ImagePortfolio

by Martin D. Flynn, NeXT Computer, Inc. (*martin_flynn@next.com*)

Overview

ImagePortfolio is designed to provide visual management of your TIFF and EPS files. Multiple TIFF and EPS files may be dragged into a portfolio window for easy viewing. Each image will maintain the same aspect ratio, but will be scaled down to fit the confines of the image cell so that many images can be viewed simultaneously. The displayed images may then be selected and cut, pasted, or dragged into other applications accepting TIFF and/or EPS files. Additionally, the list of file paths which comprise the contents of a portfolio may also be saved to a file for later retrieval.

Note: *ImagePortfolio* also provides internal support for viewing most GIF files.

Permission is granted to freely redistribute the *ImagePortfolio* program and source code, and to use fragments of this code in your own applications if you find them to be useful. The *ImagePortfolio* application and code come with no warranty of any kind, and the user assumes all responsibility for its use.

Operation

Creating a new portfolio window:

A new portfolio window may be created by selecting the 'New Portfolio...' option on the 'Files' submenu.

Opening TIFF and EPS files:

There are several ways to display TIFF or EPS files in a portfolio window.

- First, multiple TIFF and EPS files, or directories containing TIFF and EPS files, may be selected and dragged from the

Workspace Manager and dropped into a portfolio window. Directories will be searched recursively to find all image files. If the list of files dragged into the window include a previously saved *ImagePortfolio* file (with extension '.portfolio'), then the list of images specified in the file will also be added to portfolio window (the *ImagePortfolio* file must be one of the actual selected files in order for its contents to be added. If it is found within a selected directory, its contents will not be added). This method of added images to a portfolio window specifically looks for files with extension ".tiff", ".eps", ".ps", ".icon", or ".gif", and ignores all others. To cause all files to be examined for image file content, press and hold the 'Control' shift while dropping the list of files into the portfolio window. This will cause the search process to execute more slowly, but all possible images will be added to the window (except for GIF image files, which still require the ".gif" extension).

- Second, TIFF images can be cut/copied from other applications and pasted into a portfolio window. When doing so, a save panel window will be displayed which will allow you to save the image to a file on disk. When saved, the TIFF file will be added to the portfolio window. This can be useful for such things as making a file copy of an image in an Interface Builder nib file by copying the image to the pasteboard, then pasting it to a portfolio window.
- Third, a previously saved *ImagePortfolio* file (with extension '.portfolio') can be opened from the File submenu and a new

portfolio window with the images specified in the file will be created. Only images which still exist and are readable will be added to the window, all others will be ignored.

- Fourth, A group of image files can be dragged from the Workspace Manager and dropped on the *ImagePortfolio* application icon (*ImagePortfolio* must be running at the time). As the mouse cursor enters the application icon, the arrow cursor will change to a copy cursor. The mouse button may then be released. A new portfolio window will be openned with the images specified.

If multiple images are being opened, the process may be stopped at any time by pressing the 'Stop' button. The portfolio window will then contain only the images which were added up to that point in time.

Cutting/Copying/Dragging images for export to other applications:

Again there are several ways to copy images from a portfolio window into other applications. All methods include selecting the images you wish to copy by clicking and dragging across images with the mouse. Multiple images may also be selected

by holding down the shift-key while making your selection.

- First, the icon in the upper left corner of the window can be grabbed with the mouse and dragged into any application, including the Workspace Manager, that accepts image files.
- Second, holding the alternate-shift key down and grabbing an image from within one of the image cells will allow you to drag that image into another application that accepts image files.
- Third, the selected images can be copied to the pasteboard with the Command-Copy key (or Command-Cut key) and copied to another portfolio window, or into any application accepting TIFF images. If copying into another application accepting TIFF images, then only the first selected image will be available to be pasted into the application.

Showing the non-scaled image:

The images shown in a portfolio window have been scaled down to fit the bounds of the image cell. To see what the image looks like at its regular size, just double click on the desired image in the portfolio window. The selected image will appear at its normal size in its own window. The best representation for the device (color, mono) will be displayed with alpha. To

toggle through all representations of the image with alpha turned off, just hold down the Alternate-Shift key while clicking on the expanded image. To bring back the best representation shown with alpha, click on the image without the Alternate-Shift key pressed.

Resizing the reduced scale images:

Grabbing and dragging the resize bar with the mouse will change the number of images displayed in a window. To change the size of the scaled images without changing the number of images displayed in the window, grab and drag the resize bar with the 'Alternate' key pressed. When the mouse is released, the images will be rescaled to fit inside the new size window. Note that the reduced image sizes cannot be changed while images are being loaded into the window.

Changing the image title font:

The image title font can be changed by using the 'Font' submenu.

Saving the displayed list of TIFF/EPS images:

The list of file paths which comprise the contents of a portfolio may also be saved to a file for later retrieval. To do so, either select the 'Save' (command-s) or 'Save As...' (command-S) option on the File menu. The list of image file paths, displayed image cell size, and current number of viewed rows and columns, will be saved to the file.

Sorting the displayed images by name:

The images shown in a portfolio window can be sorted by name by selecting the 'Sort by Name' option on the Edit menu. Additional images added to the portfolio window will be appended to the end of the list and not automatically resorted. To resort after additional images have been added, reselect the 'Sort by Name' menu option. A Shellsort method is used to sort the images.

Saving preference defaults:

The image cell size, current number of viewed rows and columns, and the current font for a selected portfolio window can be

saved to the defaults table by displaying the 'Preferences...' panel and selecting 'Set Default'. Subsequent new portfolio windows created will have the last saved default attributes.

Technical Documentation: The Application Classes

The source header files for each of the following classes contain more detailed information.

ImagePortfolio

This is the main Application subclass object. It handles the application initialization, main menu functions, and window registration to allow files dragged from the Workspace Manager to be dropped into registered windows.

Portfolio

This class handles the delegate methods for a portfolio window. It also handles passing a list of image files to the Workspace Manager via the View instance method '`dragFile:fromRect:slideBack:event:`' so that the user can drag

these files to another application which accepts files.

PalletMatrix

This is a subclass of the Matrix class which controls the selection, cutting, pasting, deleting, resizing, and sorting of the image cells that is contains. This class is the first responder for the portfolio window.

PalletCell

This is a subclass of ActionCell which holds information on a single image, including the file path to the image and how it is to be displayed in the matrix.

GifImageRep

This is a subclass of NXCustomImageRep which provides internal support for decompressing and viewing GIF file images. The image is displayed by converting it into a bitmap, then displaying it with NXImageBitmap().

ExpandedView

This is a View subclass which is used by instances of the Portfolio class to composite a large, unscaled, image into a window of its own.

FileUtils

This class contains only factory methods which provide some utilities for manipulating file names and directories.

objectThreadPerform

This is a category of the Object class that provides thread support to all objects. Using '`forkPerform:detach:`', a thread can be created to perform some action in the background while the window manager (main thread) is free to handle events. It is used in ImagePortfolio to create a background thread that reads image files to be placed in the portfolio window, while allowing the user to scroll through the existing images. In addition to creating new threads, support is provided to allow a

forked thread to communicate with the main thread to tell it to execute actions, such as drawing in a view, and even returning values. This is accomplished through calls to '`mainThreadPerform:with:wait:`'. This method can be called from any thread, making sure that the main thread is the only thread that will process the specified action. These methods are object independent, meaning that '`mainThreadPerform:wait:`' can be sent to **any** object from **any** thread. For instance, a forked thread could set the title of some button with the following:

```
[someButtonId mainThreadPerform:@selector(setTitleNoCopy:) with:aStaticTitle wait:NO];
```

A forked thread should not try to set the title of a button itself, or execute any other method that causes drawing to occur. This is because it has no way of knowing what drawing may already be occurring, and it may in fact cause drawing to occur in a view other than where it was originally intended. The return value can also be obtained from the method executed from the main thread by setting '`wait:YES`'. For example, to wait for a return value from a method executed from the main thread you could issue the following message:

```
rtn = [myObject mainThreadPerform:@selector(showErrorPanel:) with:errorMsg wait:YES];
```

The value that '`showErrorPanel:`' returned in the main thread would be returned to the calling thread. This is helpful when a thread wishes to display an error panel, then act on the response from the user.