NSScroller

Inherits From: NSControl : NSView : NSResponder : NSObject

Conforms To: NSCoding (NSResponder)
NSObject (NSObject)

Declared In: AppKit/NSScroller.h

Class at a Glance

Purpose
An NSScroller object is a user control for scrolling a document view within a container view. You normally don't need to program with NSScrollers, as Interface Builder allows you to fully configure them with an NSScrollView.

Principal Attributes
- Scrolling by small and large increments
- Proportional knob showing visible amount of document

Creation
Interface Builder

– initWithFrame:	Initializes the NSScroller.

Commonly Used Methods

– hitPart	Indicates where the user clicked the NSScroller.
– floatValue (NSControl)	Returns the position of the NSScroller's knob.
– setFloatValue:knobProportion:	Sets the position and size of the NSScroller's knob.

Class Description

An NSScroller controls scrolling of a document view within an NSScrollView's clip view (or potentially another kind of container view). It typically displays a pair of buttons that the user can click to scroll by a small amount (called a line increment or decrement) and Alternate-click to scroll by a large amount (called a page increment or decrement), plus a slot containing a knob that the user can drag directly to the desired location. The knob indicates both the position within the document view and, by varying in size within the

slot, the amount visible relative to the size of the document view. You can configure whether an NSScroller uses scroll buttons, but it always draws the knob when there's room for it.

Interface Builder automatically associates NSScrollers with an NSScrollView, allowing you to configure most aspects of scrolling behavior without programming. If you create one programmatically using **initWithFrame:** the new NSScroller automatically collapses the shorter of its two dimensions to the standard scroller width, as returned by the **scrollerWidth** class method. In this manner a tall, narrow frame results in a vertical NSScroller and a short, wide frame results in a horizontal NSScroller.

Interaction with a Container View

NSScroller is a public class primarily for programmers who decide not to use an NSScrollView but who want to present a consistent user interface. Its use outside of NSScrollViews is discouraged except in cases where the porting of an existing application is made more straightforward. Setting up an NSScroller with a custom container view class (or a completely different kind of target) involves establishing the standard target and action as defined by NSControl, and implementing the target's action method appropriately.

As an NSScroller tracks the mouse, it sends an action message to its target with itself as the argument. The target then performs the scrolling operation based on these things:

- The orientation of the NSScroller, vertical or horizontal, which determines the axis to scroll along. This can be done by getting the NSScroller's frame and determining the longer dimension, or by keeping a reference to designated vertical and horizontal scrollers (as NSScrollView does).
- The direction and scale of scrolling. NSScroller's **hitPart** method returns this information, described in more detail below.
- If the hit part is the knob or its slot, the NSScroller's value, which indicates where to position the document view in the container view. The **floatValue** method provides this information, which the target should interpret relative to the size of the document view's frame minus the size of the container view's bounds.

As indicated above, the direction and scale of a scrolling operation is determined by the value returned from a **hitPart** message, which indicates the various parts of an NSScroller in terms of their intended result (not in terms of their location or appearance):

Value	How to Scroll
NSScrollerKnob	Directly to the NSScroller's value, as given by **floatValue**
NSScrollerKnobSlot	Directly to the NSScroller's value, as given by **floatValue**
	NSScrollerDecrementLineUp or left by a small amount
	NSScrollerDecrementPageUp or left by a large amount
	NSScrollerIncrementLineDown or right by a small amount
	NSScrollerIncrementPageDown or right by a large amount
	NSScrollerNoPartDon't scroll at all

Note: These part codes are interpreted differently depending on the method you use them with. See the individual method descriptions for **hitPart**, **rectForPart:**, and **testPart:** for details.

The four decrement/increment values require the target to calculate an appropriate amount to scroll by. Line and page amounts are up to the target or the container view to define. NSScrollView, for example, allows you to set these in Interface Builder or with its **setLineScroll:** and **setPageScroll:** methods. Once the target has scrolled the document view by a decrement or increment, it should update the NSScroller's position using **setFloatValue:**.

The container view or target should also keep tabs on its size and on the size and position of its document view. Any time these change it should update its NSScrollers using **setFloatValue:knobProportion:**. NSClipView, for example, overrides most of NSView's **setBounds...** and **setFrame...** methods to perform this updating.

Method Types

Determining NSScroller size	+ scrollerWidth
Laying out an NSScroller	– setArrowsPosition: – arrowsPosition
Setting the knob position	– setFloatValue:knobProportion: – knobProportion
Calculating layout	– rectForPart: – testPart: – checkSpaceForParts – usableParts
Drawing the parts	– drawArrow:highlight: – drawKnob – drawParts – highlight:
Event handling	– hitPart – trackKnob: – trackScrollButtons:

Class Methods

scrollerWidth

+ (float)**scrollerWidth**

Returns the width of instances. NSScrollView uses this value to lay out its components. Subclasses that use a different width should override this method.

Instance Methods

arrowsPosition

– (NSScrollArrowPosition)**arrowsPosition**

Returns the location of the scroll buttons within the receiver, as described under **setArrowsPosition:**.

checkSpaceForParts

– (void)**checkSpaceForParts**

Checks to see if there is enough room in the receiver to display the knob and buttons. **usableParts** returns the state calculated by this method. You should never need to invoke this method; it's invoked automatically whenever the NSScroller's size changes.

drawArrow:highlight:

– (void)**drawArrow:**(NSScrollerArrow)*arrow* **highlight:**(BOOL)*flag*

Draws the scroll button indicated by *arrow*, which is either NSScrollerIncrementArrow (the down or right scroll button) or NSScrollerDecrementArrow (up or left). If *flag* is YES, the button is drawn highlighted, otherwise it's drawn normally. You should never need to invoke this method directly, but may wish to override it to customize the appearance of scroll buttons.

See also: – **drawKnob**, – **rectForPart:**

drawKnob

– (void)**drawKnob**

Draws the knob. You should never need to invoke this method directly, but may wish to override it to customize the appearance of the knob.

See also: – **drawArrow:highlight:**, – **rectForPart:**

drawParts

– (void)**drawParts**

Caches images for the scroll buttons and knob. It's invoked only once when the NSScroller is created. You may want to override this method if you alter the look of the NSScroller, but you should never invoke it directly.

highlight:

– (void)**highlight:**(BOOL)*flag*

Highlights or unhighlights the scroll button that the user clicked. The receiver invokes this method while tracking the mouse; you should not invoke it directly. If *flag* is YES, the appropriate part is drawn highlighted, otherwise it's drawn normally.

See also: – **drawArrow:highlight:**, – **rectForPart:**

hitPart

– (NSScrollerPart)**hitPart**

Returns a part code indicating the manner in which the scrolling should be performed:

Value	How to Scroll
NSScrollerKnob	Directly to the NSScroller's value, as given by **floatValue**
NSScrollerKnobSlot	Directly to the NSScroller's value, as given by **floatValue**
NSScrollerDecrementLine	Up or left by a small amount
NSScrollerDecrementPage	Up or left by a large amount
NSScrollerIncrementLine	Down or right by a small amount
NSScrollerIncrementPage	Down or right by a large amount
NSScrollerNoPart	Don't scroll at all

This method is typically invoked by an NSScrollView to determine how to scroll its document view when it receives an action message from the NSScroller.

initWithFrame

– (id)**initWithFrame:**(NSRect)*frameRect*

Initializes the receiver as normal, but collapsing *frameRect*'s narrower dimension to the value returned by the **scrollerWidth** method. This enforces the appearance of vertical and horizontal NSScrollers. This is the designated initializer for the NSScroller class. Returns **self**.

knobProportion

– (float)**knobProportion**

Returns the portion of the knob slot that the knob should fill, as a floating-point value from 0.0 (minimal size) to 1.0 (fills the slot).

rectForPart:

– (NSRect)**rectForPart:**(NSScrollerPart)*aPart*

Returns the rectangle occupied by *aPart*, which for this method is interpreted literally rather than as an indicator of scrolling direction:

Value	Part
NSScrollerKnob	The knob itself
NSScrollerKnobSlot	The slot that the knob moves in
NSScrollerDecrementLine	The up or left scroll button
NSScrollerDecrementPage	The region of the slot above or to the left of the knob
NSScrollerIncrementLine	The down or right scroll button
NSScrollerIncrementPage	The region of the slot below or to the right of the knob

Note the interpretations of NSScrollerDecrementPage and NSScrollerIncrementPage. The actual part of an NSScroller that causes page-by-page scrolling varies from platform to platform, so as a convenience these part codes refer to useful parts different from the scroll buttons.

Returns NSZeroRect if the part requested isn't present on the receiver.

See also: – **hitPart**, – **testPart:**, – **usableParts**

setArrowsPosition:

– (void)**setArrowsPosition:**(NSScrollArrowPosition)*location*

Sets the location of the scroll buttons within the Scroller to *location*, or inhibits their display, as follows:

Value	Meaning
NSScrollerArrowsMaxEnd	Buttons at bottom or right
NSScrollerArrowsMinEnd	Buttons at top or left
NSScrollerArrowsNone	No buttons

Note: On Microsoft Windows scroll buttons appear at either end of the scroller rather than both on one end. A value other than NSScrollerArrowsNone is thus reinterpreted on that platform to display the buttons at either end.

See also: – **arrowsPosition**

setFloatValue:knobProportion:

– (void)**setFloatValue:**(float)*aFloat* **knobProportion:**(float)*knobProp*

Sets the position of the knob to *aFloat*, which is a value between 0.0 (indicating the top or left end) and 1.0 (the bottom or right end). Also sets the proportion of the knob slot filled by the knob to *knobProp*, also a value between 0.0 (minimal size) and 1.0 (fills the slot).

See also: – **floatValue** (NSControl), – **knobProportion**

testPart:

– (NSScrollerPart)**testPart:**(NSPoint)*aPoint*

Returns the part that would be hit by a mouse-down event at *aPoint* (expressed in the receiver's coordinate system):

Return Value	Part Identified
NSScrollerKnob	The knob itself
NSScrollerKnobSlot	The slot that the knob moves in (returned only if there's no knob)
NSScrollerDecrementLine	The up or left scroll button
NSScrollerDecrementPage	The region of the slot above or to the left of the knob
NSScrollerIncrementLine	The down or right scroll button
NSScrollerIncrementPage	The region of the slot below or to the right of the knob
NSScrollerNoPart	(*aPoint* isn't in the NSScroller)

Note the interpretations of NSScrollerDecrementPage and NSScrollerIncrementPage. The actual part of an NSScroller that causes page-by-page scrolling varies from platform to platform, so as a convenience these part codes refer to useful parts different from the scroll buttons.

See also: – **hitPart**, – **rectForPart:**

trackKnob:

– (void)**trackKnob:**(NSEvent *)*theEvent*

Tracks the knob and sends action messages to the receiver's target. This method is invoked automatically when the NSScroller receives a mouse-down event in the knob; you should not invoke it directly.

trackScrollButtons:

– (void)**trackScrollButtons:**(NSEvent *)*theEvent*

Tracks the scroll buttons and sends action messages to the receiver's target. This method is invoked automatically when the NSScroller receives a mouse-down event in a scroll button; you should not invoke this method directly.

usableParts

– (NSUsableScrollerParts)**usableParts**

Returns a value indicating which parts of the receiver are displayed and usable. This is one of:

Value	Meaning
NSNoScrollerParts	Scroller has neither a knob nor scroll buttons, only the knob slot.
NSOnlyScrollerArrows	Scroller has only scroll buttons, no knob.
NSAllScrollerParts	Scroller has at least a knob, possibly also scroll buttons.

See also: – **checkSpaceForParts**, – **arrowsPosition**