PostScript macros for Generic TeX.

to

Userjs Guide

.75em Timothy Van Zandt

Version

Authorjs address:
Department of Economics, Princeton University,
Princeton, NJ 08544-1021, USA. Internet: `tvz@Princeton.EDU`

Contents

PartWelcome to PSTricks

PSTricks is a collection of PostScript-based TEX macros that is compatible with most TEX macro packages, including Plain TEX, LATEX, TEX, and -LATEX. PSTricks gives you color, graphics, rotation, trees and overlays. PSTricks puts the icing (PostScript) on your cake (TEX)!

To install PSTricks, follow the instructions in the file "read-me.pst" that comes with the PSTricks package. Even if PSTricks has already been installed for you, give "read-me.pst" a look over.

This *Userjs Guide* verges on being a reference manual, meaning that it is not designed to be read linearly. Here is a recommended strategy: Finish reading this brief overview of the features in PSTricks. Then thumb through the entire *Userjs Guide* to get your own overview. Return to Part ? (Essentials) and read it carefully. Refer to the remaining sections as the need arises.

When you cannot figure out how to do something or when trouble arises, check out the appendices (Help). You just might be lucky enough to find a solution. There is also a LATEX file "samples.pst" of samples that is distributed with PSTricks. Look to this file for further inspiration.

This documentation is written with LATEX. Some examples use LATEX specific constructs and some donjt. However, there is nothing LATEX specific about any of the macros, nor is there anything that does not work with LATEX. This package has been tested with Plain TEX, LATEX, -LATEXand TEX, and should work with other TEX macro packages as well.

pstricks The main macro file is "pstricks.tex"/"pstricks.sty". Each of the PSTricks macro files comes with a ".tex" extension and a ".sty" extension; these are equivalent, but the ".sty" extension means that you can include the file name as a LATEX document style option.

There are numerous supplementary macro files. A file, like the one above and the left, is used in this *Userjs Guide* to remind you that you must input a file before using the macros it contains.

For most PSTricks macros, even if you misuse them, you will not get PostScript errors in the output. However, it is recommended that you resolve any TEX errors before attempting to print your document. A few PSTricks macros pass on PostScript errors without warning. Use these with care, especially if you are using a networked printer, because PostScript errors can cause a printer to bomb. Such macros are pointed out in strong terms, using a warning like this one:

[Sorry. Ignored `\beginWarning ... \endWarning`]

Keep in mind the following typographical conventions in this Userjs Guide.

All literal input characters, i.e., those that should appear verbatim in your input file, appear in upright "Helvetica" and "Helvetica-Bold" fonts.

Meta arguments, for which you are supposed to substitute a value (e.g., <angle>) appear in slanted <Helvetica-Oblique> and Helvetica-BoldOblique fonts.

The main entry for a macro or parameter that states its syntax appears in a large bold font, *except for the optional arguments, which are in medium weight*. This is how you can recognize the optional arguments.

References to PSTricks commands and parameters within paragraphs are set in Helvetica-Bold.

Part I

The Essentials

1 Arguments and delimiters

Here is some nitty-gritty about arguments and delimiters that is really important to know.

The PSTricks macros use the following delimiters:

Curly braces "<arg>"
Brackets (only for optional arguments)"[<arg>]"
Parentheses and commas for coordinates¸
"=" and "," for parameters"<par1>=<val1>," ?

Spaces and commas are also used as delimiters within arguments, but in this case the argument is expanded before looking for the delimiters.

Always use a period rather than a comma to denote the decimal point, so that PSTricks doesnjt mistake the comma for a delimiter.

The easiest mistake to make with the PSTricks macros is to mess up the delimiters. This may generate complaints from TEX or PSTricks about bad arguments, or other unilluminating errors such as the following:

=0pt=0pt

"! Use of @coor doesnjt match its definition."

"! Paragraph ended before @addcoor was complete."

"! Forbidden control sequence found while scanning use of -@arrow."

"! File ended while scanning use of ."

Delimiters are generally the first thing to check when you get errors with a PSTricks macro.

Since PSTricks macros can have many arguments, it is useful to know that you can leave a space or new line between any arguments, except between arguments enclosed in curly braces. If you need to insert a new line between arguments enclosed in curly braces, put a comment character " As a general rule, the first non-space character after a PSTricks macro should not be a "[" or "(". Otherwise, PSTricks might think that the "[" or "(" is actually part of the macro. You can always get around this by inserting a pair "" of braces somewhere between the macro and the "[" or "(".

Color

The grayscales

"black", "darkgray", "gray", "lightgray", and "white",

and the colors

"red", "green", "blue", "cyan", "magenta", and "yellow"

are predefined in PSTricks.

This means that these names can be used with the graphics objects that are described in later sections. This also means that the command (or , etc.) can be used much like "" or "", as in

[Sorry. Ignored \beginEx ... \endEx]

The commands , , etc. can be nested like the font commands as well. There are a few important ways in which the color commands differ from the font commands:

1. The color commands can be used in and out of math mode (there are no restrictions, other than proper TEX grouping).

2. The color commands affect whatever is in their scope (e.g., lines), not simply characters.

3. The scope of the color commands does not extend across pages.

4. The color commands are not as robust as font commands when used inside box macros. See page colorboxproblems for details. You can avoid most problems by explicitly grouping color commands (e.g., enclosing the scope in braces "") whenever these are in the argument of another command.

However, this is not necessary with the PSTricks LR-box commands, expect when is in effect. See Section ?.

You can define or redefine additional colors and grayscales with the following commands. In each case, <numi> is a number between 0 and 1. Spaces are used as delimitersndonjt add any extraneous spaces in the arguments.

colornum

<num> is the gray scale specification, to be set by PostScriptjs "setgray" operator. 0 is black and 1 is white. For example:

[Sorry. Ignored \beginLVerb ... \endLVerb]

colornum1 num2 num3

<num1 num2 num3> is a *red-green-blue* specification, to be set by PostScriptjs "setrgbcolor" operator. For example,

[Sorry. Ignored \beginLVerb ... \endLVerb]

colornum1 num2 num3

<num1 num2 num3> is an *hue-saturation-brightness* specification, to be set by PostScriptjs "sethsbcolor" operator. For example,

[Sorry. Ignored \beginLVerb ... \endLVerb]

colornum1 num2 num3 num4

<num1 num2 num3 num4> is a *cyan-magenta-yellow-black* specification, to be set by PostScriptjs "newcmykcolor" operator. For example,

[Sorry. Ignored \beginLVerb ... \endLVerb]

For defining new colors, the *rbg* model is a sure thing. *hsb* is not recommended. *cmyk* is not supported by all Level 1 implementations of PostScript, although it is best for color printing. For more information on color models and color specifications, consult the *PostScript Language Reference Manual*, 2nd Edition (Red Book), and a color guide.

[Sorry. Ignored \begindrivers ... \enddrivers]

Setting graphics parameters

PSTricks uses a key-value system of graphics parameters to customize the macros that generate graphics (e.g., lines and circles), or graphics combined with text (e.g., framed boxes). You can change the default values of parameters with the command as in

[Sorry. Ignored \beginLVerb ... \endLVerb]

The general syntax is: As illustrated in the examples above, spaces are used as delimiters for some of the values. Additional spaces are allowed only following the comma that separates <par>"="<value> pairs (which is thus a good place to start a new line if there are many parameter changes). E.g., the first example is acceptable, but the second is not:

[Sorry. Ignored \beginLVerb ... \endLVerb]

The parameters are described throughout this *Userjs Guide*, as they are needed.

Nearly every macro that makes use of graphics parameters allows you to include changes as an optional first argument, enclosed in square brackets. For example,

[Sorry. Ignored \beginLVerb ... \endLVerb]

draws a dotted, green line. It is roughly equivalent to

[Sorry. Ignored \beginLVerb ... \endLVerb]

For many parameters, PSTricks processes the value and stores it in a peculiar form, ready for PostScript consumption. For others, PSTricks stores the value in a form that you would expect. In the latter case, this *Userjs Guide* will mention the name of the command where the value is stored. This is so that you can use the value to set other parameters. E.g.,

[Sorry. Ignored \beginLVerb ... \endLVerb]

However, even for these parameters, PSTricks may do some processing and error-checking, and you should always set them using as optional parameter changes, rather than redefining the command where the value is stored.

Dimensions, coordinates and angles

Whenever an argument of a PSTricks macro is a dimension, the unit is optional. The default unit is set by the

[Sorry. Ignored \beginEx ... \endEx]

parameter. For example, with the default value of "1cm", the following are equivalent:

[Sorry. Ignored \beginLVerb ... \endLVerb]

By never explicitly giving units, you can scale graphics by changing the value of unit.

You can use the default coordinate when setting non-PSTricks dimensions as well, using the commands

[Sorry. Ignored \beginEx ... \endEx]

where <cmd> is a dimension register (in LATEX parlance, a klengthl), and <dim> is a length with optional unit. These are analogous to LATEXjs " and ".

Coordinate pairs have the form ¸. The origin of the coordinate system is at TEXjs currentpoint. The command lets you use polar coordinates, in the form "(<r>;<a>)", where <r> is the radius (a dimension) and <a> is the angle (see below). You can still use Cartesian coordinates. For a complete description of , see Section ?.

The unit parameter actually sets the following three parameters:

[Sorry. Ignored \beginEx ... \endEx]

These are the default units for x-coordinates, y-coordinates, and all other coordinates, respectively. By setting these independently, you can scale the x and y dimensions in Cartesian coordinate unevenly. After changing yunit to "1pt", the two js below are equivalent:

[Sorry. Ignored \beginLVerb ... \endLVerb]

The values of the runit, xunit and yunit parameters are stored in the dimension registers (also), and .

Angles, in polar coordinates and other arguments, should be a number giving the angle in degrees, by default. You can also change the units used for angles with the command °i[num]j <num> should be the number of units in a circle. For example, you might use

[Sorry. Ignored \beginLVerb ... \endLVerb]

to make a pie chart when you know the shares in percentages. ° without the argument is the same as

[Sorry. Ignored \beginLVerb ... \endLVerb]

The command is short for

[Sorry. Ignored \beginEx ... \endEx]

lets you specify angles in other ways as well.

Basic graphics parameters

The width and color of lines is set by the parameters:

[Sorry. Ignored \beginEx ... \endEx]

The linewidth is stored in the dimension register , and the linecolor is stored in the command .

The regions delimited by open and closed curves can be filled, as determined by the parameters:

[Sorry. Ignored \beginEx ... \endEx]

When fillstyle=none, the regions are not filled. When fillstyle=solid, the regions are filled with fillcolor. Other fillstylejs are described in Section ?.

The graphics objects all have a starred version (e.g.,) which draws a solid object whose color is linecolor. For example,

[Sorry. Ignored \beginMEx ... \endMEx]

Open curves can have arrows, according to the

[Sorry. Ignored \beginEx ... \endEx]

parameter. If arrows=-, you get no arrows. i=12 If arrows=<->, you get arrows on both ends of the curve. You can also set arrows=-> and arrows=<-, if you just want an arrow on the end or beginning of the curve, respectively. With the open curves, you can also specify the arrows as an optional argument enclosed in "" brackets. This should come after the optional parameters argument. E.g.,

[Sorry. Ignored \beginMEx ... \endMEx]

Other arrow styles are described in Section ?

If you set the

[Sorry. Ignored \beginEx ... \endEx]

parameter to "true", then most of the graphics objects will put dots at the appropriate coordinates or control points of the object. Section ? describes how to change the dot style.

The parameter value is stored in the conditional "".