EEPIC
Extensions to epic and LaTeX Picture Environment Version 1.1

Conrad Kwok
Department of Electrical Engineering and Computer Science
University of California, Davis

Febrary 2, 1988

1 Introduction

LaTeX provides a basic but limited picture drawing capability. EPIC[1] is an enhancement to the picture environment of LaTeX which provides a simpler and more powerful interface. It introduces new commands for drawing solid lines, dotted lines, dash lines and new environments suitable for plotting graphs.

However, EPIC still inherits many of the limitations of LaTeX in picture drawing and hence some of the functions either take a long time to accomplish or the output is not very nice looking.

tpic is preprocessor program for use with TeX. It uses a set of `\special`s graphics commands for drawing pictures. More and more DVI driver programs supports those specials. They are becoming a standard set of `\special`s for DVI files. However, the major disadvantage of tpic is that the tpic preprocessor itself is not readily available on most machines. It is written in yacc and C language. It is mainly for UNIX or similar system.

EEPIC, as an extension to both LaTeX and EPIC, tries to alleviate some of the limitations in LaTeX, EPIC and tpic by generating tpic `special`s using TeX commands instead of any preprocessor program, but at the same time provides compatibility with the original commands such that when a DVI driver which understands tpic `special`s are not available, the documents can still be formatted using standard LaTeX and EPIC. However, the output probably will not be as good as originally intended.

Currently, EEPIC extends LaTeX and EPIC in the following ways:

- Draws lines in any slopes.
- Draws circles and discs (filled circle) in any radii.
- Draws dotted lines and dash lines in a much faster way and requires much less TeX internal memory.
- Provides more line thickness options.

[1]EPIC is a LaTeX macro package written by Sunil Podar at S.U.N.Y at Stony Brook. Please read the section on installation for more information

Furthermore, EEPIC introduces several new commands for:

- drawing of ellipsis and filled ellipsis
- drawing of arcs
- drawing of splines (cubic splines using control points)
- drawing of polylines

All the affected commands in LaTeX and EPIC will be discussed in the subsequent sections. The compatibility issues will be described in the section **??**.

In version 1.1, several bugs are fixed, and several commands for area filled are added.

2 Extension to LaTeX

In LaTeX, drawing of lines and circles are done using special fonts. Therefore only limited functions are provided. The extensions in EEPIC allow users to draw lines in any slope and to draw circles in any sizes. However, the limitation of slopes for vectors remains the same in the mean time. That is the slope that can be handled is $\frac{x}{y}$ where x and y are integers in the range $[-4, 4]$. Please read LaTeX manual for details.

2.1 `\line`

The syntax of `\line` is the same as that in LaTeX:

`\line(`*x*,*y*`){`*length*`}`

But now x and y can be any integer values within the limit of TeX. Furthermore, there is no more lower limit for *length* parameter.

2.2 `\circle`

The syntax of `\circle` is the same as that in LaTeX:

`\circle{`*diameter*`}`

or

`\circle*{`*diameter*`}`

But now the *diameter* parameter can be any number acceptable by TeX and a circle with the specified diameter (exactly) will be drawn.

2.3 `\oval`

The `\oval` command is changed such that the maximum diameter of the quarter circles at the corners can be set to any values. This is done by setting the variable `\maxovaldiam` to the desire TeX dimension. The default is 40pt.

3 Extension to EPIC

EPIC is an enhancement to the Picture Environment of LaTeX. EPIC generates standard DVI files and requires only standard LaTeX fonts. Some of the functions it provides are:

```
\multiputlist  \dottedline  \putfile
\matrixput     \dashline
\grid          \drawline
```

Details can be found in the EPIC manual.

Extensions to EPIC in EEPIC include better line drawing output, faster operation and less memory requirement. The commands affected are:

1. `\drawline`
2. `\dashline`
3. `\dottedline`

And the three "`*join`" environments are indirectly affected also.

3.1 `\drawline`

The syntax of `\drawline` is:

`\drawline[`*stretch*`]`$(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$

where *stretch* is an integer between −100 and infinity. However any number greater than 0 are the same. An negative *stretch* in `\drawline` will call `\dashline`.

The thickness of the line is affected by `\thinlines`, `\thicklines` and `\Thicklines` declarations. Horizontal and vertical lines are drawn using rules.

3.2 `\dottedline`

The syntax of `\dottedline` is:

`\dottedline[`*dot character*`]{`*dotgap*`}`$(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$

where *dot character* is the character used in drawing the "dotted" line. *dotgap* is the interdot gap in terms of `\unitlength`. `\specials` will only be generated if no optional dot character is specified.

The size of the dots are affected by `\thinlines`, `\thicklines` and `\Thicklines` declarations.

3.3 `\dashline`

The syntax of `\dashline` is:

`\dashline[`*stretch*`]{`*dash-length*`}[`*inter-dot-gap*`]`$(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$

where *stretch* is an integer between −100 and infinity. If *inter-dot-gap* is not specified, dashes are drawn in solid lines, otherwise, dashes are drawn using dotted lines.

The thickness of the line is affected by `\thinlines`, `\thicklines` and `\Thicklines` declarations.

4 New Commands

EEPIC introduces a number of new commands. Except the `\path` commands, all other new commands do not have any equivalents in LaTeX and EPIC. Please read section **??** about the compatibility issues.

4.1 `\allinethickness`

Set the line thickness of all line drawing commands including lines in any slopes, circles, ellipsis, arcs, ovals and splines. Note there are only two 'l' in the command. The syntax is:

`\allinethickness{`*dimension*`}`.

4.2 `\Thicklines`

The syntax is:

```
\Thicklines
```

With the `\Thicklines` declaration, thickness of lines drawn will be about 1.5 times of `\thicklines`.

4.3 `\path`

`\path` is a fast version of `\drawline`. Optional *stretch* argument is not allowed and so it always draw solid lines. The syntax is:

`\path(`x_1`,`y_1`)(`x_2`,`y_2`)...(`x_n`,`y_n`)`

`\path` is mainly used in drawing complex paths.

4.4 `\spline`

Syntax of `\spline` is the same as `\path`.

`\spline(`x_1`,`y_1`)(`x_2`,`y_2`)...(`x_n`,`y_n`)`

`\spline` draws an Chaikin's curve which passes through only the first and last point. All other points are control points only.

4.5 `\ellipse`

The command `\ellipse` draws an ellipse by specifying the x-diameter and y-diameter.

`\ellipse{`*x-diameter*`}{`*y-diameter*`}`

or

`\ellipse*{`*x-diameter*`}{`*y-diameter*`}`

When *x-diameter* is equal to *y-diameter*, the command is equivalent to `\circle` or `\circle*`.

4.6 `\arc`

`\arc` draws an circular arc. The syntax is

> `\arc`{*diameter*}{*start-angle*}{*end-angle*}

diameter is specified in `\unitlength` and both *start-angle* and *end-angle* are in radian. *start-angle* must be within 0 and 2π and *end-angle* can be any value between *start-angle* and *start-angle* $+ 2\pi$. Arcs are drawn in clockwise direction with angle 0 pointing to the right on the paper.

4.7 `\filltype{....}`

The command specifies the type of area fill for `\circle*` and `\ellipse*`. The command itself does not draw anything. It only changes the interpretation of `*` in the two commands specified above. The syntax of the command is:

> `\filltype`{*area-fill-type*}

The legal area fill type are:

- black (default)
- white
- shade

For example, to change area fill type to white fill, the following command should be used.

```
\filltype{white}
```

These commands are only intended for advance users (those who know what they are doing). They are included mainly because `fig2epic`[2] generate these commands. The commands are:

commands	Description
`\blacken`	Black fill the next figure
`\whiten`	White fill the next figure
`\shade`	Shade the interior next figure
`\texture`	Specify the pattern used for the next `shade` command. The pattern will remain effective until it is changed by another `\texture` command. The syntax is: `\texture`{ *32 32-bit hexadecimal numbers*} An example (the default) is: `\texture{cccccccc 0 0 0 cccccccc 0 0 0 cccccccc 0 0 0 cccccccc 0 0 0 cccccccc 0 0 0 cccccccc 0 0 0 cccccccc 0 0 0 cccccccc 0 0 0}`

[2]Another program written by me to convert Fig output file to eepic format.

The exact interpretation of the above commands are probably device driver dependent. I did most of tests using `iptex` (imagen1) and several tests using `dvips`. The description below may not apply to other device drivers.

The commands that can be specified after `\blacken`, `\whiten` and `\shade` include `\path`, `\circle` (without `*`), `\ellipse` (again without `*`) and `\arc`. The drawings do not have to be closed. The imagen printer will automatically draw an imaginary line from the starting point to the end point, and then fill the figure. When using `iptex`, the outline of the figures are drawn but not in `dvips`. In another words, when using `iptex`, the command:

```
\shade\circle{10}
```

will draw a circle will the circumference in solid line and the interior is filled in the pattern active at that time. However, when using dvips, the circle will not have the circumference drawn in sold line.

5 Examples

I shamelessly stole two examples from the EPIC manual so that you can compare the results. The third and fourth examples are created by FIG and then converted to EEPIC using `fig2epic` which is also written by me.

5.1 Example 1

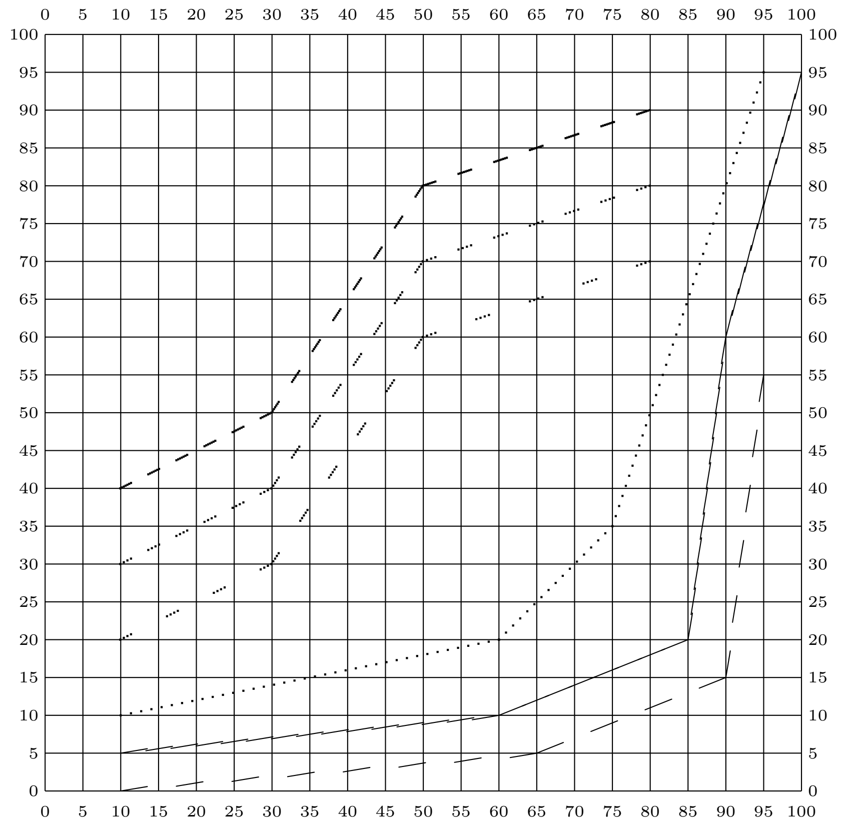

Figure 1: An example of Various Line Drawing Commands

5.2 Example 2

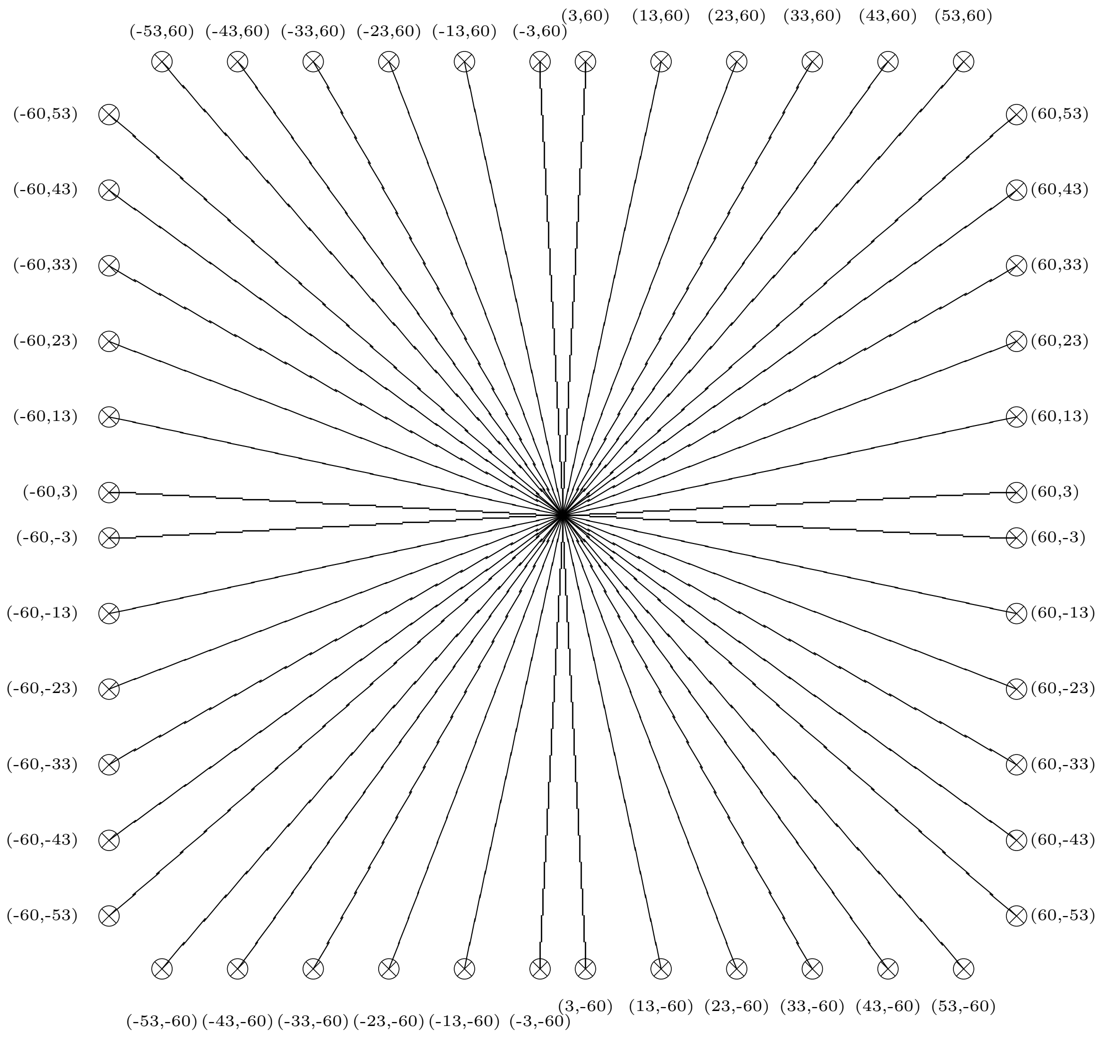

Figure 2: Test Sample: Lines of various slopes with `thinlines`