ScrollingText

by Sharon Zakhour, NeXT Developer Support Team

Overview

This example shows how to create a document window which contains additional controls in the scroller bars. The UI guidelines in 2.0 suggest that controls which determine how a scrollable document is viewed may be added to the scroller bar area. **Other controls should not be added to this area.** [See the excerpt from the UI guidelines below.]

Specific controls added:

> In the vertical scroll bar a page up button and a page down button are added beneath the scroller arrows.
>
> In the horizontal scroll bar a zoom popup list [with menu items: Set..., Fit Page, 75%, 100%, 125% and 150%] and a page form cell which can be used to "move to" a page are added to the right of the scroller.

To see an actual application which contains these controls, run WriteNow in /NextApps.

Caveats

This example does not actually implement the features of paging up, down, zooming in/out or moving to a specific page. That functionality is left to the user although hooks (in the form of printf's) are provided. It does illustrate how to set up a document which can then support this functionality.

This example does not implement tiling of windows as they are created. [This is something which Edit does and Mail does not do, for example.] However tiling is a useful feature in a real application and should be considered.

Since this example does not create it's own View subclass but uses a Text object for it's default view, it

doesn't elegantly handle background color, highlighting, etc.

Program Organization

How to build the nib files

There are two nib files in this example:

ScrollingText.nib	The main nib file contains the main menu and the classes. No actual UI objects are defined in this file. The file's owner for this nib file is ScrollingText -- a subclass of Application.
ScrollerGadgets.nib	This nib file contains the controls which will be added to the horizontal and vertical scroller bars. Once these gadgets have been added to the ScrollView, the window containing them is freed. The actual document window is created programmatically. [The Document window could have been created in this nib file but would still have needed a fair amount of massaging programmatically to add these controls.] The control objects are also connected via IB to the action items. For example, the zoom popup list button is connected to the PageScrollView method `zoomTo:`; the page up and page down buttons are connected to the method `pageButton:` and the page form cell is connected to the `pageTo:` method. The file's owner for this nib file is PageScrollView -- a subclass of ScrollView.

Classes in the Application

ScrollingText	Subclass of Application. The only method in this class is `-newDocument:sender` which is connected to the Document->New menu item in

the ScrollingText nib file.

Document	Subclass of Window. This class contains two init methods. The `-init` method calls `-initContent:style:backing:buttonMask:defer:screen` with some reasonable default values. The docView is a standard instance of Text. Rather than create an instance of ScrollView, however, this method creates an instance of...
PageScrollView	Subclass of Scrollview. This is the object which adds the controls to the scroller bars. This is also the class where page up/down, zooming in/out and moving to a specific page should be implemented. The `-tile` method is used to keep track of where these controls are positioned. The `-initFrame:` method is responsible for loading the ScrollerGadgets nib file and adding the controls to the subview of the ScrollView. The following hooks are in this class: `-pageButton:`, `-pageTo:`, `-zoomTo:` -- these hooks perform printf's which will appear in the console if you launched from Workspace or the Terminal window if you invoked it there.

Topics Of Interest

How to initialize an outlet using the setMyOutlet method.
For every outlet created for a class -- an implied setMyOutlet method exists but does not explicitly appear in the class code generated by IB. In this example, there are three outlets loaded into PageScrollView via the ScrollerGadgets nib file which need some massaging. IB limits the height of these controls [the zoom popuplist, the page label and the page number form cell] to be higher than desired given the font size. The outlets `-setZoomPopUpList:`, `-setPageLabel:` and `-setPageForm:` re-initialize the height of these controls to the desired size.

How to load a nib section for use in a document.
The controls are available in the ScrollerGadgets nib file. When the user creates a new document this nib file is loaded by the init method of PageScrollView and the objects are then inserted into the

ScrollView. This saves the effort of creating these controls programmatically and insures that the connections are pre-set.

Other Files

ScrollingText_main.m, IB.proj, Makefile, ScrollingText.iconheader	Created by Interface Builder.
PageUp.tiff, PageUpAlt.tiff, PageDown.tiff, PageDownAlt.tiff	TIFF files for up & down arrows

See also:

The ScrollDoodScroll example on the 2.0 release in /NextDeveloper/Examples/ScrollDoodScroll. The zoom popup list is implemented on the graphical window.

The "Controls" chapter of the UI Update in /NextDeveloper/Notes/UIUpdate on the 2.0 release. Excerpted from that chapter:

Controls in the Scroller Area

Controls that determine how a scrollable document is viewed can be placed within the area normally occupied by the scrollers (beneath and to the left of the document). Other sorts of controls should not be placed within this area.

Among the controls that can be placed in the scroller area are these:

- An editable text field to display the current page number can be located to the far right of the horizontal scroller. See WriteNow for an example.
- A ªzoomº pop-up list that lets the user scale the display can be located in the area of the horizontal scroller. See FrameMaker and WriteNow for examples.
- A pop-up list used to control the viewing mode for the display (e.g., preview versus drawing mode in a graphics application) can be similarly situated to the zoom pop-up list in the area of the vertical scroller.
- ªPageº scroll buttons that scroll from page to page or by viewfuls can be grouped next to the line scroll buttons in the lower left corner where the vertical and horizontal scrollers meet. Since there is no Application Kit support for page scroll buttons, but there might be in the future, a precise arrangement is not currently specified.