MiscDragCell

Inherits From:	NSActionCell
Declared In:	MiscAppKit/MiscDragCell.h

Class Description

MiscDragCell is an abstract class that lays the groundwork for draggable cells to be used within some kind of drag view (either a MiscDragView or MiscDragMatrix subclass). For an example of a concrete subclass see MiscFileDragCell, which implements dragging using the NSFilenamePboardType.

Instance Variables

@private NSImage* _**acceptingImage**;
NSCell* _**imageCell**;
NSImage* _**image**;
NSCell* _**textCell**;
NSString* _**title**;
NSBorderType _**borderType**;
NSColor* _**backgroundColor**;
BOOL _**allowSourceDragging**;
BOOL _**allowDestinationDragging**;
BOOL _**shadowIncoming**;
BOOL _**displaysTitle**;
BOOL _**titleEditable**;
BOOL _**titleMultiline**;

_acceptingImage	Image to display when accepting a drag.
_imageCell	Cell that'll draw our image.

_image	Image we'll display (usually).
_textCell	Cell that'll draw our title (if enabled)
_title	Title we'll display (if enabled)
_borderType	Border (same as NSBox)
_backgroundColor	Background color (if drawn)
_allowSourceDragging	Allow us to start a drag?
_allowDestinationDragging	Allow us to receive a drag?
_shadowIncoming	Dim the incoming source image?
_displaysTitle	Do we display a title?
_titleEditable	Can our title be edited?
_titleMultiline	Can the title span many lines?

Method Types

Class initialization	+ initialize
Drag support	+ destinationDragCell + setDestinationDragCell: + sourceDragCell + setSourceDragCell: + dimmedSourceImage + setDimmedSourceImage: + dimmedDestinationImage + setDimmedDestinationImage:
Initialization/deallocation	- init - dealloc
NSCell overrides	- acceptsFirstResponder - isOpaque

Cell accessors	- textCell
	- setTextCell:
	- imageCell
	- setImageCell:
Our images	- acceptingImage
	- setAcceptingImage:
	- setImage:
Title manipulation	- title
	- setTitle:
	- displaysTitle
	- setDisplaysTitle:
	- isTitleEditable
	- setTitleEditable:
	- isTitleMultiline
	- setTitleMultiline:
Dragging options	- allowsSourceDragging
	- setAllowsSourceDragging:
	- allowsDestinationDragging
	- setAllowsDestinationDragging:
	- acceptsForeignDrag
	- acceptsLocalDrag
	- acceptsSelfDrag
	- retainsData
	- shadowsIncoming
	- setShadowsIncoming:
	- shadowColor
	- dragImageSlidesBack
Destination dragging	- acceptingPasteboardTypes
	- cleanupAfterDestinationDrag
Dragging pasteboard	- draggingPasteboard
Source dragging	- prepareForSourceDrag
	- trackMouse:inRect:ofView:untilMouseUp:
	- calculateDragPoint:andOffset:

Drags in progress	- isSourceDragInProgress - isDestinationDragInProgress
Sizing ourselves	- calcDrawInfo: - cellSize - cellSizeForBounds: - drawingRectForBounds: - imageRectForBounds: - titleRectForBounds:
Display options	- backgroundColor - setBackgroundColor: - isBezeled - setBezeled: - isBordered - setBordered: - borderType - setBorderType: - imageToDisplay - titleToDisplay - imageToDrag
Dimmed image creation	- createDimmedSourceImage - createDimmedDestinationImageUsingImage:
Display hooks	- prepareTextCellForDisplay
Drawing	- drawWithFrame:inView: - drawInteriorWithFrame:inView:
Copying	- copyWithZone:
Archiving	- initWithCoder: - encodeWithCoder:

Class Methods

destinationDragCell

+ (MiscDragCell*)**destinationDragCell**

Returns the drag cell that's currently acting as the drag destination, or **nil** either if there's no drag going on or the destination is some other object.   If instances want to know if they are the current drag destination they should use our **isDestinationDragInProgress** instance method.

dimmedDestinationImage

+ (NSImage*)**dimmedDestinationImage**

Used internally.   Returns our dimmed destination image.   Since only a single drag can be taking place at any time I decided to use a class variable to hold the image.

dimmedSourceImage

+ (NSImage*)**dimmedSourceImage**

Used internally.   Returns our dimmed source image.   Since only a single drag can be taking place at any time I decided to use a class variable to hold the image.

initialize

+ (void)**initialize**

Our class initializer.   Sets our class version for use with archiving.

setDestinationDragCell:

+ (void)**setDestinationDragCell:**(MiscDragCell*)*currentSource*

Sets newDest as the current drag destination.   When the destination drag ends this method should be called with a **nil** argument.

setDimmedDestinationImage:

+ (void)**setDimmedDestinationImage:**(NSImage*)*newImage*

Used internally.   Sets our current dimmed image for the destination drag cell.

setDimmedSourceImage:

+ (void)**setDimmedSourceImage:**(NSImage*)*newImage*

Sets our dimmed source image to *newImage*.

setSourceDragCell:

+ (void)**setSourceDragCell:**(MiscDragCell*)*currentDest*

Sets newSource as the current drag source.   When the source drag ends this method should be called with a **nil** argument.

sourceDragCell

+ (MiscDragCell*)**sourceDragCell**

Returns the drag cell that's currently acting as the drag source, or **nil** either if there's no drag going on or the source is some other object (that is not a subclass of MiscDragCell).   If instances want to know if they are the current drag source they should use our **isSourceDragInProgress** instance method.

Instance Methods

acceptingImage

- (NSImage*)**acceptingImage**

Returns the image that's dislayed when we are accepting an image, or **nil** if there is no accepting image.   This method overrides any value returned by our **shadowsIncoming** method.

acceptingPasteboardTypes

- (NSArray*)**acceptingPasteboardTypes**

This method is for subclasses to override to return the pasteboard types that it will accept (as the destination of a drag).   In contrast, for a source drag you don't have to define what you will drag.   You can put data of any type on the pasteboard in our **prepareForSourceDrag** method.   Our implementation returns **nil**.

acceptsFirstResponder

- (BOOL)**acceptsFirstResponder**

Drag Views and Cells cannot be manipulated by anything other than the mouse so they don't accept first responder.

acceptsForeignDrag

- (BOOL)**acceptsForeignDrag**

Returns YES if we accept drags that don't originate from within our own application. The default is YES. Subclasses can override this to change this behavior.

acceptsLocalDrag

- (BOOL)**acceptsLocalDrag**

Returns YES if we accept drags that originate from within our own application. The default is YES. Subclasses can override this to change this behavior.

acceptsSelfDrag

- (BOOL)**acceptsSelfDrag**

Returns YES if we accept drags that originated from our own drag cell. The default is YES. You see this behavior in the Workspace shelf. If you drag a file from one of the cells you are still able to put it back in the same cell. If this method returns YES then **acceptsLocalDrag** should also return YES. Subclasses can override this to change this behavior.

allowsDestinationDragging

- (BOOL)**allowsDestinationDragging**

Returns YES if we are allowed to accept a drag. The default is YES.

allowsSourceDragging

- (BOOL)**allowsSourceDragging**

Returns YES if we can be the source of a dragging session, or NO if we cannot.

backgroundColor

- (NSColor*)**backgroundColor**

Returns the background color we use if happen to draw our background (which we only do if we have a border). Otherwise we are non-opaque and our view behind us will end up drawing our background. By default our background color is light gray.

borderType

- (NSBorderType)**borderType**

Returns our border type, which will be one of NSNoBorder (the default), NSBezelBorder, NSLineBorder, or NSGrooveBorder.

calcDrawInfo:

- (void)**calcDrawInfo:**(NSRect)*rect*

Doesn't do anything yet.

calculateDragPoint:andOffset:

- (void)**calculateDragPoint:**(NSPoint *)*dragPoint*
    andOffset:(NSPoint *)*offset*

This method should be overridden if you want to have some control on where the dragged image is first placed, and how far it is *offset* from the original mousedown (*dragPoint*). Making the *dragPoint* be the middle of the image is the default,

cellSize

- (NSSize)**cellSize**

Returns our current cell size.

cellSizeForBounds:

- (NSSize)**cellSizeForBounds:**(NSRect)*bounds*

This method hasn't been implemented correctly yet.   It just returns the size from our **cellSize** method.

cleanupAfterDestinationDrag

- (void)**cleanupAfterDestinationDrag**

Internally used.   This is called after a destination drag whether it was a success or failure.

copyWithZone:

- (id)**copyWithZone:**(NSZone*)*zone*

Returns a copy of the receiver.

createDimmedDestinationImageUsingImage:

- (void)**createDimmedDestinationImageUsingImage:**(NSImage*)*origImage*

Creates our dimmed destination image by copying *origImage* and compositing our shadow color over it.   This copy can be retrieved by calling [[**self** class] dimmedDestinationImage].

createDimmedSourceImage

- (void)**createDimmedSourceImage**

Creates our dimmed source image by copying our image and compositing our shadow color over it.   This copy can be retrieved by calling [[**self** class] dimmedSourceImage].

dealloc

- (void)**dealloc**

Releases our resources.

displaysTitle

- (BOOL)**displaysTitle**

Returns YES if we display our title under our image.

dragImageSlidesBack

- (BOOL)**dragImageSlidesBack**

Returns YES if the dragging image should slide back to it's destination when it isn't deposited in another view. The default is to slide back only if we retain our data (**retainsData** returns YES).

draggingPasteboard

- (NSPasteboard*)**draggingPasteboard**

Override this if you would like to use a different pasteboard for dragging. The default is the NSDragPboard.

drawInteriorWithFrame:inView:

- (void)**drawInteriorWithFrame:**(NSRect)*frame*
  inView:(NSView*)*controlView*

Draws our background if we have a border (using our background color). We then draw our image and title (if **displaysTitle** returns YES) if we have one.

drawWithFrame:inView:

- (void)**drawWithFrame:**(NSRect)*frame*
  inView:(NSView*)*controlView*

Draws the border if there is one, then offsets the NSRect and passes it to **drawInteriorWithFrame:inView:**.

drawingRectForBounds:

- (NSRect)**drawingRectForBounds:**(NSRect)*bounds*

This method hasn't been implemented correctly yet.

encodeWithCoder:

- (void)**encodeWithCoder:**(NSCoder*)*aCoder*

Archives an instance of MiscDragCell.

imageCell

- (NSCell*)**imageCell**

Returns the cell we use to draw our image. You shouldn't need to use this method unless you create a subclass and want to use a custom cell.

imageRectForBounds:

- (NSRect)**imageRectForBounds:**(NSRect)*bounds*

This method hasn't been implemented correctly yet.

imageToDisplay

- (NSImage*)**imageToDisplay**

Returns the image we should be displaying in our cell. This is checked every time we are asked to draw ourselves. If we are not the middle of a drag then we just display our image (as returned by [**self** image]). Otherwise if we are the destination drag cell we can either return our accepting image (if we have one) or a dimmed destination image (if **shadowsIncoming** returns YES). If we are the current source drag cell then we'll either return our own image, or a dimmed source image (if we don't retain our data).

imageToDrag

- (NSImage*)**imageToDrag**

By default we just return [**self** image]. Subclasses can alter this to return a different image. For instance, a subclass that drags TIFF images could override this method to return the standard TIFF icon.

init

- (id)**init**

Our designated initializer. Since **initTextCell:** and **initImageCell:** don't seem to make much sense for a drag cell you have to use this method. Calling either of the other two more common NSCell initializer methods will result in a run-time error.

initWithCoder:

- (id)**initWithCoder:**(NSCoder*)*aDecoder*

Unarchives an instance of MiscDragCell.

isBezeled

- (BOOL)**isBezeled**

Overridden from NSCell since we are using the NSBox-like methods **setBorderType:/borderType** to determine our border.   We return YES if our border type is NSBezelBorder.

isBordered

- (BOOL)**isBordered**

Overridden from NSCell since we are using the NSBox-like methods **setBorderType:/borderType** to determine our border.   We return YES if our border type is NSLineBorder.

isDestinationDragInProgress

- (BOOL)**isDestinationDragInProgress**

Returns YES if we are currently the destination of a drag.   This is most often used so we can draw shadowed images, etc.   in our drawing methods.

isOpaque

- (BOOL)**isOpaque**

If we have a border then we cover our entire area (and return YES).   If we don't, we won't (returns NO).

isSourceDragInProgress

- (BOOL)**isSourceDragInProgress**

Returns YES if we are currently the source of a drag.   This is most often used so we can draw shadowed images, etc.   in our drawing methods.

isTitleEditable

- (BOOL)**isTitleEditable**

Titles aren't currently editable so the return value doesn't really matter.

isTitleMultiline

- (BOOL)**isTitleMultiline**

Not implemented yet.

prepareForSourceDrag

- (BOOL)**prepareForSourceDrag**

This method is meant to be overridden in subclasses to allow you to put the data you want to send on the dragging pasteboard.   Our implementation returns NO.

prepareTextCellForDisplay

- (void)**prepareTextCellForDisplay**

This method is called just before the text is drawn.   We currently just set the cell's string value to the title we are displaying (whether we really display it or not depends on the value returned by **displaysTitle**).   If you extend this method make sure and do a [**super** prepareTextCellForDisplay].

retainsData

- (BOOL)**retainsData**

Returns YES if when we drag that we leave a copy of ourselves behind.   If you were to emulate the Workspace shelf then this would return NO.   The default is NO.   Subclasses can override this to change this behavior.

setAcceptingImage:

- (void)**setAcceptingImage:**(NSImage*)*anImage*

Sets the image that's displayed when we are the destination of a drag.   This would be useful for instance if we were trying to write a folder subclass that showed the open folder when it accepts the drag.

setAllowsDestinationDragging:

- (void)**setAllowsDestinationDragging:**(BOOL)*aBool*

Sets whether we are allowed to accept a drag. The default is YES.

setAllowsSourceDragging:

- (void)**setAllowsSourceDragging:**(BOOL)*aBool*

Sets whether we can be the source of a dragging session.

setBackgroundColor:

- (void)**setBackgroundColor:**(NSColor*)*newColor*

Sets our background color. By default it is light gray. See our **backgroundColor** method to see when we use our background color.

setBezeled:

- (void)**setBezeled:**(BOOL)*nowBezeled*

If *nowBezeled* is YES we will draw an NSBezelBorder. If *nowBezeled* is NO we set our border type to NSNoBorder.

setBorderType:

- (void)**setBorderType:**(NSBorderType)*aType*

Sets our border type. *aType* should be one of NSNoBorder, NSBezelBorder, NSLineBorder or NSGrooveBorder.

setBordered:

- (void)**setBordered:**(BOOL)*nowBordered*

If *nowBordered* is YES we will draw an NSLineBorder. If *nowBordered* is NO we set our border type to NSNoBorder.

setDisplaysTitle:

- (void)**setDisplaysTitle:**(BOOL)*displays*

Sets whether we display a title.

setImage:

- (void)**setImage:**(NSImage*)*anImage*

Sets the image we display.   If newImage is **nil** we set our title to be the empty string.

setImageCell:

- (void)**setImageCell:**(NSCell*)*newImageCell*

Sets the image cell we use to draw our image.   The default is just an NSCell initialized with **initImageCell:**.

setShadowsIncoming:

- (void)**setShadowsIncoming:**(BOOL)*aBool*

Sets whether we show a dimmed image when there is an incoming drag.   The default is YES.

setTextCell:

- (void)**setTextCell:**(NSCell*)*newTextCell*

Sets the cell we use to draw our title.   By default it is an instance of MiscAbbreviatedTextCell, which takes care of putting ".." at the end of titles that are too long to display.

setTitle:

- (void)**setTitle:**(NSString*)*newTitle*

Sets our title.   This may or may not be displayed under our image depending on what our **displaysTitle** method returns.   returns YES.

setTitleEditable:

- (void)**setTitleEditable:**(BOOL)*isEditable*

Not implemented yet.

setTitleMultiline:

- (void)**setTitleMultiline:**(BOOL)*isMultiline*

Not implemented yet.

shadowColor

- (NSColor*)**shadowColor**

Returns the color used to created a dimmed appearance for a destination image. The default is to return a partially transparent gray. If you want a different appearance for shadowing, you can override this method.

shadowsIncoming

- (BOOL)**shadowsIncoming**

Returns YES if we show a dimmed image when there is an incoming drag. This assumes that we are currently accepting drags. The default is YES.

textCell

- (NSCell*)**textCell**

Returns the cell we use to draw our title. You shouldn't need to use this method except in other methods like **prepareTextCellForDisplay**.

title

- (NSString*)**title**

Returns our current title. This may or may not be displayed, depending upon what **displaysTitle** returns.

titleRectForBounds:

- (NSRect)**titleRectForBounds:**(NSRect)*bounds*

This method hasn't been implemented correctly yet.

titleToDisplay

- (NSString*)**titleToDisplay**

Returns the title that'll be displayed along with our image as long as **displaysTitle** returns YES. By default we just return [**self** title] but subclasses can easily modify this method.

trackMouse:inRect:ofView:untilMouseUp:

- (BOOL)**trackMouse:**(NSEvent*)*theEvent*
  inRect:(NSRect)*rect*
  ofView:(NSView*)*controlView*
  untilMouseUp:(BOOL)*untilMouseUp*

We take care of starting a source drag, as long as the following conditions are met: (1) We allow source dragging, (2) our image is not **nil**, (3) prepareForSourceDrag returns YES, and the mouse is moved at least 5 pixels from where the original mouse down occured. Returns YES no matter what happens, though I'm not sure if that's correct.