts for installation with the Fabula software, to ensure that a Fabula storybook authored on one machine will display identically on all machines which run the software – at least in so far as the variable, user-authored content is concerned.

Suitability — This term is obviously the vaguest of the three, and almost deliberately so. Later in this document we will unpack just what makes a type font 'suitable' for use in Fabula software, covering such aspects as clarity, legibility, cultural and pedagogical suitability, utility in fitting texts to available display space, printability, and interaction with other Fabula typographic needs such as structural signalling, and highlight use.

Using 'frozen' type to minimise reliance on 'live' fonts

When analysing the above list of circustances where Fabula will use type, one of the most valuable distinctions is between [a] those circumstances in which type shapes must be generated on the fly from a font actually installed on the user's system, and [b] those circumstances in which it is possible to circumvent this need to rely on the user's system having being configured in advance with certain font resources.

Type as a raster graphic

One approach to avoiding font dependence is to provide a type message that is 'frozen' within a simple raster image. Many Web designers resort to this strategem. Most Web page content is transmitted as text, and is rendered in real time on the user's screen by the Web browser, using fontdata installed on the user's computer. However, there are circumstances where a Web page designer will insert type into small raster images, usually in the GIF file format. Here are some examples:

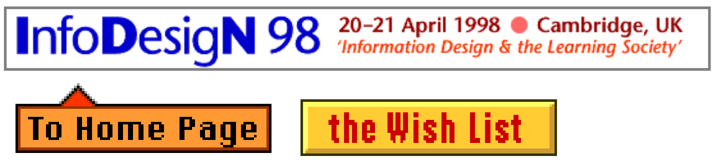

InfoDesigN 98
20–21 April 1998 ● Cambridge, UK
'Information Design & the Learning Society'
To Home Page
the Wish List

Fig. 16 Three examples of type from a Web site, 'frozen' into graphic form.

This is usually done to ensure that the text appears exactly the same on all users' machines, regardless of the type resources which are installed on their machines.

Fabula could also use this approach for on-screen buttons, perhaps not only on the main viewing screen but also in dialogue boxes, should it prove difficult to support all required languages using the standard system fonts.

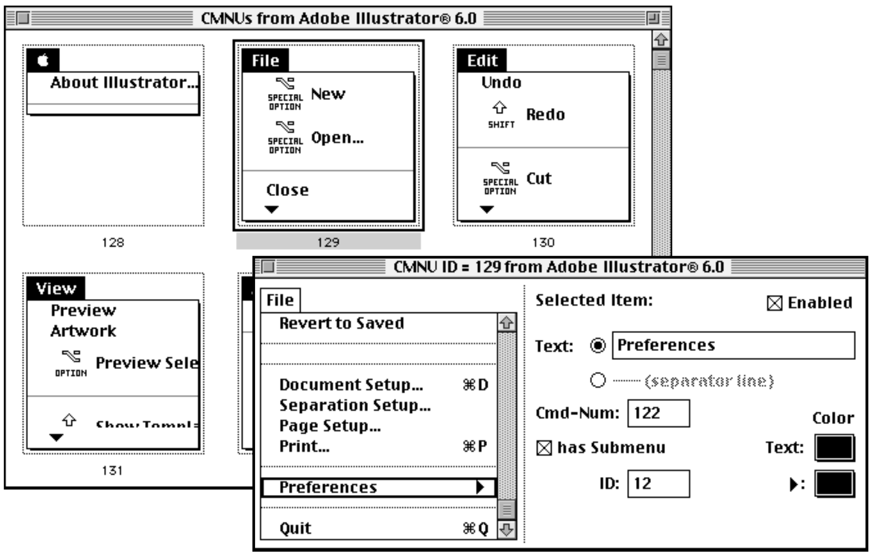
Type embedded in printable documents

It may also be possible to avoid font dependence when supplying user documentation which will be delivered in an electronic format, but has been been designed for printing out. Were Adobe Acrobat chosen as the delivery mechanism, for instance, the fonts used to author the documents could be embedded within the PDF file. This would ensure that all project languages could be supported, and that the documents will appear exactly as their authors intended them to, with no risk of degradation as a result of the reformatting that occurs when one font substitutes for another.

Using system font resources

A computer application running in a modern graphical user interface environment does not have to take responsibility for everything that appears on the screen; the operating system or windowing system provides many of the services required by applications. An application's menus, dialogue boxes and alert boxes are typically stored as number-indexed resources,[3] and these resources in turn rely on the presence of standard system fonts to turn text strings into visible typographical display.

For example, here is a view of the menus in Adobe Illustrator 6.0 for Macintosh, as they are seen when the application has been opened with ResEdit, the Apple resource editor:



Selected Item: ☒ Enabled
Text: ● Preferences
○ (separator line)
Cmd-Num: 122
☒ has Submenu
ID: 12
Color
Text: ■
▶: ■

Fig. 17 Two in a cascading series of windows launched by ResEdit in investigation of the menu resource structure of Adobe Illustrator 6.0 for Macintosh. The text content can be changed, but it is not so obvious how a different font can be invoked to display the menu content.

3 The advantage of using a ID numbering scheme to organise the GUI display resources of a computer application is that resources can then invoke each other by their ID number, regardless of the text or iconic content which they are to display. Thus by using a resource editor such as ResEdit for Macintosh or Metrowerks Windows Resource Builder, the actual displayed content of a GUI resource can be amended, yet the program will still work. This makes it easier to produce a version of an application localised to a different language.

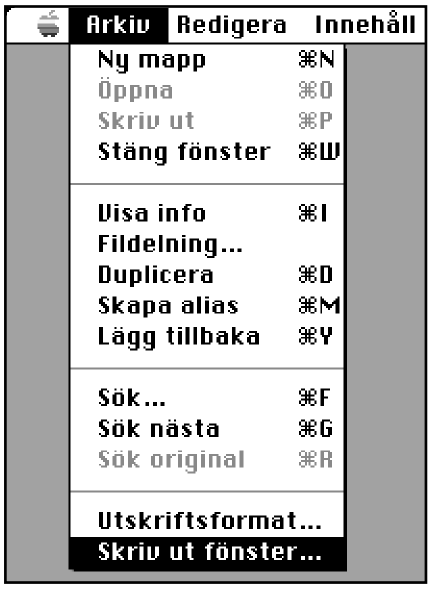


Fig. 18 The Chicago system font used as the default menu display font by Macintosh readily supports a number of languages. This example shows the Swedish version of the Mac operating system.

Other versions of the Mac operating system(e.g. Japanese, Thai, Hindi, Arabic) obviously require the use of other fonts for the menus.

In practice, these menus are rendered with a system font. Macintosh operating systems before System 8.0 used the *Chicago* bit-mapped font for this purpose; System 8.0 and above uses the subtly different font *Charcoal* instead. Within dialogue boxes, many Macintosh applications also use the *Geneva* system fonts, which perform better at small sizes. In the Windows 32-bit computing environments such as Windows 95 or Windows NT, we have been informed that a similar role is played by a system font called *SixedSys.*[4]

If Fabula relies on system fonts for interface display, these character set limitations could prevent us from deploying e.g. a Welsh user interface on menus and in dialogue boxes because the standard system fonts lack certain Welsh characters.

However, the current thinking within Fabula appears to be that for the time being we will need to support only four languages for the Fabula user interface: Dutch, English, French and Spanish.[5] We cannot envisage a circumstance in which a Fabula author will *not* understand at least one of these four languages, and it is difficult therefore for the project to justify the expense of extra time that would be required to provide interfaces, documentation and on-line help in Basque, Catalan, Frisian, Irish or Welsh.

4 It is also possible, we believe, to over-ride this default choice and specify another font if it can be guaranteed to be available on the user's computer.

5 Incidentally, the same will also be true the interface of any third-party software package which we use to install Fabula. (As it is likely that a Fabula installation will need to place resources in various parts of a computer's hard disk directories, it makes sense to automate the installation for the user by using a software installer program.)
Installer VISE from Mindvision can provide a simple interface in which the user makes a few menu choices and then runs the installation process. Freeware distributors like Fabula can apply to Mindvision for a free licence for Installer VISE, which is available with English, French, Dutch and Spanish interfaces – but not for our other languages.

The character set issue

The ‘character set’ of a language is the repertoire of characters which are commonly used to spell and punctuate texts in that language. The major Western European languages are well served on both the Macintosh and Windows platforms, but the Fabula consortium is aware of the need to assess the character sets of the lesser-used European languages to ensure that they can be supported too.

Investigation of this issue is complicated by the fact that the lesser-used languages of Europe have had to struggle for recognition. In the process, their orthographies have been slow to standardise. Also, there has been for some languages a tendency to assimilate the spellings and character sets to what is usual for the majority language of the State.

The written forms of the Basque language appear to have been affected a great deal by these phenomena. For example, a publication of the British and Foreign Bible Society displays the same passage of the Bible translated in the 1880s into three distinct versions of Basque: one version each for the Labourdin and Souletin regions of France, and a third for the Guipúzcoa region of Spain. [*B&FBS*, 1966.] The France-based versions employ the print character ç, which the Guipúzcoan version does not; conversely, the Guipúzcoan version uses ñ, which the examples from France do not.

A further complicating factor is that some languages appear originally to have had manuscript forms including accented characters that were abandoned for ease of mechanical reproduction with office and printing equipment. Again, Basque provides examples. The hard Basque ‘r’ has in the past been represented by the accented form ŕ [*Katzner*, 1997; *Aguirre*, 1961] – but now appears to be represented by ‘rr’ instead. Similar processes seem to have led to the disappearance of the Basque accented forms d̄ and t̄ and l̄ – and in Spain to a reorganisation of the alphabetic order of Basque from the former *Agaka* system to that of the Spanish dictionary [*Aguirre*, 1961].

Our investigations so far indicate that probably all the characters required by Basque, Catalan, Frisian and Irish are supplied by standard Macintosh or Windows fonts, which makes these languages as possible to typeset as Dutch, English, French and Spanish. (Note, however, that this does not necessarily make all these languages *easy* to typeset, because authors may be unaware of the special key-press sequences required to access special characters if they do not appear printed on the keys of their keyboards.)

Basque

Accounts of Basque orthography differ widely, but Fabula's Basque partners do not consider there to be problems with character sets. It appears that Basque as currently taught uses no diacritical marks, except sometimes the use of an acute accent to indicate the stress of a vowel rather than to modify how it is pronounced. It is likely that ç and ñ may also be encountered. None of these uses are problematic.

Catalan

Àà Çç Éé Èè Íí Ïï Óó Òò
Úú Üü

Frisian

Ââ Éé Ëë Êê Ôô Öö Úú Ûû

Irish

Áá Éé Íí Óó Úú

Welsh

Áá Ââ Ää Éé Êê Ëë Îî Ïï
Ôô Öö Ùù Ûû Üü
Ŵŵ Ŷŷ Ÿÿ

Welsh does indeed pose problems. While the Welsh accented 'aeiou' vowels can all be supplied from within the standard Macintosh and Windows character set, and even the rare form ÿ, the ŷ and ŵ forms (and Ŷ and Ŵ, of course) certainly do not exist in the standard fonts at all.

Character combining: not a likely solution

A decision not to support a full Welsh character set is simply not an option for Fabula: it must be done, at least for the displayed storybook texts and the word-list entries. As for how it can be done, there are basically two approaches: either [a] get the Fabula application to work with standard fonts, and use a character composition routine to superimpose a floating accent from that font over a letter, or [b] supply the application with a special font or fonts in which the desired compound letterforms are provided as pre-built glyphs.

The idea of getting the Fabula software to *compose* unusual accented forms from ordinary characters and floating accents appears at first sight to be an attractive idea, as it would obviate the need to create and supply a special font. There are precedents for this: in PageMaker software, a user can type the string '1/2' and then access the menu to run a script which produces an appearance similar to ½ by superscripting the 1, converting the simple forward slash to a shilling-mark, then downsizing the 2.

However, there are several reasons why it seems inappropriate to expect Fabula to adopt this solution:

- As the PageMaker example illustrates, forming compound characters such as accents over letters requires sophisticated text composition facilities such as baseline shift and relative size adjustment. We would pay a heavy price in development time to achieve this.
- The floating accent characters in fonts are hard to access, and in any case are not sufficient to the task. For example, there usually needs to be two forms of the circumflex: the form ˆ suitable for lower-case letters such as ŵ, and the flatter and wider form ˆ suitable for upper-case letters such as Ŵ.
- The argument for composed characters is most convincing when the facility must be available for application to any font chosen by a document designer. But Fabula is a different case: we can dictate the fonts which will be used by the application.

A special Fabula font: the most likely answer

There is therefore a strong argument in favour of creating a special font, with all the characters required by Fabula's languages. Though there may be remaining complications about how such characters are accessed while authoring, and how they are encoded in the storybook files, the use of pre-built characters would relieve Fabula software of the need to perform complex typographical transformations when laying out lines of text.

It should be noted that it would be both illegal and immoral to take a font manufacturer's design, open it in a font editor, add the required characters, generate a fresh font from this data and distribute it to Fabula end users. A special Fabula font would have to be designed *de novo*. There are in any case good reasons of screen legibility and suitability for use with a child audience which may argue for a design specially tailored to that purpose.

Punctuation issues

It should further be noted that European languages vary widely in how they indicate quoted text, which is a common feature in stories. Quotation may be indicated 'like this' or "like this" or ”like this” using standard turned quotes, or ‹like this› or «like this» or »like this« using guillemets. Probably the Fabula character set should retain all these possibilities, and also the ¡ and ¿ required by Spanish.

The ‘sort order’ issue

In discussions about the Welsh character set with Welsh language teachers, my Fabula colleagues were strongly petitioned that Fabula should allow ‘dd’ and ‘ch’ and ‘ll’ and ‘rh’ (and other pairs of letters) to be entered as a single character from the keyboard (or on-screen character map) on the grounds that:

> They are treated as a single letter because they represent a distinct sound and this is reflected in alphabetical order.[6]

It should be noted that these special consonants, which are represented by what to an English readership would be regarded as two letters rather than one, are not ligatured together in any special way as the German ß is; the Welsh ‘dd’ in *roedd* looks no different from the English ‘dd’ in *giddy*.

The simple-correspondence fallacy

The author interprets this request by the Welsh teachers as founded on an (understandable) misunderstanding on their part that it would be possible to do an alphabetic sort on a word-list *only* if there were complete identity between the following four things:

- **input:** a single keyboard or on-screen character-map interaction event which tells the computer that a character is required;
- **storage:** a single electronic record in the computer’s filing system which records that that character has been requested;
- **sorting:** a single position within a sort-order table which allows word-lists to be sorted according to that language’s rules for sorting; *and*
- **display:** a single displayed mark (or *glyph*) which is used to display that character on screen, or to print it to a page.

One of the benefits of processing texts within a computing environment is that the simple correspondence described in the above bullet list can be replaced by much more subtle arrangements, as shall be explained in the diagram on page 52 and following.

Having said that, it must be admitted that few computer applications which are designed to process text fully exploit the distinctions which can be made within a computer between the input, storage/encoding, sort ordering and display of characters. Thus it is quite understandable that the Welsh teachers felt that they would be wise to ask for special characters for ‘ll’, ‘dd’, ‘rh’ and others, which would assist word ordering.

Issues of alphabetic sort order (e.g. for dictionaries and glossaries) turn out on closer investigation to be extremely complex, and all the indications are that that there is hardly ever a one-to-one correspondence between a unique single character and its unique place in the sort sequence. The English

6 Email of 1 December 1998 from Prof. Viv Edwards to Conrad Taylor: *Re: Type and character sets*

language is one of the few examples where such a correspondence could be said to exist – and even then only if we ignore the presence of accented vowels in loan-words, and hyphens or apostrophes which appear in the middle of words such as *co-operate* and *don't,* all of which are ignored in English alphabetic sorting.[7]

Accents, conjuncts and sorting

In some languages, *a pair of letters* such as 'll' or 'ch' are regarded as representing *a single character* in the language, and that character takes its own place in the sort order. This is true of French, Spanish, Catalan, Welsh and Dutch – but not English.[8] Language communities also vary in how they regard the presence of a diacritical mark or accent above a letter. In French, each accented vowel is regarded as a distinct character, and as such it has its own unique place in the French sort order:

Fig. 19

French sort order

a	à	â	b	c	ç	ch	d	e	è	ê	ë	é
f	g	h	i	î	ï	j	k	l	ll	m	n	o
ô	p	qu	r	s	t	u	ù	û	ü	v	w	x
y	z											

Source: von Osterman, 1952

In contrast, accents in Welsh are regarded as modifications to characters, rather than characters in their own right. Accents are therefore ignored in the sort order: **á** and **â** and **ä** are all sorted as equivalents:

Fig. 20

Welsh sort order

a	b	c	ch	d	dd	e	f	ff	g	h	i	j
k	l	ll	m	n	o	p	ph	r	rh	s	t	th
s	w	y										

Accented characters sort as if unaccented

Source: von Osterman, 1952

7 For an extended discussion of sort order in the context of English index preparation, see Mulvany (1994).

8 English has two consonantal clusters, the soft 'th' and voiced 'th', which in Anglo-Saxon and early mediaeval days had their own letterforms to represent them: the characters **eth** and **thorn**, which have been retained in the Macintosh and Windows character sets because they are both used in modern Icelandic, and eth is also used in modern Faroese.

In Catalan, it would appear that accents are similarly ignored in the sort order for most letters, but the difference between **u** and **ü** when following either a **g** or a **q** is of sufficient significance to affect the sort order:

Fig. 21

Catalan sort order

a	b	c	ç	ch	d	e	f	g	gu	gü	h	i
j	k	l	l·l	ll	m	n	ny	o	p	qu	qü	r
s	t	u	v	w	x	z						

Other than as shown, accented vowels sort as if unaccented

Source: von Osterman, 1952

How does sort order affect Fabula?

It can be seen from the examples just quoted that if Fabula were to take seriously and literally the request from Welsh teachers for unique conjunct characters, as described on page 49, and go on to accede to similar requests for other languages, we would soon have a monstrous character set. Also, using sort-order as the starting point for discussions of character sets will bring up contradictions between languages which regard e.g. **â** as a character with an accent, and those which regard it as a character in its own right.

Fabula's interest in sorting will almost certainly be limited to the organisation of entries in word-lists during the authoring process. Automatic sorting of entries within word-lists may seem an attractive feature to add to the Fabula 'wish list', but there are also sound arguments for resisting such calls:

- It is not clear when a user would ever need to see an alphabetically sorted word-list. As explained on page 21, the contents of word-lists might display as individual entries rather than in a scrolling word-list display. (However, it may be attractive to be able to print out an entire word-list, in which case a dictionary order would be useful.)
- Proper sorting algorithms for some languages would have to accommodate equal sort orders for numerous single characters (e.g. **Á á Â â Ä ä** in Welsh), and also different sort orders for groups of characters (e.g. **d** and **dd**, **c** and **ch** in Welsh). This would be a significant piece of programming even for one language – to provide similar support for nine languages would be a very big effort.
- The word-list attached to a Fabula storybook will not be very long. The shorter the resource requiring alphabetical sorting, the more difficult it is to make a case for automating the process.
- It is far from clear where in a word-list entry one would start the scanning process for alphabetical sorting. Presumably **le capitan** would be sorted under **c**, not **l**.

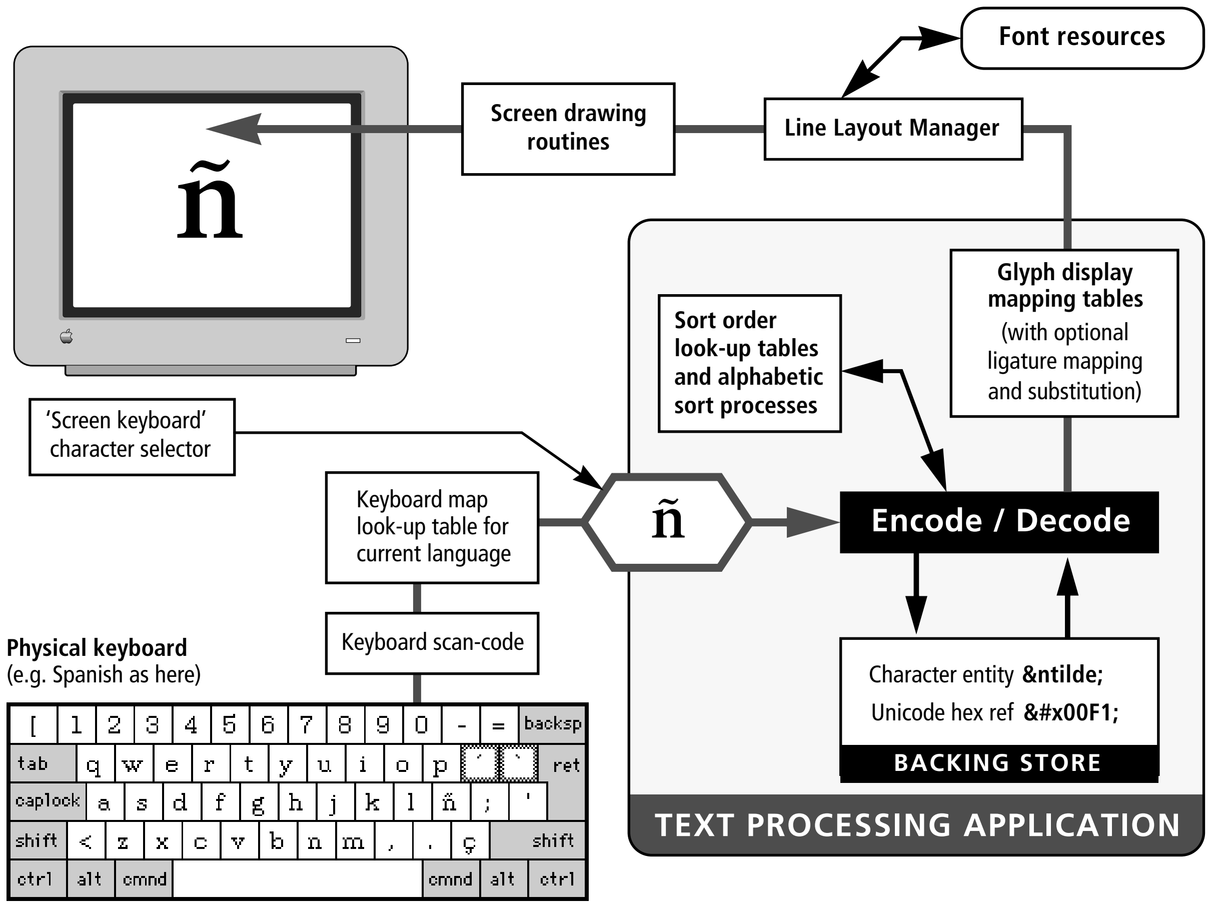

Font resources
Screen drawing routines
Line Layout Manager
ñ
Glyph display mapping tables (with optional ligature mapping and substitution)
Sort order look-up tables and alphabetic sort processes
'Screen keyboard' character selector
Keyboard map look-up table for current language
ñ
Encode / Decode
Keyboard scan-code
Physical keyboard (e.g. Spanish as here)
Character entity &ntilde;
Unicode hex ref &#x00F1;
BACKING STORE
TEXT PROCESSING APPLICATION
[1 2 3 4 5 6 7 8 9 0 - = backsp
tab q w e r t y u i o p ´ ` ret
caplock a s d f g h j k l ñ ; '
shift < z x c v b n m , . ç shift
ctrl alt cmnd cmnd alt ctrl

Fig. 22 Diagram of how the computer operating system and a text procession application such as Fabula could handle the input, storage, processing and display of a character in an alphabet.

- Perhaps it would be more economical of programming effort (and more flexible in fine-tuning the display of word-lists) to provide a quick-and-dirty batch sort facility, or even none at all – but to make quite sure that it is easy while authoring a word-list to cut and paste an entry to a new place in the sequence, and perhaps also to 'nudge' entries up and down in the list.[9]

Keyboards and scan-codes, encodings and glyphs

As has been hinted above on page 49, there is room for considerable sophistication in computerised text processing to ensure that text entry is simplified, characters are correctly identified in storage, sort order is handled appropriately for the current language and the right glyph shape is picked from to font to display it on screen. Figure 22 at the top of this page offers one such schematic arrangement of functions.

9 Because the tasks involved in creating, editing, formatting and organising a word-list are relatively discrete and quite different in nature from writing the texts and making the pages and interactions in a Fabula storybook, it may be worth considering the idea of developing a Fabula word-list preparation environment as a separate application in its own right, rather than as an integrated module within the main Fabula authoring environment.

From keyboard to character

Unlike the mechanical link from a typewriter keyboard which is rigidly determined, a computer keyboard provides a more flexible link. Circuits within the keyboard detect the key-presses, and transmit scan-codes to the computer. The operating system's main event loop learns thereby which key (or combination of keys) has been pressed.

Keyboard scan-codes do not of themselves identify which characters are required. To obtain this information, the operating system must consult a look-up table for the current language. This means that one physical keyboard can be used with a keyboard layouts suitable for several different languages; though it is always easiest to work with the keyboard layout which matches the characters screen-printed onto the tops of the keys.

The computer on which I am writing this has been set up to give me fast access to six different keyboard layouts, selectable from a menu:

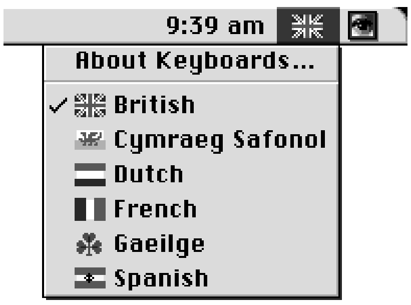
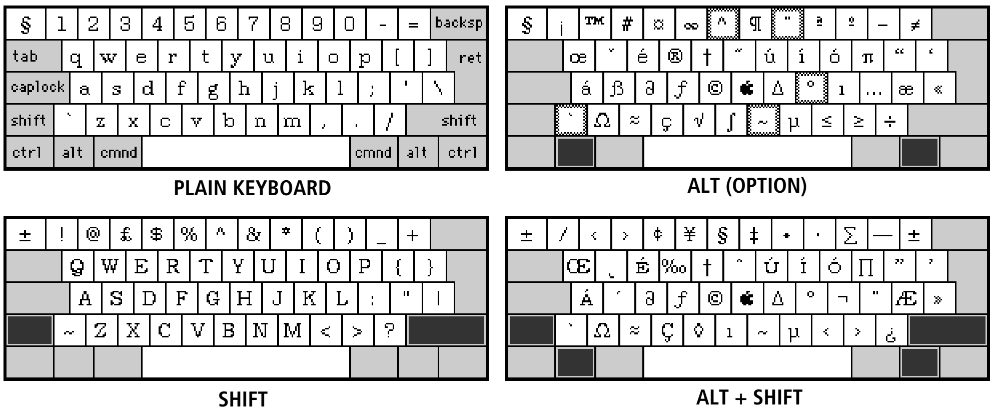


Fig. 23
The Macintosh keyboard selection menu, set up to switch between six alternate language mappings between keys pressed and characters accessed.

Each Macintosh keyboard layout look-up table assigns a different meaning to what may be thought of as four 'banks' of keys. The character selection which results from a key-press is thus determined in part by the keyboard look-up table that is active, and in part by whether a key is pressed on its own, or in conjunction with the Shift key, the Alt key, or the Shift and Alt keys together. Fig. 24 shows the Irish 'Gaeilge' keyboard map:

Irish Gaeilge keyboard map for Macintosh

Fig. 24

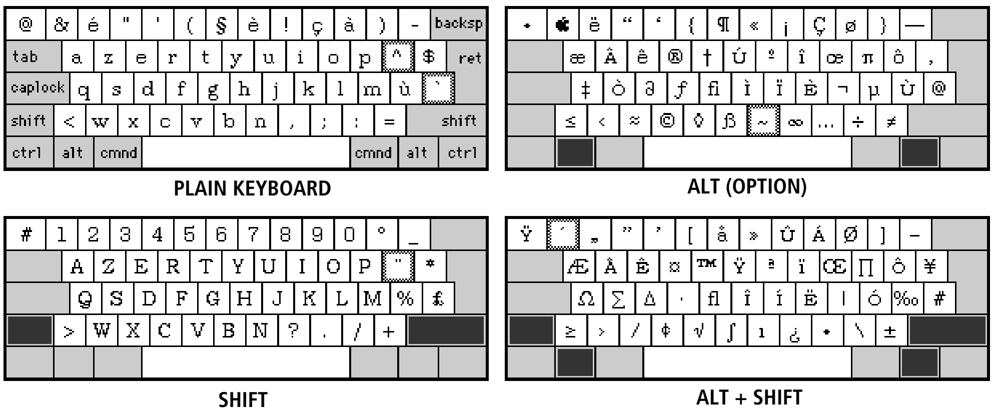


The Irish keyboard map is not greatly different from the British one, but it provides accelerated access to the accented vowels **á é í ó ú** and their capitalised equivalents by requiring only that the Alt key be depressed in addition to the key or keys for the normal, unaccented form. The British Mac keyboard requires the user who wants to set these accented vowels to go through the more complicated process of pressing Alt–e first, followed by the desired vowel over which the accent is to display.[10]

In contrast, a French keyboard presents a display which is very different from what a British user would be comfortable with.

French standard keyboard map for Macintosh

Fig. 25



From recognition to encoding for storage

The scancode conversion process which results in a computer application 'recognising' the character required usually results in the character being encoded and moved about in memory as a bytecode, that is, a simple eight-digit binary number which represents the position the character occupies in the code table. In Macintosh, for example, ñ is assigned the binary number 10010111, which in decimal notation is 150.

However, there is no reason why a program might not choose to represent a character in some other way – if not for internal use in RAM, at least for writing out to non-volatile storage, such as in a file on a hard disk. In fact there are often good reasons for re-encoding any character which is not within the most basic 7-bit codepage – the American Standard Code for Information Interchange (ASCII) – because bytecodes higher than 127 mean different things on different kinds of computer.[11]

Where text data must be portable across different computing platforms and language communities, it is most sensible to re-encode the text in a 7-bit format, and Fabula should do this to ensure that a text authored in France

10 Note that the two-step process of typing Alt–e, letting go, then typing a vowel may give the user the *illusion* that the character is being composed out of partial glyphs, and the false expectation that typing **Alt–e** followed by **r** would build the old Basque character ŕ – which it does not. All that is happening is a conversion from a keyboard scan-code sequence, through a mapping table, to the choice of a pre-built accented glyph in the font being used.

11 This is why moving a file between Macintosh to Windows often results in strange character substitutions, such as turned single quotes becoming ì or í. At least the Mac has a unified codepage for most Western European languages; in the Windows environment, the same code can mean something different from one country to the next.

on a Macintosh can be read in Spain on a Windows PC. The Extensible Mark-up Language, XML, which is to be the underlying data structure of Fabula storybooks, supports at least two such methods:

- **Character entity encodings** could be used as they have long been used in Standard Generalized Mark-up Language applications, and more recently in HTML.

  Annex D, Section D·4 of the SGML standard ISO 8879:1984 includes a number of suggested reference character entity sets such as `ISOlat1` and `ISOlat2` which would suffice for Fabula's current purposes. Thus for example, ñ would be re-encoded as `&ntilde;` and ŵ as `&wcirc;` (note: the `&` and `;` are part of the encoding: they function as start and end delimiters for the character entity).

- **Unicode hexadecimal references** can also be used with XML. Unicode is a two-byte character encoding scheme for the languages of the whole world, researched and promoted by the Unicode Consortium. The use of two-byte encoding makes it possible for Unicode to catalogue over 40,000 characters, including the ideographs of Chinese, Japanese and Korean.

  In practice within XML, the numbers are written out in 7-bit ASCII form, using four numerals in the 'hexadecimal' base-16 notation. Thus ñ is re-encoded as `&#x00F1;` and ŵ as `&#x0175;`.

Sorting tables

It is also possible to envisage that an application could use additional sets of lookup tables to assist in performing an accurate alphabetic sort for a word-list in a chosen language. I have not done much thinking about this, but imagine that the process would involve a series of re-encodings, perhaps along the following lines:

(a) The relevant indexation element in the word-list record, probably identified through mark-up as a unique XML entity, is copied to a special indexation field within the record.

(b) With reference to a look-up table, each character or character entity in the indexation field is re-encoded. This is a two-part process: in the first pass, letter pairs with a special sort order such as 'ch' in Welsh are re-encoded, and non-sorting characters such as hyphens are deleted; on the second pass, every remaining character code is converted to a number indicating that character's position in the language's sort sequence. (This may result in some quite distinct characters being converted to the same number, if they have an equivalent position in the sort order.)

(c) A sorting process then processes the sort-sequence numbers in the re-encoded contents of the indexation fields, to place the records in the required sequence.

(d) The indexation fields can now be deleted from the word-list file, which is saved in the new order.

A technique similar to this would offer powerful alphabetic sorting. It is computationally intensive, but might be appropriate for batch sorting of word-lists.

Mapping for display: characters to glyphs

A computer text-processing application can also apply the subtle power of look-up tables when it converts its internal record of a sequence of **characters** and prepares them for display or print as an arrangement of **glyphs**. Here we are using these distinct terms as they are defined in *The Unicode Standard, version* 2.0:

> The Unicode Standard draws a distinction between *characters*, which are the smallest components of written language that have semantic value, and *glyphs*, which represent the shapes that characters can have when they are rendered or displayed. There are various relationships between character and glyph: a single glyph may correspond to a single character, or to a number of characters, or multiple glyphs may result from a single character…
>
> Unicode characters represent primarily, but not exclusively, the letters, punctuation, and other signs that comprise natural language text and technical notation.[12] Characters are represented by code values that reside only in a memory representation, as strings in memory, or on disk…
>
> In contrast to characters, glyphs appear on the screen or paper as particular representations of one or more characters. A repertoire of glyphs comprises a font.[13] Glyph shape and methods of identifying and selecting glyphs are the responsibility of individual font vendors and of appropriate standards and are not part of the Unicode standard.

It is not clear at this stage in the project whether Fabula will have need of particularly sophisticated mappings between characters and glyphs, but we may take heart from the following observations:

- In a closed and private system like Fabula, which controls its own encoding and probably its own font technology, it would be possible to manage a huge and extensible repertoire of glyphs spread across a number of what to the computer appear to be separate 'fonts' – yet which are managed by the character-to-glyph mapping tables in such a way that the experience of choosing characters and have them displayed as glyphs is managed smoothly and seamlessly on the users' behalf.

- If it should turn out that e.g. 'dd' needs to be encoded in the backing store as a distinct character, a mapping table can still decompose it to two separate glyphs for display and print.

12 In addition. the Unicode standard reserves code points for the control codes used in computer communication (such as NULL and ACK) and to designate formatting boundaries in texts (such as carriage return).

13 A full repertoire of glyphs, however, might be organised by spreading it acrosss a number of separate fonts.

What kind of typeface is most suitable for Fabula?

Having dealt with the font availability, character set and language technology issues that face the Fabula project, we now come to the question of how to specify the characteristics of the type that will be used by Fabula for the following two purposes:

- **Screen display of storybook texts**
- **Screen display of word-list entries**

Additionally there is the issue of whether Fabula storybook pages or contents such as texts and word-lists should be *printable*; but for now we may assume that faces that pass the tougher test of suitability for screen reading should also be appropriate for printed output.

Research findings vis-à-vis typography for children

A concise and readable guide to what typographic choices may assist in literacy education for children is *How it looks – a teacher's guide to typography in children's books* by Sue Walker [*Walker* 1992]. It should, however, be noted that this booklet was directed towards an analysis of type in print, rather than on screen, and for the English language.

Therefore I present below some of the main advice given by Walker, followed by observations on how the nature of computer screens, and the characteristics of the Fabula languages, may require that advice to be modified.

- **Tall ascenders help to identify the word shape.** The issue here is how a typeface designer has chosen to allocate the vertical space occupied by letters between the ascenders, x-height, descenders – and, for most of our languages, the accent space too.

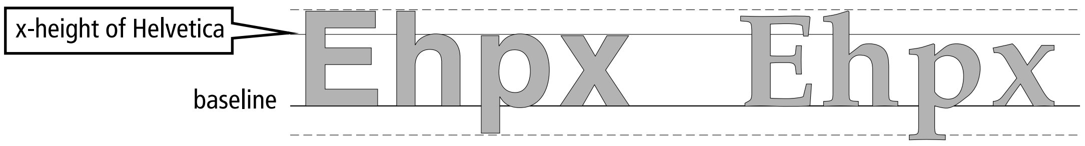


Fig. 26 A comparison between the two typefaces Helvetica (left) and Palatino (right) shows that Helvetica has more of its vertical space allocated to the x-height, which reduces the space left for character ascenders.

The argument in favour of fairly tall ascenders is that as readers become more confident in reading by word shape recognition, they will be assisted by faces which make the word shapes clearer. (On the other hand, one does not want the allocation of space to ascenders to advance to the extent that the internal details within the x-height zone become difficult to perceive.)

- **There should be a clear distinction between characters.** The point is that there should not be so much similarity between letters that the risk is run that a child might confuse one letter for another. Particularly when learning to read an unfamiliar language, or one to which a child has not been greatly exposed in print form, it is important to avoid such ambiguities:

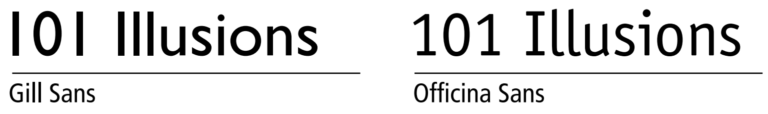


Fig. 27 Numeral one, capital I and lower case L are easily confused in Gill Sans. Officina avoids this problem.

Sans-serif typefaces often suffer from poor differentiation between letters, but it would be a mistake to leap to the conclusion that serif faces are therefore categorically better than sans-serif ones. Also, as the Officina example above shows, there are some faces which defy an easy categorisation as sans or serif: though it 'claims' to be a sans, it has serifs on the numeral 1 and the lowercase and capital I, and a turned flick at the base of the lowercase l, all of which make the letters easier to distinguish from each other.

- **Fonts should be avoided if they include quirky letterforms.** Walker does not give reasons for this recommendation, but it is clear that some letterforms could be distracting at best, confusing at worst. The figure below contains a collection of examples of letters with varying degrees of quirkiness:

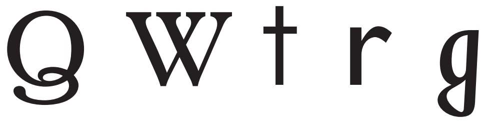


Fig. 28 A parade of odd characters. From left to right: Bookman, Plantin, Kabel, Gill Sans and Joanna Italic.

Debate about the relationship between print and handwriting

Walker [1992] also refers to an argument occasionally raised, that the letters in media from which young children are learning to read should as much as possible look like the shapes of letters which they are learning to write. As she explains:

> Some typefaces have extra 'schoolbook' or 'infant' characters. These replace those that don't look like the letters children write…
>
> The letters most usually modified are 'a', 'g', 'l' and 'y', and the numbers '4', '6' and '9'. Not all publishers, however, use the full range of modified characters available; the curly l, for example, is not widely used, yet in terms of character differentiation it might be one of the more useful modifications. One of the dangers of using infant characters is that some of the letters can be so similar as to cause confusion…

This issue is a vexed enough subject in one national context; in a cross-cultural setting, it is further complicated by the likelihood that accepted styles for children's handwriting will vary from one country to another.

Certainly there a differences within Europe in how numerals are written, especially affecting the **1** and **7**, and this would have a significant impact where a teacher sought to weave a numeracy theme into a Fabula story.

It has been argued that children should learn to recognise letters in a variety of different forms, and that therefore it is not important to try obsessively to match type to how children are taught to write. However, I do suggest that whether we assume that Fabula will choose an existing typeface or make its own, it may be worth examining the letters one by one and comparing them with children's written forms – as just one criterion among others. (The figure below contains letters from a number of fonts, some of which look more like handwritten forms than others).

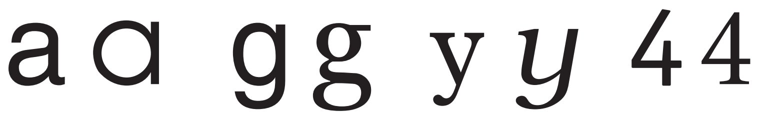


Fig. 28 Some of these characters look more like children's handwriting than others.

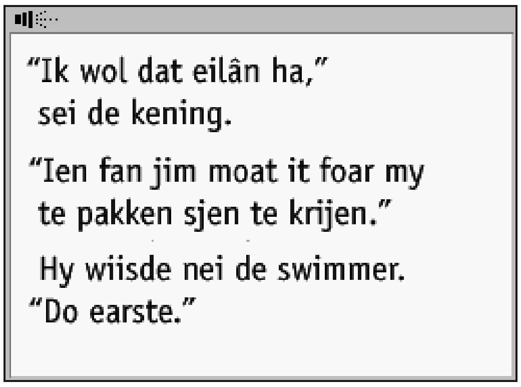
However, the criterion that letterforms should look distinct from each other is, in my view, a more important one: the second **a** above, from Avant Garde, is too similar in appearance to an **o**.

The set-width issue

In Section One of this report, there was a lot of concern with making the most of the small area provided by a computer screen. Given that, it is worth favouring typeface designs which are relatively compact in the lateral dimension. Typographers refer to such faces as having narrow set-widths.

What size of type?

I originally intended to write about the size of type later on, in the context of how type is *used* on Fabula screens. However, it is important to have some idea of how large type will be when deciding which type to choose. My suggestion is that the type in which Fabula stories are displayed on storybook screens should be relatively large, about the size which I used in creating the storybook screens demonstrated in Section One, and as shown below:



"Ik wol dat eilân ha,"
sei de kening.

"Ien fan jim moat it foar my
te pakken sjen te krijen."

Hy wiisde nei de swimmer.
"Do earste."

Fig. 30 The type for the screens displayed in Section One was set in Officina Sans at 18 points, within Adobe Photoshop image-processing software.

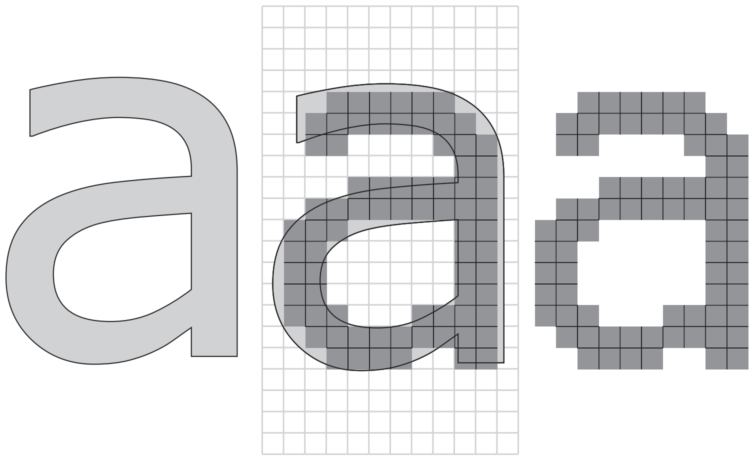


Fig. 31 This diagram shows the letter **a** in Verdana, a typeface design by Matthew Carter, in its outline form and also in the raster form in which it displays on screen.

Carter actually designed this type starting with the screen rasters in an attempt to create a font that would work well on screen, for instance when viewing Web sites. Verdana is bundled free with MS Internet Explorer and ia separately available fromMicrosoft's Web site.

There are several arguments that I would advance in support of this recommendation:

- There is a convention to make type larger in books for early readers, doubtless for good reasons, and it would seem appropriate to mirror that in Fabula's type choices.
- Fabula storybook screens are likely to be shared by a number of children, some of whom may be positioned more than a metre away from the screen. Larger type helps in this circumstance.
- With some newer computers, there is a trend to provide more pixels on screen, but use smaller pixels, which has the unfortunate effect of reducing the size at which type displays on screen.[14] Larger type would offset this.
- The pixellation of type when it displays on a computer screen tends to erode the difference between letters and make them ugly and difficult to read. Using a larger size of type helps to preserve the intended outline form and character differences better.

Type rasterisation and Fabula

As was remarked above on page 57, research findings about typefaces suitable for use in children's books has to be filtered further, using what we know about how typefaces appear on a computer screen.

Virtually all modern type is rendered as a raster image – that is, an image made up as an array of small identically sized dots. I have drawn Fig. 31 above to illustrate this.

There is more than a little historical irony to the story of how computer systems have evolved to handle type on screen, and it is not a story to which a happy ending has yet been written. Early word processing systems used screen fonts which were designed to be as legible and unambiguous as

14 The recently introduced Apple iMac is a case in point. Its megapixel display is crammed into a 14" diagonal screen.

possible within the screen technology constraints of the day, and bore no relationship to the type designs which were printed onto paper using a daisywheel printer.

When the Apple Macintosh was launched in January 1984, it came with a series of bit-mapped fonts (that is, black-and-white raster fonts), each of which was available in a number of sizes expressed in points. The typeface called 'Geneva', for instance, designed by Charles Bigelow, was supplied in 9pt, 10pt, 12pt, 14pt, 18pt, 20pt, 24 pt, 28pt, 36pt and a few larger sizes. If one chose this font in any of those sizes, it would produce a crisp 72 dpi display on the Macintosh screen, and would print reasonably on Apple's dot-matrix printer, the ImageWriter. (The Mac could simulate other sizes by interpolation between the supplied sizes, but the results neither looked good on screen nor printed well.)

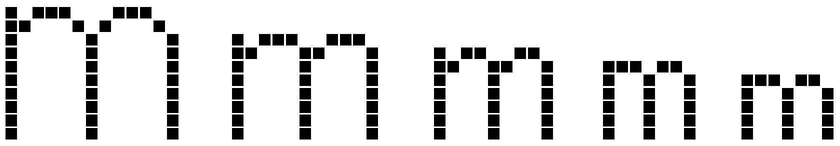


Fig. 32 A cascade of sizes of Geneva bitmapped type, as designed by Charles Bigelow for the Mac system.

Since then, a series of technical developments have progressively displaced bit-mapped type designs from the computer screen, in favour of designs stored in an outline (vector) form – the cubic curves of Adobe's PostScript Type 1 format, or the quadratic curves of Apple's TrueType format[15] – and then rasterised on demand (or 'on the fly', as the industry calls it) into bit-maps for the computer screen or the printer.

The aim of this development was to support Desktop Publishing. People wanted to see on their screens a reasonable approximation of what they would be printing out. The actual readability of the type on screen became a lesser consideration. (This has some interesting 'revenge effects', however: there is some evidence that people choose a size of type for their documents based on the ergonomics of being able to see clearly what they are typing into the word-processor, rather than the ergonomics of what would be the most user-friendly size for the reader!)

Of course, the final twist to the story is that because of the last few years' development of the World Wide Web and multimedia, and perhaps also the spreading use of email, more people are spending more of their time reading type off a computer screen – ironically, using type technologies and faces which have been optimised for print rather than screen display.

Fabula, on the other hand, is oriented primarily towards type on screen, and would benefit from any approach that can improve display under these circumstances.

15 TrueType was developed under the codenames 'Bass' and then 'Royal' before being released as TrueType and shared with Microsoft in an attempt to erode Adobe's hegemony of outline type for personal computers.

Fabula type recommendations

I have decided to approach this as if Fabula were to be commissioning its own typeface design – which may, indeed, be the case – on the grounds that this gives the opportunity to express what the ideal Fabula typeface should be. It may, of course, be found necessary to compromise…

- Fabula type will be used on screen at a small number of sizes completely under the software's control. Type on the main storybook pages is likely to display at about 18pt – the height of capital letters will display using 13–14 pixels. Type in word-lists is likely to display a little smaller, perhaps at about 14pt. The Fabula typeface should be optimised for display at those sizes.
- Font technologies permitting, hand-tuned bitmaps for those sizes may be provided to produce the highest possible display quality. For instance, this could be achieved in a Macintosh font by modifying the font's FOND resources. For TrueType Enhanced Screen Quality under Windows, delta hinting for those sizes could improve the way bitmaps are produced from outlines.
- As for variants within the font family, only a Normal and Bold need be provided – the Bold will see use only in word-lists, for purposes of differentiation. This is because italic/sloped type rasterises very poorly on screen (it has a jagged appearance).
- At the size at which the type is displayed in storybook screens, stem weights will be normalised to two pixels wide. (Experience in creating the storybook screens using Officina Sans suggests that this is just about right.)
- To save space, the set-width of Fabula type letters will be relatively compressed, but not absurdly so.
- For a better 'fit' both with children's handwriting and the limitations of screen display, the Fabula type will have a uniform-appearing stroke width. It will also be largely a sans-serif design, but serif-like strokes will be used to help differentiate characters better as well as to simulate a handwriting ductus (flow direction of the pen).
- Further to minimise any jaggedness of appearance, the Fabula type face may have tighter curves than most typefaces and make slightly more use of horizontal and vertical strokes. This has to be done very carefully, however.

The author has already started experimenting with some typeface design ideas and sketches that are informed by the above criteria – but that is a subject for another report!

A role for anti-aliasing?

One approach that is sometimes used to improve the appearance of type rasterised at low resolutions, where the screen can support multiple levels of shading, is to use ‘greyscale’ or ‘anti-aliased’ fonts. There is discussion of this in *Karow* [1987] and *André* [1997], and this author has been aware of the technique through discussions with engineers at Bitstream (type designers) and consulting projects with G2 Systems and Aston Electronic Designs (producers of screen type generators for television).

The diagram below illustrates what an anti-aliased rendering of Verdana might look like:

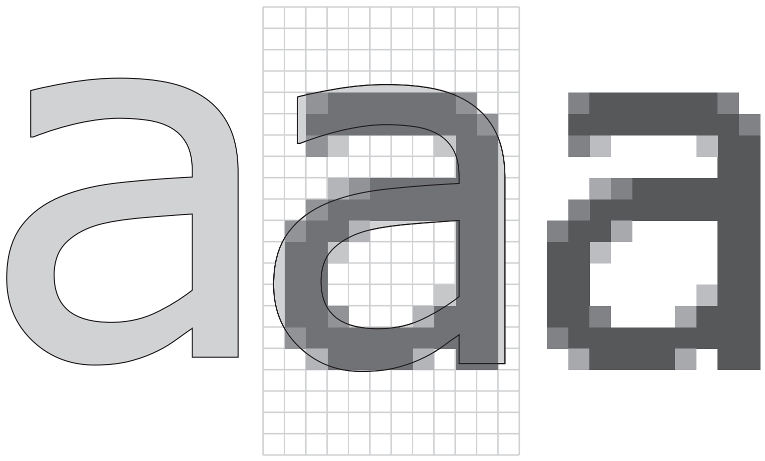


Fig. 33 The actual contours of the Verdana **a** are simulated a little more accurately if greyscale pixels are used.

Note how I have taken care not to anti-alias verticals and horizontals, to keep the design as crisp as possible.

PostScript typefaces, when rasterised through later versions of Adobe Type Manager, may optionally be requested in anti-aliased form, but Adobe’s anti-aliasing technique is extremely crude and illustrates the dangers of doing it badly: the result is a blurry mess, with even the vertical and horizontal edges smudged.

It remains to be seen whether Apple can do a better job with Mac OS 8.5’s anti-aliasing of TrueType fonts – Microsoft seem to do this reasonably in Windows NT.

Given the peculiarity of Fabula’s use of a few fixed sizes of type, it may even be possible to provide carefully hand-tuned bytemaps for the required sizes – an approach which was used by type designer Bruno Maag in designing multimedia screen fonts for Dorling Kindersley CD-ROMs (see *Taylor*, 1996].

Fabula automated quality composition

The final topic to be addressed here is how the quality of the 'typesetting' within Fabula screen displays can be optimised. The good news is that Fabula will relieve storybook authors of the need to be good designers, by taking these kinds of decision away from them. This may be regarded as somewhat tyrannical, but the despotism is intended to be benevolent.

The diagram below, which analyses how the 'expert system' inside this author's head informed the production of the screen visualisations, shows some of the likely features of Fabula composition.

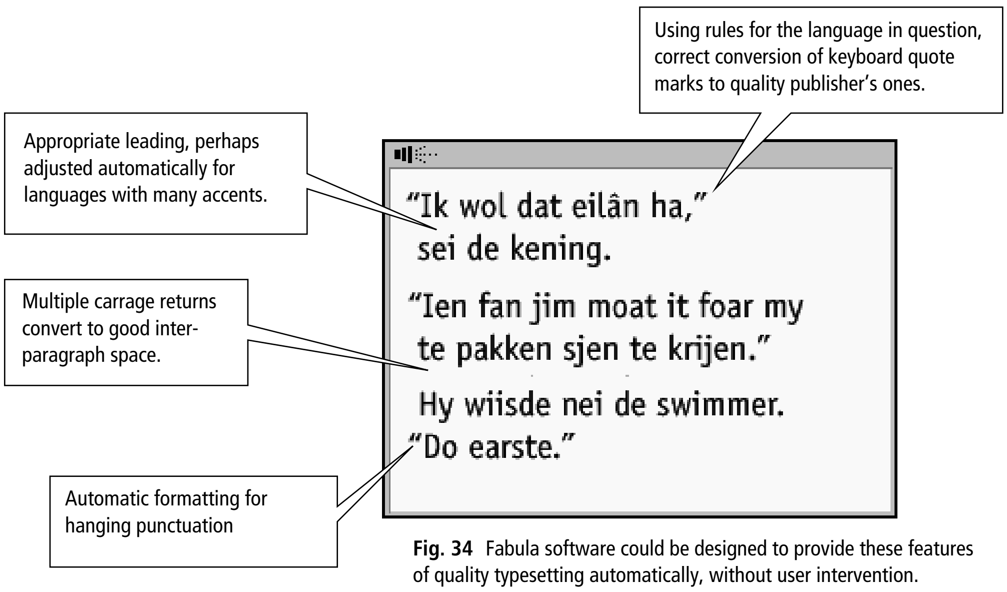

“Ik wol dat eilân ha,”
sei de kening.
“Ien fan jim moat it foar my
te pakken sjen te krijen.”
Hy wiisde nei de swimmer.
“Do earste.”
Using rules for the language in question, correct conversion of keyboard quote marks to quality publisher's ones.
Appropriate leading, perhaps adjusted automatically for languages with many accents.
Multiple carrage returns convert to good inter-paragraph space.
Automatic formatting for hanging punctuation

Fig. 34 Fabula software could be designed to provide these features of quality typesetting automatically, without user intervention.

Thus overall, by providing a quality typeface and automated composition facility, Fabula could meet its design goals of ensuring that every Fabula storybook looks as good on screen as is possible.