NAME

dps – Display PostScript imaging for the X Window System

DESCRIPTION

This manual page provides information about the Display PostScript system, implemented as an extension to the X Window System.

The INTRODUCTION section contains a brief, nontechnical description of the Display PostScript system.

The remaining sections provide the application developer with more detailed technical information about the architecture.

The REFERENCES section describes additional documentation and tells you how to use Adobe's public access file server.

INTRODUCTION

The PostScript language is a simple interpretive programming language with powerful graphics capabilities. Its primary purpose is to describe the appearance of text, graphical shapes, and images on printed or displayed pages. If an application, such as a word processing system or graphics package, produces a page description using the PostScript language, you can print the pages on a wide variety of PostScript printers and view them on monitors where the Display PostScript system is available.

The Display PostScript system is a high-performance implementation of the PostScript language for interactive computer displays. The use of the Display PostScript system ensures true WYSIWYG (What You See Is What You Get) between the display and any PostScript printer.

DISPLAY POSTSCRIPT SYSTEM ARCHITECTURE

The Display PostScript system is part of the X Window System and is implemented as an X extension. Display PostScript applications use window system features for window placement and sizing, menu creation, and event handling, while using Display PostScript features to take care of imaging inside the window.

Display PostScript system components include:

The PostScript interpreter.

The Client Library – a C language interface to the basic facilities of the Display PostScript system.

pswrap – a preprocessor that prepares PostScript language programs for invocation from a C program.

These components are discussed below.

APPLICATION BUILDING BLOCKS

Most of a Display PostScript application is written in C or another high-level language. It calls Client Library procedures to start a PostScript execution context, send programs and data to the PostScript interpreter, and get results from the interpreter. The Client Library is the application's primary interface to the Display PostScript system.

In addition, it calls *wraps* – custom PostScript language procedures developed specifically for the application. Wraps are generated by the *pswrap* translator from application-specific PostScript language code.

USING PSWRAP

pswrap is a preprocessor that takes PostScript language code as input and embeds it in C-callable procedures, or wraps. The output of *pswrap* is compiled and linked with the rest of your application, which can then call the wraps to transmit PostScript language code to the PostScript interpreter.

A Display PostScript application uses C or another high-level language to perform calculations, communicate with the window system, read and write files, and do other application processing. It uses wraps primarily for imaging tasks.

Consider a procedure, **PSWDisplayText**, that places text on the screen at a particular *x,y* coordinate. A call to this wrap from the application program might look something like this:

```
PSWDisplayText(72.0, 100.0, "Hello World");
```

The body of the **PSWDisplayText** procedure is actually written in the PostScript language. It was defined to *pswrap* as follows:

```
defineps PSWDisplayText(float X,Y; char *text)
        X Y moveto
        (text) show
endps
```

In the wrap definition, the **defineps** and **endps** keywords tell *pswrap* where a given PostScript language program begins and ends. The **defineps** statement defines the resulting procedure call. The *pswrap* translator processes this input and produces a C language source-code file. When compiled and linked with the application, the **PSWDisplayText** procedure sends a PostScript language program to the interpreter (binary-encoded for more efficient processing), causing "Hello World" to be displayed on the screen.

See the *Programming the Display PostScript System with X* for further information.

THE CLIENT LIBRARY

The Display PostScript Client Library is a linkable library of compiled C procedures that provides an interface between the application and the Display PostScript system. It creates an environment for handling imaging calls to specific Client Library procedures like **DPSmoveto** and to custom wraps written for the application.

To the application programmer, it appears that Client Library procedures directly produce graphical output on the display. In fact, these procedures generate PostScript language statements and transmit them to the PostScript interpreter for execution; the PostScript interpreter then produces graphical output that is displayed by device-specific procedures in the Display PostScript system. In this way, the Client Library makes the full power of the PostScript interpreter and imaging model available to a C language program.

The Client Library includes procedures for creating, communicating with, and destroying PostScript execution contexts. A context consists of all the information (or "state") needed by the PostScript interpreter to execute a PostScript language program. In the Client

Library interface, each context is represented by a **DPSContextRec** data structure. PostScript execution contexts are described in the *PostScript Language Reference Manual, Second Edition*.

REFERENCES

Information about the PostScript Language and the Display PostScript system is available in a number of manuals and via the public access file server described below.

POSTSCRIPT LANGUAGE MANUALS

If you're new to the PostScript language, you should first read the following manuals (published by Addison-Wesley and available from Adobe Systems Incorporated or through your technical bookstore):

PostScript Language Reference Manual, Second Edition

The standard reference for the PostScript language. Describes the PostScript imaging model and the concepts and facilities of the PostScript interpreter. Documents the PostScript language. Required reading.

PostScript Language Tutorial and Cookbook

Introduction to the PostScript language in an informal, interactive style. Contains a collection of example programs that illustrate the PostScript imaging model.

PostScript Language Program Design

Guidelines for the advanced developer to use in designing and debugging PostScript language programs. Printer-oriented, but most of the information is relevant to writing a Display PostScript application.

DISPLAY POSTSCRIPT MANUALS

Once you're up to speed in the PostScript language, read *Programming the Display PostScript System with X*, available from Addison-Wesley. This book is collection of manuals that explain how to render text and graphics with the Display PostScript extension to X. It contains the following manuals:

Programming Guide

Explains how to render text and graphics with the Display PostScript extension to X.

Client Library Reference Manual

Describes the procedural interface to the Display PostScript system. Tells how to send programs and data to a PostScript execution context, how to handle context output, how to create and terminate a context. Contains procedure definitions, programming tips, and a sample application program.

Client Library Supplement for X

Describes Display PostScript features that are specific to the X Window System, such as context creation and additional error codes.

pswrap Reference Manual

Describes how to define C-callable procedures that contain PostScript language programs. Tells how to declare input arguments and output to be received from the interpreter. Documents the pswrap command line options.

Display PostScript Toolkit for X

Describes the Display PostScript Toolkit for the X Window System. It also contains information about locating PostScript language resources and about the *makepsres* utility.

THE PUBLIC ACCESS FILE SERVER

Adobe Systems Incorporated provides a public access file server. If you have access to Internet or UUCP electronic mail, you can use the public access file server to obtain the following information:

Display PostScript system manuals

Code examples

AFM files

Documentation updates

The public access file server is a mail-response program. That is, you send it a request by electronic mail and it mails back a response. (The ‘‘Subject:’’ line is treated as part of the message by the file server.)

To send mail to the file server, use one of the following addresses:

Internet **ps-file-server@adobe.com**

UUCP **...!decwrl!adobe!ps-file-server**

To receive a quick summary of file server commands, send the following message:

help

To receive detailed information on how to use the file server, send the following message:

send Documents long.help

COLORMAP USAGE

The Display PostScript system uses entries from the default X colormap to display colors and grey values. You can configure this usage. Giving the Display PostScript system more colormap entries improves the quality of its rendering, but leaves fewer entries available to other applications since the default colormap is shared.

Resources in your .Xdefaults file control the colormap usage. Each resource entry should be of the form

DPSColorCube.visualType.depth.color: size

where

visualType is one of GrayScale, PseudoColor, or DirectColor.

depth is 1, 2, 4, 8, 12, or 24 and should be the largest depth equal to or less than the default depth.

color is one of the strings "reds", "greens", "blues", or "grays".

size is the number of values to allocate of that color.

These resources are not used for the static visual types StaticGray, StaticColor, or TrueColor. Specifying 0 for reds directs the Client Library to use only a gray ramp. This specification is particularly useful for gray-scale systems that incorrectly have

PseudoColor as the default visual.

For example, to configure a 5x5x4 color cube and a 17-element gray ramp for an 8-bit PseudoColor screen, specify these resources:

```
DPSColorCube.PseudoColor.8.reds: 5
DPSColorCube.PseudoColor.8.greens: 5
DPSColorCube.PseudoColor.8.blues: 4
DPSColorCube.PseudoColor.8.grays: 17
```

These resources use 117 colormap entries, 100 for the color cube and 17 for the gray ramp. For the best rendering results, specify an odd number for the gray ramp.

Resources that are not specified take these default values:

```
DPSColorCube.GrayScale.4.grays: 9
DPSColorCube.GrayScale.8.grays: 17

DPSColorCube.PseudoColor.4.reds: 2
DPSColorCube.PseudoColor.4.greens: 2
DPSColorCube.PseudoColor.4.blues: 2
DPSColorCube.PseudoColor.4.grays: 2
DPSColorCube.PseudoColor.8.reds: 4
DPSColorCube.PseudoColor.8.greens: 4
DPSColorCube.PseudoColor.8.blues: 4
DPSColorCube.PseudoColor.8.grays: 9
DPSColorCube.PseudoColor.12.reds: 6
DPSColorCube.PseudoColor.12.greens: 6
DPSColorCube.PseudoColor.12.blues: 5
DPSColorCube.PseudoColor.12.grays: 17

DPSColorCube.DirectColor.12.reds: 6
DPSColorCube.DirectColor.12.greens: 6
DPSColorCube.DirectColor.12.blues: 6
DPSColorCube.DirectColor.12.grays: 6
DPSColorCube.DirectColor.24.reds: 7
DPSColorCube.DirectColor.24.greens: 7
DPSColorCube.DirectColor.24.blues: 7
DPSColorCube.DirectColor.24.grays: 7
```

If none of the above defaults apply to the display, the Client Library uses no color cube and a 2-element gray ramp; that is, black and white.

SEE ALSO

pswrap(1), **dpsexec**(6)

NOTES