NSClipView

Inherits From: NSView : NSResponder : NSObject

Conforms To: NSCoding (NSResponder)
NSObject (NSObject)

Declared In: AppKit/NSClipView.h

Class at a Glance

Purpose
An NSClipView contains and scrolls the document view displayed by an NSScrollView. You normally don't need to program with NSClipViews, as NSScrollView handles most of the details of their operation.

Principal Attributes
- Efficient scrolling by copying drawn portions of the document view
- Monitoring of document view for automatic update

Creation
Interface Builder

– initWithFrame:	Initializes the NSClipView.

Commonly Used Methods

– setDocumentView:	Sets the view scrolled within the NSClipView.
– setCopiesOnScroll:	Sets whether the NSClipView copies drawn portions of the document view during scrolling.

Class Description

An NSClipView holds the document view of an NSScrollView, clipping the document view to its frame, handling the details of scrolling in an efficient manner, and updating the NSScrollView when the document view's size or position changes. You don't normally use the NSClipView class directly; it's provided primarily as the scrolling machinery for the NSScrollView class. However, you might use the NSClipView class to implement a class similar to NSScrollView.

When an NSClipView is instructed to scroll its document view, it copies as much of the already-drawn document view as possible. This allows for efficient scrolling by obviating the need to redraw large portions of the document view. The NSClipView then sends its document view a **setNeedsDisplayInRect:** message to mark as invalid the newly exposed region(s) of the document view. If copying drawn areas is inappropriate for your needs, you can turn it off by sending the NSClipView a **setCopiesOnScroll:** message with an argument of NO.

In addition to performing the details of scrolling, an NSClipView monitors its document view and sends its superview (usually an NSScrollView) a **reflectScrolledClipView:** message whenever the relationship between the NSClipView and the document view has changed. This allows the superview to update itself to reflect the change—for example, an NSScrollView uses this method to change the position of its scrollers when the user causes the document view to autoscroll or when the document view's size changes.

Method Types

Setting the document view	– setDocumentView: – documentView
Scrolling	– scrollToPoint: – autoscroll: – constrainScrollPoint:
Determining scrolling efficiency	– setCopiesOnScroll: – copiesOnScroll
Getting the visible portion	– documentRect – documentVisibleRect
Setting the document cursor	– setDocumentCursor: – documentCursor
Setting the background color	– setBackgroundColor: – backgroundColor

Overridden NSView methods
– acceptsFirstResponder
– becomeFirstResponder
– isFlipped
– rotateByAngle:
– scaleUnitSquareToSize:
– setBoundsOrigin:
– setBoundsRotation:
– setBoundsSize:
– setFrameSize:
– setFrameOrigin:
– setFrameRotation:
– setNextKeyView:
– translateOriginToPoint:
– viewBoundsChanged:
– viewFrameChanged:

Instance Methods

acceptsFirstResponder

– (BOOL)**acceptsFirstResponder**

Returns YES if the receiver has a document view, NO otherwise.

See also: – **documentView**, – **acceptsFirstResponder** (NSResponder)

autoscroll:

– (BOOL)**autoscroll:**(NSEvent *)*theEvent*

Scrolls the receiver proportionally to *theEvent*'s distance outside of it. *theEvent*'s location should be expressed in the window's base coordinate system (which it normally is), not the receiving NSClipView's. Returns YES if any scrolling is performed; otherwise returns NO.

Never invoke this method directly; instead, the NSClipView's document view should repeatedly send itself **autoscroll:** messages when the mouse is dragged outside the NSClipView's frame during a modal event loop initiated by a mouse-down event. The NSView class implements **autoscroll:** to forward the message to the receiver's superview; thus the message is ultimately forwarded to the NSClipView.

backgroundColor

– (NSColor *)**backgroundColor**

Returns the color of the receiver's background.

See also: – **setBackgroundColor:**

becomeFirstResponder

– (BOOL)**becomeFirstResponder**

If the key view selection direction of the receiver's NSWindow isn't NSSelectingPrevious, attempts to make the document view the first responder. If the direction is NSSelectingPrevious, attempts to make the receiver's previous key view (typically the containing NSScrollView) the first responder. Returns YES if successful and NO otherwise

See also: – **becomeFirstResponder:** (NSResponder), – **makeFirstResponder:** (NSWindow), – **keyViewSelectionDirection** (NSWindow)

constrainScrollPoint:

– (NSPoint)**constrainScrollPoint:**(NSPoint)*proposedNewOrigin*

Returns a scroll point adjusted from *proposedNewOrigin*, if necessary, to guarantee that the receiver will still lie within its document view. For example, if *proposedNewOrigin*'s *y* coordinate lies to the left of the document view's origin, then the *y* coordinate returned is set to that of the document view's origin.

See also: – **scrollToPoint:**

copiesOnScroll

– (BOOL)**copiesOnScroll**

Returns YES if the receiver copies its existing rendered image while scrolling (only drawing exposed portions of its document view), NO if it forces its contents to be redrawn each time.

See also: – **setCopiesOnScroll:**

documentCursor

– (NSCursor *)**documentCursor**

Returns the cursor object used when the mouse lies over the receiver.

See also: – **setDocumentCursor:**

documentRect

– (NSRect)**documentRect**

Returns the rectangle defining the document view's frame, adjusted to the size of the receiver if the document view is smaller. In other words, this rectangle is always at least as large as the receiver itself.

The document rectangle is used in conjunction with an NSClipView's bounds rectangle to determine values for the indicators of relative position and size between the NSClipView and its document view. For example, NSScrollView uses these rectangles to set the size and position of the knobs in its scrollers. When the document view is much larger than the NSClipView, the knob is small; when the document view is near the same size, the knob is large; and when the document view is the same size or smaller, there is no knob.

See also: – **reflectScrolledClipView:** (NSScrollView), – **documentVisibleRect**

documentView

– (id)**documentView**

Returns the receiver's document view.

See also: – **setDocumentView:**

documentVisibleRect

– (NSRect)**documentVisibleRect**

Returns the exposed rectangle of the receiver's document view, in the document view's own coordinate system. Note that this rectangle doesn't reflect the effects of any clipping that may occur above the NSClipView itself. To get the portion of the document view that's guaranteed to be visible, send it a **visibleRect** message.

See also: – **documentRect**

isFlipped

– (BOOL)**isFlipped**

Returns YES if the document view is flipped, NO if it isn't.

See also: – **isFlipped** (NSView)

rotateByAngle:

– (void)**rotateByAngle:**(float)*angle*

Overrides NSView's implementation to disable rotation.

scaleUnitSquareToSize:

– (void)**scaleUnitSquareToSize:**(NSSize)*newUnitSize*

Performs as NSView's implementation and updates a containing NSScrollView based on the new bounds.

scrollToPoint:

– (void)**scrollToPoint:**(NSPoint)*newOrigin*

Changes the origin of the receiver's bounds rectangle to *newOrigin*.

See also: – **constrainScrollPoint:**

setBackgroundColor:

– (void)**setBackgroundColor:**(NSColor *)*aColor*

Sets the receiver's background color to *aColor*.

See also: – **backgroundColor**

setBoundsOrigin:

– (void)**setBoundsOrigin:**(NSPoint)*aPoint*

Performs as NSView's implementation and updates a containing NSScrollView based on the new bounds.

setBoundsRotation:

– (void)**setBoundsRotation:**(float)*angle*

Overrides NSView's implementation to disable rotation.

setBoundsSize:

– (void)**setBoundsSize:**(NSSize)*aSize*

Performs as NSView's implementation and updates a containing NSScrollView based on the new bounds.

setCopiesOnScroll:

– (void)**setCopiesOnScroll:**(BOOL)*flag*

Controls whether the receiver copies rendered images while scrolling. If *flag* is YES, the receiver copies the existing rendered image to its new location while scrolling, and only draws exposed portions of its document view. If *flag* is NO, the receiver always forces its document view to draw itself on scrolling.

See also: – **copiesOnScroll**

setDocumentCursor:

– (void)**setDocumentCursor:**(NSCursor *)*aCursor*

Sets the cursor object used over the receiver to *aCursor*.

See also: – **documentCursor**

setDocumentView:

– (void)**setDocumentView:**(NSView *)*aView*

Sets the receiver's document view to *aView*, removing any previous document view, and sets the origin of the receiver's bounds rectangle to the origin of *aView*'s frame rectangle. If the receiver is contained in an NSScrollView, you should send the NSScrollView a **setDocumentView:** message instead, so that it can perform whatever updating it needs.

In the process of setting the document view, this method registers the receiver for the notifications NSViewFrameDidChangeNotification and NSViewBoundsDidChangeNotification, adjusts the key view loop to include the new document view, and updates a parent NSScrollView's display if needed using **reflectScrolledClipView:**.

See also: – **documentView**

setFrameOrigin:

– (void)**setFrameOrigin:**(NSPoint)*aPoint*

Performs as NSView's implementation and updates a containing NSScrollView based on the new bounds.

setFrameRotation:

– (void)**setFrameRotation:**(float)*angle*

Overrides NSView's implementation to disable rotation.

setFrameSize:

– (void)**setFrameSize:**(NSSize)*aSize*

Performs as NSView's implementation and updates a containing NSScrollView based on the new bounds.

setNextKeyView:

– (void)**setNextKeyView:**(NSView *)*aView*

Performs as NSView's implementation, except inserts the receiver's document view between itself and *aView* in the key view loop.

See also: – **setNextKeyView:** (NSView)

translateOriginToPoint:

– (void)**translateOriginToPoint:**(NSPoint)*aPoint*

Performs as NSView's implementation and updates a containing NSScrollView based on the new bounds.

viewBoundsChanged:

– (void)**viewBoundsChanged:**(NSNotification *)*aNotification*

Handles an NSViewBoundsDidChangeNotification by updating a containing NSScrollView based on the new bounds.

viewFrameChanged:

– (void)**viewFrameChanged:**(NSNotification *)*aNotification*

Handles an NSViewFrameDidChangeNotification by updating a containing NSScrollView based on the new frame.

Methods Implemented by an NSClipView's Superview

reflectScrolledClipView:

– (void)**reflectScrolledClipView:**(NSClipView *)*aClipView*

Notifies *aClipView*'s superview that either *aClipView*'s bounds rectangle or the document view's frame rectangle has changed, and that any indicators of the scroll position need to be adjusted. NSScrollView implements this method to update its NSScrollers.

scrollClipView:toPoint:

– (void)**scrollClipView:**(NSClipView *)*aClipView* **toPoint:**(NSPoint)*newOrigin*

Notifies *aClipView*'s superview that *aClipView* needs to set its bounds rectangle origin to *newOrigin*. *aClipView*'s superview should then send a **scrollToPoint:** message to *aClipView* with *newOrigin* as the argument. This mechanism is provided so that the NSClipView's superview can coordinate scrolling of multiple tiled NSClipViews.

See also: – **scrollToPoint:** (NSClipView)