Gray Scale Ramp Palette Example

```
/*
 * Gray scale ramp palette example
 * Display a raw data file using a gray scale ramp palette,
 * restoring the system palette on exit.  This program writes *directly* to the
 * PixMap, so it is much much faster than using SetCPixel().
 *
 * This program displays the contents of a raw data file.  The format it expects
 * is a series of bytes, each of which represents a pixel.  An individual byte's
 * value corresponds to a different shade of gray, ranging from black (for zero)
 * to white (for 255).  For instance, if you have a 256 x 256 file, using any
 * number of colors, it should be 65536 bytes long on disk.
 *
 * Since this is example code, it contains a bare minimum of the standard error
 * checks and user interface expected of a real program.  It is only intended as
 * a starting point.
 */

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

#define ScreenDepth(gdh)((**((**gdh).gdPMap)).pixelSize)

/*
 * Change these macros to customize this program.
 * COLORS       - the number of shades you want to use to display your data.  This
 *                is usually the same as the number of shades the data was created
 *                with.
 * WIDTH        - the width of the data in pixels (discrete data elements)
 * HEIGHT       - the height of the data in pixels (discrete data elements)
 * MAGNIFY - magnification factor to apply to the display.  The actual
 *                magnification is done by CopyBits().  This value can be a fraction.
 * USE_GRAY     - if you don't want to use a gray scale palette, set this to zero
 *                and the program will use the default system palette.
 */
#define COLORS     16
#define WIDTH      256
#define HEIGHT     256
#define MAGNIFY    1.0
#define USE_GRAY 1

unsigned char *Graph;      /* contents of file */

#define EXACT      0               /* used with pmTolerant, only exact matches */

void Init(void);
void ReadFile(void);
void SetGrayPalette(short depth, CWindowPtr w);
void DrawWindow(Rect bounds);
short log2(unsigned short);
void RestoreClut(CTabHandle ctab, CWindowPtr w);

main()
{
    GDHandle       myGDevice;
```

```
    Rect windRect, offBounds = { 0, 0, WIDTH, HEIGHT};
    CWindowPtr    mainWindow;
    CTabHandle    saveCTab;

    InitGraf();
    ReadFile();
    myGDevice = GetGDevice();
    saveCTab = (**((**myGDevice).gdPMap)).pmTable;
    HandToHand(&saveCTab);

    SetRect(&windRect, 15, 15 + GetMBarHeight(),
            (short)(MAGNIFY * WIDTH), (short)(MAGNIFY * HEIGHT));
    mainWindow = (CWindowPtr)NewCWindow(nil, &windRect, "\p", TRUE,
                    dBoxProc, (CWindowPtr)-1, TRUE, 0);
    SetPort(mainWindow);
#if USE_GRAY
    SetGrayPalette(ScreenDepth(myGDevice), mainWindow);
#endif
    DrawWindow(offBounds);
    while (!Button())
        SystemTask();
#if USE_GRAY
    RestoreClut(saveCTab, mainWindow);
#endif
    DisposeWindow(mainWindow);
}

#if USE_GRAY
void RestoreClut(CTabHandle ctab, CWindowPtr w)
{
    PaletteHandle pal = NewPalette((**ctab).ctSize, ctab, pmTolerant, EXACT);

    SetPalette((WindowPtr) w, pal, TRUE);
    ActivatePalette((WindowPtr) w);
}

void SetGrayPalette(short depth, CWindowPtr w)
{
    short colors = 1 << depth;
    CTabHandle ctab = GetCTable(depth);
    PaletteHandle pal = NewPalette((**ctab).ctSize, nil, pmTolerant, EXACT);
    ColorSpec *specs;
    short i;

    specs = (**ctab).ctTable;
    for (i = 0; i < colors; ++i) {
        specs[i].rgb.red = specs[i].rgb.green = specs[i].rgb.blue
            = i * 65535 / (COLORS - 1);
        specs[i].value = i;
    }
    /* this alerts the color mgr that the table changed */
    (**ctab).ctSeed = GetCTSeed();
    CTab2Palette(ctab, pal, pmTolerant, EXACT);
    SetPalette((WindowPtr) w, pal, TRUE);
    ActivatePalette((WindowPtr) w);
}
```

```
#endif              /* USE_GRAY */

void ReadFile()
{
    OSErr err;
    SFReply reply;
    short refNum;
    long size = (long)sizeof(unsigned char) * WIDTH * HEIGHT;
    Point where;

    SetPt (&where, 64, 48);
    SFGetFile(where, "\p", nil, -1, nil, nil, &reply);
    if (reply.good) {
        Graph = (unsigned char *) NewPtr(size);
        if (Graph) {
            if ((err = FSOpen(reply.fName, reply.vRefNum, &refNum)) ==
                    noErr) {
                err = FSRead(refNum, &size, Graph);
                err = FSClose(refNum);
                return;
            }
        }
    }
    ExitToShell();              /* User pressed cancel or file i/o messed up. */
}

void DrawWindow(Rect bounds)
{
    short i;
    PixMapHandle pm;
    CTabHandle ctab;
    ColorSpec *specs;

/*
 * make pixmap from scratch, this should work in future versions...
 */
    pm = (PixMapHandle)NewHandleClear(sizeof(PixMap));
    (**pm).baseAddr = (Ptr)Graph;
    (**pm).rowBytes = (1L << 15) | WIDTH;   /* hi bit means it's a PixMap */
    (**pm).bounds = bounds;
    (**pm).hRes = 72;
    (**pm).vRes = 72;
    (**pm).pixelSize = 8;
    (**pm).cmpCount = 1;
    (**pm).cmpSize = 8;
#if USE_GRAY
/*
 * munge a copy of the system color table for my offscreen world
 */
    ctab = GetCTable(log2(COLORS));
    specs = (**ctab).ctTable;
    for (i = 0; i < COLORS; ++i) {
        specs[i].rgb.red = specs[i].rgb.green = specs[i].rgb.blue
            = i * 65535 / (COLORS - 1);
        specs[i].value = i;
    }
```

```
        /* this alerts the color mgr that the table changed */
        (**ctab).ctSeed = GetCTSeed();
        (**pm).pmTable = ctab;
#else
        (**pm).pmTable = (*((CGrafPtr)thePort)->portPixMap)->pmTable;
#endif
        HLock(pm);
        CopyBits(*pm, &thePort->portBits, &(**pm).bounds, &thePort->portRect,
                        srcCopy, 0);
        HUnlock(pm);
}

/*
 * integer log base 2 function, to convert colors to bit depth
 */
short log2(unsigned short x)
{
        short t = 0;

        while (x >>= 1)
                ++t;
        return t;
}

void Init()
{
        InitGraf(&thePort);
        InitFonts();
        InitWindows();
        InitMenus();
        TEInit();
        InitDialogs(nil);
        InitCursor();
}
```