

#41: Drawing Into an Offscreen Bitmap

See also: QuickDraw

Written by:	Jim Friedlander Ginger Jernigan	July 1, 1985
Modified by:	Jim Friedlander	November 26, 1985
Updated:		March 1, 1988

This note is an example of drawing to and copying from an offscreen bitmap.

The following is a short example of drawing to, then copying from, an offscreen bitmap. The only major concern when setting this up is that the number of rowBytes in the bitmap is even. Given these constants and variables

```
CONST
    myString = 'The EYE';

VAR
    oldPort    : GrafPtr;      {saved port}
    offScreen  : GrafPort;     {our offscreen bitmap}
    ovalRect   : Rect;         {used for example drawing}
```

the offscreen port and bitmap can be allocated with the following statements:

```
GetPort(oldPort);

OpenPort(@offScreen);
SetPort(oldPort);

WITH offScreen, portBits, bounds DO BEGIN

    {make bitmap exactly the size of the port}
    portRect := thePort^.portRect;
    bounds := portRect;

    {rowBytes is size of row, rounded up to an even number of bytes}
    rowBytes := (((right - left) + 15) DIV 16) * 2;

    {number of bytes in BitMap is rowBytes * number of rows}
    {this calculation must be done with longs}
    baseAddr:= NewPtr(rowBytes * LongInt(bottom - top));

    {check MemError here to see if we had enough room for the bitmap}

END;
```

We are now ready to draw into our offscreen port:

```
SetPort(@offScreen);

{since it's just an area of memory, we'd better clear it before we start}
EraseRect(thePort^.portRect);

{Example drawing}
ovalRect := thePort^.portRect;
FillOval(ovalRect, black);
InsetRect(ovalRect, 1, 20);
FillOval(ovalRect, white);
InsetRect(ovalRect, 40, 1);
FillOval(ovalRect, black);
WITH ovalRect DO
    MoveTo((left+right-StringWidth(myString)) DIV 2, (top+bottom-12) DIV 2);
    TextMode(srcXOr);
    DrawString(myString);
```

Now we're ready to do the CopyBits, copying the image that we've drawn in the offscreen bitmap to thePort^.portBits, from the rectangle offScreen.bounds to the rectangle thePort^.portRect:

```
SetPort(oldPort);
CopyBits(offScreen.portBits, thePort^.portBits,
         offScreen.portRect, thePort^.portRect, srcCopy, NIL);
```

Notice here that if thePort^.portRect and offScreen.bounds are not the same size, CopyBits will scale the bit image that is being copied.

From MPW C, given these variables

```
char* myString = "\pThe EYE"; /* string to display */

GrafPtr  oldPort;                /* saved port */
GrafPort offScreen;             /* our offscreen bitmap */
Rect      ovalRect;             /* used for example drawing */
```

the offscreen port and bitmap can be allocated with the following statements:

```
GetPort(&oldPort);

OpenPort(&offScreen);
SetPort(oldPort);

/* make bitmap exactly the size of the port */
offScreen.portRect = qd.thePort->portRect;
offScreen.portBits.bounds = offScreen.portRect;

/* rowBytes is size of row, rounded up to an even number of bytes */
offScreen.portBits.rowBytes = (((offScreen.portBits.bounds.right -
    offScreen.portBits.bounds.left) + 15) >> 4) << 1;
```

```

/* number of bytes in BitMap is rowBytes * number of rows */
/* this calculation must be done with longs */
offScreen.portBits.baseAddr = NewPtr(offScreen.portBits.rowBytes *
    (long) (offScreen.portBits.bounds.bottom -
    offScreen.portBits.bounds.top));

/* check MemError here to see if we had enough room for the bitmap */

```

We are now ready to draw into our offscreen port:

```

SetPort(&offScreen);

/* since it's just memory, we'd better clear it before we start */
EraseRect(&qd.thePort->portRect);

/* Example drawing */
ovalRect = qd.thePort->portRect;
FillOval(&ovalRect, qd.black);
InsetRect(&ovalRect, 1, 20);
FillOval(&ovalRect, qd.white);
InsetRect(&ovalRect, 40, 1);
FillOval(&ovalRect, qd.black);
MoveTo((ovalRect.left+ovalRect.right-StringWidth(myString)) >> 1,
    (ovalRect.top+ovalRect.bottom-12) >> 1);
TextMode(srcXor);
DrawString(myString);

SetPort(oldPort);
CopyBits(&offScreen.portBits, &qd.thePort->portBits,
    &offScreen.portRect, &qd.thePort->portRect, srcCopy, NULL);

```

Please note that this way of doing things is NOT faster than simply defining a picture and then drawing it to your window. There are cases, however, such as text rotation, where it will be advantageous to do the drawing off the screen, manipulate the bit image and then copy the result to the visible window. In addition, this technique will reduce flicker, because all of the drawing done offscreen will appear on the screen at once.