Macintosh Technical Notes

#120: Drawing Into an Off-Screen PixMap

See also: *Inside Macintosh*: QuickDraw
Technical Note #41: Drawing Into an Off-Screen 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
Modified by: Rich Collyer — October 1, 1988

This Technical Note provides a simple example of drawing to, then copying from, an off-screen `PixMap`.
Changes since March 1, 1988: Added sample MPW C 3.0 code.

This example shows how to draw something in an off-screen `PixMap` and then use `_CopyBits` to copy 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 Note #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 off-screen 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;
        theDepth       : INTEGER;
        i              : INTEGER;
        err            : INTEGER;
        myCGrafPort    : CGrafPort;
        myCGrafPtr     : CGrafPtr;
        ourCMHandle    : CTabHandle;
        theMaxDevice   : GDHandle;
        oldDevice      : GDHandle;
```

We first create a color window.

```
myCWindow := GetNewCWindow(SomeID,NIL,WindowPtr(-1));
```

We need to find the maximum depth device to which we will copy the off-screen image with `_CopyBits`.

```
SetPort(myCWindow); {set to this port for the localToGlobals that follow}

SetRect(bRect,OffLeft,OffTop,OffRight,OffBottom);
IF NOT SectRect(myCWindow^.portRect,bRect,globRect) THEN
       NothingToCopy;  {nothing to do, clean up and EXIT}

{still here, so let's convert to globals}
LocalToGlobal(globRect.topLeft);
LocalToGlobal(globRect.botRight);

{figure out how much space we need for our pixel image.
we will call GetMaxDevice and get the pixel map from that --
we do this to cover the case where the pixel image that we wish
to CopyBits to spans multiple devices (of possibly different depths)}

theMaxDevice:= GetMaxDevice(globRect);{get the maxDevice}
```

Now we need to set `theGDevice` to the device with the maximum pixel depth, so that the pixel map of our new `CGrafPort` will be copied from one of the proper depth, etc.

```
oldDevice := GetGDevice;{save theGDevice so we can restore it later}
SetGDevice(theMaxDevice);    {Set to the maxdevice}
```

We then open a new `CGrafPort` which will be used for off-screen drawing.

```
myCGrafPtr := @myCGrafPort;  {initialize this guy}
OpenCPort(myCGrafPtr);       {open a new color port — this calls InitCPort}
theDepth:= myCGrafPtr^.portPixMap^^.pixelSize;
```

Now we are ready to calculate the size of the pixel image that we will need.

```
{similar formula to technote 41, except we must include pixel depth}
offRowBytes := ((((theDepth * (OffRight - OffLeft)) + 15)) DIV 16) * 2;
{make sure LONGINT math is done on the next line!}
sizeOfOff := LONGINT(OffBottom - OffTop) * offRowBytes;
OffSetRect(bRect, - OffLeft, - OffTop); {adjust for local coordinates}

{Set up baseAddr, rowBytes,bounds and pixelSize
of the PixMap in our fresh, new CPort}
myBits := NewPtr(sizeOfOff); {allocate space for the pixel image}
{real programs do error checking here}
```

Now we fix 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 a `BitMap` and `PixMap` 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 color table of the `maxDevice` so we can put it into our off-screen `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
                                                    off-screen map}
SetPort(GrafPtr(myCGrafPtr));{Set the port to the off-screen port}
```

Now we call procedure `DrawIt` (which calls the function `FillInColors`) to draw an image in the off-screen 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(myWhite);  {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 are done drawing, so set `thePort` and `theGDevice` back.

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

Now we can draw the image on the screen by using `_CopyBits` to copy the bits from the `portPix` of the off-screen `PixMap` to the `portPix` of `MyCWindow`.

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

Finally, we clean up by closing the `CGrafPort` we created, freeing the space we reserved for the pixel image of the off-screen `PixMap` 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}
```

MPW 3.0 C code:

Note: Most of the comments above apply to this C code. If you do not understand what the code is doing, try looking at the equivalent Pascal code above and the comments which are associated with that code.

```
/* Define constants for the Off-Screen Rect */
#define OffLeft       30
#define OffTop        30
#define OffBottom     250
#define OffRight      400

/* typedef BitMapPtr for use during CopyBits operation */
typedef BitMap *BitMapPtr;

long          offRowBytes, sizeOfOff;
Ptr           myBits;
Rect          destRect, globRect, bRect;
int           theDepth, i, err;
CGrafPort     myCGrafPort;
CGrafPtr      myCGrafPtr;
CTabHandle    ourCMHandle;
GDHandle      theMaxDevice, oldDevice;
Point         tempP;
```

Open a color window on screen. In MPW C, `myWindow` will be declared as a `WindowPtr` not a `CWindowPtr`, which is contrary to the way *Inside Macintosh*, Volume V documents it.

```
myWindow = GetNewCWindow(SomeID,nil,(WindowPtr) -1);

/* set to this port for the localToGlobals that follow */
SetPort((WindowPtr) myWindow);

SetRect(&bRect,OffLeft,OffTop,OffRight,OffBottom);
if (!SectRect(&(*myWindow).portRect,&bRect,&globRect))
       ExitToShell();  /*nothing to do, clean up and EXIT*/
```

MPW does not have `topLeft` or `botRight` elements for `Rect` structures, so you need to set the `tempPoint`, call `_LocalToGobal` then reset `globRect`.

```
tempP.v = globRect.top;
tempP.h = globRect.left;
LocalToGlobal(&tempP);
globRect.top = tempP.v;
globRect.left = tempP.h;

tempP.v = globRect.bottom;
tempP.h = globRect.right;
LocalToGlobal(&tempP);
globRect.bottom = tempP.v;
globRect.right = tempP.h;

theMaxDevice = GetMaxDevice(&globRect);/*get the maxDevice*/

oldDevice = GetGDevice();/*save theGDevice so we can restore it later*/
SetGDevice(theMaxDevice);    /*Set to the maxdevice*/
```

Now it is time to set up the off-screen `PixMap`.

```
myCGrafPtr = &myCGrafPort;   /*initialize this guy*/
OpenCPort(myCGrafPtr);       /*open a new color port — this calls InitCPort*/
theDepth = (**(*myCGrafPtr).portPixMap).pixelSize;

/* Bitshift and adjust for local coordinates */
offRowBytes = (((theDepth * (OffRight - OffLeft)) + 15) >> 4) << 1;
sizeOfOff = (long) (OffBottom - OffTop) * offRowBytes;
OffsetRect(&bRect, - OffLeft, - OffTop);

myBits = NewPtr(sizeOfOff);

/* Remember to be a PixMap */
(**(*myCGrafPtr).portPixMap).baseAddr = myBits;
(**(*myCGrafPtr).portPixMap).rowBytes = offRowBytes + 0x8000;
(**(*myCGrafPtr).portPixMap).bounds = bRect;

ourCMHandle = (**(**theMaxDevice).gdPMap).pmTable;
err = HandToHand(&((Handle) ourCMHandle));
/* Real programs do error checking here */

(**(*myCGrafPtr).portPixMap).pmTable = ourCMHandle;
SetPort((GrafPtr) myCGrafPtr);

/*************************************************/
/*                                               */
/*     function for setting the wanted color     */
/*                                               */
/*************************************************/
RGBColor FillInColor(r,g,b)
int     r,g,b;

{                       /*FillInColor*/

        RGBColor        theColor;

        theColor.red = r;
        theColor.green = g;
        theColor.blue = b;
        return (theColor);
}

/*****************************************************/
/*                                                   */
/*  Drawing routine which makes the background blue  */
/* then draws a red oval, white oval, and green oval */
/* After drawing to the off-screen it CopyBits to the */
/*                       screen                      */
/*                                                   */
/*****************************************************/
void DrawIt()

{
        Rect            OvalRect;
        RGBColor        myRed,myBlue,myWhite,myGreen,myBlack;

        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);
```

```
        EraseRect(&(*qd.thePort).portRect);
        RGBBackColor(&myWhite);
        RGBForeColor(&myRed);
        SetRect(&OvalRect,30,30,190,150);
        PaintOval(&OvalRect);

        InsetRect(&OvalRect,1,20);
        EraseOval(&OvalRect);

        RGBForeColor(&myGreen);
        InsetRect(&OvalRect,40,1);
        PaintOval(&OvalRect);
        RGBForeColor(&myBlack);

        SetPort((WindowPtr) myWindow);
        SetGDevice(oldDevice);

        destRect = bRect;
        OffsetRect(&destRect,OffLeft,OffTop);
        CopyBits((BitMapPtr) *(*myCGrafPtr).portPixMap,
                    &(*myWindow).portBits,&bRect, &destRect, 0, nil);

        return;
}
```

These are used to help clean up when you are done with the off-screen `PixMap`.

```
CloseCPort(myCGrafPtr);
DisposPtr(myBits);
DisposHandle((Handle) ourCMHandle);
```

Note: For optimal performance, you will want to make sure that the source `PixMap` and destination `PixMap` are aligned.