```
//
//      GRGradientButtonCell : ButtonCell
//
//      By Anders Bertelrud
//      Copyright (c) 1995-1996 Anders Bertelrud
//

#import <dpsclient/psops.h>
#import <dpsclient/wraps.h>
#import <appkit/NXImage.h>
#import <appkit/View.h>
#import "GRGradientButtonCell.h"
#import "GRGradientFunctions.h"



//
//      GRGradientButtonCell class
//
@implementation GRGradientButtonCell


// Class globals
static NXImage *        GRBlankImage = nil;         // This is needed because of problems with IB.


////////////////////////////////////// Private functions //////////////////////////////////////


static inline void GRAdjustRectangle (NXRect * rectangle, float leftInset, float topInset,
                                                        float rightInset, float bottomInset, BOOL viewIsFlipped)
        // Adjusts the given rectangle by insetting each edge by the corresponding amount. If the
        // flag ªviewIsFlippedº is YES, the adjustments are appropriate for a rectangle in a flipped
        // coordinate system; if it is NO, the adjustments will be correct only if the rectangle
        // is in a non-flipped coordinate system.
{
```

```
	if (rectangle)
	{
		rectangle->origin.x += leftInset;
		rectangle->origin.y += (viewIsFlipped ? topInset : bottomInset);
		rectangle->size.width -= (leftInset + rightInset);
		rectangle->size.height -= (topInset + bottomInset);
	}
}


static inline void GROffsetRectangle (NXRect * rectangle, float dx, float dy,
                                                        BOOL viewIsFlipped)
	// Offsets the given rectangle by the given amounts. If ªviewIsFlippedº is YES, the offsets
	// are appropriate for a rectangle in a flipped coordinate system; if it is NO, the offsets
	// will be correct only if the rectangle is in a non-flipped coordinate system.
{
	if (rectangle)
	{
		rectangle->origin.x += dx;
		rectangle->origin.y += (viewIsFlipped ? -dy : dy);
	}
}


static inline NXImage * GRCreateDimmedImageWithZone (NXZone * zone, NXImage * image)
	// Creates and returns a dimmed version of the given image by compositing it into a
	// transparent image of the same size but with 50% alpha. The resulting image can be
	// composited anywhere with NX_SOVER and will let half of the background show through.
	// In our case the background is a gray gradient; the result is a dimmed effect. The
	// NXImage is allocated in the given zone.
{
	NXImage *	dimmedImage;
	NXRect		imageBounds = {0, 0, 0, 0};

	[image getSize:&imageBounds.size];
	dimmedImage = [[NXImage allocFromZone:zone] initSize:&imageBounds.size];
```

```
	if ([dimmedImage lockFocus])
	{
		PSsetgray(NX_BLACK);
		PSsetalpha(0.5);
		NXRectFill(&imageBounds);
		[image composite:NX_SIN toPoint:&imageBounds.origin];
		[dimmedImage unlockFocus];
		return dimmedImage;
	}
	else
		return [dimmedImage free];
}



////////////////////////// Creating and destroying GRGradientButtonCells //////////////////////////


+ initialize
{
	// This is very ugly. It seems that IB cannot deal with button cells that have nil images.
	// We create a small blank image to use for "no image", and set it in -initIconCell:.
	if (self == [GRGradientButtonCell self])
	{
		NXRect	r = {0, 0, 1, 1};

		GRBlankImage = [[NXImage alloc] initSize:&r.size];
		if ([GRBlankImage lockFocus])
		{
			PSsetalpha(0.0);
			NXRectFill(&r);
			PSsetalpha(1.0);
			[GRBlankImage unlockFocus];
		}
	}
	return self;
```

```
}


- initIconCell:(const char *)iconName                                    // d.i.
{
        self = [super initIconCell:iconName];
        dimmedWhenDisabled = YES;
        [self setImage:GRBlankImage];           // See comment in +initialize
        return self;
}


- initTextCell:(const char *)textString                                  // d.i.
{
        // GRGradientButtonCells cannot display text. Sorry...
        return [self initIconCell:NULL];
}


- init
{
        return [self initIconCell:NULL];
}


- copyFromZone:(NXZone *)zone
{
        self = [super copyFromZone:zone];
        dimmedNormalImage = [dimmedNormalImage copyFromZone:zone];
        dimmedAlternateImage = [dimmedAlternateImage copyFromZone:zone];
        return self;
}


- free
{
```

```
        [dimmedAlternateImage free];
        [dimmedNormalImage free];
        return [super free];
}


////////////////////////// Setting whether to dim when cell is disabled //////////////////////////


- (BOOL)isDimmedWhenDisabled
{
        return dimmedWhenDisabled;
}


- (void)setDimmedWhenDisabled:(BOOL)flag
{
        dimmedWhenDisabled = flag;
}



////////////////////////////////////// Setting the icon //////////////////////////////////////


- setImage:(NXImage *)image
{
        [dimmedNormalImage free];
        dimmedNormalImage = nil;
        if (image != nil)
        {
//              ALAssert1([image isKindOf:[NXImage self]], "Image, if not nil, must be a kind of "
//                 "NXImage (but it's %s).", ALDS(image));
                dimmedNormalImage = GRCreateDimmedImageWithZone([self zone], image);
        }
```

```
        [super setImage:image];
        return self;
}


- setAltImage:(NXImage *)image
{
        [dimmedAlternateImage free];
        dimmedAlternateImage = nil;
        if (image != nil)
        {
//              ALAssert1([image isKindOf:[NXImage self]], "Image, if not nil, must be a kind of "
//                 "NXImage (but it's %s).", ALDS(image));
                dimmedAlternateImage = GRCreateDimmedImageWithZone([self zone], image);
        }
        [super setAltImage:image];
        return self;
}



////////////////////////////////// Drawing and highlighting //////////////////////////////////


- drawInside:(const NXRect *)cellFrame inView:(View *)view
{
        NXRect                  inside;
        register NXImage *      image;
        register BOOL           viewIsFlipped;
        register BOOL           dwd;


        viewIsFlipped = [view isFlipped];
        dwd = dimmedWhenDisabled;
        inside = *cellFrame;
```

```
// If the button cell is bordered (which actually means "raised bezel" in the case of
// ButtonCells), we adjust the drawing rectangle accordingly.
if (bcFlags1.bordered)
        GRAdjustRectangle(&inside, 1, 1, 2, 2, viewIsFlipped);
image = (dwd && cFlags1.disabled) ? dimmedNormalImage : icon.bmap.normal;

// [@@] Still something wrong with this code: if a button whose state is 1 is pressed, the
// button isn't highlighted as it should. [fixa detta sÚ smÚningom]
if (cFlags1.highlighted)
{
        // Highlighted.
        if (bcFlags1.bordered && bcFlags1.pushIn)
                GROffsetRectangle(&inside, 1, -1, viewIsFlipped);
        if (bcFlags1.lightByBackground || bcFlags1.lightByGray)
                GRDrawGrayGradient(inside, (NX_LTGRAY+NX_WHITE)/2.0, NX_WHITE);
        else
                GRDrawGrayGradient(inside, (NX_DKGRAY+NX_LTGRAY)/2.0, (NX_LTGRAY+NX_WHITE)/2.0);
        // The state-checking below is to make change-by-toggle work right.
        if (bcFlags1.lightByContents && !(bcFlags1.changeContents && cFlags1.state))
                image = (dwd && cFlags1.disabled) ? dimmedAlternateImage : icon.bmap.alternate;
}
else if (cFlags1.state)
{
        // Alternate state.
        if (bcFlags1.changeBackground || bcFlags1.changeGray)
                GRDrawGrayGradient(inside, (NX_LTGRAY+NX_WHITE)/2.0, NX_WHITE);
        else
                GRDrawGrayGradient(inside, (NX_DKGRAY+NX_LTGRAY)/2.0, (NX_LTGRAY+NX_WHITE)/2.0);
        if (bcFlags1.changeContents)
                image = (dwd && cFlags1.disabled) ? dimmedAlternateImage : icon.bmap.alternate;
}
else
        GRDrawGrayGradient(inside, (NX_DKGRAY+NX_LTGRAY)/2.0, (NX_LTGRAY+NX_WHITE)/2.0);

// If it's an icon cell, draw the icon (for now always centered).
if (cFlags1.type == NX_ICONCELL && image != nil)
```

```
    {
        NXSize    imageSize;
        NXPoint   pt;

        [image getSize:&imageSize];
        pt.x = NX_MIDX(&inside) - imageSize.width/2;
        if (pt.x < NX_X(&inside))
            pt.x = NX_X(&inside);
        if (viewIsFlipped)
        {
            pt.y = NX_MIDY(&inside) + imageSize.height/2;
            if (pt.y > NX_MAXY(&inside))
                pt.y = NX_MAXY(&inside);
        }
        else
        {
            pt.y = NX_MIDY(&inside) - imageSize.height/2;
            if (pt.y < NX_Y(&inside))
                pt.y = NX_Y(&inside);
        }
        [image composite:NX_SOVER toPoint:&pt];
    }
    bcFlags1.lastState = (cFlags1.state ? YES : NO);
    return self;
}



//////////////////////////////////// Reading from typed streams ////////////////////////////////////


- read:(NXTypedStream *)stream
{
    self = [super read:stream];
    dimmedWhenDisabled = YES;
    if (cFlags1.type == NX_ICONCELL && icon.bmap.normal != nil)
```

```
        dimmedNormalImage = GRCreateDimmedImageWithZone([self zone], icon.bmap.normal);
    if (cFlags1.type == NX_ICONCELL && icon.bmap.alternate != nil)
        dimmedAlternateImage = GRCreateDimmedImageWithZone([self zone], icon.bmap.alternate);
    return self;
}


@end
```