T E C H N O T E:
Creating Off-Screen Bitmaps When Speed is Critical

By Jim Friedlander & Ginger Jernigan
Revised by Mike Marinkovich
marink@applelink.apple.com
Apple Developer Technical Support (DTS)

This Technote provides an example of creating an off-screen bitmap by hand, drawing to it, and then copying from it to the screen. Apple encourages the use of `GWorld`s for your off-screen needs. In some cases, however, creating your own off-screens can be beneficial. The resulting off-screen bitmaps can be used like a 1-bit `GWorld`, but with improved performance.

This technique is useful for the creation of regions when `OpenRgn` is not an option. For example, in making a region from a line, draw the line in an offscreen and call `BitMapToRegion` to convert the offscreen's bitmap to a region. These offscreen bitmaps can be also be substituted for pixmaps in routines such as `CopyMask`, where the mask is black and white, and speed is of great importance.

This Technote is written primarily for those involved in speed-critical projects, such as game developers and graphics applications developers.

Note
This technique for creating off-screen bitmaps is intended for black and white uses only. If your needs include only color ports, you should review the Technote "Techniques for Creating Off-Screen Graphics Environments." ◆

Drawing Off-Screen Bitmaps

The following is an example of creating and drawing to an off-screen bitmap, then copying from it to an on-screen window. The example is supplied in both a C and Pascal versions, and will work with all compilers.

Creating an Off-Screen Bitmap in C

Let's look at a general purpose function to create an off-screen bitmap. This function creates the `GrafPort` on the heap. You can also create it on the stack and pass the uninitialized structure to a function similar to this one.

```
Boolean CreateOffscreenBitMap(GrafPtr *newOffscreen, Rect *inBounds)
{
    GrafPtr savePort;
    GrafPtr newPort;

    GetPort(&savePort);    /* need this to restore thePort after
                                                        OpenPort */

    newPort = (GrafPtr) NewPtr(sizeof(GrafPort));    /* allocate the
                                                        grafPort */
    if (MemError() != noErr)
        return false;              /* failed to allocate the off-screen
                                                          port */
    /*
    the call to OpenPort does the following . . .
        allocates space for visRgn (set to screenBits.bounds) and
        clipRgn (set wide open)
        sets portBits to screenBits
        sets portRect to screenBits.bounds
        (See Inside Mac: Imaging with QuickDraw,

        pages 2-38 to 2-39)
        side effect: does a SetPort(&offScreen)
    */
    OpenPort(newPort);
```

```
    /* make bitmap the size of the bounds that caller supplied */
    newPort->portRect = *inBounds;
    newPort->portBits.bounds = *inBounds;
    RectRgn(newPort->clipRgn, inBounds);    /* avoid wide-open clipRgn,
                                                    to be safe  */
    RectRgn(newPort->visRgn, inBounds);     /* in case newBounds is >
                                                    screen bounds */

    /* rowBytes is size of row, it must be rounded up to an even number
                                                        of bytes */
    newPort->portBits.rowBytes = ((inBounds->right - inBounds->left +
                                                    15) >> 4) << 1;

    /* number of bytes in BitMap is rowBytes * number of rows */
    /* see notes at end of Technote about using _NewHandle rather
                                                    than _NewPtr*/
    newPort->portBits.baseAddr =
            NewPtr(newPort->portBits.rowBytes * (long) (inBounds->bottom
                                                - inBounds- >top));
    if (newPort->portBits.baseAddr == nil) {  /* check to see if we had
enough room for
                                                        the bits */
        SetPort(savePort);
        ClosePort(newPort);      /* dump the visRgn and clipRgn */
        DisposPtr((Ptr)newPort); /* dump the GrafPort */
        return false;            /* tell caller we failed */
        }
    /* since the bits are just memory, let's clear them before we start*/
    EraseRect(inBounds);     /* OpenPort did a SetPort(newPort) so we
                                                        are ok */
    *newOffscreen = newPort;
    SetPort(savePort);
    return true;              /* tell caller we succeeded! */
}
```

Eliminating an Off-Screen Bitmap in C

To eliminate an off-screen bitmap created by the previous function, use this function:

```
void DestroyOffscreenBitMap(GrafPtr oldOffscreen)
{
    ClosePort(oldOffscreen);                    /* dump the visRgn and
                                                   clipRgn */
    DisposPtr(oldOffscreen->portBits.baseAddr); /* dump the bits */
    DisposPtr((Ptr)oldOffscreen);               /* dump the port */
}
```

Using an Off-Screen Bitmap in C

Now that you know how to create and destroy an off-screen bitmap, let's go through the motions of using one. First, let's define a few things to make the `NewWindow` call a little clearer.

```
#define kIsVisible true
#define kNoGoAway false
#define kNoWindowStorage 0L
#define kFrontWindow ((WindowPtr) -1L)
```

Here's the body of the test code:

```
main()
{
    char* myString = "\pThe EYE";  /* string to display */

    GrafPtr   offscreen;          /* our off-screen bitmap */
    Rect      ovalRect;           /* used for example drawing */
    Rect      myWBounds;          /* for creating window */
    Rect      OSRect;             /* portRect and bounds for off-screen
                                                          bitmap*/
    WindowPtr myWindow;

    InitToolbox();                /* exercise for the reader */
    myWBounds = qd.screenBits.bounds;  /* size of main screen */
    InsetRect(&myWBounds, 50,50);  /* make it fit better */
    myWindow = NewWindow(kNoWindowStorage, &myWBounds, "\pTest Window",
                         kIsVisible,noGrowDocProc, kFrontWindow,
                         kNoGoAway, 0);
    if (!CreateOffscreenBitMap(&offscreen, &myWindow->portRect)) {
        SysBeep(1);
        ExitToShell();
```

```
        }
    /* Example drawing to our off-screen bitmap*/
    SetPort(offscreen);
    OSRect = offscreen->portRect;  /* offscreen bitmap's local
                                              coordinate rect */
    ovalRect = OSRect;
    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);

    /* copy from the off-screen bitmap to the on-screen window.  Note
    that in this case the source and destination rects are the same size
    and both cover the entire area.  These rects are allowed to be
    portions of the source and/or destination and do not have to be the
    same size.  If they are not the same size then _CopyBits scales the
    image accordingly.
    */
    SetPort(myWindow);
    CopyBits(&offscreen->portBits, &(*myWindow).portBits,
            &offscreen->portRect, &(*myWindow).portRect, srcCopy, 0L);

    DestroyOffscreenBitMap(offscreen);    /* dump the off-screen bitmap*/
    while (!Button());     /* give user a chance to see our work of art*/
}
```

Creating an Off-Screen Bitmap in Pascal

Let's look at a general purpose function to create an off-screen bitmap. This function creates the `GrafPort` on the heap. You can also create it on the stack and pass the uninitialized structure to a function similar to this one.

```
FUNCTION CreateOffscreenBitMap(VAR newOffscreen:GrafPtr; inBounds:Rect)
                : BOOLEAN;

    savePort  : GrafPtr;
```

```
newPort   : GrafPtr;

GetPort(savePort);  {need this to restore thePort after OpenPort
                                                    changes it}

newPort := GrafPtr(NewPtr(sizeof(GrafPort)));   {allocate the
                    GrafPort}
IF MemError <> noErr THEN BEGIN
    CreateOffscreenBitMap := false;     {failed to allocate it}
    EXIT(CreateOffscreenBitMap);
END;
{
the OpenPort call does the following . . .
allocates space for visRgn (set to screenBits.bounds) and clipRgn
(set wide open)
sets portBits to screenBits
sets portRect to screenBits.bounds
(See Inside Mac: Imaging with QuickDraw, pages 2-38 to 2-39)
side effect: does a SetPort(offScreen)
}
OpenPort(newPort);
{make bitmap exactly the size of the bounds that caller supplied}
WITH newPort^ DO BEGIN {portRect, clipRgn, and visRgn are in newPort}
    portRect := inBounds;
    RectRgn(clipRgn, inBounds); {avoid wide-open clipRgn, to be safe}
    RectRgn(visRgn, inBounds);  {in case inBounds is > screen bounds}
END;

WITH newPort^.portBits DO BEGIN     {baseAddr, rowBytes and bounds
                                            are in newPort}
    bounds := inBounds;
    {rowBytes is size of row  It must be rounded up to even number
                                                    of bytes}
    rowBytes := ((inBounds.right - inBounds.left + 15) DIV 16) * 2;

    {number of bytes in BitMap is rowBytes * number of rows}
    {see note at end of Technical Note about using _NewHandle rather
                                                    than _NewPtr}
    baseAddr := NewPtr(rowBytes * LONGINT(inBounds.bottom -
                                                inBounds.top));
END;
```

```
    IF baseAddr == nilTHEN BEGIN        {see if we had enough room for
                                                              the bits}
        SetPort(savePort);
        ClosePort(newPort);              {dump the visRgn and clipRgn }
        DisposPtr(Ptr(newPort));         {dump the GrafPort}
        CreateOffscreenBitMap := false;
    END
    ELSE BEGIN
        {since the bits are just memory, let's erase them before we
                                                                  start }
        EraseRect(inBounds);            {OpenPort did a SetPort(newPort)}
        newOffscreen := newPort;
        SetPort(savePort);
        CreateOffscreenBitMap := true;
    END;;
```

Eliminating an Off-Screen Bitmap in pascal Pascal

Here is the procedure to get rid of an off-screen bitmap created by the previous function:

```
PROCEDURE DestroyOffscreenBitMap(oldOffscreen : GrafPtr);
    ClosePort(oldOffscreen);                        {dump the visRgn
                                                        and clipRgn }
    DisposPtr(oldOffscreen^.portBits.baseAddr);     {dump the bits }
    DisposPtr(Ptr(oldOffscreen));                   {dump the port };
```

Using an Off-Screen Bitmap: MPW Pascal

Now that you know how to create and destroy an off-screen bitmap, let's test one out. First, let's define a few things to make the `NewWindow` call a little clearer.

```
CONST
    kIsVisible    = true;
    kNoGoAway     = false;
    kMakeFrontWindow = -1;
    myString      = 'The EYE';  {string to display}
```

Here's the body of the test code:

```
VAR
    offscreen : GrafPtr;    {our off-screen bitmap}
    ovalRect  : Rect;       {used for example drawing}
    myWBounds : Rect;       {for creating window}
    OSRect    : Rect;       {portRect and bounds for off-screen bitmap}
    myWindow  : WindowPtr;

    InitToolbox;                        {exercise left to the reader}

    myWBounds := screenBits.bounds;    {size of main screen }
    InsetRect(myWBounds, 50,50);       {make it fit better }
    myWindow := NewWindow(NIL, myWBounds, 'Test Window', kIsVisible,
            noGrowDocProc, WindowPtr(kMakeFrontWindow), kNoGoAway, 0);

    IF NOT CreateOffscreenBitMap(offscreen,myWindow^.portRect) THEN BEGIN
        SysBeep(1);
        ExitToShell;
    END;

    {Example drawing to our off-screen bitmap }
    SetPort(offscreen);
    OSRect := offscreen^.portRect;    {offscreen bitmap's local
                                            coordinate rect }
    ovalRect := OSRect;
    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);

    {copy from the off-screen bitmap to the on-screen window.  Note
    that in this case the source and destination rects are the same size
    and both cover the entire area.  These rects are allowed to be
    portions of the source and/or destination and do not have to be the
    same size.  If they are not the same size then _CopyBits scales the
    image accordingly
```

```
    }
    SetPort(myWindow);
    CopyBits(offscreen^.portBits, myWindow^.portBits,
            offscreen^.portRect, myWindow^.portRect, srcCopy, NIL);

    DestroyOffscreenBitMap(offscreen);  {remove the evidence}

    WHILE NOT Button DO;                {give user a chance to see the
                                                              results}.
```

Summary

In the example code, the bits of the `BitMap` structure pointed to by the `baseAddr` field are allocated by a `NewPtr` call.

Keeping a large off-screen around for any length of time may lead to heap fragmentation. One alternative that lessens this problem is to get the bits via `NewHandle`, so that the Memory Manager can move them when necessary. To implement this approach, you need to keep the handle separate from the `GrafPort` (for example, in a structure that combines a `GrafPort` and a `Handle`). When using the off-screen bitmap, lock the handle and put the dereferenced handle into the `baseAddr` field. You can then unlock the handle when not using the off-screen bitmap.

Further Reference

- *Inside Macintosh: Imaging with QuickDraw*