

The `fxpicture` package

Line styles for $\text{\LaTeX}$'s `picture` environment

Written by Claude AI, with direction from CodeCogs Ltd.

<https://fxeditor.codecogs.com>

Version 1.0, 15 October 2026

Abstract

`fxpicture` adds dashed, dotted and dash-dotted lines to $\text{\LaTeX}$'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.

Contents

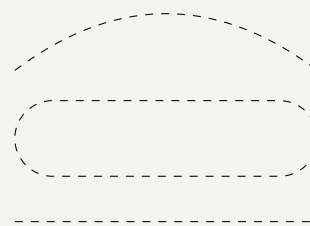
1	Introduction	1
2	Loading the package	1
3	Commands	2
4	Examples	3
4.1	Scope	3
4.2	Every kind of line	3
4.3	Arrows	3
4.4	A figure	4
5	What is dashed	4
6	How it works	4
6.1	<code>pict2e</code> 's paths	4
6.2	The dash	5
6.3	Arrows	5
6.4	If <code>pict2e</code> changes	5
6.5	SVG and other outputs	6
7	The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

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[ $\langle n \rangle$ ]`, 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 L^AT_EX'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 <i><on></i> long and gaps <i><off></i> long, until the end of the group.

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

<i><style></i>	pattern	
solid	(the default)	_____
dashed	on 3 pt, off 3 pt	- - - - -
dotted	on 1 line width, off 2 pt	
dashdot	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

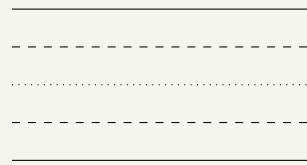
4.1 Scope

`\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}

```

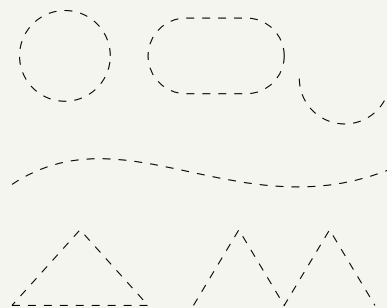


4.2 Every kind of line

```

\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}

```



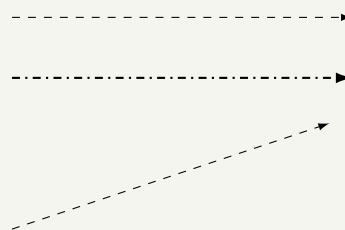
4.3 Arrows

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.

```

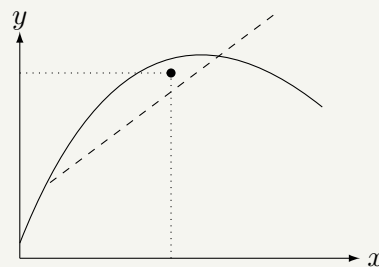
\setlength{\unitlength}{1mm}
\begin{picture}(45,30)
\linestyle{dashed}
\put(0,28){\vector(1,0){45}}
\put(0,0){\vector(3,1){42}}
\thicklines
\linestyle{dashdot}
\put(0,20){\vector(1,0){45}}
\end{picture}

```



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 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[⟨n⟩]` with a number of points, which draws dots of its own, and `\dashbox`, which has its own dashes. 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.

6 How it works

This section is for those who want to know what `fxpicture` changes in `pict2e`. The full code follows in section 7.

6.1 `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.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

7 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 \NeedsTeXFormat{LaTeX2e}
46 \ProvidesPackage{fpicture}[2026/10/15 v1.0 Line styles for the picture environment]
47
48 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{pict2e}}
49 \ProcessOptions\relax
50 \RequirePackage{pict2e}[2020/09/30]
51
52 % The dash operator of the output: PostScript for dvips, PDF otherwise. pict2e has already set
53 % its user space to points (\pIle@scale@PTtoBP), so the lengths are written in pt.
54 \ifnum\pIle@mode=\@ne
55 \def\fxp@setdash@op{setdash}
56 \else
57 \def\fxp@setdash@op{d}
58 \fi
59
60 % \fxp@dash expands, inside pict2e's \edef, to the dash operator of the current style, or to
```

```

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

```

```

123     \else
124     \edef\x{\x\space
125     \strip@pt\@wholewidth\space\pIe@setlinewidth@op
126     \pIe@linecap\pIe@linejoin\fxp@dash\space}%
127     \edef\y{\pIe@stroke@op}%
128     \fi
129     \expandafter\pIe@code\expandafter{%
130     \expandafter\x\the\pIe@GRAPH\space\y}%
131     \global\pIe@GRAPH{}\xdef\pIe@CPx{}\xdef\pIe@CPy{}%
132     \endgroup}
133 \fi
134
135 % pict2e's arrows (\vector, \Vector, \polyvector) are one filled outline, shaft and head
136 % together. With a line style, the outline is drawn without the shaft - its back corners are the
137 % only points at x=0, in both the LaTeX and the PSTricks arrowheads - and the shaft is stroked
138 % on its own, from the tail to half way into the head, under the head. Solid arrows are pict2e's
139 % own. \ltxarrows and \pstarrows still choose the arrowhead.
140 \ifnum\pIe@mode>\z@
141     \let\fxp@vector@orig\pIe@vector
142     \def\ltxarrows{\let\fxp@vector@orig\pIe@vector\ltx}
143     \def\pstarrows{\let\fxp@vector@orig\pIe@vector\pst}
144     \def\pIe@vector{%
145     \ifx\fxp@dash\@empty \fxp@vector@orig \else \fxp@dashedvector \fi}
146     \def\fxp@dashedvector{%
147     \edef\fxp@concat{\the\pIe@GRAPH}%
148     \let\fxp@lineto\pIe@lineto
149     \let\pIe@lineto\fxp@headlineto
150     \fxp@vector@orig
151     \let\pIe@lineto\fxp@lineto
152     \ifdim\@xdim>\z@
153     \edef\fxp@head{\the\pIe@GRAPH}%
154     \global\pIe@GRAPH\expandafter{\fxp@concat}%
155     \@tempdimc\@xdim \advance\@tempdimc\@linelen \divide\@tempdimc\tw@
156     \pIe@moveto\z@\z@ \pIe@lineto\@tempdimc\z@
157     \pIe@strokeGraph
158     \global\pIe@GRAPH\expandafter{\fxp@head}%
159     \fi}
160     \def\fxp@headlineto#1#2{\ifdim#1=\z@ \else \fxp@lineto{#1}{#2}\fi}
161 \fi
162
163 \endinput

```

8 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.