Using external EPS graphics in METAPOST
The exteps module
Version 0.1

Palle Jørgensen

10th August 2005

Contents

1. Introduction

This document describes the use of the exteps module for inclusion of external eps figures in METAPOST figures. Unlike the previous attempt (epsincl) it make no use external programmes; it is solely written in METAPOST.

The EPS graphics is included using the *special* command in METAPOST.

2. Using exteps

To illustrate the use of the exteps module an example is given below. Between the begineps and endeps commands both settings can be set, as well as special drawing commands can be added. The output of the example can be seen in figure 1 and the original picture.

```
input exteps

beginfig(1);
  begineps "pallej.eps";
    base := (25,25);
%    width=10cm;
    clip := true;
    grid := true;
    epsdrawdot(36pct,80pct) withpen pencircle scaled 10pct
        withcolor blue;
    epsdrawdot(60.5pct,80pct) withpen pencircle scaled 10pct
        withcolor blue;
  endeps;
draw origin withpen pencircle scaled 50 withcolor red + green;
endfig;

end.
```

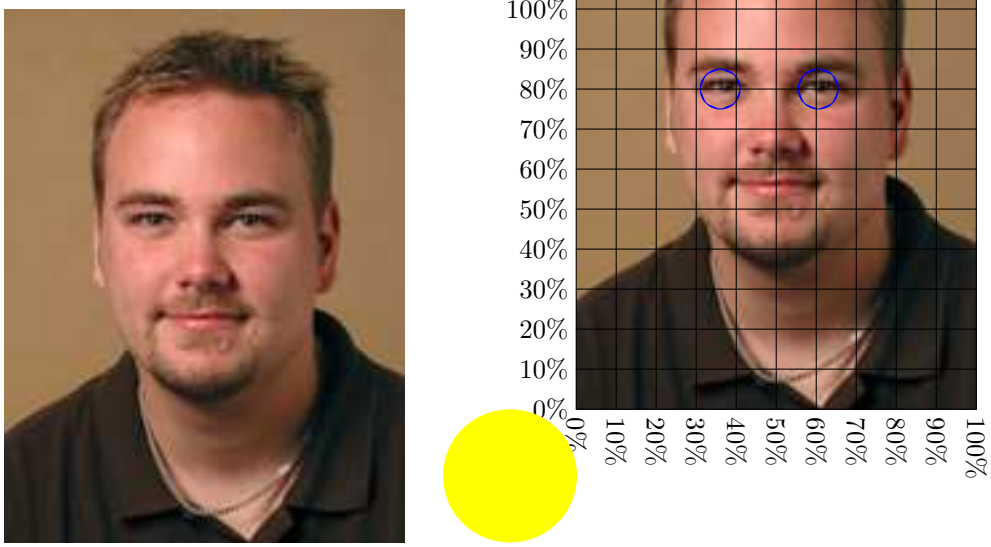

Figure 1: The original (left) and the with exteps modified picture

2.1. Settings

The parameters of the settings and their defaults can be seen the table below.

Parameter	Type	Default	Description.
angle	numeric	0	The counterclockwise rotation of the EPS figure.
clip	boolean	false	If true, the EPS figure is clipped to its bounding box.
base	pair	(0,0)	The offset of the lower left corner of the EPS picture.
scale	pair	(1,1)	The scale of the picture.
width	numeric	No default	Specify the width of the picture; overrules the scale setting.
height	numeric	No default	Specify the height of the picture; overrules the scale setting.
grid	boolean	false	If true a grid is draw on top of the picture; mostly (only?) meant to help when drawing on top of the EPS figure.
gridstep	numeric	10	The distance in percent between the lines of the grid.

2.2. Special values

`begineps` saves the original bounding box of the EPS picture in the values `llx`, `lly`, `urx` and `urx`. These values can be used in the settings, and for drawing commands. Furthermore a numeric value `pct` is set. This is a length that is one percent of the width of the picture.

If for instance one wants the picture to be placed at the same place on the page as the original picture it is simply typing

```
base:=(llx,lly);
```

between `begineps` and `endeps`.

2.3. Drawing commands

When `begineps` is called a special picture, `epspicture`, is created. To draw on this picture, and whence drawing on the EPS picture the special commands `epsdraw`, `epsfill`, `epsfilldraw`, `epsdrawdot` and `epslabel` are defined. They work as the normal drawing commands, but now adds to the `epspicture`.

At `endeps` the `epspicture` is scaled, rotated and translated in the same way as the included EPS figure.

3. Limitations of `exteps`

- As the eps files are included without any special wrapping some PostScript commands may cause trouble, especially the `showpage` command. However, when included into other documents, e.g. a TEX document, this is not a problem.

  If you have to get the picture out as it is, I can recommend the programme `mps2eps` written by Jon Edvardsson. `mps2eps` can be found at `http://www.ida.liu.se/~joned/download/mps2eps/`.

- Furthermore `exteps` only looks at the first line in the document that says `%%BoundingBox: ...`

  Whence it will cause trouble if this line does not provide the bounding box; some PostScript drivers may write `%%BoundingBox: (atend)`. This is not supported.

- As the modlue makes it possible to include external EPS pictures it may not possible to use the output with PDFTEX.

4. Comments and Bug Reports

All comments, questions and bug reports, both on the module itself as well as this document may be sent to Palle Jørgensen, `hamselv@pallej.dk`.

A. Source code of exteps

```
picture epspicture;

%% String handling tool
string string_split[];
def splitstring expr S =
  begingroup
    save __splitctr; numeric __splitctr; __splitctr = 0;
    save __prevchar; string __prevchar, __currentchar;
       __prevchar = " ";
    for i = 0 upto infinity:
      __currentchar := substring(i, i+1) of S;
      if (__currentchar = " ") and (__prevchar = " "):
        relax;
      elseif (__currentchar <> " ") and (__prevchar = " "):
        string_split[__splitctr] := __currentchar;
      elseif (__currentchar <> " ") and (__prevchar <> " "):
        string_split[__splitctr] := string_split[__splitctr
           ] & __currentchar;
      elseif (__currentchar = " ") and (__prevchar <> " "):
        __splitctr := __splitctr+1;
      fi
      __prevchar := __currentchar;
    endfor
  endgroup;
enddef;
%% End string handling tool

def begineps text F =
  begingroup;
    save file;                  string file;         file = F;
    save angle;                 numeric angle;       angle = 0;
    save clip;                  boolean clip;        clip = false
       ;
    save scale;                 pair scale;          scale
        = (1,1);
    save base;                  pair base;           base =
       origin;
    save __bbxfound;            boolean __bbxfound;  __bbxfound
        = false;
    save grid;                  boolean grid;        grid = false
       ;
```

```
    save gridstep;              numeric gridstep;   gridstep
        = 10;
    save __base;                pair __base;
    save  __eps__currentline; string __eps__currentline;
    save __bbxline;             string __bbxline;
    save llx, lly, urx, ury;    numeric llx, lly, urx, ury;
    save pct;                   numeric pct;
    save width;                 numeric width;
    save height;                numeric height;
%% Finding the bounding box
    forever:
      __eps__currentline := readfrom F;
      if substring(0,14) of __eps__currentline = "%%
          BoundingBox:":
        __bbxline := substring(14, infinity) of
            __eps__currentline;
        __bbxfound := true;
        splitstring __bbxline;
        llx = scantokens string_split[0];
        lly = scantokens string_split[1];
        urx = scantokens string_split[2];
        ury = scantokens string_split[3];
      fi
      exitif __bbxfound;
    endfor
    closefrom F;
    __base = -(llx,lly);
    pct = (urx - llx)/100;
%% To ensure the right bounding box of the output file
%% a picture with the same size as the eps figure is added.
    epspicture := nullpicture;
    setbounds epspicture to ((0,0)--(0,ury-lly)--(urx-llx,ury-
        lly)--(urx-llx,0)--cycle);
enddef;

def endeps =
%% Drawing the grid
  if grid:
    for i = 0 step gridstep*pct until (urx - llx):
      epsdraw (i,0)--(i,ury - lly);
      epslabel.bot(((decimal.(i/pct) & "%") infont defaultfont
          ) rotated -90,(i,0));
    endfor
    for i = 0 step gridstep*pct until (epsilon + ury - lly):
      epsdraw (0,i)--(urx - llx,i);
```

```
      epslabel.lft(((decimal.(i/pct) & "%") infont defaultfont
        ),(0,i));
    endfor
  fi
%% Calculating scale if width and/or height is known
  if (known width) and (known height):
    scale := (width/(urx - llx),height/(ury - lly));
  elseif known width:
    scale := (width/(urx - llx),width/(urx - llx));
  elseif known height:
    scale := (height/(ury - lly),height/(ury - lly));
  fi
%% The graphics inclusion commands
  special "gsave";
  if base <> origin:
    special decimal.xpart.base & " " & decimal.ypart.base & "
      translate";
  fi
  if scale <> (1,1):
    special decimal xpart.scale & " " & decimal ypart.scale
      & " scale";
  fi
  if angle <> 0:
    special decimal angle & " rotate";
    epspicture := epspicture rotatedaround(origin)(angle);
  fi
    if __base <> origin:
    special decimal.xpart.__base & " " & decimal.ypart.__base
      & " translate";
  fi
  if scale <> (1,1):
    epspicture := epspicture scaled xpart.scale
    if xpart.scale <> ypart.scale:
      yscaled (ypart.scale/xpart.scale)
    fi;
  fi
  if clip:
    special "newpath ";
    special decimal llx & " " & decimal lly & " moveto";
    special decimal llx & " " & decimal ury & " lineto";
    special decimal urx & " " & decimal ury & " lineto";
    special decimal urx & " " & decimal lly & " lineto";
    special decimal llx & " " & decimal lly & " lineto";
    special "closepath clip";
  fi
```

```
  special "%%BeginDocument: " & file;
  forever:
    __eps__currentline := readfrom file;
    exitunless __eps__currentline <> EOF;
    special __eps__currentline;
  endfor
  special "%%EndDocument: " & file;
  special "grestore";
  closefrom file;
  if base <> (0,0):
    epspicture :=  epspicture shifted base;
  fi
  addto currentpicture also epspicture;
endgroup;
enddef;

%% Special drawing commands
def epsfill expr c = addto epspicture contour c _op_ enddef;

def epsdraw expr p =
  addto epspicture
  if picture p:
    also p
  else:
    doublepath p withpen currentpen
  fi
  _op_
enddef;

def epsfilldraw expr c =
  addto epspicture contour c withpen currentpen
  _op_ enddef;

def epsdrawdot expr z =
  addto epspicture contour makepath currentpen shifted z
  _op_ enddef;

def epslabel = epsdraw thelabel enddef;

endinput
```