Newest Additions - Editing Macros

Macros do what you'd expect. Kind of. This being the first version with macros of any sort, they're still pretty rudimentary. You can assign a string of text to be sent whenever you press specified key sequences. All key-sequences consist of cmd+option+<some other key>. To try it out, bring up the Macro Editor by selecting "Edit Macros..." from the Options menu.

Because Macros make use of straight IRC commands, you will have to utilize commands using the slash (/). If you're not familiar with commands such as /msg, /topic, /mode, etc.. you will need to do so before macros will become effective for you. To learn about commands in IRC, type at any time when connected, /help for a list of help commands, and more specifically, /help <commandname> for specific help on that. An example of the result of typing '/help topic' is as follows:

-help_US- *** Help on TOPIC
-help_US- Usage: TOPIC [<channel>] [<topic for channel>]
-help_US- Changes the topic for the named channel.
-help_US- You have to be on the channel to change the topic and if
-help_US- the channel mode is +t then only a CHAN-OP can change
-help_US- the topic.

If you are familiar with these commands, then hopefully, the Macro Editor dialog will be pretty self-explanatory. Click on the Add button and a new dialog box appears. This is where you create a macro. You assign it a key, you give it some sort of descriptive name, and you specify the text you'd like to have sent when you execute this macro. Here's various things you can do with macros:

1. Have a pure cheesy simple macro. Maybe you're on a channel where you laugh a lot. You could make a macro for cmd-opt-H whose text is "HAHAHAHAHAHAHAHA!!!!" and then instead of having to type all those grueling "HA"s, you can just hit cmd-opt-H instead. NOTE! Macros automatically send a carriage return at the end of the macro.

2. Combine a couple pure cheesy lines into a single macro. If you often create a channel, set some modes, and set a topic, a macro can automate this for you. Individual lines are separated by semicolons (;) in macro text. For example, you can create a macro:

"/join #lonely; /mode #lonely +tn; /topic #lonely Come talk to me! PLEASE! I beg you!"

This will join the channel, and assuming you're the first person to join #lonely, will also change the topic and change the modes for that channel, all using a single keystroke. If you weren't the first user on the channel #lonely, your macro would begin to fail at the second command of setting modes and the topic unless you are given op status.

3. Use Homer's macro substitution variables. Homer has 5 substitution variables, which, when found in macro text, will be replaced with other text. They are:

$topchannel - this will be changed to the name of the current topmost channel window
$lastjoiner - name of the last person to join the topmost channel
$lastmsger - name of the last person to send you a private message
$myname - your current nickname
$usertext - the current contents of the user input floating window.

Examples of use:

"Wow, look, I'm on $topchannel!" will send "Wow, look, I'm on #Macintosh!" (assuming that #macintosh is your topmost channel window.

"/kick $topchannel $lastjoiner" will kick the most recent person to join the top channel.

"/msg $lastmsger I don't wanna talk to you, jerky.; Wow, that $lastmsger guy sure is a jerk." will tell the last guy who messaged you that you don't wanna talk to him, and then send a line to the top channel saying he's a jerk.

"/msg #macintosh $usertext" - this will send whatever text I've typed into the user input floater to the #macintosh channel. Note that I DON'T hit return, I just press the macro key sequence, and it sends it and clears the user text area. This type of command is handy for sending text to channels that are buried beneath other channels (if you don't want to use cursor focusing).

That's it for now. I know they're limited, they'll probably get more powerful in the future, but at least they're a start for now.

A note to experienced IRC users

Some of you will no doubt try out Homer and find it lacking, because it hasn't implemented some slash command or the other. I started writing Homer because I thought the entire ircII command-line syntax was baroque and confusing, and was preventing a lot of people from experiencing IRC because they'd log on and not know what to do. I felt that the Macintosh could provide a clean interface to IRC without requiring any slash commands. I think I'm accomplishing that goal, slowly. It's taking me a while to determine what the appropriate GUI is for some features, and just takes me a long time to effectively implement others. My policy is that I don't implement slash commands for anything that can't already be done using Homer's Mac interface (there are a few exceptions, such as /trace, but they're relatively minor). Anyway, the point is, you won't find any handy-dandy quick reference list of slash commands, because Homer encourages you not to want them in the first place. If you try it out and find that it's just not for you, try out ircle, another Macintosh IRC client. Unlike Homer, ircle has taken a very traditional approach and users of the unix ircII client will feel right at home with it. It implements many more slash commands than does Homer, and also allows you to create .ircrc files, which Homer doesn't (again, because for the most part, there's no need). Good luck with whichever one you choose.