PostScript Printer Driver Optimization Case Study

Adobe Developer Support

Technical Note #5042

31 March 1992

Adobe Systems Incorporated

Adobe Developer Technologies
345 Park Avenue
San Jose, CA 95110
http://partners.adobe.com/

PN LPS5042

Contents

PostScript Printer Driver Optimization Case Study

1 Introduction

Adobe wants to ensure that owners of PostScript™ printers and PostScript language software get full value from their investment; therefore, we have participated with major software developers in optimization case studies. One of these vendors, Scribe Systems (formerly Unilogic Ltd.) allowed us to publish the following case study using a former version of their PostScript language driver as an example.

Scribe Systems publishes a sophisticated text-formatting program called Scribe®, which runs on a variety of mainframes and minicomputers. Scribe was one of the first programs to support PostScript language output, and Adobe has used Scribe extensively. Among other projects, camera-ready copy for the original *PostScript Language Reference Manual* and *PostScript Language Tutorial and Cookbook* were produced with Scribe.

Although Scribe's original PostScript printer driver was written very early (in 1984), its printing speed on an eight page per minute printer had always been acceptable. This optimization project began when the LPS-40 printer from Digital Equipment Corporation became available.

The architecture on the LPS-40 printer has a high degree of parallelism between the basic PostScript interpreter and the lower level graphics primitives. Imagine the PostScript interpreter building display lists that a separate processor uses to produce pages. One interesting thing is that the execution profile of where PostScript language programs spend their time changes as we speed up different parts of the PostScript interpreter and move it to new printer/processor architectures, although many optimizations to a PostScript printer driver will usually improve performance on all interpreters.

The LPS-40 printer can easily be driven at a speed of 40 pages per minute on simple documents. Output from the *enscript* program (from Adobe's TranScript™ package) is quite fast. But when we tried a sample typeset document from Scribe (mostly 12 point text), we noticed printer throughput dropping significantly. Scribe's original PostScript language driver averaged about 4.8 seconds to compute and print one page. This print speed of 12.5

ppm was far below the rated speed of the printer. Afterwards, the technical staffs of Scribe Systems and Adobe cooperated on a case study in which we learned important details about PostScript language optimization.

2 The Optimization Process

In the next several pages we will discuss and illustrate the effects of various changes made to the Scribe prolog. For reference, here is the original version of the prolog:

```
%%EndComments
% PostScript Prolog for Scribe
/BS { /SV save def} def
/ES { showpage SV restore} def
/SC { setrgbcolor} def
/RST {  100 div } def
/CVTXY { RST 792 sub abs exch RST exch} def
/FMTX matrix def
/RDF {
   WFT findfont SLT 0 eq { %ifelse
      SSZ scalefont
   }{ %else
   SSZ 0 SLT sin SLT cos div SSZ mul SSZ 0 0 FMTX astore
      makefont
   } ifelse setfont
}def
/SLT 0 def
/SI {  /SLT exch def RDF } def
/WFT /Courier findfont def
/SF { /WFT exch def RDF } def
/SSZ 10 def
/SS { /SSZ exch def RDF } def
/MT { CVTXY moveto } def
/XM { RST currentpoint exch pop moveto } def
/UL {
   gsave newpath CVTXY moveto RST dup 2 div 0 exch
      rmoveto setlinewidth RST 0 rlineto stroke grestore
}  def
/PB { /PV save def CVTXY translate pop } def
/PE { PV restore } def
/SH /show load def
/MSS { SSW RST 0 rmoveto } def
/SNS { SSW add /SSW exch def MSS } def
/MX { /SSW exch def SH MSS } def
/M2 { SH MSS MSS } def
/M { SH MSS SSW } def
/M+ { SH 1 SNS } def
/M- { SH -1 SNS } def
%%EndProlog
```

2.1 Using bind and Unrolling Nested Procedures

While investigating with an execution profiler, we found that 25% of the print time was spent in the interpreter. Our first idea was to speed up the application's PostScript language prolog by "unrolling" some definitions and applying the **bind** operator to each procedure before it is stored in the dictionary. (Note that this is a transformation that any site can make unassisted—no changes are needed to the application driver itself, only to the prolog resource file.) Here is a portion of the prolog after these modifications were made.

```
% PostScript Prolog
/BS { /SV save def } bind def
/ES { showpage SV restore } bind def
/SC { setrgbcolor } bind def
/CVTXY { .01 mul 792 sub abs exch .01 mul exch } bind def
/FMTX matrix def
/RDF {
   WFT findfont SLT 0 eq { %ifelse
      SSZ scalefont
   }{ %else
      SSZ 0 SLT sin SLT cos div SSZ mul SSZ 0 0 FMTX astore makefont
   } ifelse setfont
} def
/SLT 0 def
/SI { /SLT exch def RDF } bind def
/WFT /Courier findfont def
/SF { /WFT exch findfont def RDF } bind def
/SSZ 10 def
/SS { /SSZ exch def RDF } bind def
/MT { .01 mul 792 sub abs exch .01 mul exch moveto } bind def
/XM { .01 mul currentpoint exch pop moveto } bind def
/UL {
   gsave newpath CVTXY moveto .01 mul dup 2 div 0 exch
      rmoveto setlinewidth .01 mul 0 rlineto stroke
   grestore
} bind def
/PB { /PV save def CVTXY translate pop } bind def
/PE { PV restore } bind def
/SH /show load def
/MSS { SSW 0 rmoveto } bind def
/SNS { .01 mul SSW add dup /SSW exch def 0 rmoveto } bind def
/MX { /SSW exch .01 mul def show SSW 0 rmoveto } bind def
/M2 { show SSW 2 mul 0 rmoveto } bind def
/M { show SSW 0 rmoveto } bind def
/M+ { show .01 SSW add dup /SSW exch def 0 rmoveto } bind def
/M- { show -.01 SSW add dup /SSW exch def 0 rmoveto } bind def
%%EndProlog
```

This change resulted in an improvement of print speed to 14.5 ppm. The speedup was even noticeable on a LaserWriter®, so these changes pay off for all users of the application. Also, the prolog grew only slightly.

2.2 Using widthshow Instead of moveto show

The next changes made were more complicated. The output of the application driver looked like the following code.

```
7200 14512 MT (We)
308 MX(observe)M (today)M (not)M (a)M (victory)M
(of)M (party)M (but)M (a)M (celebration)M-
(of)M (freedom,)M (symbolizing)M (an)M (end)M (as)M (well)
SH
```

To achieve careful line justification, the driver used **show** to set a single word at a time and used the equivalent of a **moveto** between words. We hand-crafted a version of the output file using the **widthshow** operator for word sequences with constant space between words.

This results in fewer PostScript language commands being transmitted and executed, without loss of print quality. Fewer procedure calls are made, because the interpreter overhead for **show** is incurred much less often. This results in less scanner and interpreter overhead, a smaller print file, and better performance inside the inner **show** loop.

Here is the extra procedure we defined and a look at how the file changed.

```
/W { 0 32 3 -1 roll widthshow } bind def
7200 14512 MT
.08 (We observe today not a victory of party but a )W
.07 (celebration of freedom, symbolizing an end as well)W
```

The performance made a quantum leap. This version of the file printed at 28 ppm. The application had all of the necessary information to generate this code. It was simply a matter of designing the driver somewhat differently. Better crafting of the definition of the procedures and exact syntax of the generated PostScript language might yield even better results.

2.3 Coordinate Systems and Arithmetic

Further changes were then made. We changed the underlying coordinate system of the translator: We let the matrix machinery do most of the arithmetic work (eliminating the CVTXY procedure in the original prolog). Where appropriate, numeric constants were specified as real numbers rather than integers, so the interpreter wouldn't need to do type conversion at every execution. (For example, 0 0 **moveto** was replaced with 0.0 0.0 **moveto**.)

We also noticed that in the output script, when a font change occurred, an intermediate **findfont**, **makefont**, and **setfont** were performed twice, once to change the *typeface* and once to change the *size*. This is the sort of accidental mismatch between the application's world view and the PostScript language imaging model that can creep into any driver design.

The following is the final version of the prolog after these changes were applied.

```
%%EndComments
% PostScript Prolog for Scribe
% Copyright (c) 1984, 1986 UNILOGIC, Ltd.
/BS {  /SV save def 0.0 792.0 translate .01 -.01 scale } bind def
/ES { showpage SV restore } bind def
/SC { setrgbcolor } bind def
/FMTX matrix def
/OMTX matrix currentmatrix def
/RDF {
   WFT SLT 0.0 eq { %ifelse
      SSZ 0.0 0.0 SSZ neg 0.0 0.0 FMTX astore
   }{ %else
      SSZ 0.0 SLT sin SLT cos div SSZ mul SSZ 0.0 0.0 FMTX astore
   } ifelse
   makefont setfont
} bind def
/SLT 0.0 def
/SI { /SLT exch cvr def RDF } bind def
/WFT /Courier findfont def
/SF { %def
   findfont dup /WFT exch def FMTX makefont setfont
} bind def
/AF { %def
   findfont /WFT exch def /SSZ exch 100.0 mul def
   WFT SSZ 0.0 0.0 SSZ neg 0.0 0.0 FMTX astore makefont setfont
} bind def
/SSZ 1000.0 def
/SS { /SSZ exch 100.0 mul def RDF } bind def
/MT /moveto load def
/XM { currentpoint exch pop moveto } bind def
/UL { %def
   gsave newpath moveto dup 2.0 div 0.0 exch rmoveto
   setlinewidth 0.0 rlineto stroke grestore
} bind def
/PB { /PV save def translate OMTX setmatrix pop } bind def
/PE { PV restore } bind def
/SH /show load def
/MSS { SSW 0.0 rmoveto } bind def
/SNS { SSW add dup /SSW exch def 0.0 rmoveto } bind def
/MX { /SSW exch def show SSW 0.0 rmoveto } bind def
/M2 { show SSW 2.0 mul 0.0 rmoveto } bind def
/M { show SSW 0.0 rmoveto } bind def
/W { 0.0 exch 32 exch widthshow } bind def
/M+ { show 1.0 SSW add dup /SSW exch def 0.0 rmoveto } bind def
/M- { show -1.0 SSW add dup /SSW exch def 0.0 rmoveto } bind def
%%EndProlog
% ...
%%Page: 1 1
BS
15 /Times-Bold AF
27182 8205 MT (Kennedy’s)SH
27390 9989 MT (Inaugural)SH
27974 11773 MT (Address)SH
12 /Times-Roman AF
```

```
7200 14512 MT 8 (We observe today not a victory of party but a )W
7 (celebration of freedom, symbolizing an end as well)W
7200 15709 MT
29
(as a beginning, signifying renewal as well as change. For I have
sworn before you )W
30 (and Almighty)SH
```

2.4 Summary

This final version runs at 34.3 pages per minute, as compared with the original speed of 12.5 ppm. The performance improvement is noticeable on all PostScript printers, not just the extremely fast printers, since the changes were made to the program's "source code."

The most important optimizations observed in this case study were

- using **bind** with procedure bodies
- eliminating unnecessary multi-level (nested) procedure calls (14.5 ppm)
- using **widthshow**, where appropriate, for line justification (28 ppm)
- carefully using the coordinate system and scaling arithmetic (34.5 ppm)

Appendix: Changes Since Earlier Versions

Changes since May 4, 1991 version

- Document was reformatted in the new document layout and minor editorial changes were made.

Changes since January 16, 1989 version

- A few layout styles were changed, trademark information was cleaned up, and a few other minor modifications were done.
- The cover addresses and phone numbers were updated.

Index