CHAPTER 1

Introduction

THE POSTSCRIPT® LANGUAGE is a simple interpretive programming language with powerful graphics capabilities. Its primary application is to describe the appearance of text, graphical shapes, and sampled images on printed or displayed pages, according to the Adobe imaging model. A program in this language can communicate a description of a document from a composition system to a printing system or control the appearance of text and graphics on a display. The description is high-level and device-independent.

The page description and interactive graphics capabilities of the PostScript language include the following features, which can be used in any combination:

- Arbitrary shapes made of straight lines, arcs, rectangles, and cubic curves. Such shapes may self-intersect and have disconnected sections and holes.
- Painting operators that permit a shape to be outlined with lines of any thickness, filled with any color, or used as a clipping path to crop any other graphic. Colors can be specified in a variety of ways: grayscale, RGB, CMYK, and CIE-based. Certain other features are also modeled as special kinds of colors: repeating patterns, smooth shading, color mapping, and spot colors.
- Text fully integrated with graphics. In the Adobe imaging model, text characters in both built-in and user-defined fonts are treated as graphical shapes that may be operated on by any of the normal graphics operators.
- Sampled images derived from natural sources (such as scanned photographs) or generated synthetically. The PostScript language can describe images sampled at any resolution and according to a variety of color models. It provides a number of ways to reproduce images on an output device.

- A general coordinate system that supports all combinations of linear transformations, including translation, scaling, rotation, reflection, and skewing. These transformations apply uniformly to all elements of a page, including text, graphical shapes, and sampled images.

A PostScript page description can be rendered on a printer, display, or other output device by presenting it to a PostScript interpreter controlling that device. As the interpreter executes commands to paint characters, graphical shapes, and sampled images, it converts the high-level PostScript description into the low-level raster data format for that particular device.

Normally, application programs such as document composition systems, illustrators, and computer-aided design systems generate PostScript page descriptions automatically. Programmers generally write PostScript programs only when creating new applications. However, in special situations a programmer can write PostScript programs to take advantage of capabilities of the PostScript language that are not accessible through an application program.

The extensive graphics capabilities of the PostScript language are embedded in the framework of a general-purpose programming language. The language includes a conventional set of data types, such as numbers, arrays, and strings; control primitives, such as conditionals, loops, and procedures; and some unusual features, such as dictionaries. These features enable application programmers to define higher-level operations that closely match the needs of the application and then to generate commands that invoke those higher-level operations. Such a description is more compact and easier to generate than one written entirely in terms of a fixed set of basic operations.

PostScript programs can be created, transmitted, and interpreted in the form of ASCII source text as defined in this book. The entire language can be described in terms of printable characters and white space. This representation is convenient for programmers to create, manipulate, and understand. It also facilitates storage and transmission of files among diverse computers and operating systems, enhancing machine independence.

There are also binary encoded forms of the language for use in suitably controlled environments—for example, when the program is assured of a fully transparent communications path to the PostScript interpreter. Adobe recommends strict adherence to the ASCII representation of PostScript programs for document interchange or archival storage.

1.1 About This Book

This is the programmer's reference for the PostScript language. It is the definitive documentation for the syntax and semantics of the language, the imaging model, and the effects of the graphics operators.

- Chapter 2, "Basic Ideas," is an informal presentation of some basic ideas underlying the more formal descriptions and definitions to come in later chapters. These include the properties and capabilities of raster output devices, requirements for a language that effectively uses those capabilities, and some pragmatic information about the environments in which the PostScript interpreter operates and the kinds of PostScript programs it typically executes.
- Chapter 3, "Language," introduces the fundamentals of the PostScript language: its syntax, semantics, data types, execution model, and interactions with application programs. This chapter concentrates on the conventional programming aspects of the language, ignoring its graphical capabilities and use as a page description language.
- Chapter 4, "Graphics," introduces the Adobe imaging model at a device-independent level. It describes how to define and manipulate graphical entities—lines, curves, filled areas, sampled images, and higher-level structures such as patterns and forms. It includes complete information on the color models that the PostScript language supports.
- Chapter 5, "Fonts," describes how the PostScript language deals with text. Characters are defined as graphical shapes, whose behavior conforms to the imaging model presented in Chapter 4. Because of the importance of text in most applications, the PostScript language provides special capabilities for organizing sets of characters as fonts and for painting characters efficiently.
- Chapter 6, "Device Control," describes how a page description communicates its document processing requirements to the output device. These include page size, media selection, finishing options, and in-RIP trapping.
- Chapter 7, "Rendering," details the device-dependent aspects of rendering page descriptions on raster output devices (printers and displays). These include color rendering, transfer functions, halftoning, and scan conversion, each of which is device-dependent in some way.

- Chapter 8, “Operators,” describes all PostScript operators and procedures. The chapter begins by categorizing operators into functional groups. Then the operators appear in alphabetical order, with complete descriptions of their operands, results, side effects, and possible errors.

The appendices contain useful tables and other auxiliary information.

- Appendix A, “LanguageLevel Feature Summary,” summarizes the ways the PostScript language has been extended with new operators and other features over time.
- Appendix B, “Implementation Limits,” describes typical limits imposed by implementations of the PostScript interpreter—for example, maximum integer value and maximum stack depth.
- Appendix C, “Interpreter Parameters,” specifies various parameters to control the operation and behavior of the PostScript interpreter. Most of these parameters have to do with allocation of memory and other resources for specific purposes.
- Appendix D, “Compatibility Strategies,” helps PostScript programmers take advantage of newer PostScript language features while maintaining compatibility with the installed base of older PostScript interpreter products.
- Appendix E, “Character Sets and Encoding Vectors,” describes the organization of common fonts that are built into interpreters or are available as separate software products.
- Appendix F, “System Name Encodings,” assigns numeric codes to standard names, for use in binary-encoded PostScript programs.
- Appendix G, “Operator Usage Guidelines,” provides guidelines for PostScript operators whose use can cause unintended side effects, make a document device-dependent, or inhibit postprocessing of a document by other programs.

The book concludes with a Bibliography and an Index.

The enclosed CD-ROM contains the entire text of this book in Portable Document Format (PDF).

1.2 Evolution of the PostScript Language

Since its introduction in 1985, the PostScript language has been considerably extended for greater programming power, efficiency, and flexibility. Typically, these language extensions have been designed to adapt the PostScript language to new imaging technologies or system environments. While these extensions have introduced significant new functionality and flexibility to the language, the basic imaging model remains unchanged.

Extensions are organized into major groups, called *LanguageLevels*. Three LanguageLevels have been defined, numbered 1, 2, and 3. Each LanguageLevel encompasses all features of previous LanguageLevels as well as a significant number of new features. A PostScript interpreter claiming to support a given LanguageLevel must implement all features defined in that LanguageLevel and lower. Thus, for example, a feature identified in this book as "LanguageLevel 2" is understood to be available in all LanguageLevel 3 implementations as well.

This book documents the entire PostScript language, which consists of three distinct groups of features. Features that are part of the LanguageLevel 2 or LanguageLevel 3 additions are clearly identified as such. Features that are not otherwise identified are LanguageLevel 1.

A PostScript interpreter can also support extensions that are not part of its base LanguageLevel. Some such extensions are specialized to particular applications, while others are of general utility and are candidates for inclusion in a future LanguageLevel.

The most significant special-purpose extension is the set of features for the Display PostScript® system. Those features enable workstation applications to use the PostScript language and the Adobe imaging model for managing the appearance of the display and for interacting with the workstation's windowing system. The Display PostScript extensions were documented in the second edition of this book but have been removed for this edition. Further information is available in the *Display PostScript System* manuals.

Appendix D describes strategies for writing PostScript programs that can run compatibly on interpreters supporting different LanguageLevels. With some care, a program can take advantage of features in a higher LanguageLevel when available but will still run acceptably when those features are not available.

1.3 LanguageLevel 3 Overview

In addition to unifying many previous PostScript language extensions, LanguageLevel 3 introduces a number of new features. This section summarizes those features, for the benefit of readers who are already familiar with LanguageLevel 2.

- *Functions.* A PostScript function is a self-contained, static description of a mathematical function having one or more arguments and one or more results.
- *Filters.* Three filters have been added, named **FlateDecode**, **FlateEncode**, and **ReusableStreamDecode**. Some existing filters accept additional optional parameters.
- *Idiom recognition.* The **bind** operator can find and replace certain commonly occurring procedures, called *idioms*, typically appearing in application prologs. The substituted procedure achieves equivalent results with significantly improved performance or quality. This enables LanguageLevel 3 features to work in applications that have not yet been modified to use those features directly.
- *Clipping path stack.* The **clipsave** and **cliprestore** operators save and restore just the clipping path without affecting the rest of the graphics state.
- *Color spaces.* Three color spaces have been added: **CIEBasedDEF** and **CIEBased–DEFG** provide increased flexibility for specifying device-independent colors; **DeviceN** provides a means of specifying high-fidelity and multitone colors.
- *Color space substitution.* Colors that have been specified in **DeviceGray**, **DeviceRGB**, or **DeviceCMYK** color spaces can be remapped into CIE-based color spaces. This capability can be useful in a variety of circumstances, such as for redirecting output intended for one device to a different one or for producing CIE-based colors from an application that generates LanguageLevel 1 output only (and thus is unable to specify them directly).
- *Smooth shading.* It is now possible to paint with a color that varies smoothly over the object or region being painted.
- *Masked images.* A sampled image can be clipped by a mask as it is painted. The mask can be represented explicitly or encoded with a color key in the image data. This enables the background to show through parts of the image.
- *CID-keyed fonts.* This font organization provides a convenient and efficient means for defining multiple-byte character encodings and for creating base fonts containing a very large number of character descriptions.

- *Font formats*. Support has been added for additional types of base fonts, including CFF (Compact Font Format), Chameleon®, TrueType™, and bitmap fonts.
- *Device setup*. There are many additional page device parameters to control colorant selection, finishing options, and other features. Any device can now produce arbitrary separations, even in a monochrome printing system (which can mark only one colorant at a time).
- *In-RIP trapping*. Certain products support *trapping*, which is the automatic generation of overlaps to correct for colorant misregistration during the printing process.
- *Color rendering intent*. A PostScript program can specify a *rendering intent* for color reproduction, causing automatic selection of an appropriate CIE-based color rendering dictionary.
- *Halftones*. Several standard halftone types have been added. They include 16-bit threshold arrays and more flexible tiling organizations for improved color accuracy on high-resolution devices. Halftone supercells increase the number of gray levels achievable on low-resolution devices.

1.4 Related Publications

A number of publications related to this book are listed in the Bibliography; some notable ones are mentioned here. For more details, see the Bibliography.

1.4.1 The Supplement

The *PostScript Language Reference Supplement* documents PostScript language extensions that are available in certain releases of Adobe PostScript® software. A new edition of the *Supplement* is published along with each major release of Adobe PostScript software.

The *Supplement* documents three major classes of extensions:

- New PostScript language features that have been introduced since the most recent LanguageLevel and that are candidates for inclusion in the next LanguageLevel.
- Extensions for controlling unique features of products, such as communication parameters, print engine options, and so on. Certain PostScript language features, such as **setdevparams**, **setpagedevice**, and the named resource facility,

are designed to be extended in this way. Although the framework for this is a standard part of the PostScript language, the specific extensions are product-dependent.

- LanguageLevel 1 compatibility operators, principally in the **statusdict** dictionary. Those features were the LanguageLevel 1 means for controlling unique features of products, but they have been superseded. They are not formally a part of the PostScript language, but many of them are still supported in Adobe PostScript interpreters as a concession to existing applications that depend on them.

1.4.2 Font Formats

PostScript interpreters support several standard formats for font programs, including Adobe Type 1, CFF (Compact Font Format), TrueType, and CID-keyed fonts. The PostScript language manifestations of those fonts are documented in this book. However, the specifications for the font files themselves are published separately, because they are highly specialized and are of interest to a different user community. A variety of Adobe publications are available on the subject of font formats, most notably the following:

- *Adobe Type 1 Font Format* and Adobe Technical Note #5015, *Type 1 Font Format Supplement*
- Adobe Technical Note #5176, *The Compact Font Format Specification*
- Adobe Technical Note #5012, *The Type 42 Font Format Specification*
- Adobe Technical Note #5014, *Adobe CMap and CID Font Files Specification*

1.4.3 Document Structure

Some conventions have been established for the structure of PostScript programs that are to be treated as documents. Those conventions, while not formally part of the PostScript language, are highly recommended, since they enable interoperability with applications that pay attention to them.

- Adobe Technical Note #5001, *PostScript Language Document Structuring Conventions Specification*, describes a convention for structuring PostScript page descriptions to facilitate their handling and processing by other programs.

- Adobe Technical Note #5002, *Encapsulated PostScript File Format Specification*, describes a format that enables applications to treat each other's output as included illustrations.

1.4.4 Portable Document Format (PDF)

Adobe has specified another format, PDF, for portable representation of electronic documents. PDF is documented in the *Portable Document Format Reference Manual.*

PDF and the PostScript language share the same underlying Adobe imaging model. A document can be converted straightforwardly between PDF and the PostScript language; the two representations produce the same output when printed. However, PDF lacks the general-purpose programming language framework of the PostScript language. A PDF document is a static data structure that is designed for efficient random access and includes navigational information suitable for interactive viewing.

1.5 Copyrights and Trademarks

The general *idea* of using a page description language is in the public domain. Anyone is free to devise his or her own set of unique commands that constitute a page description language. However, Adobe Systems Incorporated owns the copyright for the list of operators and the written specification for Adobe's PostScript language. Thus, these elements of the PostScript language may not be copied without Adobe's permission. Additionally, Adobe owns the trademark "PostScript," which is used to identify both the PostScript language and Adobe's PostScript software.

Adobe will enforce its copyright and trademark rights. Adobe's intentions are to:

- Maintain the integrity of the PostScript language standard. This enables the public to distinguish between the PostScript language and other page description languages.
- Maintain the integrity of "PostScript" as a trademark. This enables the public to distinguish between Adobe's PostScript interpreter and other interpreters that can execute PostScript language programs.

However, Adobe desires to promote the use of the PostScript language for information interchange among diverse products and applications. Accordingly, Adobe gives permission to anyone to:

- Write programs in the PostScript language.
- Write drivers to generate output consisting of PostScript language commands.
- Write software to interpret programs written in the PostScript language.
- Copy Adobe's copyrighted list of commands to the extent necessary to use the PostScript language for the above purposes.

The only condition of such permission is that anyone who uses the copyrighted list of commands in this way must include an appropriate copyright notice. This limited right to use the copyrighted list of commands does not include a right to copy this book, other copyrighted publications from Adobe, or the software in Adobe's PostScript interpreter, in whole or in part.

The trademark PostScript® (or a derivative trademark, such as PostScript® 3™) may not be used to identify any product not originating from or licensed by Adobe. However, it is acceptable for a non-Adobe product to be described as being PostScript-compatible and supporting a specific LanguageLevel, assuming that the claim is true.