MiscDragView

Inherits From:	View : Responder : Object
Declared In:	misckit/MiscDragView.h

Class Description

MiscDragView is an abstract class that implements the methods needed for both source and destination dragging. It supports both a delegate and the target/action paradigm. When a drag view is the successful recipient of a drag session, the target is sent the action message (if there is a target and action). Alternately, the delegate (if it responds) will be sent messages such as **didInitiateSourceDrag:**, **didFinishSourceDrag:**, **didInitiateDestinationDrag:** and **didFinishDestinationDrag:**, which should be pretty self explanatory. One word of caution though. Since most of the delegate messages are sent from methods that are part of the NXDragging protocols, and are usually overridden in subclasses, make sure you call the superclass's implementation too. The documentation usually warns you when this could be a problem.

The minimum needed to create a subclass that allows simple drag and drop (source and destination) is as follows:

±Override **initDragTypes** and register the pasteboard types you will accept (needed for destination dragging only).

±Override **setupSourceForDragging** in order to set the dragImage and put the relevant data on the pasteboard (needed for source dragging only).

±Override **performDragOperation:** to take the data from the pasteboard, when a successful dragging session has occurred (needed for destination dragging only).

Unless you would like the most simple of drag views, you will find it necessary to override other methods, mostly the ones that are in the two NXDragging protocols. For more information refer to /NextLibrary/Documentation/NextDev/GeneralRef/02_ApplicationKit/Protocols/NXDraggingDestination.rtf and NXDraggingSource.rtf.

Instance Variables

id **theImage;**
NXSize **imageSize;**
int **border;**
BOOL **allowSourceDragging;**
BOOL **allowDestinationDragging;**
BOOL **freeWhenDragDone;**
id **delegate**;
id **target**;
SEL **action**;

theImage	Image currently displayed in the view.
imageSize	The original size of theImage just in case we need it for scaling.
border	Defines the border type that will surround the view.
allowSourceDragging	Determines if we are allowed to be the source of a drag.
allowDestinationDragging	Determines if we are allowed to be the destination of a drag.
freeWhenDragDone	Determine if we are responsible for freeing the dragImage after a drag session.
delegate	The object that receives notification messages.
target	The object that will receive the action message when view is dragged into.

action	The message sent to the target when the view is dragged into.

Method Types

Initializing a MiscDragView	+ initialize ± initFrame: ± initDragTypes ± free
Setting view and drag images	± image ± setImage: ± setImageByFilename: ± dragImage ± setDragImage: ± setDragImage:freeWhenDone:
Dragging options	± setAllowSourceDragging: ± allowSourceDragging ± setAllowDestinationDragging: ± allowDestinationDragging ± acceptForeignDrag ± acceptLocalDrag ± acceptSelfDrag ± retainData ± shadowIncoming ± shadowColor
Source dragging methods	± mouseDown: ± cleanupAfterSourceDrag ± setupForSourceDrag

	± slideback
	± draggingPasteboard
	± calculateDragPoint::
	± draggingSourceOperationMaskForLocal:
	± draggedImage: endedAt: deposited:
Destination dragging methods	± draggingEntered:
	± draggingUpdated:
	± draggingExited:
	± prepareForDragOperation:
	± performDragOperation:
	± concludeDragOperation:
	± cleanupAfterDestinationDrag
Delegate methods	± delegate
	± setDelegate:
	± sourceDragInitiated:
	± sourceDragFinished:
	± destinationDragInitiated:
	± destinationDragFinished
Target/Action methods	± target
	± setTarget:
	± action:
	± setAction:
	± stringValue
	± setStringValue:
	± takeStringValueFrom:
Displaying the border	± borderType
	± setBorderType:
Archiving	± read:

± write:
± awake

Class Methods

initialize
+ **initialize**

Sets the archive version of this class. Returns **self**.

Instance Methods

acceptForeignDrag
- (BOOL)**acceptForeignDrag**

If the view will accept a drag that did not initiate from within the same application, **YES** is returned (this is the default). Override to return **NO** in your subclass if this is not the behavior you would like.

See also: - **acceptLocalDrag**

acceptLocalDrag
-(BOOL)**acceptLocalDrag**

Returns YES if the view will accept a drag that initiated from within the same application. This is the default. Override in a subclass if you would like different behavior.

See also: - **acceptForeignDrag**

acceptSelfDrag
    - (BOOL)**acceptLocalDrag**

By default this method returns YES, which means the view will accept a drag that initiated from itself.   Override this method in a subclass and return NO if you do not want the default behavior.

action
    -(SEL)**action**

Returns the action that will be sent to the target when the receiver is the destination to a successful drag.

See also: - **setAction:**

allowDestinationDragging
    - (BOOL)**allowDestinationDragging**

Returns YES if the view is allowed to be the destination of a dragging session.

See also: - **setAllowDestinationDragging:**

allowSourceDragging
    - (BOOL)**allowSourceDragging**

Returns YES if the view is allowed to initiate a dragging session.

See also: - **setAllowSourceDragging:**

awake
    - **awake**

Calls **initDragTypes** to register dragging types. Returns **self**.

borderType
- (int)**borderType**

Returns the type of border that will be displayed around the view. It will be one of NX_BEZEL, NX_GROOVE, NX_LINE, or NX_NOBORDER. These are the same styles as the Box class.

See also: - **setBorderType:**

calculateDragPoint: andOffset:
- **calculateDragPoint:**(NXPoint *)*dragPoint* **andOffset:** (NXPoint *)*offset*

This method does nothing in MiscDragView, but is there for subclasses that would like more control of where the drag image first appears in relation to the mouse. The *dragPoint* is the NXImage's location given in the view's coordinate system. The *offset* is the current mouse location relative to the mouse down location that started the drag. So far I have just found that adjusting the *dragPoint* can be useful to center the dragged image relative to the cursor. Returns **self.**

cleanupAfterDestinationDrag
- **cleanupAfterDestinationDrag**

Frees the copy we made of the dragged image (if we are the destination view). If you purposely end a drag by returning NO from either **prepareForDragOperation**: or **performDragOperation**:, make sure you call this method to clean up. If your subclass creates an extra resources because of a destination drag, this is probably the place to free them. Returns **self.**

cleanupAfterSourceDrag
- **cleanupAfterSourceDrag**

This method is called after a source drag has completed. It frees the image in the view if the data is non-retained, and also frees the dragImage if it was instructed to. Returns **self.**

concludeDragOperation:
- **concludeDragOperation:** *sender*

This method is part of the NXDraggingSource protocol. If you subclass this method, make sure to call [super **concludeDragOperation:** sender] because this method sends the **didFinishSourceDrag:** message to the delegate as well as sending it's action message to the target (if there is one). Returns **self**.

See also: **-prepareForDragOperation:, -performDragOperation:**

delegate
- **delegate**

Returns the current delegate, or **nil** if no delegate is currently set.

See also: - **setDelegate**

draggingPasteboad
- **draggingPasteboard**

By default this method returns NXDragPboard. Override if your subclass uses a different pasteboard.

dragImage
- (NXImage *)**dragImage**

Returns the current image used for dragging. You should not free the returned image.

See also: - **setDragImage:**

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

This method is part of the NXDraggingSource protocol. If you override this method in a subclass, make sure to call this class's method so the **didFinishSourceDrag:** message is sent to the delegate (if it reponds to it). Returns **self**.

draggingEntered:
-(NXDragOperation)**draggingEntered:** *sender*

This method is part of the NXDraggingDestination protocol. Since this method contains most all the testing for types of destination dragging that are allowed, if you plan to override it, do so in the following way:

```
(NXDragOperation)draggingEntered: sender
{
    ( your code here )
    return [super draggingEntered: sender];
 }
```

See also: - **draggingUpdated:**, - **draggingExited:**

draggingExited:
-**draggingExited:** *sender*

This method is part of the NXDraggingDestination protocol. This method passes the message **didFinishDestinationDrag:** NO to the delegate, since the dragging image exited the view, and the dragging did end (as far as the destination view is concerned). If you override this method later on, make to include a call to this class's method so the delegate method will be sent. Returns **self**.

See also: - **draggingUpdated:**, - **draggingEntered:**

draggingSourceOperationMaskForLocal:
- (NXDragOperation)**draggingSourceOperationMaskForLocal:** (BOOL)*flag*

This method is part of the NXDraggingSource protocol. The default value returned by this view is **NXDragOperationAll**.

draggingUpdated:
-(NXDragOperation)**draggingUpdated:** *sender*

This method is part of the NXDraggingDestination protocol. By default it returns the same value that **draggingEntered:** first returned when the view was entered by the dragged image.

See also: - **draggingExited:**, - **draggingEntered:**

free
- **free**

Frees the image if needed.

image
-(NXImage *)**image**

Returns the image that is currently displayed in the view. If there is no image displayed, **nil** is returned.

See also: - **setImage:**

initDragTypes
-**initDragTypes**

The default implementation does nothing, but subclasses are expected to override it. You should register the types of data you will accept in a dragging session. Returns **self**. Below is the initDragTypes method from MiscIconWell:

```
- initDragTypes
{
   const char *const types[] = {NXFilenamePboardType};

   [self registerForDraggedTypes: (const char *const *)&types count: 1];
   return self;
}
```

initFrame:

-initFrame: (const NXRect *)*frameRect*

This class's designated initializer. Sets the receiver's defaults and calls **initDragTypes**. Returns **self**.

mouseDown:

- mouseDown: (NXEvent *)*theEvent*

Overridden in order to begin a drag session. First, if source dragging is allowed and the mouse is moved far enough to begin a drag, **setupForSourceDrag** is called to put the data on the pasteboard and choose an image for dragging. If **setupForSourceDrag** returns **YES**, **calculateDragPoint: andOffset:** is called to allow each subclass some finer control on where the image appears when a drag begins. As long as the drag image is not **nil, didInitiateSourceDrag:** is sent to the delegate, and the source dragging begins. Be careful when subclassing this method (I suppose you do have the source code though).

performDragOperation:

- (BOOL)**performDragOperation:** *sender*

This method is part of the NXDraggingDestination protocol. The default (for this class) is to return YES. If you override this method and are going to return NO for some reason (to stop the drag) make sure to call **cleanupAfterDestinationDrag**.

See also: - **prepareForDragOperation:**, - **concludeDragOperation:**

prepareForDragOperation:

- (BOOL)**prepareForDragOperation:** *sender*

This method is part of the NXDraggingDestination protocol. The default (for this class) is to return YES. If you override this method and are going to return NO for some reason (to stop the drag) make sure to call **cleanupAfterDestinationDrag**.

See also: - **performDragOperation:**, - **concludeDragOperation**

read:

- **read:** (NXTypedStream *)*stream*

Unarchives a MiscDragView object. Returns **self**.

See also: - **write:**

retainData

- (BOOL)**retainData**

Returns **YES** if the view keeps the data after a source drag has been completed.

setAction:

- **setAction:** (SEL)*anAction*

Sets the action message that is sent to target when an image is dragged successfully into our drag view. Returns **self.**

See also: - **action**

setAllowDestinationDragging:

- **setAllowDestinationDragging:** (BOOL)*aBool*

Sets whether the view is allowed to be the destination of a dragging session. If set to **NO**, the view will not accept any dragged images. Returns **self.**

See also: - **allowDestinationDragging**

setAllowSourceDragging:

- **setAllowSourceDragging:** (BOOL)*aBool*

Sets whether the view is allowed to initiate a dragging session. Returns **self.**

See also: - **allowSourceDragging**

setBorderType:

- **setBorderType:** (int)*borderType*

Sets the border type that surrounds the view. The types of borders are the same as the types for the Box class, which can be one of NX_BEZEL, NX_GROOVE, NX_LINE, or NX_NOBORDER. Returns **self**.

See also: - **borderType**

setDelegate:

- **setDelegate:** *theDelegate*

Sets *theDelegate* to be the object to receive the dragging notification messages. The messages that are sent are pretty self explanatory. They are one of **didInitiateSourceDrag:**, **didFinishSourceDrag:**, **didInitiateDestinationDrag:** and **didFinishDestinationDrag:**. Returns **self**.

See also: - **delegate**

setDragImage:
  - **setDragImage:** (NXImage *)*anImage*

Sets *anImage* to be the image used for a source drag. You are responsible for freeing the dragImage (unless it is the same as the image displayed in the view) when you no longer need it. This mehod calls
[self setDragImage:freeWhenDone: **NO**].

See also: - **dragImage,** **± setDragImage:freeWhenDone:**

setDragImage:
  - **setDragImage:** (NXImage *)*anImage* **freeWhenDone:** (BOOL)*freeIt*

Sets *anImage* to be the image used for a source drag. If *freeIt* is **YES**, then the view will free the dragImage when the drag session has been completed, otherwise you are responsible for freeing it (unless it is the same as the image displayed in the view) when you no longer need it.

See also: - **dragImage,** **± setDragImage:**

setImage
  - **setImage:** (NXImage *)*anImage*

Sets *anImage* to be the image displayed in the view, while freeing the previous image. *anImage* can be **nil** if you would like the view to be blank. Returns **self**.

See also: - **image**

setImageByFilename:
  - **setImageByFilename:** (const char *)*aFilename*

Loads the image specified by *aFilename* and passes the result to **setImage:**. If *aFilename* is NULL, any image in the view will be cleared. Return **self**.

See also: - **setImage:**, **± image**

setTarget:
- **setTarget:** *aTarget*

Sets the object that will be the target of the action message when our view has successfully been dragged into. Returns **self**.

See also: - **target**

setupForSourceDrag
- (BOOL)**setupForSourceDrag**

This method does nothing by default, and is meant to be overridden in subclasses. The overridden method should take care of putting the data on the pasteboard and choosing the image to be dragged. You should return **YES** if the data was put on the pasteboard successfully. If you return a **NO**, the drag session will not begin. Returns **self**.

See also: - **mouseDown:**

setStringValue:
- **setStringValue:** (const char *)*aValue*

By default this method does nothing. It is up to subclasses to override to set something meaningful with this method. It is also necessary to override this method if you wish to use **takeStringValueFrom:** Returns **self**.

See also: - **stringValue**

shadowColor
- (NXColor)**shadowColor**

Returns the color that will be used to shadow the image if [self shadowIncoming] returns YES. To "dim" the image, make sure to use alpha. By default we return NX_COLORLTGRAY with 0.5 alpha.

See also: - **shadowIncoming**

shadowIncoming
- (BOOL)**shadowIncoming**

Returns YES if a "dimmed" image should appear when we are the destination drag view. This is the default. If your subclass would doesn't want the shadowed images then subclass to return NO.

See also: - **shadowColor**

slideback
- (BOOL)**slideback**

Used to determine if an image that was unsuccessfully dragged will slideback to us, if we are the destination view. By default it returns [self retainData] because if we are not retaining data we do not want a slideback, otherwise why not.

stringValue
- (const char *)**stringValue**

The default is to return NULL. This method is here for subclasses to override and return something meaningful if they wish.

See also: - **setStringValue:**

takeStringValueFrom:
- **takeStringValueFrom:** *sender*

This method will only work correctly if your subclass has overridden the method **setStringValue:** to do something meaningful. Returns **self** if sender responds to stringValue, otherwise **nil** is returned.

See also: - **stringValue,** **± setStringValue:**

target

- target

Returns the object that will receive the action message when an image has successfully been dragged into our view.

See also: **-setTarget:**

write:

-write: (NXTypedStream *)*stream*

Write a MiscDragView object and it's current state to an archive.

See also: **-read**