Technical Note PR23

Position-Independent PostScript

CONTENTS

This technical note describes a method for inserting position-independent PostScript into QuickDraw pictures.

[Mar 01 1988]

Introduction

There is a problem with pictures that contain PostScript code. Sometimes the PostScript code that is inserted into the picture is dependent on the position of the picture on the page. The problem arises when these pictures are cut or copied from their original position, and pasted into another position or even into another document. The PostScript code will not know the new location of the picture, and will not execute correctly.

The solution for this problem, is to provide some way for the PostScript code to determine the current location of the picture relative to the page that it is being printed on. This is done by inserting QuickDraw calls to position the LaserWriter's pen before inserting the position dependent PostScript code. When the PostScript in the picture is executed, the LaserWriter's pen location will be in a location relative to the position of the picture on the page.

The following example illustrates a method for positioning the LaserWriter's pen before inserting any PostScript code into the picture. The method uses QuickDraw calls to position the LaserWriter's pen, and will work with any application that supports QuickDraw pictures. Applications do not have to be changed to be able to print pictures which use this technique; normal calls to `DrawPicture` will work.

The following code fragment will create a picture that contains PostScript to draw a rectangle around the picture's frame:

```
FUNCTION CreatePicture(pictureRect: Rect): PicHandle;
    CONST
        PostScriptBegin = 190;
        PostScriptEnd = 191;
        PostScriptHandle = 192;
    VAR
        PSString     : Str255;
        PSHandle     : Handle;
        theError     : OSErr;
    BEGIN
        (* Create a new Picture. *)
        CreatePicture := OpenPicture(pictureRect);
        ClipRect(pictureRect);

        (* Set the pen size to 0,0 so the following Line will not *)
         (* be shown, it is only sent to position the pen.*)
         PenSize(0,0);

        (* Move the QuickDraw pen to the first pixel inside the *)
        (* the picture's frame.  This by itself will not *)
        (* change the LaserWriter's pen location! *)
        MoveTo(pictureRect.left, pictureRect.top);

        (* Force the LaserWriter's pen location to match the  *)
        (* QuickDraw pen location.  Actually any drawing command, *)
        (* such as LineTo or even DrawString will cause the *)
         (* LaserWriter's pen location to change.  This call to *)
         (* the Line procedure only causes the coordinates of *)
         (* the above MoveTo to be flushed to the LaserWriter. *)
         (* Because of the PenSize call above, no Line is drawn. *)
         Line(0,0);

        (* Reset the pen to its default size. *)
        PenSize(1,1);

        (* The LaserWriter's pen location has now been changed.  The *)
        (* PostScript currentpoint operator will now return the *)
         (* location of the center pixel of the Picture. *)
        (* Get the PostScript ready to be sent. *)
        (* currentpoint        - push the current Point onto the stack. *)
        (* newpath            - Begin a new Line segment. *)
         (* MoveTo            - Move to the currentpoint we saved above. *)
         (* nn nn rlineto        - frame the Picture with lines. *)
         (* stroke             - Draw the frame. *)
         PSString :=
        '100 0 rlineto 0 100 rlineto -100 0 rlineto 0 -100 rlineto stroke '
                ;
         PSString[length(PSString)] := CHR(13); (* Don't forget CR. *)
         theError:=PtrToHand(Ptr(ORD4(@PSString)+1),
        PSHandle,length(PSString));
         IF theError <> noErr THEN HandleError;

        (* Send the PostScript code to the LaserWriter. *)
        PicComment(PostScriptBegin,0,nil);

            (* QuickDraw calls made between the PostScriptBegin *)
            (* and PostScriptEnd PicComments will be ignored by *)
            (* devices that support PostScript. *)

            PicComment(PostScriptHandle,GetHandleSize(PSHandle),PSHandle);

        PicComment(PostScriptEnd,0,nil);

        (* Kill off the Handle we created and close the picture. *)
         DisposHandle(PSHandle);
        ClosePicture;
```

See the LaserWriter Reference Manual for more information about `PicComments`. See the *PostScript Language Reference Manual* for more information about the `currentpoint` operator.

There are some important guidelines to follow when sending PostScript directly to the LaserWriter. See the PostScript Commands section of Technical Note #91, for a complete description of these guidelines.

Back to top

References

The Print Manager

QuickDraw

LaserWriter Reference Manual

Technical Note M.IM.gifComments -- Optimizing for the LaserWriter--PicComments

PostScript Language Reference Manual , Adobe Systems

PostScript Language Cookbook and Tutorial , Adobe Systems

Back to top

Downloadables

Acrobat version of this Note (K). Download