Considering the Direction of Cursor Movement for Efficient Traversal of Cascading Menus

Masatomo Kobayashi[1] *Takeo Igarashi*[1, 2]
[1]Department of Computer Science, The University of Tokyo / [2]PRESTO, JST
7-3-1 Hongo, Bunkyo, Tokyo 113-0033, Japan
E-mail: {kobayash, takeo}@is.s.u-tokyo.ac.jp

ABSTRACT
Cascading menus are commonly seen in most GUI systems. However, people sometimes choose the wrong items by mistake, or become frustrated when submenus pop up unnecessarily. This paper proposes two methods for improving the usability of cascading menus. The first uses the direction of cursor movement to change the menu behavior: horizontal motion opens/closes submenus, while vertical motion changes the highlight within the current menu. This feature can reduce cursor movement errors. The second causes a submenu to pop up at the position where horizontal motion occurs. This is expected to reduce the length of the movement path for menu traversal. A user study showed that our methods reduce menu selection times, shorten search path lengths, and prevent unexpected submenu appearance and disappearance.

KEYWORDS: GUI, cascading menus, pointing devices.

INTRODUCTION
The most common technique for handling hierarchical menus is the *cascading* menu, such as the "Start Menu" in Windows® and "Menu Bar" in Mac® OS. The conventional method used to produce menu cascading is as follows: move the mouse cursor to the target option and wait for a while (or press the mouse button) until a submenu appears along the right border of the parent menu. This method is widely used in almost every cascading menu in various GUI systems.

The typical behavior of cascading menus, however, tends to cause incorrect selection changes or unnecessary submenu appearance due to straying mouse movement. Since each menu item with a string-based label is long from side to side, the path to a submenu is elongated, which frequently causes movement errors (See Figure 1). A longer or narrower horizontal path reduces the efficiency of mouse operations, especially with these tunnel-steering tasks [1]. It is possible to prevent unintentional selection changing and the appearance of submenus by increasing the delay before submenus appear; however, this is just a trade-off between speed (or the energy required to hold the mouse button down) and accuracy.

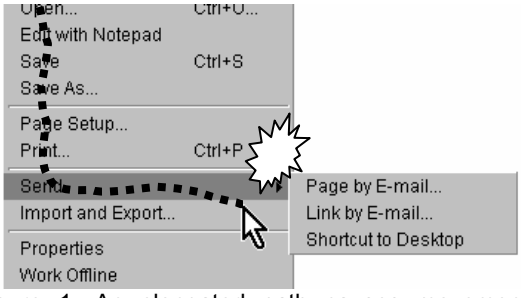

Figure 1: An elongated path causes movement errors, such as unexpected selection changes and submenu appearance.

The traditional cascading behavior also tends to extend the width of a menu chain excessively, because submenus are located to the right of the parent menu item regardless of the landscape shape of the menu items. Therefore, menu chains often need to be folded back at the right side of the screen in order to show the latest menu inside the screen. Such a folded menu chain causes irregular overlapping of menus, which makes menu traversal difficult.

This paper proposes a combination of two techniques as solutions to these problems. First, we distinguish two classes of operations involving cascading menus (changing the local selection and opening/closing menus) based on the direction of cursor movement: vertical motion changes the highlighted menu item within the current menu, whereas horizontal motion opens a submenu or closes the current menu. This feature is designed to reduce mouse movement errors. Second, the newly appearing submenu pops up near the cursor. This feature clearly shortens the path to the submenu, which reduces mouse movement errors and performance time. Moreover, the second technique reduces the width of menu chains, preventing irregular overlapping of menus.

We conducted a user study in order to compare our techniques with the conventional menu-cascading strategy in terms of the efficiency of menu traversal. This experiment investigated the general performance of typical interactions between people and cascading menus. The results showed that our techniques generally improved the speed and accuracy of menu-selecting tasks involving cascading menus.

RELATED WORK

Some menu techniques use the direction of movement of the pointing device. One of the most interesting techniques uses circular menus, such as *FlowMenu* [3], and other kinds of *pie* [2] or *marking* [4] menus. They take advantage of two dimensions by arranging menu items as in a pie chart, rather than the typical vertical list, to shorten the average movement to a menu item. Circular menus are especially suitable for pen-based or finger-based interfaces. The hierarchical structures of circular menus, however, are not as visually effective as those of vertical menus [5]. A circle takes more room to display than a corresponding vertical list, and the cascaded circles tend to overlap one another.

In contrast to circular menus that use movement direction to determine a selection, our techniques use it to control the selection process. Since they are just methods for handling menus, one can implement our techniques on any menu system that is visually identical to the traditional system.

CONSIDERING THE MOVEMENT DIRECTION

The first technique divides the traversal of cascading menus into two classes: *internal* and *external*. These two traversal classes are separated according to the associated direction of mouse cursor movement. The system determines the movement direction by comparing the horizontal component of the mouse-movement vector, dx, with the vertical component, dy. The movement direction is vertical if $|dx| < |dy|$; otherwise it is horizontal. Figure 2 shows the basic relationships between directions and events.

Internal traversal corresponds to changing the highlight, which is handled by vertical motion of the cursor. When the mouse cursor enters the territory of a selectable menu item that is currently not highlighted, this item is highlighted. If the mouse cursor exits the rectangular frame of the current menu or enters the territory of a non-selectable item, such as a separator, no item in the current menu is highlighted. Note that the territory of a menu item is unlimited horizontally within the width of the screen.

External traversal includes a submenu popping up, closing the current menu, and selecting a terminal menu item, which is handled by the horizontal motion of the mouse cursor. When the cursor moves to the right and the movement distance, d, exceeds a threshold, d_1, the system shows the submenu that is related to the currently highlighted item. Pressing the mouse button can also open the submenu. Once the submenu appears, it becomes active and the current menu becomes inactive. When a terminal item is highlighted and the user moves the cursor to the right, the item is selected and the menu selection task is concluded. Nothing occurs if there is no highlighted item in the menu. When the mouse cursor moves to the left, crossing over the left border of the current menu, this menu closes itself and activates the parent menu.

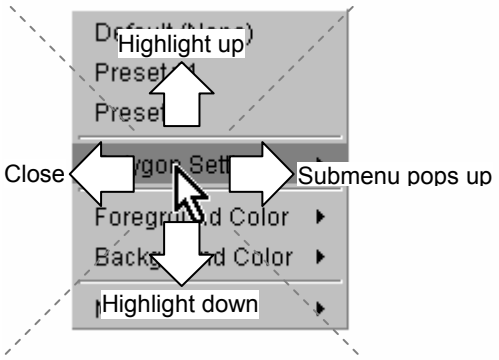

Figure 2: A user must move the mouse to the right to have a submenu pop up or to select a terminal item, to the left to close the current menu, and up or down to change the highlight.

In addition, we dim currently inactive menus so that users can clearly perceive the difference between the active menu and the others. This feature is important, since the menu under the mouse cursor is not always active with our techniques. In fact, users cannot change the active menu without a horizontal motion or clicking the mouse.

We believe that our modified menu behavior is still easy to use for most users since the horizontal/vertical policy is natural for them. According to our informal observation, users mostly move the mouse cursor horizontally and vertically, but rarely diagonally, although the optimal path is often diagonal in traditional cascading menus.

OVERLAPPING MENUS

The second technique causes a submenu to pop up at the position where horizontal motion occurs. Menus overlap their parent menus, to shorten the path to the terminal item. According to the steering law, shorter paths should cause fewer movement errors. As a result, we can prevent unnecessary selection changes and submenu appearance. Note that we might have a problem in integrating this technique with the traditional delay strategy. Unintended submenus appear automatically in such a menu system and hide the currently active menu unnecessarily.

Our second technique puts the upper left corner of a newly appeared submenu at coordinate (x, y_n), where x is the horizontal position where the horizontal motion started and y_n is the vertical location of the middle of the n^{th} item, when the n^{th} item is highlighted and it has a submenu (see Figure

3). If there is not enough space below or to the right of the menu, the system adjusts the location of the submenu so that it all appears within the screen area.

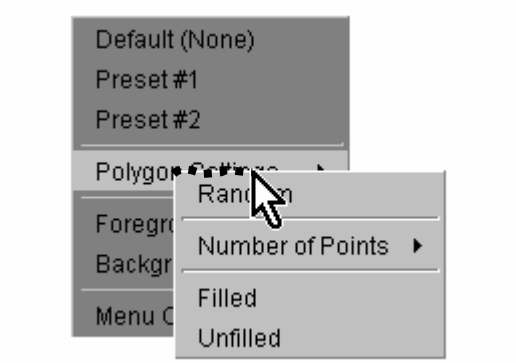

Figure 3: A submenu is located at the position where horizontal motion occurred.

In a straightforward implementation of our technique, a submenu could obscure too much of the parent menu when the horizontal motion occurred near the left side of the screen. We place a relative constraint on the location of the newly appearing submenu to prevent this. The constraint is:

$$e = x_s - x_p \geq e_1$$

where x_s and x_p are the horizontal locations of the left borders of the submenu and the parent menu, respectively, and e_1 is a constant that is defined as the minimum offset of the horizontal location of the submenu from that of the parent menu. The distance threshold, d_1, can then be adjusted so that the inequality is satisfied, as shown in Figure 4. That is:

$$d_1' = \max\{d_1, d_1 + e_1 - (x - x_p)\}$$

Then, d_1' places an appropriate offset between the submenu and the parent menu so that the hierarchical structure of the menus is visually clear.

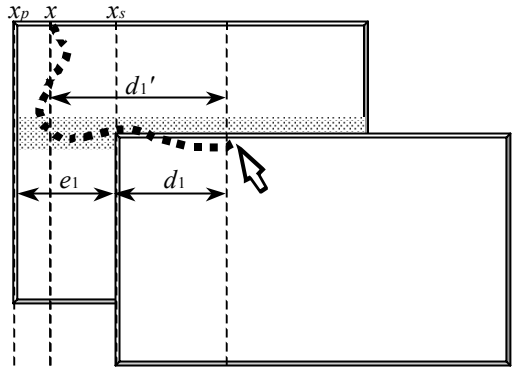

Figure 4: Adjusting d_1' keeps the proper offset $e \geq e_1$, so that the menus are aligned orderly.

EVALUATION

In this section, we describe a user study that we conducted to compare our techniques with the conventional menu-cascading strategy.

Participants

Ten right-handed students attending the local university were recruited as participants. Eight participants were frequent mouse users and six were frequent users of conventional cascading menus. None had any experience with our techniques.

Apparatus

The experiment was conducted on a Toshiba DynaBook SS 3500 with a 12.1-inch TFT LCD (1024 × 768 pixel resolution), equipped with a Logitech Wheel Mouse. The software used in the experiment was developed in Java™ 1.4.1.

Procedure

Before testing, the participants were briefed on the purpose of the experiment and the method of operating both our cascading menu and a traditional cascading menu. The participants were ordered to select a menu item, following the instruction displayed on the title bar of the window.

The menu-selection process starts with a mouse click action on a certain item in the menu bar. It ends up with the item selecting action on the instructed terminal menu item. The participants were allowed to study the instructions as long as needed before starting each menu-selection process.

Design

The experiment consisted of eight blocks: four for each cascading menu. The first block was a warm-up, and the remaining three were for data collection. Each block had 12 trials of menu-selection. The 12 targets were randomly preselected from the total of 521 menu items. The hierarchical levels of the target items ranged from two to five. Three trials were for each level. The default distance threshold, d_1, was set to 24 in pixels. Figure 5 shows screenshots of the software used in the test.

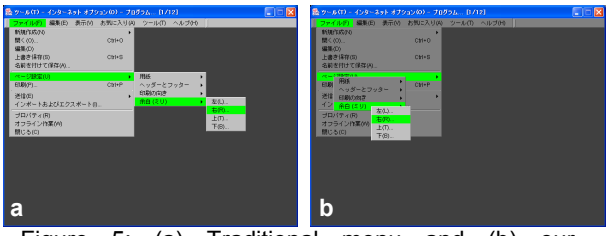

Figure 5: (a) Traditional menu and (b) our direction-based menu used in the experiment.

The experiment used a within-subjects design. Each participant was tested with both cascading menus. Half of the participants had their four blocks of the traditional menu first, followed by four blocks of our menu. The other half of the participants had the reverse order.

Result

Figure 6 shows the average selection time, movement path length, and unexpected submenu appearance per trial with standard errors. Each is the average of 12 × 3 × 10 = 360 trials. The cascading strategy had a significant effect on each measure ($p < .005$). Our techniques reduced the selection time by 12%, the movement path length by 31%, and unexpected submenu appearance by 85%. We used the Wilcoxon signed-ranks test to determine the significance of the results.

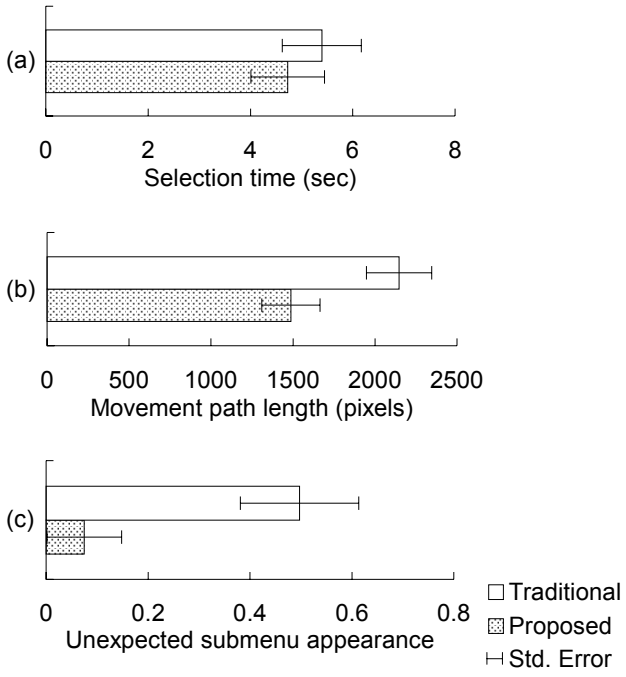

Figure 6: (a) Selection time, (b) movement path length, and (c) unexpected submenu appearance.

DISCUSSION AND FUTURE WORK

A number of studies have documented the fact that performance changes with practice when using menu-selection systems [6]. Users not only learn the arrangement of menu items, but also the behavior of the system. In fact, some participants in our experiments traversed menu hierarchies more fluently with the traditional behavior than with our techniques. Considering the reduction in the movement path length, the selection time should also be reduced after using our system several times.

Our direction-based strategy might be effective only when the user knows which menu contains the target option. The traditional delay strategy allows a user to browse submenus just by slowly moving the mouse cursor up or down over the current menu. This might be preferable for the user who does not know where the target is and needs to search for it. Our techniques, however, force such a user to repeat the unnecessary active process: move right to have a menu pop-up temporarily, review its contents, and move left to close it in disappointment. We need to conduct additional research on the menu-*searching* task in addition to the menu-*selecting* task used in this experiment.

The proposed techniques can be applied to menu systems that have a traditional appearance, which means that our direction-based strategy is easy to implement on existing systems. The cascading menu implemented with our techniques can be seen as a gesture-based system; yet it is easy to use, even for users who are accustomed to traditional cascading menus. Moreover, our techniques can be used with various pointing devices, such as mice, styluses, and fingers. We plan to apply our techniques to existing menu systems, and to make them available for general use.

ACKNOWLEDGMENTS

We thank all the participants in the user study for their help.

REFERENCES

1. Accot, J. and Zhai, S. Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp.295-302, ACM Press, 1997.

2. Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. An Empirical Comparison of Pie vs. Linear Menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp.95-100, ACM Press, 1988.

3. Guimbretiére, F. and Winograd, T. FlowMenu: Combining Command, Text, and Data Entry. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pp.213-216, ACM Press, 2000.

4. Kurtenbach, G. and Buxton, W. User Learning and Performance with Marking Menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 258-264, ACM Press, 1994.

5. Lee, E.S. and Raymond, D.R. Menu-Driven Systems. In *Encyclopedia of Microcomputers*, Volume 11, pp.101-127, Marcel Dekker, Inc., 1992.

6. Norman, K.L. *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*. Ablex Publishing Corporation, 1991.