MiscTabMatrixPalette v.2.1 (Apr 25, 1997)

Enhancements:
- Corrected "noisy" pixels
- Added custom IBInspector for MiscTabButtonCell

Bugs:
- As the NSMatrix sometimes draws as little as possible, it may happen that the cells are not correctly refreshed when changing selected cell. This particularly happens when using keyEquivalents, or multiple rows (when cells adjacent to neither old selection, nor new selection)
- Some problems with the IBInspector...
- Position of tabs is implemented only for standard (top) position.

MiscTabMatrixPalette v.2.0b1 (Apr 1st, 1997)

Here is a new version of the palette. This palette is totally OpenStep compliant (even STRICT_OPENSTEP), and introduces a bunch of new features:
- it now uses a subclass of NSButtonCell, MiscTabButtonCell. This allows the user to use all the features the NSButtonCell implements, like adding images, formatting title, using mnemonics, etc.
- you can use matrix with multiple rows: tab display will be correctly handled, but there are some limitations and bugs (see next paragraphs).
- it uses NSControl pre-defined colors, but in a static way: the colors are defined in static variables, during +initialization of MiscTabButtonCell. Could this pose a problem when changing screen bit depth or using different bit depth screens?
- as the implementation has been severally modified, some bugs in display process have disappeared. You can now correctly display a matrix with only one column, the tabs are correctly redraw when resigning firstResponder, etc. The drawInteriorWithFrame:inView: does no longer ask its controlView to redraw some other cells (but it has some inconvenients :-()

Though, it still has a few limitations, mainly inherent to the fact the we use NSMatrix and, sorry, there are some known bugs:
- a "noisy" pixel is drawn on the base line of tabs. I just need some more time to correct it.
- when using multiple rows matrices, the base line of the tabs is not always correctly redrawn, especially when using arrow keys to navigate inside the matrix. NSMatrix does redraw only the old-selected and new-selected cells, whereas if you click with the mouse, the whole matrix is redrawn; that leaves unrefreshed cells.
- after a mouseDown, the cells are not correctly displayed (does the matrix clip the view?),

but everything is completely refreshed after mouseUp.

What can be done further?
- correct the bugs!
- allow the user to navigate with the arrow keys in the matrix without selecting a tab, until the user pushes <SPACE> or <RETURN>.
- allow tab matrix to be used upside-down, i.e. put the matrix at the base of a view, not at its top.
- allow the matrix to be used in a vertical way, on the left or right side of a view. This will be a challenge to do it, as NSButtonCell can not display text rotated to 90°. We could also display text vertically, one char per line, but IMHO this would not be very nice.
- in order to fully support multiple rows matrices, subclass NSMatrix to allow row rearranging when selecting a tab which is not on the bottom row, or at least display "hidden" tabs in a different way (ghostly), to have a consistent behaviour.

StÝphane CorthÝsy
stephane@lysis.ch

Conversion to OpenStep: July 1996.

Here it is... There seem to be some quirks with OpenStep bugs (mainly font setting) but essentially it all works.
Also, a decision should be made about it using NSAttributedString for the text drawing, etc. I figure it's best to make it STRICT_OPENSTEP compatible though... I should probably add support for the Windows system colors. Maybe I should use the NSInterfaceStyle to have it draw those lovely Windows tabs on the OpenStep/NT? This is totally untested on OpenStep for Mach.

-sean
Sean.Hill@iphysiol.unil.ch

MiscTabActionCell v1.5 (Feb 4, 1995) MiscKit 1.4.1

This version adds most of the enhancements of UITabActionCell versions 1.4 and 1.5. The Makefile stuff was intentionally left out for uniformity with the rest of the MiscKit.

-Don Yacktman
don@misckit.com

Changes for UITabActionCell (1.4, 1.5):

I added code to check to see if the window depth can handle above two-bit gray. If it can it uses the .5 grayscale. In addition cells scale to the matrix by default.

I love those autoscaling tabs!

Sean Hill
shill@iphysiol.unil.ch

PS Sorry for taking out the version and make file enhancements.... They caused more warnings and problems in a generic development environment than I wanted to answer questions about.

Several enhancements were added to try to make TabMatrix emulate EOF IB's StackView more faithfully:

1. Visual indication of a disabled tab is provided as with most NEXTSTEP UI objects: the tab label's text is light gray for a disabled tab and black for an enabled tab.

2. An unselected tab is now dark gray instead of medium gray (0.5). The use of either dark or light gray text as an indication of a disabled tab isn't very readable on mono systems due to the dithering required to display 0.5 gray.

3. The fixed-resolution tab end tiffs have been replaced by device-independent pswraps that draw the tab ends. This eliminates the need to provide the tiffs as resources to each app that uses TabMatrixPalette.

4. Automatic scaling of tab sizes to accommodate different fonts is now supported.

5. The palette view has been enhanced to provide a dark gray border across the top of the tabs to eliminate the chopped-off appearance.

6. Version information is recorded in the palette object files and the palette itself through the addition of headers in each source file and a modified version of MiscKit's buildversion.

7. Project Makefiles have been enhanced:
    a. automatically create and install in /usr/local/lib the palette library that must be linked into any app using the palette;
    b. automatically create Makefile.dependencies if they don't already exist (note that this will fail when multiarchitecture binaries are being built unless the broken Makefile.dependencies target is replaced in /NextDeveloper/Makefiles/app/common.make by the Makefile.dependencies target included in the common.make.mods file).

Good job everyone!!

Art Isbell
(art@cubicsol.com)

MiscTabActionCell v1.3 (January 24, 1995) MiscKit 1.4.0

This version Misc-ifies the TabActionCell. It has been added to the MiscKit with the permission of the authors.

Enjoy-

-Don Yacktman
Don_Yacktman@byu.edu

UITabActionCell v1.3 (January 22, 1995)

This version touches up a few bugs which have been reported. The bugs are:

1. The spacing between the characters was funny because we were using the printing font instead of explicitly setting the screen font. Now, if there is a screen font we use it. Using other fonts than Helvetica should work properly now. Printing should no longer give an error.

2. The lowest line of the extenders is no longer cut off.

3. The selected cell has a white top border.

Also, it includes a nice icon drawn by Stefanie Herzer. Thanks to all!

Enjoy-

Sean Hill
shill@iphysiol.unil.ch

UITabActionCell v1.2

As I was about to release v1.1 to the net, a colleague here found yet another bug in my implementation. When the mouse was dragged across the matrix of cells, the highlighting was all wrong and the action wasn't being sent. So I delved into the code and fixed this. Because of the way NeXT implemented cells, it wasn't fun, but now it works just like the

tabs do in Interface Builder!

Have fun,

Bill Edney
bedney@firstsoft.com

UITabActionCell v1.1 (not released)

A small update to TabMatrixPalette to make it a little cleaner visually. I had originally done the TabMatrix using a selection cell, which gave it the behavior I wanted (namely, to both switch the cell and send its action on mouseDown instead of mouseUp). A number of folks (including Mark Onyschuk and Sean Hill) have worked on it since then and changed from using a subclass of SelectionCell to a subclass of ActionCell. This was cool in that things could be hooked up in IB, but bothered me because the cell didn't switch its look (nor send its action) on mouseDown, but mouseUp, which gave it a funny look and feel. The cell would push in on mouseDown and only look proper when the mouse went up. So, in the spirit of good hackware, I went back in and fixed this. This really makes the thing look and feel much nicer!

Have fun,

Bill Edney
bedney@firstsoft.com

UITabActionCell

Here is the latest version of the Public Domain UITabCell Palette.

I touched this up a little so that the dark gray view is no longer needed and the cell itself is a subclass of an ActionCell so that it can be used with actions and tags. I think it makes a more useable palette now. What I would like to see is the edges of the tab be drawn by postscript so that we could make really huge fat tabs. But it's probably best like it currently is.

I also implemented the calcSize method so that it knows the proper size of the cell.

Thanks to all who have created this before me!

Laboratoire de Neuro-Heuristique Work: ++41 021 692.5516
Institut de Physiologie Fax: ++41 021 692.5505
Rue du Bugnon, 7 Sean.Hill@iphysiol.unil.ch

CH-1005 Lausanne SWITZERLAND

Several weeks ago, a TabView much like the one featured in EOF's new Interface Builder was posted. Here's a palette built around the same code, compiled for both black and white hardware.

Only one change was made to accommodate IB. TabSelectionCell adds a test for the kind of superview it is being asked to draw in -- normally this is a Matrix but in the IB editor, the cell is asked to draw in some other kind of View.

You can now drag a Matrix of these Cells and, in conjunction with a palette like TTools from NeXTanswers, build a switching file-folder like UI completely within IB.

Many thanks to the original poster!

Regards,
Mark R. Onyschuk

M. Onyschuk and Associates Inc. 389 Leslie St. Toronto Canada, M4M 3E3
NEXTSTEP SOFTWARE DEVELOPMENT phone +1.416.462.3954

After seeing Jean-Marie's new addition to Interface Builder, I just couldn't help wanting those cool tabs for my very own! Since I don't have IB source :-), I just had to figure it out for myself.

There are two caveats (bugs?):

1) If you put these into a ScrollView, sometimes the little tab ends won't redraw themselves properly when scrolled after being clicked on. Since you probably shouldn't be putting a control like this into a ScrollView, this really shouldn't present a problem (although if someone want's to send me a bug fix, that'd be appreciated).

2) The first and last tab cells are 7 pixels (half a tab image width) shorter than the rest of the cells. This is due to a drawing issue with drawing inside the matrix. If the images on the first and last cell aren't shifted in, they get clipped by the matrix and don't look very good. If your cell's are relatively large, this really isn't very noticeable. The real way to fix this is to get in there, determine how many cells there are, and distribute this difference among all of them. Again, anyone wanting to send me this code should fell very free to do so.

Have fun,

Bill Edney
bedney@firstsoft.com

P.S. This code is totally free from any licenses or warranties or anything. In other words, use it however you want but don't blame me if it hoses something up!
