$SPAD/src/lib xspadfill.c

The Axiom Team

December 3, 2016

Abstract

Contents

This file contains the routines needed to dither using the spadcolors. The routines will have names such as XSpadFill, ... The user simply gives the normal arguments as with the corresponding XFill routine, with two additional arguments which choose the shade and the hue.

The file will maintain twoGC's: stippleGC - will be used when stippling the backgrounds. solidGC - will be used when the background should be solid

The user should call XSpadInit to get everthing going. This routine has the job of Initializing the dithering routines, and getting the colors all into place.

1 License


```
/*
Copyright (c) 1991-2002, The Numerical ALgorithms Group Ltd.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    - Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.

    - Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the
      distribution.

    - Neither the name of The Numerical ALgorithms Group Ltd. nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
```


—— * ——

```
#ifndef MSYSplatform
```

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

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>

#include "spadcolors.h"

#include "xspadfill.h1"
#include "xshade.h1"
#include "xdither.h1"
#include "spadcolors.h1"

extern unsigned long *spadColors;
static GC  stippleGC, solidGC;
Colormap cmap;
int SpadFillInit = 0;
long white, black;
int max_spad_shades;
extern Display *dsply;

extern int totalHues;
extern int totalDithered;
extern int totalSolid;
extern int totalShades;
extern int totalColors;
extern int maxGreyShade;

int
XInitSpadFill(Display *dsply, int scr, Colormap * mapOfColors, int * hues,
              int *solid, int * dithered, int * shades)
{
    XColor BlackColor, WhiteColor;
    XColor retColor;
    int maxSolid;

    SpadFillInit = 1;


    /*
     * First thing I should do is get the GC's
     */
    stippleGC = XCreateGC(dsply, RootWindow(dsply, scr), 0, NULL);
    solidGC = XCreateGC(dsply, RootWindow(dsply, scr), 0, NULL);
    XSetArcMode(dsply, solidGC, ArcPieSlice);
    XSetArcMode(dsply, stippleGC, ArcPieSlice);
```

```
cmap = DefaultColormap(dsply, scr);
*mapOfColors = cmap;
XAllocNamedColor(dsply, cmap, "Black", &BlackColor, &retColor);
XAllocNamedColor(dsply, cmap, "White", &WhiteColor, &retColor);
black = BlackColor.pixel;
white = WhiteColor.pixel;

/*
 * Now I check to see if I am on a monochrome display. If so then I
 * simply set totalHues to be one, and total Shades to be 2. I also have
 * to allocate balck and white colors. This I put into the first two
 * memory locations of spadcolors.
 *
 * was      if(DisplayPlanes(dsply, scr) < 2)  changed temporarily to < 8
 * because of problems with screens with 4 planes . Now if we don't have
 * 8 planes to play with  we switch to monochrome
 */

if (DisplayPlanes(dsply, scr) < 8) {
    *dithered = totalDithered = maxGreyShade = XInitShades(dsply, scr);
    spadColors = (unsigned long *) malloc(2 * sizeof(unsigned long));
    spadColors[0] = BlackColor.pixel;
    spadColors[1] = WhiteColor.pixel;
    *hues = totalHues = 1;
    *solid = totalSolid = 2;
    *shades = totalColors = totalShades = totalDithered;
    return (totalColors);
}

/*
 * Now I have to get all the spad colors as every good spad program
 * should Now I should initialize the dithering routines
 */

*dithered = totalDithered =
    XInitDither(dsply, scr, stippleGC, black, white);

if ((maxSolid=makeColors(dsply,scr,&cmap,&spadColors,&totalSolid)) > 0) {
    *solid = totalSolid + 2;
    *hues = totalHues = maxSolid / totalSolid;
    *shades = totalShades = (totalSolid + 1) * (totalDithered - 1) + 1;
    totalColors = totalHues * totalShades;
    return (totalColors);
}
else {

    /*
     * makeColors managed to fail -- switch to mono
```

```
         */
        *dithered = totalDithered = maxGreyShade = XInitShades(dsply, scr);
        spadColors = (unsigned long *) malloc(2 * sizeof(unsigned long));
        spadColors[0] = BlackColor.pixel;
        spadColors[1] = WhiteColor.pixel;
        *hues = totalHues = 1;
        *solid = totalSolid = 2;
        *shades = totalColors = totalShades = totalDithered;
        return (totalColors);
    }
}


void
XSpadFillSetArcMode(Display *dsply, int mode)
{
    XSetArcMode(dsply, solidGC, mode);
    XSetArcMode(dsply, stippleGC, mode);
}

GC
SpadFillGC(Display *dsply,int  hue, int theshade,char * fill_routine)
{
    int dither;
    int color;


    if (!SpadFillInit) {
        fprintf(stderr,
           "Tried to use SpadFillGC before calling XInitSpadFill\n");
        exit(0);
    }

    if (theshade >= totalShades) {
        fprintf(stderr, "Shade %d out of range\n",theshade);
        exit(-1);
    }
    if (hue >= totalHues) {
        fprintf(stderr, "Error Hue %d is out of range\n",hue);
        exit(-1);
    }
    dither = ((theshade) % (totalDithered - 1));
    if (dither != 0) {
        XChangeDither(dsply, stippleGC, dither);
        if (theshade < totalDithered) {    /* Dither to black */
            color = totalSolid * hue;
            XSetForeground(dsply, stippleGC, black);
            XSetBackground(dsply, stippleGC, spadColors[color]);
        }
        else if (theshade > (totalShades - totalDithered)) {
```

```
            /* Dither to white */
            color = ((theshade) / (totalDithered - 1)) + totalSolid * hue - 1;
            XSetForeground(dsply, stippleGC, spadColors[color]);
            XSetBackground(dsply, stippleGC, white);
        }
        else {
            color = ((theshade) / (totalDithered - 1)) + totalSolid * hue - 1;
            XSetForeground(dsply, stippleGC, spadColors[color]);
            XSetBackground(dsply, stippleGC, spadColors[color + 1]);
        }
        return (stippleGC);
    }
    else {
        if (theshade == 0)
            XSetForeground(dsply, solidGC, black);
        else if (theshade == (totalShades - 1))
            XSetForeground(dsply, solidGC, white);
        else {
            color = ((theshade) / (totalDithered - 1)) + totalSolid * hue - 1;
            XSetForeground(dsply, solidGC, spadColors[color]);
        }
        return (solidGC);
    }

}

unsigned long
XSolidColor(int hue, int theshade)
{
    if (hue >= totalHues)
        return -1;
    if (theshade >= totalSolid)
        return -1;
    return (spadColors[hue * (totalSolid) + theshade]);
}

void
XSpadFillRectangle(Display *dsply, Drawable drawable, int x, int y,
                   unsigned int width, unsigned int height,
                   int hue, int theshade)
{

    XFillRectangle(dsply, drawable,
                   SpadFillGC(dsply, hue, theshade, "XSpadFillRectangle"),
                   x, y, width, height);

}


void
```

```
XSpadFillRectangles(Display *dsply, Drawable drawable,
                    XRectangle * rectangles, int nrectangles,
                    int  hue, int theshade)
{


    XFillRectangles(dsply, drawable,
                        SpadFillGC(dsply, hue, theshade, "XSpadFillRectangle"),
                           rectangles, nrectangles);

}


void
XSpadFillPolygon(Display *dsply, Drawable drawable, XPoint * points,
                 int npoints, int shape, int mode, int hue, int theshade)
{
    XFillPolygon(dsply, drawable,
                        SpadFillGC(dsply, hue, theshade, "XSpadFillRectangle"),
                        points, npoints, shape, mode);

}

void
XSpadFillArc(Display *dsply, Drawable drawable, int x, int y,
             unsigned int width, unsigned int height,
             int angle1, int angle2,int hue,int  theshade)
{

    XFillArc(dsply, drawable,
                    SpadFillGC(dsply, hue, theshade, "XSpadFillRectangle"),
                    x, y, width, height, angle1, angle2);
}


void
XSpadFillArcs(Display *dsply, Drawable drawable, XArc * arcs, int narcs,
              int hue, int theshade)
{
    XFillArcs(dsply, drawable,
                     SpadFillGC(dsply, hue, theshade, "XSpadFillArcs"),
                     arcs, narcs);
}


#endif /* MSYSplatform */
```

———-

References

[1] nothing