Embedding fonts in MetaPost output

Troy Henderson

Abstract

MetaPost [1, 2, 3] is a powerful graphics language (by John Hobby) based on Donald Knuth's METAFONT [4] with high quality PostScript output. An outstanding feature of MetaPost is that typeset fonts in the output graphics are consistent with those in TeX-based documents. However, MetaPost does not embed these fonts inside the output PostScript files. This article addresses a specific technique for performing such an embedding.

1 Introduction

MetaPost is one of the most elegant means for generating high quality vector graphics. The language itself is very mathematical in nature and consists of statements that draw and fill paths and label typeset text. The very name itself indicates that MetaPost is a language that generates another language, namely PostScript. With PostScript as output, the graphics are perfectly scalable to any arbitrary resolution. John Hobby, its author, writes:

> "[MetaPost] is really a programming language for generating graphics, especially figures for TeX [5] and troff documents."

This quote by Hobby indicates that MetaPost figures are not only intended to be embedded inside of TeX-based documents but also require TeX to be complete. This is apparent when examining the PostScript containing typeset text which is output by MetaPost. MetaPost does not embed the fonts used inside of the output files. The philosophy for this is that there is no real need for such embedding if the figures are going to appear inside a TeX document because TeX itself will embed the necessary fonts. However, as with any programming language, the source is often compiled (and viewed) several times before the user completes each figure. So, at least for debugging purposes, self-contained output is often desired. That is, we often want PostScript output which has all necessary fonts embedded.

2 Embedding the fonts

According to the statement above, a naïve approach to performing this embedding is to simply include the figure inside a TeX-based document, run TeX, and use Dvips to generate the stand-alone PostScript graphic. These steps highlight the embedding process; however, several details must be addressed in order to create a fully functional approach. In particular, in order to embed the figure into a TeX document, the size of the figure must first be determined so that the paper size of the resulting DVI document is correct. If the paper size is too small, then the TeX→Dvips process will clip the figure. Determination of the appropriate paper size can be done either during or after the MetaPost process.

To make this concrete, suppose we have a MetaPost file `foo.mp` with the contents:

```
beginfig(1)
   draw commands
endfig;
beginfig(2)
   draw commands
endfig;
  ⋮
end
```

The Perl script `mpstoeps`, available from

`http://ctan.org/tex-archive/graphics/metapost/contrib/tools/mpstoeps/`,

can automate the above process. `mpstoeps` assumes that the filename of each figure is of the form `foo_1.mps`, `foo_2.mps`, ... as opposed to the canonical `foo.1`, `foo.2`, ... naming scheme. `mpstoeps` transforms a MetaPost figure into a stand-alone EPS in a method explained in greater detail in Section 3. Furthermore, `mpstoeps` tightens the bounding box of the resulting EPS so that it matches that of the original MetaPost output.

3 Nuts and bolts

As an alternative to the *magical* `mpstoeps`, we may also use MetaPost itself to determine the width and height of the paper size needed to include the figure in a TeX document. As a first step in accomplishing this task, we place

```
numeric w,h;
w := xpart urcorner bbox currentpicture
     - xpart llcorner bbox currentpicture;
h := ypart urcorner bbox currentpicture
     - ypart llcorner bbox currentpicture;
```

immediately before the `endfig` statement. Once these lengths `w` and `h` are determined, we continue by identifying a good basename for the working files.

```
string base;
base:=jobname&"_"&decimal(charcode);
```

We now begin writing an external LaTeX file which will use the `geometry` and `graphicx` packages.

```
write "\documentclass{article}" to base&".tex";
write "\usepackage{geometry}" to base&".tex";
write "\usepackage{graphicx}" to base&".tex";
```

The `geometry` package is used to guarantee that the output will have the precise geometry (i.e. paper size, margins, etc.) needed.

```
write "\geometry{papersize={"
     & decimal(ceiling(w)) & "bp,"
     & decimal(ceiling(h)) & "bp}}"
     to base&".tex";
write "\geometry{margin={0bp,0bp}}"
     to base&".tex";
write "\geometry{noheadfoot,nomarginpar}"
     to base&".tex";
```

Once these preliminaries for the LaTeX document are established, we then insert the MetaPost output file and complete the document.

```
write "\begin{document}" to base&".tex";
write "\thispagestyle{empty}" to base&".tex";
write "\noindent\includegraphics{"
     & jobname & "." & decimal(charcode) & "}"
     to base&".tex";
write "\end{document}" to base&".tex";
write EOF to base&".tex";
```

Now that the document is complete, we output a few messages so that the user knows the precise commands to execute in order to create the stand-alone EPS. This must be done because MetaPost (for security reasons) will not call external commands.

```
message "===============================";
message "Execute these commands to generate "
       & base&".eps:";
message "latex " & base& ".tex";
message "dvips -E -T " & decimal(ceiling(w))
       & "bp," & decimal(ceiling(h))
       & "bp -q -o "&base&".eps "&base&".dvi";
message "===============================";
message "";
```

It is worth noting that even though this process uses LaTeX, the MetaPost process itself does not necessarily use LaTeX to process the text. On most MetaPost installations, the default processor for text labels is plain TeX. Furthermore, the above process for embedding fonts usually increases the bounding box of the figure by at least 1 bp on each side, unlike `mpstoeps`(which simply preserves the original bounding box).

4 Typesetting fonts using LaTeX

As previously mentioned, for most MetaPost distributions, the default processor for text labels is plain TeX. However, some users find it convenient to use LaTeX to process the text. For example, LaTeX users are accustomed to using `$\frac{a}{b}$` to create fractions; this command (as well as many others) is not available in TeX. A typical MetaPost source file `bar.mp` which uses LaTeX to process the text may be organized in the following manner.

```
verbatimtex
\documentclass{amsart}
\begin{document}
etex
beginfig(1)
   draw commands
endfig;
beginfig(2)
   draw commands
endfig;
  ⋮
end
```

However, this format for `bar.mp` alone is not sufficient to force LaTeX to process the text. The `mpost` command must also be instructed to use LaTeX. This can be done via

`mpost -tex=latex bar.mp`.

To make this preference the default under teTeX, we set the environment variable `TEX=latex`.

5 Working example

We now illustrate the processes mentioned above by applying them to a simple MetaPost figure. We will use two copies of the figure — one with `mpstoeps` and the other with the process described in Section 3. Both figures will be defined in a MetaPost source file `tri.mp`.[1] In order to use LaTeX to process the text labels, we preface the code with:

```
verbatimtex
\documentclass{article}
\begin{document}
etex
```

We then draw the figure with the following commands:

```
picture pict;
beginfig(1)
  u:=36;
  w:=fontsize defaultfont;
      x1=u;x2=u*cosd(120);
  y1=0;y2=u*sind(120);
  draw (x1,y1)--(x2,y2)--(x2,-y2)--cycle;
  label(btex $a$ etex,(x1-w,y1));
  label.lft(btex $A$ etex,(x2,y1));
  label(btex $b$ etex,(x2-w*cosd(120),y2-w*
      sind(120)));
```

[1] The electronic version of this article contains `tri.mp` embedded as an attachment to the PDF.

```
   label.lrt(btex $B$ etex,((x1+x2)/2,-y2/2));
   label(btex $c$ etex,((u-w)*cosd(120),(-u+w)*
       sind(120)));
   label.urt(btex $C$ etex,((x1+x2)/2,y2/2));
   pict:=currentpicture;
endfig;
```

We store the picture into pict since we want to reuse it in the next figure. We then "reload" it into the next figure by

```
beginfig(2)
   currentpicture:=pict;
   write commands from Section 3
   message commands from Section 3
endfig;
```

Finally, we append the canonical closing statements for MetaPost.

```
end
```

Once tri.mp is compiled, we rename tri.1 to tri_1.mps and apply mpstoeps. This provides the stand-alone EPS tri_1.eps with all fonts embedded:

```
mv tri.1 tri_1.mps
mpstoeps tri_1.mps
```

Furthermore, we are also instructed to execute

```
latex tri_2.tex
dvips -E -T 69bp,67bp -q -o tri_2.eps tri_2.dvi
```

After following these steps, we obtain tri_2.eps, which is virtually identical to tri_1.eps, and both of these EPS files have their fonts embedded. This mutual figure is shown below:

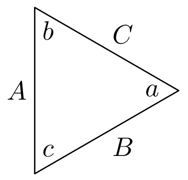

Figure 1: Output of tri.mp.

6 Conclusion

It is worth mentioning that although both stand-alone EPS graphics in Section 5 look virtually identical to the original MetaPost output, they are significantly larger in file size. This drastic difference is clearly due to size of the embedded fonts.

Also, renaming the MetaPost output files using the .mps naming scheme is a convenient method for using either LaTeX or pdfLaTeX to compile the document. The latter does not allow arbitrary PostScript graphics, but does support MetaPost output — as long as the file is renamed with extension .mps.

As a final note, arbitrary EPS files must first be converted to PDF before they can be included with pdfLaTeX. Thanks to Hans Hagen, many distributions of TeX now include a utility called mptopdf which provides a method of easily converting such graphics to PDF.

References

[1] J. D. Hobby. Drawing graphs with MetaPost. Technical Report 164, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Also available at http://www.tug.org/docs/metapost/mpgraph.pdf.

[2] J. D. Hobby. Introduction to MetaPost. EuroTeX '92 Proceedings, pages 21–36, 1992. Also available at http://www.tug.org/docs/metapost/mpintro.pdf.

[3] J. D. Hobby. A user's manual for MetaPost. Technical Report 162, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Also available at http://www.tug.org/docs/metapost/mpman.pdf.

[4] D. E. Knuth. *METAFONT the Program*, volume D of *Computers and Typesetting*. Addison Wesley, Boston, 1986.

[5] D. E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison Wesley, Boston, 1986.

◇ Troy Henderson
Dept. of Mathematical Sciences
United States Military Academy
West Point, NY 10996
USA
troy@tlhiv.org
http://www.tlhiv.org