 Version 1.0.

Introduction to the MiscFindPanel

This document contains the following major sections:

- Incorporating the find panel into a project
- Other notes on the find panel bundle
- The SearchableText categories and string-searching functions
- Frequently Asked Questions about the MiscFindPanel

The MiscFindPanel class is designed to be a (mostly) self-contained bundle which can be incorporated into a project with minimal effort. Additional categories, protocols, and functions support the bundle and add powerful features of their own. This product requires NeXTSTEP 3.x

or higher. It was written by Christopher J. Kane (kane@gac.edu). Please feel free to contact the author with any questions, comments, bug reports (and/or fixes!), or suggestions about this bundle or the related facilities.

Additional documentation of interest is located (under the MiscKit/Documentation directory of the MiscKit distribution) in:

Classes/MiscFindPanel.rtf	Class documentation
Classes/MiscFindPanelClass.rtf	Application class category
Protocols/SearchableText.rtfd	String-searching object protocol
Categories/MiscSearchText.rtf	Text class category
Functions/MiscTBMK.rtf	Fast string-searching functions

Incorporating the find panel into a project

Barring the need to debug the MiscFindPanel class itself (and hopefully you won't have to do that),

the find panel bundle can be compiled once and installed into a project directory and (for the most part) forgotten. The most difficult part of the process of adding a find panel to your project is implementing the methods of the SearchableText protocol.

In this version of the MiscFindPanel class, the find panel searches for a "first conformer" (see the document **MiscFindPanel.rtf**) in the same places a responder is searched for when an action message destined for the first responder is sent (for instance, the "cut:" message from the Edit/Cut menu item). The objects in the key and main windows' responder chains, those windows' delegates, and the NXApp object and its delegate are the objects that, potentially, the find panel might operate on. This means that, while the SearchableText protocol could be implemented by any object, the only objects that the find panel might try to operate on are Responders of some type: views in a window and windows themselves typically (the exceptions being the key and main windows' delegates and NXApp's delegate, which may be objects of any class). Most commonly, it is a View of some sort in the main window. For most projects, this is not a problem; the intent in adding a find panel to the application is to operate on a view in the main window.

There are five simple steps to adding a find panel to your project:

1. Implement the SearchableText protocol in one of your objects.

2. Build the *MiscFindPanel.bundle* and install it in your project's directory. Either:
   a. On the command line: "make install INSTALLDIR=*path*" within the *MiscFindPanel* directory, or
   b. In Project Builder: load the file *MiscFindPanel/PB.project* and type "install INSTALLDIR=*path*" in the argument text field before building.

   Replace *path* with the path to your project's directory.

3. Copy or link the files *MiscFindPanelClass.h, MiscFindPanelClass.m, MiscFindPanel.h,* and *SearchableText.h* into your project's directory.

4. Add the new files to your project:
   a. Add *MiscFindPanel.bundle* to "Other Resources".
   b. Add *MiscFindPanelClass.m* to "Other Sources".
   c. Add *MiscFindPanelClass.h, MiscFindPanel.h,* and *SearchableText.h* to "Headers".

5. In Interface Builder, add a Find menu to the Edit menu of the main menu of your application,

and enable each of the menu cells (unless you plan on enabling them programmatically). Using the class inspector, add the methods (findNext:, findPrevious:, enterSelection:, jumpToSelection:, orderFrontFindPanel:) to the FirstResponder class. Make one connection from each of the Find menu cells to the First Responder object in the File Window, connecting the appropriate method.

Your application must be linked with the shared library *libNeXT_s*. It probably already is. If you want to strip the main executable of your application, or the install process is going to strip it for you, you need to add the line

APP_STRIP_OPTS = $(DYLD_APP_STRIP_OPTS)

to the file *Makefile.postamble* (creating the file if it doesn't exist). This will cause those symbols which might be needed by the dynamically loaded MiscFindPanel class to not be removed from the main executable. See the manual pages *rld(3)* and *strip(1)* if you want to know more.

The MiscFindPanel class can also be statically linked into an application, and the localized files added to others in a project's language directories. In this case, the MiscFindPanelClass category could still be used to "catch" messages and forward them to the find panel, but some modifications

would have to be made so that the code does not try to dynamically load the MiscFindPanel class. The MiscFindPanel code itself should work without modification. In this case, of course, an application could be "fully" stripped.

Other notes on the find panel bundle

Localization

The *MiscFindPanel.bundle* contains translated panels and strings for the languages: English, French, German, Italian, Spanish, and Swedish. If you are not supporting some or all of these languages in your project, you will want to remove the appropriate .lproj directory(s) in the *MiscFindPanel.bundle*, so that if, for instance, your application does not support Spanish, a user will never see a Spanish find panel and the rest of your application in some other language.

If you are supporting more than one language in your application, you will want the Find menu items to be localized as well. The following table shows possible translations for the Find menu

items. Copying and pasting the text out of this document may be the easiest way to get the characters with the diacriticals.

English	Find	Find Panel...	Find Next	Find Previous	Enter Selection	Jump to Selection	
French	Rechercher	Panneau de recherche...		Rechercher le suivant	Rechercher le prÝcÝdent	Entrer la sÝlection	Aller Õ la sÝlection
German	Suchen	Dialogfenster "Suchen"		Weitersuchen (vorwÙrts)	Weitersuchen (röckwÙrts)	Auswahl übernehmen	Zur Auswahl springen
Italian	Trova	Pannello di ricerca...	Trova il seguente	Trova il precedente	Riporta la selezione	Salta alla selezione	
Spanish	Buscar	Panel de bósqueda...	Buscar siguiente	Buscar anterior	Introducir selecciín	Pasar a selecciín	
Swedish	Sðk	Sðkpanel...	Sðk nÙsta	Sðk fðregÚende	Kopiera markering	GÚ till markering	

Known bugs

· The MiscFindPanel class's internal _calcFindPanelTarget instance method assumes that the responder chains of the key and main windows end with some responder that has a next responder of nil (this "end of the chain" responder need not be the windows themselves, though it is them if the windows' next responders have not been explicitly set). This also means that a responder chain that loops is also a bad thing. Under "normal" usage, this bug will not manifest

itself.

Possible future enhancements

· The ability to add one or more accessory views
· Fix first conformer searching so that possible loops are avoided
· Implement a generic SearchMatrix category (unless someone beats me to it)
· Add message support for the different SearchErr values of SearchableText

Miscellanea

Help is not provided for the find panel and its controls. Under NeXTSTEP 3.x, in my opinion, help text from a bundle is not integrated well into the NXHelpPanel, so I've opted not to provide any. Also, help is much more application-specific than the find panel itself, and I also did not want to translate the help text I would have provided into the non-English languages. Help text similar to that contained in Edit.app for its find panel is suitable for this one as well.

The find panel does not save its frame to the defaults database. Simply use the appropriate

Window methods if you want this behavior, sending them to the find panel.

The SearchableText categories and string-searching functions

To ease the incorporation of the find panel into a project, two ready-to-use string-searching packages (MiscTBMK.[ch], regexpr.[ch]) and an implementation of the SearchableText protocol for the Text class (MiscSearchText.[hm]) are included with the MiscFindPanel distribution. See the documents **Categories/MiscSearchText.rtf**, **Functions/MiscTBMK.rtf** for additional information.

The regular expression routines

The regular expression package is by Tatu Ylonen (ylo@ngs.fi), and is compatible with the GNU regexpr package (at the time of this package's writing). This is the version posted to the comp.sources.misc newsgroup, v27i023, with the patch posted to the same newsgroup, v29i059. In addition, some changes, marked with "(cjk)" in the source, have been made to accomodate the

use of these routines under NEXTSTEP.

The copyright and license notice for the regular expression code:

> Copyright (c) 1991 Tatu Ylonen, Espoo, Finland
>
> Permission to use, copy, modify, distribute, and sell this software and its documentation is hereby granted without fee, provided that the above copyright notice appears in all source code copies, the name of Tatu Ylonen is not used to advertise products containing this software or a derivation thereof, and all modified versions are clearly marked as such.
>
> This software is provided "as is" without express or implied warranty.

The header text to the two comp.sources.misc newgroup postings that contained this package:

> Submitted-by: ylo@ngs.fi (Tatu Ylonen)
> Posting-number: Volume 27, Issue 23

Archive-name: regexpr/part01

Regexpr is a regular expression package. It is free (meaning that you may do anything you want with it); the original motivation for writing it was not being able to use the GNU library in a commercial application.

Some of the features include:
- fully compatible with gnu regex library (I run emacs with this library for several weeks as a test)
- can handle arbitrary data, including binary characters
- can handle split data
- compiles and runs also on 16 bit machines (eg. MSDOS)
- does not use alloca
- fairly easy to extend and modify (easier than the gnu version anyway)
- speed comparable to that of the GNU library (searches seem a bit faster, matches about the same and compiling a bit slower than in the gnu library)
- there are some extensions (enabled if RE_ANSI_HEX is set in syntax):

\vnn	for accessing registers > 9 (useful if RE_NREGS > 10)
\xhh	specifies character in hex
\a	ascii 7
\b	ascii 8
\f	ascii 12
\n	ascii 10
\r	ascii 13
\t	ascii 9
\v	ascii 11

I have not written any documentation; see the header file and documentation GNU Regex library in GNU Emacs distribution.

Send comments, bug fixes and suggestions to Tatu Ylonen <ylo@cs.hut.fi>.

Submitted-by: ylo@ngs.fi (Tatu Ylonen)
Posting-number: Volume 29, Issue 59

Archive-name: regexpr/patch01
Patch-To: regexpr: Volume 27, Issue 23

This patch contains the following changes to the regexpr module.

- Matching agains registers (the \1 construct) did not work properly (the fastmap was computed incorrectly).
- Assert bounds in re_search_2 were too loose, and have been changed to reflect the actual behaviour.
- The copyright notice has been clarified, but no significant changes have been made.

The only real bug reported so far has been the problem with matching against registers, and it is now fixed.

The patch is a context diff inside a shar.

Tatu Ylonen <ylo@cs.hut.fi>

Frequently Asked Questions about the MiscFindPanel

This section presents a number of often-asked questions and comments about MiscFindPanel, and the related categories and files, and my answers or responses to them. (And a few questions and answers I made up myself.)

Thanks especially to Scott Anguish (sanguish@digifix.com), Charles C. Lloyd (clloyd@gleap.sccsi.com), and Don Yacktman (Don_Yacktman@byu.edu) for asking or inspiring many of these questions.

Topics:

6. Those translations
7. Objects other than Text

Question 1: The "non-standard" interface
The find panel offers all the standard features, but doesn't present the "standard" interface found in Edit. Why is this? Did you consider making the MiscFindPanel an exact clone of the find panel in Edit?

That's how it began, of course. But I made a few changes:

1. The Find TextField and Replace With TextField are not lined up as in a Form. I didn't see any good reason why this had to be, and maximizing the "area to type in" seemed more important.

2. The position of the message text in the Edit find panel seemed to be arbitraryÐthere was no reason to prefer it being on the left. But there *did* seem to be a good reason to move the Replace All Scope radio buttons to the left: it associates the matrix moreso with the Replace All button, and also reduces the amount of mouse movement required after acting in the matrix. Myself, the only reason I operate on those radio buttons is that I am immediately going to Replace All. I very briefly looked at having the message text in the center, but that was not aesthetically pleasing, and

harder for the eye to pick out somehow.

3. The Previous and Next buttons on the German version of the Edit panel are translated awfully (well, the translations mangle the panel). I think my translations are sufficient and make the panel look better.

The biggest (and most often commented on) change is #2, above. I don't think that the changes are significant enough to cause confusion. I think that in the few-hundredths-of-a-second period, as the user's eyes track horizontally across the panel, the eye will not be tracking for an empty area of the background color, but rather for color changes from that of the background: the square shape of the boxes, the black text, or most likely (in my opinion), the white dot in the Replace All radio matrix.

So it seems to matter less that the Replace All matrix and friends have been left rotated one place, than that the Replace All matrix has a similar look to the one in Edit's find panel. And if a developer *really* wants the look of the interface that Edit has supposedly "standardized", she or he can simply modify the .nib files (though they are not as simple as they appear).

I used this same design in an application I wrote (Schematik.app, a NeXT front-end to MIT

Scheme) two years ago, and as of now, two years and thousands of users later, there has never been a complaint, nor even a comment, about the "non-standard" find panel user interface.

People are highly adaptable creatures. How many thousands of people use NewsGrazer (as an example, sorry Jayson), with its non-standard command-key usage, every day and live through the experience?

Question 2: Having other objects do the searching
Why not integrate the regex and searching functionality into the MiscFindPanel?

Another NeXT application developer and I, as it happens, discussed this at length. He thought the MiscFindPanel should also have searching functionality. I disagreed (and implemented it without that functionality). There were two reasons I had:

1. I wanted a generic sort of panel. I didn't want to complicate the developer's job if the developer had a different regex package she or he wanted to use (for instance). This is the reason for having a SearchableText protocol, too. I didn't want to leave developers out in the cold however, so I included a good set of regex routines.

2. I wanted to follow the general NeXT paradigm for panels; and a more general object-oriented one. Does the SavePanel actually save the file? Does the PrintPanel actually do the printing? (Well, no the printer does, but in software...) Panels are generally vehicles that users use to have actions performed, but which don't actually do them. (NXHelpPanel is something of a special case.) Thus the MiscFindPanel is a vehicle for the user to specify searching parameters, but does not do the searching. A Text object (for instance) is best equipped to search the text itself.

Question 3: The searching code is not in the bundle
Does the regex library get put into the bundle automatically? What about putting the regex code in the bundle too?

This relates to question #2. There is no point in including the regex code with the MiscFindPanel object file, since the MiscFindPanel doesn't use it. The Misc_TBMK searching routines and the SearchText category do not go into the bundle for the same reason.

This non-MiscFindPanel.bundle stuff is provided as a convenience to the developer, to make the panel a bit easier to use. But the decision to use it or not remains with the developer; it is not

gratuitously included in the bundle. And it's easy enough for a developer to add the files to a project and have them linked into the proper place.

Question 4: SearchableText rather than MiscSearchableText
I was wondering why you hadn't changed the SearchableText to have the "Misc" prefix that nearly everything else has....

There were a few reasons...

1. I didn't like the sound of "MiscSearchableText".
2. Protocol names should be descriptive (I think) of some property of itself that an object wants to advertise. The "Misc", being an almost-word in itself, seemed to act as an adverb to "Searchable", and seemed to distorted the meaning of the name.
3. Protocol names only conflict with other protocol names. From the NEXTSTEP 3.x developer documentation (NextDev/Concepts/ObjectiveC/3_MoreObjC/MoreObjC.rftd):

> *Unlike class names, protocol names don't have global visibility. They live in their own name space.*

Only class names, function names, non-static global variables, and #defines and typedefs in header files need to have the "Misc" prefix to avoid the majority of name clashes. Protocol and category name clashes are still possible, but much more unlikely.

Question 5: Separate bundle rather than .bproj
Why is the MiscFindPanel not in a .bproj?

Well, I don't know. Probably because I didn't originally implement it that way. I've used subprojects in the past, and have never been impressed. I just wasn't inclined in that direction.

You shouldn't need to compile the MiscFindPanel.bundle more than once (per project at least). Subprojects are useful if you are doing development on "subsystems".

Question 6: Those translations
Some of your translations don't seem to be correct...

Quite possibly. Most of my translations are translations taken from NS 3.0/3.1, others were gotten from semi-informed opinions, and the rest are "best guesses". NeXT may not have used the

best translations, but it seems likely that they used the most *suitable* one (the length of a translation is one consideration). And the others are probably comprehensible, even with a mis-conjugated verb or mis-pluralized noun here and there.

I am **very** willing to listen to translation suggestions, and/or to work with someone on translating the MiscFindPanel to another language. Please e-mail me.

Question 7: Objects other than Text
How about a find panel that works with NXBrowsers or in a Matrix. Will MiscFindPanel handle this too?

Yes, sort of. An object must conform to the SearchableText protocol, but if it does, the MiscFindPanel will work with it. I created the SearchableText protocol with Matrixs, selection lists, and, of course, the Text class in mind. My goal was to create a general protocol which any object that had text it wanted to "vend" for searching or replacing operations could implement. For a Text-like object, this is simple. For a Matrix, a programmer would have to decide how to map a linear sort of text model to two/three dimensions (depending on your interpretation of the text in a

Matrix). This is not all that difficult either, and there is lots of documentation to explain things. If you are having trouble, feel free to e-mail me.

I'd be interested in implementations of the SearchableText protocol for NeXT's *Kit classes, as well as others, to be included (with copyright notices and documentation, as appropriate) in future releases of the MiscFindPanel package. Please e-mail me if you would like to contribute.