Graphics
W o r k s h o p

by Alex Raftis

Overview

This is a nifty little program for reading and writing files in various picture formats. It will also let to affet and play around with the images a bit while you've got them loaded. Before you get bored with the menus (since you've

probably figured them out already, you might want to skip down to *Programmers Stuff*.

The Menus

Currently, here's what works:

Info

Info Panel¼

Displays informtion regarding me, the programmer.

Help¼

Displays this panel, to tell you what it does, assuming you haven't

figured it out for yourself yet.

Document

Open¼

Opens a new picture document, in any of the supported converter types. The current list is TIFF, PGM, PBM, PPM, XBM, and GIF. It accomplishes this daunting task via a message to the NXBitmapImage-RepControl of *(int)runOpenPanel: (id)openPanel*.

Save

Saves the current picture. It should never call up the save panel, since you can never really make a "new" document. It's really kind of pointless at the moment, since there are no image transformations you

can run on the object.

Save As¼
This calls forth the Save Panel via the message - *(int)runSavePanel: (id)savePanel withFilename: (const char *)filename* in the NXBitmap-ImageRepControl object.

Windows
Everything under this panel run as expected, like every other Windows menu in every other NeXTStep application.

Tools
Data¼

This displays information about the current image, such as width, height, number of colors, etc< It also displays the current image type in english.

Print¼
Prints out the current view. This is currently very crude printing, so the image may display cropped or otherwise not all there. Accomplished via the print message to the current view.

Hide
Hides the application or cause total world devastation, depending on your security clearance.

Quit
Quits the application.

Programmer's Stuff

So you like what this object does and you want to use it? Well, this will tell you how, but first a few brief words about the source. The main documention is in the header files (where about the only comments appear). They will tell you about what a method expects, however, this will tell you how they relate to each other. The final release will have a much more comprehensive documentation release.

Overview

You might also want to know what this program does, I suppose. Well, it's sort of a rapper that fits around an NXBitmapImageRep, thus the object name NXBitmapImageRepControl. It's purpose is to make the NXBitmapImageRep a more useful object by supporting more types of graphics formats. It does this through calls to *objc_loadModules()* and *objc_unloadModules()* calls (NeXStep Refernce, Volume 2, page 3-157). So, when you request it to load a module, it looks out on disk for a converter (call thus since it converts the format to the internal NeXT format and back again). It reverses the process when saving and image.

Converters

Now, what's in a converter? A converter is basicially an unlinked object module, created with a *cc -c <filename> -o <format.extension>.bcvt* system

call. The object is of type converter and all object of this type must respond to certain messages in a predescribed way. Meeting these specifications fully will allow programs to easily access many types of bitmap formats.

The converters are stored in library folders about the file system. They will be using in the following order: ~/Library/Converters is first, followed by a converter in /LocalLibrary/Converters, which is followed by /NextLibrary/Converters. All recognized converters should have the *.bcvt* suffix.

Image Manipulation Tools

Also, since many of these converters need a standard set of routines and utilities for manipulating images, they may request an object to do this of their

sender, usually the NXBitmapImageRepControl object. This object is also stored on disk in a library folder under the name of *Image.tools*. The object contains a list a standard functions for converting bitmaps from color to black and white to getting individual pixels from an image. Because this program is linked at run time, it may be modified at a later date to improve the functionality of the routines and it's algorhythms. This can be especially important, since this is the prime place something might cause a crash.

Of course, these tools are availble to any application programmer.

The Steps to Using It

Now that you know what it uses, here's how to use it. For examples of any of these callings, see the source to *GraphicsWorkshop*.

1. Initialize the object NXBitmapImageRepControl object. This will cause the program to search for converters, as while as link in the ImageControl object for later instantiations. It should only ever be called once.
2. When ready, create an open panel, but to run it, call the *runOpenPanel* method. This will show the open panel to all file types found in Step 1. It will return the same values as the Open Panels, run open panel method. See NeXTStep Reference, Volume1.
3. Once you have from 1 to *n* filenames, pass these names, one at a time, to the *openAndReturnImage* method. This will return an NXBitmap-ImageRep for each image.
4. Manipulate the image any way you'd like.

5. Now that you're ready to save, create a save panel, but run it via the *runSavePanel* method. This will bring up a panel, with the images current type linked in. Should the user select a new type, that converter will be linked in, it's specialization panel poped up, and it's file extension added to the bitmaps filename. If it returns YES, then you can move onto step 6, otherwise, the user canceled the save.
6. Call the *saveImage* method to save the bitmap. This will link the correct converter, if necessary, and call it to save the image.
7. That's it.

Some Pointers

- You can call the routines to link converter maually. This is useful when needed to bi-pass the open and save panels.

- File extensions are *very* important. This is how the object can tell what kind of converter is being used. May sure when passing names, that the filename always has a valid extension, otherwise thing won't work nicely for the user.
- When running a save panel, the linked converter's specializeation parameters only last as long as the next save panel.
- If you'd like to open multiple images in a file, you'll have to link converters yourself and message them via the openMulitple and saveMultiple calls.
- As long as you don't use open an save panals, this object should be able to run without the window server.

Points on Writing Converters

- You only have to implement 1 routine, however, you must respond to all

messages. You only need to implement getFormatName.

- It's really kind of pointless to not implement Reading and Writing of single images to streams, however, should you wish to not do this, simply make sure to always return NO for the one not implemented.
- If you don't wish to implement the "all" routines, also return NO, however, it is preferable that this return a single image inside and NXImage object, if the format only supports one image per file. If it supports multiple images, this should definitely be implemented (of course, it's not for gif :-( ).
- You only need to implement customSaveView if you'd like, but always return nil when you don't. The width parameter is the max width of the save panel. This helps when laying things out.
- Init and free are always called, however, they don't need to do anything

unless you want them to.

Bugs and Quirks

- The convert24toPalette method should implement a Meduim Cut algorhythm, but doesn't. The algorhythm used will work well for larger palettes of high contrast colors. This may be fixed in the future (not necessarily by me.)
- Many of the converters (all but tiff) don't even attempt a response to readAll and writeAll.
- CYMK color handling capabilities are near nil. You can get the pixels (possibly not when under messed configuration), and that's about it. Neither convert to BW or RGB currently works.

Fixes as of October 10, 1991

- PNM converters not handle comments. Oh-Ah.
- The GIF converter now handles truncated files. The problem was with filling the blank pixels via a call to memset and not always having 8 bit pixels. This was fixed by explicitly filling the extra pixels with the background color.
- Conversions to 1 bit now work all the time. (Well, most of the time. Something obscure still happens on odd cases [I think only 1 in 8.]) There was a mistake in the manual about internal formatted that led me to believe that 1 bit per sample images would not be padded at the end of the line. This is not so, and I was misled because my original test

image's width was divisible by 8.

- Previously unknown fixes to ImageControl object with messed RGB values. It seems that I was getting the R and B values swapped. This has been fixed for RGB, I can't swear what'll happen with CMYK. I think you'll get C <> K and M <> Y, but I have no images to test with.
- Behavior of Save panels corrected.
- TIFF saving under JPEG will now drop to LZW when appropiate, ie, with images of less than 4 bits per sample. (Conserves space and prevents exceptions to the JPEG routines.)
- Added Convert To menu items. Currently, only conversion to BW and 12 Bit Color are implemented. I hope to fix the rest soon. Note that the conversion of less < 4 bits per sample are a little flakly in that a value of 3 expands to 12, not 15 like it should. I'm working on it.

- Added pretty colors to the title in the help pages. BTW, you can only do this in edit by hand. Yech.

Final Notes

I can be reached at the following email address:

alex@data.ACS.CalPoly.EDU	For NeXTMail
alex@cosmos.ACS.CalPoly.EDU	For standard email

Addional Copyrights

Gif loading and Saving-