NOTE: This Technical Note has been retired. Please see the Technical Notes page for current documentation.

Technical Note PR01
Color Printing

CONTENTS

This discusses color printing in a Macintosh application.

[Feb 01 1986]

Introduction

Whereas the original eight-color model of QuickDraw was sufficient for printing in color on the ImageWriter II, the introduction of Color QuickDraw has created the need for more sophisticated printing methods.

The first section describes using the eight-color QuickDraw model with the ImageWriter II and ImageWriter LQ drivers. Since the current Print Manager does not support Color GrafPorts, the eight-color model is the only method available for the ImageWriters.

The next section describes a technique that can be used for printing halftone images using PostScript (when it is available). It also describes a device-independent technique for sending the PostScript data. This technique can be used on any LaserWriter driver 3.0 or later. It will work with all LaserWriters except the the LaserWriter IISC.

It is very likely that better color support will be added to the Print Manager in the future. Until then, these are the best methods available.

Back to top

Part 1, ImageWriters

The ImageWriter drivers are capable of generating each of the eight standard colors defined in QuickDraw by the following constants:

```
    whiteColor
    blackColor
    redColor
    greenColor
    blueColor
    cyanColor
    magentaColor
```

To generate color all you need to do is set the foreground and background colors before you begin drawing (initially they are set to `blackColor` foreground and `whiteColor` background). To do this you call the QuickDraw routines `ForeColor` and `BackColor` as described in *Inside Macintosh* . If you are using QuickDraw pictures, make sure you set the foreground and background colors before you call `ClosePicture` so that they are recorded in the picture. Setting the colors before calling `DrawPicture` doesn't work.

The drivers also recognize two of the transfer modes: `srcCopy` and `srcOr`. The effect of the other transfer modes is not well defined and has not been tested. It may be best to stay away from them.

Caveats

When printing a large area of more than one color you will encounter a problem with the ribbon. When you print a large area of one color, the printer's pins pick up the color from the back of the ribbon. When another large area of color is printed, the pins deposit the previous color onto the back of the ribbon. Eventually the first color will come through to the front of the ribbon, contaminating the second color. You can get the same kind of effect if you set, for example, a foreground color of yellow and a background color of blue. The ribbon will pick up the blue as it tries to print yellow on top of it. This problem is partially alleviated in the 2.3 version of the ImageWriter driver by using a different printing technique.

The ribbon goes through the printer rather quickly when printing large areas. When the ribbon comes through the second time the colors don't look too great.

Back to top

Part 2, LaserWriters

Using the PostScript 'image' Operator to Print Halftones

About 'image'

The PostScript image operator is used to send Bitmaps or Pixmaps to the LaserWriter. The image operator can handle depths from 1 to 8 bits per pixel. Our current LaserWriters can only image about twenty shades of gray, but the printed page will look like there's more. Being that the image operator is still a PostScript operator, it expects its data in the form of hexidecimal bytes. The bytes are represented by two ASCII characters(0-9,A-F). The image operator takes these parameters:

The first three are the width, height, and depth of the image, and the matrix is the transformation matrix to be applied to the current matrix. See the *PostScript Language Reference Manual* for more information. The image data is where the actual hex data should go. Instead of inserting the data between the first parameters and the image operator itself, it is better to use a small, PostScript procedure to read the data starting from right after the image operator. For example:

```
    640 480 8 [640 0 0 480 0 0]
    {currentfile picstr readhexstring pop}
    image
```

In the above example, the width of the image is 640, the height is 480, and the depth is 8. The matrix (enclosed in brackets) is setup to draw the image starting at QuickDraw's 0,0 (top left of page), and with no scaling. The PostScript code (enclosed in braces) is not executed. Instead, it is passed to the image operator, and the image operator will call it repeatedly until it has enough data to draw the image. In this case, it will be expecting 640*480 bytes. When the image operator calls the procedure, it does the following:

1. Pushes the current file which in this case is the stream of data coming to the LaserWriter over AppleTalk. This is the first parameter to `readhexstring`.
2. Next `picstr` is pushed. `picstr` is a string variable defined to hold one row of hex data. The PostScript to create the picstr is:

3. Now `readhexstring` is called to fill `picstr` with data from the current file. It begins reading bytes which are the characters following the image operator.
4. Since `readhexstring` leaves both the string we want, and a boolean that we don't want on the stack, we do one pop to kill of the boolean. Now the string is left behind for the image operator to use.

So using the above PostScript code you can easily print an image. Just fill in the width height and depth, and send the hex data immediately following the PostScript code.

Setting Up for 'image'

Most of the users of this technique are going to want to print a Color QuickDraw `PixMap`. Although the image command does a lot of the work for you, there are still a couple of tricks that are recommended for performance.

Assume the Maximum Depth

Since the current version of the image operator has a maximum depth of 8 bits/pixel, it is wise to convert the source image to the same depth before imaging. This can be done very simply by using an offscreen `GrafPort` that is set to 8 bits/pixel, and then using `CopyBits` to do the depth conversion for you. This will do a nice job of converting lower resolution images to 8 bits/pixel.

Build a Color Table

An 8-bit deep image can only use 256 colors. Since the image that you are starting with is probably color, and the image you get will be grayscale, you need to convert the colors in the source color table into PostScript grayscale values. This is actually easy to do using the Color Manager. First create a table that can hold 512 bytes. This is 2 bytes for each color value from 0 to 255. Since PostScript wants the values in ASCII, you need two characters for each pixel. Now loop through the colors in the color table. Call `Index2Color` to get the real RGB color for that index, and then call `RGB2HSL` to convert the RGB color into a luminance value. This value will be expressed as a `SmallFract` which can then be scaled into a value from 0 to 255. This value should then be converted to ASCII, and stored at the appropriate location in the table. When you are done, you should be able to use a pixel value as an index into your table of PostScript color values. For each pixel in the image, send two characters to the LaserWriter.

Sending the Data

Once you have set up the color table, all that left to do is to loop through all of the pixels, and send their PostScript representation to the LaserWriter. There are a couple of ways to do this. First is to use the low-level Print Manager interface and stream the PostScript using the `stdBuf PrCtlCall`. Although this seems like it would be the fastest way, the latest version of the LaserWriter driver (5.0) converts all low-level calls to their high level equivalent before executing them. Because of this, the low-level interface is no longer faster than the high level. In an FKEY I have written, I use the high-level Print Manager interface, and send the data via the `PostScriptHandle` PicComment. This way, I can buffer a large amount of data, before actually sending it. Using this technique, I have been able to image a Mac II screen in about 5 minutes on a LaserWriter Plus, and about 1.5 minutes on a LaserWriter II NTX.

Back to top

References

QuickDraw

The Printing Manager

PostScript Language Reference Manual, Adobe Systems

Back to top

Downloadables

	Acrobat version of this Note (K).	Download