FileQueue

INHERITS FROM	Object
WRITTEN BY	Carl F. Sutter

Version 1.0, Fri Jun 25 1993, This class is in the Public Domain. No guaranties are made to its usefulness or correctness.

CLASS DESCRIPTION

The FileQueue Class is a simple Interface Builder Module to implement a queue of strings, like a list of files. A panel can be displayed that will show the files in the queue, and allow the user to remove or adjust the order of items. The calling object has the option of requesting items immediately, or via a target-action technique to receive files when the FileQueue instance is ready.

FEATURES

• The items can be returned via an immediate function to get the next item, or by notifying the queue to send the next available item - this target-action like technique allows the pause button to be used effectively.

• The control panel is configured in the IB, including the matrix used to hold the items. This allows the font, spacing, and positioning to be set quite easily.

• The queue was designed to hold a list of files pending processing, but can hold any list of strings.

INSTANCE VARIABLES

Declared in FileQueue	id	queuePanel
	id	queueScroll
	id	queueMatrix
	BOOL	bPaused
	BOOL	bSendWhenReady
	int	nNumRows
	int	nSelectedRows
	int	nNumSelected
	id	target
	SEL	action

queuePanel	outlet for panel
queueScroll	outlet for custom ScrollView
queueMatrix	outlet for TextFieldCell Matrix
bPaused	flag for paused status
bSendWhenReady	flag to send next available item
nNumRows	number of items in the queue
nSelectedRows	array of selected item indexes
nNumSelected	number of selected (highlighted) items
target	object to send next item to
action	method to receive next item

METHOD TYPES

Creating a FileQueue object	+ new + new:action:
Outlet Initialization Methods	- setQueuePanel: - setQueueScroll: - setQueueMatrix:
Adding and Retrieving Items	- addItem: - retrieveNext - retrieveNextWhenReady
Setting Target and Action	- setTarget: - setAction:
Actions from the Controller	- show:
Actions from the Panel	- pause: - remove: - toTop: - toBottom:
Internal Methods	- setup - countOneSelected: - sendNext - windowWillResize:toSize:

CLASS METHODS

new

+ new

Creates a new FileQueue object.

new:action:

+ **new:**(id)targ
action:(SEL)act
Creates a new FileQueue object. The target object and selector to call when ready to send another item are specified.

INSTANCE METHODS

addItem:

– (BOOL)**addItem:**(const char *)szFileName

Adds the given string to the bottom of the queue. Return value indicates success.

countOneSelected:

– (BOOL)**countOneSelected:**(id)cell

Internal routine that is called to enumerate the list of selected cells.

pause:

– **pause:**(id)sender

Pauses the queue. This only works if the next queue item is requested by the retrieveNextWhenReady method. This action is called from the panel, and is connected in the IB.

remove:

– **remove:**(id)sender

Removes the highlighted items from the queue. This action is called from the panel, and is connected in the IB.

retrieveNext

– (char *)**retrieveNext**

Immediately returns the next item on the queue. This circumvents the pause button. The normal way is to use retrieveNextWhenReady, and let the queue call your object when an item becomes available.

retrieveNextWhenReady

– (char *)**retrieveNext**

Signals the queue to send the next available string from the queue to the target and action specified with other methods. The item may not be sent right away if there are no items on the queue at the time or if the pause button is pressed.

sendNext

– sendNext

Internal routine that send the next selected cell to the target and action specified in other methods.

setAction:

– setAction:(SEL)aSelector

Sets the method that gets called when a new queue item is ready and requested using the retrieveNextWhenReady method. The selector must have one character string as it's argument, for example: nextFromQueue:(char *)szFileName.

setTarget:

– setTarget:(id)targ

Sets the target object for receiving the next item when ready. The target-action system used use is not in the conventional form. The action selector must have one argument of type (char *), which will contain the string from the queue. This target-action technique of using the queue is best, since the pause button is used. The alternative is to use the retrieveNext method to immediately get the top queue item.

setQueueMatrix:

– setQueueMatrix:(id)anObject

Sets the queueMatrix outlet from the IB connection. The queueMatrix is a matrix of TextFieldCells, which can be created in the IB by <Alternate>-dragging a corner of a TextField item. Using the IB to make a prototype matrix of TextFieldCells lets the font style, spacing etc. be set outside of the program. The width and position are automatically adjusted to fit the queueScroll, and all cells drawn in the IB are deleted. This method is called automatically when the nib file is loaded.

setQueuePanel:

– setQueuePanel:(id)anObject

Sets the queuePanel outlet from the IB connection. This method is called automatically when the nib file is loaded.

setQueueScroll:

– **setQueueScroll:**(id)anObject

Sets the queueScroll outlet from the IB connection. The queueScroll is an IB Custom View that has the ScrollView class set as its "attribute". This technique of defining it in the IB lets the size be set with the mouse quite easily. This method is called automatically when the nib file is loaded.

setup

– **setup**

Internal routine to initialize various aspects of the queue, including loading in the nib file.

show:

– **show:**(id)sender

Displays the queue panel. Usually called from the controlling object in response to a menu selection to display the queue.

toBottom:

– **toBottom:**(id)sender

Moves the highlighted items to the bottom of the queue (lowest priority). This action is called from the panel, and is connected in the IB.

toTop:

– **toTop:**(id)sender

Moves the highlighted items to the top of the queue (highest priority). This action is called from the panel, and is connected in the IB.

windowWillResize:toSize:

– **windowWillResize:**(id)sender **toSize:**(NXSize *)frameSize

Handles the delegate message sent from the panel when the user is resizing the window. The size is kept from getting too small, so that the buttons don't get hidden.

CONSTANTS AND DEFINED TYPES

```
/* Maximum number of items selectable on the list */
#define  MAXSELECTABLE           1024
```