Simple Introduction to PostScript
Scott D. Anderson

PostScript® is 2-D graphics language, developed by Adobe Systems (`www.adobe.com`). It's an unofficial standard for printer languages; all the printers in this department have rendering engines for PostScript. Many typesetting languages produce PostScript (PS) as their output, including the one that I used, called LaTeX. Very often, Windows and Macintosh applications have an option under the "Print" menu to print the result to a file; that file is usually PostScript code. Many fonts are defined by PostScript, and even those that aren't are often based on outlines defined by Bézier curves, just as PostScript fonts are. (If you're interested in learning more about fonts, try `http://www.microsoft.com/typography/default.asp` and `http://www.truetype.demon.co.uk/`.) PostScript is also the basis for Portable Document Format (PDF), which many documents are distributed with. (For example, the IRS distributes forms in PDF.)

PostScript (PS) files are plain ASCII, like C++, and so they can be created by any text editor, such as vi or Emacs. Not too many people program in PS; it's mostly written by programs. In that respect, it's a little like assembly language. However, it's a lot more powerful and fun than assembly language.

From the class directory, copy the file `spelman-logo.eps`. EPS is a standard abbreviation for "embedded PostScript." EPS is designed to be modular, so that EPS file can be plugged into other PS files. For example, I can include that figure in this one: .

You can view a PS file by either sending it to a printer (which wastes paper) or using a free PS rendering engine called "GhostScript." A nice X-windows interface to GhostScript is called "GhostView," and the program is called simply, `ghostview`. Run it on the `spelman-logo.eps` file. Many of the buttons won't have any effect (because the PS commands in the file override them), but you can play around with magnification. Try running `ghostview` on the file for this handout, too; the file is `PS-handout.ps`.

Now, open up the `spelman-logo.eps` file in an editor (vi or Emacs) and read the code. The first few lines are comments. Some of those comments are actually used by other programs, such as the Bounding Box comment. A bounding box is the smallest rectangle that the figure fits inside; that makes it easier to figure out how much space to allocate for the figure, when it's inserted in another graphic. Change the width and height to 300 each and reload the file into `gv`.

The PS language is a stack-oriented, post-fix language, which means that functions get their arguments from the stack and push their results onto the stack. Thus, in the text of a program, the function comes *after* its arguments, but you also have to be aware of what stuff is lying around on the stack.

The `gsave` function takes no arguments and it saves the current graphics state. This means we can restore afterwards (see the end of the file) so that if we are embedded in another PS file, we won't mess anything up.

The next line sets the line width to 2 points; there are 72 points to an inch. Try setting it to a different value and reload the file.

The next line scales the x and y axes by one-half, shrinking all the distances. Try changing the first number to 2. What does that do?

The PS coordinate system has the origin at the lower left, with x increasing to the right and y increasing up the page. To create a series of line segments, we use the `moveto` and `lineto` functions, but that doesn't put anything on the page. All it does is define a *path*. Once the path is defined (modeled), it can be stroked (like a stroke of the pen), filled (with a solid color or an image); we can even have text move along that path.

Using graph paper, try to plot the points and see how the program works. Try modifying the points in the file and see what results you get. Try changing the word `stroke` to `fill` and see what you get.

Try adding the following to the file. What do you get?

```
/Helvetica findfont 48 scalefont setfont
10 100 moveto
(SC) show
newpath
10 150 moveto
(SC) false charpath
0.5 setgray
fill
```

Symbols are denoted with a slash and strings are delimited with parentheses. The normal way to display a font is `show`, but you can do special effects with `charpath`; it gives you a path (as we did above), which can be stroked or filled or whatever. However, if you zoom way in (try right-clicking on the region), you'll see that one scales better than the other.

Try zooming in on the triangle. Notice the sharp corners. Try putting the following code into the file before drawing the triangle:

```
1 setlinejoin
```

Change the 1 to a 2; what happens then? Change it to a zero; what does that do?

We don't have time for you to learn very much PS, but we can take a look at some examples and do some fiddling around. If you'd like to learn more, see me.