POSTSCRIPT™
Software From Adobe

THE DISPLAY POSTSCRIPT™ SYSTEM

A Technical Overview for Developers
of X Window System™ Applications

September 1992

Table of Contents

The Display PostScript System

The Display PostScript™ system solves the WYSIWYG problem.

The graphical computer desktop, though it boosts user productivity, significantly increases development time and costs. Developers of X applications can use toolkits to help them image on different displays, but the toolkits do not solve the problem of creating the same image on a high-resolution printer. WYSIWYG has led users to expect their computer display to look as good as their hard copy and match it line for line. The Display PostScript system helps you provide your end users with true WYSIWYG.

The Display PostScript system is an advanced graphics software technology that extends the full power and flexibility of the PostScript language to the computer screen. The Display PostScript system helps you develop graphical applications for the screen that are portable and use the same PostScript language code you write for the printer. The PostScript imaging model frees you from display-specific details such as screen resolution and available colors.

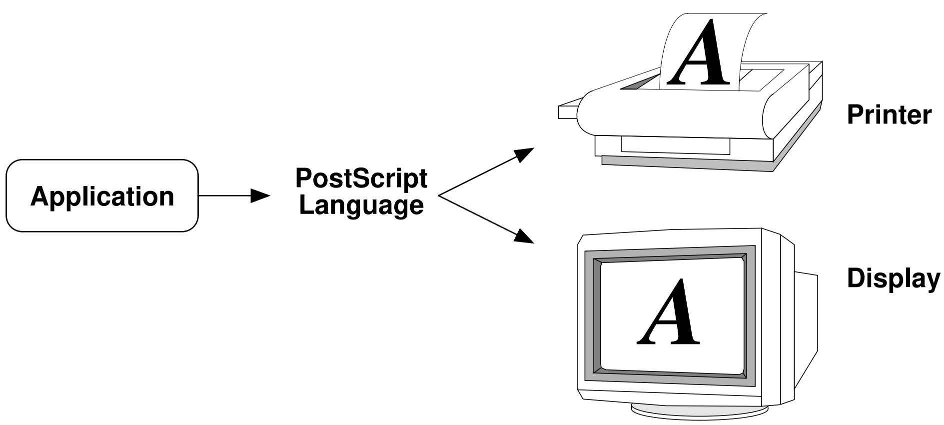

Figure 1. The same PostScript language images on the page and on the screen

In the hard copy world, the PostScript language allows one printer driver to support many devices with differing characteristics. The PostScript interpreter produces the best quality output possible for a given device, and does it in a device-independent manner. You have to master only one imaging model to communicate with a wide range of printer and imagesetter architectures. The same PostScript imaging model, featuring the same PostScript interpreter, is available for the screen in the Display PostScript system.

For developers of X applications, the Display PostScript system has been implemented as an extension to the X Window System™ and as a host-based program.

Making the Most of the PostScript Language

A Display PostScript application uses the PostScript language for text and imaging in application windows. The PostScript language makes your imaging device independent. And the Display PostScript system lets you take advantage of device independence on screens as well as on printing devices.

Your application can display its graphics on any computer where the Display PostScript system is available, regardless of display architecture, pixel density, and color capability. You no longer need to use different languages to talk to different devices.

The Components of an Application

A Display PostScript application consists of two language components, which are often written by different members of a development team.

- C or another high-level language. You develop functions that an application normally performs—reading and writing files, accepting input from the keyboard and mouse, communicating with the operating system and the window system—in the programming language of your choice.

- PostScript language. You write PostScript language routines that display text and graphics and send them to the PostScript interpreter wrapped in a C language interface. In this way you can create your own custom library of graphics routines.

Building a Display PostScript Application

The imaging routines your application can call come from three sources:

- The Display PostScript Client Library.
- PostScript language routines that you create yourself.
- Toolkits.

The Display PostScript Client Library defines data structures and procedures that your application uses to send requests to the PostScript interpreter. No matter what operating system you're dealing with, you can send PostScript language programs to the interpreter in the same way, using the same procedure calls. Therefore, portability problems are minimized when you're developing an application for more than one environment.

The same PostScript language programs that display the graphic elements of your application can send imaging commands to a PostScript printer. Thus, you halve your imaging development time because you're writing code for one imaging model instead of two.

Sending PostScript Language Code to the Interpreter

Your application can communicate with the PostScript interpreter in three ways:

- By sending graphical objects
- By sending imported PostScript language from external sources
- By calling wraps and single-operator procedures

Applications commonly need to manipulate graphics—scalable pie charts and columns of text, for example—as objects. The Display PostScript system supports efficient rendering of common graphics as objects by means of such operators as **xshow** and user path operators. The Display PostScript Toolkit provides procedures that your application can call to send graphical objects to the PostScript interpreter.

Your application can receive PostScript language code from external sources such as files and cut-and-paste selections from another application. For example, the code may represent scanned photographs or illustrations. You can send this code to the PostScript interpreter by calling procedures in the Display PostScript Client Library.

If you want to compile PostScript language procedures together with your application code, you can use a translator called *pswrap* (part of the Display PostScript Software Development Kit). Because the *pswrap* translator "wraps" each PostScript language program in a C-callable interface, the resulting procedures are known as wraps.

The following example shows a sample wrap definition, the C procedure call that you use to execute the wrap, and the PostScript language code that the wrap sends to the interpreter.

Wrap definition:

```
defineps PSWGrayCircle(float x, y, radius)
    newpath
    x y radius 0.0 360.0 arc
    closepath
    0.5 setgray
    fill
endps
```

Procedure call:

```
PSWGrayCircle(25.4, 17.7, 40.0);
```

PostScript language code equivalent:

```
newpath
25.4 17.7 40.0 0.0 360.0 arc
closepath
0.5 setgray
fill
```

Putting It All Together

Your application source code contains calls to wraps, toolkit routines, and Client Library routines. Figure 2, "Creating a Display PostScript application," shows how you bring together wraps, libraries, toolkits, and application code to make an executable program.

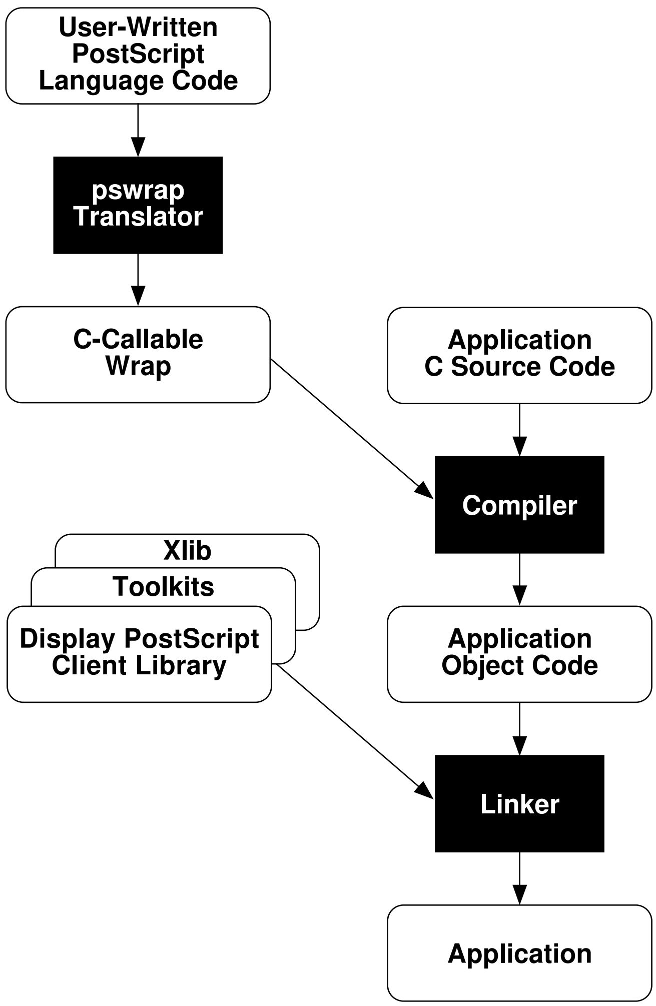

Figure 2. Creating a Display PostScript application

You compile your application source code, together with any wraps you have prepared, to produce application object code. Then you link in the Client library, Xlib, and any toolkits you are using.

The Integrated Solution—The Display PostScript Extension

The Display PostScript system is available as an extension to the X Window System. Figure 3 shows how the components of the Display PostScript extension communicate with the application and the window system when the Display PostScript extension is integrated with the X server.

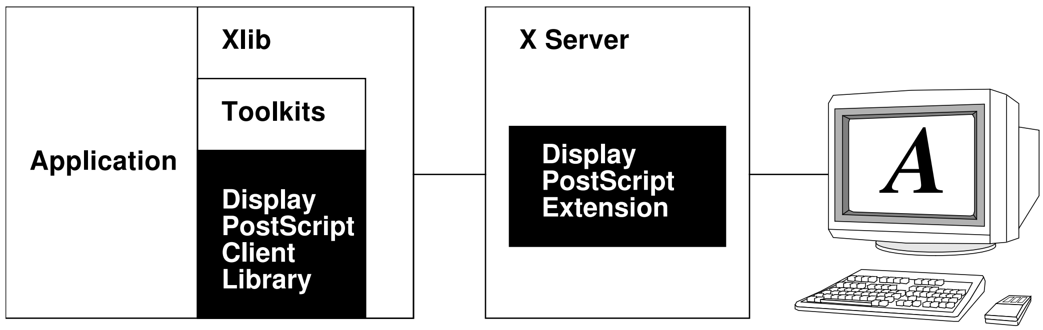

Figure 3. Information flow between an application and display through the Display PostScript extension to X

The X server and its associated window manager handle window operations such as displaying and positioning windows, while the Display PostScript extension handles imaging *within* the window. The application communicates with the Display PostScript extension and the X server as follows:

- The application sends PostScript language code to the PostScript interpreter in the Display PostScript extension and in return may receive numeric or textual information. The PostScript interpreter also sends status events to the application.

  The Display PostScript extension can also detect "hits" for the application—the PostScript language provides operators that tell the application whether or not a mouse click touches a graphic element.

- The application sends X protocol requests to the X server and in return may receive data. The X server also sends notification events to the application.

The Distributed Solution—Display PostScript NX Network Software

The Display PostScript extension to X is available on major platforms through OEMs and system integrators. However, some of your target customers may be X Window System users who don't have access to the Display PostScript extension. By bundling your application with Display PostScript NX network software, you ensure that your Display PostScript application runs on any network where X runs. The same application code that produces high-quality Display PostScript images on your display works with either the Display PostScript extension or Display PostScript NX network software.

Display PostScript NX network software is a host-based program that contains a PostScript interpreter. Instead of being integrated with the X Window System, the Display PostScript NX program communicates with the X server in the same way an application does. This arrangement lets your application use the Display PostScript system with any X server. Your Display PostScript application is not limited to an X server that implements the Display PostScript extension—it can display on networked X terminals and personal computers as well as workstations. Most users need not know whether they are using the Display PostScript extension or a Display PostScript NX program.

How Display PostScript NX Works

The Display PostScript NX program (shown in Figure 4) works in a similar manner to the Display PostScript extension to X portrayed in Figure 3. The application sends PostScript language code to the Display PostScript NX program over the network. The PostScript interpreter transforms the code into X protocol and sends it to the X server for rendering in the application window.

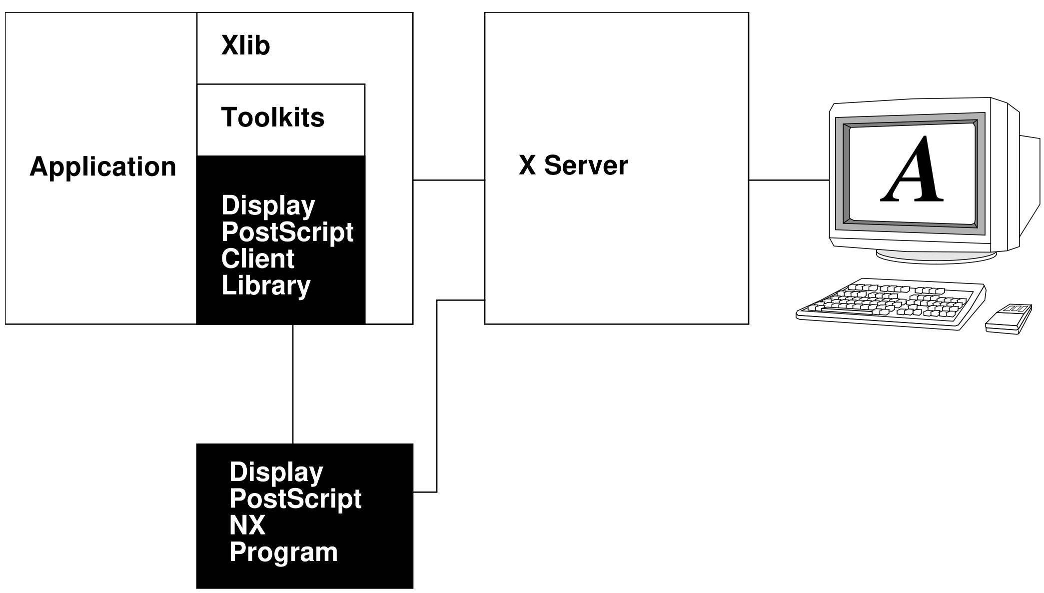

Figure 4. Information flow between an application and display through the Display PostScript NX program

When your Display PostScript application is initialized, the Client Library checks the X server to see if the Display PostScript extension is present. If there is no extension, the Client Library searches the user environment to find a Display PostScript NX program on the network and uses its PostScript interpreter. Either way, your application makes the same procedure calls, and the same PostScript interpreter rasterizes your graphics on the screen.

Special Features of the Display PostScript System

The Display PostScript system has been optimized for use with displays through extensions to the PostScript language.

- Font selection, explicit character positioning, and coordination between outline and bitmap fonts have all been optimized for display use.

- For fast data transfer and fast interpretation, Adobe has added binary encodings to the PostScript language. Client Library routines are normally sent to the interpreter in binary-encoded formats. The *pswrap* translator automatically generates binary-encoded PostScript language code.

- To conveniently update only part of a window, the Display PostScript system provides operators such as **viewclip**.

- To allow multithreading, the Display PostScript system provides multiple execution contexts. Each application's context has its own thread of control, its own set of stacks, its own private virtual memory, and its own graphics state. As the user switches between applications (and thus between contexts), each application has access to its current "state" as soon as it receives control, without cycle-wasting updates.

Development Tools and Adobe Developers Association

Adobe has created a series of Display PostScript Software Development Kits (SDKs) to assist in application development. The SDKs include a comprehensive set of tools and reference materials for the developer. Adobe offers the SDKs to help independent software vendors reduce development time and create more powerful applications. The SDKs include manuals, sample code, toolkit libraries and utilities that have been developed specifically for creating Display PostScript applications. The SDKs also include an interactive PostScript language debugger called *Displaytalk*™.

The Adobe Developers Association is an organization that supports serious development teams who are interested in building Display PostScript applications. Through the Association, developers can receive:

- 50% discounts on Adobe Systems retail application products and Display PostScript SDKs.
- Access to the Developer Support telephone line.
- Discounts of 40-60% off hardware offered by participating OEMs.
- Mailings including new developer tools and other information such as press announcements, newsletters, and technical documents.
- Access to marketing materials to assist you in showcasing your products.

Call 415-961-4111 to find out how to become a member or to order the Software Development Kit for your development platform.

The Display PostScript System—Its Advantages are Visible

Decreased development costs, increased application portability, reusable code, high performance graphics, and freedom from display-specific details: the Display PostScript system offers X Window System developers a host of advantages.