Custom Menu

CustomMenu.tiff ¬

by Gideon King
Gideon@CSARC.otago.ac.nz

Introduction

Someone suggested that it would be a good idea to have menus that could be dragged off to create custom menus for the menu items you find useful. I thought this would be a good idea, so I set about implementing it.

I have got most of the way with this project, but there are still a few bugs in it and I have run out of time to fix them. If you fix any bugs, please let me have the revised source code. When it's fully debugged, I'll submit it to MiscKit.

Building the CustomMenu palette

Nothing special here - you may want to change the paths for your palette etc in the Makefile.preamble though.

How to include Custom Menus in your project

· Drag a custom menu object into your application's main nib
· Connect the Quit menu item to the application's *Main menu*, and get it to send performClose: to the *Main menu* (the poseAs: sorts this out at runtime).
· Get the lightningCursor image from the CustomMenu palette, and put it in your project.
· Include the libCustomMenu library in your project

Please note:
Because of interaction issues between interface builder and the custom menu classes, I have disabled the use of custom menus during Interface Builder test mode. This will not stop you using IB test mode - it just means that you won't be able to test the custom menus in IB.

There is currently a bug in the save/restore methods which I would like someone to fix and send me the revised code, so I have allowed you the option of not trying to save/restore the menus by the use of the switch on the inspector.

User instructions

· To create a custom menu in your Custom Menu enabled application, just Alt-drag a menu item off a menu. You will be asked to confirm that you really do want to create a new menu.
· Once you have a custom menu, you can then tear it off and drag new items on to it from any other manus.
· You can have as many custom menus as you like, and as many items on any custom menu as you like.
· To remove an item from a custom menu, just Control-click the item you want to remove. When you delete the last item from a menu, the menu will be removed.

Known bugs

1. The main one at the moment is archiving and unarchiving the custom menu items. What I tried to do was to use a NXWriteRootObject() for each of my main menu menucells which had a custom menu attached to it, and thus wrote the cells and their submenus to my file. When I reload the menu cells from the file and put them in my main menu, they appear ok, and have their submenus, but there are a few odd things start happening. You sometimes have to click twice on the item to make the submenu appear/disappear. The submenu items sometimes don't display their hotkeys (although the items still work). The submenus appear a few pixels to the right of where they should. When I try to remove a custom menu item from the main menu, it gives me postscript errors and falls over. And other weird behaviour. The menuCells I am using are standard menuCells. The menus that are stored are a subclass of a subclass of a menu. The first subclass poses as the Manu class.

2. I don't know how to get rid of a submenu cleanly. When you control-click on the last menu item in a custom menu, I want to remove the supermenu's item controlling this menu (this works), and then free myself (the submenu). This doesn't work (even using perform:with:afterDelay:), and when I next go to use the main menu, it says I'm trying to talk to a freed object. As a work around, I just close the menu, and as you have no way of reopening it, it is effectively gone, and it is not archived and reloaded. NeXT must have a solution to this as they do it in edit when you close your
GDB windows.

3. Can someone please tell me how to get the lightningCursor to automatically be added to the application using the palette, rather than them having to find it in the palette?

4. Just a question or two - how would I go about getting Workspace to load something like this as a bundle? Would it be possible to make all applications have this feature automatically (I use poseAs to become the Menu class)? Even as a palette, you still need to link the library in to the application.

Please address any correspondence to gideon@csarc.otago.ac.nz

Class Documentation

Version 0.9 -- September 18, 1995

DraggableMenu

Inherits From:	Menu : Panel : Window : Responder : Object
Adopted Protocols:	NXDraggingSource, NXDraggingInfo
Declared In:	CustomMenu.subproj/DraggableMenu.h

Class Description

DraggableMenu makes menu cells on the menu able to be dragged. It also includes a few things required by the CustomMenu class (I guess these extra bits could be put into a category if you felt like it).

It is designed to be used as part of the CustomMenu palette, and if you intend to use it separately, you will have to change some or all of:

- setSaveAndRestore:(BOOL)yesNo;
- (BOOL)getSaveAndRestore;
- awakeFromNib;
- performClose:sender;
- write:(NXTypedStream *)stream;
- read:(NXTypedStream *)stream;

Instance Variables

NONE And don't declare any or the poseAs: will break!

Global Variables

Unfortunately these are necessary as I can't add instance variables, and I need state information.

const char *MHMenuCellPboardType
My private pasteboard for menu cells.

BOOL __saveAndRestoreFlag__
Whether to save and restore menus.

BOOL __haveUnarchivedMenu__
State variable needed for save/restore.

Method Types

Version (RCS/CVS)	- version
Configuration	- setSaveAndRestore: - getSaveAndRestore
Archiving	- awakeFromNib - write: - read: - performClose:
Dragging	- mouseDown:

	- draggedImage:beganAt: - draggingSourceOperationMaskForLocal: - draggedImage:endedAt:deposited: - ignoreModifierKeysWhileDragging - isDraggingSourceLocal
Internal support	- attachSubmenuUsing:

Instance Methods

attachSubmenuUsing:

± **attachSubmenuUsing:aList**

Used internally as part of draggedImage:endedAt:deposited:.

awakeFromNib

± **awakeFromNib**

Loads the stored custom menus from the custom menu file. Returns **self**.

draggedImage:beganAt:

± **draggedImage:(NXImage *)image beganAt:(NXPoint *)screenPoint**

Returns **self**. Used as part of the dragging source protocol.

draggedImage:endedAt:deposited:
± draggedImage:(NXImage *)image endedAt:(NXPoint *)screenPoint deposited:(BOOL)flag

Creates a new custom menu if the menu cell has been dragged out into thin air (i.e. not onto a custom menu).

draggingSourceOperationMaskForLocal:
± (NXDragOperation)draggingSourceOperationMaskForLocal:(BOOL)flag

Used as part of the dragging source protocol.

getSaveAndRestore
± (BOOL)getSaveAndRestore

Tells you whether you are saving and restoring the menus.

ignoreModifierKeysWhileDragging
± (BOOL)ignoreModifierKeysWhileDragging

Returns **YES**.

isDraggingSourceLocal

± **(BOOL)isDraggingSourceLocal**

Returns **YES**.

mouseDown:

± **mouseDown:(NXEvent *)theEvent**

Starts the dragging of a menu cell if the Alternate key is down. Returns **self**.

performClose:

± **performClose:sender**

Overrides the Menu method, so we have a method to connect to before the poseAs: is executed. If the sender's title is not "Quit", will get the Menu class to deal with it, otherwise terminates your application after saving your custom menus if required.

read:

± **read:(NXTypedStream *)stream**

Overrides the standard read method to read the __saveAndRestoreFlag__, and if not running in IB test mode, poses as the Menu class.

setSaveAndRestore:

± **setSaveAndRestore:(BOOL)yesNo**

Turns the saving and restoring of menus on or off. Returns **self**.

version

± **(const char *)version**

Returns the RCS/CVS ident (put in there by our CVS Manager application). Avoids compiler warnings we would otherwise get.

write:

± **write:(NXTypedStream *)stream**

Overrides the standard write method to save the __saveAndRestoreFlag__.

CustomMenu

Inherits From: DraggableMenu : Menu : Panel : Window : Responder : Object
Adopted Protocols: NXDraggingDestination
Declared In: CustomMenu.subproj/CustomMenu.h

Class Description

CustomMenu is the destination for a DraggableMenu, and allows you to drag the menu cells into the custom menu. It also allows you to delete items from the menu.

Instance Variables

NONE

Method Types

Version (RCS/CVS)	- version
Creating new instances	- myInit - init - initTitle:
Deleting menu items	- mouseDown:
Dragging destination	- draggingEntered: - prepareForDragOperation: - performDragOperation:
Internal support	- removeCell:

Instance Methods

draggingEntered:

± (NXDragOperation)draggingEntered:sender

Returns **NX_DragOperationCopy**.

init

± **init**

Overrides Menu's init so I can register as the drag destination using myInit. Returns **self**.

initTitle:

± **initTitle:(const char *)aTitle**

Overrides Menu's initTitle: so I can register as the drag destination using myInit. Returns **self**.

mouseDown:

± **mouseDown:(NXEvent *)theEvent**

Allows removal of menu cells when you have the Control key down.

myInit

± **myInit**

Registers as the dragging destination for Draggable Menus. Called from init and initTitle:. Returns **self**.

performDragOperation:

± (BOOL)performDragOperation:sender

Adds the dragged cell to the Custom Menu. Returns **YES**.

prepareForDragOperation:

± (BOOL)prepareForDragOperation:sender

Returns **YES**.

removeCell:

± removeCell:aCell

Used internally by mouseDown: to remove a menu cell from the menu. Returns **self**.

version

± (const char *)version

Returns the RCS/CVS ident (put in there by our CVS Manager application). Avoids compiler warnings we would otherwise get.