

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 $\text{\LaTeX}2.09$, This section adds the new `\qbezier` command for drawing curves.

`\qbezier` `\qbezier[$\langle N \rangle$]($\langle AX,AY \rangle$)($\langle BX,BY \rangle$)($\langle CX,CY \rangle$)` plots a quadratic Bezier curve from ($\langle AX,AY \rangle$) to ($\langle CX,CY \rangle$), with ($\langle BX,BY \rangle$) 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 }
height 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),
          9 <*2ekernel | def>
          10 \long\gdef\picture#1{\pictur@#1}
          11 \gdef\pictur@(#1){%
          12 \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)})}
          13 </2ekernel | def>
          14 <*autoload>
          15 \def\pictur@{\@autoload{pict}}
          16 \def\picture{\pictur@\picture}
          17 </autoload>

\@picture
          18 <*2ekernel | def>
          19 \gdef\@picture(#1,#2)(#3,#4){%
          20 \@picht#2\unitlength
          21 \setbox\@picbox\hb@xt@#1\unitlength\bgroup
          22 \hskip -#3\unitlength
          23 \lower #4\unitlength\hbox\bgroup
          24 \ignorespaces}

\endpicture
          25 \gdef\endpicture{%
          26 \egroup\hss\egroup
          27 \ht\@picbox\@picht\dp\@picbox\z@
          28 \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).

```

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

```

```

\multiput #3 had better be a (.
33 \gdef\multiput(#1,#2)#3{%
34   \@xdim #1\unitlength
35   \@ydim #2\unitlength
36   \@multiput{ }

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

\@killglue
44 \gdef\@killglue{\unskip\@whiledim \lastskip >\z@{do{\unskip}}
45 \</2ekernel| def)

\thinlines
\thicklines
46 \< *2ekernel| def)
47 \gdef\thinlines{\let\@linefnt\tenln \let\@circlefnt\tencirc
48   \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
49 \gdef\thicklines{\let\@linefnt\tenlnw \let\@circlefnt\tencircw
50   \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}
51 \</2ekernel| def)
52 \< *autoload)
53 \def\thinlines{\pictur@\thinlines}
54 \def\thicklines{\pictur@\thicklines}
55 \</autoload)

\linethickness
56 \< *2ekernel| def)
57 \gdef\linethickness#1{\@wholewidth #1\relax \@halfwidth .5\@wholewidth}
58 \</2ekernel| def)
59 \< *autoload)
60 \def\linethickness{\pictur@\linethickness}
61 \</autoload)

\ishortstack
62 \< *2ekernel| def)
63 \gdef\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}

\@ishortstack
64 \gdef\@shortstack[#1]{%
65   \leavevmode
66   \vbox\bgroup
67     \baselineskip-\p@\lineskip 3\p@
68     \let\mb@l\hss\let\mb@r\hss

```

```

69    \expandafter\let\csname mb@#1\endcsname\relax
70    \let\\\@stackcr
71    \@ishortstack}

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

\@stackcr
\@ixstackcr 73 \gdef\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
74 \gdef\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}

\@istackcr
75 \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
    fi
  fi
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
76 \newif\if@negarg

\line
77 \gdef\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
78 \@linelen #3\unitlength

```



```

79 \ifdim\@linelen<\z@\@badlinearg\else
80 \ifnum\@xarg=\z@\@vline
81 \else \ifnum\@yarg=\z@\@hline \else \@sline\fi
82 \fi
83 \fi}

```

\@sline

```

84 \gdef\@sline{%
85 \ifnum\@xarg<\z@\@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
86 \else \@negargfalse \@yyarg \@yarg \fi
87 \ifnum \@yyarg >\z@\@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
88 \ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
89 \ifnum\@xarg>6 \@badlinearg\@xarg \@ne \fi
90 \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.

```

91 \ifdim\wd\@linechar=\z@
92 \setbox\@linechar\hbox{.}%
93 \@badlinearg
94 \fi
95 \ifnum \@yarg >\z@\let\@upordown\raise \@clnht\z@
96 \else\let\@upordown\lower \@clnht \ht\@linechar\fi
97 \@clnwd \wd\@linechar
98 \if@negarg
99 \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
100 \else
101 \let\reserved@a\relax
102 \fi
103 \@whiledim \@clnwd <\@linelen \do
104 {\@upordown\@clnht\copy\@linechar
105 \reserved@a
106 \advance\@clnht \ht\@linechar
107 \advance\@clnwd \wd\@linechar}%
108 \advance\@clnht -\ht\@linechar
109 \advance\@clnwd -\wd\@linechar
110 \@tempdima\@linelen\advance\@tempdima -\@clnwd
111 \@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
112 \if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
113 \multiply\@tempdima \@m
114 \@tempcnta \@tempdima
115 \@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
116 \@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
117 \divide\@tempdima \@m
118 \advance\@clnht \@tempdima
119 \ifdim \@linelen <\wd\@linechar
120 \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).

```

121 \ifdim \@linelen = \z@
122 \else
123 \picture@warn
124 \fi
125 \else\@upordown\@clnht\copy\@linechar\fi}

```

\@hline

```

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

```

\getlinechar

```

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

```

\@vector

```

133 \gdef\@vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
134 \@tempcnta \ifnum \@xarg<\z@ -\@xarg\else\@xarg\fi
135 \ifnum \@tempcnta<5\relax
136 \@linelen #3\unitlength
137 \ifdim \@linelen<\z@ \@badlinearg\else
138 \ifnum \@xarg =\z@ \@vvector
139 \else \ifnum \@yarg =\z@ \@hvector \else \@svector\fi
140 \fi
141 \fi
142 \else\@badlinearg\fi}

```

\@hvector

```

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

```

\@vvector

```

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

```

\@svector

```

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

```

\@getlarrow

```

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

```

\@getrarrow

```

161 \gdef\@getrarrow(#1,#2){\@tempcntb #2\relax
162 \ifnum \@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
163 \ifcase \@tempcntb\relax \@tempcnta 45 % '55
164 \or
165 \ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
166 24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
167 \else\@tempcnta 58 \fi\fi\or
168 \ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
169 24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
170 \@tempcnta #1\relax\multiply\@tempcnta
171 \sixt@@n \advance\@tempcnta -\tw@ \else
172 \@tempcnta #1\relax\multiply\@tempcnta
173 \sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
174 \char\@tempcnta}

```

\@vline

```

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

```

\@upline

```

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

```

\@downline

```

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

```

\@upvector

```

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

```

\@downvector

```

184 \gdef\@downvector{\@downline\lower \@linelen
185   \hb@xt@\z@{\@linefnt\char 63 % '77
186   \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 < \@dashcnt
        do \copy\@dashbox
            \@tempcnta := \@tempcnta + 1
        od
    }
    \@tempcnta := 0
    \put(0,Y){\hskip \@dashdim
        while \@tempcnta < \@dashcnt
        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

\@makepicbox(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
206 \do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
207 \put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
208 \do{\copy\@dashbox\advance\@tempcnta \@ne }}%
209 \@dashdim #3\unitlength
210 \@dashcnt \@dashdim \advance\@dashcnt 200
211 \@dashdim #1\unitlength\divide\@dashcnt \@dashdim
212 \ifodd\@dashcnt \@dashdim \z@
213 \advance\@dashcnt \@ne \divide\@dashcnt \tw@
214 \else
215 \divide\@dashdim \tw@ \divide\@dashcnt \tw@
216 \advance\@dashcnt \m@ne
217 \setbox\@dashbox\hbox{\hskip -\@halfwidth
218 \vrule \@width \@wholewidth
219 \@height \@dashdim}\put(0,0){\copy\@dashbox}%
220 \put(#2,0){\copy\@dashbox}%
221 \put(0,#3){\lower\@dashdim\copy\@dashbox}%
222 \put(#2,#3){\lower\@dashdim\copy\@dashbox}%
223 \multiply\@dashdim \thr@@
224 \fi
225 \setbox\@dashbox\hbox{\vrule \@width \@wholewidth
226 \@height #1\unitlength}\@tempcnta\z@
227 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
228 \do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }}%
229 \vskip\@dashdim}}\@tempcnta\z@
230 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
231 \do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }}%
232 \vskip\@dashdim}}\@makepicbox(#2,#3)}

```

CIRCLES AND OVALS

USER COMMANDS:

`\circle{D}` : Produces the circle with the diameter as close as possible to $D * \text{\unitlength}$. `\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]` : Same 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      := \@ovdy/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 producing the Top Bottom Left or Right of an oval.
\if@ovb 233 \newif\if@ovt
\if@ovl 234 \newif\if@ovb
\if@ovr 235 \newif\if@ovl
236 \newif\if@ovr

```

```

237 </2ekernel | def>
238 <*2ekernel | autoload>

\@ovxx
\@ovyy 239 \newdimen\@ovxx
\@ovdx 240 \newdimen\@ovyy
\@ovdy 241 \newdimen\@ovdx
\@ovro 242 \newdimen\@ovdy
\@ovri 243 \newdimen\@ovro
244 \newdimen\@ovri

245 </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.

246 <*2ekernel | def>

\@getcirc

247 \gdef\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
248 \@tempcnta\@tempdima
249 \@tempdima 4\p@ \divide\@tempcnta\@tempdima
250 \ifnum \@tempcnta >10\relax
251 \picture@warn
252 \@tempcnta 10\relax
253 \fi
254 \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
Warn if requirements for oval or circle can't be met.
255 \else \picture@warn \fi
256 \multiply\@tempcnta 4\relax
257 \setbox \@tempboxa \hbox{\@circlelft
258 \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.
259 \def\@picture@warn{\@latex@warning{%
260 \string\oval, \string\circle, or \string\line\space
261 size unavailable}}

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

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

\@oval
264 \gdef\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
265 \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue

```

```

266 \@tfor\reserved@a :=#3\do{\csname @ov\reserved@a false\endcsname}%
267 \@ovxx
268 #1\unitlength \@ovyy #2\unitlength
269 \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx\else \@ovyy \fi
270 \advance \@tempdimb -2\p@
271 \@getcirc \@tempdimb
272 \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
273 \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
274 \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
275 \@circlefnt \setbox\@tempboxa
276 \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
277 \if@ovl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
278 \if@ovt \@ovhorz \kern -\@ovxx \fi
279 \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
280 \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
281 \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
282 \endgroup}

```

\@ovvert

```

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

```

\@ovhorz

```

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

```

\circle

```

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

```

\@circle

```

297 \gdef\@circle#1{%
298   \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
299   \ifdim \@tempdimb >15.5\p@ \getcirc\@tempdimb
300   \@ovro\ht\@tempboxa
301   \setbox\@tempboxa\hbox{\@circlefnt
302     \advance\@tempcnta\tw@ \char \@tempcnta
303     \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
304     \advance\@tempcnta\tw@
305     \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
306     \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@

```

```

307     \put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
308     \else \circ\@tempdimb{96}\fi\endgroup}

\@dot Internal form of \circle*.
309 \gdef\@dot#1{\@tempdimb #1\unitlength \circ\@tempdimb{112}}

\@circ
310 \gdef\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
311     \@tempcnta\@tempdima \@tempdima \p@
312     \divide\@tempcnta\@tempdima
313     \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
314     \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
315     \advance\@tempcnta #2\relax
316     \circlefont \char\@tempcnta}

317 </2ekernel | def>
318 <*2ekernel | autoload>

\@xarg Counters used for manipulating the ‘slope’ arguments.
\@yarg 319 \newcount\@xarg
\@yyarg 320 \newcount\@yarg
321 \newcount\@yyarg

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

\@xdim Length registers.
\@ydim 323 \newdimen\@xdim
324 \newdimen\@ydim

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

\@linelen Length of the line currently being built.
326 \newdimen\@linelen

\@clnwd Height and width of current line segment.
\@clnht 327 \newdimen\@clnwd
328 \newdimen\@clnht

\@dashdim \dashbox internal registers.
\@dashbox 329 \newdimen\@dashdim
\@dashcnt 330 \newbox\@dashbox
331 \newcount\@dashcnt

Initialization: “\thinlines”
332 \let\@linefont\tenln
333 \let\@circlefont\tencirc
334 \@wholewidth\fontdimen8\tenln
335 \@halfwidth .5\@wholewidth
336 </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.

```
337 <*2ekernel | def>
338 <def>\ifx\qbeziermax\undefined
339 \gdef\qbeziermax{500}
340 <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 (.
341 \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 (.
342 \gdef\bezier#1) #2(#3) #4({\@bezier#1)(#3){}

\@bezier

```
343 \gdef\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
344   \ifnum #1=\z@
345     \@ovxx #4\unitlength
346     \advance\@ovxx -#2\unitlength
347     \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
348     \@ovdx #6\unitlength
349     \advance\@ovdx -#4\unitlength
350     \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
351     \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
352     \@ovyy #5\unitlength
353     \advance\@ovyy -#3\unitlength
354     \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
355     \@ovdy #7\unitlength
356     \advance\@ovdy -#5\unitlength
357     \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
358     \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
359     \@multicnt
360     \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
361     \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
362     \ifnum \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax \fi
363   \else \@multicnt#1\relax \fi
364   \@tempcnta\@multicnt \advance\@tempcnta\@ne
365   \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
366   \multiply\@ovdx \tw@
367   \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
368   \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
369   \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
370   \multiply\@ovdy \tw@
371   \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
372   \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
```

```

373 \setbox\@tempboxa\hbox{%
374     \hskip -\@halfwidth
375     \vrule \@height\@halfwidth
376     \@depth \@halfwidth
377     \@width \@wholewidth}%
378 \put(#2,#3){%
379     \count@\z@
380     \@whilenum{\count@<\@tempcnta}\do
381         {\@xdim\count@\@ovxx
382             \advance\@xdim\@ovdx
383             \divide\@xdim\@multicnt
384             \multiply\@xdim\count@
385             \@ydim\count@\@ovyy
386             \advance\@ydim\@ovdy
387             \divide\@ydim\@multicnt
388             \multiply\@ydim\count@
389             \raise \@ydim
390             \hb@xt@\z@{\kern\@xdim
391                 \unhcopy\@tempboxa\hss}%
392             \advance\count@\@ne}}
393 </2ekernel | def>

```