Introduction to Scientific Typesetting
Lesson 12: Verbatim Text and Drawing in LaTeX

Ryan Higginbottom

January 20, 2012

Verbatim Text

The Basics of pstricks

Color

Verbatim Text

Writing About LaTeX (or other code)

If you are writing *about* LaTeX or any other code within your document, chances are you'll want to include some examples of that code.

The problem is: how do you allow your reader see *exactly* what you're typing? Most of the time this is difficult to do.

This is known as the problem of typing *verbatim* text.

This section relies heavily on pages 145–148 of *More Math into LaTeX*.

Typing Verbatim Text Inline

The command `\verb` allows you to type verbatim text inline. The character that comes directly after `\verb` is a delimiter, so you need to type that character again after the verbatim text is done.

Type: `It is not hard to type \verb+\%+`
Get: It is not hard to type `\%`

Choose your delimiter carefully! The delimiter cannot appear in the verbatim text itself without causing problems!

If you want to have `2+2=4` show up in verbatim, you cannot use `+` as your delimiter! Use `!`, `|`, `-` or something else.

```
\verb!2+2=4!
```

Rules for Verbatim Text Inline

Some rules to follow when using `\verb`:

- The entire `\verb` command must be typed on a single line in your `.tex` file.
- You cannot have a space between `\verb` and your delimiter.
- The `\verb` command cannot appear in the argument of another command.
- The `\verb` command cannot appear in an aligned math environment.
- Do not use `*` as a delimiter.

Displayed Verbatim Text

If you have a lot of verbatim text to type at once, use the `verbatim` environment.

```
\begin{verbatim}
verbatim text goes in here
\end{verbatim}
```

It is not necessary, but I believe you will get better behavior from this environment by including `\usepackage{verbatim}` in your preamble.

Rules for the `verbatim` Environment

Some rules to follow:

- A `verbatim` environment cannot be placed within another `verbatim` environment.
- A `verbatim` environment cannot be placed within the argument of another command.
- You must have `\end{verbatim}` on a line by itself, with no spaces before it.

There are a few other packages which can be used to typeset verbatim text: `alltt`, `listings`, and `fancyvrb`. Look into them if you like. (I use `fancyvrb` in my slides.)

Example

Open the first example file (`.tex`), build and view.

Verbatim Text in Presentations

You will need to make a small adjustment if you'd like to have a slide with verbatim text within a presentation.

- If you are using `beamer`, you'll need the `[fragile]` option on any frame with verbatim text. That is, you'll need:

```
\begin{frame}[fragile]
slide contents with verbatim text
\end{frame}
```

- If you are using `powerdot`, you'll need the `method=file` option on any slide with verbatim text. That is, you'll need:

```
\begin{slide}[method=file]{Title}
slide contents with verbatim text
\end{slide}
```

The Basics of pstricks

The Motivation

It is very inconvenient to have to include a graphic every time you'd like a picture in your document or presentation. Additionally, if you are sharing a file with someone else, this means that you'll have to send along the picture file(s) too. It's much more convenient to use a package where LaTeX code generates your pictures for you.

Enter `pstricks`!

Because

Graphics can

spice things up!

You'll need `\usepackage{pstricks}` in the preamble.

Getting Started

While `pstricks` commands can be used within the flow of text, it is more common to draw pictures that are set off by themselves. This happens within a `pspicture` environment.

You need to specify the dimensions of the environment. (One unit is 1 cm by default.) This happens by specifying the *lower left* and *upper right* coordinates of the environment. (Think of it as a grid!)

```
\begin{center}
\begin{pspicture}(0,0)(4,4)
...
\end{pspicture}
\end{center}
```

Basic Graphical Objects

Here are some pstricks commands for simple geometric objects.

- \psline(0,0)(2,1) — A line starting at $(0, 0)$ and ending at $(2, 1)$. Also valid: \psline(0,0)(2,1)(3,4).
- \psframe(0,0)(3,2) — A rectangle with corners at $(0, 0)$, $(3, 0)$, $(3, 2)$, and $(0, 2)$.
- \pspolygon(0,0)(3,1)(1,2) — A triangle with vertices at $(0, 0)$, $(3, 1)$, and $(1, 2)$. A polygon can have any number of vertices and pstricks will close up the polygon properly. *Order matters!*
- \pscircle(1,2){.5} — A circle with radius 0.5 cm centered at $(1, 2)$.
- \psellipse(0,0)(2,3) — An ellipse centered at $(0, 0)$ with horizontal radius 2 cm and vertical radius 3 cm.

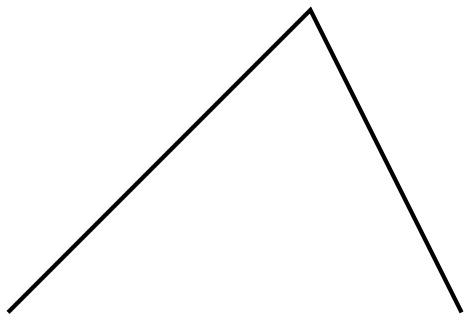
Curves and Rounding

There are two versions of a curve to mention.

- `\pscurve(0,0)(.5,0)(1,3)(-1,4)` — This interpolates a curve through the points specified. (The `showpoints=true` option can help you adjust things here.)
- `\psccurve(0,0)(.5,0)(1,3)(-1,4)` — This interpolates a *closed* curve through the points.

The `linearc` option:

For `\psline` and `\pspolygon`, the `linearc` option rounds the corners of your objects (default is 0pt):



`linearc=8pt`

Example

Open the second example file (`.tex`).

Make sure your Build Profile is `LaTeX => PS => PDF`.

Build and view.

Modifications

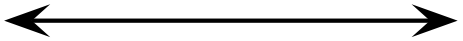
There are some easy ways to modify these basic commands. These are optional arguments.

- `linewidth=1.5pt` — changes the thickness of the lines you draw; default is 0.8 pt;
- `linecolor=red` — changes the color of your line; default is black;
- `fillstyle=solid`, `fillcolor=blue` — applies to objects that can be filled; default `fillstyle` is none; you must specify a `fillcolor` when you want a solid `fillstyle`; see posted document for `fillstyles` and `linestyles`
- `\pscircle*(0,0){1}` — all graphics objects have a starred version which draws a solid object of color `linecolor`

Arrows

On any line or curve you can draw arrows, and `pstricks` has *a lot* of options for arrowhead styles. (See posted document.) These are given with the `arrows` option. Here are a few:

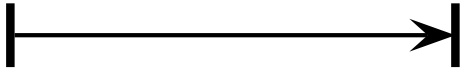
`arrows=<->`



`arrows=*-o`

`arrows=<<-<`

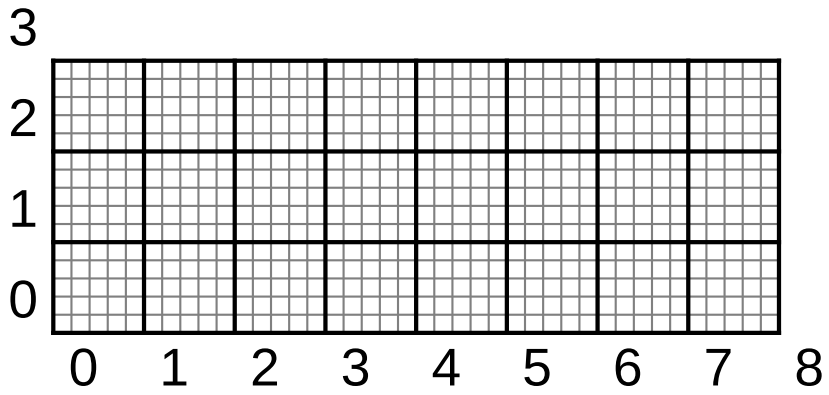
`arrows=|->|`



Grids

A powerful way to make grids:

`\psgrid(x0,y0)(x1,y1)(x2,y2)` — draws a grid with opposing corners at $(x1, y1)$ and $(x2, y2)$; the numbering starts at $(x0, y0)$.




```
\begin{center}
\begin{pspicture}(0,0)(8,3)
\psgrid(0,0)(0,0)(8,3)
\end{pspicture}
\end{center}
```

Options for Grids

Here are some ways to change grids.

Option	Default
gridcolor	black
gridlabels	10 pt
subgriddiv	5

The unit dimensions affect psgrid a great deal, but they affect everything else too. The default unit is 1 cm, but this can all be changed.

xunit	horizontal units
yunit	vertical units
runit	radial units
unit	changes all 3 at once

Example:

```
\psgrid[unit=1in,gridlabels=8pt,
  subgriddiv=8](0,0)(0,0)(8,4)
```

Framed Boxes

There are a number of options for framing text:

- `\psframebox` — draws a simple box around your text; the *-version is filled with `fillcolor` rather than `linecolor` as in geometric objects;
- `\psdblframebox` — draws a double frame;
- `\psshadowbox` — draws a single frame with a shadow;
- `\pscirclebox` — draws a circle around your text;
- `\psovalbox` — draws an oval around your text.

Options for Framed Boxes

Here are some options you might want to use:

- `framesep` — distance between each side of a frame and what's inside; default is 3pt;
- `boxsep` — true/false; should LaTeX space things according to the frame or the stuff inside the frame?
- `framearc` — rounds the corners on `\psframebox`; this needs to be a number between 0 and 1, higher is more rounded;

Example: This is nice text.

```
\psframebox[framesep=8pt,framearc=.6]{nice text}.
```

Example

Open the third example file (`.tex`).

Build and view.

Using `psset`

If you want your lines to be 2pt thick throughout your document, it is going to be a pain for you to type

```
linewidth=2pt
```

on every object you draw. Instead, type

```
\psset{linewidth=2pt}
```

before your `pspicture` environment and this sets the option *globally*.

Example:

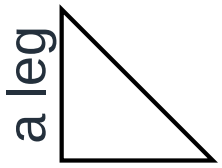
```
\psset{unit=.5cm,linewidth=1pt,linecolor=blue}
```

Placing Objects and Text

Inside a `pspicture` environment, graphics objects are drawn, but text needs to be placed. This is done using these two main commands:

- `\rput*[refpoint]{rot}(x,y){stuff}`
  - `refpoint` is the center of the box by default, but it could be `l`, `r`, `t`, `b`, or a combination like `lt`.
- `\uput*{labelsep}[refangle]{rot}(x,y){stuff}` — This places *stuff* a distance *labelsep* from (x, y) in the direction *refangle*. It is useful for labeling things.

```
\pspolygon(0,0)(1,0)(0,1)
\rput{90}(-.2,.5){a leg}
```



Let's practice!

Open the fourth example file (`.pdf`) and reproduce it.

You only need to reproduce the objects/text in the grid. (But you do not need to reproduce the grid itself or the text below the grid.)

Color

The Default Colors

The pstricks package recognizes 11 colors by name:

black	darkgray	gray	lightgray
white	red	green	blue
cyan	magenta	yellow	

So in addition to using this to color pstricks objects (fillcolor=green), you can use this to color text if you like.

```
...use this to \textcolor{red}{color text} if...
```

There are several methods available to define new colors; we'll focus on two of them.

- grayscale colors
- RGB colors

Defining New Colors

Here are the two methods.

- `\newgray{color}{num}` — This gives the name *color* to the gray created. *num* needs to be a number between 0 (black) and 1 (white).

  Example:

  `.33`

- `\newrgbcolor{color}{num1 num2 num3}` — This gives the name *color* to the color resulting from *num1* red, *num2* green, and *num3* blue. These should all be numbers between 0 and 1.

  Example:

  `.5 0 .5`

Matching Colors

What do you do if you want to match a nice color you've found somewhere?

1. Find an image of it somewhere. If it's a web page, PrintScreen, open Paint, and Paste.
2. Open the image in MSPaint. (This is a really low-budget image editor; nicer ones have this function and more.)
3. Choose the eye-dropper tool and click on the area of the picture that has the color you want.
4. Go to the menu Colors, then Edit Colors, then click on Define Custom Colors.
5. Your desired color should be selected and its RGB values are listed.
6. To use in `pstricks`, divide each RGB value by 255 to get the proper decimal.

An Example

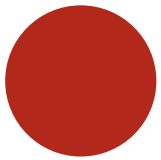
Suppose I want to capture the color maroon in the W&J logo. (See posted image file.)

We find:

Red	178
Green	41
Blue	27

```
\newrgbcolor{wjred}{.698 .16 .106}
```

Now I can type in this color or draw a circle filled with this color!



Let’s practice!

Open the fifth example file (`.pdf`) and reproduce it.