ConTEXt

TPIC Conversion

category: ConTEXt Support Macros

version: 1997.07.05

date: March 19, 1998

author: Hans Hagen

This modules implements the conversion of graphic TPIC specials using METAPOST.

We reimplement the TPIC specials using the special mimmicking mechanism implemented in the support module `supp-spe` as well as the METAPOST run–time support implemented in `supp-mps`.

1
```
\ifx\undefined\writestatus     \input supp-mis \relax \fi
\ifx\undefined\mimmickspecials \input supp-spe \relax \fi
\ifx\undefined\MPgraphicbox    \input supp-mps \relax \fi
```

2
```
\writestatus{loading}{Context Support Macros / TPIC Conversion}
```

Beware: we haven't activated both mechanism yet. This is to be done in the calling module.

3
```
\unprotect
```

When we want to mimmick TPIC specials in PDFTEX, we need to map its graphic primitives into PDF ones. The main problem in doing so is that PDF does not support b-splines directly and also does not offer us something to draw arcs. Of course all this scan be implemented in TEX, and the first implementation of this module did so, but the results were not that satisfying. Not having used these specials before, I had for instance to find out that the TPIC specials were not that unambiguesly defined.

Then, while discussing something else, Sebastian Ratz told me that the Web2c implementation that PDFTEX is base upon, offers some rather discutable, but nevertheless handy feature:

```
\write18{execute program with arguments}
```

Knowing this, I immediatelly decided to throw away the old conversion macros and use the marvelous METAPOST, TEX related, drawing program to do the conversion in as high a quality as possible.

implementation we're going to present here, not only uses for drawing purposes, but also uses the more efficient METAPOST features to store the path.
Table 1 lists the TPIC specials as mentioned in the LATEX Graphics Companion and the relevant part of the DVIPS source. This list shows us that we have to store the path before we can use it, simply because we don't know in advance what actions to apply on it.

tag	**arguments**	**meaning**
pn	w	set linewidth
pa	$x\ y$	add point to path
fp		draw/fill path
ip		fill path
da	l	draw dashed path
dt	l	draw doted path
sp	d	draw spline
ar	$x\ y\ r_x\ r_y\ b\ e$	draw (partial) arc
ia	$x\ y\ r_x\ r_y\ b\ e$	fill (partial) arc
sh	s	fill next path

Tabel 1 The TPIC special syntax.

The first problem we have to take care of is the fact that there is no decent begin or end of the drawing process defined. We can however be quite sure that writers of packages using these specials will put them into a box, simply because else this is the most common used way to treat something TEX as as a whole, like:

```
\hbox{\special{}\special{}...}
```

We just start a picture as soon as the first special is encountered, so this becomes:

```
\hbox{\openpicture\newspecial{}\newspecial{}...
```

The first step in opening the picture is to start a group. Now we can savely use the egroup that closes the box to also end the picture.

4
```
\def\startTPICspecials%
  {\bgroup
   \let\startTPICspecials=\relax
   \aftergroup\stopTPICspecials
   \startwritingMPgraphic
   \writeMPgraphic{pair p[];}}
```

As soon as we begin a picture, we inhibit nesting by relaxing the start macro. The first METAPOST action we take is declaring an array of pairs named *p*.

Ending the picture is invoked by closing the current group. Because the TPIC picture comes out mirrored, we have to reflect the current METAPOST picture, stored in the system variable *currentpicture*, around the *x*-axis.

5
```
\def\stopTPICspecials%
  {\writeMPgraphic
     {currentpicture:=currentpicture reflectedabout ((0,0),(4095,0));}%
   \stopwritingMPgraphic
   \flushMPgraphics
   \loadcurrentMPgraphic{}%
   \setbox\MPgraphicbox=\hbox to \!!zeropoint
     {\kern-\wd\MPgraphicbox
      \vbox to \!!zeropoint{\box\MPgraphicbox\vss}\hss}%
   \ht\MPgraphicbox=\!!zeropoint
   \wd\MPgraphicbox=\!!zeropoint
   \dp\MPgraphicbox=\!!zeropoint
   \box\MPgraphicbox
   \egroup}
```

Here the macro `\stopwritingMPgraphic` has to take care of executing and including the METAPOST code.

We need to keep track of the number of elements that form the path. This is needed because we don't know in advance how the points are to be connected.

6
```
\newcount\TPICcounter
```

When a path is draw, we can connect the points using a smooth curve of drawing straight lines. A closed path can be drawn or filled.

7
```
\newif\ifTPICdraw
\newif\ifTPICfill
\newif\ifTPICcurve
```

The TPIC specials permit specifying the line and fill color as well as the linetype, which can be solid, dashed or dotted. We'll save those specifications as a METAPOST string, using:

8
```
\let\TPIClinetype =\empty
\let\TPICgrayscale=\empty
```

The magic reduction factor .07227 is needed to map the TPIC 1/1000 of an inch to POSTSCRIPT points. We cannot delegate this task to METAPOST because this program does not accept values greater than 4095.

I won't discuss all the specifics used in implementing the specials. The METAPOST part is rather trivial. Many specials have much in common, so the amout of code is not that large.

```
9   \redefinespecial pa \using#1 #2\endspecial
      {\startTPICspecials
       \bgroup
       \global\advance\TPICcounter by 1
       \dimen0=#1pt \dimen0=.07227\dimen0
       \dimen2=#2pt \dimen2=.07227\dimen2
       \writeMPgraphic{p[\the\TPICcounter]:=(\the\dimen0,\the\dimen2);}%
       \egroup}

10  \redefinespecial pn \using#1\endspecial
      {\startTPICspecials
       \bgroup
       \dimen0=#1pt \dimen0=.07227\dimen0
       \writeMPgraphic{pickup pencircle scaled \the\dimen0;}%
       \egroup}

11  \redefinespecial sh \using#1\endspecial
      {\startTPICspecials
       \bgroup
       \edef\g{#1}%
       \edef\g{\ifx\g\empty.5\else#1\fi}%
       \xdef\TPICgrayscale{withcolor (\g,\g,\g)}%
       \egroup}

12  \redefinespecial wh \using#1\endspecial
      {\mimmickspecial sh \using0\endspecial}

13  \redefinespecial bk \using#1\endspecial
      {\mimmickspecial sh \using1\endspecial}

14  \redefinespecial da \using#1\endspecial
      {\startTPICspecials
       \bgroup
       \edef\l{#1}%
       \ifx\l\empty
         \gdef\TPIClinetype{dashed evenly}%
       \else
         \dimen0=#1in
         \ifdim\dimen0<\!!zeropoint \dimen0=-\dimen0\fi
         \edef\f{\the\dimen0 \space}%
         \dimen0=.5\dimen0
         \edef\h{\the\dimen0 \space}%
         \xdef\TPIClinetype{dashed dashpattern (on \h off \f on \h)}%
       \fi
       \egroup
       \TPICcurvefalse\TPICdrawtrue
       \drawTPICpath\using#1\endspecial}
```

```
15  \redefinespecial dt \using#1\endspecial
      {\startTPICspecials
       \bgroup
       \edef\l{#1}%
       \xdef\TPIClinetype{dashed withdots \ifx\l\empty\else scaled #1in\fi}%
       \egroup
       \TPICcurvefalse\TPICdrawtrue
       \drawTPICpath\using#1\endspecial}
```

```
16  \redefinespecial fp \using#1\endspecial
      {\startTPICspecials
       \TPICcurvefalse\TPICdrawtrue
       \ifdim0#1pt=\!!zeropoint
         \drawTPICpath\using#1\endspecial
       \else\ifdim0#1pt<\!!zeropoint
         \mimmickspecial dt\using#1\endspecial
       \else
         \mimmickspecial da\using#1\endspecial
       \fi\fi}
```

```
17  \redefinespecial sp
      {\startTPICspecials\TPICdrawtrue\TPICcurvetrue\drawTPICpath}
```

```
18  \redefinespecial ip
      {\startTPICspecials\TPICfilltrue\drawTPICpath}
```

```
19  \redefinespecial ar
      {\startTPICspecials\TPICdrawtrue\drawTPICarc}
```

```
20  \redefinespecial ia
      {\startTPICspecials\TPICfilltrue\drawTPICarc}
```

These substitutes use two auxiliary macros that take care of actually drawing the shape or arc. Here we use the stored linetype (solid, dashed, dotted) and color (grayscale).

```
21  \def\drawTPICpath\using#1\endspecial
      {\bgroup
       \ifTPICdraw
         \def\TPICgrayscale{}%
       \fi
       \writeMPgraphic
         {\ifTPICfill fill\fi\ifTPICdraw draw\fi\space
          for i:=1 upto \the\TPICcounter-1:
            p[i]\ifTPICcurve..\else--\fi
          endfor
          p[\the\TPICcounter]
          \ifTPICfill\ifTPICcurve..\else--\fi cycle \fi
          \TPIClinetype\space\TPICgrayscale;}%
       \resetTPICvariables
       \egroup}
```

I have to admit that at the moment I wrote this macro, I could not write this piece of METAPOST. Fortunately Thortsen Ohl promptly answered the question I posted to the METAFONT discussion list.

22
```
\def\drawTPICarc\using#1 #2 #3 #4 #5 #6\endspecial
  {\bgroup
   \ifTPICdraw
     \def\TPICgrayscale{}%
   \fi
   \dimen 0=#1pt\dimen 0=.07227\dimen 0
   \dimen 2=#2pt\dimen 2=.07227\dimen 2
   \dimen10=#3pt\dimen10=.14454\dimen10
   \dimen12=#4pt\dimen12=.14454\dimen12
   \dimen20=#5pt
   \dimen22=#6pt
   \writeMPgraphic
     {\ifTPICfill fill\fi\ifTPICdraw draw\fi \space
      \ifTPICfill\else subpath 4/3.14159*(\the\dimen20,\the\dimen22) of \fi
      fullcircle xscaled \the\dimen10 \space yscaled \the\dimen12 \space
      shifted (\the\dimen0,\the\dimen2)
      \TPIClinetype \space \TPICgrayscale;}%
   \resetTPICvariables
   \egroup}
```

Resetting the variables need to be done globally because we cannot be sure if any further grouping is used by the envelopping macros.

23
```
\def\resetTPICvariables%
  {\global\TPICcounter=0
   \global\TPICfillfalse
   \global\TPICdrawfalse
   \global\let\TPIClinetype=\empty
   \global\let\TPICgrayscale=\empty}
```

I have to admit that by using the METAPOST Bézier cubics routines these implementation does produce better curves then most DVI drivers do using the TPIC prescribed b-splines. Take for instance the sequence:

```
\special{pa 2000 1000}
\special{pa 1000 2000}
\special{pa 0000 1000}
\special{pa 1000 0000}
\special{pa 2000 1000}
\special{sp}
```

One would expect that this code produced a closed circle, but the curve that comes out using b-splines is far from round. We can however savely asume that the arc producing specials will be used for drawing circle fragments, while the path specials will be used for arbitraty curves. And for b-splines to produce nice curves, one will often use many points to get the desired results. Therefore, using the METAPOST Bézier curves will certainly produce similar and even better graphics, except in those rare cases where one uses delinberately the not that accurate features of b-splines. Hereby the user is warned.

24
```
\protect
```

25
```
\endinput
```