init

Setting up the items addItem:
insertItem:at:
removeItem:
removeItemAt:
indexOfItem:
count

Interacting with the trigger Button
changeButtonTitle:
getButtonFrame:

Activating the PopUpList popUp:

Getting the user's selection selectedItem

(SEL)action

Returns the action sent to the PopUpList's target when an item is selected from the list. This is act
message of the PopUpList's Matrix.

setAction:, action (Matrix), target

addItem:(const char *)title

Adds an item with the name title to the bottom of the PopUpList, and returns the MenuCell created
equivalent can be added, for example). If an item with the name title already exists in the PopUpL
that item is returned.

insertItem:at:, removeItem:

changeButtonTitle:(BOOL)flag

If flag is YES, then when a selection is made from the list, the title of the selected item becomes th
trigger that sent the popUp: message (nearly always a Button, but sometimes a Matrix). This make
the user as a pop-up list, with a small rectangular knob as the icon. If flag is NO, then the Button's
so that it appears to the user as a pull-down list, with a small inverted triangular mark for an icon.
(that is, a pop-up list). Returns self.

(unsigned int)count

Returns the number of items in the PopUpList. If the PopUpList is configured as a pull-down list,
the MenuCell that holds the pull-down list's title.

font

Returns the Font used to draw the items in the PopUpList.

setFont:

getButtonFrame:(NXRect *)bFrame

Returns self, and by reference in bFrame the frame for the trigger that last popped up the PopUpLi
frame is set to (0.0, 0.0), so this method effectively returns the size of the trigger.

(int)indexOfItem:(const char *)title

this method.

insertItem:(const char *)title at:(unsigned int)index

Inserts an item with the name title at position index in the PopUpList. The item with an index of 0
Returns the newly inserted MenuCell.

If an item with a title of title already exists in the PopUpList, it's removed and the new one is adde
moves title to a new position, though if the item removed was at a position before index, the new it
inserted at index + 1. If you want to move an item, it's better to invoke removeItem: explicitly and
at:.

addItem:, removeItemAt:

popUp:trigger

Pops up the PopUpList over the location of trigger, after resizing the PopUpList to be as wide as tr
action method sent by the trigger object to the target PopUpList. If the mouse goes up in an item o
Matrix that displays the PopUpList's entries sends the action message to the target. If the PopUpL
(set with changeButtonTitle:), trigger's title is set to the title of the selected item in the PopUpList
altered by this method. Returns self.

This method works if and only if the following conditions are met. The Application object's curre
mouse-down, and that mouse-down must have occurred within trigger's frame this method can the
invoked only as a result of a mouse-down occurring in trigger. trigger must also be either a subcla
responds to the messages title and setTitle:, or a subclass of Matrix whose selected Cell responds t
there are no items in the PopUpList and trigger's title is NULL, this method does nothing.

If trigger's title isn't in the PopUpList, it's added as an item at the top before the PopUpList pops u
with the item having the same title as trigger (either a pop-up list's selected item or the ªtitleº item
exactly over trigger if possible if this would cause part of the list to be off the top or bottom of the
shifted up or down so that it can fit on screen.

If any of the MenuCells in the PopUpList's Matrix bring up submenus (that is, have a Menu as a ta
submenuAction: as the action message), they are changed to simply be title-displaying MenuCells,
up their submenus. Essentially, this means that you can't create a hierarchical PopUpList with this
completely override this method.

setAction:, setTarget:, changeButtonTitle:

removeItem:(const char *)title

Removes and returns the MenuCell with the name title. If there is no such MenuCell, returns nil.

removeItemAt:

removeItemAt:(unsigned int)index

Removes the MenuCell for the item at the specified index. Returns the MenuCell at that location,
such MenuCell.

The target of the PopUpList can get the title of the selected item in one of two ways. Since the sen
message is actually the PopUpList's Matrix, the target can ask the Matrix for its selected Cell, and
its title. Also, the PopUpList is the Matrix's Window, so the target can retrieve that and then send
PopUpList. These two methods can be coded as follows:

selectedCell: (Matrix), title (Cell)

setAction:(SEL)aSelector

Sets the action sent to the PopUpList's target when an item is selected. The action message is actu
containing the MenuCells that make up the PopUpList. Returns self.

A pull-down list does send its action if the mouse goes up in its title item.

action, setAction: (Matrix), setTarget:

setFont:fontObject

Sets the Font used to draw the PopUpList's items. The PopUpList does redraw itself, but since it r
the screen when it receives this message, this shouldn't cause any undesirable side-effects. Return

font (Matrix)

setTarget:anObject

Sets the object to which an action will be sent when an item is selected from the PopUpList. The a
by the Matrix containing the MenuCells that make up the PopUpList. Returns self.

target, setTarget: (Matrix), setAction:

sizeWindow:(NXCoord)width :(NXCoord)height

Your code should never invoke this method, though you're free to override it. This method is over
because PopUpList needs to surround itself with a dark gray border, and thus needs to be one pixel
Menu. It simply adds 1.0 to each dimension and sends sizeWindow:: to super. Returns self.

target

Returns the object to which the action will be sent when an item is selected from the list. The defa
causes the action message to be sent up the responder chain. The target is actually sent the action b
Matrix.

setTarget:, target (Matrix), action