Geometric Diversions with TEX, METAFONT and METAPOST

Karel Horák
Mathematical Institute of the Academy of Sciences
Žitná 25, 115 67 Praha 1,
Czech Republic
horakk@math.cas.cz
http://www.math.cas.cz

Abstract

Recently METAPOST has been preferred to METAFONT, although this need not be the general case. Each can be used, with its own specific advantages, and readable source text can be written for both great compilers! We provide some advice on how to do it in such a way that practically no change in incorporating corresponding figures into a document is necessary. This includes typesetting labels using Alan Hoenig's labtex macros, which can be done independently even if graphics produced by METAPOST instead of METAFONT are then used in the document. Some examples of geometric diversions are given.

Résumé

Ce dernier temps METAPOST est de plus en plus utilisé, vis-à-vis de METAFONT, ce qui n'est pas indispensable. En effet, on peut utiliser chacun des deux dans des cas spécifiques et profiter de leurs avantages respectifs ; d'autre part, on peut écrire du code lisible par les deux compilateurs ! Nous proposons un certain nombre de conseils pour ce faire, de manière à ne pas changer la manière dont une figure est insérée dans un document. Cela comprend également la composition de légendes aux figures en utilisant les macros labtex d'Alan Hoenig, ce qui peut être fait indépendamment du fait si les graphiques ont été produits par METAPOST au lieu de METAFONT. Nous proposons également un certain nombre de divertissements géométriques.

Introduction

There are many ways to assign labels to selected points of a picture. One easy example is given in the *TEXbook* (p. 389), which introduces the macro `\point`. Using variable `\unit` (or possibly `\xunit` and `\yunit`, why not?) one can label any inserted graphics and change the position of its labels dynamically, should the picture be rescaled, by changing `\unit`. But if we are concerned with pictures prepared by METAFONT or METAPOST, there are more convenient possibilities as well.

A better way is to use communication between METAFONT and TEX via a *tfm* file. There are two basic possibilities on how to store coordinates of selected points: via fontdimens or via kerning. Both of these ideas belong, as far as I know, to Alan Hoenig (see [2,3]).

The first way has one important drawback: the number of fontdimens is very limited. The other way (`labtex.mf` and `labtex.tex` macros) gives the user much more freedom to store coordinates. This very short macro file uses for that task kerning with the first three characters (it can be easily modified to use more if necessary, as METAFONT's limit on the number of kerning pairs is now more than 32,000).

Inspired by [3], Alan Jeffrey proposed in [5] another way to manipulate labels of METAFONT diagrams. Because of deficiencies in the file-handling capabilities of METAFONT he used the *log* file for communicating between METAFONT and TEX, extracting the desired information with the *grep* utility.

Clearly, METAPOST can solve all these problems easily and smoothly, as it has own labelling mechanism which uses TEX for typesetting labels, which are then converted into low-level METAPOST commands. Does this mean that we should rewrite all our old programs written for METAFONT, as we did when we changed from the old METAFONT 78 to the new one? Not at all! John Hobby, author of METAPOST, has arranged for all METAFONT sources (with some exceptions) to compile with METAPOST using a special format file `mfplain.mp`.

While there are many good reasons to prefer METAPOST to METAFONT (use of colour, no problems with higher resolution, etc.), I am still preparing many black and white documents with geometric pictures printed in normal resolution and I am accustomed to use both: METAFONT, when proofing, and only then (if necessary) METAPOST for the final compilation. For such a reason I rather often prefer to use labtex (after ten years of

excellent experience) for labels as it works in both cases without any substantial change. The aim of this short contribution is to stress that labtex can also be used with METAPOST without any serious problem.

It could be done directly: METAPOST stores the metric information into a tfm file exactly as METAFONT does; the only drawback is then the great number of trivial files generated. So it seems better to produce pictures by METAPOST *without* labtex, while the appropriate metric file with related kerning information given by labtex's pointing commands will be regenerated by METAFONT without any garbage later. As we want to use the same source file twice, we presuppose using `mfplain.mp` as the format. It can be done by the following fork (testing if colour *red* is known):

```
mode_setup;
u#:=1mm#;
define_pixels(u);
if known red:
    % labtex information is not used
def labtex (text t)= enddef;
warningcheck:=0;
else:
input labtex
fi
```

A similar fork can be used for any case where METAFONT and METAPOST behave differently (culling of some region could be alternatively changed to clipping, etc.).

The pros and the cons

One big advantage was mentioned before: one can easily change old sources from bitmap to outline without too much work.

For me, another advantage of this attempt is that I can easily preview typeset pictures (without possible colours) with the emTEX previewer, while previewing PostScript with Ghostscript is substantially slower (even if much faster now than some years ago).

Another advantage of this concept is that one can easily combine METAPOST output with other *eps* pictures, a problem that was solved only by some special tricks thanks to Bogusław Jackowski and his colleagues.

One possible disadvantage (mentioned in [5]) is that this concept does not involve communication from TEX to METAFONT (e.g. size of the label to leave some void space in colored region). Such a feature could of course be added using ideas from [2], if one wished to typeset many more such labels.

One of the advantages of METAPOST's own labelling mechanism is that one can easily transform the labels (rotate, scale, skew...), use color, erase appropriate surrounding space of label, etc. These of course could be done with labtex using TEX `\special` commands and PostScript macros (PSTricks can do all of this).

Using labtex with METAPOST

What is the main difference between a character generated by METAFONT and the corresponding PostScript picture produced by METAPOST(&mfplain)? Their dimensions are known to TEX! Dimensions (width, height, and depth of each character are given by parameters of `beginchar` in the METAFONT source file and one must take some care to guess these dimensions as best as possible, or even better to construct the picture depending on them (i.e. using variables w, h, d for defining basic points of any figure). On the other hand, dimensions in eps files output by METAPOST are read by TEX from the corresponding bounding box, which is computed by METAPOST according to the real dimensions of the resulting picture. For alternative use in TEX it is in most cases sufficient to correctly position the reference point of the picture. So we must first change the lower left corner of the bounding box to be at the origin.

The labtex macros are designed to typeset the picture first, then labels. This is even more important for the PostScript variant of the `\fig` macro, as PostScript draws the picture sequentially layer after layer, so labels typeset in advance could disappear completely under some coloured part of the picture.

The following adaptation of Hoenig's `\fig` macro can do the right job, changing the reference point of the resulting picture.

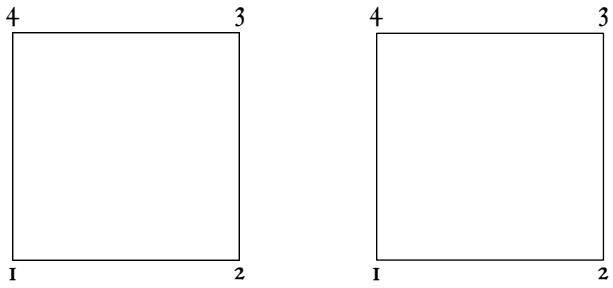
```
\input labtex
\input epsf
\newcount\epsfurX   \newcount\epsfurY
\def\MPfig#1{% usage:
    %%%%\MPfig{figure number}{labels}
    %%%%\endfig
 \figcount=#1 \catcode`\^^M=5
          %%%%to be sure in obeylines
 \setbox\figbox=
    \hbox{\labtex\char\figcount }%
 \global\figwd=\wd\figbox
 \global\fight=\ht\figbox
 \global\figdp=\dp\figbox
 \setbox\figbox=
    \hbox{\epsffile{\fontname\labtex.#1}}%
    \epsfurX\epsfurx\epsfurY\epsfury
 \setbox\figbox=
    \hbox{\epsffile[0 0 {\the\epsfurX} %
    {\the\epsfurY}]{\fontname\labtex.#1}}%
 \zerobox\figbox%
 \hbox\bgroup\box\figbox
       %%%%% now add \point'ing commands
       %%%%% and labels
 \vrule width0pt height\fight depth\figdp}
\def\zerobox #1{\ht#1=0pt \dp#1=0pt \wd#1=0pt }
```

Of course, one should never forget to compile the source in the right order! Using labtex, the order of labels is also important, as we have mentioned earlier (i.e. we must typeset labels in the same order as the labtex macro stores coordinates of their anchor points).

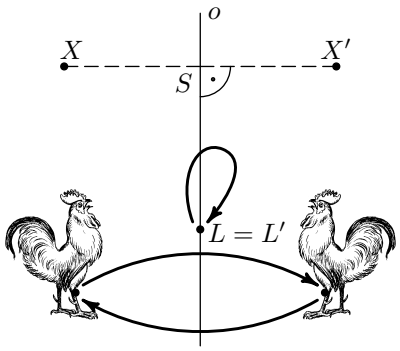
Once the picture is ready, one can easily correct the labels only in the TEX file without recompiling the METAFONT source.

Examples and comments

Next you can see two identical trivial pictures of a square with its vertices labeled by 1, 2, 3, and 4. The first is typeset with a bitmap font, the other with an eps file produced with METAPOST from the same source.



Another example shows an illustration from [6] using two instances of a scanned picture converted to eps:



When testing figures using Alan's original macros, I rediscovered (after some delay, of course) a somewhat strange feature: figures must be numbered sequentially starting from zero, which actually I am not accustomed to. A small change in the original code removes this necessity. Instead of

```
if not known chars[i]:
   beginchar(i,0,0,0); endchar;
   if not known fpf: fpf=i; fi
fi
```

one can find the number of the last known figure without any such assumption:

```
if not known chars[i]:
   beginchar(i,0,0,0); endchar;
   else: fpf:=i;  % last known figure
fi
```

On the other hand, if I am preparing pictures for anybody who or does not use TEX at all, or does not want to manipulate METAFONT sources, the most secure way is to prepare pictures, typeset them and then convert to EPS using another useful utility of Gdańsk provenance (the `ps_conv` package of P. Pianowski & B. Jackowski).

To my own sorrow I have not succeeded in learning more about ConTEXt and its way of manipulating METAPOST graphics. Then I could probably forget about PSTricks and others... Of course it surely cannot happen without making progress to use a real operating system (here I do not think of Windows of any kind). I am going to be twice as old as TEX is, perhaps this is the right time to make substantial progress...

TEX and METAFONT are genuinely designed to cooperate and there are many ways to enjoy it.

The last example (on the following page) is known to all my friends who, for the last six years, have received every New Year a calendar to be cut out and glued together to form a polytope.

References

[1] John D. Hobby: A User's Manual for METAPOST. `http://ctan.org/tex-archive/graphics/metapost`.

[2] Alan Hoenig: When TEX and METAFONT work together. Proceedings of the 7th European TEX Conference, Prague, 1992.

[3] Alan Hoenig: Labelling figures in TEX documents. *TUGboat* 12(1), 1991, pp. 125–128.

[4] Alan Hoenig: TEX Unbound. LATEX & TEX Strategies for Fonts, Graphics, & More. Oxford University Press, Oxford-New York, 1998.

[5] Alan Jeffrey: Labelled diagrams in METAFONT. *TUGboat* 12(2), 1991, pp. 227–229.

[6] František Kuřina: 10 geometrických transformací (10 geometric transformations; in Czech). Prometheus, Praha, 2002.