KeyMenu: A Keyboard Based Hierarchical Menu

Kent Lyons, Nirmal J. Patel and Thad Starner
College of Computing and GVU Center
Georgia Institute of Technology
Atlanta, GA 30332-0280 USA
{kent,merik,thad}@cc.gatech.edu

Abstract

KeyMenu is a keyboard based hierarchical menu system intended for use on a wearable computer. It is designed for use in conjunction with the Twiddler one–handed wearable keyboard. Unlike traditional menus and shortcut keys, our system employs a one–to–one mapping between buttons on the keyboard and menu items. With the addition of a pop-up menu interaction technique, the KeyMenu design consistently supports both novice and expert users.

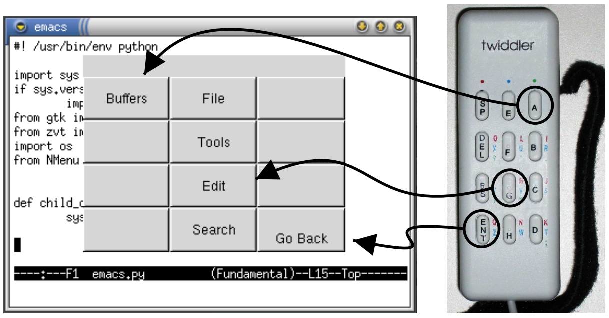

Figure 1. Each key on the Twiddler (right) corresponds directly to a menu item on KeyMenu (left).

1. Introduction

Menus provide great support for a novice computer user; a novice can explore application menus with a mouse to find available commands. Experts however, tend to abandon the mouse in favor of keyboard based methods such as shortcut keys for activating the same functions. The transition from novice to expert and mouse to keyboard is frustrating because of the difference in interaction. Wearable computers' unique characteristics confound the interaction further [4, 2]. KeyMenu is a keyboard based hierarchical menu designed to provide a consistent interaction and support both novice and expert wearable computer users.

The traditional method of activating menus with a pointing device is often not appropriate on a wearable computer. The wearable user is regularly interacting with the physical world, and the user's primary focus of attention is not on her head–up display. Coupled with a small screen size, this style of use hinders the hand–eye coordination needed to manipulate a pointer on the display. Furthermore, physical control of pointing devices becomes increasingly difficult when the user is mobile. For instance, erratic pointer movement was prominent in preliminary data collected of a wearable user's interaction in a realistic setting [3]. Although use of a pointing device is problematic, touch typing with the Twiddler is still possible while in motion.

Traditional menus can be navigated without a mouse; however, the large variety of interaction techniques available to execute commands poses problems. For example, the web browser Mozilla supports several methods of closing the current window. With the mouse, the user can select *File* and then *Close*. Using the keyboard, the user can press "Alt+F" to open the *File* menu and activate *Close* directly by pressing "C". Finally, an expert user can bypass the menu entirely and use shortcut keys, pressing "Ctrl+W" from the main window. The abundance of inconsistent mouse and keyboard interactions hinders the transition from novice to expert as a user must perform a completely different action to utilize a different method. KeyMenu uses a keyboard only technique that relies on the same key sequences for both novice and expert use to overcome these issues.

2. KeyMenu Design and Interaction

KeyMenu is designed around a grid of three columns and four rows of menu items that corresponds directly to the keys on the Twiddler keyboard (Figure 1). The user activates KeyMenu by pressing and releasing the "Alt" key on the top of the Twiddler (not shown) with her thumb. The current position in the menu hierarchy is displayed at the top of KeyMenu providing feedback to the user (Figure 2). The user navigates the menu hierarchy by pressing the keys in the corresponding positions on the keyboard in sequence.

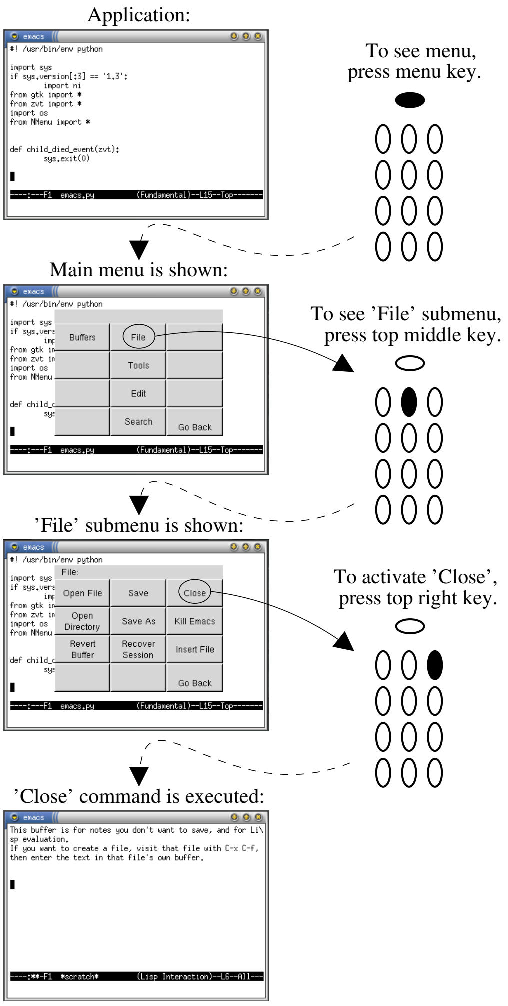

Figure 2. KeyMenu in use: the left column shows the application state, and the right column represents the keys pressed on the keyboard from the user's perspective.

Since our menu design relies on a correspondence between the physical and virtual, it is important to ensure that the user's model of the two spaces match. Anecdotally, we have found with the Twiddler that the user's conception of left and right results from holding the keys facing away. This accounts for the mirrored mapping in Figure 1.

KeyMenu utilizes an intentional delay in displaying the menu similar to Marking Menus [1]. Marking Menus are radial menus [5] operated with a pointer. The menu supports novice users by showing the menu after a short delay ($\frac{1}{3}s$), providing a prompt of available menu options. An expert makes the same physical motion as the novice but does not pause. Thus, a novice can progress to an expert skill level without needing to learn new actions. The transition follows from repeated use of frequent commands where eventually the sequence is encoded in procedural memory. KeyMenu uses a similar pop–up technique but replaces the use of a pointer with key–presses on the Twiddler.

Figure 2 illustrates a novice user's interaction with KeyMenu set up to control the text editor Emacs. The left column shows the application and menu feedback. The right column represents the keys pressed (from the user's perspective) to navigate the menu. The user activates the menu by pressing the menu key with her thumb and waits a short period ($\frac{1}{3}s$) for the system to display the main menu. The user next presses the top middle key activating the *File* sub–menu. Finally, the top right key is pressed activating the desired *Close* command, and the menu is dismissed. An expert user might remember the whole key sequence for a command and can enter it without pause. In this case, the menu is never displayed, and the command is executed directly similar to a traditional shortcut key. By using the pop–up menu with delay, the KeyMenu always uses the same key sequences to execute a command regardless of expertise.

3. Future Work

A formal evaluation of the KeyMenu design is needed. While our preliminary use seems positive, there are several issues to examine. For instance, investigating the development of procedural memory and the transition from novice to expert would be useful. Menu layout could also be explored. The keys in the top row and center column tend to be the easiest to press and are probably the best locations to put the most commonly used commands, but future work would be needed to confirm this assumption. Finally, it would be interesting to explore how KeyMenu could be generalized to operate on other devices. For instance, KeyMenu might be suitable for use with a mobile phone's three column by four row keypad and display.

4. Acknowledgments

This work is funded in part by NSF Career Grant #0093291 and the NIDRR Wireless RERC.

References

[1] G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *Proceedings of CHI'94*, pages 258–264, 1994.

[2] K. Lyons. Everyday wearable computer use: A case study of an expert user. In *Proceedings of Mobile HCI 2003*, Udine, Italy, 2003.

[3] K. Lyons and T. Starner. Mobile capture for wearable computer usability testing. In *Proceedings of IEEE International Symposium on Wearable Computing (ISWC 2001)*, Zurich, Switerland, 2001.

[4] T. Starner. *Wearable Computing and Context Awareness*. PhD thesis, MIT Media Laboratory, Cambridge, MA, May 1999.

[5] N. E. Wiseman, H. U. Lemke, and J. O. Hiles. Pixie: A new approach to graphical man-machine communications. In *Proceedings of 1969 CAD Conference Southhampton, IEEE Conference Publication 51*, page 463, 1969.