to be placed. This macro will place pictures from the Pictures window—to place pictures from EPS or PICT files, you would replace the `picture` keyword with `illustration` or `pictfile`, respectively.

PostScript Graphics and Special Effects

As we've seen, you can use the `\special` command to include Encapsulated PostScript illustrations; you can also include PostScript instructions directly in your Textures documents with the `\special{postscript...}` and `\special{rawpostscript...}` commands.

Programming in PostScript

Much like TEX itself, PostScript is an extremely versatile, general-purpose, device-independent language, though oriented to low-level graphics rather than to high-level typography. By the same token, it is not necessarily easy to use. But some functions are relatively straightforward, and give you effects that you can achieve only in PostScript. For example, PostScript instructions allow you to do the following:

- rotate portions of the page
- specify halftone screens
- fill characters with a pattern
- outline characters with any line width

Figure 9-1. Special effects with PostScript.

Although you can include PostScript instructions directly in a Textures document, you should first gain experience with the PostScript language before trying to do so. The *PostScript Language Tutorial and Cookbook* and the *PostScript Language*

Reference Manual, written by Adobe Systems, are required reading if you want to program in PostScript.

The results of direct PostScript instructions will *not* be visible in your Typeset window: any graphics or special PostScript effects will show up only in the printed output. Also be aware that no PostScript error checking by Textures is possible—PostScript commands are passed straight through to the printer for processing there. Any errors in your PostScript instructions will produce either unpredictable output or, more likely, no output at all.

Placing PostScript Commands in a Textures File

You can insert PostScript instructions directly in a Textures file with the `\special{postscript...}` command, followed by the PostScript instructions that you want to send to the printer. A simple example of such a PostScript sequence is:

```
\special{postscript newpath 0 0 72 0 360 arc stroke}
```

This draws a one-inch circle centered on the current point.

NOTE: Like all special commands, the `\special{postscript...}` command may contain TEX macros that are expanded before transmission to the printer—TEX scans through PostScript text that appears in special commands and performs its usual macro processing. While this offers great potential to the experienced programmer, some PostScript sequences that look like TEX commands may trip TEX up. Unless you know both TEX *and* PostScript quite well, it is safer to include your commands in a separate PostScript file.

For an example, see the sample file "PostScript example."

Including a PostScript File

You can send a *file* of PostScript instructions to the printer by using the `\special{postscriptfile...}` command followed by the name of the file. This is very similar to including an EPS file, but unlike an EPS file you cannot view the graphic on the screen. There is also a `rawpostscriptfile` keyword, parallel to the `rawpostscript` command introduced in the next section.

As with the `postscript` special command, Textures places `postscriptfile` instructions in an environment scaled so that there are 72 coordinate units to one inch and (0,0) is at the current TEX position.

PostScript Preamble Definitions

```
\special{prePostScript definitions}
\special{prePostScriptfile filename}
```

These commands, for sophisticated PostScript programming, add definitions to the PostScript dictionary used by Textures. The first form adds literal PostScript definitions; the second form adds the contents of an external file of PostScript code. These dictionary definitions may then be used by PostScript special commands within the typeset document, reducing the overhead of lengthy PostScript inclusions.

These PostScript preamble commands must appear on the first output page of the document.

Programmer's Note: The PostScript Environment

When you include PostScript instructions, you should be aware of the surrounding environment in which those instructions will be placed:

- `postscript`, `postscriptfile`: Textures sets up a PostScript environment that looks "normal." That is, Textures sets up a coordinate system scaled so that there are 72 coordinate units to one inch, and translates the coordinate system so that (0,0)—the origin of PostScript's coordinate grid—is at the current TEX position. The entire PostScript inclusion is bracketed by PostScript `save` and `restore` save-state operators, so that your PostScript instructions will not modify Textures' drawing environment.

- `rawpostscript`, `rawpostscriptfile`: These commands inserts PostScript instructions directly into Textures' output stream, letting you modify the Textures drawing environment.*

* Just to keep you working, Apple's QuickDraw and Adobe's PostScript use inverse coordinate grids. QuickDraw is screen-oriented, and the origin point is the upper-left corner of the screen. Textures' internal PostScript

It's best to gain familiarity with Textures' use of PostScript by examining the PostScript output from the printer driver. Textures' internal dictionary is subject to change without notice; use these commands at your own risk!

- `illustrator`: This command inserts PostScript instructions in the Adobe Illustrator idiom directly into the Textures output stream. This command is applicable only when typeset pages are being saved in Illustrator format, via the Save As... command; it is ignored otherwise. Note that other PostScript inclusions are ignored while creating Illustrator format files.

framework matches QuickDraw, so the vertical axis is inverted from the usual PostScript model.