

Mouser the Browser,

or

How I Learned to Stop Worrying and Love MacApp®

by M.Boetcher

July 31, 1989

Introduction

First off I have to level with you--you can't use Mouser to edit your source code. Yet. The second important thing to know is that before you can browse, you must run your source files through the parser, so Mouser can build an index file that remembers where all the classes and methods are located. I know you're disappointed that you can't just throw away MPW and spend the rest of your life blissfully programming with Mouser, but that day has not yet arrived.

On the other hand, I've got good news and bad news. The good news is that Mouser will parse both Object Pascal AND C++ source code. The bad news is that, due to my religious affiliations, the Pascal side has received most of the attention and is a lot more presentable. Not to worry! The C++ parser will be spiffed up Real Soon Now! Ha, ha!

Parse Those Files

The first thing you have to do before you can browse is parse your source files. Launch Mouser, and click on Cancel when the SFGGetFile box comes up (unless you've already done some parsing on the sly). Now, the menu bar looks like this:

 **File Edit Find Parser Browser Windows**

Go to the Parser menu and make sure the language you want is checked. (The default is Pascal, of course!) Now choose New Parse from the File menu. What you'll get is a custom SFGGetFile box that prompts you for the source files to parse. If your code uses MacApp, you probably have several files all in one folder--in this case you can leave the Parse All Files box checked. (If you're parsing MacApp itself, start with the PInterfaces folder.) If you want to parse some but not all of the files in a folder, uncheck the box. The check box prompt tells you which kind of files the parser will look for. If it's Pascal it will parse files with names that end in '.p'. If it's C++, it will look for filenames ending in '.h', '.c', and '.cp'.

Are you ready to parse? If so, click on the Parse button. If you're not ready to commit yourself, click on Cancel, deferring the joy of browsing to another time.

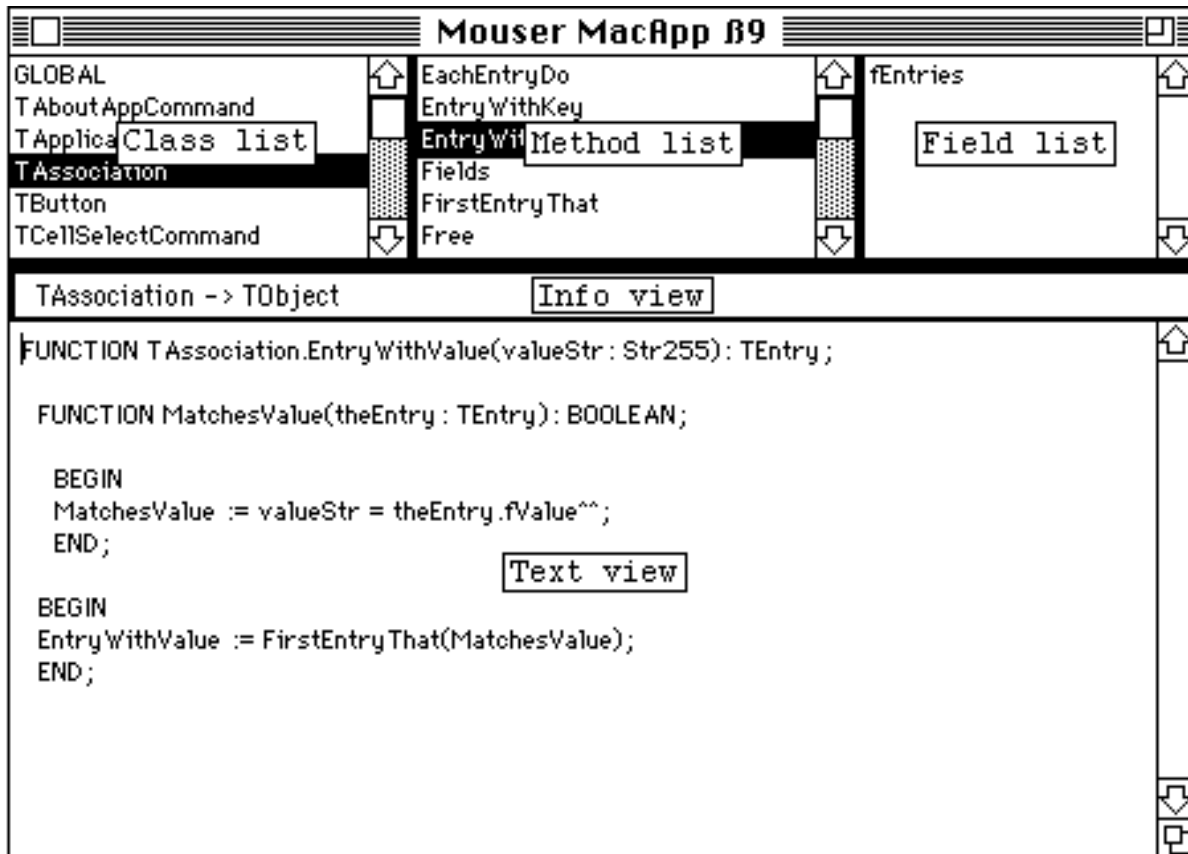
Mouser will start parsing away, and a dialog box will inform you of which file it is working on. When it's done chewing on the file(s) you gave it, the SFGGetFile box will re-appear. If you're parsing MacApp, go to the Libraries folder, click on the name of a text file, and Parse again. When you've parsed all the source files, both interface and implementation, click on Done to bring up the long-awaited browser window.

Browsing for Beginners

Before you get carried away, DON'T FORGET to SAVE YOUR FILE! You don't want to have to parse it all over again, do you?

Now, look at that window. You should see a list of names in the leftmost pane--these are the names of classes defined in your source files. They are in alphabetical order. If you defined any global (non-object) routines, externals, or inlines, they will be lumped together under the name GLOBAL (Pascal only). Click once on a class name. In the middle pane should appear a list of methods defined for that class, and if the class has any fields, they should show up in the right pane. The narrow box just under the three top panes ("info view") displays the class hierarchy for the selected item in the left pane.

To see the source code for a method or routine, click on its name in the middle pane. The bottom pane will fill with text. (Yes, I know the tabs don't look right; blame it on MacApp.) When you click on a name in the right pane, the field's definition will be displayed in the info view. But now the class hierarchy has disappeared! How to get it back? Go up to the Browser menu and choose Hierarchy of <ClassName>. Phew! It's back.



At this point there are so many things you can do that I'll just list the features of Mouser and briefly describe each one.

Our Feature Presentation

Here are some handy shortcuts:

- Option-click on a class name to create a new window containing the class definition.
- Option-click on a method or field name to create a new window containing its source code.

Note that when you create a new window by option-clicking, that window "belongs" to the browser document you created it from. When you close the document window all the subsidiary windows it owns will be closed as well. This is a Feature.

- If you hold down the Control key before mousing down in the scrollbar thumb, when you drag the thumb the view will scroll along with it. This is sort of an experiment in the wild, wacky, world of Macintosh User Interface. By the way, it doesn't work with the Scroll2 CDEF, those funky double scrollbars.

The Find menu

Items in the Find menu work on the current text view and the current method. "Current" means the topmost window. If the current window is a browser, the current method is the selected item in the middle pane. If the current window is a source code window, there may or may not be a current method, depending on the window's contents.

The first three menu items--**Find...**, **Find Same**, and **Find Selection**--are supposed to work pretty much like their MPW namesakes. Not only do they use the same command keys, but holding down the shift key lets you search backwards!

Search All Documents is a flag used by the Find Code and Find Inherited items. It lets you choose whether to search the current document only, or to search all open documents. That's right, boys and girls--you can have multiple documents open, such as MacApp and your application, and Mouser will automatically search the MacApp document for methods it can't find in your application's document.

Search For Entire Word is a flag used by the first three Find items as well as Find References. The default setting causes those searches to report hits only when the search string is surrounded by non-alphanumeric characters; e.g. searching for 'abc' in string 'xabc.abc-xyz' will highlight the second occurrence. When this item is unchecked, searches will report all hits of the target string.

Case Sensitive Search. Like the item above it, this item affects Find..., Find Same, Find Selection, and Find References. The default is off - case insensitive search. C++ users may want to turn this on.

Find Code for <selection> is enabled only if you have selected some text in the text view. This command tries to find a method or field with the selection name, searching the current class and its class hierarchy. If it finds a match, it opens a window containing the source code for that method or field. Mouser strips non-alphanumeric and trailing characters from the selection string before doing the search. For example, if you double-clicked on ICommand to get the following selection (thanks to TextEdit):

```
ICommand(itsCmdNumber, itsTEView.fDocument, itsTEView, NIL);
```

Mouser would strip off the left paren and everything following it, and search for a method named ICommand. When in doubt, check the menu.

If the string you are inquiring about is a component of a local variable, parameter or field, Mouser finds it anyway! Double-click on InsertEntry in the TDialogView.ParamTxt method:

```
fParamTxt.InsertEntry(keyStr, valueStr);
```

The menu item will read 'Find Code For fParamTxt.InsertEntry'. Mouser looks up the class of fParamTxt, goes to the TAssociation class, finds the source code for the InsertEntry method, and opens a new window displaying it.

Find Inherited <methodName> is enabled whenever there is a current method. This command goes up the class hierarchy of the current class, searching for a method with the same name. When and if it finds one, it creates a new source code window for that method. For instance, if the topmost window shows the code for the method TPopup.Draw, the menu item will read 'Find Inherited Draw' and will open a new window for TControl.Draw. If you then choose Find Inherited Draw again, you will get yet another window containing TView.Draw.

Find References to <target> is enabled whenever there is a current method. <target> is either the current text selection, or the current method. This command searches each method of each class in the current document for the target string. It creates a window containing the list of references in the top pane. You can stop the search by pressing Command-Period.

Once you have a References window, you can click on an item in the method list in the top pane, and the text view will show that method's source code. Somehow this all seems real familiar! At this point you can use commands from the Find menu to find selections, inherited methods, etc.

The Browser menu

Items in the Browser menu apply to the current browser document window.

The first two items, **Hierarchical Classes** and **Alphabetical Classes**, let you display the list of classes either alphabetically (the default) or hierarchically. Try it and see. If your application doesn't do a lot of subclassing, the hierarchical list won't look much different from the alphabetical list. The MacApp class list looks a lot more dramatic, especially the TView hierarchy.

Hierarchy of <ClassName> simply updates the class hierarchy display in the info view beneath the list panes.

All Methods of <ClassName> creates a window containing a list of all the methods you can use with the selected class. The list includes methods defined for the class itself and for all its superclasses. You can click on a method in the list to see the source code.

Implementations of <MethodName> creates a window containing a list of all the methods with the selected method name. For instance, choosing Implementations of DoIt in MacApp will find TAboutAppCommand.DoIt, TCellSelectCommand.DoIt, and so on and so forth. Like All Methods above, the list window will retrieve the source code when you click on a method in the list.

Source File of <MethodName> displays the name of the file which contains the source code for the current method. The file name is written to the one-line info view. Writers, this one's for you!

The Windows menu

This menu keeps track of all the windows you create. The browser (document) windows are always the first items in the menu. Below the divider are the rest of the windows, of which there are currently three types: method or field source code ('<ClassName>.<MethodName>'), class definition ('Class <ClassName>'), and references ('References to <blah>'). The current window is always marked with a check mark. You can activate a different window by choosing its name from the Windows menu, à la MPW.

Growing Panes

You may have noticed, as you waved your mouse about the browser window, that when the cursor passes over one of the thick lines dividing the panes it changes its appearance. That's because those thick lines mark areas for resizing the panes. You can grab a divider line and drag that sucker around. Have fun.

A Creditable Attempt

Mouser would not have been possible without the herculean efforts of world-class programmer, Bo3b Johnson. As many of you know, Bo3b wrote the original version, called Bowser, and cast his source code to the four programming winds so that others could make of it what they would. After changing the user interface and mucking about with the parser, architecture, and just about everything else, I decided to rename it to Mouser (which rhymes with "browser") in honor of my cats. Not that they've ever even seen a real mouse, mind you, but if they did, I'd like to think that their actions would bring credit to their species. Anyway, thanks Bo3b, I hope I spelled your name right.

The unit implementing view splitting was submitted to the world-famous MacApp recipe contest by Douglas Stein, of BBN Systems and Technologies Corporation. Thank you, Doug, for coding something I wouldn't want to do myself.

I'd also like to thank Steve Friedrich for offering encouragement, motivation, and subtle humor, and Dave Ramsey for technical support in the areas of cars, comix, computers, and cats.

Steve Burbeck said I had to put this in:

MacApp is a registered trademark of Apple Computer, Inc.! But you knew that already.

Anomalies, Conundrums and Malapropisms

As I mentioned in the introduction, the C++ parser is not as spiffy as the Object Pascal parser. Some of the things you'll notice are:

- If you're using `struct` instead of `class` to define your C++ classes, Mouser won't recognize them as classes. Them's the breaks.
- The C++ parser does not keep track of non-member or external functions. Maybe later.
- Sometimes the C++ parser gets confused. For instance, if it sees an instance variable that contains parentheses it thinks it's a function. It also dislikes seeing extra stuff after the right paren of a function's parameter list (like `const`). You may end up with some nonsensical items in the method list.

There may be other glitches or bugs associated with parsing C++ files. It hasn't been extensively tested. Y'all be sure and let me know about 'em, hear?

Known problems:

- The horizontal scroll bar in the browser window won't show its face no matter how much I threaten it. (And I call myself a MacApp engineer?)
- Double-clicking on text selects more than you want it to. This is basic TextEdit, folks.
- Tabs don't look right. MacApp should really do something about this, right, Steve?
- I can't resize the browser window panes as small (big) as I'd like. I'll look into it.
- If your fields don't begin with a lowercase "f", they'll get lumped in with the methods in the middle pane. This should be fixed in the future; it requires a parse file format change.

Symptoms and solutions:

If you get an empty browser after parsing, it means you didn't parse any files that had classes in them! You need to parse BOTH interface files and implementation files.

If you click on a method name and get the message 'Unable to blah blah because of a disk error', it probably means the method's source file is no longer in the folder it was in when you parsed it. The parse file remembers the exact path of each source file, and if they get moved around Mouser gets annoyed and starts giving out alarming error messages.

If you run out of memory while parsing, either increase the size of the Mouser application or try parsing the files one by one, doing the interface files first.

So, all you MacApp fans out there, send your cards, letters, and comments to me, Mary Boetcher, MailStop 22AE, AppleLink BOETCHER1. I'll try to brace myself for the flood.