GradientKit.tiff ¬

by

Anders Bertelrud
anders@vt.edu
Written: December 1995
Released: April 1996

Introduction

This is a little hack I wrote late last year to make NeXTSTEP 3.3 window title bars blue instead of black. On the net I saw a screen shot of a prerelease of 4.0, and I liked the blue gradients used for the title bars. I decided I wanted them immediately. The gradient push buttons also looked quite nice, so I subclassed ButtonCell and Button to provide this behaviour. These button classes also provide another feature: they can automatically make the displayed icon dimmed when the button is disabled.

The gradient title bars require the subclassing of an undocumented AppKit class. This is rather unorthodox, and there will almost certainly be compatibility problems under NeXTSTEP 4.0 / OpenStep. But by then all the title bars should be blue anyway, so this hack will become worthless. The classes that implement gradient buttons, though, should continue to work. They don't use any hidden features or secret classes. But hopefully in 4.0 the NSButton class will draw gradients, making these subclasses unnecessary.

In the meantime, enjoy.

For the record...

Let me state for the record that I have not seen any prerelease of NeXTSTEP 4.0 or of OpenStep, nor have I had its user interface described to me by anyone who has. The inspiration for and the design of this little hack came solely from the screen shot that was made publicly available on the net last year.

Comments

Since this is a quick hack, I have spent very little time on documentation. But these few comments might help anybody interested in using it.

All classes, constants, etc. are prefixed with ªGRº (for gradients). The gradient code is actually part of a private code library, and it was easiest to just replace my two-character prefix with ªGRº.

The title bars can actually be of any colour, not just blue. See the section ªDetails of gradient title barsº for more info.

The *GRGradientFunctions* module defines two functions that are used to draw gradients. They are completely stand-alone and can be used separately from any of the other source code in this package. The *GRGradientFrameView* class implements the gradient title bars, and it depends only upon the GRGradientFunctions module. The *GRGradientButtonCell* and *GRGradientButton* classes implement gradient push buttons, and also depend only on GRGradientFunctions. Gradient title bars and gradient buttons can thus be used separately.

Gradient title bars are only drawn on the windows that belong to the application containing the GRGradientFrameView class. In order to do a systemwide replacement, the shared libraries would have to be patched. This would require root access. [If anyone comes up with another way of achieving system-wide gradient title bars, please let me know.]

The following two sections describe what you need to do to use gradient title bars and gradient push buttons, respectively.

Details of gradient title bars

If you use *AppInspector* (in /NextDeveloper/Demos) to examine any NeXTSTEP 3 application, you will find that each window has an instance of the undocumented class FrameView. It seems that this FrameView

instance is actually at the top of the window's view hierarchy, and that the window's official content view (as returned by Window's -contentView method) is a subview of this frame view. By creating a subclass of FrameView that overrides the drawing method, and by having that subclass pose as the FrameView class, the appearance of the title bars of all windows in the application can be changed. Note that this doesn't change the appearance of the miniaturize and close buttons.

The GRGradientFrameView class is a subclass of the undocumented FrameView class. To use it, invoke the +install method of the GRGradientFrameView class. The related method +installIfAppropriate checks the default "GRGradientTitleBarsEnabled" (in the defaults database), and if the value is "Yes" gradient title bars are installed.

The +install or +installIfAppropriate method should be invoked before any Window instances are created. So the place to do this is in the main file, after the application object has been created but before the main nib file is loaded:

```
/* Generated by the NeXT Project Builder
   NOTE: Do NOT change this file -- Project Builder maintains it.
*/
#import <appkit/appkit.h>
#import "GRGradientFrameView.h"

void main(int argc, char *argv[]) {
    [Application new];
    [GRGradientFrameView install];
    if ([NXApp loadNibSection:"abc.nib" owner:NXApp withNames:NO])
        [NXApp run];
    [NXApp free];
    exit(0);
}
```

Be sure to uncheck ProjectBuilder's "Generate Main File on Save" option before you modify the generated main file.

Besides the default to enable/disable gradient title bars, there are two other ones that control the appearance of the title bars. They are "GRGradientTitleBarHue" and "GRGradientTitleBarSaturation". The value of the former is a real number between 0.0 and 1.0 that defines the hue of the gradient, and the latter is the saturation (also a real number between 0.0 and 1.0). A saturation value of zero will make the title bars gray. If you change the hue, you probably need to change the saturation too in order to make the title bars

look good.

The title bars of the key window and of menus are drawn using the full saturation value. The main window, if different from the key window, is drawn using half the given saturation (these are windows that normally show up as dark gray). Title bars of inactive windows (normally shown in flat light gray) receive none of the given saturation, making them light gray (with a gradient, though).

Definition of these defaults is optional. If neither is defined, values that make the title bars look similiar to those in the NeXTSTEP 4.0 screenshot will be used.

A small ugliness: the minimum and maximum brightnesses of gradients is hardcoded in GRGradientFrameView. Very bad design, I know, but this is after all a hack. And you have the source code. **:** -)

A larger ugliness: right now the title bar is assumed to be 23 pixels tall. This is **really** bad design, but I couldn't find a convenient way of determining the height of the title bar. Since nobody I know uses anything but Helvetica 12pt (maybe we're all just boring people...), the issue hasn't come up.

Details of gradient buttons

The GRGradientButtonCell and GRGradientButton classes implement buttons with gradient backgrounds. This is quite straightforward, no tricky stuff here. GRGradientButtonCell does the actual work. It always draws a gradient with a centered icon on top (as in the NeXTSTEP 4.0 screen shot). It currently does not support text labels.

Something it does support is dimmed versions of icons. When a normal NeXTSTEP ButtonCell showing text is disabled, the text is shown in dark gray. Great. But when a ButtonCell showing an icon is disabled, there's no visual clue to that effect. So instances of GXGradientButtonCell create dimmed versions of the normal and alternate icons, and optionally show one of those icons when the button cell is disabled. By default the option is on, but it can be turned changed using the -setDimmedWhenDisabled: method.

One small problem is that the dimmed images take up extra memory. In practice, the icons shown on buttons tend to be rather small, so this shouldn't really be a big problem. The code isn't very smart; it creates dimmed versions of images even if the button cell is never disabled. This could be fixed quite easily. Mostly a bunch of if-statements. Perhaps in the future...

GRGradientButtonCells can be used in Matrices, and they obey normal ButtonCell settings (except settings related to text, since GRGradientButtonCells don't display text). The GRGradientButton class simply sets the cell class to be GRGradientButtonCells, and provides the convenience method -setDimmedWhenDisabled: that just passes the message along to the button cell. This is what the AppKit's Button class does in many of its methods.

Final comments

The source code is provided in an Interface Builder palette. By building the palette and installing it in InterfaceBuilder, you will be able to drag gradient buttons off the palette and edit them like any other buttons in InterfaceBuilder. The sample application is an example of use of these classes. Be sure to link in the GRGradientFunctions module and the GRGradientButtonCell and GRGradientButton classes if you drag them off the InterfaceBuilder palette and into your nib files. Also keep in mind that if a nib file includes an object dragged off a palette, that palette will need to be available to InterfaceBuilder whenever that nib file is opened.

I hope these comments provide enough information for anybody who wants to to start using the included code. The source code contains (a few) more comments. If you have any questions or comments, feel free to email me at *anders@vt.edu.* And if you make any cool improvements, I'd love to hear from you!

Anders Bertelrud
April 23 1996