MoreWheel: Multimode Scroll-Wheeling Depending on the Cursor Location

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

Takeo Igarashi
Department of Computer Science,
The University of Tokyo / JST PRESTO
7-3-1 Hongo, Bunkyo, Tokyo, Japan
Tel: 81-3-5841-4109
takeo@acm.org

ABSTRACT
This paper describes the design of a scrolling interaction that supports all line, page, and absolute scrolling with a scroll wheel only. This technique changes the scrolling speed dynamically depending on the position of the mouse cursor. It also allows users to scroll a window to an absolute position by dragging with the wheel button down. We developed a prototype utility for applying the proposed technique to Microsoft Windows.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Interaction Design, Mouse, Document Scrolling

INTRODUCTION
Scrolling is one of the most common operations in many computer applications, including web browsers, word processors, spreadsheets, and drawing tools. The typical way to scroll involves using the wheel on the mouse or the scroll bar on the right side of the window. However, these methods are relatively weak as follows:

A scroll wheel allows users to scroll a document line by line. In addition, common Windows' systems support fast, continuous scrolling by moving the mouse with the wheel button down. However, scrolling in this mode is difficult to control because of the lack of centering feedback [6]. In addition to continuous scrolling, some modern systems, such as Mac OS, support automatic scrolling acceleration, which increases the unit distance of scrolling depending on the speed of wheel rotation. This technique allows users to scroll a long distance with less finger motion [4]. However, it sometimes does not meet users' common intentions, such as "I would like to scroll down two pages", because it does not support page scrolling.

The scroll bar provides various scrolling functions. First, users can scroll line by line by clicking one of the small buttons at the top or bottom of the bar. Second, users can scroll page by page by clicking the margins around the scroll knob. Third, users can scroll directly to the specified position by dragging the scroll knob. However, using a scroll bar involves a difficulty related to Fitts' law [3] because the component is thin and the scroll knob and buttons are small. Moreover, dragging the knob also involves a difficulty related to the steering law [1] because it forces users to move the mouse cursor along a long, thin path. To reduce this difficulty, in practice, common systems provide margins along either side of the scroll bar, where users can continue dragging the knob. Nevertheless, users often miss the margins. Mouse handling likely becomes less accurate while scrolling because users look at the contents of the window, rather than at the scroll bar.

We propose an interaction design called MoreWheel that enhances the functionality of the scroll wheel so that users can scroll a window line-by-line, page-by-page, and even directly to an absolute position with only the scroll wheel. Our goal is to provide a scrolling experience with the scroll wheel that is comparable to or even beyond that with scroll bars, with less difficulty related to Fitts' law and the steering law. To meet this goal, we introduced a variable-speed scrolling technique based on the position of the mouse cursor. With this technique, users can change the scrolling speed continuously and 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.

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

	MoreWheel	Scroll Wheels	Scroll Bars
Line Scroll	Wheeling	Wheeling	Clicking
Fast Scroll	Wheeling	Dragging	-
Page Scroll	Wheeling	-	Clicking
Absolute Scroll	Dragging	-	Dragging

RELATED WORK
Many techniques have been introduced to improve scrolling performance. For example, Brewster et al. [2] enhanced the scroll bar with auditory effects, and Igarashi and Hinckley [5] enhanced rate-based scrolling with an automatic zooming mechanism. However, modifications for scroll wheels are relatively rare. One of the few is the automatic scrolling acceleration proposed by Hinckley et al. [4].

UIST '06, October 15-18, 2006, Montreux, Switzerland

Several researchers have compared the performance of scroll wheels with other scrolling techniques. Zhai et al. [6] found that wheeling was less efficient than using TrackPoint or even scroll bars. Hinckley et al. [4] showed that traditional wheeling performed well only for short distances. Despite such inefficiency, the mouse wheel has become an essential device for desktop computing. Therefore, we believe that it is important to attempt to improve this device.

DESIGN DETAILS

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 1-(a) describes the mapping from the position of the mouse cursor to the scrolling speed. The scrolling speed, v, is calculated as follows:

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

where y is the relative position to the center of each window, which is –1 when the mouse cursor is at the bottom of the window and +1 when it is at the top of the window. v_0 is the normal scrolling speed (the speed of line scrolling), and v_{page} is the speed of page scrolling. Figure 1-(b) shows the resulting speed curve. We designed the mapping between the scrolling speed and the cursor location so that the resulting scrolling behavior would not confuse users. First, the behavior is the same as traditional scrolling when the cursor is near the center of the window. Second, the scrolling speed is the same as page scrolling when the cursor is near the top or bottom of the window. This behavior will assist users who would like to scroll down by two pages. Third, the speed curve is continuous, so that users can switch between slow and fast scrolling seamlessly. Although users need to move the mouse cursor to change the scrolling speed, this causes little difficulty because the user need not grab a small knob or move it along a narrow path as in a scroll bar.

We also examined different mappings, such as discrete mapping (Figure 1-(c)) and horizontal gradation (Figure 1-(d)). However, an informal user study showed that the majority preferred the first mapping. This is understandable, since discrete mapping would confuse users because even a small motion causes a drastic change in the scrolling behavior if the mouse cursor is near the border. Conversely, it is not obvious why vertical gradation is better than horizontal gradation. In addition, several participants commented that they sometimes failed to scroll at the intended speed though our technique improved the overall performance of scrolling. This problem could be eliminated if visual feedback, such as changing the appearance of the mouse cursor, was in place.

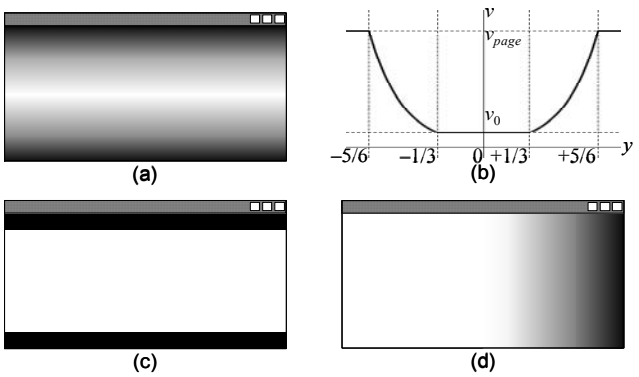

Figure 1. If the cursor is in a dark area, the scrolling speed becomes faster: (a) continuous, vertical mapping; (b) the speed curve for (a); (c) discrete, vertical mapping; and (d) continuous, horizontal mapping.

Absolute Scrolling

In our technique, wheel-dragging results in absolute scrolling. 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 grab the small knob; the scroll bar simply acts as a visual feedback to indicate the current position.

IMPLEMENTATION

A prototype system was implemented as a resident utility for Microsoft Windows. It hooks and rewrites mouse events to change the behavior of scrolling depending on the cursor location. Therefore, the utility works in any application that uses standard components, such as scroll bars and views. This will be an advantage for our technique over more sophisticated techniques for scrolling enhancement, such as [5], which are relatively difficult to apply to existing software.

REFERENCES

1. Accot, J., and Zhai, S. Performance evaluation of input devices in trajectory-based tasks: An application of the Steering Law, In *Proceedings of CHI'99*, 1999, pp.466-472.
2. Brewster, S.A., Wright, P.C., and Edwards, A.D.N. The design and evaluation of an Auditory-Enhanced ScrollBar, In *Proceedings of CHI'94*, 1994, pp.173-179.
3. 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.
4. Hinckley, K., Cutrell, E., Bathiche, S., and Muss, T. Quantitative analysis of scrolling techniques, In *Proceedings of CHI'02*, 2002, pp.65-72.
5. Igarashi, T., and Hinckley, K. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of UIST'00*, 2000, pp.139-148.
6. Zhai, S., Smith, B.A., and Selker, T. Improving browsing performance: A study of four input devices for scrolling and pointing tasks, In *Proceedings of INTERACT'97*, 1997, pp.286-292.