Offscreen Graphics Optimizing your application for visual speed and smoothness

Suppose your application draws individual graphics objects and it needs to redraw part of a window that has been covered by another window or menu. Your application may be able to put an offscreen bitmap onto the screen faster than it could re-create the drawing steps, and at the same time avoid the choppy effect that can arise from drawing a large number of separate objects.

In a multi-tasking situation you may want to creat an image offscreen to avoid having some other application or desk accessory change the graphics environment in the midst of your processing. An offscreen graphics environment that you create cannot be modified by any other application.

Most important, today's Macintosh computer systems may have several screens, of different sizes, pixel depths, and color capabilities, and the user may decide to put your application's window on (or even across) any of them. By preparing the image offscreen in a graphics world that you create, you can control the image's characteristics. When the image is ready for display, you can check the graphics environment and react accordingly, such as by calling **CopyBits** if everything is as you expect, or by adjusting the image to match the pixel depth of the deepest screen your window touches. If the environment is not suitable - for example, if the user has moved the window for your multicolor display of the Bayeux tapestry to a 9-inch black-and-white monitor - you can display the best image possible and issue a gentle remonstrance.

An offscreen graphics world is an extention of the CGrafPort record. It contains a CGrafPort describing the offscreen port, a reference to the offscreen device associated with the offscreen graphics world, and other state information. The actual structure is kept private by Apple to allow for future extentions. The pointer type of the offscreen graphics world is called a GWorldPtr, which is the same thing as a CGrafPtr. On black-and-white machines lacking **Color QuickDraw**, the graphics world pointer points to an extension of the original GrafPort record.

Your use of the **Graphics Devices Manager** routines depends on the degree to which you need to control the characteristics of your images. If your only concern is to avoid flicker by presenting a complete image, you can choose to create an offscreen CGrafPort record but not a new graphics devices record. Your offscreen CGrafPort is then linked to the current graphics device, from which it takes characteristics such as pixel depth. If you need to control the pixel depth or create a new color table, you must create a new graphics device record.

You create an offscreen graphics world with the **NewGWorld** function. You can specify that a new GDevice record is to be created, or you can specify that an existing GDevice record is to be used, such as the graphics device record of the main screen (the screen with the menu bar) or of the deepest screen touched. In either case, the **Graphics Devices Manager** uses that graphics device record's characteristics to create the pixel map for your image. If you use a screen's GDevice record, then before doing any drawing you should check to be sure that the device's depth or color/black-and-white settings haven't been changed by the user. If they have, call the **UpdateGWorld** function.

To address the PixMap record created for an offscreen graphics world, use the **GetGWorldPixMap** function. Don't dereference the graphics world pointer to get the pixel map.

Before actually drawing to or from the offscreen graphics world, call **LockPixels** to lock the offscreen buffer in memory. As soon as the drawing is completed, always call **UnlockPixels**.

When the user resizes or moves a window, changes the pixel depth of screens a window intersects, or modifies a color table, you should call the **UpdateGWorld** function; your application may be able to reflect those changes in the offscreen graphics world without having to re-create it and redraw its content. Calling **UpdateGWorld** when the window moves can ensure a maximum refresh speed when using **CopyBits** to move the offscreen image on screen.

When you no longer need your offscreen graphics world, dispose of it by calling **DisposeGWorld**.

Example

```
//  This is a simple demonstration of how to use PlotCIcon to copy to an
//  offscreen pixmap, and then copy it back and display the 'cicn' in a window.

#include <QDOffscreen.h>
// Assumes inclusion of <MacHeaders>

#define MY_ICON   128   // ID of  cicn resource to plot

main()
{
    Rect rBounds; // Bounding rectangle of sample window
    Rect  r; // Rect used to display icon in
    CWindowPtr wind;      / WIndow to display ICON in
    CIconHandle theIcon;   // Handle to ICON to display

    // Offscreen world variables
    GWorldPtr currPort;    // Saves the current port prior to setting up
                           // offscreen world
    GDHandle currDev;      // Saves the current device prior to setting up
                           // offscreen world
    short err;             //  error returned from

    static Rect dOffBounds = {0,0,360,360};   // Bounds of OffScreen Graphics
                                              // World
    static GWorldPtr gMyOffG;    // Pointer to OffScreen Graphics World


    // Initialize the ToolBox
    InitGraf(&thePort);
    InitWindows();
    InitMenus();
    InitFonts();
    InitDialogs(nil);


    // Create Color Window
    SetRect(&rBounds,40,40,400,400);
    wind = (CWindowPtr)NewCWindow (nil, &rBounds, "\pThe ICON",
            TRUE, 0, (WindowPtr)-1, TRUE, 0);

    // Grab Icon from Resource File
    theIcon = GetCIcon (MY_ICON);

    if (theIcon) {

        // Build Offscreen Graphics world
        GetGWorld(&currPort,&currDev);

        // Create Offscreen Graphics world.
        err = NewGWorld(&gMyOffG, 0, &dOffBounds, nil, nil, 0);



        if (!err) {
            // Lock down Pixels that we are drawing to so that memory will not
            // move
            LockPixels (gMyOffG->portPixMap);
            // Setup drawing area to be our offscreen graphics world
            SetGWorld (gMyOffG, nil);
            // Plot Icon
            SetRect (&r, 40,40, 100,100);
            // Clear Rectangle, so that CopyBits will not copy extra background
            EraseRect (&r);
            PlotCIcon (&r, theIcon);
            // Done drawing, now reset Port etc.
            SetGWorld (currPort, currDev);

            // Now copy offscreen drawing to Window

            // Set ForeColor to black, and BackColor to white
            // NOTE:  This is necessary to prevent copybits from displaying
            // unwanted colors  in the copied image.
            ForeColor (blackColor);
            BackColor (whiteColor);
            CopyBits ( (BitMap *) (*(gMyOffG->portPixMap)),
                        &((GrafPtr)wind)->portBits, &r, &r, srcCopy, nil);

            // Now unlock Pixels.
            UnlockPixels (gMyOffG->portPixMap);

        }
    }

    MoveTo(150,200);
    DrawString("\pPress Mouse To Exit..");
    while (!Button()) ;
}

// The following is the Rez description file for the icon
// used in the code example above

data 'cicn' (128) {
    $"0000 0000 8008 0000 0000 0020 0020 0000"
```

```
        $"0000 0000 0000 0048 0000 0048 0000 0000"
        $"0002 0001 0002 0000 0000 0000 0000 0000"
        $"0000 0000 0000 0004 0000 0000 0020 0020"
        $"0000 0000 0004 0000 0000 0020 0020 0000"
        $"0000 FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF 8000 0001 8000 0001 8000"
        $"0001 8000 0001 87FF FFE1 86FF FF61 87EF"
        $"FFE1 87BF 7FE1 87FF FDE1 85EF FFE1 87FF"
        $"FFE1 87FA FFE1 87EF DFE1 87FB FFE1 86FF"
        $"FDE1 87FF FFE1 87FF FFE1 87FF FFE1 87BD"
        $"FFE1 87FF 7FE1 87FF FEE1 87FF FF61 87FF"
        $"FBE1 87BF FFE1 87FF 7FE1 87FF FFE1 8000"
        $"0001 8000 0001 8000 0001 8000 0001 FFFF"
        $"FFFF 0000 0000 0000 0003 0000 0000 0000"
        $"DDDD 0001 EEEE EEEE EEEE 0002 FFFF FFFF"
        $"0000 0003 0000 0000 0000 FFFF FFFF FFFF"
        $"FFFF D555 5555 5555 5557 D555 5555 5555"
        $"5557 D555 5555 5555 5557 D555 5555 5555"
        $"5557 D540 0000 0000 0157 D542 0000 0000"
        $"8157 D540 0200 0000 0157 D540 2000 8000"
        $"0157 D540 0000 0008 0157 D548 0200 0000"
        $"0157 D540 0000 0000 0157 D540 0022 0000"
        $"0157 D540 0200 0800 0157 D540 0020 0000"
        $"0157 D542 0000 0008 0157 D540 0000 0000"
        $"0157 D540 0000 0000 0157 D540 0000 0000"
        $"0157 D540 2008 0000 0157 D540 0000 8000"
        $"0157 D540 0000 0002 0157 D540 0000 0000"
        $"8157 D540 0000 0020 0157 D540 2000 0000"
        $"0157 D540 0000 8000 0157 D540 0000 0000"
        $"0157 D555 5555 5555 5557 D555 5555 5555"
        $"5557 D555 5555 5555 5557 D555 5555 5555"
        $"5557 FFFF FFFF FFFF FFFF"
};
```