

```

/*
LC -bl -ms -v -cfistgmc -dl -j73 WheelGrad.c
blink from lib:c.o WheelGrad.o to WheelGrad lib lib:lc.lib lib:amiga.lib
Quit
*/

/*
 * WheelGrad.c - simple example of colorwheel and gradient slider
 *
 * Puts up a colorwheel and gradient slider and changes the gradient slider
 * color based on where the colorwheel knob is moved. This will get you
 * pointed in the right direction.
 *
 * The code will attempt to open the deepest possible screen by querying
 * the display database.
 */

#include <exec/types.h>
#include <exec/memory.h>
#include <intuition/intuition.h>
#include <intuition/intuitionbase.h>
#include <intuition/screens.h>
#include <graphics/displayinfo.h>
#include <intuition/gadgetclass.h>
#include <gadgets/colorwheel.h>
#include <gadgets/gradientslider.h>
#include <dos/dos.h>

#include <clib/intuition_protos.h>
#include <clib/exec_protos.h>
#include <clib/dos_protos.h>
#include <clib/graphics_protos.h>
#include <clib/colorwheel_protos.h>

#include <stdio.h>
#include <stdlib.h>

#ifdef LATTICE
int CXBRK(void) { return(0); } /* Disable Lattice CTRL/C handling */
int chkabort(void) { return(0); } /* really */
#endif

struct Library *IntuitionBase = NULL;
struct Library *GfxBase = NULL;
struct Library *ColorWheelBase = NULL;
struct Library *GradientSliderBase = NULL;

struct load32
{
    UWORD    132_len;
    UWORD    132_pen;
    ULONG    132_red;
    ULONG    132_grn;
    ULONG    132_blu;
};

void main(void)
{
    struct Screen *Myscreen;
    struct Window *Mywindow;
    struct IntuiMessage *msg;
    struct Gadget *colwheel, *gradslid;

#define GRADCOLORS 16 /* Set to 4 for ECS to ensure enough color wheel pens */

    ULONG colortable[96], mywinsig;
    struct load32 color_list[GRADCOLORS + 1];
    WORD penns[GRADCOLORS + 1];
    WORD i;
    BOOL Closeflag = FALSE;
    struct ColorWheelRGB rgb;
    struct ColorWheelHSB hsb;
    WORD numPens;
    ULONG modeID = HIRES_KEY;
    UWORD maxdepth;
    DisplayInfoHandle displayhandle;
    struct DimensionInfo dimensioninfo;

```

```

    ULONG gddires;
    ULONG exitvalue = RETURN_FAIL;

    if (IntuitionBase = OpenLibrary("intuition.library",39))
    if (GfxBase = OpenLibrary("graphics.library",39))
    if (ColorWheelBase = OpenLibrary("gadgets/colorwheel.gadget",39L))
    if (GradientSliderBase=OpenLibrary("gadgets/gradientslider.gadget",39L))
    if (displayhandle = FindDisplayInfo(modeID))
    if (gddires=GetDisplayInfoData(displayhandle,(UBYTE *) &dimensioninfo,
                                sizeof(struct DimensionInfo),
                                DTAG_DIMS,NULL))

    {
        maxdepth = dimensioninfo.MaxDepth;

        Myscreen = OpenScreenTags(NULL,
                                SA_Depth,          maxdepth,
                                SA_SharePens,       TRUE,
                                SA_LikeWorkbench,   TRUE,
                                SA_Interleaved,     TRUE,
                                SA_Title,          "WheelGrad Screen",
                                TAG_DONE);

        if (Myscreen)
        {
            /* Get colors and set up gradient slider as color 0. */

            /* get the RGB components of color 0 */
            GetRGB32(Myscreen->ViewPort.ColorMap,0L,32L,colortable);
            rgb.cw_Red   = colortable[0];
            rgb.cw_Green = colortable[1];
            rgb.cw_Blue  = colortable[2];

            /* now convert the RGB values to HSB, and max out B component */
            ConvertRGBToHSB(&rgb,&hsb);
            hsb.cw_Brightness = 0xffffffff;

            numPens = 0;
            while (numPens < GRADCOLORS)
            {
                hsb.cw_Brightness = 0xffffffff - ((0xffffffff / GRADCOLORS) * numPens);
                ConvertHSBToRGB(&hsb,&rgb);

                penns[numPens] = ObtainPen(Myscreen->ViewPort.ColorMap,-1,
                                           rgb.cw_Red,rgb.cw_Green,rgb.cw_Blue,PEN_EXCLUSIVE);
                if (penns[numPens] == -1)
                    break;

                /* Set up LoadRGB32() structure for this pen */
                color_list[numPens].132_len = 1;
                color_list[numPens].132_pen = penns[numPens];
                numPens++;
            }
            penns[numPens] = ~0;
            color_list[numPens].132_len = 0;

            /* Create gradient slider and colorwheel gadgets */
            gradslid = (struct Gadget *)NewObject(NULL,"gradientslider.gadget",
                                                GA_Top,          50,
                                                GA_Left,         177,
                                                GA_Width,        20,
                                                GA_Height,       100,
                                                GA_ID,           1L,
                                                GRAD_PenArray,    penns,
                                                PGA_Freedom,      LORIENT_VERT,
                                                TAG_END);

            colwheel = (struct Gadget *)NewObject(NULL, "colorwheel.gadget",
                                                GA_Top,          50,
                                                GA_Left,         50,
                                                GA_Width,        120,
                                                GA_Height,       100,
                                                WHEEL_Red,        colortable[0],
                                                WHEEL_Green,      colortable[1],
                                                WHEEL_Blue,       colortable[2],
                                                WHEEL_Screen,     Myscreen,
                                                WHEEL_GradientSlider, gradslid, /* connect gadgets */
                                                GA_FollowMouse,    TRUE,

```

```

        GA_Previous,      gradslid,
        GA_ID,            7L,
        TAG_END);

if (gradslid && colwheel)
{
    if (Mywindow = OpenWindowTags(NULL, WA_Left,      10,
        WA_Top,        20,
        WA_Height,     200,
        WA_Width,      400,
        WA_Title,      "WheelGrad Window",
        WA_CustomScreen, Myscreen,
        WA_IDCMP,       IDCMP_CLOSEWINDOW | IDCMP_MOUSEMOVE,
        WA_SizeGadget,  TRUE,
        WA_DragBar,     TRUE,
        WA_CloseGadget, TRUE,
        WA_Gadgets,     gradslid,
        TAG_DONE))
    {
        mywinsig = 1 << Mywindow->UserPort->mp_SigBit;

        do
        {
            Wait(mywinsig);

            while (msg = (struct IntuiMessage *)GetMsg(Mywindow->UserPort))
            {
                switch (msg->Class)
                {
                    case IDCMP_CLOSEWINDOW:
                        Closeflag = TRUE;
                        break;
                    case IDCMP_MOUSEMOVE:

                        /*
                         * Change gradient slider color each time
                         * colorwheel knob is moved. This is one
                         * method you can use.
                         */

                        /* Query the colorwheel */
                        GetAttr(WHEEL_HSB,colwheel,(ULONG *)&hsb);

                        i = 0;

                        while (i < numPens)
                        {
                            hsb.cw_Brightness =
                                0xffffffff - ((0xffffffff / numPens) * i);
                            ConvertHSBToRGB(&hsb,&rgb);

                            color_list[i].l32_red = rgb.cw_Red;
                            color_list[i].l32_grn = rgb.cw_Green;
                            color_list[i].l32_blu = rgb.cw_Blue;
                            i++;
                        }
                        LoadRGB32(&Myscreen->ViewPort,(ULONG *)color_list);
                        break;
                    }
                }
                ReplyMsg((struct Message *)msg);
            }
        } while (Closeflag == FALSE);
        CloseWindow(Mywindow);
    }

    /* Get rid of the gadgets */
    DisposeObject(colwheel);
    DisposeObject(gradslid);

    /* Always release the pens */
    while (numPens > 0)
    {
        numPens--;
    }
}

```

```

        ReleasePen(Myscreen->ViewPort.ColorMap,penns[numPens]);
    }

    CloseScreen(Myscreen);
    exitvalue = RETURN_OK;
}
else
    printf("Failed to open screen\n");
}

if (gdidres == 0)
    printf("Screen mode dimension information not available\n");

if (displayhandle == NULL)
    printf("Failed to find HIRES_KEY in display database\n");

if (GradientSliderBase)
    CloseLibrary(GradientSliderBase);
else
    printf("Failed to open gadgets/gradientslider.gadget\n");

if (ColorWheelBase)
    CloseLibrary(ColorWheelBase);
else
    printf("Failed to open gadgets/colorwheel.gadget\n");

if (GfxBase)
    CloseLibrary(GfxBase);
else
    printf("Failed to open graphics.library\n");

if (IntuitionBase)
    CloseLibrary(IntuitionBase);
else
    printf("Failed to open intuition.library\n");

exit(exitvalue);
}

```

