```
//
//	GRGradientFrameView : FrameView
//
//	By Anders Bertelrud
//	Copyright (c) 1995-1996 Anders Bertelrud
//

#import <math.h>
#import <dpsclient/psops.h>
#import <defaults/defaults.h>
#import <appkit/Application.h>
#import "GRGradientFunctions.h"
#import "GRGradientFrameView.h"



//
//	Names of defaults strings
//
#define		GRGradientTitleBarsEnabled		"GRGradientTitleBarsEnabled"
#define		GRGradientTitleBarHue			"GRGradientTitleBarHue"
#define		GRGradientTitleBarSaturation	"GRGradientTitleBarSaturation"



//
//	GRGradientFrameView class
//
@implementation GRGradientFrameView


// Class globals
static float	GRGradientHue = .666667;			// Hue used for gradient
static float	GRMenuSaturation = 0.5;				// Saturation for menus
static float	GRKeyWindowSaturation = 0.5;		//	- " -	for key window
static float	GRMainWindowSaturation = 0.25;		//	- " -	for main window
static float		GROtherWindowSaturation = 0.0;		//	- " -	for all other windows



+ (void)install
{
	[self poseAs:[FrameView self]];
}


+ (void)installIfAppropriate
{
	const char *	enabledValue;

	enabledValue = NXGetDefaultValue([NXApp appName], GRGradientTitleBarsEnabled);
	if (enabledValue && strcmp(enabledValue, "Yes") == 0)
		[self install];
}


+ initialize
{
	if (self == [GRGradientFrameView self])
	{
		NXDefaultsVector	titleBarDefaults = {
						{GRGradientTitleBarsEnabled, "NO"},
						{GRGradientTitleBarHue, "0.6666667"},
						{GRGradientTitleBarSaturation, "0.5"},
						{NULL}
					};
		const char *		huePreference;
		const char *		saturationPreference;


		// Register defaults.
		NXRegisterDefaults([NXApp appName], titleBarDefaults);
		// Get hue and saturation constants from the defaults database.
		huePreference = NXGetDefaultValue([NXApp appName], GRGradientTitleBarHue);
		if (huePreference && huePreference[0])
			GRGradientHue = atof(huePreference);
		saturationPreference = NXGetDefaultValue([NXApp appName], GRGradientTitleBarSaturation);
		if (saturationPreference && saturationPreference[0])
		{
			GRMenuSaturation = atof(saturationPreference) * 1.0;
			GRKeyWindowSaturation = atof(saturationPreference) * 1.0;
			GRMainWindowSaturation = atof(saturationPreference) * 0.5;
			GROtherWindowSaturation = atof(saturationPreference) * 0.0;
		}
	}
	return self;
}


static inline void GRClipRectFill (NXRect * origRect, const NXRect * visRect)
	// When this function is compiled with -O optimization, the compiler generates assembly
	// code identical to NeXT's private ClipRectFill.
{
	if (NXIntersectionRect(visRect, origRect))
		NXRectFill(origRect);
}


static inline void GRClipFrameRect (NXRect * origRect, const NXRect * visRect)
	// When this function is compiled with -O optimization, the compiler generates assembly
	// code identical to NeXT's private ClipFrameRect.
{
	NXRect	r = *origRect;
	NXRect	sliceRect;

	if (NXContainsRect(visRect, &r))
		NXFrameRect(&r);
	else
	{
		register int	i;

		for (i = NX_XMIN; i <= NX_YMAX; i++)
		{
			NXDivideRect(&r, &sliceRect, 1.0, i);
			GRClipRectFill(&sliceRect, visRect);
		}
	}
}


static void _GRDrawTitleBar (id self, NXRect titleBarRect, const NXRect * visRect,
							float saturation, float startBrightness, float endBrightness,
							float titleGray, BOOL menuStyle)
{
	NXRect			sliceRect;


	// Rita ramen.
	if (menuStyle)
	{
		// Rita ðvre och vÙnstra kanten.
		PSsetgray(NX_DKGRAY);
		NXDivideRect(&titleBarRect, &sliceRect, 1, NX_XMIN);
		GRClipRectFill(&sliceRect, visRect);
		NXDivideRect(&titleBarRect, &sliceRect, 1, NX_YMAX);
		GRClipRectFill(&sliceRect, visRect);

		// Rita nedre och hðgra kanten.
		PSsetgray(NX_BLACK);
		NXDivideRect(&titleBarRect, &sliceRect, 1, NX_XMAX);
		GRClipRectFill(&sliceRect, visRect);
		NXDivideRect(&titleBarRect, &sliceRect, 1, NX_YMIN);
		GRClipRectFill(&sliceRect, visRect);
	}
	else
	{
		PSsetgray(NX_BLACK);
		GRClipFrameRect(&titleBarRect, visRect);
		NXInsetRect(&titleBarRect, 1, 1);
	}

	// Rita ðvre och vÙnstra kanten.
	PSsethsbcolor(GRGradientHue, saturation * .5, .8);
	NXDivideRect(&titleBarRect, &sliceRect, 1, NX_XMIN);
	GRClipRectFill(&sliceRect, visRect);
	NXDivideRect(&titleBarRect, &sliceRect, 1, NX_YMAX);
	GRClipRectFill(&sliceRect, visRect);

	// Rita nedre och hðgra kanten.
	PSsethsbcolor(GRGradientHue, saturation, .2);
	NXDivideRect(&titleBarRect, &sliceRect, 1, NX_XMAX);
	GRClipRectFill(&sliceRect, visRect);
	NXDivideRect(&titleBarRect, &sliceRect, 1, NX_YMIN);
	GRClipRectFill(&sliceRect, visRect);

	// Rita gradienten.
	GRDrawHSBGradient(titleBarRect, GRGradientHue, saturation, startBrightness, endBrightness);

	// Rita titeln.
	[self _drawTitleStringIn:visRect withColor:titleGray];
}


- _drawTitledFrame:(const NXRect *)rectangles :(int)nrRectangles
{
	const NXCoord	titleBarHeight = 23;
	const NXRect *	visRect = rectangles;
	NXRect			titleBarRect, r;
	NXRect			restOfWindowRect;
	restOfWindowRect = bounds;
	NXDivideRect(&restOfWindowRect, &titleBarRect, titleBarHeight, NX_YMAX);
	r = titleBarRect;
	if (NXIntersectionRect(visRect, &r))
	{
		if ([window isKeyWindow])
			_GRDrawTitleBar(self, titleBarRect, &rectangles[0], GRKeyWindowSaturation,
			   .2, .5, NX_WHITE, NO);
		else if ([window isMainWindow])
			_GRDrawTitleBar(self, titleBarRect, &rectangles[0], GRMainWindowSaturation,
			   .2, .5, NX_WHITE, NO);
		else
			_GRDrawTitleBar(self, titleBarRect, &rectangles[0], GROtherWindowSaturation,
			   .4, .7, NX_BLACK, NO);
	}
	[super _drawTitledFrame:&restOfWindowRect :1];
	return self;
}


- _drawMenuFrame:(const NXRect *)rectangles :(int)nrRectangles
{
	const NXCoord	titleBarHeight = 23;
	const NXRect *	visRect = rectangles;
	NXRect			titleBarRect, r;
	NXRect			restOfWindowRect;

	restOfWindowRect = bounds;
	NXDivideRect(&restOfWindowRect, &titleBarRect, titleBarHeight, NX_YMAX);
	r = titleBarRect;
	if (NXIntersectionRect(visRect, &r))
	{
		_GRDrawTitleBar(self, titleBarRect, &rectangles[0], GRMenuSaturation, .2, .5, NX_WHITE,
		   YES);
	}
	[super _drawMenuFrame:&restOfWindowRect :1];
	return self;
}



@end
```