

The `fxpicture` package

Line styles and grids for L^AT_EX's `picture` environment

Written by Claude AI, with direction from CodeCogs Ltd.

<https://fxeditor.codecogs.com>

Version 1.1, 15 October 2026

Abstract

`fxpicture` adds dashed, dotted and dash-dotted lines to L^AT_EX's `picture` environment. It builds on `pict2e`, and gives it two declarations, `\linestyle` and `\dashpattern`. Like `\linethickness`, they last until the end of the group, and they apply to every line that `pict2e` strokes, from `\line` and `\vector` to `\oval`, `\circle` and Bézier curves. It also draws grids, with `epic`'s `\grid`.

Contents

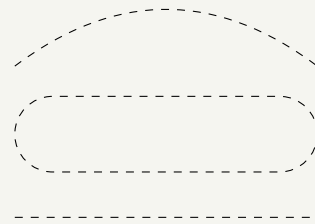
1	Introduction	2
2	Loading the package	2
3	Commands	2
4	Examples	3
4.1	Scope	3
4.2	Every kind of line	4
4.3	Arrows	4
4.4	A figure	5
5	Grids	5
6	What is dashed	6
7	How it works	6
7.1	<code>pict2e</code> 's paths	6
7.2	The dash	7
7.3	Arrows	7
7.4	Grids	7
7.5	If <code>pict2e</code> changes	8
7.6	SVG and other outputs	8
8	The code	9
9	Author, licence and history	12

1 Introduction

The `picture` environment is the drawing language built into $\text{\LaTeX}$. The `pict2e` package removes its old limits: lines and arrows at any slope, circles of any size, ovals, arcs, polygons and Bézier curves, all drawn by the output driver. Neither package draws a dashed or dotted line, though. The nearest are `\dashbox`, which dashes only the frame of a box, and `\qbezier[⟨n⟩]`, which draws a curve as n separate dots.

`fxpicture` fills that gap. It asks the driver to dash the path, which is what PostScript and PDF were designed for, so dashes run on round corners and curves, and a dashed picture is no larger or slower to typeset than a solid one.

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,32)
  {\linestyle{dashed}
   \put(5,2){\line(1,0){40}}
   \put(25,13){\oval(40,10)}
   \put(5,22){\qbezier(0,0)(20,15)(40,0)}
  }
\end{picture}
```



A working implementation. `fxpicture` was made alongside the `picture` support of CodeCogs' online equation editor, which draws the same line styles as SVG in the browser. To try it, see <https://fxeditor.codecogs.com>.

2 Loading the package

```
\usepackage{fxpicture}
```

`fxpicture` loads `pict2e`, and passes any options it is given on to it, so a driver or an arrow style can be chosen in the usual way:

```
\usepackage[pstarrows]{fxpicture}
```

If `pict2e` is already loaded, load `fxpicture` after it. `fxpicture` needs a `pict2e` with a driver. With `pict2e`'s `original` option, which leaves $\text{\LaTeX}$'s own picture commands in place, there is nothing to dash, and the line styles do nothing.

3 Commands

<code>\linestyle{⟨style⟩}</code>	Sets the style of the lines that follow, until the end of the group. The styles are solid , dashed , dotted and dashdot .
<code>\dashpattern{⟨on⟩}{⟨off⟩}</code>	Dashes the lines that follow with dashes $\langle on \rangle$ long and gaps $\langle off \rangle$ long, until the end of the group.
<code>\grid(⟨wd⟩,⟨ht⟩)(⟨dx⟩,⟨dy⟩)</code>	A grid $\langle wd \rangle$ by $\langle ht \rangle$, with lines every $\langle dx \rangle$ and every $\langle dy \rangle$, its lower left corner at the point. Section 5.
<code>\grid(...)(...)[⟨x0⟩,⟨y0⟩]</code>	The same grid, each line numbered, counting from $\langle x0 \rangle$ and $\langle y0 \rangle$.

The named styles, and a pattern of your own, look like this:

$\langle style \rangle$	pattern	
<code>solid</code>	(the default)	
<code>dashed</code>	on 3 pt, off 3 pt	
<code>dotted</code>	on 1 line width, off 2 pt	
<code>dashdot</code>	on 3 pt, off 2 pt, on 1 line width, off 2 pt	
<code>\dashpattern{3pt}{2pt}</code>	on 3 pt, off 2 pt	

The patterns are the same as TikZ's `dashed`, `dotted` and `dash dot`, so a `picture` and a TikZ drawing on the same page look alike.

Lengths. The arguments of `\dashpattern` are lengths, such as `3pt` or `1mm`, or plain numbers, which count as multiples of `\unitlength`, like other `picture` lengths. A plain number uses `\unitlength` as it is when `\dashpattern` is given, not when the line is drawn. Neither length may be negative, and they cannot both be zero.

Dots follow the line width. The dots of `dotted` and `dashdot` are as long as the line is thick. The width is read when each line is drawn, so the dots grow with `\thicklines` or `\linethickness`, and they stay square:

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,12)
  \linestyle{dotted}
  \put(0,10){\line(1,0){40}}
  \thicklines
  \put(0,5){\line(1,0){40}}
  \linethickness{1.2pt}
  \put(0,0){\line(1,0){40}}
\end{picture}
```



Round dots. For round dots, combine `pict2e`'s `\roundcap` with a `\dashpattern` whose dashes have no length. Every dash then becomes a dot as wide as the line. Note that without `\roundcap`, dashes of zero length can't be seen.

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,4)
  \linethickness{1.2pt}
  \roundcap\dashpattern{0pt}{3pt}
  \put(1,2){\line(1,0){38}}
\end{picture}
```



4 Examples

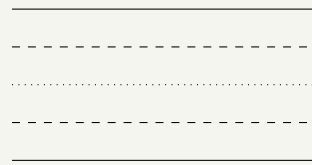
4.1 Scope

`\linestyle` and `\dashpattern` are declarations, so a pair of braces limits them to a part of a picture. The last style given wins, so `\linestyle{solid}` switches back to solid lines:

```

\setlength{\unitlength}{1mm}
\begin{picture}(40,22)
\put(0,20){\line(1,0){40}}
{\linestyle{dashed}
\put(0,15){\line(1,0){40}}
{\linestyle{dotted}
\put(0,10){\line(1,0){40}}}
\put(0,5){\line(1,0){40}}
\linestyle{solid}
\put(0,0){\line(1,0){40}}}
\end{picture}

```

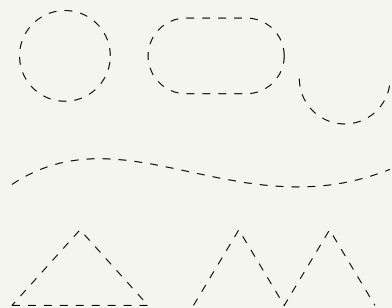


4.2 Every kind of line

```

\setlength{\unitlength}{1mm}
\begin{picture}(50,40)
\linestyle{dashed}
\put(7,33){\circle{12}}
\put(27,33){\oval(18,10)}
\put(44,30){\arc[180,360]{6}}
\put(0,16){\cbezier(0,0)(15,10)(30,-6)(50,2)}
\polygon(0,0)(18,0)(9,10)
\polyline(24,0)(30,10)(36,0)(42,10)(48,0)
\end{picture}

```



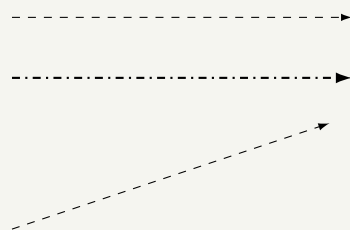
4.3 Arrows

The shaft of an arrow is dashed, and its head stays solid. Both of `pict2e`'s arrowheads can be used, and the head keeps its shape.

```

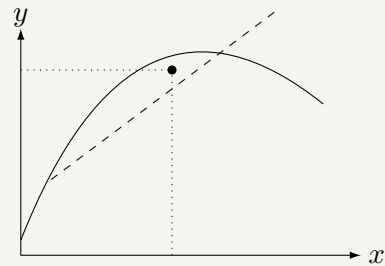
\setlength{\unitlength}{1mm}
\begin{picture}(45,30)
\linestyle{dashed}
\put(0,28){\vector(1,0){45}}
\put(0,0){\vector(3,1){42}}
\thicklines
\linestyle{dashdot}
\put(0,20){\vector(1,0){45}}
\end{picture}

```



4.4 A figure

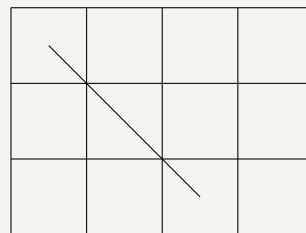
```
\setlength{\unitlength}{1mm}
\begin{picture}(50,36)(-5,-5)
\put(0,0){\vector(1,0){45}}
\put(0,0){\vector(0,1){30}}
\put(46,-1){\textit{x}}\put(-1,31){\textit{y}}
\qbezier(0,2)(15,40)(40,20)
{\linestyle{dotted}
\put(20,0){\line(0,1){24.5}}
\put(0,24.5){\line(1,0){20}}}
{\linestyle{dashed}
\put(4,10){\line(4,3){30}}}
\put(20,24.5){\circle*{1.2}}
\end{picture}
```



5 Grids

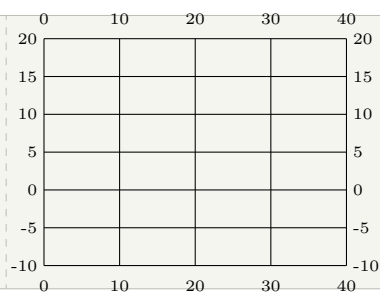
`\grid` is `epic`'s, with the same arguments and the same drawing: lines every $\langle dx \rangle$ across and every $\langle dy \rangle$ up, and, with $[\langle x0 \rangle, \langle y0 \rangle]$, a number at each end of every line. The numbers count up by the steps.

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,30)
\grid(40,30)(10,10)
\put(5,25){\line(1,-1){20}}
\end{picture}
```



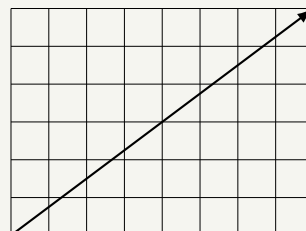
There are as many lines each way as whole steps fit in the size, and one more. A size that is not a whole number of steps leaves its far side open, as `epic`'s does. A grid is drawn in the line width in force, and `epic` suggests `\tiny` for the numbers:

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,30)
\thinlines
\tiny
\grid(40,30)(10,5)[0,-10]
\end{picture}
```



For a light grid to draw on, give it a thin line of its own:

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,30)
{\linethickness{0.1pt}
\grid(40,30)(5,5)}
\thicklines
\put(0,0){\vector(4,3){40}}
\end{picture}
```



Lengths. The sizes and steps may be numbers, in `\unitlength`, or lengths, such as 2cm or 5mm, as the `picture` package allows elsewhere. The numbers count up by the steps in `\unitlength`, so a grid 20 by 10 in steps of 5 is numbered 0, 5, 10, 15 and 20 whatever `\unitlength` is. The numbers on $\langle x0 \rangle$ and $\langle y0 \rangle$ are whole numbers.

Line styles. A grid's lines are TeX rules, as `epic`'s are, so `\linestyle` does not dash them.

Unlike `epic`. `epic` draws its grid in a box as wide as the grid. Written straight in a `picture`, rather than in a `\put`, that box moves whatever follows it to the right by the grid's width. `fxpicture`'s takes no room, so a `\put` after it lands where its coordinates say. In a `\put`, the two are the same, pixel for pixel:

```
\begin{picture}(60,40)
  \grid(60,40)(10,10)           % epic: the line below lands at (69,37)
  \put(9,37){\line(1,1){21}}    % fxpicture: at (9,37)
\end{picture}
```

With `epic`. If `epic` (or `eepic`, which loads it) is loaded, before or after `fxpicture`, its `\grid` is the one used, and `fxpicture` says so in the log. A document written for `epic` therefore draws as it always did. Write `\put(0,0){\grid...}` for a picture that draws the same with either.

6 What is dashed

- **Dashed:** every path that `pict2e` strokes, from `\line`, `\Line`, `\polyline`, `\polygon`, `\circle`, `\oval`, `\arc`, `\qbezier`, `\cbezier`, and `\moveto ... \strokepath`, to the shafts of `\vector`, `\Vector` and `\polyvector`.
- **Not dashed:** filled shapes, which are `\circle*`, `\polygon*`, `\arc*`, `\fillpath` and arrowheads. Also `\qbezier[\langle n \rangle]` with a number of points, which draws dots of its own, `\dashbox`, which has its own dashes, and `\grid`, whose lines are rules. Text is not affected.

Each path starts its pattern afresh, so every line starts with a dash. The pattern carries on round the corners of one path, such as a polygon or an oval, as PostScript and PDF always dash. The pattern is not stretched to fit the length of the line, so a line may end part way through a gap.

7 How it works

This section is for those who want to know what `fxpicture` changes in `pict2e`. The full code follows in section 8.

7.1 `pict2e`'s paths

`pict2e` draws each shape in the same way. The command, `\oval` for example, builds a path with `\pIIE@moveto`, `\pIIE@lineto` and `\pIIE@curveto`, which collect PostScript or PDF operators in a token register, `\pIIE@GRAPH`. Then `\pIIE@strokeGraph` or `\pIIE@fillGraph` hands the register to `\pIIE@drawGraph`. That macro puts the drawing state in front of the path and the paint operator after it, and passes the whole to the driver as one literal. For a stroke, the state is the scale from points to big points, the line width, and any line cap and line join, for example in PDF:

```
q 0.99626 0 0 0.99626 0 0 cm 0.4 w 0 0 m 113.81 0 1 S Q
```

The literal is wrapped in a save and a restore (`q ... Q` in PDF, `save ... restore` in PostScript), so nothing set in it lasts past the shape.

7.2 The dash

`fxpicture` replaces `\pIIe@drawGraph` with a copy that writes one more operator, the dash, after the line join, in the branch that strokes. Fills are left as they are. The dash is the macro `\fxp@dash`, which the styles set:

```
q 0.99626 0 0 0.99626 0 0 cm 0.4 w [3 3] 0 d 0 0 m 113.81 0 1 S Q
```

Lengths are written in points, since `pict2e` has already scaled its user space to points. The operator is `d` for PDF, from `pdfTeX`, `LuaTeX`, `XeTeX` and `dvipdfm(x)`, and `setdash` for PostScript, from `dvips`. `pict2e`'s mode tells `fxpicture` which to use.

`\fxp@dash` is expanded inside the `\edef` that builds the state. So `dotted` and `dashdot` write the line width there, `\strip@pt \@wholewidth`, and their dots take the width of the line being drawn. `\dashpattern` works out its lengths at once, which is why a plain number takes `\unitlength` as it is then.

The solid style leaves `\fxp@dash` empty. So a picture with no `\linestyle` comes out exactly as `pict2e` makes it, byte for byte. Since every shape is saved and restored, a dash never reaches past its own shape, and nothing has to be undone at the end of a group.

7.3 Arrows

`pict2e` draws an arrow as one filled outline, which goes round the head and the shaft together. Filling ignores dashes, so with a line style `fxpicture` draws an arrow in two parts:

1. It runs `pict2e`'s own arrowhead code with `\pIIe@lineto` changed to skip the points at $x = 0$. In both the `LATEX` and the `PSTricks` arrowheads, these are the only points of the outline at $x = 0$: the two back corners of the shaft. What is left is the head, in its own shape.
2. It strokes the shaft as a line of its own, from the tail to half way into the head, and then fills the head on top. The dashes so end under the head.

Solid arrows are `pict2e`'s, untouched. `\ltxarrows` and `\pstarrows` are redefined to choose the head for both cases. An arrow shorter than its head has no shaft, and is drawn as its head alone.

7.4 Grids

`\grid` is `epic`'s drawing, inside a `\put(0,0)` of its own. Like any `\put`, that is a box of no width, which is why it moves nothing that follows.

- **Lines.** The number of lines each way is `TeX`'s integer division of the size by the step, plus one. The upright lines are rules `\@wholewidth` wide and `\langle ht \rangle` high, drawn with `\multiput`; the level lines are rules `\langle wd \rangle` long.
- **Numbers.** The four rows of numbers are `epic`'s boxes, as follows.
 - Below: `\makebox(0,-2)[t]`, whose top is 2 units under the bottom line.

- Above: `\makebox(0,0)[b]`, raised 2 mm. `epic` raises them with a `\vspace{2mm}` inside the box; `fxpicture` does the same with `\raisebox`.
- Left: `[r]`, ending a space before the line.
- Right: `[l]`, starting a space after it.

The two draw the same grid to the pixel. `fxpicture` only defines `\grid` at `\begin{document}`, and then only if nothing has defined it already. That is how `epic` keeps its own, loaded before or after.

7.5 If pict2e changes

`fxpicture` replaces a macro that is internal to `pict2e`. When loaded, it compares `\pIIe@drawGraph` with the one it was written for, from `pict2e` 0.4b of 2020, and warns if they differ.

7.6 SVG and other outputs

The line styles map directly onto other drawing systems. In SVG, `\linestyle{dashed}` is `stroke-dasharray="3 3"`, and `\dashpattern{3pt}{2pt}` is `stroke-dasharray="3 2"`, in points. SVG, like PDF, starts the pattern afresh on each path. This is how the editor at <https://fxeditor.codecogs.com> draws them.

8 The code

```
1 %% fpicture.sty
2 %%
3 %% Line styles for LaTeX's picture environment, on top of pict2e.
4 %%
5 %% Written by Claude AI (Anthropic), with direction from CodeCogs Ltd.
6 %% A working implementation: https://fxeditor.codecogs.com
7 %%
8 %% Copyright (C) 2026 CodeCogs Ltd.
9 %%
10 %% This work may be distributed and/or modified under the conditions of the LaTeX Project
11 %% Public License, either version 1.3c of this license or (at your option) any later version.
12 %% The latest version of this license is in https://www.latex-project.org/lppl.txt
13 %%
14 %% This work has the LPPL maintenance status `maintained'. The Current Maintainer of this
15 %% work is CodeCogs Ltd. This work consists of the files fpicture.sty and fpicture-doc.tex,
16 %% and the derived file fpicture-doc.pdf.
17 %%
18 %% \usepackage{fpicture}          % loads pict2e; options are passed to it
19 %%
20 %% \linestyle{solid}              % the default
21 %% \linestyle{dashed}            % on 3pt, off 3pt
22 %% \linestyle{dotted}            % on one line width, off 2pt
23 %% \linestyle{dashdot}           % on 3pt, off 2pt, on one line width, off 2pt
24 %% \dashpattern{<on>}{<off>}     % a dash pattern of your own, eg \dashpattern{3pt}{2pt}
25 %%
26 %% Both are ordinary declarations, like \linethickness: they last to the end of the group, and
27 %% apply to every line pict2e strokes - \line, \vector's shaft, \Line, \polyline, \polygon,
28 %% \circle, \oval, \arc, \qbezier, \cbezier and \strokepath. Filled shapes (\circle*, \polygon*,
29 %% \arc*, arrowheads) are not dashed.
30 %%
31 %% {\linestyle{dashed}
32 %%   \put(10,10){\line(1,0){40}}
33 %%   \put(10,20){\oval(40,10)}
34 %%   \put(10,30){\qbezier(0,0)(20,15)(40,0)}
35 %% }
36 %%
37 %% The lengths of \dashpattern are lengths (3pt, 1mm), or bare numbers as multiples of
38 %% \unitlength as it is when \dashpattern is given. The pattern starts afresh at the beginning
39 %% of each path. The named styles follow TikZ's (dashed, dotted, dash dot), so a picture and a
40 %% TikZ drawing side by side match; dotted and dashdot dots are as long as the line is thick,
41 %% whatever \linethickness is when the line is drawn.
42 %%
43 %% Works with each of pict2e's drivers (pdfTeX, luatex, xetex, dvips, dvipdfm).
44 %%
45 %% \grid(<wd>,<ht>)(<dx>,<dy>)      % a grid, its lower-left corner at the point
46 %% \grid(<wd>,<ht>)(<dx>,<dy>)[<x0>,<y0>] % its lines numbered from x0 and y0
47 %%
48 %% epic's \grid, drawn as epic draws it - the same lines, the same numbers in the same places -
49 %% but taking no room: written straight in the picture rather than in a \put, epic's moves what
50 %% follows along by the grid's width, this one leaves it where it is. The lengths may have units
51 %%
52 %% as the picture package allows; the numbers count up by the steps in \unitlength. Its lines
53 %% are rules, as epic's are, so \linestyle does not dash them. When epic is loaded, before or
54 %% after this package, epic's \grid is the one used.
55 %%
56 \NeedsTeXFormat{LaTeX2e}
57 \ProvidesPackage{fpicture}[2026/10/15 v1.1 Line styles and grids for the picture environment]
58
59 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{pict2e}}
60 \ProcessOptions\relax
61 \RequirePackage{pict2e}[2020/09/30]
```

```

61
62 % The dash operator of the output: PostScript for dvips, PDF otherwise. pict2e has already set
63 % its user space to points (\pIIE@scale@PTtoBP), so the lengths are written in pt.
64 \ifnum\pIIE@mode=\@ne
65   \def\fxp@setdash@op{setdash}
66 \else
67   \def\fxp@setdash@op{d}
68 \fi
69
70 % \fxp@dash expands, inside pict2e's \edef, to the dash operator of the current style, or to
71 % nothing for solid lines - so a picture without \linestyle comes out exactly as pict2e makes it.
72
73 \let\fxp@dash\@empty
74 \def\fxp@w{\strip@pt\@wholewidth}
75
76 \namedef{fxp@style@solid}{\let\fxp@dash\@empty}
77 \namedef{fxp@style@dashed}{%
78   \def\fxp@dash{ [3 3] 0 \fxp@setdash@op}}
79 \namedef{fxp@style@dotted}{%
80   \def\fxp@dash{ [\fxp@w\space 2] 0 \fxp@setdash@op}}
81 \namedef{fxp@style@dashdot}{%
82   \def\fxp@dash{ [3 2 \fxp@w\space 2] 0 \fxp@setdash@op}}
83
84 \DeclareRobustCommand*\linestyle[1]{%
85   \@ifundefined{fxp@style@#1}{%
86     {\PackageError{fxpicture}{Unknown line style `#1'}%
87       {The line styles are solid, dashed, dotted and dashdot;\MessageBreak
88         \protect\dashpattern{on}{off} makes one of your own.}}%
89     {\@nameuse{fxp@style@#1}}%
90   }
91 \ignorespaces}
92
93 \DeclareRobustCommand*\dashpattern[2]{%
94   \@defaultunitsset\@tempdima{#1}\unitlength
95   \@defaultunitsset\@tempdimb{#2}\unitlength
96   \ifdim\@tempdima<\z@ \fxp@baddash \else
97   \ifdim\@tempdimb<\z@ \fxp@baddash \else
98   \ifdim\dimexpr\@tempdima+\@tempdimb\relax=\z@ \fxp@baddash \else
99   \edef\fxp@dash{ [\strip@pt\@tempdima\space\strip@pt\@tempdimb] 0 \noexpand\fxp@setdash@op}%
100 \fi\fi\fi
101 \ignorespaces}
102 \def\fxp@baddash{%
103   \PackageError{fxpicture}{Bad \protect\dashpattern}%
104   {Its lengths can't be negative, nor both zero.}}
105
106 % pict2e strokes every path through \pIIE@drawGraph; this is pict2e's own (v0.4b, 2020) with
107 % \fxp@dash after the line join, in the branch that strokes. With no driver (pict2e's
108 % 'original' option) pict2e draws nothing of its own, and there is nothing to dash.
109 \ifnum\pIIE@mode>\z@
110   \def\fxp@drawGraph@pIIE{%
111     \edef\x{\pIIE@debug@comment\space
112       \pIIE@scale@PTtoBP}%
113     \if@tempswa
114       \edef\y{\pIIE@fill@op}%
115     \else
116       \edef\x{x\space
117         \strip@pt\@wholewidth\space\pIIE@setlinewidth@op
118         \pIIE@linecap\pIIE@linejoin\space}%
119       \edef\y{\pIIE@stroke@op}%
120     \fi
121     \expandafter\pIIE@code\expandafter{%
122       \expandafter\x\the\pIIE@GRAPH\space\y}%
123     \global\pIIE@GRAPH{}\xdef\pIIE@CPx{}\xdef\pIIE@CPy{}%
124   }
125 \endgroup

```

```

123 \ifx\pIIE@drawGraph\fxp@drawGraph\pIIE\else
124 \PackageWarningNoLine{fxpicture}{%
125 This pict2e's \string\pIIE@drawGraph\space is not the one\MessageBreak
126 fpicture was written for; line styles may not work}%
127 \fi
128 \def\pIIE@drawGraph{%
129 \edef\x{\pIIE@debug\comment\space
130 \pIIE@scale@PTtoBP}%
131 \if@tempswa
132 \edef\y{\pIIE@fill@op}%
133 \else
134 \edef\x{\x\space
135 \strip@pt\@wholewidth\space\pIIE@setlinewidth@op
136 \pIIE@linecap\pIIE@linejoin\fxp@dash\space}%
137 \edef\y{\pIIE@stroke@op}%
138 \fi
139 \expandafter\pIIE@code\expandafter{%
140 \expandafter\x\the\pIIE@GRAPH\space\y}%
141 \global\pIIE@GRAPH{}\xdef\pIIE@CPx{}\xdef\pIIE@CPy{}%
142 \endgroup}
143 \fi
144
145 % pict2e's arrows (\vector, \Vector, \polyvector) are one filled outline, shaft and head
146 % together. With a line style, the outline is drawn without the shaft - its back corners are the
147 % only points at x=0, in both the LaTeX and the PSTricks arrowheads - and the shaft is stroked
148 % on its own, from the tail to half way into the head, under the head. Solid arrows are pict2e's
149 % own. \ltxarrows and \pstarrows still choose the arrowhead.
150 \ifnum\pIIE@mode>\z@
151 \let\fxp@vector@orig\pIIE@vector
152 \def\ltxarrows{\let\fxp@vector@orig\pIIE@vector\ltx}
153 \def\pstarrows{\let\fxp@vector@orig\pIIE@vector\pst}
154 \def\pIIE@vector{%
155 \ifx\fxp@dash\@empty \fxp@vector@orig \else \fxp@dashedvector \fi}
156 \def\fxp@dashedvector{%
157 \edef\fxp@concat{\the\pIIE@GRAPH}%
158 \let\fxp@lineto\pIIE@lineto
159 \let\pIIE@lineto\fxp@headlineto
160 \fxp@vector@orig
161 \let\pIIE@lineto\fxp@lineto
162 \ifdim\@xdim>\z@
163 \edef\fxp@head{\the\pIIE@GRAPH}%
164 \global\pIIE@GRAPH\expandafter{\fxp@concat}%
165 \@tempdimc\@xdim \advance\@tempdimc\@linelen \divide\@tempdimc\tw@
166 \pIIE@moveto\z@\z@ \pIIE@lineto\@tempdimc\z@
167 \pIIE@strokeGraph
168 \global\pIIE@GRAPH\expandafter{\fxp@head}%
169 \fi}
170 \def\fxp@headlineto#1#2{\ifdim#1=\z@ \else \fxp@lineto{#1}{#2}\fi}
171 \fi
172
173 % epic's \grid(wd,ht)(dx,dy)[x0,y0], in a \put(0,0) of its own so that it takes no room. The
174 % lines
175 % are epic's rules: as many each way as the integer division of size by step, and one more. The
176 % numbers are epic's \makebox(0,-2)[t] below, (0,0)[b] above (raised the 2mm epic's \vspace{2mm}
177 % gives), [r] on the left and [l] on the right, a space from the grid.
178 \newcount\fxp@gridcnt
179 \newcount\fxp@gridstep
180 \newcount\fxp@gridn
181 \newdimen\fxp@gridw
182 \newdimen\fxp@griddx
183 \newdimen\fxp@griddy
184 \def\fxp@gridcmd(#1,#2)(#3,#4){%

```

```

185 \ifnextchar[{\fxp@grid(#1,#2)(#3,#4)}{\fxp@grid(#1,#2)(#3,#4)[@,@]}}
186 \long\def\fxp@grid(#1,#2)(#3,#4)[#5,#6]{%
187 \@defaultunitsset\fxp@gridw{#1}\unitlength
188 \@defaultunitsset\fxp@gridh{#2}\unitlength
189 \@defaultunitsset\fxp@griddx{#3}\unitlength
190 \@defaultunitsset\fxp@griddy{#4}\unitlength
191 \ifdim\fxp@griddx>\z@ \ifdim\fxp@griddy>\z@
192 \put(0,0){%
193 % The upright lines, and their numbers.
194 \fxp@gridn\fxp@gridw \divide\fxp@gridn\fxp@griddx \advance\fxp@gridn\@ne
195 \multiput(0,0)(#3,0){\fxp@gridn}{\hb@xt@{\z@}{\hskip-\@halfwidth
196 \vrule\@width\@wholewidth\@height\fxp@gridh\@depth\z@\hss}}%
197 \ifx#5@ \else
198 \fxp@gridstep\fxp@griddx \divide\fxp@gridstep\unitlength
199 \global\fxp@gridcnt=#5\relax
200 \multiput(0,0)(#3,0){\fxp@gridn}{\makebox(0,-2)[t]{%
201 \number\fxp@gridcnt\global\advance\fxp@gridcnt\fxp@gridstep}}%
202 \global\fxp@gridcnt=#5\relax
203 \multiput(0,#2)(#3,0){\fxp@gridn}{\makebox(0,0)[b]{\raisebox{2mm}{%
204 \number\fxp@gridcnt}\global\advance\fxp@gridcnt\fxp@gridstep}}%
205 \fi
206 % The level lines, and theirs.
207 \fxp@gridn\fxp@gridh \divide\fxp@gridn\fxp@griddy \advance\fxp@gridn\@ne
208 \multiput(0,0)(0,#4){\fxp@gridn}{\vrule\@height\@halfwidth\@depth\@halfwidth
209 \@width\fxp@gridw}%
210 \ifx#6@ \else
211 \fxp@gridstep\fxp@griddy \divide\fxp@gridstep\unitlength
212 \global\fxp@gridcnt=#6\relax
213 \multiput(0,0)(0,#4){\fxp@gridn}{\makebox(0,0)[r]{%
214 \number\fxp@gridcnt\global\advance\fxp@gridcnt\fxp@gridstep}}%
215 \global\fxp@gridcnt=#6\relax
216 \multiput(#1,0)(0,#4){\fxp@gridn}{\makebox(0,0)[l]{%
217 \number\fxp@gridcnt\global\advance\fxp@gridcnt\fxp@gridstep}}%
218 \fi}%
219 \else \fxp@badgrid \fi \else \fxp@badgrid \fi
220 \ignorespaces}
221 \def\fxp@badgrid{%
222 \PackageError{fxpicture}{Bad \protect\grid}{Its steps must be more than zero.}}
223 % Decided at \begin{document}, so that epic, loaded before or after, keeps its own.
224 \AtBeginDocument{%
225 \ifundefined{grid}%
226 {\let\grid\fxp@gridcmd}%
227 {\PackageInfo{fxpicture}{\string\grid\space is already defined (by epic?): kept}}}
228
229 \endinput

```

9 Author, licence and history

fxpicture was written by Claude AI (Anthropic), with direction from CodeCogs Ltd.

Copyright © 2026 CodeCogs Ltd.

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3c of this license or (at your option) any later version. The latest version of this license is in <https://www.latex-project.org/lppl.txt>. This work has the LPPL maintenance status ‘maintained’. The Current Maintainer of this work is CodeCogs Ltd.

For a working implementation, see <https://fxeditor.codecogs.com>.

History

1.0 (2026/10/15) First release.

1.1 (2026/10/15) \grid, epic's, taking no room.