KeyMenu

Keyboard-Operated Menus for the Apple Macintosh

version 1.00

E. Jason Scheck
March 11, 1989

OVERVIEW

KeyMenu is an startup program *(INIT)* and Control Panel device *(cdev)* which allows arbitrary menu selections (not just those with keyboard equivalents) to be using only the keyboard. The manner in which *KeyMenu* operates is a superset to the method that is employed by *Microsoft Word 3.0*. *KeyMenu* requires System 4.2 or above, and the 128K or later ROMs. It will not load under lesser configurations.

KeyMenu is invoked when a user-defined key sequence is pressed. The default "hot-keys" are the same as those used in Microsoft Word 3.0 – the period key on the keypad, or Command-Tab. Once invoked, the arrow, number and character keys, as well as keys on the keypad, are used to select menu items. Unlike *Word*, hierarchial menus can also be selected and traversed. A corresponding Control Panel interface can be used to change the operating features.

INSTALLATION

KeyMenu is installed by simply dropping the *KeyMenu* file into the System Folder and rebooting. Its icon will be shown as it is installing at startup. If there was any problem in loading, or if the mouse button is held while starting up, an icon with a big X through it will be displayed.

USAGE

KeyMenu uses the keypad, as well as a few keys on the main keyboard for input. The keys that are used are described in the tables below. Keys that can only be pressed when they reside on the numeric keypad are shown in brackets, such as [4]. The actions mentioned in this section refer to the default settings only. Some characteristics may be changed by using the Control Panel interface (see below). The case of all alpha characters is insignificant. Any reference to "exiting *KeyMenu,"* refers to the present invocation only. Pressing the hot-key again will again activate *KeyMenu.* Rebooting (without the *KeyMenu* file in the System Folder, or while holding the mouse button down) is the only way to completely remove *KeyMenu*.

After invocation, the entire menu bar is hilited (*menu selection mode*), and the following keys may be used:

Key	Action
0 - 9	Select the *n*th menu from the left of the menu bar, where 0 is the leftmost menu (usually the Apple menu).
←	Select the leftmost menu in the menu bar.
→	Select the rightmost menu in the menu bar.
Command - Alpha Key (A-Z)	If present, selects the command equivalent of the given letter.
Alpha Key (A-Z)	Select the first menu from the left of the menu bar that starts with the given letter.
Clear or Cmd-Period	Exits this invocation of *KeyMenu.*

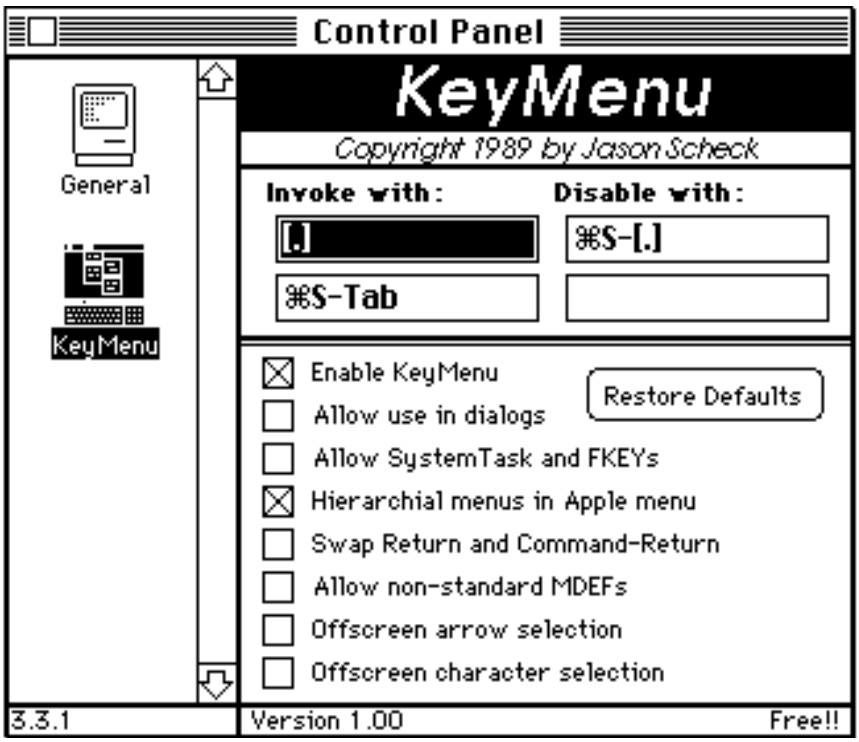
Once a menu has been selected (*item selection mode*), the following keys are active:

Key	Action
←, [4], Cmd-Tab, Shift-Tab	Selects the menu to the left of the current menu in the menu bar, wrapping around to the rightmost menu is the leftmost is current.
→, [6], *or* Tab	Selects the menu heading to the right of the current, wrapping to the left if necessary.
↑ or 8	Selects the next active menu item, wrapping around to the bottom, if necessary. If no item is presently selected, the bottommost visible menu item is selected.
↓ or 2	Selects the previous active menu item, wrapping, if necessary. If no item is selected, the first menu item is selected.
Cmd - Alpha Key (A-Z)	If present, selects the command equivalent of the given letter.
Alpha Key (A-Z)	Select the first menu item down from the current (wrapping if necessary) that starts with the given letter.
Return or Enter	If the current selection is a hierarchial menu item, steps into the hierarchial menu, making it the current menu. If no item is selected, *KeyMenu* exits. Otherwise, the selected item is chosen and returned to the application.
[.] *(Keypad period)*.	Returns to the menu selection mode.
Cmd-Period	Exits this invocation of *KeyMenu.*
Cmd-Return or Cmd-Enter	Returns the current selection to the application, regardless of whether it is a hierarchial menu item.
Clear	If the current menu is a hierarchial menu, steps into the previous menu. If not, exits *KeyMenu.*

If the mouse is pressed in the menu bar or in the current base menu, the regular MenuSelect is called to process the button press, exiting *KeyMenu*.

CONFIGURATION.

As was mentioned above, a Control Panel interface is included. After selecting the Control Panel desk accessory from the Apple menu, select the *KeyMenu* icon. A display similar to the following will appear:



This figure shows the default settings. Each setting will be described:

Invoke with:

Up to two "hot-keys" can be defined to invoke *KeyMenu*. The default values are Keypad-Period (show as [.]) and Command-Shift-Tab. To change any of the hot-keys, click the mouse in the appropriate box, and press the desired key combination. The following symbols are used to denote modifier keys:

⌘ Command Key
⇧ Shift key
⌥ Option Key
^ Control Key

Any key shown in brackets, such as [5], is a key on the keypad. The Caps Lock key is ignored. Choose **Clear** under the **Edit** menu to clear a key entry.

Disable with:

Either of these keys will cause *KeyMenu* to become disabled until either is pressed again. These entries are set in the same way as for the invoking keys, above.

Enable KeyMenu

This setting overrides the hot-keys. *KeyMenu* will never be activated until this setting is enabled. It will still be loaded at startup, however.

Allow use in dialogs

By default, *KeyMenu* will not activate itself when a dialog window is frontmost. Unchecking this item will tell *KeyMenu* to attempt invoke itself, regardless of the type of window that is frontmost when the hot-key is pressed.

Allow SystemTask and FKEYs

By default, *KeyMenu* will not service desk accessories or allow FKEYs to be used while it is waiting for keyboard input (which is the same as the mouse menu selection routine). Checking this options will turn off this restriction. This, incidentally, provides a great way to capture screen images showing application menus (much better than the *Camera* desk accessory)! However, undesirable visual effects may occur when desk accessories change their displays while a menu is active. In addition, some FKEYs will not function correctly when *KeyMenu* is active.

Hierarchial menus in Apple menu

When checked, this option will allow a hierarchial menu to be entered if it is under the leftmost (usually the Apple) menu. Unchecking this may be desirable if the *hierDA* cdev is installed (see the Compatibility section below),

Swap Return and Command-Return

Selecting this will cause the meanings of Return and Command-Return to be switched. The action when on a hierarchial menu item will then be to return, instead of stepping into it. The hierarchial menu will then be entered only if Command-Return is pressed.

Allow nonstandard MDEFs

By default, only menus that consist of standard text items may have items selected. An exception is also always made for the Apple menu, which is not a standard text menu under Multifinder. When this option is selected, *KeyMenu* will attempt to choose from all menus. This may or may not be successful, depending upon the type of menu. For instance, the **Lines** menu in MacDraw 1.9.5 will function correctly, while the **Patterns** menus will not. Ironically, this option must be selected for *KeyMenu* to work under Microsoft Word 4.0, which uses its own menu definition procedure.

Offscreen arrow selection

For a variety of reasons relating to the way standard menus work on the Macintosh, scrolling of menus isn't supported. Selecting this option will allow the selection of menu items that aren't being displayed when using the up and down selection keys.

Offscreen character selection

When selected, allows the selection of nondisplayed items by pressing an alpha key.

Restore defaults

Restores all settings to the defaults as shown in the figure above.

Choosing **Clear** from the Edit menu will clear the currently selected hot-key. Note that, because all keypresses are used to select hot-keys, the Command-B command equivalent will not function as a command equivalent. **Undo** is also supported.

There are two different configurations that are used by the Control Panel device (cdev). The *disk configuration* is stored in the *KeyMenu* file on disk. In addition, there is a *memory configuration* that is used by the actual INIT program, and is loaded at startup. Usually these two are the same. However, if there is a disk error (such as a write-protected disk), rendering the cdev unable to save the new settings, the two configurations will be different. When the *KeyMenu* cdev is first chosen from the *Control Panel,* a beep will sound if the two configurations are found to be different, and the memory configuration will be shown. Similarly, if an error occurs when writing out the new configuration when closing the cdev (either by closing the *Control Panel* itself, or by choosing another icon from the Control Panel window), a beep will sound. In addition, when closing, the cdev will also attempt to

find the memory configuration and save the changes there also. There will be no notification if this is unsuccessful (if *KeyMenu* was loaded at startup, the memory configuration should always get changed). Note that no settings are changed until the *KeyMenu* cdev is closed.

COMPATIBILITY

This section describes some compatibility issues associated with *KeyMenu*.

Let me first state that I only have regular access to a Macintosh Plus, and *KeyMenu* has therefore not been extensively tested on any other system (though it has been run on a Mac SE, Mac II, and SE/30 and seemed to run properly, though the cdev display was a little bit bland on a color screen). It should function as well on any Macintosh computer, but I make no promises. In addition, most of my testing has been under System 6.0.2. I have been running some version of *KeyMenu* for almost one year, and, in the last four months, have *never* had a machine crash that was attributable to *KeyMenu* (other than the TMON problem listed below). I feel that it is very stable, though I don't run a large number of different programs.

Some applications, notably an old version of *FullPaint*, handle events in unusual way; *KeyMenu* doesn't function from within *FullPaint.* It will also bomb under the 1.0d5 version of ResEdit, but runs fine under version 1.2b3.

KeyMenu coexists with *hierDA* by jbx. Any hierarchial menu item selected from a menu generated by *hierDA* by *KeyMenu* will not be processed correctly. In addition, canceling (via Clear or Command-Period) *KeyMenu* will sometimes cause a desk accessory to be erroneously selected.

KeyMenu must be loaded after *TMON Startup.*

There may be compatibility problems, with some macro programs, as both they and *KeyMenu* are vying for user events. If there are problems, experiment with the loading order of the various INITs.

KeyMenu necessarily makes assumptions about the internal structure of menus. If the "Allow nonstandard MDEFs" option is selected, and if the custom MDEF uses a non-standard structure, there may be problems. I haven't run into this, but it could happen.

THE THOUGHT POLICE

KeyMenu obeys as many of Apple's programming guidelines as possible. However, a number of deviations were necessary for its functioning to be possible. Some of these considerations are listed below (nonprogrammers may not get much out of this discussion):

1) Menu structure. The semi-documented (meaning it is documented, but is listed as for "informational purposes only") internal menu bar structure understandably had to be used.

2) MBAR resource. I was unable to find any listing specifying that the MBAR #0 resource is used in the Macintosh Plus. I use it anyway, and it seems to work fine.

3) Submenu delay. The user-interface guidelines specify a delay time between the time a hierarchial menu item is selected, and its appearance on the display, but gives no way of finding the time. I traced through the `_MenuSelect` trap, and found the variable, which I now use.

4) Tail patching. This is probably the worse offense. The `_GetOSEvent` trap had to be patched, which involved using a tail patch (and no, the `JGNEFilter` filter would *not* have worked in this instance, for a variety of reasons).

That's all I remember for now; I'm sure there are others.

DISCLAIMER AND DISTRIBUTION

To the best of my knowledge, *KeyMenu* functions only as is specified here. I would be *very* interested to know otherwise.

KeyMenu is free. This is not as much a display of my generosity as showing my not wanting to be committed to supporting *KeyMenu* for the $100 or so that I might receive. However, since I have become very dependent on it, it will almost assuredly keep up to date with any system that will run on the Mac Plus). I will do my best to support *KeyMenu* (limited, of course, by my hardware configuration). I do ask that, if you use *KeyMenu*, or have only tried it, you drop me a line telling me. I can be reached at:

US Mail: E. Jason Scheck
61344 Blakely Rd #B3
Bend, OR 97702

EMail: jasons@tekred.CNA.TEK.COM