

Baskerville

The Annals of the UK T_EX Users' Group
ISSN 1354-5930

Editor: Editor: Sebastian Rahtz

Vol. 4 No. 1
February 1998

Articles may be submitted via electronic mail to baskerville@tex.ac.uk, or on MSDOS-compatible discs, to Sebastian Rahtz, Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, to whom any correspondence concerning *Baskerville* should also be addressed.

This reprint of *Baskerville* is set in Times Roman, with Computer Modern Typewriter for literal text; the source is archived on CTAN in `usergrps/uktug`.

Back issues from the previous 12 months may be ordered from UKTUG for £2 each; earlier issues are archived on CTAN in `usergrps/uktug`.

Please send UKTUG subscriptions, and book or software orders, to Peter Abbott, 1 Eymore Close, Selly Oak, Birmingham B29 4LB. Fax/telephone: 0121 476 2159. Email enquiries about UKTUG to uktug-enquiries@tex.ac.uk.

Contents

I	Editorial	3
1	<i>Baskerville</i> articles needed	3
1.1	L ^A T _E X 2 _ε	3
1.2	TUG'94	3
1.3	Colophon	3
II	Mixing and matching PostScript fonts	4
1	Introduction	4
2	Matching fonts	5
2.1	Matching heights	5
2.2	Matching widths	6
2.3	Matching weight	7
2.4	Results	8
3	Conclusion	9
III	Building virtual fonts with 'fontinst'	10
1	Introduction	10
2	A problem with fonts	10
3	A solution: virtual fonts	11
4	A problem with virtual fonts	11
5	A solution: the 'fontinst' package	11
6	Using the 'fontinst' package	12
IV	Do you really need virtual fonts?	13
0.1	Easy, totally general reencoding	15
V	Further thoughts on virtual fonts	17
VI	Colour slides with L ^A T _E X and seminar	20
1	Slides and L ^A T _E X	20
2	Using the seminar style	20
3	Frame styles	21
4	Interleaving notes, and selecting subsets	23
5	Control over slide size, fonts and magnification	23
6	Advanced use: customing the seminar control file	23
VII	Back(s)lash	27
VIII	Topical tip: Numbering theorems and corollaries in L ^A T _E X	29

IX Malcolm's Gleanings	30
0.1 Book review	30
0.2 Information design journal	32
1 Nonsense	32
X Letters to the editor	34
1 A T _E X front-end in <i>NextStep</i>	34
2 Command line T _E X for ever	34
3 Love L _A T _E X nods	35
XI UKTUG Business Reports	36
1 Membership of UK T _E X Users Group (1994)	36
1.1 Membership Data	36
2 UKTUG accounts 1 October 1992 to 19 August 1993	37
2.1 Statement of Income and Expenditure	37
2.2 Balance sheet	37
2.3 Position with regard to opening balance	37

I Editorial

1 *Baskerville* articles needed

We need material for *Baskerville*! Please send your interesting articles to the editor, and delight fellow T_EX users. Please note the following schedule of copy deadlines:

Issue	Submit material for publication	Submit last-minute notices	PostScript file sent to production team
4.2	Mar 21	Mar 28	Apr 4
4.3	May 23	May 30	Jun 6
4.4	Aug 15	Aug 22	Aug 29
4.5	Oct 17	Oct 24	Oct 31

Each issue of *Baskerville* will have a special theme, although articles on any T_EX-related subject are always welcome. Contributions on the themes for the first half of 1994 are eagerly solicited:

- ☞ *Baskerville* 4.2 will be a special issue on L^AT_EX 2_ε (which may be fully released by then);
- ☞ *Baskerville* 4.3 will be a ‘back to basics’ special issue on mathematical and tabular typesetting.

Baskerville regularly publishes articles answering common T_EX questions, and these are available as technical notes to UKTUG members.

1.1 L^AT_EX 2_ε

Shortly after *Baskerville* 3.1 was sent out, the first release of L^AT_EX 2_ε appeared in the electronic networks, followed after Christmas by the launch of the *L^AT_EX Companion* to tell us how to use it all. 1994 is going to be a good year! Those with access to the Internet can fetch the L^AT_EX 2_ε code from ftp.tex.ac.uk, /tex-archive/macros/latex2e/core. First get the file called ‘features.tex’ and discover what it is all about. UKTUG members without network access can send an SAE to the *Baskerville* editor or the UKTUG treasurer to receive a copy of the macros on Mac or PC disk (do not send us disks, please!).

And there is the conference! Not just the package; not just a book; but *two whole days* of information about L^AT_EX 2_ε, given by the people who wrote it! Grab the application form with this *Baskerville* and fill it in now. We want to see as many UKTUG members there as possible.

1.2 TUG’94

Earthquakes and all, who can resist the chance to visit Southern California, and do T_EX at the same time? From Point Lobos to Hollywood, from Knuth to Clint, from colour to \csname: all the world will be at the **1994 T_EX Users Group Meeting**, to be held in Santa Barbara from the 31st July to the 4th August. The theme this year is just ‘Innovation’ — find out what’s new! Not just papers, but tutorials, debates, bowling matches . . . Leslie Lamport, Tom Rokicki, and Joachim Schrod are keynote speakers at the conference, and you too can still submit a paper by contacting the TUG office, or via the *Baskerville* editor. But paper or not, plan to be there. The *Baskerville* editor will buy a beer for anyone who comes with a copy of this issue . . .

1.3 Colophon

This issue of *Baskerville* deals especially with issues of PostScript, to go with the group’s January meeting on the subject of using PostScript fonts. In the next issue we will include a transcription of the question and answer session, and the full gory details of telling L^AT_EX 2_ε about a new PostScript font.

This issue of the journal was created entirely with the test distribution of L^AT_EX 2_ε and printed on a Hewlett Packard LaserJet 4. It was set in ITC New Baskerville Roman, with Computer Modern Typewriter for literal text. Production and distribution was undertaken in Cambridge by Robin Fairbairns and Jonathan Fine.

II Mixing and matching PostScript fonts

Angus Duggan
Harlequin Ltd.
Barrington Hall
Barrington
Cambridge CB2 5RG
angus@harlequin.co.uk

1 Introduction

The Apple LaserWriter¹ was the product that introduced PostScript² to the world, bringing in its wake a major change in the publishing and printing industry. PostScript is now used everywhere from the home to high-quality printing presses.

PostScript also made scalable font technology popular; instead of using bitmaps for one particular resolution and font size, outlines can be scaled to the size required quickly. The original LaserWriter came with a set of thirteen scalable outline fonts, often known as the “LaserWriter 13”. The LaserWriter fonts are shown in table 1; there are eight faces from Linotype AG (the Times and Helvetica families), four from IBM (the Courier family), and one from Adobe (Symbol). There is one serified text face family (Times) and an accompanying mathematical symbol set (Symbol), one sans-serif text face family (Helvetica), and one monospaced family (Courier).

Times-Roman	Helvetica	Courier
Times-Bold	Helvetica-Bold	Courier-Bold
Times-Italic	Helvetica-Oblique	Courier-Oblique
Times-BoldItalic	Helvetica-BoldOblique	Courier-BoldOblique
Symbol		

Table 1. The original 13 LaserWriter fonts

The choice of typeface styles for the LaserWriter was well-informed—for computer usage a monospaced font is required. The mathematical symbol set, while not complete, encouraged technical writers to invest in PostScript laser printers. A choice of a serif and sans-serif font families was provided for text setting.

Unfortunately, the particular typefaces chosen look terrible when used together.

The weight and width differences between the LaserWriter 13 fonts make the page look blotchy; some words jump out at the reader, some seem to vanish, leaving pale holes in the page. Variations in width, ex-height (the height of the lowercase letters, traditionally measured from the lowercase x), and to a lesser extent capital height make the text less legible. Given these problems with documents composed with the original LaserWriter fonts, why not use other fonts? The answer to this question depends on the purpose of the document.

If a document is being created for personal reading, or for a number of people at one place, then purchasing a set of typefaces which complement each other is an admirable solution. Similarly, if the document is being prepared for publication, purchasing and using the fonts suggested by the designer or printer is advised.

If the PostScript source of the document is being distributed to a wider audience, with no knowledge of the facilities on which it will be printed, then the story is different. The original thirteen LaserWriter fonts are the only fonts that can be guaranteed to be available on any PostScript laser printer or previewer, anywhere in the world.³ There is also the major consideration that giving away the fonts used in a document (even if they are embedded in the document)

¹LaserWriter is a trademark of Apple Computer, Inc.

²PostScript is a trademark of Adobe Systems Incorporated.

³There is a larger set of 35 fonts which was distributed with the LaserWriter Plus which are available on most PostScript printers now. There are still original LaserWriters in use, so I still consider the original 13 fonts as the only guaranteed fonts.

may be illegal under copyright law or licensing agreements. Even if the fonts required to print a document are freely distributable, using the resident fonts has the beneficial side effect of reducing the size of the document.

A growing number of scientific papers, software manuals and technical notes are being made available in PostScript form, and many of these use the ugly combination of the original LaserWriter fonts.

Fortunately, the outline descriptions of fonts in PostScript allow us to do something about the variations in width, height and weight of the fonts. Anamorphic scaling can be used to squash wide characters, stretch short characters, and even slant characters to create obliqued fonts if desired. PostScript also has tricks which can be used to thicken light characters.

2 Matching fonts

To match the width and height of characters from different fonts better, we need to scale the characters *anamorphically*; that is, to alter their aspect ratio. The idea is to make the weight, width, x- and cap height of the fonts more consistent, so that they have a similar colour. (The colour of a font is the amount of ink or toner that is placed on the page when printing; open, wide fonts have a less colour than close, heavy fonts.) The change of letterform still provides cues to the distinction between the elements of the page, but the annoying distractions of light spaces and dark blobs will be removed. The more even height of characters will aid legibility.

TeX cannot scale characters anamorphically, so a small amount of PostScript and virtual font work will be necessary. All of the matching steps I will describe can be done in PostScript alone, but virtual fonts make the process of using TeX with the altered fonts much easier. The examples given are for use with Rokicki's `dvips` driver, but the techniques can be adapted to most DVI to PostScript converters.

2.1 Matching heights

The first step in the process of scaling fonts to match is to sort out the height differences. There are two parts to consider; to match the ex-height so that running text looks good, and to match the capital height so that headers look good. Figure 1 shows the wide variation in width, weight and height of the LaserWriter fonts.

Widths:	Times	Helvetica	Courier
	HLGYXM 43.88pt	HLGYXM 42.23pt	HLGYXM 36.0pt
	hlgyxm 30.56pt	hlgyxm 31.67pt	hlgyxm 36.0pt
Cap heights:	HHH LLL GGG YYY XXX MMM		
Ex-heights:	hhh ll1 ggg yyy xxx mmm		

Figure 1. Width, height and weight variation in the LaserWriter fonts

TeX can be used to match the ex-heights of fonts automatically. The following (relatively obscure) piece of LaTeX code uses the "ex" fontdimen to construct and use a LaTeX2e font definition for a font scaled to match the ex-height of the current font. (This piece of code relies on a macro which does long division of one integer by another, returning a fractional result. For clarity's sake, this macro is not included.)

```

%% need long division routine
\input longdiv.sty % omitted for clarity!

\def\psexfont#1#2#3#4#5{%
  \@tempdima=1ex% ex-height of current font
  \font\tmp=#5\space at\f@size pt
  \tmp\@tempdimb=1ex% ex-height of loaded font
  \@tempcnta\@tempdima \@tempcntb\@tempdimb
  % long division result in ex@scale macro
  \long@divide\ex@scale\@tempcnta\@tempcntb
  \edef\psex@sizes{<->[\ex@scale]#5}%
  \DeclareFontShape{#1}{#2}{#3}{#4}%
    {\psex@sizes}{}%
}

% Times-Roman at same ex-height as current font
\DeclareFontFamily{OT1}{times-xm}{}
\psexfont{OT1}{times-xm}{m}{n}{ptmr}

```

There are disadvantages to this method; it wastes one of T_EX's precious font slots for every font loaded, and sometimes the fontdimen is not accurate, either because of rounding errors in the conversion or because of incorrect information in the original AFM file (Adobe Font Metric — the standard files containing metric information for PostScript fonts).

A method which avoids the loss of the font slot is to create a virtual font containing the scaled font. The amount to scale the font by can be determined either by comparing the ex-height parameters in the original AFM files, or printing out large character samples and measuring the ex-heights if the first method does not give good results. If the second method is used, be sure to print a character with a flat top to it, as characters with rounded tops usually overshoot the x-height deliberately. A virtual font file for the scaled font can be created by using `afm2tfm` to create a virtual property list of the encoded file, and using a utility such as my own `makevpl` (available by anonymous ftp from `ftp.dcs.ed.ac.uk:pub/ajcd/vplutils.tar.Z`) to re-scale the virtual property list file. For example, the following command would scale the Times-Roman font up by 15%.

```
makevpl -at 11.5 ptmr:extex >stimes.vpl
```

This virtual property list file can then be compiled into a virtual font file with `vptovf`. The capital heights can be matched in a similar way.

Rather than arbitrarily matching all of the PostScript fonts to the height of one of them, it is a good idea to match them to the height of the default T_EX font (*i.e.* Computer Modern Roman). If any symbols are required which are not provided by the PostScript fonts, the symbols can be slipped in without the result looking too ugly.

The proportions of ex-height to capital height are different for each of the LaserWriter fonts, so a single scaling factor will in general not be sufficient to match both of the ex-height and capital height. The easiest way to get different scaling factors for the capitals and lowercase is to make two virtual fonts with the desired scalings as described above, and merge them using a utility such as `makevpl` or Alan Jeffrey's `fontinst`. It is not desirable to match both the ex-heights and cap-heights of Courier unless the widths are adjusted to keep the matched font monospaced.

This matching process only needs doing for one member out of each family of fonts; the other members should use the same scaling ratios to stay consistent with each other. Figure 2 shows the results of matching the ex-heights and cap-heights of the LaserWriter fonts.

```

Matched cap heights
  HHH LLL GGG YYY XXX MMM
  hhh ll l ggg yyy xxx mmm

Matched ex heights
  HHH LLL GGG YYY XXX MMM
  hhh ll l ggg yyy xxx mmm

Matched cap and ex heights
  HHH LLL GGG YYY XXX MMM
  hhh ll l ggg yyy xxx mmm

```

Figure 2. LaserWriter fonts with matched ex- and/or cap-heights

2.2 Matching widths

Matching the widths of the fonts is one of the easiest effects to achieve. The `afm2tfm` program has an option to extend or compress PostScript fonts. For example, a virtual font for the Times-Roman font extended to 110% of its normal width by the command can be created by the command:

```
afm2tfm Times-Roman -e 1.1 -v ptmr Times-Ext
```

This will create a virtual property list file called `ptmr.vpl`, which can be scaled up or down as described above to match the heights of the fonts, and then compiled into a virtual font with `vptovf`. A T_EX font metric (TFM) file will also be generated, which should be put in an appropriate directory for T_EX to find it. A line needs to be inserted into the `psfonts.map` file to tell `dvips` about the pseudo-font Times-Ext, which it will create from the Times-Roman base font if it is used:

```
Times-Ext "/Times-Roman 1.1 ExtendFont"
```

Extending or compressing fonts in this way has the undesirable effect of altering the ratio of the horizontal strokes to the vertical strokes; these fonts are not true compressed or extended designs, and there is unfortunately nothing that can be done in PostScript to counteract this effect.

When matching the font widths, it is undesirable to make all of the em-widths the same; the design of Courier requires more space than Times Roman or Helvetica. What is desired is a more acceptable balance of widths so that the most compact font (Times) does not look bad when used beside the widest font (Courier). Figure 3 shows a comparison of the original widths of the characters and a possible choice of new widths.

Normal laserwriter fonts					
Times		Helvetica		Courier	
HLGYXM	43.88pt	HLGYXM	42.23pt	HLGYXM	36.0pt
hlgxym	30.56pt	hlgxym	31.67pt	hlgxym	36.0pt
Height and width matched laserwriter fonts					
Times matched	105 %	Helvetica matched	100 %	Courier matched	90 %
HLGYXM	47.1816pt	HLGYXM	39.93031pt	HLGYXM	32.82541pt
hlgxym	30.6948pt	hlgxym	26.15741pt	hlgxym	32.80664pt

Figure 3. Comparison of original widths and new widths

Font family	Cap height ratio	ex-height ratio	Extension	Stroke Width
Times	10.24	9.57	105 %	
Helvetica	9.46	8.26		
Courier	12.05	10.13	90 %	20

Figure 4. Parameters

2.3 Matching weight

Matching the weights of the fonts is one of the most awkward effects to achieve. The intention is to achieve a more even gray colour from pages with mixed fonts. Courier is the main problem in this respect; most versions of Courier are significantly lighter than Times Roman and Helvetica. The versions of Courier from some foundries (e.g. Bitstream) are heavier than the Adobe version usually found in PostScript printers.

Some early versions of the Courier fonts had a painting type 1, meaning that the font was rendered by a single line down the centre of each stroke; these fonts could be made lighter or darker by increasing the width of the line used. More recent versions of Courier are defined as outlines, with which the same trick cannot be used. A similar effect can be achieved by rendering the character outline with an increased linewidth on top of filled character. This can be done by the following PostScript commands which create a new type 3 (user defined) font which places both outlined and filled characters on top of each other:

```

%!
% Courier-Heavy font definition
/Courier-Heavy
10 dict begin
  /FontType 3 def
  /FontMatrix [0.001 0 0 0.001 0 0] def
  /FontName /Courier-Heavy def
  /Courier dup findfont 1000 scalefont def
  /Encoding Courier /Encoding get def
  /FontBBox [ % adjust for outline width
    Courier /FontBBox get aload pop
    2 {10 add 4 1 roll} repeat
    2 {10 sub 4 1 roll} repeat
  ] def
  /Courier-Outline dup % outlined Courier
  Courier dup length 1 add dict begin
    {
      1 index /PaintType eq {

```

```

        pop 2 def
      } {
        1 index /FID eq {
          pop pop
        } {
          def
        } ifelse
      } ifelse
    } forall
  /StrokeWidth 20 def
  currentdict
end definefont def
/charstring ( ) def % string for charcode
/BuildChar { % dict charcode
  exch begin
    charstring dup 0 4 -1 roll put
    Courier setfont
    dup stringwidth FontBBox aload pop
    setcachedevice % set char metrics
    0 0 moveto
    gsave
      dup show % fill character
    grestore
    Courier-Outline setfont
    show % draw outline
  end
} def
currentdict
end
definefont pop

```

If the PostScript character cache is large enough this method will not cause too much of a slowdown because each thickened character will be constructed only once for each size used. The size of each character cannot be easily extracted from the original Courier font, so each character takes the maximum size of cache needed, determined from the original font's bounding box.

Dvips can be made aware of this new font by putting the following line in the `psfonts.map` file:

```
rpcrsb Courier-Heavy <coursb.pf3
```

The font metrics of this heavier font are the same as the original Courier font, so the AFM file for the original font can be copied and used to generate the virtual font and TFM file needed for use with \TeX . Figure 5 shows how the weight of the new heavy version of Courier compared with the original Courier and Courier-Bold.

Courier	HLGYXMhlgxym 72.0pt
Courier-Heavy	HLGYXMhlgxym 72.0pt
Courier-Bold	HLGYXMhlgxym 72.0pt

Figure 5. Comparison of Courier weights

This technique increases the width and the height of the character by the width of the stroke used on the outlined character, so another iteration around the height and width matching steps may be needed to improve the results. The extra width is added on all sides of the character, so the baseline may also need adjusting, by altering the coordinates of the `moveto` command in the PostScript header.

This technique can really only be used to thicken characters. If characters are thinned by painting a white outline over the filled character, it will not yield a satisfactory result. At small sizes the inside of the character outline may not have any space in it, and so the final character will have gaps in it.

2.4 Results

The techniques used here do not add much (if anything) to the final size of documents; the PostScript header files downloaded are very small. Table 4 shows the height and width ratios which I am currently using; fine tuning these

ratios for nicer looking output will take some more time. The Symbol font should probably follow the treatment of the Times family, since it was designed to be complementary to Times.

And finally, a sample of the resulting output. Matching both the ex-height and cap height without adjusting their relative widths tends to make the capitals look wider; some more experiments to reduce this effect will be necessary.

Times New Roman, or Times Roman as it is often known, was designed by the typographer Stanley Morison for The Times. The design was based on Monotype Plantin 113; Morison did not draw the designs himself (he was not a designer), but got an artist working for The Times to draw it and revise the drawings until he was satisfied. Helvetica was originally produced by the Swiss Haas'sche foundry in 1957 under the name Neue Haas-Grotesk, to a design by Max Meidinger. It was recut by the German Linotype firm, who renamed it Helvetica. Courier was designed in 1952 for IBM by Howard Kettler, merging the geometric "egyptians" Stymie, Memphis, Beton, and Rockwell. The names **ptmr**, **phvr**, and **pcrr** are used for the normal shapes and weights of Times, Helvetica and Courier in Karl Berry's font naming scheme, used by **dvips**.

3 Conclusion

Anamorphic scaling by its very nature distorts the shapes of the characters. In general, distorting a pleasing typeface will not give another pleasing typeface. The techniques described in this note make minor distortions to a set of typefaces in order to make their use together more pleasing. These techniques are only useful in limited circumstances, *i.e.* when the only fonts you can rely on using are the base LaserWriter fonts. As noted in the introduction, there are circumstances where this is the case, and in these cases almost anything is better than the ugly sight of Times, Helvetica and Courier mixed together in their natural state.

III Building virtual fonts with ‘fontinst’

Alan Jeffrey
alanje@cogs.susx.ac.uk

1 Introduction

This document gives a brief overview of the `fontinst` package. The `fontinst` package is used to build *virtual fonts* (VFs) which allow PostScript fonts to be used as drop-in replacements for the Computer Modern fonts in \TeX .

Below, I’ll describe VFs briefly, and describe how they can be built using the `fontinst` package.

2 A problem with fonts

One of the biggest problems about using fonts in \TeX is *encodings*, that is the order the characters come in the font. For example, the default encoding for Adobe’s Times-Roman font is the ‘Adobe Standard encoding’:

```
~ ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; i = i ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ " ] ^ `
‘ a b c d e f g h i j k l m n o
p q r s t u v w x y z - — ” ~
```

The default encoding for \TeX , however, is the ‘ \TeX text encoding’. The Adobe Times-Roman font in the ‘ \TeX text encoding’ is:⁴

```
Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω ff f f
■
■ ! # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; = ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ ]
‘ a b c d e f g h i j k l m n o
p q r s t u v w x y z
```

There are many other competing encodings: ‘ISO Latin-1’, ‘ \TeX extended text’ (or ‘Cork’), ‘Macintosh’, the list is seemingly endless.

In addition, different encodings contain different glyphs. The \TeX text encoding is supposed to contain a dotless ‘j’ character, and a slash for building ‘Ÿ’ and ‘Ł’, but very few fonts contain these characters, and their places are taken by black squares above.

The problem of incompatible font encodings is addressed in \TeX by *virtual fonts*.

⁴The \TeX nically minded may note that the glyphs ‘Ÿ’ and ‘Ł’ are not normally in the ‘ \TeX text encoding’. This is because Computer Modern has a special ‘Ÿ-slash’ glyph for building ‘Ÿ’ and ‘Ł’, which Adobe Times-Roman does not have. Its place is therefore taken by a black square, and there are ligatures with ‘Ÿ’ and ‘Ł’ to produce ‘Ÿ’ and ‘Ł’. Thus this font is drop-in compatible with Computer Modern, despite the lack of an ‘Ÿ-slash’ glyph.

3 A solution: virtual fonts

As far as \TeX is concerned a virtual font (VF) is a font like any other. It has a \TeX font metric file, which contains the dimensions of each character, together with ligatures, kerning, and other typographical information.

However, a VF does not have an associated bitmap, Type 1 font, TrueType font, or other information about what the font should look like.

Instead, a VF has an associated `.vf` file, which contains a small fragment of `dvi` file for each character in the font. This `dvi` fragment may contain characters from other fonts, rules or `\specials`.

For example, the ‘Adobe Standard’ encoded Times-Roman font above is a ‘raw’ Type 1 font, but the ‘ \TeX text’ encoded Times-Roman font is a virtual font.

- The ‘ff’, ‘ffi’ and ‘fff’ ligatures are faked by putting an ‘f’ next to an ‘f’, ‘fi’ or ‘ff’.
- The missing ‘dotless j’ and ‘i-slash’ are rules, together with a ‘Warning: missing glyph’ `\special`.
- The Greek upper case come from the Symbol Type 1 font.
- The other characters come from the Times-Roman Type 1 font.

Any `dvi` driver which understands VFs and can use Type 1 fonts can use the \TeX text Times-Roman VF as a drop-in replacement for Computer Modern.

4 A problem with virtual fonts

One stumbling block about using VFs is that they are not very easy to generate. Despite having been in existence for four years, there are very few tools for creating VFs.

The most important tool is Knuth’s `vptovf`, which converts *Virtual Property Lists* (VPLs) into VFs. Unfortunately, the VPL language is rather opaque; for example the VPL code for the Adobe Times character ‘ff’ is:

```
(CHARACTER D 11
  (CHARWD R 6.47998)
  (CHARHT R 6.81995)
  (CHARDP R 0.0)
  (MAP
    (SELECTFONT D 1)
    (SETCHAR D 102)
    (MOVERIGHT R -0.17993)
    (SETCHAR D 102)
  )
)
```

Editing VPL files by hand is something of a black art, and there are few tools for manipulating them.

The main tool for generating VPLs is Rokicki’s `afm2tfm`, which converts the *Adobe Font Metric* (AFM) files which come with every PostScript font into VPLs. Unfortunately, `afm2tfm` cannot produce fonts with more than one raw font (for example the ‘ \TeX text’ encoded Times-Roman uses Symbol for the upper case Greek) and had problems with math fonts.

5 A solution: the ‘fontinst’ package

The `fontinst` package is designed to read AFMs and produce VPLs. It:

- Is written in \TeX , for maximum portability (at the cost of speed).
- Supports the \TeX text, \TeX math, and extended \TeX text encoding.
- Allows fonts to be generated in an arbitrary encoding, with arbitrary ‘fake’ characters—for example the ‘ff’ character can be faked if necessary by putting an ‘f’ next to a ‘f’.
- Allows caps and small caps fonts with letter spacing and kerning.
- Allows kerning to be shared between characters; for example ‘Å’ can be kerned as if it were an ‘A’. This is useful, since many PostScript fonts only include kerning information for characters without accents.
- Allows the generation of math fonts.
- Allows more than one PostScript font to contribute to a \TeX font; for example the ‘ff’ ligature can be taken from the Expert encoding, if you have it.
- Automatically generates an `fd` file for use with $\LaTeX 2\epsilon$.

The `fontinst` package is available as freeware from the CTAN archives, along with a selection of VFs which have been generated with `fontinst`.

Version 0.19 of `fontinst` is described in the proceedings of the Aston TUG AGM (*TUGboat* 14(3)). This description is now largely out of date.

The VFs generated by `fontinst` will be the standard VFs for use with Sebastian Rahtz's `psnfss` package for L^AT_EX 2_ε.

6 Using the 'fontinst' package

The `fontinst` package comes with full documentation in the file `fontinst.tex`. The simplest way to start to use `fontinst` is to edit the file `fonttime.tex`, shown in Table 1. This tells T_EX to create the Adobe Times Roman fonts in the 'T_EX extended text' (T1) encoding, using the files:

- `ptmr0.afm`, `ptmri0.afm`, `ptmb0.afm` and `ptmbi0.afm`, the Times-Roman AFM files.
- `latin.mtx`, the *T_EX metric* file containing the default Latin characters.
- `T1.etx` and `T1c.etx`, the *T_EX encoding* files containing the 'T_EX extended text' and 'T_EX extended tex caps & small caps' encodings.

This produces a number of PL and VPL fonts, which can be converted into T_EX fonts using `pltotf` and `vptovf`.

For example by replacing every occurrence of `ptm` by `ppl` you can install the Adobe Palatino fonts.

If you generate any fonts with `fontinst` which you think other people might want to use, please send them to me, and if I like them, I'll include them in the `fontinst` contributors directory.

```
\input fontinst.sty

\needsfontinstversion{1.303}

\installfonts
  \installfamily{T1}{ptm}{}
  \installfont{ptmrq}{ptmr0,latin}{T1}{T1}{ptm}{m}{n}{}
  \installfont{ptmrcq}{ptmr0,latin}{T1c}{T1}{ptm}{m}{sc}{}
  \installfont{ptmriq}{ptmri0,latin}{T1}{T1}{ptm}{m}{it}{}
  \installfont{ptmbq}{ptmb0,latin}{T1}{T1}{ptm}{bx}{n}{}
  \installfont{ptmbcq}{ptmb0,latin}{T1c}{T1}{ptm}{bx}{sc}{}
  \installfont{ptmbiq}{ptmbi0,latin}{T1}{T1}{ptm}{bx}{it}{}
\endinstallfonts

\bye
```

Table 1. The file `fonttime.tex`

IV Do you really need virtual fonts?

Berthold Horn

Y&Y 71172.524@compuserve.com

Introduction

Since many people feel very strongly about these issues, I'll need to go into some detail to try and sway their opinion. I'll first discuss why virtual fonts are *not* needed for use of non-CM fonts — *or* for reencoding. Then I'll explain what virtual fonts *may* actually be useful for (and why even for those purposes there are better ways of going about things).

Part of the confusion may result from lumping together of virtual font support with support for re-encoding of fonts. One most *definitely* needs re-encoding, but it is not part of VF, and in fact, virtual fonts *per se* are inadequate for this task (see below).

This is a deeply religious issue, so I don't expect to make too many instant converts. We all fall in love with the software tools we use and often assume that the way they do things in the *only* way they can be done, or perhaps even that they implement the *best* way. Unfortunately, to make my points, I'll be stepping on some toes (lightly I hope).

By the way, many of the points I wish to make cannot be made without referring to specific programs. I hope what I say will not sound *too* much like advertising.

Naturally, there will be many that disagree with what I say here. I'll be happy (or maybe not!) to read their comments and respond. From InterNet, send email to 71172.524@compuserve.com.

People say to me:

But you know that we (T_EX people) need these virtual fonts to cope with non-T_EX fonts.

The key point is that this statement is just plain wrong! The fact is that *one* particular implementation of a printer driver (DVIPS) does *force* the user to use virtual fonts to do just about anything. This is a valid approach, but one that requires the user to deal with more complexity than is really needed. It is *not* reasonable to generalize from this example to all d_vi processors, based on the limitations of a particular implementation.

Why does DVIPS need virtual fonts?

The need for VF in DVIPS is mostly a result of the fact that a companion utility (AFM2TFM) is unable to make proper TFM files complete with ligatures and kerning *without* using virtual fonts (AFM2TFM also is unable to make TFM files for math fonts).

In fact, you *can* use DVIPS without VF if you use a utility other than AFM2TFM to make TFM files. For example, some people buying Y&Y fonts for use with DVIPS use ready-made TFM files supplied with the fonts that do not require VF (or they run AFMtoTFM on a PC to make new TFM files for whatever encoding they desire).

A second reason for the forced use of VF in DVIPS is the use of a somewhat contorted way of dealing with the font encoding issue — three mappings instead of just one, see later. (We won't even talk about the odd way this was handled in old versions of DVIPS).

CM and non-CM fonts can be used without virtual fonts

It is possible to use non-CM and CM fonts (TrueType, 'PostScript' Type 1, BitStream Speedo etc) *without* resorting to virtual fonts, provided you have a driver that can do this (*e.g.* DVIPSONE and d_viWindo) and a companion utility (AFMtoTFM) that can make proper TFM files complete with ligatures and kerning *without* needing VF.

By the way, the encoding issue is a very important and often misunderstood item. Since T_EX thinks of characters *only* in terms of numbers, and since the CM fonts have hard-wired encoding, many T_EX users are unaware of what this is all about. Someone always working with the same programs and on the same platform may not be aware that there is an issue. But that is another story. Just keep in mind that a file contains *numeric codes* — *not* characters. There must be conventions for what glyph each numeric code corresponds to — and there is no single 'right' encoding.

It is *not* necessary to use virtual fonts to reencode a font.

Users of Y&Y software use scalable outline fonts *without* VF. Y&Y doesn't sell or support PK bitmapped fonts (except in some half-baked way). So it should be obvious that 'it works' — for otherwise they would have been out of business long ago!

One can use non-CM fonts — with full support for ligatures, kerning *and* reencoding — without resorting to virtual fonts.

A large fraction of sales from Y&Y are to service bureaus, publishers, and T_EX consultants — ‘power users’ that need all the most advanced features. If VF was needed to do any of the things they want to do, then you can bet that it would be supported!

The need for font re-encoding and the inability of VF to provide proper re-encoding

The most commonly claimed reason for need to use VF is that font encoding must be controlled. Now the virtual font itself — like everything in T_EX — treats characters merely as *numbers* — it has no concept of character other than as a number. Hence VF itself can *only* permute the numbers of from 0–255. That is, it can move characters around in the space of integers from 0–255.

But: most fonts have many unencoded characters. There may be 224 or 500 or even over a thousand characters, yet only 170 may show up in the ‘raw’ encoding of the font. To use the font properly it has to be re-encoded. There is a ‘cmap’ or ‘encoding vector’ that maps the integers from 0–255 to characters (usually specified by some mnemonic name like ‘space’). To use such a font properly, the dvi printer driver or dvi previewer has to be able to reencode the font to a user specified encoding vector. Note that this has nothing to do with virtual fonts, it is a capability needed in the dvi processor whether or not virtual fonts are supported — and in fact cannot be provided by VF itself.

To state this emphatically: virtual fonts themselves are inadequate for reencoding, since they cannot make unencoded characters accessible. And once your dvi processor has its own mechanism for doing *reencoding*, there is no longer a need for VF to attempt to do this!

Let me show how easy this encoding business really can be. Imagine an ASCII file with a list of numbers and character name pairs. This file contains up to 256 lines such as the following:

```
32      space
33      exclam
34      quotedbl
35      numbersign
...

```

This is an encoding vector file. It fully defines the encoding to be used — in a totally clear and explicit way. Now such an encoding vector file can be used by the dvi processor (the user specifies for a font that needs reencoding, which of these encoding vectors to use), as well as by the utility used to create the TFM metric file.

Compare this to DVIPS’s complex mechanism of ‘input encoding’, ‘output encoding’, plus virtual font remapping (permuting 0–255). *Three* ‘mappings’, where just *one* is perfectly adequate!

If we don’t need VF for reencoding, then what are virtual fonts good for?

1. Making a fake smallcaps font;
2. Add new composite/accented characters;
3. Making new fonts that contain characters from two existing fonts;
4. Changing the side-bearings and advance widths of characters in a font;
5. Achieving weird and wonderful effects by packaging T_EX dvi commands for drawing rules and \specials as ‘characters’

What are the drawbacks of using virtual fonts for these purposes?

1. The font designer will be in pain when he sees his creation mutated using virtual fonts to create fake smallcaps! A smallcaps font should have properly designed ‘small caps letters’, *not* scaled replicas of the uppercase letters. Making a smallcaps font this way is not quite as evil as making a bold font by smearing a regular face, but it comes close. . . Admittedly, many fonts do not have companion ‘expert fonts’ or smallcaps versions, so one is tempted to make up a fake smallcaps font, but its not a good idea.
2. Most text fonts contain 58 ‘standard’ accented/composite characters that cover ISO Latin 1 and some more. These can be easily used directly *if* your dvi driver provides for reencoding. Curiously DVIPS/AFM2TFM instead uses the virtual font mechanism to compose the accented character from base and accent. This is not a good idea since the designer of a quality font often makes a composite that is not *exactly* achievable by superimposing base and accent. Aring, Ccedilla, and ccedilla are particular cases of glyphs usually *not* made by superimposing base and accent, but by designing an outline. Also, the rendering at some resolutions will not be as good, since the hinting for the composite can take into account interactions between the base and accent. This is *not* possible if the two parts are drawn separately. (By the way, AFMtoTFM can be used to insert convenient pseudo ligatures for the accented characters in the TFM file).

3. Combining parts of two fonts seems like a legitimate use for virtual fonts. It comes in handy, for example, when an ‘expert font’ contains the small caps letters, but not the upper case letters. But see below.

So what are the drawbacks of using virtual fonts for these purposes (And how can one achieve the same results some other way)?

The main problem is: Only \TeX knows anything about virtual fonts.

Now if \TeX is all you ever use, *and* if you don’t use text in your illustrations, then that may be just fine with you. But in a lot of professional work, \TeX does not live in isolation. Illustrations are created using graphics applications of all sorts. These can be inserted into the text in EPSF or TIFF or other form. (Where we come to meet the nightmare of non-standardization of `\special` – but that is another diatribe. . .)

Now if the illustration has any text in it, it is usually desirable to have the font used in the nomenclature match the text font. So the graphic application has to be able to use the same fonts as \TeX . Hence PK bitmapped fonts are not useful, one needs to use fonts in some established industry standard form such as Type 1 or TrueType (note that virtually all fonts commonly used with \TeX are now available in T1 format, including CM, AMS, \LaTeX , SUTeX etc). And this won’t work if the font is a ‘virtual font’. What to do?

Font manipulation tools

Well, in the \TeX world we tend to be somewhat myopic. We try to do everything in \TeX , or using tools that come with \TeX . Sometimes we go through amazing contortions to do this, even when it can be done quite easily some other way (for example making graphs by drawing millions of dots in \TeX rather than using PostScript).

There *are* tools available for manipulation fonts in Adobe Type 1 format. These create *real* fonts that can be used not just with \TeX . Such tools can:

1. combine characters from different fonts into one font;
2. create new composite/accented characters (add ISO Latin 2 glyphs say);
3. make obliqued versions of a font (although the designer may not agree);
4. adjust the side-bearings and advance widths of characters;
5. and about a dozen other things. . .

It should be clear from the above that I have pretty strong feelings on this issue! But that is only to counter the pretty strong feelings many users — particularly in the Unix / University world — have on this issue!

0.1 Easy, totally general reencoding

In `dviWindo` and `DVIPSONE` you can use *any* encoding (for printing *and* for on screen display), and using arbitrary encoding is as simple as adding a line like

```
tir Times-Roman *remap* isolat1l
```

to a ‘font substitution’ file — or, running a batch file called `encode` (this should all be one one line):

```
encode isolat1l c:\afm c:\tfm c:\psfonts
c:\windows W tir tii tib tibi
```

(the latter takes care of all four styles of Times-Roman). What could be simpler? This is not to say that it is easy to implement this! In fact, the operating systems, PostScript printer drivers, Adobe Type Manager, and clone printers conspire to actually make it very hard. But the user need not suffer!

MathTime version 1.1

Lets talk about the MathTime fonts for a second. It certainly saved some work to not have to make up the glyphs for the letters in the math italic font MTMI, but instead to ‘borrow’ them from Times-Italic (with major changed in side-bearings and advance widths). And virtual fonts make it possible to splice together RMTMI and Times-Italic to make a MTMI font.

However, this has been the source of very many headaches! Virtual font fanatics please pay close attention!

First of all, there are eight (8) versions of true Adobe Times-Italic alone. And different printers have built in different versions. For example, many TI printers use the old 001.002 version, while most QMS printers use the (almost) latest version 001.007. So what you say? Well, while the advance widths of the characters have not changed since 001.002 (thank god), the glyph shapes *and* side-bearings have. Just for example, the lower case ‘z’ in 001.002 has a short flat bottom right on the baseline, while the ‘z’ in 001.007 has a distinctive ‘swash’ bottom which descends way below the baseline and comes much further over to the right where it ends in a bulb. Subscript position designed to work with 001.002 will cause collision when used with 001.007! Conversely, a subscript on ‘z’ will look too far away when used with version 001.002, because the fonts were actually tuned for 001.007.

We won't even talk about 'clones' of Times, such as the one by BitStream, which are used in some low-cost laser printers. These have entirely different 'color' for a start and different side-bearings and shapes.

And what about that Linotronic to which you have entrusted generation of the final copy of your book (at \$3–\$8 per page)? What version of Times-Italic is it using? Are you willing to risk it?

So what is the solution? Don't use virtual fonts!

The IBM PC version of MathTime version 1.1 from Y&Y comes with true Adobe Times 001.007 and an installation procedure that creates a *real* MTMI that (i) can be used with any application, not just \TeX , and (ii) has 'wired in' the version of Times-Italic for which RMTMI was designed. No 'surprises' are possible!

By the way, service bureaus are in the habit of asking for the fonts separately from the PostScript file (this is a hang over from a bygone era, but that is another story). And they want a *real* font – their image setter doesn't know anything about *virtual* fonts.

Unfortunately the tools for combining RMTMI and Times-Italic, adjusting side-bearings and advance widths etc are quite sophisticated and not available on other platforms (particularly if you care about hinting, since most tools for manipulating fonts destroy the hinting). So many users find themselves in the unfortunate position of having to buy the IBM PC version *and* utilities for converting from PC to Mac or Unix format.

Some remaining minor issues

There are some other, less important issues. Implementation of VF in the `dvi` processor creates a significant performance hit. The seriousness of this depends on the cleverness of the implementor, and for printer drivers its probably not a big concern (since 300 milli-seconds per page is not noticeably slower than 150 milli-seconds per page). The performance hit in `dvi` previewers is more serious. Try Textures with a file that calls for VF fonts versus the same basic text with non-VF fonts (which gives up some of the advantages of the assembly language coding in Textures 1.6).

V Further thoughts on virtual fonts ...

Yannis Haralambous

Yannis.Haralambous@univ-lille1.fr

In a paper I published in 1993 (“Virtual fonts: great fun, not for grand wizards only!”) I have already addressed many of Berthold Horn’s arguments; nevertheless I would like to take the opportunity to respond a little further.

The basic argument of Horn is that PostScript drivers can reencode fonts, so that virtual fonts are unnecessary for plain reencoding. This is certainly true, but unfortunately *only in a very limited scope*.

To produce accented letters, many PostScript fonts contain “composite character data”: these are just translation coordinates for character parts, which will be composed to produce the result. PostScript interpreters know about these characters and can automatically take care of characters defined in that way in the font. But these accented characters cover only the West European range (excluding of course Welsh and Maltese); a PostScript interpreter is not clever enough to define a new composite character, for example, a *z* as needed in Polish, or a *w* as needed in Welsh. Alan Jeffrey’s utility can do this very easily; this is far more than just plain reencoding, but is everyday practice for virtual fonts.

The reader may have little interest in exotic languages (like the Polish or Welsh in the previous paragraph); but DC fonts have additional features which are implemented in virtual Cork-like PostScript fonts by Alan Jeffrey’s virtual font creation tool:

- certain characters may need special kerning (such as the little zero for the perthousand sign, the German single and double opening quotes etc.);
- some symbols (such as $_$, $\$$, \pounds) may be missing; these can be taken from other fonts;
- the glyph ‘-’ is used twice: once for the text dash, and once for the hyphenation dash (cf. [5] why these are separate characters); I doubt that reencoding can assign the same glyph to two different positions (?);
- the uppercase version of β is made out of two ‘S’ letters; this is too much to ask for a poor PostScript interpreter. . .
- PostScript fonts can contain ligatures, but not *smart* ligatures: if you want your Dutch ‘*e*’ to become an ‘*e*’ at the end of a line, you need a begin-of-word ligature, something trivial for \TeX .

Virtual fonts are one of the most important aspects of the \TeX system. This is not just the case for exotic situations; I voluntarily do not speak of Arabic and other extremely important uses of virtual fonts in oriental languages; virtual fonts are important for all of us Occidental language writers. A PostScript font has poor typographical properties (no smart ligatures, restricted character composition, since you have to remain in the same font and the same size, etc.); by the use of virtual fonts, \TeX ’s typographical possibilities can be added to the font: a virtual font structure makes a PostScript font richer.

... It is not necessary to use virtual fonts to reencode a font. . .

True. But we want more than just re-encoding: word processors like Word or WordPerfect simply reencode fonts; \TeX can do more out of a PostScript font, and the proof can be found in the virtual fonts made by Alan Jeffrey’s utility.

Horn states that virtual fonts cannot make unencoded characters accessible. This is certainly true, and—as he says—this issue is solely solved by reencoding of fonts. But it is not an argument against the use of virtual fonts: one can always reencode a font into some universal encoding, for example `ISOLat1n1`. The latter might be universal, but is still not Cork. Some extra work must be done to make a Cork-like font out of it, and this is best handled by a virtual font.

I agree that reencoding is the only way to make characters such as the Thorn or Eth appear; but it should be only one step of the printing progress, between others.

... Making a fake smallcaps font. . . A smallcaps font should have properly designed small caps letters. . .

Horn is *absolutely* right when he says that one should rather adopt an ‘Expert’ font than faking small caps by scaling regular caps. Now, suppose you buy that Expert font. What’s the next step? You will discover that Expert fonts do not contain uppercase letters (cf. [1], page 602). Is there a possibility of merging the regular and expert fonts into what we expect to be small-caps font, using plain reencoding? I’m afraid not, since reencoding means “assigning glyphs to positions, *inside* a font” and not “*between* fonts”; you will have to use virtual fonts. Alan Jeffrey’s utility automatically finds out if there is an expert font and what characters it contains. It then either creates fake small caps, or takes (just)

reprinted from Baskerville

Volume 4, Number 1

the real small caps from the expert font. Furthermore, there cannot possibly be any kerning pairs between small caps and uppercase letters in the PostScript fonts since these are not contained in the same 256-characters table. But the \TeX virtual font can contain such kerning pairs (some of them, like TA or VA being quite important); after a little experimenting the quality-conscious user will easily add the most important kerning pairs to the VPL file.

... Only \TeX knows anything about virtual fonts. . .

Actually Horn is not saying “do not use virtual fonts”, but “do not use PK fonts”, since “they will never be able to enter into illustrations”. He continues:

... Well, in the \TeX world we tend to be somewhat myopic. Hence PK bitmapped fonts are not useful, one needs to use fonts in some established industry standard form such as Type 1 or TrueType. . .

To my (myopic?) eyes the PostScript world seems much more myopic. For many years poor PostScript fonts have been designed; in the meantime the \TeX community kept saying “fonts without metaness are anti-typographic” but (apart from a few exceptions, like Jacques André’s papers on pointsize dependent PostScript font code, cf. [2] and [3]) metaness seemed to be tabou outside the \TeX world; suddenly two years ago the goddess Adobe declared that fonts without metaness are no good, and introduced a new object of veneration: Multiple Master fonts. These are extremely complex and memory consuming, but still much poorer than METAFONT created fonts; nevertheless (myopic) PostScript font users consider them as the *non plus ultra*.

METAFONT can do things PostScript cannot even dream of. Try to adjust gray density of Hindi, Arabic and Latin text on the same page with PostScript fonts. Horn says that *scaled small caps are fake small caps*. I say: scaled fonts are always faked: *all PostScript fonts are faked when used in a size different that their design size* (and most of the time we don’t even know what that design size is; these are things the customer had better not find out. . .).

Erik-Jan Vens has developed a tool to convert PostScript fonts to METAFONT. This opens new horizons to digital typography, since we can manipulate these fonts using METAFONT tools. DVI drivers which do not read PK files will never take advantage of these methods (cf. [6]).

... (note that virtually all fonts commonly used with \TeX are now available in Type1 format, including CM, AMS etc. . . .

But I would add the word ‘obsolete’ after ‘CM’: IMHO, CM fonts are *just good enough to write English*. It is quite an irony that the text you are reading this very moment is written in English⁵, but here in Europe hundreds of millions of people communicate through other languages, which cannot be hyphenated with CM fonts (cf. [4]). Of course, nobody will ever force the only-English-writing- \TeX -user to use DC fonts, but can progress be stopped?

Finally, Horn omits a very important issue: there is a tool called DVICopy (written by Peter Breitenlohner). Using DVICopy one can *de-virtualize* a document, that is *replace characters from a virtual font by the real character(s) they represent*. This eliminates all communication problems: suppose I have created a document using a PostScript font, which itself is encoded in some standard encoding. For this I have used a virtual font, which my correspondent might not necessarily have. By devirtualizing my DVI file, I obtain a new DVI file which uses precisely and exclusively the real font on which my virtual font was based. In the case of PostScript fonts, this means that if my virtual fonts were constructed upon Adobe Standard encoded PostScript fonts (that’s the usual encoding for PostScript fonts) a de-virtualized DVI file will contain references to these original PostScript fonts only, which makes it as portable as a DVI file can be.

I would like to close this paper by some general remarks on the “ \TeX world”, as I see it: I don’t believe \TeX users are myopic or isolated from the rest of the world. On the contrary, they see problems that commercial programs can barely handle, and solve them through \TeX without even making much noise about it. In the last few years it has happened that there has been much more development in public domain \TeX ware, than in commercial software around \TeX . Important innovations have always appeared first in public domain software⁶. Many times commercial software has adopted those innovations; but there are also many \TeX features yet undiscovered by the commercial world—and many commercial products still at the stone age of \TeX . This is a sad consequence of the fact that \TeX is a public domain program, whose “official” development has stopped, and has been unofficially taken over by mostly unorganized volunteers: this makes both the charm and the pain of \TeX history. Virtual fonts may be one of the innovations that all commercial products haven’t adopted yet—or maybe not; but we should think twice before giving

⁵Bien que j’aurais pu changer de langue à tout instant ; si j’écris en anglais ce n’est pas pour la gloire de la langue mais pour faciliter la lecture au lecteur britannique. Passons. . .

⁶With a single exception: the user interface. Public domain software is never as user-friendly as commercial ones.

away virtual fonts in exchange for something poorer (PostScript font reencoding), when we can equally well use both at the same time, and produce even better results.

References

- [1] Adobe Systems International, PostScript Language Reference Manual, second edition, Addison Wesley, 1990.
- [2] Jacques André, ‘Adapting Character Shape to Point Size’, *PostScript Review*, April 1991.
- [3] Jacques André and Irène Vatton, ‘Contextual Typesetting of Mathematical Symbols—Taking Care of Optical Scaling’, submitted to *Electronic Publishing*, 1993.
- [4] Yannis Haralambous, ‘ \TeX conventions concerning languages’, *\TeX and TUG News*, Volume 1, Number 4, 1992.
- [5] Yannis Haralambous, ‘Virtual Fonts: Great Fun, not for Wizards Only’, *Minutes and ApendiceS* 93.1, Nederlandstalige \TeX Gebruikersgroep, 1993.
- [6] Yannis Haralambous, ‘Parameterization of PostScript fonts through METAFONT — an alternative to Adobe’s multiple-master fonts’, to appear in *Proceedings of Raster Imaging and Digital Typography*, Darmstadt 1994.

VI Colour slides with L^AT_EX and seminar

Michel Goossens (CERN)
Sebastian Rahtz (ArchaeoInformatica)

1 Slides and L^AT_EX

Many L^AT_EX users want to take advantage of T_EX's high-quality typesetting when they produce overhead slides for a presentation. This facility was originally provided by a separate package, S_LT_EX, but that had a number of disadvantages:

- it was limited to a set of specially-scaled Computer Modern fonts and it was not easy to adapt to other fonts;
- the user was required to have two separate files, one for control information and the other for the actual slides;
- the control of colour and overlays was limited and crude;
- There was only one 'style' for slides, and writing a different layout (to, say, put a logo on each slide) was not documented.

L^AT_EX users now have a variety of fonts, and vast numbers of styles, to choose from, but S_LT_EX has lagged behind. When L^AT_EX 2_ε was released at the end of 1993, this included a simple L^AT_EX document class (already available in the New Font Selection Scheme, version 2) to emulate S_LT_EX without the overhead of a separate macro package. However, there is a much better L^AT_EX package which has been available for some time now—*seminar*; if used in conjunction with a PostScript printer, and a set of useful macros called P_STricks,⁷ this offers almost every imaginable facility, including:

- ☞ Fancy frames, headers and footers;
- ☞ Landscape and portrait slides in the same document;
- ☞ Coloured text and tables;
- ☞ Interleaving of annotations and slides;
- ☞ Slide 'chapters' and list of slides;
- ☞ Overlays.

seminar is a normal L^AT_EX package which can be used with almost all other L^AT_EX packages (such as those to change font, include graphics etc). Its main job is to produce transparencies, but it can also make accompanying notes from the same file. It is compatible with *AMS-L^AT_EX* and L^AT_EX 2_ε.

2 Using the *seminar* style

Usage is simple; begin your document in the normal way⁸ with

```
\documentclass{seminar}
```

The slide environments are

```
\begin{slide}  
...  
\end{slide}  
  
\begin{slide*}  
...  
\end{slide*}
```

Where `slide` is for landscape slides and `slide*` is for portrait slides. By default, the document is typeset in *landscape* mode, but if you include the `portrait` package option, the document is typeset in portrait mode. *Typesetting* the document in landscape mode is different from *printing* it in landscape mode; you have to worry about

⁷The *seminar* package and P_STricks are the work of Timothy van Zandt (tvz@princeton.edu).

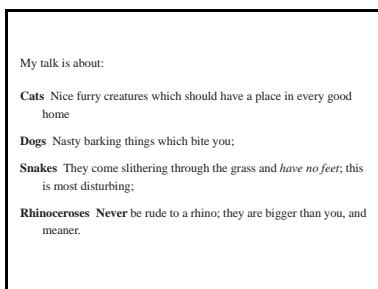
⁸We are assuming L^AT_EX 2_ε here, just to remind you to upgrade.

the orientation of the page when printing, but with `dvips` this is simple, and taken care of in the local control file described below.

So the default output⁹ from this input:

```
\documentclass{seminar}
\usepackage{times}
\begin{document}
\begin{slide}
My talk is about:
\begin{description}
\item[Cats] Nice furry creatures
  which belong in every good home;
\item[Dogs] Nasty barking things
  which bite you;
\item[Snakes] They come slithering
  through the grass and \emph{have
  no feet}; this is most disturbing;
\item[Rhinoceroses] {\bfseries Never}
  be rude to a rhino; they are bigger
  than you, and meaner.
\end{description}
\end{slide}
\end{document}
```

will look like:



Most slides will be no more complicated than this, using standard \LaTeX environments like `itemize`, `enumerate` and `tabular`.

3 Frame styles

A variety of slide framing styles are available, set with the `\framestyle` command; the following are some of the predefined ones (some assume you have a PostScript printer), using the `\slideframe` command:

On the fifth day of Christmas, my true love gave
to me:

none

1. Five overfull hboxes
2. Four fontdimens missing
3. Three nested endgroups
4. Two undefined commands
5. ... and a token in \TeX 's stomach

⁹We have added a package “`times`” so that the output will reduce properly to thumbnails for this article.

shadow

On the fifth day of Christmas, my true love gave to me:

1. Five overfull hboxes
2. Four fontdimens missing
3. Three nested endgroups
4. Two undefined commands
5. . . . and a token in T_EX's stomach

1

double

On the fifth day of Christmas, my true love gave to me:

1. Five overfull hboxes
2. Four fontdimens missing
3. Three nested endgroups
4. Two undefined commands
5. . . . and a token in T_EX's stomach

1

oval

On the fifth day of Christmas, my true love gave to me:

1. Five overfull hboxes
2. Four fontdimens missing
3. Three nested endgroups
4. Two undefined commands
5. . . . and a token in T_EX's stomach

1

Similarly, a variety of page styles (the headers and footers) are available with the `\pagestyle` command, such as:

empty

On the fifth day of Christmas, my true love gave to me:

1. Five overfull hboxes
2. Four fontdimens missing
3. Three nested endgroups
4. Two undefined commands
5. . . . and a token in T_EX's stomach

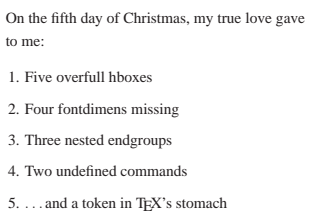
plain

On the fifth day of Christmas, my true love gave to me:

1. Five overfull hboxes
2. Four fontdimens missing
3. Three nested endgroups
4. Two undefined commands
5. . . . and a token in T_EX's stomach

1

align



On the fifth day of Christmas, my true love gave to me:

1. Five overfull hboxes
2. Four fontdimens missing
3. Three nested endgroups
4. Two undefined commands
5. ... and a token in TeX's stomach

Both slide frames and page styles can be customized; for instance, the examples in this paper (e.g. Figure 1) are suitable for use at CERN.

4 Interleaving notes, and selecting subsets

It is easy to intersperse your slides with notes to yourself; these can be simply placed between the `slide` environments or enclosed in a specific `note` environment. You can use any \LaTeX commands in these notes, and include your whole article here if desired. When you want to print the slides, a variety of package options can be used:

slidesonly Only the slides are printed;

notesonly Only the slides are printed;

notes The slides are interleaved with the notes;

article The document notes are typeset like a normal \LaTeX paper, and the slides are placed as figures (reduced to half size).

The `\slideplacement` command can be used to affect how slides are placed in the `article` format; the possible parameters are:

float (default) Slides are floated

float* Slides are floated, but if two column format is chosen they will span both columns

here Slides are placed where they occur in the notes

Further detailed control of the interaction between slides and notes is given in the *User's Manual*.

Selected slides can be included or excluded with the `\onlyslides` or `\noteslides` commands which a parameter of a comma-separated list of slides; this can be numbers, ranges (e.g. 5–10) or \LaTeX `\ref` commands referring to `\label` commands in the slides.

5 Control over slide size, fonts and magnification

There are a great number of parameters by which the user can change any of the following either on a slide-by-slide basis, or for the whole document:

- Slide height and width;
- Top, bottom, left and right margins;
- Text justification (it is ragged right by default);
- Page breaking by varying tolerance of over-running material;
- Inter-line spacing;
- Point size, and choice of fonts.

How to change the default settings is explained in detail in the *User's Guide*.

Because `seminar` works by magnifying pages, sophisticated users should read the manual to see how to deal with setting and changing TeX dimensions. Most users need not worry about this—in commands like `\epsfig` you should always express your ‘width’ and ‘height’ requests in fractions of the line size anyway.

6 Advanced use: customing the `seminar` control file

The `seminar` package always starts by trying to find a file called `seminar.con` on the `TEXINPUTS` path; this gives the user or site an opportunity to conveniently customize the defaults. The `seminar.con` file can contain any \LaTeX commands, including inputting style files. Our figures were typeset using a `seminar.con` set up for CERN; the contents of this are given below, with explanation of what is being done. It also shows how higher-level functions can be added which the average user would not want to program for themselves.

First, we set up landscape macros for the `dvips` driver.

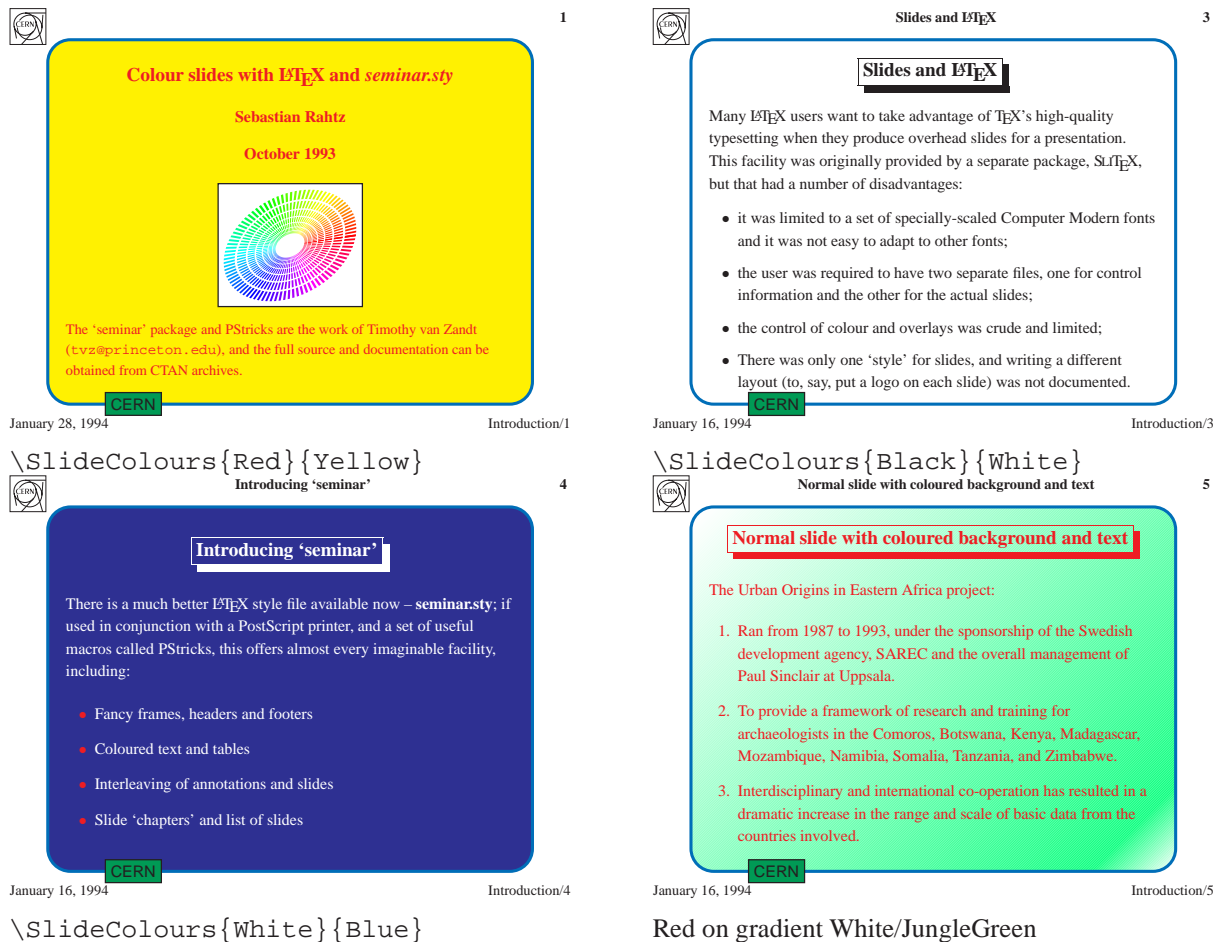


Figure 1. Colour in slide background and foreground: (simulated with grey levels)

```
\newcommand{\printlandscape}{%
  \special{papersize=297mm,210mm}}
```


We will assume PostScript printers, and gain nice PostScript effects like rounded box corners; these will need some extra style files:



```
\input semcolor.sty
\input fancybox.sty
```

For slide 'sections', list of contents, we use another style file:

```
\input slidesec.sty
```

This allows us to use various commands, some of which are used below. We can also produce list of slides, in two layouts:

<p>\listofslides</p>  <p>2</p>	<p>\Slidecontents</p> <p>Frame styles</p> <p>7</p>
--	---

<p>List of Slides</p> <p>Introduction</p> <p>3 Slides and L^AT_EX</p> <p>4 Introducing 'seminar'</p> <p>5 Normal slide with coloured background and text</p> <p>Frame styles</p> <p>6 Frame styles</p> <p>Text colours and colour tables</p> <p>8 Text colors</p> <p>9 Colour tests</p> <p>10 A multi-page coloured table</p> <p>11 Z schemas built up with overlays</p> 	<p>Introduction</p> <p>✓ • Slides and L^AT_EX 3</p> <p>✓ • Introducing 'seminar' 4</p> <p>✓ • Normal slide with coloured background and text 5</p> <p>Frame styles</p> <p>✓ • Frame styles 6</p> <p>Text colours and colour tables</p> <p>⇒ • Text colors 8</p> <p>• Colour tests 9</p> <p>• A multi-page coloured table 10</p> <p>• Z schemas built up with overlays 11</p> 
--	---

January 16, 1994	Introduction/2	January 16, 1994	Frame styles/2
------------------	----------------	------------------	----------------

For slide headings, there is a predefined `\slideheading` command; we will amend this so that it is typeset with a 'shadow'. The `slidechapter` command is also defined (the code is not given here) which allows the user to break the slides into groups; the slide chapter title will be given in the bottom right corner with this CERN style.

```

\def\@empty{}
\renewcommand{\makeslideheading}[1]{%
  \gdef\theslideheading{#1}%
  \def\@tempa{#1}%
  \ifx\@tempa\@empty\else
    \begin{Sbox}
      \begin{Bcenter}
        \large\bfseries#1
      \end{Bcenter}
    \end{Sbox}
    \centerline{\shadowbox{\TheSbox}}
    \vspace{1ex minus 1ex}
  \fi
}

```

Now the CERN page and frame styles; the plain 'cern' style just places registration '+' marks, and the date:

```

\newpagestyle{cern}%
  {{\color{Black}\small
    {\bfseries +} \hfil \today
    \hfil {\bfseries+}}}%
  {{\color{Black}\small
    {\bfseries +} \hfil \thepage
    \hfil {\bfseries+}}}%

```

Whereas the 'cernsections' style has section headings and a logo:

```

\newpagestyle{cernsections}%
  {{\color{Black}\small
    \raisebox{-.5cm}[0cm][0cm]{%
      \epsfig{figure=cernlogo.ps,height=.8cm}}
    \hfil {\bfseries\theslideheading} \hfil
      {\bfseries\thepage} }}%
  {{\small\color{Black}\today
    \hfil \thechapterheading /\inchap }}%

```

For the slide frames, we define a frame with the word "CERN" set in a coloured box on the lower left; this is done

using the PSTricks macros (which are automatically included by the ‘semcolor’ option above). The colour commands are those predefined in dvips’s `color.pro` header file:

```
\newslideframe{cern}{\SlideFront
\boxput(-0.7,-1.11){\psframebox%
[linecolor=black,fillcolor=ForestGreen,
fillstyle=solid]{\hbox{\normalsize
\sffamily\color{Black}CERN}}}{#1}%
\color{Black}}
```

Finally we make sure that each slide starts with the current foreground colour.

```
\def\everyslide{\SlideFront}
\def\theslideheading{}
```

The user uses the command `\SlideColours`, with two parameters, which are colour names for foreground and background. A synonym is defined for black on white. We have to be a bit careful defining the frame border, because by default it is coloured using the POSTSCRIPT ‘setgray’ operator, and that might not work with the colour separation, so we define an explicit blue frame (for variety).

```
\newslideframe{blueframe}[%
\psset{linecolor=NavyBlue,%
linewidth=\slideframewidth,%
framesep=\slideframesep,%
cornersize=absolute,%
linearc=.5cm%
}]{\psframebox{#1}}
\def\SlideColours#1#2{%
\gdef\SlideFront{\color{#1}}%
\slideframe{\Framedefault}%
\slideframe*[\psset{fillcolor=#2,%
fillstyle=solid}]{blueframe}%
}
\def\SlideColours#1#2{%
\gdef\SlideFront{\color{#1}}%
\slideframe{\Framedefault}%
\slideframe*%
[\psset{linecolor=Black,%
fillcolor=#2,fillstyle=solid}]%
{splain}%
}
\def\blackandwhite{\SlideColours{Black}{White}}
```

The slide defaults will be for a detailed layout and CERN logo, with yellow writing on a blue background:

```
\pagestyle{cernsections}
\slideframe{cern}
\def\Framedefault{cern}
\SlideColours{Yellow}{RoyalBlue}
```

VII Back(s)lash

Jonathan Fine

J.Fine@uk.ac.cam.pmms

Welcome to the first of a series of columns devoted to the subtleties of programming \TeX . The focus will be on the primitive commands and low-level features of \TeX . This column is devoted to `\csname` and to avoiding its side effects.

The reader might do well to begin by turning to [40]. (This means page 40 of *The \TeX book*). Exercise 7.7 on that page, and its solution, describe a `\ifundefined` macro. This macro has three shortcomings. The first is that

```
\ifundefined{relax}
```

will come out to be true! The second is that

```
\ifundefined{xyz}
```

will define `\xyz` to be `\relax`, if `\xyz` is not defined. This is often not what is wanted. The third problem is that the process of defining `\xyz` will, if done within a group, add an item to the save stack [301]. This can cause problems in processing \LaTeX documents which have a lot of cross-references.

To see this, we will use introduce a macro

```
\def\typeshow #1%
{%
  \immediate\write 16
  {> \string #1 = \meaning #1.}%
}%
```

which will log the meaning of a token for us.

Here are two extracts from a `.log` file, which record its interactive use.

```
*{\expandafter\typeshow\csname xyz\endcsname}
> \xyz = \relax.
```

Notice the braces to confine the redefinition of `\xyz` to a group. This next example shows that `\xyz` really is

```
*\typeshow\xyz
> \xyz = undefined.
```

undefined.

When `\csname` is performed within a group, any (local) assignment it might perform will be restored when the group ends. If we can end the group *before* the `\typeshow` is called, then we will get the original, undefined, meaning. Here is how to do it.

```
\begingroup
  \expandafter
\endgroup
\expandafter
\typeshow
\csname xyz\endcsname
```

The group will begin. The `\expandafters` will have `\csname` brought into operation. When the `\endcsname` is reached, execution passes to the token after the first `\expandafter`, which is the `\endgroup`. This restores the value of `\xyz`. Now `\typeshow` gets the token `\xyz`, *with the meaning of* `undefined`.

Here is a lightly edited version of the log file for the above code, entered interactively, with elaborations on the previous comments interspersed.

```
*\begingroup
{\begingroup}
```

Here the first line is entered, and acted upon.

```
* \expandafter
{\expandafter}
```

reprinted from *Baskerville*

Volume 4, Number 1

The primitive (and expandable) command `\expandafter` is now being processed. It looks for the next token,

```
*\endgroup
```

here it is, and then expands the one after, which is

```
*\expandafter
{\expandafter}
```

which will again cause for a token to be read

```
*\typeshow
```

here it is, and the one after is

```
*\csname xyz\endcsname
{\csname}
```

which is now executed. It will form a control sequence `\xyz`. Execution now passes to the token after the first `\expandafter`, which is:

```
{\endgroup}
{restoring \xyz=undefined}
```

and *this has the desired effect of restoring the original value*. The rest proceeds as before

```
\typeshow #1->\immediate \write 16
  {> \string #1 = \meaning #1.}
#1<-\xyz
> \xyz = undefined.
```

which is what we want.

Exercise 1. Obtain the same effect, but by using `\aftergroup` to export the token `\xyz` out of the group. Discuss the relative merits of the two methods.

Exercise 2. Write a macro which tests as to whether (the meaning of) a token is expandable. Consult [212–215].

VIII Topical tip: Numbering theorems and corollaries in L^AT_EX

R. A. Bailey

Goldsmiths' College, University of London

Question 1 We Mathematicians can't use L^AT_EX. We need to be able to choose how to label our theorems. For example, I like to have my important theorems numbered in a sequence Theorem A, Theorem B and so on, and the less important theorems numbered Theorem 1, Theorem 2 and so on. You can't do that in L^AT_EX.

Answer Oh yes you can, and using nothing more than you can find in *L^AT_EX: A Document Preparation System* by Leslie Lamport, hereafter called *The Manual*.

Pages 58–59 of *The Manual* show how to set up a simple theorem environment. The command

```
\newtheorem{thm}{Theorem}
```

creates an environment called `thm`. Then each use of this environment produces something whose heading is **Theorem**. It is true that these theorems are numbered 1, 2, 3, etc. To obtain something numbered A, B, C, etc., use the numbering commands given on page 92 of *The Manual*. Thus

```
\newtheorem{main}{Theorem}
\renewcommand{\themain}{\Alph{main}}
```

creates an environment called `main` whose heading is also **Theorem** but whose instances are numbered A, B, ... Cross-references work correctly too: if you label the third `main` with `\label{mmm}` and refer to it with `Theorem-\ref{mmm}` then it will be called Theorem C.

Question 2 Journal editors are so fussy. They all want me to number my corollaries in different ways. The first wants corollaries numbered in the same sequence as theorems; the second wants them numbered in a separate sequence of their own; the third wants the corollaries after Theorem 7 to be numbered Corollary 7.1, Corollary 7.2, etc.; while the fourth also wants the corollaries to start renumbering after each theorem, but wants the corollaries after Theorem 7 to be numbered Corollary 1, Corollary 2 etc. How do I do all of this?

Answer It is not hard to do these things, because L^AT_EX is provided with the `newtheorem` command. I shall assume that you have defined an environment `thm` as in the answer to Question 1. The instructions on pages 58–59 of *The Manual* show us how to satisfy the first three editors. For the first, put

```
\newtheorem{cor}[thm]{Corollary}
```

and you will get an environment called `cor` whose instances are called **Corollary** numbered in the same sequence as the theorems. For the second, put

```
\newtheorem{cor}{Corollary}
```

and for the third put

```
\newtheorem{cor}{Corollary}[thm]
```

For the fourth editor, we need the extra information from page 92. The third command above makes the `cor` counter start again after each `thm`, but it causes the Corollary number to be printed as, say, 7.1 rather than 1. We can cure this by putting

```
\newtheorem{cor}{Corollary}[thm]
\renewcommand{\thecor}{\arabic{cor}}
```

In each of the four cases you get an environment called `cor` whose instances are called **Corollary**. Only the system of numbering is different.

You should now be able to work out how to make the corollaries after Theorem 5 come out as Corollary 5a, Corollary 5b, and so on.

IX Malcolm's Gleanings

Malcolm Clark

cudax@uk.ac.warwick.csv

0.1 Book review

Computers and Typography 'compiled by' Rosemary Sassoon, Intellect, Oxford, 1993, 208pp, ISBN 1-871516-23-4.

On the title page of this book, the compiler notes that the customary words "edited by" were omitted at her request. She goes on to say that the book is "an example of what this is all in aid of—typographic excellence in the computer age". A bold claim, and an interesting inference that typographic excellence is not the customary bedfellow of computer 'mediated' books. As if to underline the typographic excellence, the title page faces a reproduction of a page from Aldus Manutius' *Hypnerotomachia Poliphili* of 1499.

Before looking at whether these claims are justified, what of the content? For whom is the book intended? The cover suggests that it is invaluable for "all concerned with teaching, design or who produce documents of all kinds". In the preface, Sassoon suggests that the purpose is to bridge the gap between the computer people and the typography people, but mainly to raise the awareness of letterforms and layout, rather than to educate those in the typographic world to the appropriate use of computers. It seeks therefore to educate the computer user to a higher level of understanding of 'typography', however widely defined.

The book is organised into five parts: each part contains two or three contributions. Part 1 covers 'Spacing and layout': the contribution by Gunnlaugur Se Briem, *Introduction to text massage*, illustrates one recurrent difficulty in the book—typographers and designers tend to be aware only of the desk top publishing end of computer-assisted typography. His recommendation to search and replace the ligatured letters is a shade risible, though on the whole his advice is sound. But how practical is it to look at each line ending to check hyphenation, lift the baseline to adjust parentheses (sometimes), fiddle with the leading, and so on. Should we not be looking for better models of line and page make up which recognise the potential problems and solve them for us? Did every jobbing printer take this much care? James Hartley (*The layout of computer-based text*) examines some aspects of layout, starting with a questionnaire, and going on to more general matters of the distribution of space, and how it can be used to enhance content. It is indeed true that the use of white space is poorly appreciated by many: that increased use of white space might make something more useful (and less wasteful) is not a concept readily grasped, until some useful and pertinent examples like these are thrust under people's noses. Richard Southall's *Presentation rules and rules of composition in the formatting of complex text* is a highly literate explanation which draws together the views of 'traditional' typographers from Moxon, Fournier, Brun, De Vinne and Tschichold in order to show how their 'rules' may, or may not be applied in computer based composition systems. Southall's in-depth knowledge of the working of \TeX and \LaTeX gives him a unique position, and he develops some rather telling criticisms. His remarks are more generally applicable and help to provide a useful set of criteria for the assessment of computer based systems.

In Part 2, *Typographic choices—Latin and other alphabets*, Ari Davidow examines some of the problems facing the typesetting of Hebrew (*Digital hebrew in a monolingual world*). This is an anecdotal discussion, with a few interesting points. Its description of computer software (almost all Macintosh based) is a snapshot already out of date. He is concerned solely with *wysiwyg* type input. The observation that italic or slanted letter forms in Hebrew are seldom satisfactory is worth hearing, although perhaps diminished slightly by the illustration which was inserted upside down. Elwyn and Michael Blacker, *Spoiled for choice*, have little to say about other alphabets, but something to say about computer typography, and, more important, about some of the typographic choices that were made in creating this book. At least they believe that fine typography is attainable with computer technology (albeit "in the hands of a skilled designer with mastery of the optical considerations", faint encouragement for the \LaTeX enthusiast). And then they mention some of the design considerations and problems they faced with the book. They also comment on their use of Bembo, with additional characters chosen from the expert font. The use of the expert Bembo font is perplexing. Although chosen in part because it has small capitals, these seem very thin and weedy to me, as if they have been simply optically scaled from the 'normal' capital. Examination of the book suggests that this, and their "detailed checking of a proof" may have been in the realm of good intentions rather than solid achievement.

Part 3, *More technical issues involved in type design* contains two papers. The first is *Some aspects of the effects of reprinted from Baskerville*

Volume 4, Number 1

technology on type design by Mike Daines, which concentrates on the advantages which Peter Karow's Ikarus system has had on digital type. He also brings in many of the other potential tools available, especially those for the Macintosh. Another useful and considered paper by Richard Southall, *Character description techniques in type manufacture*, looks at two traditional (i.e. non-electronic) methods of the production of type, and two digital techniques. The objective here is to identify the strengths and weaknesses of the changing technologies, and the areas in which they are most (or least appropriate). From his description of the processes involved, Southall develops a 'systematic view' of the manufacture of type. This has the merit of providing a plausible model which we can use, and may give the basis for some qualitative comparisons. Actually, by the end of this paper I am left surprised that any acceptable typefaces were ever produced in any technology, given the inherent problems at each stage.

The penultimate section, *Lessons to be learned from the history of typography* includes what I found to be one of the most demanding papers, balanced by one which I found agreeably optimistic. Fernand Baudin's *Education in the making and shaping of written words* is a polemic, and although it traces an argument going back to the days of Villon, and emphasises the importance of handwriting (along the way consigning Marshall McLuhan to one of the outer hells), I was left unclear how the final conclusion was derived from the route and its many byways. But one useful point which is reiterated is that the study of type must not be to the exclusion of the study of space. A consensus is appearing. Alan Marshall's contribution, *A typographer by any other name* came as a welcome relief after this fundamentalism. He puts many of the problems in perspective, and provides a thankfully optimistic conclusion, which seems both balanced and realistic. He appreciates that all major technological changes have their problems, that they start with a period of emulation, and then innovation—there are repeated examples in the printing industry. His observation that Orwell had argued that the advent of Penguin's paperbacks all but signalled the end of civilization as we know it helps place in perspective similar contemporary claims of an apocalyptic nature. Perhaps most telling, he suggests that the pool of typographic knowledge is not limited, but is expanding, encouraged by the technologies becoming available.

The last section, *Research and the perception of type*, I found difficult to integrate with the stated objectives of the text. Rosemary Sassoon's own contribution, *Through the eyes of a child—perception and type design*, is an account of designing a typeface which would aid children learning to read. Some of her observations on legibility are interesting and intriguing, but they are hard to relate to computer in general, or the more specific needs of computer aided publishing for a wide market. For educationalists and teachers there is probably much here. Perhaps not surprisingly, she also makes a plea for handwriting. The final paper is daunting. Roger Watt, in *The visual analysis of pages of text*, describes some experiments the visual perception of printed pages. He analyses the same text with different inter word and inter line spacing. The technique of analysis is claimed to have some reasonable closeness to the way in which the human visual system works. In this analysis he identifies a number of different perceived 'structures', which he then relates to the specifics of the text, like sentence breaks, rivers, words, inter line space and so on. Perhaps contrary to received wisdom, he suggests that rivers may be useful, as landmarks for navigation in a text. The result is the conclusion that it should be possible to specify the 'riveriness' and 'wordiness' desired (the visual effect), and then find the appropriate word and line spacing. This seems a little radical, and the views of some typographers on this could be interesting. Clearly it is appropriate to attempt to bring in a more physiological appreciation of how type is understood, rather than the typographers' often hand waving generalisations, but this is not a straightforward paper. It is not clear how far the conclusions may be generalised, either to english texts in general, or to texts in other languages, where word length, and the distribution of ascenders and descenders may be quite different. How it would generalise to non-Latin texts is another mystery, or, as academics say "more work needs to be done".

There is the feeling that some contributors view the changes as a shock to the system, whilst others know it has all happened before, and that while some things will deteriorate, new possibilities will arise, and things will become possible about which we have not yet dreamed. The curious appeals to handwriting as the basis of success have a very luddite ring to my ears.

One of the factors which worried me about the book was the extent to which it achieved its aim as a "model of good typographic practices". Frankly, it lacks consistency, and there are far too many typos. Perhaps the erratic application of a house style is one thing, but mistakes are something more serious. These blemishes and inconsistencies highlight a notable omission from the book: discussion on the real difference between markup systems and those which demand that the text be dealt with interactively—i.e. a *wysiwyg* system. Many of the small problems of style can be more easily resolved through markup systems. If the goal is to produce something which is even, markup can ensure that the rules are carried out remorselessly each time, while the use of more 'flexible' systems actually requires much more thought and discipline right through the book production.

Unless this volume had been presented as some model for the typographically unkempt, it would not be appropriate to pick up on the small faults, but sadly, it does seem to fall into the same pit in which it sees others. On the other hand,

the overall design of the book is pleasing. Even the very ragged right works quite well (especially when hyphenation is all but suppressed), and the wide central margins are used quite intelligently as a location for captions to figures. It is obvious that the book was designed ‘spread by spread’, allowing for what the reader actually sees. The interplay of white space is attractive.

It is not a book for novices; nor is it a book for power users. It falls awkwardly between a number of stools. Taken individually, the papers are interesting, stimulating, and often provocative. But taken as a whole I just cannot discern the linking thread, or the theme which binds it into more than a book of loosely connected essays. It veers from the general, or at least broad, to the very specific, from which something more can be inferred. Placing these side by side gives a very uneven intellectual feel to the whole thing. It feels as if Sassoon asked some of her friends to contribute something to a book on typography and computers, without specifying the aim too tightly, and lo! we have the results in our hands. The central concern of the book still worries me. Sassoon says that she hopes people will “never again be satisfied with second best”. Elsewhere in the book are appeals to “fine typography”. I would have preferred to see an appeal to “fitness for purpose”.

This review is based on one which appears in the *Information design journal*, vol 7, no 2, 1993, p161–6

0.2 *Information design journal*

One of the curses of the (L)T_EX world is that many proponents become infected with a thirst for matters typographic. It’s an odd affliction, since many of the victims have a scientific/technical background, and the way education seems to be set up in many countries is based on the belief that science and technology are antithetical to anything aesthetic. And typography is largely an aesthetic medium—or is presented as such. How do we acquire knowledge and satisfy the hunger of our desire? There are a few books around (in my view one of the best is Ruari McLean’s *Typography*), but precious few journals. A few designerly magazines exist (I like XYZ) but they do tend to be a little elitist and introspective. What is there for those of us accustomed to reading ‘academic’ journals. I’ve yet to see a copy of *Visible Language*, although Knuth has published there from time to time. I’ve at last found something interesting, appropriate and local – *Information design journal*. It’s not really just typography, but there is much in it which is typographical.

The *call for papers* describes the readership as multidisciplinary and that contributions are welcomed on a range of topics related to the communication of information of social, technical and educational significance. Looking over the last four issues, I note an interest in forms design (both questionnaires and bills: this is also one of my interests—it fascinates me that it is so difficult to design satisfactory forms), in information signing (like directions, maps), in information symbols (like those ISO symbols for almost anything, most of which I find odd and misleading – this is quite distressing for icon based computer systems. . .). There also seems to be a wish to test comparisons—in other words, to test hypotheses rather than make hand waving generalisations. But there are other articles which aim to convince by qualitative argument.

The range of papers in each issue is broad too; not just in content, but also in style. In a sense each issue becomes more informal as you read through it. The key articles are refereed, as one would hope, but there are reviews of one sort or another. Somehow it achieves a pleasant balance between rigour and informality. I therefore commend it to you as a useful journal to read and browse through. For more information, contact Fred Eade, Idj subscriptions, PO Box 1978, Gerrards Cross, Bucks, SL9 9BT.

1 Nonsense

The major event in the T_EX world over the last few weeks (nay, months) must be the test release of L^AT_EX 2_ε. To the surprise of many, this arrived in December, just in time to disrupt family Christmases throughout the world. Good timing. Since it was truly a test release, it did not have all the bits that we have been led to expect in the *Companion*. In passing, printed and bound copies of the *B_TE_X Companion* are stated to exist. Frank Mittelbach says he has one (but then, he would. . .). I wouldn’t have thought he needed one, unlike the rest of us. It seems to have been relatively painless to install, from the messages which flitted around, although running it gives you even more file name extensions to contend with—just when you thought you had come to grips with the profligacy of L^AT_EX in creating extra files for itself!

It’s a relief to see something substantive like this out for use. If there are worries though, it must be whether this will distract attention from the serious matter in hand—L^AT_EX 3. On the other hand, it will soften us up a little, first by accustoming us to regular upgrades/updates (just like Word for Windows!), but more importantly ensuring that the communications channels work consistently. To a large extent this is going to be software distributed and supported electronically. One of the features I like is that queries will not be entertained if you are using an ‘obsolete’ version of L^AT_EX 2_ε.

I'm becoming confused how I should write \LaTeX ! Just the logo—mostly I can handle \LaTeX itself. If I look through TTN and *TUGboat*, I can find quite a few instances where the preferred form is given as $L^A\TeX$, or even $L^A\TeX$ —this latter form is especially prevalent when you see it written as $(L^A)\TeX$. Maybe consistency will return when the results of the A-in- \LaTeX competition are announced.

Is the NTS project poised to take over the world? News from the NTS project is always to be treasured, since it has all the hallmarks of an inner cabal composed of a secret elite: Phil Taylor's article in *TUGboat* revealed that besides trying singlehandedly to resurrect the economies of eastern europe, it is proposing to start to issue a 'canonical \TeX kit' (you can always tell when Phil is involved: 'canonical' sprouts everywhere!). This has the laudable aim to identify what a standard ('canonical') implementation should contain, and to liaise with developers and implementors to ensure that this is distributed with each \TeX implementation. Praiseworthy and necessary as this step is, I'm not myself clear how this relates to the desire to develop a new typesetting system. In the same issue of *TUGboat*, Nelson Beebe encourages vendors to include his `bibclean` utilities with each distribution. Will this be part of the NTS canon too?

Of course there is more. The simple existence of a piece of software does not mean that it has all the same attributes when run on different platforms. I am minded of `Makeindex`, which exists in some different incarnations with differing capabilities in terms of size of index it can handle. Since the aim of the canon is to ease the transfer of documents from site to site, the support software must be capable of handling the same sizes of problems too. Will the project be taking on this role of guardian of compatibility?

I suspect that underlying this is another agenda altogether. Identify the project to implementors and developers as the (self-selected) body in the \TeX world which somehow authorises the suitability of \TeX -related applications. In this way it makes itself the legitimate heir to Knuth as far as this sort of software development is concerned. It's a strategy that might work.

You may wonder how it leaves the user groups who are already starting to produce this sort of ' \TeX kit'. I do.

TUGboat readers will have noted that the journal is pretty well on schedule. My December issue arrived at the beginning of the year. For many people this is a welcome sign. There was a time when we felt lucky to get *TUGboat* within about 6 months of its hypothetical publication date (even then, better than EP-odd!). There has been a price to pay. Frequency is still a little problematic (two issues this year came out very close together, but you could just say that one was late and its successor on time), but more significant, one issue, the conference proceedings, is virtually half of the total mass—in other words, three 'normal' issues constitute about the same amount of verbal as the conference. Last year ran to about 450 pages: in 1989, it was over 750. Even arguing that TTN is removing some 'mass', then the volume is still slimmer. We could also argue that the multiplicity of 'competing' journals has taken some articles away (but a cursory glance of the Dutch group's MAPS will demonstrate that much is just recycled between journals). Is there a worrying trend in motion: thin and timely?

X Letters to the editor

1 A T_EX front-end in *NextStep*

Like most readers of *Baskerville*, I greatly enjoyed S. Rahtz's survey of T_EX front-ends in the December 1993 number. I noticed that he had not mentioned one very interesting and useful such system—Tom Rokicki's implementations of T_EX for the *NextStep* operating system—and I am writing to briefly discuss its most interesting features. *NextStep* is a superior graphical user interface that sits on top of BSD4.3 Unix. Now that NeXT has ceased the manufacture of their trademark black Motorola hardware, the Intel-486 implementation of *NextStep* is their flagship product.

Although you can invoke 'NeXTT_EX' via the usual command line, the value of this T_EX lies in the integrated environment, called *T_EXView*, to which it belongs. Begin by preparing a usual source file with a `tex` extension, and double-click to begin T_EXing. *T_EXView* automatically begins its preview as soon as the first page is ready. That is, while T_EX is still typesetting the remainder of the document, page 1 is already there for your perusal. Simple mouse click commands zoom this display and drag and scroll the preview image in the preview window. `dvips` is a part of *T_EXView*, and so it is possible to include color in your typesetting, and very easily, too. *NextStep* is built around Display PostScript, so *T_EXView* readily offers all PostScript fonts in the document for onscreen display, and color if you use a color monitor. Of course, included `epsf` files are also displayed. (Color is rendered in an appropriate shade of gray on the typical PostScript b&w printer.) There is an option for 'printing' to fax.

I can't resist the temptation to note two extensions Tom has added to T_EX proper. If the first line of a source file is `~&foo`, then the command

```
tex myfile
```

invokes the format file `&foo`; that is, it is equivalent to the command `tex &foo myfile`. Output stream 18 will pipe commands to Unix. It's possible to sort and input an index file (say) in one fell swoop via commands like:

```
\immediate\write18{mysort <index.raw >index.sort}
\input index.sort
```

in your source file.

Alan Hoenig

17 Bay Avenue, Huntington, NY 11743 USA

`ajhjj@cunyvm.cuny.edu`

2 Command line T_EX for ever

I notice from at least two articles in the current issue of *Baskerville* (Vol. 3, No. 2) that there appears to be some *zeitgeist* within which the long-established, traditional and highly logical method of using T_EX and its adjuncts is brought into question; I refer, of course, to the articles by R. Allan Reese (p. 3, col. 1, para. -2), and by your esteemed self (p. 4, col. 1, para. -5).

In particular, I wish to take issue with your assertion that: "Every T_EX user knows that the traditional command-line way of working (the 'edit; compile; {preview, print}' cycle) is far from ideal." This assertion, Sir, is blatantly and demonstrably flawed. There exists at least one T_EX user (and, I suggest Sir, many many more) who is *completely* satisfied with this way of working, and who regards any and every attempt to protect the intellectually-challenged from the realities of *real* computing by encapsulating trivial tasks in a so-called 'development environment' as a fruitless and totally misguided activity.

I remain, Sir, your most humble and obedient servant:

Philip Taylor.

P.S. I see that our esteemed sometime Chairman, Malcolm Clark, now has a *doppelganger* who is also contributing to the columns of *Baskerville*; who is this pretender to the throne who dares assert "We've been nice guys for too long."?

reprinted from *Baskerville*

Volume 4, Number 1

3 Love L^AT_EX nods

I see from the current issue of *Baskerville* (Vol. 3, No. 2) that the dotfill leaders for the table of contents no longer align; is this yet another demonstration of the inferiority of L^AT_EX when compared to the Real Thing?

I remain, Sir, Yours etc.,
Philip Taylor

XI UKTUG Business Reports

1 Membership of UK TeX Users Group (1994)

This issue of *Baskerville* is being posted to 1993 members. Below I present the details of the 1992/3 Income and Expenditure and Balance sheet. The cost of distributing each issue of *Baskerville* is quite considerable and the group's funds, whilst reasonably healthy, cannot stand avoidable expenditure. Please renew your membership as soon as possible. I am grateful to those who have already paid, as it helps the committee to plan its expenditure for the year. 1994 memberships have been acknowledged by email (or paper mail if no email is available). Please contact me if you have renewed and **not** received an acknowledgement.

1.1 Membership Data

	1993	1994
UKTUG	31	11
TUG	11	
TUG and UKTUG	136	47
TUG and UKTUG (student)	3	3

(as at 7th January 1994)

2 UKTUG accounts 1 October 1992 to 19 August 1993

2.1 Statement of Income and Expenditure

INCOME

Membership (see separate table)		£6,248.80	
LaTeX3.0 contributions		£402.50	
UK Book sales		£43.30	
Sale of mailing labels		£45.00	
Income from meetings			
October 1992	£230.00		
January 1993	£982.50		
April 1993	£1,878.89		
Subtotal	<u>£3,091.39</u>	£3,091.79	
TUG'93 Conference fees	£35,640.79	£35,640.79	
Total income		<u>£45,472.09</u>	

EXPENDITURE

Postage, copying, stationery	£400.92		
Committee Expenses	£429.20		
TeX and TUG News Printing	£4,053.06		
Books	£59.88		
Meeting costs: EPS	£191.24		
October 1992	£304.60		
January 1993	£509.23		
April 1993	£3,400.94		
Bank charges	£30.00		
Bounced cheque	£10.00		
LaTeX3.0 fund	£263.00		
TUG'93 conference	<u>£28,197.58</u>		
Subtotal	£37,849.65		
Total expenditure		<u>£37,849.65</u>	
SURPLUS		<u>£7,622.44</u>	

2.2 Balance sheet

CURRENT ASSETS

Debtors: TTN	£4,053.06		
TUG(LaTeX)hax)	£3,338.90		
Cash in hand	£0.00		
Cash in bank	<u>£13,494.35</u>		
Total assets	£20,886.31	£20,886.31	

CURRENT LIABILITIES

Creditors: TUG'93	(£7,443.21)		
TUG memb. fees	(£4,549.00)		
Donation to TUG'93 travel fund	(£500.00)		
LaTeX3.0 fund	<u>(£389.50)</u>		
Total liabilities	(£12,881.71)	(£12,881.71)	

BALANCE		<u>£8,004.60</u>	
---------	--	------------------	--

2.3 Position with regard to opening balance

OPENING BALANCE	£5,872.22		
SURPLUS	<u>£7,622.44</u>		
CLOSING BALANCE	£13,494.66		