#27: MacDraw's 'PICT' File Format

See also: QuickDraw
Color QuickDraw
Technical Note #21—
QuickDraw's Internal Picture Definition
Technical Note #91—
Optimizing for the LaserWriter—Picture Comments

Written by: Ginger Jernigan August 20, 1986
Updated: March 1, 1988

This Technical Note originally described the PICT file format used by MacDraw and the picture comments that MacDraw used to communicate with the LaserWriter Driver. All of this information is now available from more appropriate sources.

Descriptions of the PICT file format and the contents of PICTs can be found in *Inside Macintosh Volume V* and Technical Note #21.

The picture comments supported by the LaserWriter driver are described in Technical Note #91.

MacDraw is now a product of Claris. For information about MacDraw's specific use of PICT and for information about other Claris products, contact:

Claris Technical Support
440 Clyde Avenue
Mountain View, CA 94043

(415) 962-0371
AppleLink: CLARIS.TECH

#120: Drawing Into an Offscreen PixMap

See also: QuickDraw
Technical Note #41—Drawing Into an Offscreen Bitmap
Technical Note #119—
Determining If Color QuickDraw Exists
Technical Note #129—SysEnvirons

Written by: Jim Friedlander & Rick Blair May 4, 1987
Modified by: Rick Blair July 1, 1987
Updated: March 1, 1988

This technical note provides a simple example of drawing to, then copying from, an offscreen `PixMap`. **Changes since 5/87:** Set `theGDevice` to the `MaxDevice` before the `OpenCPort` call to get more of the `PixMap` set up correctly. This saves code and, more importantly, improves its chances of being compatible in the future. Secondly, fixed a typo in the `OffRowBytes` calculation.

This example shows how to draw something in an offscreen `PixMap` and then `CopyBits` it back to the screen. It handles the case where multiple screens of different pixel depths are present.

Before we can make any Color QuickDraw calls, we must be sure that Color QuickDraw is present (see Technical Notes #119 and #129 for details). Then, given the following types, constants and variables:

```
CONST
   OffLeft          = 30;
   OffTop           = 30;
   OffBottom        = 250;
   OffRight         = 400;

   {These constants for the bounds of the offscreen PixMap are chosen
   because we know what the extent of the drawing will be and we want to
   restrict the size of the map as much as possible.}

TYPE
   BitMapPtr        = ^BitMap;   {for type coercion in the CopyBits call}

VAR
   offRowBytes      : LONGINT;
   sizeOfOff        : LONGINT;
   myBits           : Ptr;
   destRect         : Rect;
   globRect         : Rect;
   bRect            : Rect;
```

Now to fix up the PixMap location- and size-specific information:

```
WITH myCGrafPtr^.portPixMap^^ DO BEGIN
    baseAddr := myBits;
    rowbytes := offRowBytes + $8000; {remember to be a PixMap}
    bounds := bRect;
END;                            {with}
```

Color QuickDraw distinguishes between new and old style ports by checking the high bit of `rowBytes`, which is why we add `$8000` to `OffRowBytes` in the above code. Now we need to clone the `maxDevice`'s color table so we can put it into our offscreen `PixMap`.

```
ourCMHandle := theMaxDevice^^.gdPMap^^.pmTable;
err := HandToHand(Handle(ourCMHandle)); {clone it}
{real programs do error checking here}

myCGrafPtr^.portPixMap^^.pmTable := ourCMHandle; {put the cloned,
      correctly set-up Color Table into the offscreen map}
SetPort(GrafPtr(myCGrafPtr));  {Set the port to the offscreen port}
```

Now we call procedure `DrawIt` (which calls the function `FillInColors`) to draw an image in the offscreen port:

```
FUNCTION FillInColor(r,g,b: Integer): RGBColor;
{small utility routine to return an RGBColor}
    VAR
       theColor          : RGBColor;

    BEGIN                          {FillInColor}
       WITH theColor DO BEGIN
           red := r;
           green := g;
           blue := b;
       END;
       FillInColor := theColor;
    END;                           {FillInColor}

PROCEDURE DrawIt;

    VAR
       OvalRect          : Rect;
       myRed,myBlue,myWhite,
       myGreen, myBlack : RGBColor;

    BEGIN                          { DrawIt }
       {get our colors set up}
       myRed := FillInColor(-1,0,0);
       myBlue := FillInColor(0,0,-1);
       myGreen := FillInColor(0,-1,0);
       myWhite := FillInColor(-1,-1,-1);
       myBlack := FillInColor(0,0,0);
       PenMode(PatCopy);
       RGBBackColor(myBlue); {set the backcolor of the current port}
       EraseRect(thePort^.portRect); {blue it out}
       RGBBackColor(white);  {set back to white}
```

```
        RGBForeColor(myRed);{set the forecolor of the current port}
        SetRect(OvalRect,30,30,190,150);
        PaintOval(OvalRect);

        InsetRect(OvalRect,1,20);
        EraseOval(OvalRect); {erase oval to white}

        RGBForeColor(myGreen);{draw the final oval in green}
        InsetRect(OvalRect,40,1);
        PaintOval(OvalRect);
        RGBForeColor(myBlack);
END;                        { DrawIt }
```

Now we're done drawing, so set `thePort` and `theGDevice` back:

```
SetPort(MyCWindow);
SetGDevice(oldDevice);
```

Now we can draw the image onscreen by `CopyBits`ing the bits from the offscreen `PixMap`'s `portPix` to `MyCWindow`'s `portPix`:

```
destRect := bRect;
OffSetRect(destRect,OffLeft,OffTop); {adjust for coordinates}
CopyBits(BitMapPtr(MyCGrafPtr^.portPixMap^)^, MyCWindow^.portBits,
    bRect, destRect, 0, NIL);
```

and, finally, we clean up by closing the `CGrafPort` we created, freeing the space we reserved for the offscreen `PixMap`'s pixel image and disposing of the color table we allocated:

```
CloseCPort(myCGrafPtr);         {Close our port}
DisposPtr(MyBits);              {clean up}
DisposHandle(Handle(ourCMHandle)); {get rid of color table we cloned}
```

Note: For optimal performance, you will want to make sure that the source and destination `PixMaps` are aligned—this will be the subject of a future technical note.