ScrollTiff

by Phillip C. Dibner, NeXT Developer Support Team

Overview

This MiniExample demonstrates how to load a TIFF file into a ScrollView. To load an image, just select the "Open..." item in the application's main menu.

Program Organization

How to build the nib file

Start up a new application in Interface Builder, as usual. Select the main window (labeled "My Window") and click on the "Inspector..." item in the "Tools" submenu. You will see a Window Inspector with all of the "Controls" check boxes turned on. Click on the "Close" check box so that the check mark disappears; this ensures that the main window will have no close box. This MiniExample provides no way to create a new window, so it should not allow the user to destroy the one that appears when the application is launched.

Next, drag a CustomView from the Palette into the application's main window, and size it as desired. With the Custom View still selected, select "Layout" in the IB main menu, and click on "Group in ScrollView" in the submenu. This creates a new ScrollView that has your Custom View as its DocView .

Now we need to subclass the Custom View so that we can define a drawSelf method for it. Open the Classes suitcase in the File Window. In the classes browser, select View, and invoke the Subclass operation. Name the new subclass TiffDocView. Then select the "Inspector" item in IB's "Tools" submenu, and click on your Custom View. TiffDocView will be one of the obects in the Inspector's scrolling list. Select TiffDocView and press the OK button. The label on your CustomView should change to TiffDocView.

Go to the Menu Palette, and drag a menu Item into the ScrollTiff main menu. Relabel it "Open..." This will be the menu item that loads a new image into the ScrollView.

Next create a controller that will read an image into the TiffDocView. Go back to the Classes

browser, select Object, and execute Subclass on the Operations pull-down menu. Name the new object ImageReader.

The ImageReader will need an outlet that allows it to write an image into your TiffDocView. Select Attributes in the Inspector, and add an Outlet named imageView. Add an Action called readImage. Instantiate the ImageReader.

Now create a connection from the ImageReader to the TiffDocView, and from the "Open..." item in the menu to the readImage action in the ImageReader object.

You will want your ScrollView to resize when you change the size of the main window. Select the ScrollView, open the Inspector, and select the Autosizing feature. Be sure that it is the ScrollView, not the Custom View inspector which is displayed. Your Custom View, the TiffDocView, is autosized with the ScrollView, but autosizing for the Scrollview must be set explicitly. Click on the sections of the horizontal and vertical lines inside the box that represents your view. The portions of the line inside the box will now look like springs. This allows your TiffDocView to track the size of the window when it is made larger or smaller. Be sure to click OK to save this feature into your nib file.

That's it. You can unparse both of the objects, and add code as in the objective C files provided with this MiniExample.

Major Classes in the Application

TiffDocView The DocView within the ScrollView where the TIFF image is displayed. The TiffDocView allocates an NXImage object to manage the TIFF and composites the image data onto the screen.

ImageReader This is the controller object that responds to the main menu, places an OpenPanel on the screen to let the user select a TIFF file, and tells the TiffDocView to read in and display the image.

Other Peculiarities

Here is one item worthy of note in the drawSelf:: method for the Tiff DocView. Although three NXrects can be passed in to the drawSelf method, only one NXrect can be changed during a scrolling operation, so we only composite the first one into the TiffDocView. The ScrollView that contains the TiffDocView handles redrawing after other screen manipulations that obscure a corner of the TiffDocView (and thus require three NXRects to redraw).

Note also that the rectangle passed to drawSelf:: is actually the area of the TiffDocView that needs to be updated. It's only because we're using the very same coordinate system in the NXImage and the TiffDocView that we can use the same NXRect for the compositing operation. If we had changed either coordinate system, we would have had to compute the appropriate NXRect to composite into the TiffDocView.

Topics Of Interest

This is intended to be a simple, very focused example, and there are certain features not included that would be easy to add.

In particular, the code in this example will work with EPS as well as TIFF files. The only thing that prevents this is the "types" argument in the runModalForDirectory::: call in ImageReader.m. This prevents you from opening anything but TIFF files.

If you add "eps" to the types string, you'll be able to scroll EPS images, too. Go ahead - try it. You'll notice that when an EPS file is loaded on top of a TIFF image, the TIFF remains visible as a background for the EPS image. In order to view an EPS image without this background, you would have to clear the TiffDocView first. This excercise has been left to the reader.
(Hint: check out PSsetgray and NXRectFill.)

It would also be useful to provide for the creation of new windows, and permit existing ones to be closed. It is not difficult to provide this feature, but an explanation would be a distraction from the primary focus of this example.

Files

ImageReader.[hm]	Files for the ImageReader class.
TiffDocView.[hm]	Files for the TiffDocView class.
ScrollTiff.nib	File containing the user interface configuration and connections.
ScrollTiff_main.m, IB.proj, Makefile, ScrollTiff.iconheader	Created by Interface Builder.

Other References

Please refer to the class description of ScollView for a thorough explanation of the relationship between a ScrollView and its DocView.

Not valid for 1.0
Valid for 2.0