Exploiting Extra Spatial Information to Improve Everyday Graphical User Interface Operations

Masatomo Kobayashi
Department of Computer Science, the University of Tokyo
7-3-1 Hongo, Bunkyo, Tokyo, Japan
+81-3-5841-4091
kobayash@is.s.u-tokyo.ac.jp

ABSTRACT
WIMP-based GUI is now a global standard and we use it literally all day, everyday. However, some operations are frustratingly tedious to perform and small improvements in these standard operations can bring significant benefit to many people. In this thesis, we propose to exploit extra spatial information such as mouse location and movement to enrich standard GUI operations. By exploiting these previously not-intensively-used resources, we can enrich interactions with minimum modification to the familiar operations. Specifically, we propose the following four interaction techniques. 1) considering the direction of cursor movement to improve cascading menu traversal, 2) introducing throw-and-catch interaction to make drag-and-drop suspendable, 3) considering the position of the cursor to enhance wheel-based scrolling, and 4) introducing multiple distributed cursors to cover wide screens. We describe design and implementation of these. We hope our experience will lead to further improvements of other GUI operations and provide valuable information for designing new interaction paradigms.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Graphical User Interfaces (GUI)

Additional Keywords and Phrases: WIMP, Cascading Menus, Scroll Wheels, Drag and Drop, Pointing, Gestures

INTRODUCTION
WIMP-based GUI significantly improved the usability of computer applications and has been the standard for decades. Although it is widely considered to be superior to command-based interactions, many usability issues remain unsolved such as difficulty in pointing distant object on a large screen and many attempts have been made to address these problems.

Most previous attempts are roughly divided into three categories. One is to introduce new devices that replace standard mouse and display. This approach requires significant investment for both developers and end users, making quick adaptation difficult. Second approach is to introduce new interaction paradigm such as 3D desktop environment, but it requires re-design of whole environment and difficult to incorporate into existing framework. Third approach is to introduce intelligent behavior into the system such as adaptation and prediction. This approach can be very nice but at the same time there is a danger that the system behaves in against user's intention.

Our goal is to improve the usability of everyday GUI operation without requiring the user to switch over to new device or new paradigm, hoping to provide immediate help who are experiencing frustration in current GUI. We also would like to avoid taking intelligent system approach in order to keep the sense of direct manipulation [9]. To achieve these goals, we propose to use resources that have not been used in standard GUI operations, that is, extra spatial information such as mouse location and movement. By simply considering these additional resources, we can improve the usability of everyday operations with minimizing the overhead such as redesign or learning of a new system.

Specifically, we propose the following four interaction techniques in this thesis: 1) considering the direction of cursor movement to improve cascading menu traversal [7], 2) introducing throw-and-catch interaction to make drag-and-drop suspendable [6], 3) considering the position of the cursor to enhance wheel-based scrolling [8], and 4) introducing multiple distributed cursors covering a screen to reduce time to point distant object. The users of WIMP interfaces already have a strong mental model and interaction style through their long experience. Thus, we designed these interaction techniques to preserve look and feels and interaction styles of traditional WIMP systems as much as possible so that our techniques can be seamlessly integrated into existing environments.

D-MENUS (IMPROVING CASCADING MENUS)
Cascading menus are commonly seen in most GUI systems. However, people sometimes choose wrong items by mistake and become frustrated when submenus pop up unnecessarily

UIST'07, October 7-10, 2007, Newport, Rhode Island, USA.
ACM 978-1-59593-679-2/07/0010

or target menus close accidentally. The problem is that the users are forced to navigate through horizontally long, narrow path to get to a sub-menu (Figure 1).

We propose to use the mouse cursor's movement direction to improve cascading menu traversal. Instead of responding to the cursor location, our cascading menu responds to the moving direction of the cursor: horizontal motion opens and closes submenus, while vertical motion changes the highlight within the current menu. This feature can reduce cursor movement errors. Meanwhile, users can traverse a menu hierarchy in a traditional style: moving the cursor vertically to choose an item within a menu and horizontally to traverse between parent and child menus. Our cascading menu shows a submenu at the position where horizontal motion occurs (Figure 2). This is expected to reduce the length of the movement path and the selection time, reducing the difficulty related to the steering law [1]. Our technique can improve the usability of cascading menus, preserving the appearance of traditional menus and the interaction style of menu traversal. A user study showed that our methods reduce menu selection times, shorten search path lengths, and prevent unexpected submenu appearance and disappearance.

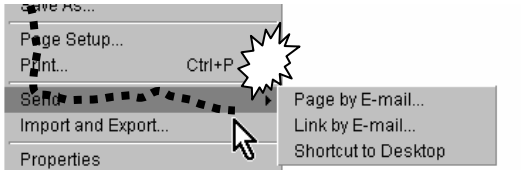


Figure 1: In a standard cascading menu, the user needs to follow long, narrow path to get to a sub menu.

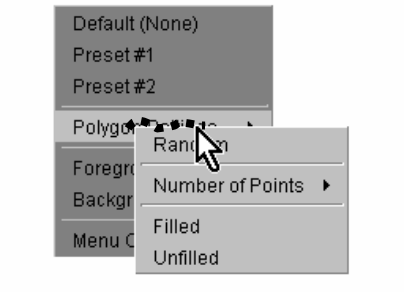


Figure 2: The horizontal movement of the cursor immediately opens the submenu at the cursor location.

BOOMERANG (IMPROVING DRAG-AND-DROP)

The drag-and-drop technique is frequently used in modern desktop environments. However, it has many usability issues. One of the most serious problems is that once users start dragging an object, very few actions can be taken while holding the object. For example, users cannot switch to the foreground window by clicking the title bar of a background window, although such actions are often necessary to make the drop target visible. Dragicevic [4] addressed this problem by allowing one to temporarily switch the foreground window with a *leafing through* gesture. However, users may also need to scroll a window to a distant target, open a hidden folder in the folder tree, or change the active tab, sheet, and slide. These operations are difficult or even impossible to perform using existing techniques.

To address this problem, we developed the "boomerang" technique, which allows users to suspend a dragging operation using a throw-and-catch metaphor. Once dragging is suspended, users can perform another operation, including those described above, before dropping the object. Figure 3 and 4 illustrate a basic behavior of the technique. First, users can start dragging an object as in a traditional dragging interaction. To perform another operation before dropping, they *throw* the object by releasing the mouse button while continuing to move the pointer. The object flies out of the screen, and its location when it was thrown is marked by an animated, translucent circle. At this time, users can complete any operation to make the hidden target visible such as, for example, scrolling a window. After finishing, users move the mouse pointer to the translucent circle to make the thrown object fly back into the screen, and *catch* it by clicking on the object, and then resume the drag-and-drop operation. We also propose advanced operations including grouping, copying, and deleting of dragged objects to help users in some scenarios.

In existing environments, it is possible to suspend ongoing dragging actions using a temporary folder. However, this operation is not a complete alternative to the clipboard because it actually moves the dragged file to the temporary folder. In addition, different types of dragged items, e.g., text and images, need different types of temporary spaces, e.g., text editor and paint tool, respectively. The boomerang technique is an attempt to use the glass pane placed on top of all windows as a general temporary workspace as an alternative to the clipboard.

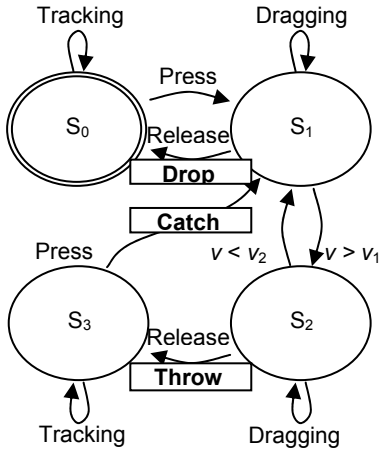
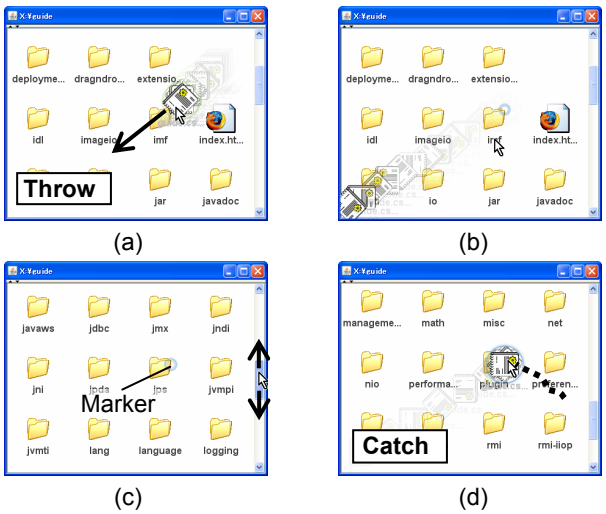


Figure 3: A state diagram of boomerang interactions. The transitions between S_0 and S_1 are the same as normal interactions.



(a) (b) (c) (d)

Figure 4: (a) A throwing gesture suspends a dragging operation; (b) the thrown object flies out of the screen; (c) users do something else, e.g., window scrolling; and (d) they catch the thrown object and resume dragging.

MOREWHEEL (IMPROVING SCROLLING)

Scrolling is one of the most common operations in many computer applications. The typical way to scroll involves using the wheel on the mouse or the scroll bar on the right side of the window. However, the scroll wheel has less functionality than the scroll bar, i.e., it does not support page-by-page and absolute scrolling, which the scroll bar supports. Meanwhile, the scroll bar seems less efficient, forcing users to carefully click small buttons and drag small knobs.

We propose to enhance wheel-based scrolling by considering the location of mouse cursor on the screen. Our technique, MoreWheel, introduces a variable-speed scrolling technique based on the position of the mouse cursor. With this technique, users can change the scrolling speed dynamically, from the speed of line scrolling to that of page scrolling. In addition, we use the dragging action with the wheel button down for absolute scrolling instead of variable-speed scrolling. Table 1 compares the functionality of our design with that of traditional scroll wheels and scroll bars. Our design allows users to scroll a window page-by-page, line-by-line, and even at an intermediate speed by rotating the wheel only, considering the cursor location.

Table 1. Supported functionality and how the functionality is invoked in each interface.

	MoreWheel	Scroll Wheels	Scroll Bars
Line Scroll	Wheeling at the center	Wheeling	Clicking
Page Scroll	Wheeling near the boundary	-	Clicking
Absolute Scroll	Dragging	-	Dragging

Relative Scrolling

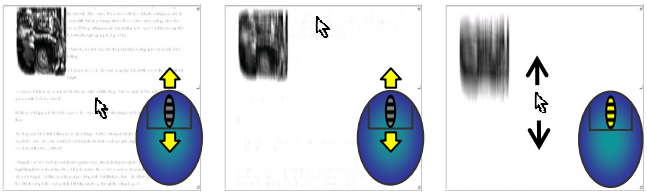
Our technique allows users to scroll at any speed between the speed of line scrolling and that of page scrolling. The speed depends on the position of the mouse cursor (Figure 5-a, b). The scrolling speed, v, is calculated using a function as follows:

$$v = \begin{cases} v_0 & for\ |y| < 1/3 \\ v_{page} & for\ |y| > 5/6 \\ v_0 k^{2|y|-2/3} & otherwise \end{cases} \qquad k = \frac{v_{page}}{v_0}$$

This function represents a mapping from the position of the mouse cursor to the scrolling speed. We designed and examined several mapping functions for target applications including documents that require vertical scrolling only, documents involving a little horizontal scrolling, and maps that require fully capability of horizontal scrolling as well as vertical scrolling.

Absolute Scrolling

In our technique, wheel-dragging results in absolute scrolling (Figure 5-c). When a user moves the mouse cursor within a window with the wheel button down, the window acts as if the scroll knob were dragged. That is, users can move the knob to any position without moving the mouse cursor onto the scroll bar. Therefore, the size of the scroll bar could be reduced to save screen space because users need not to grab the small knob; the scroll bar simply acts as a visual feedback to indicate the current position. We also propose another method to invoke absolute scrolling: dragging with both left and right buttons down. It can preserve the common behavior of wheel-dragging that causes rate-based scrolling.



(a) (b) (c)

Figure 5: (a) If the cursor is at the center, wheeling causes line scrolling; (b) if the cursor is near the boundary, it causes page scrolling; and (c) wheel-dragging causes absolute scrolling.

NINJA CURSORS (IMPROVING POINTING)

Pointing is the most fundamental operation in WIMP interfaces. Therefore many techniques have been proposed to improve the efficiency of pointing tasks in various contexts. These techniques tried to reduce the index of difficulty based on the Fitts' law:

$$ID = \log_2(D/W + 1)$$

where D and W are the distance to and the width of the target,

respectively. For example, the Dock in Mac OS X enlarges the target icon to increase W. Drag-and-pop [3] temporarily move target objects toward the pointing cursor to reduce D. Delphian Desktop [2] allows users to jump to the target location. These techniques use a prediction algorithm to determine the target object. Hence, the efficiency depends on the precision of the algorithm.

We propose a method called "ninja cursors", which increases the number of cursors to reduce D without any prediction algorithm. In this technique, multiple cursors move synchronously following physical mouse movement (Figure 6). Users can point a target object with a cursor that locates nearest to the object. Using n cursors, the expected D for 2-D pointing tasks is reduced following the equation:

$$D' = \frac{D}{\sqrt{n}}$$

To prevent two or more cursors from pointing some objects at once, we introduce a criterion: not more than one cursor can point an object at a time. Our technique satisfies this criterion using a simple FIFO (first in, first out) algorithm to determine the priority of cursors. That is, during one cursor, C_i, is pointing an object, O_i, another cursor, C_j, cannot enter the inside of an object O_j until C_i leaves O_i even if the user moves the mouse to point O_j with C_j. Once C_i leaves O_i, C_j can point O_j. If two or more cursors are in the "waiting" state, the cursor waiting for the longest time becomes able to point an object when C_i leaves O_i. To help users recognize the state of cursors, the cursor pointing an object and ones waiting for another cursor to leave the pointed object are highlighted using different colors.

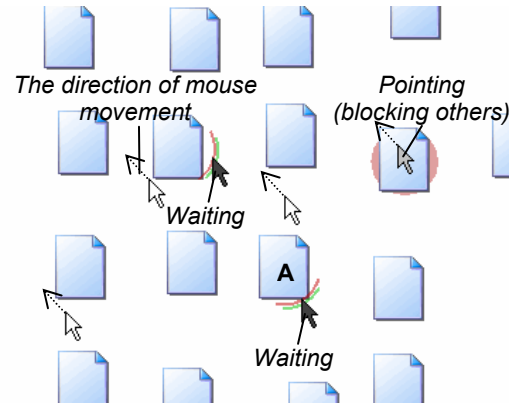


Figure 6: The user attempts to point the icon "A". The gray cursor currently points the highlighted icon. The black cursors wait for the gray one to leave the icon. Using multiple cursors, the minimal distance to the target object is expected to be reduced.

In general, our technique tries to improve the efficiency of pointing operation by mapping single mouse movement to multiple cursors' movement. We are developing several applications based on this basic concept. For example, desktop icons, toolbar buttons, and links in a Web page will be pointed much easier with our technique at least if the number and arrangement of cursors are suitable for each context. We also plan to conduct experiments to determine the relationship between efficiency and the number and arrangement of cursors.

FUTURE DIRECTIONS

We have introduced our attempts to improve everyday GUI operations such as pointing and drag-and-drop by exploiting extra spatial information such as mouse location and movement. Although we believe that these techniques are individually useful as they are, many research questions are unanswered yet, such as, whether exploitation of extra resource interferes with other interactions, and how to present these enhanced behavior to the first-time users, and to what extent these techniques improve user experience. We plan to address these issues through user studies and discuss the result in the dissertation.

REFERENCES

1. Accot, J. and Zhai, S. Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. In *Proceedings of CHI'97*, 1997, pp.295-302.
2. Asano, T., Sharlin, E., Kitamura, Y., Takashima, K, and Kishino, F. Predictive Interaction Using the Delphian Desktop, In *Proceedings of UIST'05*, 2005, pp.133-141.
3. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., and Zierlinger, A. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-Operated Systems. In *Proceedings of INTERACT'03*, 2003, pp.57-64.
4. Dragicevic, P. Combining Crossing-based and Paper-based Interaction Paradigms for Dragging and Dropping between Overlapping Windows. In *Proceedings of UIST'04*, 2004, pp.193-196.
5. Fitts, P.M. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement, *Journal of Experimental Psychology*, Volume 47, 1954, pp.381-391.
6. Kobayashi, M. and Igarashi, T. Boomerang: Suspendable Drag-and-Drop Interactions Based on a Throw-and-Catch Metaphor. In *Proceedings of UIST'07*, in press.
7. Kobayashi, M. and Igarashi, T. Considering the Direction of Cursor Movement for Efficient Traversal of Cascading Menus. In *Proceedings of UIST'03*, 2003, pp.91-94.
8. Kobayashi, M. and Igarashi, T. MoreWheel: Multimode Scroll-Wheeling Depending on the Cursor Location, In *Adjunct Proceedings of UIST 2006*, 2006, pp.57-58.
9. Shneiderman, B. Direct Manipulation: A Step beyond Programming Languages. *Computer*, Vol.16, no.8, 1983, pp.57-69.