pspicture.sty

D. P. Carlisle

16 June 1992

1 Introduction

`pspicture` is a re-implementation, and extension of, LaTeX's `picture` environment, using PostScript `\special`'s. This has several advantages, mainly that lines of arbitrary slope and thickness may be specified, and there is no limit on the size of the circles that may be drawn[1].

One disadvantage is that the picture can no longer be previewed on a `dvi` previewer, such as `xdvi`. To help with this problem, a companion style option, `texpicture`, may be used while developing a document, this uses the standard picture commands as much as possible, and silently omits any picture objects that can not be drawn with standard LaTeX.

A second disadvantage, is that a `dvi` file produced with `pspicture` will contain embedded `\special` commands. These commands will only work with the driver program for which they were intended. This makes the `dvi` file less portable. `pspicture` will by default use `\special`'s set up for Rokicki's `dvips` program, although it should be easy to modify the code to work with other PostScript drivers. A DocStrip option for a version of `dvi2ps` is included with this distribution.

1.1 Commands Available

`\circle` `\circle*` Use as described in the LaTeX book but with no maximum diameter. The thickness of the circle is altered by the `\linethickness` command. The size of the circle produced by `\circle*` is not affected by `\linethickness`, so it is not the same as 'filling in' the circle drawn by `\circle`.

`\oval` Use as described in the LaTeX book, but as there is no maximum diameter for the circular arcs, the oval (in the absence of the optional `[tr]` etc) always consists of two semi-circular arcs joined by a pair of parallel lines. To obtain a 'rectangle

[1] There is a certain amount of overlap between this style option and the widely available `eepic` option. However when I wrote the first version of this, in 1989, I was not aware of `eepic`, and `pspicture` has been reasonably popular in Manchester, even though `epic` and `eepic` have been installed.

with rounded corners' the oval command has a second optional argument (given first !).

`\oval[20](100,200)[t]`

Produces the top half of an oval with quarter circles of radius 20*unitlength. If unitlength = 1pt then this is equivalent to the standard oval command. In general `\oval[R](x,y)` uses circular arcs of radius $\min(R, x/2, y/2)$.

`\line` `\vector` Use as described in the LaTeX book but with no restriction on the available slopes. The thickness of a sloping line is altered by the `\linethickness` command.

`\Line` `\Vector` New forms of the line and vector commands.

`\put(x1,y1){\Line(x2,y2)}`

produces a line from (x1,y1) to (x1+x2,y1+y2) and similarly for `\Vector`.

`\Curve` Like `\Line` except that it produce a curve!

`\put(x1,y1){\Curve(x2,y2){m}}`

produces a curve from (x1,y1) to (x1+x2,y1+y2). the amount of curvature is controlled by m but try 1 or -1 first. m does not have to be an integer. Negative numbers curve the opposite way to positive numbers.

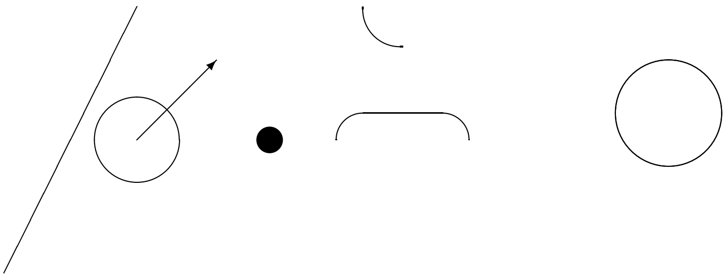
`\thinlines` `\thicklines` `\linethickness` These commands alter the thickness of **all** lines including slanted lines and circular arcs.

`\arrowlength` A new command which specifies the size of the arrowhead drawn by the `\vector` and `\Vector` commands. Like `\linethickness` it does not get multiplied by `\unitlength`. At present the arrowhead is triangular. If a head with curved sides more like the standard LaTeX head is required the definition of `!A` in pspicture.ps should be altered.

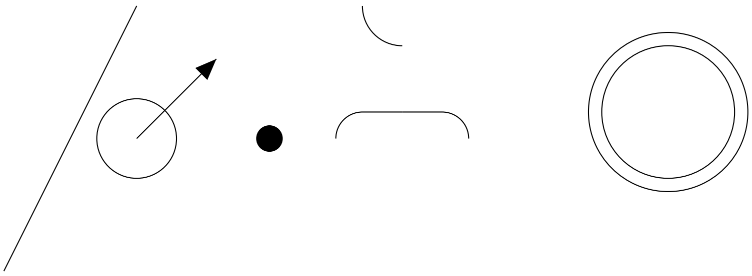
Other `picture` mode commands are not altered by this style, and so may be used, just as described in the LaTeX book. These include: `\put`, `\multiput`, `\makebox`, `\framebox`, `dashbox` and `\shortstack`.

2 Examples

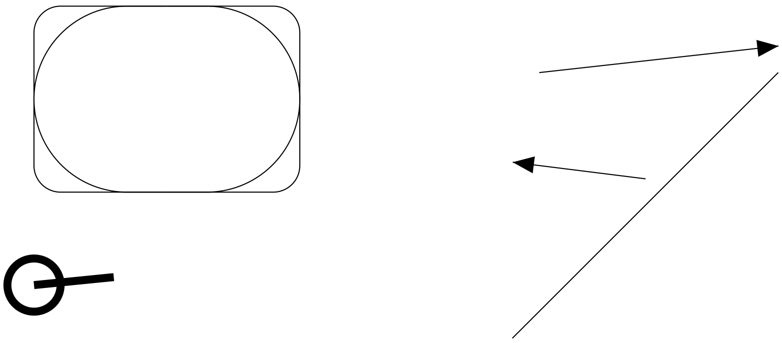
A picture built with LaTeX's line and circle fonts.



The same picture built with PostScript `\special`'s.



Some extra features not available using the standard picture mode.



3 pspicture.sty

```
⟨*pssty⟩
```

First we set up the code that is specific to the driver program that is being used. If the driver can incorporate a header file, define `\PS@header` appropriately, `\PS@special` should expand to the format for inline PostScript code. The driver should protect this code with a (g)save (g)restore pair. dvips is treated specially so that it will be the default driver if this file is used without being stripped. If you find definitions of these macros which work for the driver you use, email me, and I will include them in the next release.

`\PS@header` Code to include the PostScript header.

```
⟨*dvips⟩
      \def\PS@header#1{\special{header=#1}}
⟨/dvips⟩
⟨dvi2ps⟩\def\PS@header#1{\typeout{Print with the option -i #1}}
\PS@header{pspicture.ps}
```

`\PS@special` The format of the `\special` command for inline PostScript.

```
⟨*dvips⟩
      \def\PS@special#1{\special{"#1}}
⟨/dvips⟩
⟨dvi2ps⟩\def\PS@special#1{\special{pstext="#1"}}
```

`\strippt` Strip the final 'pt' off the string returned by `\the`.

```
{\catcode`t=12\catcode`p=12\gdef\noPT#1pt{#1}}
\def\strippt#1{\expandafter\noPT\the#1\space}
```

`\@circle` Internal name for `\circle`.

```
\def\@circle#1{%
  \@tempdimb #1\unitlength
  \PS@special{%
      \strippt\@wholewidth
      \strippt\@tempdimb
      !C}}
```

`\@dot` Internal name for `\circle*`.

```
\def\@dot#1{%
  \@tempdimb #1\unitlength
  \PS@special{%
      \strippt\@tempdimb
      !D}}
```

`\line` Line with a LaTeX style slope specification.

```
\def\line(#1,#2)#3{%
  \@linelen=#3\unitlength
  \PS@special{%
```

```
        \strippt\@wholewidth
        #1
        #2
        \strippt\@linelen
        !L}}
```

`\vector` Line and arrow head with a LaTeX style slope specification.

```
\def\vector(#1,#2)#3{%
  \@linelen=#3\unitlength
  \PS@special{%
        \strippt\@arrowlength
        \strippt\@wholewidth
        #1
        #2
        \strippt\@linelen
        !V}}
```

`\oval` If no optional argument appears, use a default of maximum radius of TeX's maximum length.

```
\def\oval{%
  \@ifnextchar[%
   {\@ov@l}%
   {\count@=\maxdimen \divide\count@ by \unitlength \@ov@l[\count@]}}
```

`\@ov@l` Look for an optional `tlbr` argument.

```
\def\@ov@l[#1](#2,#3){%
  \@ifnextchar[{\@oval[#1](#2,#3)}{\@oval[#1](#2,#3)[]}}%
```

`\@oval` The PostScript version of the `\oval` command will print each quarter of the oval separately, each quarter will only be printed if the appropriate argument is 1. An optional argument of `t` causes the arguments for the two bottom quarters to be set to 0, similarly, `r` causes the two left quarters to be set to 0. Thus an argument [tr] will set the bottom and left quarters to 0, resulting in only the top right quarter being printed.

```
\def\@oval[#1](#2,#3)[#4]{\begingroup
  \@tempdimb #1\unitlength
  \@ovxx #2\unitlength
  \@ovyy #3\unitlength
  \def\r{\def\TL{0 }\def\BL{0 }}%
  \def\l{\def\TR{0 }\def\BR{0 }}%
  \def\t{\def\BL{0 }\def\BR{0 }}%
  \def\b{\def\TL{0 }\def\TR{0 }}%
  \def\TL{1 }\def\BL{1 }\def\TR{1 }\def\BR{1 }%
  \@tfor\@tempa :=#4\do{\csname\@tempa\endcsname}%
  \PS@special{%
        \BR\BL\TR\TL
        \strippt\@wholewidth
```

```
        \strippt\@tempdimb
        \strippt\@ovxx
        \strippt\@ovyy
        !O}%
  \endgroup}
```

\Line

New style \Line command.

```
\def\Line(#1,#2){%
  \@ovxx #1\unitlength
  \@ovyy #2\unitlength
  \PS@special{%
        \strippt\@wholewidth
        \strippt\@ovxx
        \strippt\@ovyy
        !L2}}
```

\Curve

Not particularly good, but it will do for now.

```
\def\Curve(#1,#2)#3{%
  \@ovxx #1\unitlength
  \@ovyy #2\unitlength
  \PS@special{%
        \strippt\@wholewidth
        \strippt\@ovxx
        \strippt\@ovyy
        #3
        !C2}}
```

\Vector

New style \Vector command.

```
\def\Vector(#1,#2){%
  \@ovxx #1\unitlength
  \@ovyy #2\unitlength
  \PS@special{%
        \strippt\@arrowlength
        \strippt\@wholewidth
        \strippt\@ovxx
        \strippt\@ovyy
        !V2}}
```

\@arrowlength

Length of an arrow head.

```
\newdimen\@arrowlength
```

\arrowlength

Set the length of an arrow head.

```
\def\arrowlength#1{\@arrowlength #1}
\arrowlength{8pt}
```

If this file is used as a .sty file without being stripped, we want to stop here. The \endinput must not be at the beginning of the line, or DocStrip will stop here as well!.

```
        \endinput
⟨/pssty⟩
```

4 texpicture.sty

A dummy style file so that documents using `pspicture.sty` can be previewed or printed (as much as possible) using a dvi (not PostScript) previewer or printer driver.

Just change '`pspicture`' to '`texpicture`' in the `\documentstyle` options list.

```
⟨*texsty⟩
```

```
\@warning{texpicture.sty in operation:^^J\@spaces
LaTeX document with pspicture.sty before printing}
```

`\Line` `\Vector` `\arrowlength` `\Curve`

Define all these new commands to silently gobble their arguments.

```
\def\Line(#1,#2){}
\def\Vector(#1,#2){}
\def\arrowlength#1{}
\def\Curve(#1,#2)#3{}
```

`\@badlinearg`

If a vector or line is called with a slope specification that is not allowed by standard LaTeX, `\@badlinearg` is called to produce the error message. We do not want to see these errors, so:

```
\def\@badlinearg{}
```

`\oval` `\@@v@l` `\@@vv@l`

Give the standard `\oval` command another optional argument (which will be ignored), to match the extra argument defined in `pspicture.sty`.

```
\let\@@v@l\oval
\def\@@@v@l[#1]{\@@v@l}
\def\oval{\@ifnextchar[{\@@@v@l}{\@@v@l}}
```

```
⟨/texsty⟩
```

5 pspicture.ps

The PostScript header file for use with `pspicture.sty`. Probably this should use the PostScript dictionary mechanism, to keep identifiers local to this package, but for now, just give them names begining with `!`.

```
⟨*ps⟩
```

`!BP`

PostScript uses TeX's `bp`, that is 1/72 of an inch, not TeX's `pt`, 1/72.27 of an inch, but it is inconvenient to get TeX to output in `bp`, so we need to scale the PostScript.

```
/!BP{
```

```
72 72.27 div dup scale
}def
```

!A Arrow head:
⟨*arrow length*⟩ `!A`

```
/!A{
newpath
0 0 moveto
dup neg dup .4 mul rlineto
.8 mul 0 exch  rlineto
closepath
fill
} def
```

!V `\vector(`⟨x⟩`,`⟨y⟩`)`
⟨*arrow length*⟩ ⟨*line width*⟩ ⟨x⟩ ⟨y⟩ ⟨*len*unitlength*⟩ `!V`

```
/!V{
!BP
/!X exch def
/!y exch def
/!x exch def
newpath
0 0 moveto
!x 0 eq {0  !y 0 lt {!X neg}{!X} ifelse}
       {!x 0 lt {!X neg}{!X}ifelse  !X !y mul !x abs div} ifelse
lineto
setlinewidth % @wholewidth
currentpoint
stroke
translate
!y !x atan
rotate
!A % @arrowlength
}def
```

!L `\line(`⟨x⟩`,`⟨y⟩`)`
⟨*arrow length*⟩ ⟨*line width*⟩ ⟨x⟩ ⟨y⟩ ⟨*len*unitlength*⟩ `!L`

```
/!L{
!BP
/!X exch def
/!y exch def
/!x exch def
newpath
0 0 moveto
!x 0 eq {0  !y 0 lt {!X neg}{!X} ifelse}
       {!x 0 lt {!X neg}{!X}ifelse  !X !y mul !x abs div} ifelse
lineto
```

```
  setlinewidth % @wholewidth
  stroke
  }def
```

!C \circle{⟨*diam*⟩}
⟨*line width*⟩ ⟨*diam*unitlength*⟩ !C

```
/!C{
  !BP
  0 0 3 2 roll
  2 div 0 360 arc
  setlinewidth % @wholewidth
  stroke
  }def
```

!D \circle*{⟨*diam*⟩}
⟨*diam*unitlength*⟩ !D

```
/!D{
  !BP
  0 0 3 2 roll
  2 div 0 360 arc fill
  }def
```

!O \oval[⟨*max-radius*⟩](⟨*x*⟩,⟨*y*⟩)[⟨*tlbr*⟩]
⟨*br*⟩⟨*bl*⟩⟨*tr*⟩⟨*tl*⟩
⟨*line width*⟩ ⟨*max-radius*unitlength*⟩ ⟨*x*unitlength*⟩ ⟨*y*unitlength*⟩ !O
The first four arguments should be either 0, denoting that that quarter should not be drawn, or 1, to draw a quarter oval.

```
/!O{
  !BP
  /!y exch 2 div def
  /!x exch 2 div def
  /!r exch !x !y
```

Ghostscript appears to have a `min` operator, so the following 2 lines could be coded as `min min`, but it's not in the Ref. Manual, and it doesn't work on my printer!

```
    2 copy gt {exch} if pop
    2 copy gt {exch} if pop
      def
  setlinewidth % @wholewidth
  1 eq
  {newpath
   !x neg 0 moveto
   !x neg !y 0 !y !r arcto 4 {pop} repeat
   0 !y lineto
   stroke}if
  1 eq
  {newpath
```

```
  !x  0 moveto
  !x  !y 0 !y !r arcto 4 {pop} repeat
  0 !y lineto
  stroke}if
 1 eq
 {newpath
  !x neg 0 moveto
  !x neg !y neg 0 !y neg  !r arcto 4 {pop} repeat
  0 !y neg lineto
  stroke}if
 1 eq
 {newpath
  !x  0 moveto
  !x  !y neg 0 !y neg !r arcto 4 {pop} repeat
  0 !y neg lineto
  stroke}if
 }def
```

!V2 `\Vector(`$\langle x\rangle$`,`$\langle y\rangle$`)`

$\langle$*arrow length*$\rangle$ $\langle$*line width*$\rangle$ $\langle$*x*unitlength*$\rangle$ $\langle$*y*unitlength*$\rangle$ `!V2`

```
/!V2{
 !BP
 2 copy exch
 atan
 /a exch def
 2 copy
 newpath
 0 0 moveto
 lineto          % <x*unitlength> <y*unitlength>
 3 2 roll
 setlinewidth % @wholewidth
 stroke
 translate       % <x*unitlength> <y*unitlength>
 a rotate
 !A % @arrowlength
 }def
```

!L2 `\Line(`$\langle x\rangle$`,`$\langle y\rangle$`)`

$\langle$*line width*$\rangle$ $\langle$*x*unitlength*$\rangle$ $\langle$*y*unitlength*$\rangle$ `!L2`

```
/!L2{
 !BP
 newpath
 0 0 moveto
 lineto          % <x*unitlength> <y*unitlength>
 setlinewidth % @wholewidth
 stroke
 }def
```

`!C2` `\Curve(`$\langle x\rangle$`,`$\langle y\rangle$`){`$\langle\pm\rangle$`}`

$\langle$*line width*$\rangle$ $\langle$*x*unitlength*$\rangle$ $\langle$*y*unitlength*$\rangle$ $\langle\pm\rangle$ `!C2`

```
/!C2{
  !BP
  /!s exch def
  /!y exch def
  /!x exch def
  newpath
  0 0 moveto
  0 0
  !x 2 div !y 10 div !s mul add
  !y 2 div  !x 10 div  !s mul sub
  !x !y
  curveto
  setlinewidth % @wholewidth
  stroke
  }def
```

$\langle$/ps$\rangle$