Macintosh
Technical Notes

Developer Technical Support

#297: Pictures and the Printing Manager

Written by: Zz Zimmerman April 1991

This technical note described some problems and features of using Quickdraw pictures with the Printing Manager. In general, if your application prints Quickdraw pictures, you should read this note.

Introduction

Most applications support Quickdraw pictures to some degree. They will allow you to import or export picture files, as well as using the PICT resource format on the clipboard to support Cut & Paste with other applications. Unfortunately, there are some problems that occur with pictures at print time, and that's what I want to cover here.

You PICT When?

One of the problems that comes up at print time is the use of picture comments. Some applications store their data in a native format, and only create pictures at print time to enable the use of picture comments. For each page of the document, they open a new picture, record the Quickdraw calls that described the document, along with any picture comments they want to use, and finally close the picture. When this is done, they call DrawPicture to print the picture, and then start the whole process over again for the next page.

This method is supported and fully compatible with future system software, but is not required. The Printing Manager spools each page of a document into a Quickdraw picture. Since the Printing Manager already has a picture open, it is totally legal to send a picture comment (via the PicComment call) in between calls to PrOpenPage and PrClosePage without having them recorded in a picture. The Printing Manager has already replaced the StdComment procedure with its own anyway, so the PicComment call will be intercepted and supported correctly by the

Printing Manager. If the only reason you are recording into pictures is so that you can use PicComments, you can avoid the overhead at print time by simply sending the comments directly.

Feeling PICT On?

Even if you aren't sending picture comments, you may still need to create a picture at print time. In general, you should try to create any pictures you need prior to calling PrOpen. This is because there is no way to predict how much memory a particular printer driver will require. Instead, you need to make as much memory available as possible. If you are creating pictures with the Printing Manager open, the chances are good that you are using memory you can't afford to waste.

If you need to create a picture with the Printing Manager open, and memory is not a problem, you should still be aware of some potential problems. First of all, keep in mind the Printing Manager receives data from the application by replacing the Quickdraw GrafProcs stored in the GrafPort returned by PrOpenDoc. One of these procedures is the StdComment procedure which is called each time the application calls the Quickdraw PicComment routine. Since the Printing Manager has these routines patched, creating a picture in the Printing Manager's GrafPort can cause problems. If you must create a picture between calls to PrOpenPage/PrClosePage, you should be sure to set the port to a standard Quickdraw GrafPort before calling OpenPicture. Any GrafPort that was created by Quickdraw (as opposed to the Printing Manager) will work fine.

If you do create a picture at print time, you may experience a syndrome we call 'floating picture comments'. That is, calls made by your application to the PicComment routine will be recorded in a different order than you made them. This will usually cause them to effect the wrong part of the picture, and lead to endless confusion. The best solution to this problem is to create any pictures that your application will need **before** opening the Printing Manager.

Scaling Pictures - Mountains from Mole Hills

Another problem is a basic problem with pictures that seems to show up more at print time. The problem concerns the scaling of pictures using the destination rectangle passed to DrawPicture. This method will work for most pictures, but problems arise with more complex pictures, and for pictures that contain text. The problem is the method that Quickdraw uses to scale the text stored within pictures. When scaling, Quickdraw tries to handle the text scaling intelligently by changing the size of the font being used, as opposed to just scaling the bits. Unfortunately, the widths used by bitmapped fonts are not always linear (ie. the 12 point width isn't exactly 1/2 of the 24 point width). Because of this, you can run into problems with lines of text getting slightly longer or shorter as the picture is scaled. In many cases, the error is insignificant, but if you are trying to draw a line of text that fits exactly into a box (a spreadsheet cell for example), you might be surprised to see the line of text extending beyond the box when the picture is scaled.

There can also be problems when using certain picture comments or imbedded PostScript. In the case of the TextCenter picture comment, you specify an offset to the center of rotation. This offset is usually based on the metrics of the font being used. If you scale the picture, Quickdraw decides to use a different font, and the offset you originally specified will be incorrect.

The easiest way to solve these problems is to scale the picture yourself. Especially if you are trying to scale by a large amount. For example, some applications create a picture at 72 dpi (ie.

dots per inch), and then scale it to 288 dpi for printing by simply increasing the destination rectangle by 4x. This is asking a lot of the system, and will result in the text problems described above. Instead, you should either draw the picture into its original frame, and let the Printing Manager handle scaling it to the resolution of the device, or handle the scaling yourself by parsing the picture and playing it back opcode by opcode instead of calling DrawPicture.

One last thing to watch for when scaling pictures is integer overflows. It's usually pretty rare that you will overflow a coordinate when creating a Quickdraw picture, but it is not so hard to do when scaling a picture. For example, some applications will draw something offscreen to make sure the Printing Manager has configured the clip region and other related structures. They usually do this by moving the cursor to (-32767,-32767), and then draw a pixel. This works fine to initialize the Printing Manager, and the pixel isn't actually seen on the output. The problem occurs when you try to scale this picture. If you try to make it bigger, Quickdraw will adjust to coordinate (-32767,-32767) which will end up overflowing. The only way to solve these

problems is to look for these kinds of operations in the picture before trying to scale it with DrawPicture.

Pictures Within Pictures–Is Nesting the Best Thing?

One cool feature of Quickdraw pictures is the ability to nest them easily. Basically, you can call OpenPicture, and then call DrawPicture with multiple pictures, and when you call ClosePicture, they will all have been recorded into one picture. Very cool. The problem comes when you start using the Begin/End form of picture comments. There are some comments like PostScriptBegin/PostScriptEnd, and TextBegin/TextEnd that have a begin comment that is followed by an end comment. When using these comments, it is very important to make sure that you have an end for each begin that you have sent. If the nesting gets off, you will, at the least, get incorrect output, though it is more likely that the Printing Manager will actually crash. If your application is generating picture comments, it is very simple to make sure that you have an end for each begin. But when nesting a picture that you have imported from another application, it is important to know how its comments will interact with yours.

In most cases, you can simply call DrawPicture to render the picture to the Printing Manager and you don't have to worry. But if you are creating a picture for export, you may have to nest multiple pictures from multiple creators into the same picture to be exported. If this is the case, it is important to make sure that all of your begin comments have matching end comments **before** attempting to insert another picture. If this is done, you can insert the imported picture without having to worry about the comments it contains. If all of your begin/ends are matched, you can assume that the imported picture will be just as valid.

On the other hand, if you have a begin comment, and want to insert a picture before inserting the appropriate end comment, you must parse the picture to be inserted to make sure it is not using the same comment pair. If it is, and you insert it, you will have problems.

So make sure that all your begins and ends are matched, and don't try to insert other pictures between begin/end pairs of comments. If you find that you have to insert a picture between a pair of begin/end comments, you must parse the picture to be inserted to make sure that it does not use the same comments.

Penalty for Quickdraw - Clipping

Here's a subtle fact about Quickdraw pictures. If you call OpenPicture, and then record some drawing operations, and you don't explicitly specify a clipping region, Quickdraw will specify one for you. In fact, Quickdraw will use the last clip region stored in the GrafPort that you are using when you call OpenPicture. This has been a surprise to many a developer when they

record a picture, and a big piece of it ends up missing when they draw it. This isn't specific to print time, it can happen on the screen too, but it happens enough that it's worth mentioning.

D' Resolution

If you've read Technical Note #275, Features of 32 Bit Quickdraw 1.2, you probably noticed the new font and resolution information. Basically, fonts are now stored in pictures by name, not by ID. This means that fonts stored in pictures will be displayed correctly on any Macintosh without fonts remapping to other faces. The other new addition to the picture format is horizontal and vertical resolution information. Applications that create pictures using the new OpenCPicture call will be able to tell Quickdraw the native resolution of the picture data. So if

you're a scanner that is scanning at 200 dpi, you will be able to store your data at 200 dpi (instead of scaling it to 72 dpi first). When an application subsequently opens the picture, it can determine the picture's resolution and take the necessary steps to display it correctly (ie. scaling down for display on the 72 dpi screen, or scaling up for display on 300 dpi devices like the LaserWriter).

Conclusion

Quickdraw pictures can be used successfully at print time, if you avoid the problems described above. Although there is a little overhead required by some of the workarounds, most are very simple to implement, and will help you avoid future compatibility problems.

Further Reference:

- *PostScript Language Reference Manual*, Adobe Systems Inc.
- *LaserWriter Reference Manual*, Addison-Wesley

PostScript is a registered trademark of Adobe Systems Incorporated.