EyeWindows: Using Eye-Controlled Zooming Windows for Focus Selection

David Fono and Roel Vertegaal
Human Media Lab
Queen's University
Kingston, ON K7L 3N6, Canada
{ fono, roel } @ cs.queensu.ca

ABSTRACT

Most windowing systems from the past twenty years use independent overlapping windows. However, today's user is often engaged in numerous simultaneous tasks, with multiple windows vying for screen space. In these cases, overlapping windows can easily obscure each other, forcing the user to make constant manual adjustments in order to switch and monitor tasks. We have developed EyeWindows, an alternative windowing technique that uses eye-controlled zooming windows. In EyeWindows, information is distorted rather than obscured, and switching tasks is a simple matter of looking at the appropriate window.

KEYWORDS: Attentive User Interfaces, Windowing Systems, Alternative Input, Eye Tracking, Zooming Interfaces.

INTRODUCTION

Today's use of windowing systems is characterized by large numbers of open windows and frequent shifts of user attention. Users engage in multiple simultaneous tasks and switch continuously between them. Further, users prefer their open tasks to remain visible, to promote situational awareness. Today's overlapping windowing systems facilitate neither of these behaviours, since information is easily occluded by numerous windows. Users are therefore forced to spend a significant amount of time on manual windowing operations. This problem has prompted research into windowing systems that allow users to switch tasks with greater ease, as well as keep more tasks simultaneously visible.

Several zooming interfaces have been developed for task switching and awareness, such as fisheye views [2]. However, with the exception of Apple's Exposé [1], zooming techniques have not been applied to windowing systems. By selectively magnifying and shrinking non-overlapping windows, it is possible for a user to work effectively on a selection of focus tasks, while other tasks remain easily accessible in the user's periphery. Furthermore, the effectiveness of such a technique could be increased by basing focus window selection on a direct

measurement of user task focus, as provided by an eye tracking device. There is a good case for using eye input for focus window targeting operations. First, the eyes typically acquire a target well before manual pointing is initiated [5]. Second, eye muscles operate much faster than hand muscles [5]. Finally, the eyes provide a continuous and parallel signal that frees the hands for other tasks.

EYE-CONTROLLED ZOOMING WINDOWS

EyeWindows is a non-overlapping windowing system that combines zooming windows with eye-controlled focus selection. The user selects a focus window by looking at it and pressing a keyboard trigger. Upon selection, the window smoothly zooms to a fixed size and can be interacted with as usual, while surrounding windows shrink to avoid being obscured by the focus window. These distorted windows act as *eycons*, providing live previews of their content in the user's peripheral vision. Eyecons seamlessly zoom back to full size when activated by the user.

We have developed two EyeWindows prototypes using different windowing techniques based on the theme of eye-controlled zooming windows. For both prototypes, we deployed a Tobii [4] eye tracker. The Tobii integrates eye tracking capabilities into a 17" screen connected to a server which relays eye input coordinates to client computers over TCP/IP.

Prototype 1: EyeWindows Media Browser

Our first prototype design for EyeWindows explores the task of parallel media browsing. Figure 1a shows how we implemented a digital media browser that facilitates the exploration of multiple audiovisual sequences in one tiled windowing environment. The EyeWindows browser renders any number of concurrent media streams onto the display by dividing the screen into a corresponding number of connected tiles. Initially, all tiles have the same size. Figure 1b shows how a tile zooms when the user selects it. Surrounding tiles accommodate the change by shrinking and moving away proportionally.

Tile size adjustments are propagated across surrounding tiles using the original Elastic Window algorithm [3]. This algorithm preserves the relative proportions of adjoining tiles, by adjusting the border of each tile proportionately to the distance of that border from the
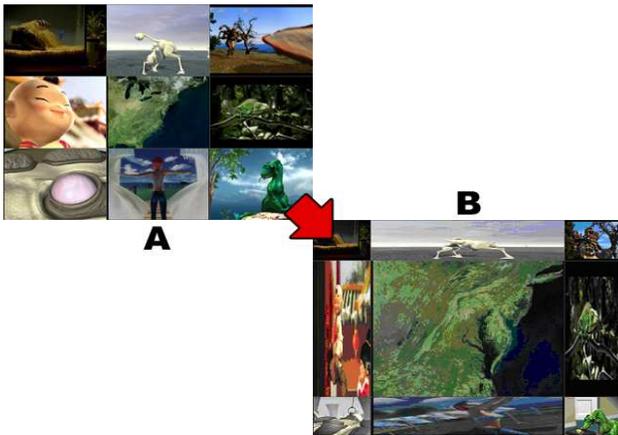
Figure 1. When the user selects the central tile in figure 1a, it zooms to a fixed size to allow closer inspection, in figure 1b. The surrounding tiles shrink to accommodate the change.

screen's edge. For instance, if the right border of an expanding tile moves to the right by 30 pixels, and the right border of an adjoining tile is located halfway between the expanding border and the screen's right edge, then the latter border moves by 15 pixels. A full explanation of the algorithm can be found in [3]. Tiles have a minimum horizontal and vertical size. If a border adjustment requires that a tile is shrunk below this size, the adjustment propagates to the next border. Failing this, the original adjustment is not allowed.

Prototype 2: Desktop EyeWindows

For our second prototype, we adapted elastic windowing to function in a normal desktop environment with arbitrary positioning of windows. The system augments the standard Mac OS 10.3 desktop without eliminating any of the usual manual functionality. When a user selects a focus window using the eye-controlled trigger, all overlapping and adjacent windows shrink and move aside to accommodate the zoom. Similarly, when an application creates a new window overlapping existing windows, those windows move aside to remain visible.

Window size adjustments are propagated using an adapted elastic algorithm. Upon activation, a focus window expands until it reaches full horizontal and vertical resolution. If the focus window can fully expand without colliding with surrounding windows, the size of those windows remains unchanged. If an expanding border collides with a neighbouring window partway, then the borders of that window are adjusted to accommodate the remainder of the expansion. Since the two borders are now immediately adjacent, the adjustments are calculated the same as in our tiled prototype. As the neighbouring window moves, the algorithm is recursively applied onto any other windows it collides with. Figure 2b shows the results of this technique when the upper-left window in Figure 2a is selected. A buffer gap of at least 5 pixels is maintained between windows at all times. As in the first prototype, windows have a minimum width and height.
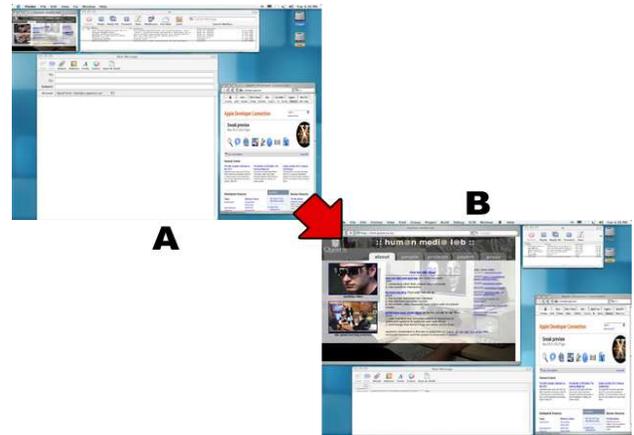

Figure 2: Activating the upper left window in figure 2a causes it to expand, as in figure 2b. Subsequently, this window's bottom border collides with the lower left window, which shrinks vertically. The right border collides with the upper right window, which shrinks and slides horizontally. Since no border of any moving window collides with the lower right window, it remains unaffected.

OBSERVATIONS AND DISCUSSION

Initial observations of users working with both prototypes indicate that zooming windows perform better than manual windowing techniques when frequent shifts of attention between windows are required. Users appreciated the ability to keep all their tasks onscreen at once, while simultaneously concentrating on one or two enlarged focus tasks. Furthermore, when engaged in manual input to an application, users found it extremely convenient to switch focus simply with a look and a key press. These results suggest that key activated eye control is an effective mode for focus window selection when manual input is already overloaded by application functionality.

REFERENCES

1. Apple Computers, Inc. Mac OS X Exposé. www.apple.com/macosx/features/expose/, 2003.
2. Furnas, G.W. Generalized Fisheye Views. In Proceedings of CHI'86 Conference on Human Factors in Computing Systems. Boston: ACM Press, 1986, pp. 16-23.
3. Kandogan, E. and Shneiderman, B. Elastic Windows: Evaluation of Multi-window Operations. In Proceedings of ACM CHI'97 Conference on Human Factors in Computing Systems. Atlanta: ACM Press, 1997, pp. 250-257.
4. Tobii Systems. www.tobii.se, 2003.
5. Zhai, S. What's in the Eyes for Attentive Input. Special Issue on Attentive User Interfaces, Communications of ACM Vol. 46, No. 3, 2003, pp. 34-39.