APPENDIX 4

Simple text display

You will very often want to put text in your figures. PostScript's font-handling capabilities are extremely good, but most of the techniques for high quality font management are designed to be automated by some other program, because good text—especially mathematical text—requires a lot of computation to get font choices, spacing, and sizes right.

In this Appendix I'll explain how to place simple text in PostScript figures. and also a few playful possibilities. What I explain here will be adequate for many purposes. (See Appendix 7 for more sophisticated techniques.)

1. Simple PostScript text

The simplest, essentially the only simple, way to put text into pictures is to use the almost universally available PostScript fonts to assemble your text 'by hand', i.e. by thinking out for yourself what layout, font choice, and text size are to be. This is a relatively straightforward process, and probably your best choice, if the text you want to include is not too complicated.

There are three steps to be carried out each time you want to use a new font.

(1) You must decide which font you are going to use. There is a limited choice of fonts guaranteed to be available
fonts:PostScript:1 in all environments. The choice is, roughly, from this list:

Times-Roman	Times-Italic	Times-Bold	Times-Bolditalic
Helvetica	Helvetica-Oblique	Helvetica-Bold	Helvetica-Bold-Oblique
Courier	Courier-Oblique	Courier-Bold	Courier-Bold-Oblique
Symbol			

The Helvetica fonts have no serifs, and display well on a computer. The Courier fonts have uniform character
findfont:1 spacing. The Symbol font contains Greek letters. You load a font with the command `findfont` applied to the name of the font.

scalefont:1 (2) You decide what scale you are going to use it at. With the command `scalefont`, you set what is essentially the vertical size of the letters in terms of the current unit.

setfont:1 (3) You apply the command `setfont`.

If the current unit is one inch, for example, this will give you letters $1/4''$ high:

```
/Helvetica-Bold findfont
0.25 scalefont
setfont
```

The command `findfont` loads the font named onto the stack. The command `scalefont` sets the size of the font on the stack, leaving the font there. The command `setfont` sets the current font to the one on the top of the stack. The current font is part of the graphics state, so it is affected by the `gsave` and `grestore` commands.

Of course, you can switch back and forth among several fonts. If you are going to do this, you will probably want to write procedures to do this efficiently, rather than having to go through the whole sequence above.

You can use other PostScript fonts, too, but you ought to include them explicitly in your file. I'll say something about that later. I'll also explain later how to display all the characters in a font.

To put text on a page, after you have set a current font, you move to where you want the text to begin, and then
show:1 use the `show` command. The text itself is made into a **string** by enclosing it within parentheses. Thus

```
0 0 moveto
(Geometry) show
```

after the previous command sequence will give you

Geometry

You can print out the value of a variable on the page. To do this, you must convert the variable to a string with the command cvs, using an empty string of sufficient size to hold the variable's value.

riables:displaying:2 cvs:2

```
0 0 moveto
(x = ) show
x (       ) cvs show
```

will produce this, if $x = 3$:

x = 3

A font is part of the graphics state, and the way it's displayed depends entirely on the current coordinate system. You can understand exactly what happens if you keep this in mind:

- *PostScript treats letters as paths.*

Scaling of a font is in terms of the current units. Changing the scale of the entire figure will also affect a font's true size, along with the size of everything else. But you can also shear a font or reverse it by suitable concat operations. You can get interesting effects.

Geometry

2. Outline fonts

font encryption:2

If characters are paths, you should be able to do with them all the things you can do with ordinary paths. This is almost true—the only exception is that often the inner details of fonts are hidden from close inspection. This is for legal reasons. Characters from a font, along with all other images, are subject to copyright, but because of their high reproducibility they are often encrypted. On the other hand, there are a lot of high quality fonts which are in the public domain and unencrypted.

The command show applied to a string fills in a special way the path generated by the string. But you can access the path itself by using the command charpath, which appends the path of the string to the current path. You could then fill it, but for that task this wouldn't be a very efficient way of proceeding. One more interesting thing you can do, however, is stroke it or clip to it. The command charpath takes two arguments, a string and a boolean value. Use true if you intend to stroke the outline, false if you intend to fill or clip. It is also a good idea to set the values of linejoin and linecap to something other than 0.

```
1 setlinejoin
1 setlinecap

/Helvetica-Bold findfont
48 scalefont
setfont

gsave
newpath
0 0 moveto
(ABC) false charpath
clip

1 0 0 setrgbcolor
newpath
2 setlinewidth
-24 8 96 {
  /i exch def
  i 0 moveto
  40 40 rlineto
} for
stroke
grestore

newpath
0 0 moveto
(ABC) true charpath
0.5 setlinewidth
stroke
```