APPENDIX 3

Structured PostScript documents

A PostScript program is just a sequence of PostScript commands to be interpreted in the order in which they are encountered. It swallows one command after another. Once it has executed a sequence of commands, it essentially forgets them. In short, the PostScript interpreter knows nothing about the global structure of your file, **pages:in a PostScript file:1** and in particular has no idea of the separate pages as individual items.

comments:document structure:1 But there are conventions which allow you to put such a structure in your file. These are called **document structure comments**. You will likely have seen these, if you ever peeked at a PostScript file produced by some other program. For example, the program `dvips` that turns the `.dvi` files produced by the mathematical **%%BoundingBox:1** typesetting program TeX into PostScript might produce a file that looks like this:

```
%!PS-Adobe-2.0
%%Creator:  dvips(k) 5.86 Copyright 1999 Radical Eye Software
%%Title:  a3.dvi
%%Pages:  2
%%PageOrder:  Ascend
%%BoundingBox:  0 0 596 842
%%DocumentFonts:  Helvetica-Bold Palatino-Roman CMTT10
%%EndComments
%DVIPSWebPage:  (www.radicaleye.com)
%DVIPSCommandLine:  dvips a3.dvi -o a3.ps
%%BeginProcSet:  texc.pro
...
%%EndProlog
%%BeginSetup
%%Feature:  *Resolution 600dpi
TeXDict begin
%%PaperSize:  letter
%%EndSetup
%%Page:  1 1
1 0 bop 0 191 a Fd(Appendix)24 b(3.)36 b(Structured)24
b(P)l(ostScript)i(documents)0 540 y Fc(A)h(PostScript)h(pr)o(ogram)f
...
%%Trailer
end
userdict /end-hook known{end-hook}if
%%EOF
```

Indeed, this is the PostScript file making up this very appendix! All those lines starting with %% are **document** **DSC:1** **structure comments** (with acronym **DSC**) meant to be interpreted by a program such as a PostScript viewer that allows it to output some information about the source of this document and allow a reader to move around in it from one page to the other, not necessarily in the order in which the pages were naturally encountered. Some of these comments are more important than the others, in particular those allowing moving around among the pages. And any PostScript program that intends to allow this sort of page-by-page viewing must follow certain conventions that make it feasible.

A PostScript program following DSC conventions should begin with a line such as

```
%!PS-Adobe-2.0
```

that tell a viewing program that the appropriate conventions have been followed. The `2.0` refers to a version number for the conventions. I have to confess that in practice a lot of slop is tolerated here, and that I just put that very line at the beginning of almost all my own PostScript programs! This may very well cause the viewer to excrete nasty looking warning messages, but these can be ignored, or even switched off by choosing options suitably.

Second, the page structure has to be indicated. Towards the beginning of the file there should be a line like

```
%%Pages: 7
```

and as each page is begun there should be a line like

```
%%Page: 1 1
```

The duplication of numbers here is not necessary, but in hand-written PostScript code is probably the right thing to do (one number indicates the real page number, the other a nominal page number).

The leading line `%!PS-Adobe-xxx` and the `%%Pages: x` and `%%Page: x x` comments are the minimal set of comments needed to guide the viewer. But in order for the viewer not to be confused, it is extremely important to make the partition into pages meaningful:

- *In a PostScript file setting up a page structure, each page must be independent of every other page, so that pages may be interpreted correctly no matter the order in which they are read.*

This means above all that no variable may be defined on one page and used on another without an independent definition. As a general rule, all procedures to be used in the program should be put in a preliminary section of the file called the **prolog**. A coordinate system should be set up anew on each page, and encapsulated within a `gsave ... grestore` pair (as I have recommended already in Chapter 1). Here is a rather simple PostScript program demonstrating these points:

```
%!PS-Adobe-2.0
%%Pages: 2
/page-begin {
  gsave
  72 dup scale
  1 72 div setlinewidth
} def
/page-end {
  grestore
  showpage
} def
%%Page: 1 1
page-begin
...
% draw something
...
page-end
%%Page: 2 2
page-begin
...
% draw something else
...
page-end
```