Hans Hagen
pragma@wxs.nl

metapost

Making stand alone METAPOST graphics

keywords
METAPOST, pdf, pdfTeX

abstract
When a METAPOST graphic uses fonts, the **PostScript** file is not self contained and hardly usable outside TeX. One can however use TeX itself, or actually **pdfTeX**, to create such a graphic. Although this method uses an **ConTeXt** module, the solution provided here is independant of this macro package. The macros responsible for the process are collected in the file mptopdf.tex.

The file `mptopdf` provides a quick way to convert METAPOST files to PDF using a slightly stripped down plain TeX, PDFTeX, and a few CONTEXT modules.

First generate a format, which in WEB2C looks like:

```
pdftex --ini mptopdf
```

Since this conversion only works with PDFTeX or PDF-ε-TeX, the session is aborted when another TeX is used. When finished, the resulting `fmt` file should be moved to the right location.

The conversion itself is accomplished by:

```
pdftex &mptopdf \relax filename.number
```

The `\relax` is needed since we don't want to process the file directly. Instead we pick up the filename using `\everypar`. Since this file is still the first one we load, although delayed, the jobname is as we expect. So, at least in WEB2C, the result of the conversion comes available in the file `filename.pdf`. This conversion process is roughly compatible with:

```
texexec --pdf --fig=c --result=filename.pdf filename.number
```

This uses CONTEXT, and is therefore slower.

The implementation is rather simple, since we use some generic CONTEXT modules. Because we need a few register allocation macros, we preload plain TeX. We don't load fonts yet.

1
```
\input syst-tex
```

We check for the usage of PDFTeX, and quit if another TeX is used.

2
```
\ifx\pdfoutput\undefined
  \message{Sorry, you should use pdf(e)TeX instead.}
  \expandafter \endinput
\fi
```

The conversion to PDF is carried out by macros, that are collected in the file:

3
```
\input supp-pdf
```

We use no output routine.

4
```
\output{}
```

Since we need to calculate and set the bounding box, we definitely don't want to indent paragraphs.

5
```
\parindent=0pt
```

We use `\everypar` to pick up the filename and process the METAPOST graphic.

6
```
\everypar{\processMPfile}
```

The main macro shows a few PDFTEX primitives. The main work is done by the macro `\convertMPtoPDF` which is defined in `upp-pdf}. Thi` macro interprets the METAPOST file. Close reading of this macro will probably learn a few (PDF) tricks. Apart from some path transformations, which are needed since PDF has a different vision on paths, the graphic is positioned in such a way that accuracy in PDF xforms is guaranteed.

7
```
\def\processMPfile#1 %
  {\pdfoutput=1
   \setbox0=\vbox{\convertMPtoPDF{#1}{1}{1}}%
   \ifdim\wd0<1in \message{[warning:  width<1in]}\fi
   \ifdim\ht0<1in \message{[warning: height<1in]}\fi
   \pdfpageheight=\ht0
   \pdfpagewidth=\wd0
   \voffset=-1in
   \hoffset=\voffset
   \box0
   \bye}
```

Since acrobat has troubles with figures smaller than 1 inch, we issue a warning. When embedding graphics in documents, a size less that 1 inch does not harm.

The resulting PDF file is about as efficient as such a self contained file can be. However, if needed, this PDF file can be converted to EPS using for instance the `pdftops` program (in WEB2C) or GHOSTSCRIPT.

8
```
\dump
```