# USING EXTERNAL EPS GRAPHICS IN METAPOST

## THE exteps MODULE

## VERSION 0.41

Palle Jørgensen

26th September 2006

## 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 programs[1]; it is entirely written in METAPOST.

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

---

[1] This is only partly true, as METAPOST is unable to handle large files; a workaround is described in appendix B on page 15

## 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 and the original picture can be seen in figure 1.

```
input exteps

prologues:=2;

beginfig(1);
  begineps "pallej.eps";
    base := (25,25);
    clipping := true;
    grid := true;
    epsdrawdot(36pct,80pct) withpen pencircle scaled 10pct withcolor blue;
    epsdrawdot(60.5pct,80pct) withpen pencircle scaled 10pct withcolor blue;
    epsdraw (35pct,60pct)..(48pct,54pct){right}..(61pct,60pct) withpen pencircle
        scaled 2pct withcolor red;
    endeps;
draw origin withpen pencircle scaled 50 withcolor red + green;
endfig;
```

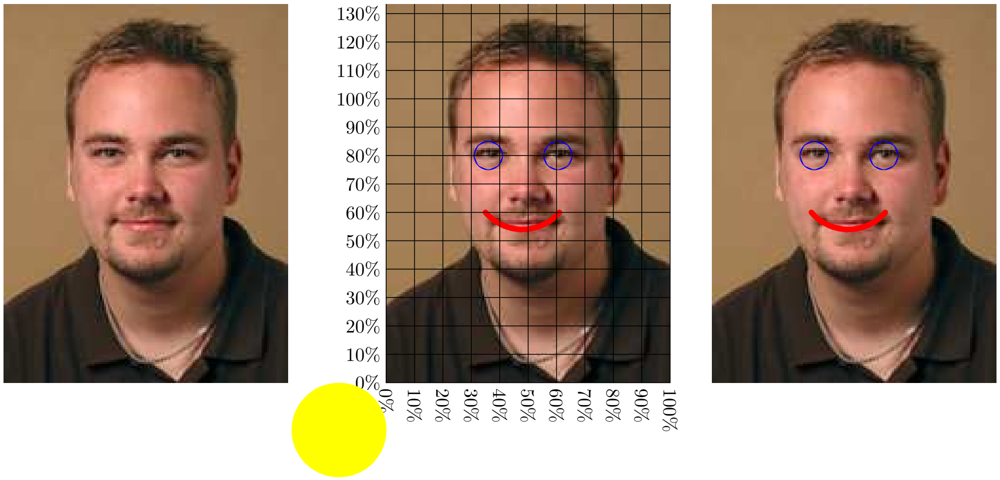

Figure 1: The original (left) and the with `exteps` modified picture (middle). To the right there is one without the grid and the dot at the origin.

## 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. |
| clipping[2] | boolean | false | If true, the EPS figure is clipped to its bounding box. |
| clippath | path | Bounding box of the EPS file | Path to which the EPS picture is clipped, if the *clipping* switch is *true.* |
| base | pair | (0,0) | The offset of the lower left corner of the EPS picture. |
| scale | pair | (1,1) | The scaling 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. |
| gridllx | numeric | 0 | The x-part of the lower left corner of the grid; in pct |
| gridlly | numeric | 0 | The y-part |
| gridurx | numeric | *full width* | The x-part of the upper rightt corner of the grid |
| gridury | numeric | *full height* | The y-part |

## 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.

---

[2] In version 0.1 named *clip*

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`.

## 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.

## 4. Clipping the EPS picture

From version 0.2 it is possible to do advanced clipping of the EPS picture.

This is done by specifying the path *clippath* along which the EPS picture is to be clipped, and setting *clipping* to *true*.

A minor example and the result in figure 2 on page 6:

```
beginfig(4);
  begineps "pallej.eps";
    base := (25,25);
    clipping := true;
    clippath := (50pct,10pct)..(15pct,70pct)..(50pct,130pct)..(85pct,70pct)..cycle;
  endeps;
endfig;
```

Please note that this does not clip the `epspicture`. You can do this manually by specifying

```
clip epspicture to clippath;
```

or

```
setbounds epspicture to clippath;
```

The section about the grid on page 6 also provides an example of the clipping commands.

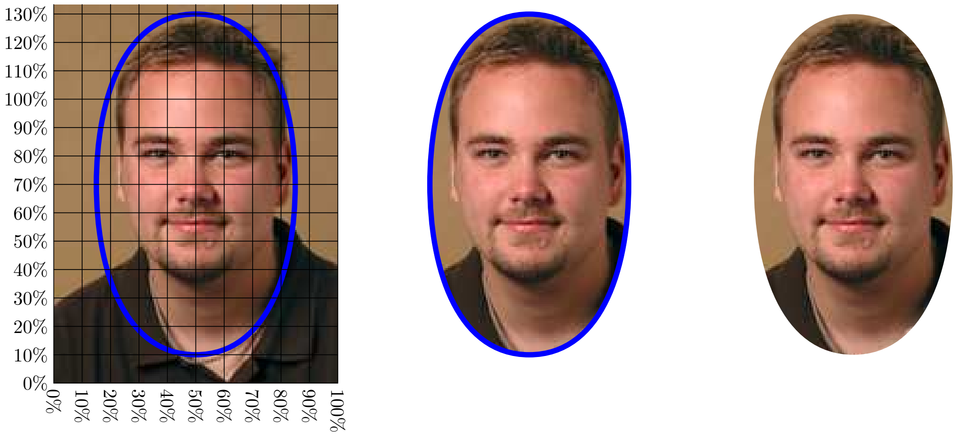

Figure 2: The clipped picture to the right, and the original with a grid to the left. The clippath is marked with blue. The picture in the midlle is created by keeping the blue line on top of the clipped picture.

## 5. THE GRID

It is possible to finetune the settings og the grid, for instance when clipping the picture. The example below shows the impact of setting the values of `gridstep`, `gridllx`, `gridlly`, `gridurx`, and `gridury`. The result can be seen in figure 3 on page 7.

```
beginfig(6);
  begineps "pallej.eps";
    clipping := true;
    clippath := (20pct,40pct)−−(80pct,40pct)−−(80pct,110pct)−−(20pct,110pct)−−
         cycle;
    setbounds epspicture to clippath;
    scale := (1.5,1.5);
    grid := true;
    gridstep := 5;
    gridllx := 20;
    gridlly := 40;
    gridurx := 80;
    gridury := 110;
  endeps;
endfig;
end.
```

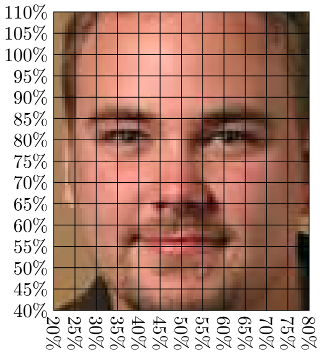

Figure 3: The picture with the finetuned grid, and some clipping.

## 3. Limitations of `exteps`

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

  Thus 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 all of the graphics inclusion is done with METAPOST, it is limited by METAPOST's memory capacity. More specific on the string pool size.

  Read more about this problem, and about the *delfin* workaround perl script in Appendix B on page 15

- As the module makes it possible to include external EPS pictures it may not be possible to use the output with pdfTEX.

  A way to get around this is to use the program epstopdf.

  epstopdf is located on CTAN at `http://tug.ctan.org/tex-archive/support/epstopdf/`.

  Another possible work-around for this is to use the program purifyeps to "clean up" the PostScript picture.

  purifyeps is located on CTAN at `http://tug.ctan.org/tex-archive/support/purifyeps/`.

  Yet another work-around is to use the program pstoedit to generate METAPOST code form EPS files and include this into your METAPOST file.

See `http://www.pstoedit.net/pstoedit/`.

# 4. CHANGES

## 1. FROM VERSION 0.1 TO 0.2

- Adding advanced clipping, see section 4 on page 5.
- Renaming the *clip* switch to *clipping*.
- Eliminating the `showpage` "problem" in version 0.1.

## 2. FROM VERSION 0.2 TO 0.3

- Improvement of the grid drawing commands. The grid is now drawn *after* the scaling of the picture.
- Introducing a workaround for large files. This includes a Perl script.

# 5. 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.

# 6. THIS DOCUMENT

Source of this document and the used example can be found at `http://pallej.dk/exteps/`.

# A. Source code of exteps

```
picture epspicture;
string extra_begineps; extra_begineps = "";
string extra_endeps; extra_endeps = "";
boolean extepsverbose; extepsverbose = true;

%% 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 clipping; boolean clipping; clipping = 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;
    save clippath; path clippath;
    save largefile; boolean largefile; largefile = false;
    save gridllx; numeric gridllx;
    save gridlly; numeric gridlly;
    save gridurx; numeric gridurx;
    save gridury; numeric gridury;
%% Finding the bounding box
    forever:
      __eps__currentline := readfrom F;
      if substring(0,14) of __eps__currentline = "%%BoundingBox:":
        exitif __eps__currentline = EOF; % PATCH D. Roegel 23−sep−2006
        __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
    if not __bbxfound:
      message "Warning:␣␣No␣bounding␣box␣found.";
      message "␣␣␣␣␣␣␣␣␣␣Setting␣bounding␣box␣to␣0␣0␣1␣1";
      llx = lly = 0;
      urx = ury = 1;
    fi
    closefrom F;
    scantokens extra_begineps;
    __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;
    clippath = (0,0)−−(0,ury−lly)−−(urx−llx,ury−lly)−−(urx−llx,0)−−cycle;
    setbounds epspicture to clippath;
enddef;

def endeps =
%% 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 angle <> 0:
    special decimal angle & "␣rotate";
  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;
    special decimal xpart.scale & "␣" & decimal ypart.scale & "␣scale";
  fi
  if angle <> 0:
    epspicture := epspicture rotatedaround(origin)(angle);
  fi
%% Drawing the grid !! After the scaling :−)
  if unknown gridllx:
    gridllx = 0;
  fi
  if unknown gridlly:
    gridlly = 0;
  fi
  if unknown gridurx:
    gridurx = (urx − llx)/pct;
  fi
  if unknown gridury:
    gridury = (ury − lly)/pct;
  fi
  if grid:
    save __gridpicture; picture __gridpicture; __gridpicture := nullpicture;
    for i = gridllx*pct step gridstep*pct until (epsilon + gridurx*pct):
      addto __gridpicture doublepath (i*xpart.scale,gridlly*pct*ypart.scale)−−(i*
          xpart.scale,gridury*pct*ypart.scale) withpen currentpen;
      addto __gridpicture also thelabel.bot(((decimal.(i/pct) & "%") infont
          defaultfont) rotated −90, (i*xpart.scale,gridlly*pct*ypart.scale));
    endfor
    for i = gridlly*pct step gridstep*pct until (epsilon + gridury*pct):
      addto __gridpicture doublepath (gridllx*pct*xpart.scale,i*ypart.scale)−−(
          gridurx*pct*xpart.scale,i*ypart.scale) withpen currentpen;
      addto __gridpicture also thelabel.lft(((decimal.(i/pct) & "%") infont
          defaultfont), (gridllx*pct*xpart.scale,i*ypart.scale));
    endfor
    if angle <> 0:
      __gridpicture := __gridpicture rotatedaround(origin)(angle);
    fi
    addto epspicture also __gridpicture;
  fi
  if clipping:
    save __clippath; path __clippath;
    __clippath=clippath shifted (llx,lly);
    special "newpath␣" & decimal.xpart.point 0 of __clippath
    & "␣" & decimal.ypart.point 0 of __clippath & "␣moveto";
    for i = 0 upto length.__clippath−1:
      special decimal.xpart.postcontrol i of __clippath & "␣" &
      decimal.ypart.postcontrol i of __clippath & "␣" &
      decimal.xpart.precontrol (i+1) of __clippath & "␣" &
      decimal.ypart.precontrol (i+1) of __clippath & "␣" &
      decimal.xpart.point (i+1) of __clippath & "␣" &
      decimal.ypart.point (i+1) of __clippath & "␣curveto";
    endfor;
    special "closepath␣clip";
  fi
  special "save";
  special "userdict␣begin";
  special "/showpage␣{}␣def";
    special "%%BeginDocument:␣" & file;
    if largefile:
      special "%%␣MetaPost␣exteps␣large␣file−>" & file;
      if extepsverbose:
        message "exteps␣notification:␣␣File␣" & file & "␣not␣inserted␣into␣" &
            jobname & "." & decimal.charcode;
        message "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Run␣′delfin␣" & jobname & "." &
            decimal.charcode & "′␣to␣insert␣" & file;
        message "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣This␣is␣caused␣by␣setting␣′
            largefile:=true′";
        message "";
      fi
    else:
      if extepsverbose:
        message "Inserting␣" & file & "␣into␣" & jobname & "." & decimal.
            charcode;
      fi
      forever:
        __eps__currentline := readfrom file;
        exitunless __eps__currentline <> EOF;
        special __eps__currentline;
      endfor
    fi
    special "%%EndDocument:␣" & file;
    scantokens extra_endeps;
    special "end␣restore";
    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

## B. Large EPS files

In case of a too large EPS file, the `exteps` module causes an error message from METAPOST, due to the limited memory capacity of METAPOST.

The error message looks somewhat like this:

```
camel25:~/tmp% mpost et.mp
This is MetaPost, Version 0.641 (Web2C 7.5.2)
(/usr/local/TeX/texmf/web2c/cp8bit.tcx)
(et.mp (/users/pallej/texmf/metapost/exteps.mp)
Inserting sk.eps into et.1
! MetaPost capacity exceeded, sorry [pool size=476396].
<read>

<forever> __eps__currentline:=readfrom.file;
                                            exitunless.__eps__currentline<>E...

endeps->...<>EOF;special.__eps__currentline;endfor
                                                  .fi.special"%%EndDocument:...
l.14      endeps
```

A workaround for this problem is to set the value `largefile` to `true`:

largefile:=**true**;

exteps now writes

*%% MetaPost exteps large file−>file.eps*

into the METAPOST output file. Afterwards one must run the Perl script `delfin` onto the METAPOST output file.

First run METAPOST:

```
This is MetaPost, Version 0.641 (Web2C 7.5.2)
(/usr/local/TeX/texmf/web2c/cp8bit.tcx)
(et.mp (/users/pallej/texmf/metapost/exteps.mp)
exteps notification:  File sk.eps not inserted into et.1
                      Run 'delfin et.1' to insert sk.eps
                      This is caused by setting 'largefile:=true'
[1] )
1 output file written: et.1
Transcript written on et.log.
camel25:~/tmp%
```

and then `delfin`

```
camel25:~/tmp% delfin et.1
This is delfin version 0.1
Delfin, the Exteps Large File INserter
Inserting sk.eps into et.1
camel25:~/tmp%
```

It is possible to turn off the `exteps` notification; just set the (global) value `extepsverbose` to false

```
extepsverbose = false;
```

If you are unable to use the `delfin` program, it is still possible to do the finishing. Just open the METAPOST output file in your favourite editor, and replace the line mentioned above with the entire EPS file.

## 1. USING delfin

Usage of the `delfin` program:

```
delfin [options] file.n [file.m [file.l ... ]]
Options:
    -h  Print this message end exit
    -q  Be quiet
    -v  Display version and license and exit
    -V  Display version number and exit
```

## 2. SOURCE OF delfin

```
#! /usr/bin/perl −w
```

[license stuff etc.]

```
use strict;
our($opt_h,$opt_q,$opt_v,$opt_V,$opt_i);

use Getopt::Std;
getopts('−helpvqVi');

use Env qw(HOME);

my $progversion = 0.12;
my $progname = "delfin";
my $prognamelong = "Delfin,␣the␣Exteps␣Large␣File␣INserter";
if ($opt_h) { version(); help(); exit; }
if ($opt_v) { version(); exit; }
if ($opt_V) { versionbrief(); exit; }

unless ( $opt_q ) {
    print "This␣is␣$progname␣version␣$progversion\n";
    print "$prognamelong\n";
}

foreach ( @ARGV ) {
    my $mpsfile = $_;
    my $elffound = 0;
    open (MPSIN, "$mpsfile") or die "Cannot␣open␣file␣$mpsfile";
    my @OUT;
    foreach ( <MPSIN> ) {
        if ($_ =~ /^%% MetaPost exteps large file−>/) {
            $elffound = 1;
            my $epsfile = (split(/−>|\n/,$_))[1];
            unless ( $opt_q ) {
                print "Inserting␣$epsfile␣into␣$mpsfile\n";
            }
            open(EPS, "$epsfile") or die "cannot␣open␣file␣$epsfile";
            push (@OUT, <EPS>);
        } else {
            push (@OUT, $_);
        }
    }
    close MPSIN;
    if ( $elffound ) {
        open (MPSOUT, ">$_") or die "Cannot␣write␣to␣file␣$_";
        print MPSOUT @OUT;
        close MPSOUT;
    }
    else {
        unless ( $opt_q ) {
            print "No␣file␣to␣insert␣into␣$mpsfile\n"
        }
    }
}

sub help {
    print << "EOF";
Usage:
```

```
$progname [options] file.n [file.m [file.l ... ]]
Options:
    −h\tPrint this message end exit
    −q\tBe quiet
    −i\tIgnore configuration file
    −v\tDisplay version and license and exit
    −V\tDisplay version number and exit
See exteps.pdf for further documentation
(texdoc exteps on most unix systems)
EOF
}

sub version {
    print << "EOF"
This is $progname version $progversion
$prognamelong
Copyright 2005 by Palle Jorgensen
The license of $progname is GNU General Public License (GPL)
EOF
}

sub versionbrief {
    print "$progversion\n";
}
```