init
initTitle:
Setting up the Menu commands addItem:action:keyEquivalent:
setItemList:
itemList
Finding Menu items findCellWithTag:
Building submenus setSubmenu:forItem:
submenuAction:

rightMouseDown:

Archiving read:

write:
awake

alloc (Object), + setMenuZone:

alloc (Object), + menuZone:

addItem:(const char *)aString
action:(SEL)aSelector
keyEquivalent:(unsigned short)charCode

Adds a new command named aString to the bottom of the receiving Menu and returns the MenuCe
MenuCell's action method is set to aSelector, but its target is nil. charCode is set as the MenuCell'
command name and key equivalent aren't checked for duplications within the same Menu (or any
assign them uniquely. The new MenuCell is enabled, but has no tag or alternate title your code m
never set a MenuCell's icon.

This method doesn't automatically redisplay the Menu. Upon the next display message, the Menu
to fit and displayed.

setSubmenu:forItem:

awake

Checks whether an unarchived Menu should have a close Button. Your code shouldn't invoke this
by the read: method. Returns self.

read:

close

Overrides Window's close method. Ensures that attached submenus are closed along with the rece

close (Window)

findCellWithTag:(int)aTag

Returns the MenuCell that has aTag as its tag, or nil if no such Cell can be found. If your applicati
tages, each MenuCell should have a unique tag.

findCellWithTag: (Matrix), setTag: (ActionCell)

getLocation:(NXPoint *)theLocation forSubmenu:aSubmenu

Returns the location in screen coordinates at which the lower-left corner of the receiving Menu's s
drawn. Menu invokes this method whenever it brings up a submenu. By default, the submenu is t
supermenu, with its title bar aligned with the supermenu's. Your code need never directly use this
override it to cause the submenu to be attached at a different location.

submenuAction:

init

Initializes and returns the receiver, a new instance of Menu, displaying the title ªMenuº. All other
described in the initTitle: method below.

initTitle:

initTitle:(const char *)aTitle

Initializes and returns the receiver, a new instance of Menu, displaying the title aTitle. The Menu
upper left corner of the screen, and has no command items. A new Menu must receive an orderFro
displayed on the screen the Application object takes care of this for standard Menus.

The Menu is created as a buffered window, of style NX_MENUSTYLE and button mask NX_CLO
a Menu hides its close button until it's torn off from its supermenu). All Menus have an event mas
keyboard events, so they never become the key window or main window.

addItem:action:keyEquivalent:, init

itemList

Returns the Matrix of MenuCells used by the Menu, which your code can use to add or rearrange c
directly. Be sure to send sizeToFit after altering the Matrix, as the Menu won't know that the Mat

setItemList:, sizeToFit

mouseDown:(NXEvent *)theEvent

Overrides the Responder method to catch a mouse-down event instead of passing it along, so that t

method is overriden fromWindow's to resize the Matrix if needed before redisplaying the Menu at

moveTo:: (Window)

read:(NXTypedStream *)stream

Reads the Menu from the typed stream stream. Returns self.

awake, write:

rightMouseDown:(NXEvent *)theEvent

Pops the menu up under the cursor position in theEvent. Before doing so, this method saves the cu
(including selected cells, attached submenus, menu positions, and so on). The menu is popped up
or submenus attached. The Menu is tracked as for a mouseDown: event. On mouse-up, the Menu'
that the original Menu arrangment on screen isn't changed. Returns self.

mouseDown:

setAutoupdate:(BOOL)flag

If flag is YES, the Menu will invoke the update action for each MenuCell whenever it receives an
messageÐusually sent by the Application object when autoupdating of windows is enabled. If NO
update its MenuCells on receiving an update message.

update, setAutoupdate: (Application), setUpdateAction:forMenu: (MenuCell)

setItemList:aMatrix

Sets the Menu's Matrix of items to aMatrix. A following display message will size the Menu to fi
drawing. Returns the old Matrix.

itemList, display

setSubmenu:aMenu forItem:aCell

Sets aMenu as the submenu of the receiver, controlled by the MenuCell aCell. aCell's target is set
submenuAction: and its icon to the arrow indicating that it brings up a submenu. Doesn't remove
If aMenu was on screen, it won't be removed from the screen or moved until it's first brought up a
aCell.

submenuAction:

sizeToFit

Action method sent to a submenu associated with an entry in a Menu. If sender's Window is a visi
attaches and displays itself as a submenu of the sender's Menu otherwise, does nothing. sender sh
containing the MenuCell that brings up the submenu. Returns self.

setSubmenu:forItem:

update

Updates the Menu's items. If the Menu has been set to autoupdate, this method gets the update act
its MenuCells and sends that method to the first of the following that responds to it: the Menu's de
NXApp's delegate. If a MenuCell's update action returns YES, that MenuCell is redrawn.

setAutoupdate: setUpdateAction:forMenu: (MenuCell)

windowMoved:(NXEvent *)theEvent

Overrides the Window method to implement tear-off Menu behavior. When a submenu is torn off
supermenu is unhighlighted. The submenu is flagged as detached, is moved to the appropriate win
its close Button. Returns self.

windowMoved: (Window)

write:(NXTypedStream *)stream

Writes the receiving Menu to the typed stream stream. Returns self.

read: