PAScrollViewDeluxe

By Jeff Martin (jmartin@next.com (415) 780-3833)

Inherits From: ScrollView: View : Responder : Object

Declared In: PAScrollViewDeluxe.h

Class Description

This object is an enhanced subclass of ScrollView. It adds the following features:

Support for a 'top view', a view at the top of the scroll view that scrolls horizontally with the document but remains static when the document is scrolled vertically. This is useful to add column headers, rulers, etc, to a document. The top view can be added from inside of IB by connecting the topView outlet.

Support for a 'left view', a view to the left of the scroll view that scrolls vertically with the document, but remains static when the document is scrolled horizontally. This is useful for adding rulers, line numbers, etc, to a document. The left view can be added from inside of IB by connecting the leftView outlet.

Automatic support for rulers. If a topView or leftView has not been set, a call to setShowTopView:YES, setShowLeftView:YES, showRulers: or toggleRulers: will instanciate a member of rulerClass (see setRulerClass:). Depending on which ruler is called, the scrollview will try to call setHorizontal or setVertical on the new rulerClass instance. The default class is Ruler (from Draw). A ruler that supports flipping and smooth zooming is forthcoming.

Support for synchronizing other scroll views. Other ScrollViews can be made to scroll, size and zoom with respect to the docView by using the addSyncViews: methods. This is useful for ruler type views that need to exist outside of the scrollers. One synchronized view of each type can be added in IB by connecting a given ScrollView to the syncViews, horizSyncViews or vertSyncViews outlets. Be sure and connect to a ScrollView and not its content view (this can be tricky for ScrollViews without scrollers). Synchronized ScrollViews should not have scrollers of their own (but they can).

Automatic support for page up/down & left/right buttons. Simply call the method setPageUpDownButtonsVisible:YES or setPageLeftRightButtonsVisible:YES and these are added automatically. They are removed temporarily if the scroller is too small(less than an inch), if there is nothing to be scrolled or if

the scrolling area is less than 2 pages.

Automatic support for zooming. Simply call the method setShowZoomButton:YES and a popup list containing zoom values will be added to the horizontal scroller. Zooming is implemented by calling zoomTo:(float)zoomX : (float)zoomY on the docView(and topView, leftView, syncViews, horizSyncViews and vertSyncViews). If the docView does not respond to this method or returns NO from this method, automatic zooming is performed by scaling the clipView by the zoom amount. The 'Set...' item in the zoom popup allows for arbitrary scaling. The 'Fit' menu item calculates the zoom value necessary to fit the docView entirely in the scrollview. The default zooming behavior tends to fail on documents that contain NXImages. Implement zoomTo:: in your custom view to explicitly scale NXImages (returning NO if you don't otherwise handle zoom).

Support for adding arbitrary views to the vertical and horizontal scrollers. This is useful for adding little gadgets like a 'goto page' control inside of the horizontal scroller. The page up/down & left/right buttons as well as the zoom button use this facility. Page left/right is assumed to be the first in the horizScrollerViews list if it exists. The zoom button is next. Other views should be added at [horizScrollerViews count]. Page up/down is assumed to be first in the vertScrollerViews list. Scroller views are temporarily removed in reverse order if the scroller is not long enough to accommodate them.

The PAScrollViewDeluxe palette allows you to create a PAScrollViewDeluxe by command clicking the 'Group in ScrollView' menu item. The code is a complete hack inside of the PAScrollViewDeluxeInspector code (at the bottom) and introduces a bug into IB that you can no longer drag the default TextObject/ScrollView in (it just

disapears).

Instance Variables

```
id     topView;                 // View at top that scrolls horiz with docView
BOOL   topViewVisible;          // Wheter to show topView
ClipView *topClip;              // The ClipView associated with topView

id          leftView;           // View at left that scrolls vert with docView
BOOL        leftViewVisible;    // Whether to show leftView
ClipView *leftClip;             // The ClipView associated with leftView

Class       rulerClass;         // The class to use for default top/left view
NXSize      rulerSize;          // The size to use for default top/left view

id          syncViews;          // List of views to be sync'ed with docView
id          horizSyncViews;           // List of views to be sync'ed horizontally
id          vertSyncViews;      // List of views to be sync'ed vertically
```

```
id          horizScrollerViews;          // List of views in the horizontal scroller
id          vertScrollerViews;           // List of views in the vertical scroller

Matrix      *pageUpDownButtons;          // Matrix with page up/down buttons
BOOL        pageUpDownButtonsVisible;    // Whether to show page up/down buttons

Matrix      *pageLeftRightButtons;       // Matrix with page left/right buttons
BOOL        pageLeftRightButtonsVisible; // Whether to show page left/rt buttons

Matrix      *zoomButton;                 // Button with zoom popUp.
BOOL        zoomButtonVisible;           // Whether to show zoom popUp

Panel       *zoomPanel;                        // Panel for arbitrary scale
TextField   *zoomText;                   // Text field in zoomPanel
```

Method Types

```
// Query and set the topView
- topView;
```

```
- setTopView:view;

// Query and set whether topView is visible
- (BOOL)topViewVisible;
- setTopViewVisible:(BOOL)flag;

// Convenience methods for setting topView visible inside of IB
- showTopView:sender;
- hideTopView:sender;
- toggleTopView:sender;

// Query and set the leftView
- leftView;
- setLeftView:view;

// Query and set whether leftView is visible
- (BOOL)leftViewVisible;
- setLeftViewVisible:(BOOL)flag;

// Convenience methods for setting leftView visible inside of IB
- showLeftView:sender;
```

- hideLeftView:sender;
- toggleLeftView:sender;

// Convenience methods for showing/hiding/toggling top/left views as a pair
- showRulers:sender;
- hideRulers:sender;
- toggleRulers:sender;

// Query and set the default top/left view class
- (Class)rulerClass;
- setRulerClass:(Class)class;

// Query and set the default top/left view size
- (NXSize)rulerSize;
- setRulerSize:(NXSize)size;

// Query, add and remove views that are sync'ed horizontally with docView
- syncViews;
- addSyncView:view at:(int)at;
- removeSyncView:view;
- removeSyncViewAt:(int)at;

```
// Query, add and remove views that are sync'ed horizontally with docView
- horizSyncViews;
- addHorizSyncView:view at:(int)at;
- removeHorizSyncView:view;
- removeHorizSyncViewAt:(int)at;

// Query, add and remove views that are sync'ed vertically with docView
- vertSyncViews;
- addVertSyncView:view at:(int)at;
- removeVertSyncView:view;
- removeVertSyncViewAt:(int)at;

// Query, add and remove views in the horizontal scroller
- horizScrollerViews;
- addHorizScrollerView:view at:(int)at;
- removeHorizScrollerView:view;
- removeHorizScrollerViewAt:(int)at;

// Query, add and remove views in the vertical scroller
- vertScrollerViews;
```

- addVertScrollerView:view at:(int)at;
- removeVertScrollerView:view;
- removeVertScrollerViewAt:(int)at;

// Query and set whether page up/down buttons are visible or needed
- (BOOL)pageUpDownButtonsVisible;
- setPageUpDownButtonsVisible:(BOOL)flag;
- (BOOL)needPageUpDownButtons;

// Query and set whether page left/right buttons are visible or needed
- (BOOL)pageLeftRightButtonsVisible;
- setPageLeftRightButtonsVisible:(BOOL)flag;
- (BOOL)needPageLeftRightButtons;

// Query and set whether zoom buttons are visible
- (BOOL)zoomButtonVisible;
- setZoomButtonVisible:(BOOL)flag;

// This method tries to call zoomTo:: on support views. Failing that, it scales
- (BOOL)zoomTo:(float)zoomX :(float)zoomY;

```
// Archiving
- write:(NXTypedStream *)stream;
- read:(NXTypedStream *)stream;
```

Class Methods

leftView, setLeftView:
topView, setTopView:

topView and leftView return the current top and left views. If none exists, it allocates a view of ruler class, sets it to be the top or left view and returns it.

setTopView: places the given view inside the scrollview (inside a clip view, topClip) at the top of the scroll view at its origional hieght but at the width of the docView. It returns the oldTopView.

setLeftView: places the given view inside the scrollview (inside a clip view, leftClip) at the left of the scroll view at its origional width but at the height of the docView. It returns the oldLeftView.

(BOOL)topViewVisible, setTopViewVisible:(BOOL)flag

(BOOL)leftViewVisible, setLeftViewVisible:(BOOL)flag

topViewVisible returns whether or not the topView is visible.
setTopViewVisible: will install the current topView inside a clipView on top of the docView if set to YES and will remove the existing topView if set to NO. Retiles the views, but does not call display.Returns self.
leftViewVisible returns whether or not the leftView is visible.
setLeftViewVisible: will install the current leftView inside a clipView on left of the docView if set to YES and will remove the existing leftView if set to NO. Retiles the views, but does not call display. Returns self.

showTopView:, hideTopView: toggleTopView:
showLeftView:, hideLeftView: toggleLeftView:

These convenience methods simply wrap around setTopViewVisible: and setLeftViewVisible and can be set to be called from menus or controls inside of InterfaceBuilder.
getFrameSize:forContentSize:horizScroller:vertScroller:borderType:

showRulers:, hideRuler: toggleRulers:

These are convenience methods for showing/hiding/toggling top and left views as a pair. They wrap around the setTopViewVisible and setLeftViewVisible. These methods can be set to be called from menus or controls inside of InterfaceBuilder. They all return self.

rulerClass, setRulerClass:(Class)class

If the PAScrollView deluxe is asked to show top or left views when none has been set, it attempts to allocate an instance of 'rulerClass' (assumed to be a view). If the instance responds to setHorizontal or setVertical, this will be called.

rulerSize, setRulerSize:(NXSize)size

When PAScrollViewDeluxe allocates a default top/left view, it sets the top one to be of height rulerSize.height and the left one to be of width rulerSize.width. If a topView or leftView are added programatically, the rulerSize.height and rulerSize.width are set respectively.

syncView, addSyncView:at:, removeSyncView:, removeSyncViewAt:
horizSyncViews, addHorizSyncView:at:, removeHorizSyncView: & ViewAt:
vertSyncViews, addVertSyncView:at:, removeVertSyncView: & ViewAt:

syncViews are ScrollViews that are to be scrolled, sized and (optionally)zoomed with respect to the docViews position, size and zoom. horizSyncViews are only affected in the horizontal direction, while vertSyncViews are only affected in the vertical direction. Group a view inside of a ScrollView, disable its horizontal and vertical scrollers, and use one of the addMethods. In IB you can actually set one of each type view by setting the

syncViews, horizSyncViews or vertSyncViews outlet to a ScrollView. It will be added to the list when the outlets are set.

The syncViews, horizSyncViews and vertSyncViews methods return the list of the views that are currently being syncronized in the respective direction(can be NULL if there are none).

addSyncView:at:, addHorizSyncView:at: and addVertSyncView:at: add scrollviews to be synchronized in the respective direction at the given location in the list(use [[myPASV syncViews] count] to add to end). Returns self.

removeSyncView:, removeHorizSyncView:, removeVertSyncView: remove the given view from its respective list by calling removeSyncAt: with the view's index.

removeSyncViewAt:, removeHorizSyncViewAt: and removeVertSyncViewAt: remove ScrollViews from the respective list of syncronized ScrollViews. Returns self.

horizScrollerViews, addHorizScrollerView:at:, removeHorizScrollerView:
vertScrollerViews, addVertScrollerView:at:, removeVertScrollerView:

ScrollerViews are views embedded inside of the vertical or horizontal scrollers. The are frequently simple controls like a "Goto Page:" control. In fact the page up/down & left/right buttons as well as the zoomButton are horizScrollerViews (assumed to be at 0 and 1 respectively if they exist).When added these views are sized to fit into the scroller(ie, horizontal scroller views are constrained to the horizontal scroller's height).

The horizScrollerViews and vertScrollerViews methods return the list of the views that are currently in the respective scroller (can be NULL if there are none).

addHorizScrollerView:at: and addVertScrollerView:at: add a view to their respective list at the given

location(use [[myPASV vertSyncViews] count] to add to end). They returns self.

removeHorizScrollerView: and removeVertScrollerView: removes the given view from the respective scroller list. Returns self.

removeHorizScrollerViewAt: and removeVertScrollerViewAt: removes the view at the given location from the respective scroller list. Returns self.

(BOOL)pageUpDownButtonsVisible, setPageUpDownButtonsVisible:(BOOL)flag, (BOOL)needUpDownButtons
(BOOL)pageLeftRightButtonsVisible, setPageLeftRightButtonsVisible:(BOOL)flag (BOOL)needPageLeftRightButtons

These methods query and set whether the respective button set is visible.

The setButtonsVisible method calls the respective add or remove scrollerView method with the 'at' value equal to zero.

The needPageButtons methods return whether the page buttons are actually needed (ie, if the docView is smaller than the contentView or the scrollable area is less than 2 pages, the buttons are not needed).

zoomButtonVisible, setZoomButtonVisible:

These methods query and set whether the zoom button is visible.

setZoomButtonVisible: either adds the zoomButton to the vert scroller via - addHorizScrollerView: or removes

via removeHorizScrollerView. Returns self.

zoomTo:(float)zoomX :(float)zoomY

This method tries to call zoomTo:: on the docView and all of the accessory views (topView, leftView, syncViews, horizSyncViews, vertSyncViews) with the given scale (1 is full size). If the views implement zoomTo:: and actually do the zoom, they should return YES. If they just implement zoomTo:: to get notification of a zoom or to scale dependent pieces(like NXImages), they should return NO. If zoomTo:: is not implemented or returns NO, automatic scaling takes place(on the ClipView).