ltpictur.dtx

Johannes Braams David Carlisle Alan Jeffrey
Leslie Lamport Frank Mittelbach Chris Rowley
Rainer Schöpf

2003/12/30

1 Picture Mode

Picture mode commands. In addition to the commands available in LaTeX2.09, This section adds the new `\qbezier` command for drawing curves.

`\qbezier` `\qbezier[`⟨*N*⟩`](`⟨*AX,AY*⟩`)(`⟨*BX,BY*⟩`)(`⟨*CX,CY*⟩`)` plots a quadratic Bezier curve from `(`⟨*AX,AY*⟩`)` to `(`⟨*CX,CY*⟩`)`, with `(`⟨*BX,BY*⟩`)` as the third Bezier point, using $N + 1$ points equally spaced parametrically. If $N = 0$ (the default value), then a sufficient number of points are used to draw a connected curve–except that at most `\qbeziermax` + 1 points are drawn. A "point" is a square of side `\@wholewidth`.

`\bezier` In addition, to be compatible with the old `bezier` package, a variant of this command, `\bezier`, is defined, in which the first argument is not optional.

```
\unitlength     = value of dimension argument
\@wholewidth    = current line width
\@halfwidth     = half of current line width
\@linefnt       = font for drawing lines
\@circlefnt     = font for drawing circles

\linethickness{DIM} : Sets the width of horizontal and vertical lines
    in a picture to DIM.  Does not change width of slanted lines
    or circles.  Width of all lines reset by \thinlines and
    \thicklines

\picture(XSIZE,YSIZE)(XORG,YORG)
  BEGIN
    \@picht :=L YSIZE * \unitlength
    box \@picbox :=
         \hb@xt@ XSIZE * \unitlength
           {\hskip -XORG * \unitlength
            \lower YORG * \unitlength
            \hbox{
            \ignorespaces    %% added 13 June 89
  END

\endpicture ==
```

```
   BEGIN
                    } \hss }
                    heigth of \@picbox := \@picht
                    depth  of \@picbox := 0
                    \mbox{\box\@picbox}   %% change 26 Aug 91
   END

 \put(X, Y){OBJ} ==
   BEGIN
     \@killglue
     \raise Y * \unitlength  \hb@xt@ 0pt { \hskip X * \unitlength
                                           OBJ \hss
}
     \ignorespaces
   END

 \multiput(X,Y)(DELX,DELY){N}{OBJ} ==
   BEGIN
    \@killglue
    \@multicnt := N
    \@xdim  := X * \unitlength
    \@ydim  := Y * \unitlength
    while \@multicnt > 0
      do \raise \@ydim \hb@xt@ 0pt { \hskip \@xdim
                                           OBJ \hss   }
         \@multicnt := \@multicnt - 1
         \@xdim     := \@xdim + DELX * \unitlength
         \@ydim     := \@ydim + DELY * \unitlength
      od
    \ignorespaces
   END
```

`\shortstack`[POS]{TEXT} : Makes a `\vbox` containing TEXT stacked as a one-column array, positioned l, r or c as indicated by POS.

The '2ekernel' code ensures that a `\usepackage{autopict}` is essentially ignored if a 'full' format is being used that has picture mode already in the format.

```
1 ⟨2ekernel⟩\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion
```

`\@wholewidth`
`\@halfwidth`

```
2 ⟨*2ekernel | autoload⟩
3 \newdimen\@wholewidth
4 \newdimen\@halfwidth
```

`\unitlength`

```
5 \newdimen\unitlength \unitlength =1pt
```

`\@picbox`
`\@picht`

```
6 \newbox\@picbox
7 \newdimen\@picht
8 ⟨/2ekernel | autoload⟩
```

\picture #1 should be white space.

\pictur@ #1 should be a ((eating any white space before the bracket),

```
⟨*2ekernel | def⟩
\long\gdef\picture#1{\pictur@#1}
\gdef\pictur@(#1){%
  \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}
⟨/2ekernel | def⟩
⟨*autoload⟩
\def\pictur@{\@autoload{pict}}
\def\picture{\pictur@\picture}
⟨/autoload⟩
```

\@picture

```
⟨*2ekernel | def⟩
\gdef\@picture(#1,#2)(#3,#4){%
  \@picht#2\unitlength
  \setbox\@picbox\hb@xt@#1\unitlength\bgroup
    \hskip -#3\unitlength
    \lower #4\unitlength\hbox\bgroup
      \ignorespaces}
```

\endpicture

```
\gdef\endpicture{%
  \egroup\hss\egroup
    \ht\@picbox\@picht\dp\@picbox\z@
    \mbox{\box\@picbox}}
```

In the definitions of \put and \multiput, \hskip was replaced by \kern just in case arg #3 = "plus". (Bug detected by Don Knuth. changed 20 Jul 87).

```
\long\gdef\put(#1,#2)#3{%
  \@killglue\raise#2\unitlength
  \hb@xt@\z@{\kern#1\unitlength #3\hss}%
  \ignorespaces}
```

\multiput #3 had better be a (.

```
\gdef\multiput(#1,#2)#3{%
  \@xdim #1\unitlength
  \@ydim #2\unitlength
   \@multiput(}
```

\multiput

```
\long\gdef\@multiput(#1,#2)#3#4{%
  \@killglue\@multicnt #3\relax
  \@whilenum \@multicnt >\z@\do
    {\raise\@ydim\hb@xt@\z@{\kern\@xdim #4\hss}%
     \advance\@multicnt\m@ne
     \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
  \ignorespaces}
```

\@killglue

```
\gdef\@killglue{\unskip\@whiledim \lastskip >\z@\do{\unskip}}
⟨/2ekernel | def⟩
```

```
\thinlines
\thicklines
⟨*2ekernel | def⟩
\gdef\thinlines{\let\@linefnt\tenln \let\@circlefnt\tencirc
  \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
\gdef\thicklines{\let\@linefnt\tenlnw \let\@circlefnt\tencircw
  \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}
⟨/2ekernel | def⟩
⟨*autoload⟩
\def\thinlines{\pictur@\thinlines}
\def\thicklines{\pictur@\thicklines}
⟨/autoload⟩

\linethickness
⟨*2ekernel | def⟩
\gdef\linethickness#1{\@wholewidth #1\relax \@halfwidth .5\@wholewidth}
⟨/2ekernel | def⟩
⟨*autoload⟩
\def\linethickness{\pictur@\linethickness}
⟨/autoload⟩

\ishortstack
⟨*2ekernel | def⟩
\gdef\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}

\@ishortstack
\gdef\@shortstack[#1]{%
  \leavevmode
  \vbox\bgroup
    \baselineskip-\p@\lineskip 3\p@
    \let\mb@l\hss\let\mb@r\hss
    \expandafter\let\csname mb@#1\endcsname\relax
    \let\\\@stackcr
    \@ishortstack}

\@ishortstack
\gdef\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\egroup}

\@stackcr
\@ixstackcr
\gdef\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
\gdef\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}

\@istackcr
\gdef\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}
```

```
\line(X,Y){LEN} ==
BEGIN
 \@xarg    := X
 \@yarg    := Y
 \@linelen := LEN * \unitlength
 if \@xarg = 0
    then \@vline
    else if \@yarg = 0
```

```
            then \@hline
            else \@sline
          if
 if
END

\@sline ==
 BEGIN
   if \@xarg < 0
     then @negarg := T
          \@xarg  := -\@xarg
          \@yyarg := -\@yarg
     else @negarg := F
          \@yyarg := \@yarg
   fi
   \@tempcnta := |\@yyarg|
   if \@tempcnta > 6
     then error: 'LATEX ERROR: Illegal \line or \vector argument.'
          \@tempcnta := 0
   fi
   \box\@linechar := \hbox{\@linefnt \@getlinechar(\@xarg,\@yyarg)
}
    if \@yarg > 0 then \@upordown = \raise
                         \@clnht := 0
                  else \@upordown = \lower
                       \@clnht := height of \box\@linechar
    fi
    \@clnwd  := width of \box\@linechar
    if @negarg
      then \hskip - width of \box\@linechar
           \reserved@a == \hskip - 2* width of box \@linechar
      else \reserved@a == \relax
    fi
  %% Put out integral number of line segments
    while \@clnwd <  \@linelen
      do  \@upordown \@clnht \copy\@linechar
          \reserved@a
          \@clnht := \@clnht + ht of \box\@linechar
          \@clnwd := \@clnwd + width of \box\@linechar
      od

  %% Put out last segment
    \@clnht := \@clnht - height of \box\@linechar
    \@clnwd := \@clnwd - width of \box\@linechar
    \@tempdima   := \@linelen - \@clnwd
    \@tempdimb   := \@tempdima - width of \box\@linechar
    if @negarg  then \hskip -\@tempdimb
                else \hskip  \@tempdimb
    fi
    \@tempdima   := 1000 * \@tempdima
```

```
    \@tempcnta   := \@tempdima / width of \box\@linechar
    \@tempdima   := (\@tempcnta * ht of \box\@linechar)/1000
    \@clnht := \@clnht + \@tempdima
    if \@linelen < width of box\@linechar
        then \hskip width of box\@linechar
        else \hbox{\@upordown \@clnht \copy\@linechar}
    fi
END

\@hline ==
  BEGIN
    if \@xarg < 0 then  \hskip -\@linelen \fi
    \vrule height \@halfwidth depth \@halfwidth width \@linelen
    if \@xarg < 0 then  \hskip -\@linelen \fi
 END

\@vline == if \@yarg < 0 \@downline else \@upline  fi


\@getlinechar(X,Y) ==
  BEGIN
    \@tempcnta := 8*X - 9
    if Y > 0
      then \@tempcnta := \@tempcnta + Y
      else \@tempcnta := \@tempcnta - Y + 64
    fi
    \char\@tempcnta
  END

\vector(X,Y){LEN} ==
BEGIN
 \@xarg    := X
 \@yarg    := Y
 \@linelen := LEN * \unitlength
 if \@xarg = 0
    then \@vvector
    else if \@yarg = 0
           then \@hvector
           else \@svector
         if
 if
END

\@hvector ==
  BEGIN
    \@hline
    {\@linefnt if \@xarg < 0 then  \@getlarrow(1,0)
                             else  \@getrarrow(1,0)
                fi}
  END
```

```
\@vvector == if \@yarg < 0 \@downvector else \@upvector  fi

\@svector ==
 BEGIN
  \@sline
  \@tempcnta := |\@yarg|
    if  \@tempcnta < 5
        then  \hskip - width of \box\@linechar
              \@upordown \@clnht \hbox
                         {\@linefnt
                          if @negarg then \@getlarrow(\@xarg,\@yyarg)
                                     else \@getrarrow(\@xarg,\@yyarg)
                          fi }
        else  error: 'LATEX ERROR: Illegal \line or \vector argument.'
    fi
 END

\@getlarrow(X,Y) ==
 BEGIN
  if Y = 0
    then \@tempcnta := '33
    else \@tempcnta := 16 * X  -  9
         \@tempcntb := 2 * Y
         if \@tempcntb > 0
           then  \@tempcnta := \@tempcnta  +  \@tempcntb
           else  \@tempcnta := \@tempcnta  -  \@tempcntb +  64
         fi
  fi
  \char\@tempcnta
 END

\@getrarrow(X,Y) ==
 BEGIN
  \@tempcntb := |Y|
  case of \@tempcntb
    0 : \@tempcnta := '55
    1 : if X < 3
          then \@tempcnta :=  24*X - 6
          else if X = 3
                 then \@tempcnta := 49
                 else \@tempcnta := 58  fi
        fi
    2 : if X < 3
          then \@tempcnta :=  24*X - 3
          else \@tempcnta := 51      % X must = 3
        fi
    3 : \@tempcnta := 16*X - 2
    4 : \@tempcnta := 16*X + 7
  endcase
```

```
    if Y < 0
      then \@tempcnta := \@tempcnta + 64
    fi
    \char\@tempcnta
   END
```

\if@negarg

```
\newif\if@negarg
```

\line

```
\gdef\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@linelen #3\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vline
      \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
    \fi
  \fi}
```

\@sline

```
\gdef\@sline{%
  \ifnum\@xarg<\z@ \@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
  \else \@negargfalse \@yyarg \@yarg \fi
\ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
\ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
\ifnum\@xarg>6 \@badlinearg\@xarg \@ne \fi
\setbox\@linechar\hbox{\@linefnt\@getlinechar(\@xarg,\@yyarg)}%
```

If we have something like `\line(5,5){30}` the `\@linechar` will not contain a char and later on we will end in an infinite loop. So we check the width of the box and put in something as an emergency fix if necessary.

```
\ifdim\wd\@linechar=\z@
   \setbox\@linechar\hbox{.}%
   \@badlinearg
\fi
\ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
\@clnwd \wd\@linechar
\if@negarg
  \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
\else
     \let\reserved@a\relax
\fi
\@whiledim \@clnwd <\@linelen \do
  {\@upordown\@clnht\copy\@linechar
   \reserved@a
   \advance\@clnht \ht\@linechar
   \advance\@clnwd \wd\@linechar}%
\advance\@clnht -\ht\@linechar
\advance\@clnwd -\wd\@linechar
\@tempdima\@linelen\advance\@tempdima -\@clnwd
\@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
\if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
\multiply\@tempdima \@m
\@tempcnta \@tempdima
```

```
\@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
\@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
\divide\@tempdima \@m
\advance\@clnht \@tempdima
\ifdim \@linelen <\wd\@linechar
   \hskip \wd\@linechar
```

Warn if line gets so short that it can't be printed. But don't warn if it is exactly zero since that was probably deliberate (e.g., to get a vector head only).

```
   \ifdim \@linelen = \z@
   \else
     \@picture@warn
   \fi
   \else\@upordown\@clnht\copy\@linechar\fi}
```

\@hline

```
\gdef\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
\vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
\ifnum \@xarg <\z@ \hskip -\@linelen \fi}
```

\getlinechar

```
\gdef\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
  \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
  \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
  \char\@tempcnta}
```

\vector

```
\gdef\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
  \ifnum\@tempcnta<5\relax
  \@linelen #3\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vvector
      \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
    \fi
  \fi
  \else\@badlinearg\fi}
```

\@hvector

```
\gdef\@hvector{\@hline\hb@xt@\z@{\@linefnt
 \ifnum \@xarg <\z@ \@getlarrow(1,0)\hss\else
    \hss\@getrarrow(1,0)\fi}}
```

\@vvector

```
\gdef\@vvector{\ifnum \@yarg <\z@ \@downvector \else \@upvector \fi}
```

\@svector

```
\gdef\@svector{\@sline
  \@tempcnta\@yarg \ifnum\@tempcnta <\z@ \@tempcnta -\@tempcnta\fi
  \ifnum\@tempcnta <5%
    \hskip -\wd\@linechar
    \@upordown\@clnht \hbox{\@linefnt  \if@negarg
    \@getlarrow(\@xarg,\@yyarg)\else \@getrarrow(\@xarg,\@yyarg)\fi}%
  \else\@badlinearg\fi}
```

\@getlarrow

```
\gdef\@getlarrow(#1,#2){\ifnum #2=\z@ \@tempcnta 27 % '33
  \else
  \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n
  \advance\@tempcnta -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
  \ifnum \@tempcntb >\z@ \advance\@tempcnta \@tempcntb
  \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
  \fi\fi\char\@tempcnta}
```

\@getrarrow

```
\gdef\@getrarrow(#1,#2){\@tempcntb #2\relax
\ifnum\@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
\ifcase \@tempcntb\relax \@tempcnta 45  % '55
\or
\ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
\else\@tempcnta 58 \fi\fi\or
\ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta -\tw@ \else
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
\char\@tempcnta}
```

\@vline

```
\gdef\@vline{\ifnum \@yarg <\z@ \@downline \else \@upline\fi}
```

\@upline

```
\gdef\@upline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \@linelen \@depth \z@\hss}}
```

\@downline

```
\gdef\@downline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \z@ \@depth \@linelen \hss}}
```

\@upvector

```
\gdef\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}%  % '66
  \raise \@linelen \hb@xt@\z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}
```

\@downvector

```
\gdef\@downvector{\@downline\lower \@linelen
      \hb@xt@\z@{\@linefnt\char 63  % '77
      \hss}}
```

\dashbox{D}(X,Y) ==
BEGIN
leave vertical mode
\hb@xt@ 0pt {
\baselineskip := 0pt
\lineskip := 0pt

```
%% HORIZONTAL DASHES
      \@dashdim := X * \unitlength
      \@dashcnt := \@dashdim + 200 % to prevent roundoff error
      \@dashdim := D * \unitlength
      \@dashcnt := \@dashcnt / \@dashdim
      if \@dashcnt is odd
        then \@dashdim := 0pt
             \@dashcnt := (\@dashcnt + 1) / 2
        else \@dashdim := \@dashdim / 2
             \@dashcnt := \@dashcnt / 2 - 1
             \box\@dashbox    := \hbox{\vrule height \@halfwidth
                                  depth \@halfwidth width \@dashdim}
             \put(0,0){\copy\@dashbox}
             \put(0,Y){\copy\@dashbox}
             \put(X,0){\hskip -\@dashdim\copy\@dashbox}
             \put(X,Y){\hskip -\@dashdim\box\@dashbox}
             \@dashdim := 3 * \@dashdim
      fi
      \box\@dashbox := \hbox{\vrule height \@halfwidth
                         depth \@halfwidth width D * \unitlength
                         \hskip D * \unitlength}
      \@tempcnta := 0
      \put(0,0){\hskip \@dashdim
               while \@tempcnta < \@dascnt
                 do \copy\@dashbox
                    \@tempcnta := \@tempcnta + 1
                 od
               }
      \@tempcnta := 0
      put(0,Y){\hskip \@dashdim
               while \@tempcnta < \@dascnt
                 do \copy\@dashbox
                    \@tempcnta := \@tempcnta + 1
                 od
               }

%% vertical dashes
      \@dashdim := Y * \unitlength
      \@dashcnt := \@dashdim + 200 % to prevent roundoff error
      \@dashdim := D * \unitlength
      \@dashcnt := \@dashcnt / \@dashdim
      if \@dashcnt is odd
        then \@dashdim := 0pt
             \@dashcnt := (\@dashcnt + 1) / 2
        else \@dashdim := \@dashdim / 2
             \@dashcnt := \@dashcnt / 2 - 1
             \box\@dashbox    := \hbox{\hskip -\@halfwidth
                                       \vrule width \@wholewidth
                                         height \@dashdim  }
             \put(0,0){\copy\@dashbox}
```

```
                \put(X,0){\copy\@dashbox}
                \put(0,Y){\lower\@dashdim\copy\@dashbox}
                \put(X,Y){\lower\@dashdim\copy\@dashbox}
                \@dashdim := 3 * \@dashdim
            fi
            \box\@dashbox := \hbox{\vrule width \@wholewidth
                                   height D * \unitlength       }
            \@tempcnta := 0
            put(0,0){\hskip -\halfwidth
                     \vbox{while \@tempcnta < \@dashcnt
                            do \vskip D*\unitlength
                               \copy\@dashbox
                               \@tempcnta := \@tempcnta + 1
                            od
                           \vskip \@dashdim
                          } }
            \@tempcnta := 0
            put(X,0){\hskip -\halfwidth
                     \vbox{while \@tempcnta < \@dashcnt
                            do \vskip D*\unitlength
                               \copy\@dashbox
                               \@tempcnta := \@tempcnta + 1
                            od
                           \vskip \@dashdim
                          }
                    }
        }     % END DASHES

        \@imakepicbox(X,Y)
    END
```

\dashbox

```
187 \gdef\dashbox#1(#2,#3){\leavevmode\hb@xt@\z@{\baselineskip \z@skip
188 \lineskip \z@skip
189 \@dashdim #2\unitlength
190 \@dashcnt \@dashdim \advance\@dashcnt 200
191 \@dashdim #1\unitlength\divide\@dashcnt \@dashdim
192 \ifodd\@dashcnt\@dashdim \z@
193 \advance\@dashcnt \@ne \divide\@dashcnt \tw@
194 \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
195 \advance\@dashcnt \m@ne
196 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
197 \@width \@dashdim}\put(0,0){\copy\@dashbox}%
198 \put(0,#3){\copy\@dashbox}%
199 \put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
200 \put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
201 \multiply\@dashdim \thr@@
202 \fi
203 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
204 \@width #1\unitlength\hskip #1\unitlength}\@tempcnta\z@
205 \put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
```

```
\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
\@dashdim #3\unitlength
\@dashcnt \@dashdim \advance\@dashcnt 200
\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
\ifodd\@dashcnt \@dashdim \z@
\advance\@dashcnt \@ne \divide\@dashcnt \tw@
\else
\divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \m@ne
\setbox\@dashbox\hbox{\hskip -\@halfwidth
\vrule \@width \@wholewidth
\@height \@dashdim}\put(0,0){\copy\@dashbox}%
\put(#2,0){\copy\@dashbox}%
\put(0,#3){\lower\@dashdim\copy\@dashbox}%
\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
\multiply\@dashdim \thr@@
\fi
\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
\@height #1\unitlength}\@tempcnta\z@
\put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
\do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}\@tempcnta\z@
\put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
\do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}}\@makepicbox(#2,#3)}
```

CIRCLES AND OVALS

USER COMMANDS:

`\circle{D}` : Produces the circle with the diameter as close as possible to D * `\unitlength`. `\put(X,Y){\circle{D}}` puts the circle with its center at (X,Y).

`\oval(X,Y)` : Makes an oval as round as possible that fits in the rectangle of width X * `\unitlength` and height Y * `\unitlength`. The reference point is the center.

`\oval(X,Y)[POS]` : Save as `\oval(X,Y)` except it draws only the half or quadrant of the oval indicated by POS. E.G., `\oval(X,Y)[t]` draws just the top half and `\oval(X,Y)[br]` draws just the bottom right quadrant. In all cases, the reference point is the same as the unqualified `\oval(X,Y)` command.

`\@ovvert {DELTA1} {DELTA2}` : Makes a vbox containing either the left side or the right side of the oval being constructed. The baseline will coincide with the outside bottom edge of the oval; the left side of the box will coincide with the left edge of the vertical rule. The width of the box will be `\@tempdima`.

DELTA1 and DELTA2 are added to the character number in
\@tempcnta
to get the characters for the top and bottom quarter circle pieces.

\@ovhorz : Makes an hbox containing the straight rule for either the top or the bottom of the oval being constructed. The baseline will coincide with bottom edge of the rule; the left side of the box will coincide with the left side of the oval. The width of the box will be \@ovxx.

\@getcirc {DIAM} : Sets \@tempcnta to the character number of the top-right quarter circle with the largest diameter less than or equal to DIAM. Sets \@tempboxa to an hbox containing that character. Sets \@tempdima to \wd \@tempboxa, which is the distance from the circle's left outside edge to its right inside edge. (These characters are like those described in the TeXbook, pp. 389-90.)

```
\@getcirc {DIAM} ==
  BEGIN
    \@tempcnta       := integer coercion of (DIAM + 2pt)
                                          + 2pt added 1 Nov 88
    \@tempcnta       := \@tempcnta / integer coercion of 4pt
    if \@tempcnta > 10
      then \@tempcnta := 10 fi
    if \@tempcnta > 0
      then \@tempcnta := \@tempcnta-1
      else LaTeX Warning: Oval too small.
    fi
    \@tempcnta       := 4 * \@tempcnta
    \@tempboxa       := \hbox{\@circlefnt \char \@tempcnta}
    \@tempdima       := \wd \@tempboxa
  END

\@put{X}{Y}{OBJ} ==
  BEGIN
    \raise Y \hb@xt@ 0pt{\hskip X OBJ \hss}
  END

\@oval(X,Y)[POS] ==
  BEGIN
    \begingroup
      \boxmaxdepth := \maxdimen
      @ovt := @ovb := @ovl := @ovr := true
      for all E in POS
        do  @ovE := false od
      \@ovxx      := X * \unitlength
      \@ovyy      := Y * \unitlength
```

```
       \@tempdimb := min(\@ovxx,\@ovyy)
       \@getcirc{\@tempdimb-2pt}   %% "-2pt" added 7 Dec 89
       \@ovro     := \ht \@tempboxa
       \@ovri     := \dp \@tempboxa
       \@ovdx     := \@ovxx - \@tempdima
       \@ovdx     := \@ovdx/2
       \@ovdy     := \@ovyy - \@tempdima
       \@ovdy     := \@ovyy/2
       \@circlefnt
       \@tempboxa :=
           \hbox{
                 if @ovr
                   then \@ovvert{3}{2} \kern -\@tempdima
                 fi
                 if @ovl
                   then \kern \@ovxx \@ovvert{0}{1} \kern
-\@tempdima
                        \kern -\@ovxx
                 fi
                 if @ovt
                   then \@ovhorz \kern -\@ovxx
                 fi
                 if @ovb
                   then \raise \@ovyy \@ovhorz
                 fi
                }
       \@ovdx    := \@ovdx + \@ovro
       \@ovdy    := \@ovdy + \@ovro
      \ht\@tempboxa := \dp\@tempboxa := 0
       \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
    \endgroup
   END

 \@ovvert {DELTA1} {DELTA2} ==
   BEGIN
      \vbox to \@ovyy {
                      if @ovb
                        then \@tempcntb := \@tempcnta + DELTA1
                             \kern -\@ovro
                             \hbox { \char \@tempcntb }
                             \nointerlineskip
                        else \kern \@ovri \kern \@ovdy
                      fi
                      \leaders \vrule width \@wholewidth \vfil
                      \nointerlineskip
                      if @ovt
                        then \@tempcntb := \@tempcnta + DELTA2
                             \hbox { \char \@tempcntb }
                        else \kern \@ovdy \kern \@ovro
                      fi
```

```
                    }
  END

\@ovhorz ==
  BEGIN
   \hb@xt@ \@ovxx{
                  \kern \@ovro
                  if @ovr
                    then
                    else \kern \@ovdx
                  fi
                  \leaders \hrule height \@wholewidth \hfil
                  if @ovl
                    then
                    else \kern \@ovdx
                  fi
                  \kern \@ovri
                 }
  END

\circle{DIAM} ==
  BEGIN
   \begingroup
   \boxmaxdepth := maxdimen
   \@tempdimb := DIAM *\unitlength
   if \@tempdimb > 15.5pt
     then \@getcirc{\@tempdimb}
          \@ovro := \ht \@tempboxa
          \@tempboxa := \hbox{
                  \@circlefnt
                  \@tempcnta := \@tempcnta + 2
                  \char \@tempcnta
                  \@tempcnta := \@tempcnta - 1
                  \char \@tempcnta
                  \kern -2\@tempdima
                  \@tempcnta := \@tempcnta + 2
                  \raise \@tempdima \hbox { \char \@tempcnta }
                  \raise \@tempdima \box\@tempboxa
                 }
          \ht\@tempboxa := \dp\@tempboxa := 0
          \@put{-\@ovro}{-\@ovro}{\@tempboxa}
     else
          \@circ{\@tempdimb}{96}
   fi
  \endgroup
  END

 \circle*{DIAM}  ==  \@dot{DIAM} ==
\@circ{DIAM*\unitlength}{112}
```

\@circ{DIAM}{CHAR} ==
 BEGIN
  \@tempcnta := integer coercion of (DIAM + .5pt)/1pt.
  if \@tempcnta > 15 then \@tempcnta := 15 fi
  if \@tempcnta > 1 then \@tempcnta := \@tempcnta - 1 fi
  \@tempcnta := \@tempcnta + CHAR
  \@circlefnt
  \char \@tempcnta
 END

\if@ovt
\if@ovb
\if@ovl
\if@ovr

If producing the Top Bottom Left or Right of an oval.

```
\newif\if@ovt
\newif\if@ovb
\newif\if@ovl
\newif\if@ovr
```

⟨/2ekernel | def⟩
⟨∗2ekernel | autoload⟩

\@ovxx
\@ovyy
\@ovdx
\@ovdy
\@ovro
\@ovri

```
\newdimen\@ovxx
\newdimen\@ovyy
\newdimen\@ovdx
\newdimen\@ovdy
\newdimen\@ovro
\newdimen\@ovri
```

⟨/2ekernel | autoload⟩

`\advance\@tempdima 2pt\relax` added 1 Nov 88 to fix bug in which size of drawn circle not monotonic function of argument of `\circle`, caused by different rounding for dimensions of large and small circles.

⟨∗2ekernel | def⟩

\@getcirc

```
\gdef\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
  \@tempcnta\@tempdima
  \@tempdima 4\p@ \divide\@tempcnta\@tempdima
  \ifnum \@tempcnta >10\relax
      \@picture@warn
      \@tempcnta 10\relax
  \fi
  \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
```

Warn if requirements for oval or circle can't be met.

```
    \else \@picture@warn \fi
  \multiply\@tempcnta 4\relax
  \setbox \@tempboxa \hbox{\@circlefnt
  \char \@tempcnta}\@tempdima \wd \@tempboxa}
```

\@picture@warn

Generic warning for lines, vectors (used in `\@sline`) and oval or circle (used un `\@getcirc`) are not available at right size.

```
\def\@picture@warn{\@latex@warning{%
     \string\oval, \string\circle, or \string\line\space
     size unavailable}}
```

```
  \@put
        \gdef\@put#1#2#3{\raise #2\hb@xt@\z@{\hskip #1#3\hss}}

  \oval
        \gdef\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2)[]}}

 \@oval
        \gdef\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
          \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
          \@tfor\reserved@a :=#3\do{\csname @ov\reserved@a false\endcsname}%
          \@ovxx
          #1\unitlength \@ovyy #2\unitlength
          \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx\else \@ovyy \fi
          \advance \@tempdimb -2\p@
          \@getcirc \@tempdimb
          \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
          \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
          \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
          \@circlefnt \setbox\@tempboxa
          \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
          \if@ovl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
          \if@ovt \@ovhorz \kern -\@ovxx \fi
          \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
          \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
          \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
          \endgroup}

\@ovvert
        \gdef\@ovvert#1#2{\vbox to\@ovyy{%
            \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
              \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
            \else \kern \@ovri \kern \@ovdy \fi
            \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
            \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
              \hbox{\char \@tempcntb}%
            \else \kern \@ovdy \kern \@ovro \fi}}

\@ovhorz
        \gdef\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
            \if@ovr \else \kern \@ovdx \fi
            \leaders \hrule \@height \@wholewidth \hfil
            \if@ovl \else \kern \@ovdx \fi
            \kern \@ovri}}

 \circle
        \gdef\circle{\@inmatherr\circle\@ifstar\@dot\@circle}

\@circle
        \gdef\@circle#1{%
          \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
           \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
              \@ovro\ht\@tempboxa
             \setbox\@tempboxa\hbox{\@circlefnt
```

```
      \advance\@tempcnta\tw@ \char \@tempcnta
      \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
      \advance\@tempcnta\tw@
      \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
        \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
      \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
   \else  \@circ\@tempdimb{96}\fi\endgroup}
```

`\@dot` Internal form of `\circle*`.

```
\gdef\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}
```

`\@circ`

```
\gdef\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
   \@tempcnta\@tempdima \@tempdima \p@
   \divide\@tempcnta\@tempdima
   \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
   \advance\@tempcnta #2\relax
   \@circlefnt \char\@tempcnta}
```

⟨/2ekernel | def⟩

⟨∗2ekernel | autoload⟩

`\@xarg` `\@yarg` `\@yyarg` Counters used for manipulating the 'slope' arguments.

```
\newcount\@xarg
\newcount\@yarg
\newcount\@yyarg
```

`\@multicnt` Counter used in `\multiput`, and also `\multicolumn`.

```
\newcount\@multicnt
```

`\@xdim` `\yxdim` Length registers.

```
\newdimen\@xdim
\newdimen\@ydim
```

`\@linechar` Box for holding a line segment character, for sloping lines.

```
\newbox\@linechar
```

`\@linelen` Length of the line currently being built.

```
\newdimen\@linelen
```

`\@clnwd` `\@clnht` Height and width of current line segment.

```
\newdimen\@clnwd
\newdimen\@clnht
```

`\@dashdim` `\@dashbox` `\@dashcnt` `\dashbox` internal registers.

```
\newdimen\@dashdim
\newbox\@dashbox
\newcount\@dashcnt
```

Initialization: "\thinlines"

```
\let\@linefnt\tenln
\let\@circlefnt\tencirc
\@wholewidth\fontdimen8\tenln
\@halfwidth .5\@wholewidth
```

⟨/2ekernel | autoload⟩

1.1 Curves

The new \qbezier command, based on the old \bezier defined in bezier.sty.

```
\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
  BEGIN
    IF N = 0
       THEN \@xdima := |BX - AX|
            \@xb := |CX - BX|
            \@xa := Max(\@xa, \@xb)
            \@ya := |BY - AY|
            \@yb := |CY - BY|
            \@ya := Max(\@ya, \@yb)
            @sc := Max(\@xa, \@ya)
            %% The coefficient .5 below is the degree of overlap of
            %% successive points, where 1 is no overlap and 0 is
            %% complete overlap.  A coefficient of C multiplies
            %% the number of points plotted by 1/C.
            %%
            \@xa := .5 * \@halfwidth
            @sc := @sc / \@halfwidth
            @sc := Max(@sc, qbeziermax)
        ELSE @sc := N
    @scp := @sc+1
    \@xb := 2 * (BX - AX) * \unitlength
    \@xa := ((CX-AX)*\unitlength - \@xb)/@sc
    \@yb := 2 * (BY - AY) * \unitlength
    \@ya := ((CY-AY)*\unitlength - \@yb)/@sc
    \@pictdot := square rule of width \@wholewidth
    \count@ := 0
    WHILE \count@ < @scp
      DO  \@xdim := ((\count@*\@xa + @xb) / @sc) * \count@
          \@ydim := ((\count@*\@ya + @yb) / @sc) * \count@
          plot pt with relative coords (\@xdim,\@ydim)
          \count@ := \count@+1
      OD
```

\qbeziermax The maximum number of points to plot.

⟨∗2ekernel | def⟩

⟨def⟩*\ifx\qbeziermax\@undefined*

```
\gdef\qbeziermax{500}
⟨def⟩\fi
```

In the code below, to save registers `\@a` ... are not used. Instead other registers are reused.

```
\newcounter{@sc} -> \c@multicnt
\newcounter{@scp} -> \@tempcnta
\newdimen\@xa ->  \@ovxx
\newdimen\@xb ->  \@ovdx
\newdimen\@ya ->  \@ovyy
\newdimen\@yb ->  \@ovdy
\newsavebox{\@pictdot} -> \@tempboxa
```

`\qbezier` Main user-level command to plot quadratic bezier curves. `#2` should be `(`.

```
\newcommand\qbezier[2][0]{\bezier{#1}#2}
```

`\bezier` Form of `\bezier` compatible with 2.09 `bezier.sty`, but modified to ignore spaces between its arguments. `#2` should be white space, and `#4` should be `(`.

```
\gdef\bezier#1)#2(#3)#4({\@bezier#1)(#3)(}
```

`\@bezier`

```
\gdef\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
  \ifnum #1=\z@
      \@ovxx #4\unitlength
        \advance\@ovxx -#2\unitlength
        \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
      \@ovdx #6\unitlength
        \advance\@ovdx -#4\unitlength
        \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
        \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
      \@ovyy #5\unitlength
        \advance\@ovyy -#3\unitlength
        \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
      \@ovdy #7\unitlength
        \advance\@ovdy -#5\unitlength
        \ifdim \@ovdy<\z@  \@ovdy -\@ovdy \fi
        \ifdim \@ovyy<\@ovdy \@ovyy  \@ovdy \fi
      \@multicnt
         \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
      \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
      \ifnum \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax \fi
  \else \@multicnt#1\relax \fi
  \@tempcnta\@multicnt \advance\@tempcnta\@ne
  \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
      \multiply\@ovdx \tw@
  \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
      \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
  \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
       \multiply\@ovdy \tw@
  \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
      \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
```

```
  \setbox\@tempboxa\hbox{%
            \hskip -\@halfwidth
            \vrule \@height\@halfwidth
                   \@depth \@halfwidth
                   \@width \@wholewidth}%
   \put(#2,#3){%
     \count@\z@
     \@whilenum{\count@<\@tempcnta}\do
        {\@xdim\count@\@ovxx
           \advance\@xdim\@ovdx
           \divide\@xdim\@multicnt
           \multiply\@xdim\count@
         \@ydim\count@\@ovyy
           \advance\@ydim\@ovdy
           \divide\@ydim\@multicnt
           \multiply\@ydim\count@
         \raise \@ydim
           \hb@xt@\z@{\kern\@xdim
                      \unhcopy\@tempboxa\hss}%
         \advance\count@\@ne}}}
⟨/2ekernel | def⟩
```