EyeWindows: Evaluation of 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

In this paper, we present an attentive windowing technique that uses eye tracking, rather than manual pointing, for focus window selection. We evaluated the performance of 4 focus selection techniques: eye tracking with key activation, eye tracking with automatic activation, mouse and hotkeys in a typing task with many open windows. We also evaluated a zooming windowing technique designed specifically for eye-based control, comparing its performance to that of a standard tiled windowing environment. Results indicated that eye tracking with automatic activation was, on average, about twice as fast as mouse and hotkeys. Eye tracking with key activation was about 72% faster than manual conditions, and preferred by most participants. We believe eye input performed well because it allows manual input to be provided in parallel to focus selection tasks. Results also suggested that zooming windows outperform static tiled windows by about 30%. Furthermore, this performance gain scaled with the number of windows used. We conclude that eye-controlled zooming windows with key activation provides an efficient and effective alternative to current focus window selection techniques.

ACM Classification: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Windowing systems

Keywords: Attentive User Interfaces; Alternative Input; Eye Tracking

INTRODUCTION

Windowing systems of commercial desktop interfaces have experienced little change over the last 20 years. Today's systems employ the same basic technique of allocating display space using manually arranged, overlapping windows into the task world. However, the task world of current computer users is radically different from that of 20 years ago. In particular, today's user is characterized by frequent shifts of attention between tasks [25]. We can identify a number of reasons for this. Firstly, users are more likely to work on multiple simultaneous tasks. As a result, many application windows may be open at any given time (see Figure 1). This means windows are likely to occlude one another, obscuring the user's view into the task world. Secondly, due to the emergence of the Internet, current users are frequently interrupted by digital communications, such as email and instant messaging notifications [17, 22]. The above two factors conspire to increase the number of window operations required to reveal information pertaining to an event or task of interest. This problem has prompted research into windowing systems that allow users to shift attention with greater ease, e.g., through zooming techniques, as in Sideshow [6], Fisheye views [9], Elastic Windows [13, 14] and Mac OS X Exposé [1]. While most of this work emphasizes the use of manual input for task management, there has been little work on windowing systems that sense the user's attention using more direct means.

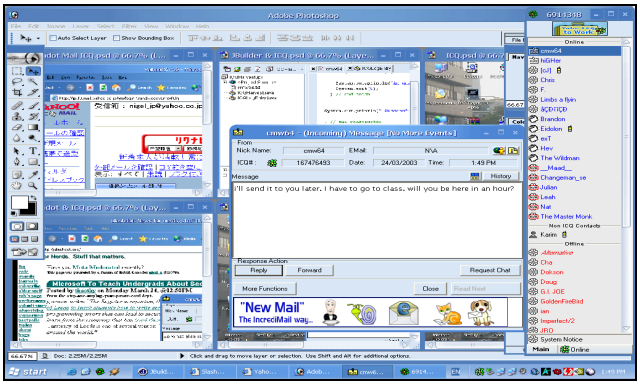

Fig. 1. Typical clutter in today's windowing systems.

Using an alternate channel for sensing the attention of the user for parts of a display has two key benefits. First, it allows an undisrupted use of manual input for task-oriented activities. Second, it allows for continuous accommodation to shifts in user attention. By tracking the focus of attention of users, interfaces may actively avoid obscuring the user's view. By designing a more careful and attention-sensitive placement of windows and dialog boxes, annoying visual interruptions may thus be reduced [10, 16, 20].

CHI 2005, April 2–7, 2005, Portland, Oregon, USA.

Summary of the Paper
In this paper, we explored how eye tracking may be used to manage tasks more effectively within a limited display space. EyeWindows is a non-overlapping windowing system that uses eye tracking to decide which window is within the focus of user attention. EyeWindows also uses zooming windows to manage display space according to user focus. The user selects a window for focus by looking at it while pressing a trigger key. Subsequently, the focus window zooms to a useable resolution, with surrounding windows shrinking and moving into the periphery of the user's vision. First, we will discuss prior work, after which we present two prototypes of our system. We then discuss an empirical evaluation of eye-based focus window selection. Results indicate that eye input is faster than either mouse-based or key-based selection when hands are occupied by content-related activity. Finally, we present an evaluation of the performance of zooming windows, in comparison to a tiled windowing technique. Results show that zooming windows outperform static tiles when window content is greater than available display space.

PREVIOUS WORK

According to Baudisch et al. [2], the problem of allocating screen real estate is indeed one of allowing users to optimize their attention space. The human eye can only attend to a relatively small portion of display space at any one time. Fine-grained visual acuity is limited by the retina to an area of about 2 degrees of visual angle: the fovea [7]. As a result, at any one time, only a limited display area of about 2-5 cm around the location of an eye fixation is made available for higher-level cognitive processing [7]. Other areas of a display are observed with peripheral vision, providing crucial information for understanding the context of foveated information. Peripheral vision also guides eye movements *(saccades)*. During a saccade, visual processing in the brain is attenuated. This allows a computer display to imperceptibly alter the resolution of displayed information during eye movements. Displays that employ this technique are known as Gaze-Contingent Displays (GCDs) [2]. A good example of a static windowing system that exploits the limited resolution of human peripheral vision is the Focus Plus Context screen [3].

Windows of Attention

Although Focus Plus Context works well for tasks that involve a single large window, multitasking requires a mechanism for choosing one amongst many visual tasks. According to Smith et al., WIMP (*Windows, Icons, Menus and Pointers*) [19] interfaces were originally designed for this purpose. *Windows* represent foreground tasks at high resolution, and occupy the bulk of display space in the center of vision. *Icons* represent peripheral tasks at low resolution in the periphery of the user's vision. *Pointers* allow users to communicate their focus of attention to graphic objects. By clicking icons to open windows, and by positioning, resizing and closing windows, users use their pointing device to manually manage their attention space. However, by covering other windows and icons, front or *focus* windows may obscure part of the global context of tasks in an overlapping windowing system. A number of solutions have been designed for this problem:

a) Tiled Windows. According to Bly and Rosenberg [4], we can expect a tiled windowing system to be preferable over an overlapping one in task situations where the contents of the window conform to a predetermined window arrangement. Their evaluations show that novices perform better with tiled windows than with overlapping windows. Experts also perform better with tiled windows, but only in *regular tasks*, tasks that require few window manipulations in order to view the required contents. For *irregular tasks*, in which not all content is immediately visible, expert users perform better with overlapping windows.

b) Elastic Windows. Tiled windowing does not easily lend itself to control over spatial organization of a large number of documents. Kandogan and Shneiderman [13,14] suggested a space-filling tiled windowing system that allows hierarchical organization. Their Elastic Windows system allows users to organize windows by dragging items into a container window. The window group stretches like elastic material when resized, with surrounding window groups shrinking proportionally to fill the remaining space. Evaluations show Elastic Windows outperforming overlapping windows in most of the studied tasks.

c) Fisheye Views. Furnas [8] suggested early on that if displays would behave like a fisheye lens, users would be able to spatially organize a large number of documents without overlap. There have been many studies on the use of fisheye views, particularly in groupware systems [9]. In a fisheye view the magnification or "zoom factor" of a window depends dynamically on its distance to the pointer. The fisheye lens causes the focus window at the location of the pointer to expand, while other windows drift to the edge of the screen. However, Gutwin [9] showed that continuous fisheye magnification actually slows down focus window targeting. This slowdown arises because the magnification lens makes windows appear to move in the direction opposite to pointer movement. While speed-coupled flattening [9] reduces this effect, we considered continuous fisheye magnification detrimental to eye-controlled targeting.

Interacting with Multiple Windows

In addition to optimizing the way in which screen real estate is allocated in situations with many open windows, we investigated how we might optimize the process of selecting a focus window. Most current windowing systems use focus window selection techniques based on manual pointing devices. However, there are a number of problems with these techniques, particularly when they are combined with an overlapping windowing system. Firstly, manual input may be overloaded by focus selection tasks. This means that the hands cannot perform a content-related task while targeting a focus window. For example, the use of a mouse for focus window selection may introduce homing times while typing on a keyboard. Secondly, uncovering a window obscured by the current focus window in an overlapping windowing sys-

tem may require several pointing actions and keystrokes. First, the current focus window may need to be resized by clicking inside its zoom box and dragging its outline. Next, the pointer needs to be moved to click inside the title bar. Next, the window needs to be dragged to the side such that the new focus window is revealed. Finally, the user needs to click inside the new focus window. Thus, overlapping windows may reduce the efficiency of focus window targeting when many windows are open.

Using Eye Input for Focus Window Selection

We experimented with altogether removing the need for manual pointing in focus window targeting actions. It has been noted that eye input is inappropriate for manual control tasks, since the eye is a perceptual organ. Eye movements are frequently unintentional, so a direct mapping to explicit commands would lead to frequent unintended effects (the Midas Touch problem [11]). Furthermore, eye tracking yields inexact measurements. In developing their MAGIC system [26], Zhai et al. found that eye tracking is inefficient for direct cursor control, but useful for warping the cursor's approximate "home" position. This technique is effective because eye movements merely set the context for successive manual cursor movement. Eye tracking could be applied similarly to focus window selection. Selecting a new focus window merely redefines the context for content-related manual operations. Furthermore, windows in a non-overlapping environment are usually large enough to allow for inexact gaze measurements.

There is a good case for using eye input for focus window targeting. Firstly, the eyes typically acquire a target well before manual pointing is initiated [25]. Secondly, eye muscles operate much faster than hand muscles [25]. Finally, the eyes provide a continuous signal that frees the hands for other tasks. One of the few systems to deploy eye input for focus window targeting was Gaze-Orchestrated Dynamic Windows by Bolt [5]. The system simulated a composite of 40 simultaneously playing television episodes on one large display. Via a pair of eye tracking glasses it sensed when the user looked at an image, turning off the soundtracks of other episodes. If users looked at one episode for a few seconds, the system would "zoom in" to fill the screen with that image. Unfortunately, early eye tracking technology was severely restricted in resolution. Recent advances have made it possible to invisibly integrate an eye tracker with a head movement tolerance of up to 30 cm and an accuracy of better than 1 degree into a 17" LCD screen [21] (see Figure 2).

Empirical Evaluations

It is important to note that the use of eye input for focus window targeting is quite distinct from that of using eye input for cursor positioning. While there are a number of experimental studies evaluating eye-controlled cursor positioning, there are few studies that examine eye-controlled focus selection [7,11,12,18,24]. Results of performance comparisons between eye-based and mouse-based selection of on-screen targets are mixed. Ware & Mikaelian suggested that selection of on-screen targets with eye movements follows Fitts' law [24]. Its index of performance is similar to the mouse, but its selection time (the intercept in Fitts' law) is twice as fast. Jacob [11,12] studied the use of an eye tracker for target selection, but did not report quantitative comparisons. Sibert and Jacob [18] reported that dwell-time activated eye-based selection of large on-screen targets was about twice as fast as mouse-based selection. Wang et al. [23] discussed an evaluation of eye-based selection of Chinese characters for text entry. Users chose one of 8 on-screen characters by looking at the character while pressing the space bar. Results showed that eye-based selection was no faster than traditional key-based selection. They attributed this to eye tracking lag, and the fact that their task was dominated by decision time rather than selection time.

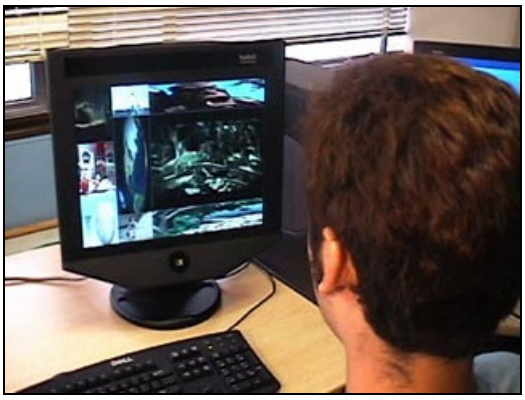

Figure 2. User with Tobii eye tracker running EyeWindows.

EYEWINDOWS: EYE-CONTROLLED ZOOMING WINDOWS

We will now discuss EyeWindows, an eye-controlled windowing system that uses eye tracking as a means for parallel selection of focus windows. To facilitate eye control, we explored two different non-overlapping windowing techniques. The first prototype explores the use of elastic windowing for the task of media browsing. The second prototype uses a hybrid elastic windowing technique that allows arbitrary placement of windows for arbitrary tasks on a desktop. Both prototypes feature automatic zooming of the focus window after selection. Before we discuss each windowing technique, we will first describe some of the details of the hardware and software implementation.

System Implementation

For eye input, we deployed a Tobii [21] eye tracker. The Tobii integrates tracking capabilities into a 17" screen, and communicates eye position over TCP/IP to client computers running EyeWindows. The media browser was implemented using the Microsoft Directshow C++ API, and runs on a 1.5 GHz Intel PC under Windows XP. The second prototype was implemented using Apple's Cocoa API, and runs on an 800 MHz G4 under Mac OS 10.3. Requiring only a single 15-second calibration, the Tobii eye tracker allows 30*x*15*x*20 cm of user head movement with an average on-screen accuracy of 1 cm. Users can move in and out of view of the tracker without having to recalibrate.

Prototype 1: EyeWindows Media Browser

Our first prototype design for EyeWindows explored the parallel browsing of online digital media. Figure 4 shows how we implemented a digital media browser that facilitates the exploration of multiple audiovisual sequences in one tiled windowing environment. A sequence may consist of a live webcast or an online movie file. EyeWindows renders any number of live media streams onto the display by dividing the screen into a corresponding number of connected tiles. Each tile is active, and plays concurrently to allow the user to simultaneously explore all material in the set.

Eye-Controlled Zooming Windows

Figures 4 and 5 show how windows can be enlarged to reveal more detail. The user selects a window for focus by looking at it and pressing the spacebar. Upon selection, the focus window is automatically zoomed to a predefined size. The zoom function animates the window's four borders away from its center coordinate until a preset maximum size is reached, or until the user selects another focus window. Zoom is applied linearly through time, and completes within 1 *s*. As in Elastic Windows, surrounding tiles accommodate the change by shrinking proportionally. Enlarging focus windows in this manner reveals visual detail and allows the user to inspect the video on display. Shrinking the remaining windows prevents them from being obscured, thus allowing the user to maintain peripheral awareness of all videos while examining the focus window. Since focus window selection is done with the eyes, the user continuously observes the videos without having to refocus attention on peripheral input devices.

Elastic Windowing Algorithm

Tile size adjustments are propagated across tiles using the original Elastic Window algorithm [13]. This algorithm adjusts the border of each tile proportionately to the location of that border. Figure 3 shows how the border movement of Tile A is propagated to Tiles B, C, and D. The right border

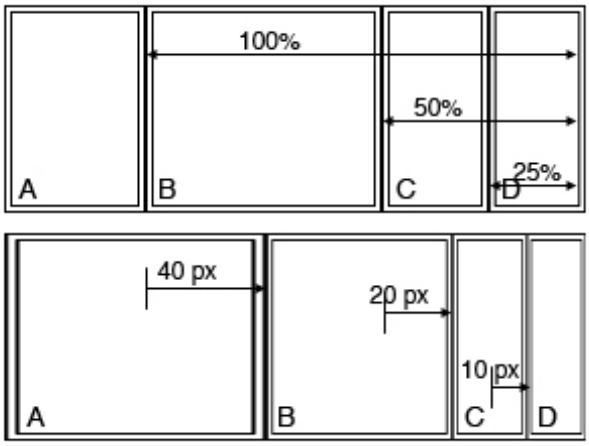

Figure 3. Example of an elastic window border operation. Window A's right border is moved to the right by 40 pixels. These pixels are distributed proportionally over the other borders. B's border moves 50% of the adjustment or 20 pixels, C's border moves 25% of the adjustment or 10 pixels.

of tile A is adjusted by 40 pixels. The distance between the right border of tile B and the edge of the screen is 50% of the distance between the original border of tile A and the edge of the screen. For the right border of tile C, that relative distance is 25%. Subsequently, the right border of tile B is moved 50% of the adjustment, or 20 pixels, while the right border of tile C is moved 25% of the adjustment, or 10 pixels. A full explanation of the algorithm, including the calculation of propagation adjustments, can be found in [13]. Tiles have a minimum horizontal and vertical size. If a border adjustment requires that a tile be shrunk below this size, the adjustment propagates to the next border. Failing this, the original adjustment is not allowed.

Auditory Zoom

In addition to visual zooming of the focus window, EyeWindows allows users to focus on the audio stream of the focus

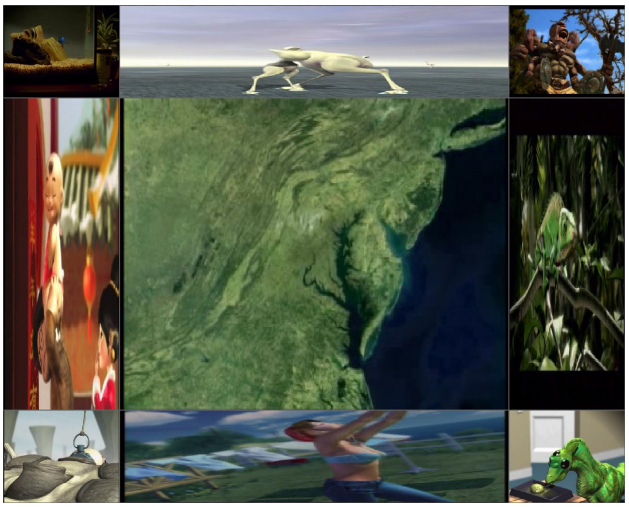

Figure 4. A digital media browser with nine EyeWindows tiles. The center window is the current focus window (images courtesy ACM SIGGRAPH 2001 Video Program).

Figure 5. To select a new focus window, the user looks at a tile while pressing the space bar (here the bottom left tile). As the tile zooms, all surrounding windows shrink.

window by attenuating audio from other tiles. Attenuation is logarithmically proportional to tile surface area. Each tile's stereo balance is panned proportionally to the on-screen location of the tile.

Scenario 1: Digital Media Browsing

The following scenario illustrates the use of EyeWindows for the task of searching a movie segment in a multimedia database with a limited sample set.

User Alex is looking for the name of the author of a movie he saw while visiting the SIGGRAPH conference. He remembers a segment showing a beautifully ray-traced glass dropping off a table. Alex enters the URL of the SIGGRAPH video program in EyeWindows. The system scans the URL for movies files, loading each of the nine movies found into a separate tile of equal size. As the movies start playing, Alex focuses sequentially on each tile for detailed inspection of its contents. As he focuses on the center tile it expands, allowing him to inspect the movie sequence in great detail, while remaining aware of activity in the other movies in his peripheral vision (see Figure 4). As he watches, Alex anticipates hearing the sound of breaking glass. When he hears the sound originating from the bottom left tile, he shifts his gaze, expanding the tile. As the closing credits roll, resolution is now sufficient for Alex to note the name of the author of the clip (see Figure 5).

User Observations: Overloaded Visual Input

We performed initial evaluations comparing eye tracking to the mouse for focus window selection using our prototype. Ten users engaged in a video searching task similar to the scenario described above, with selection time measured for each input technique. Users were also asked to comment on their experiences using the prototype. Overall, users found the system useful and satisfying for media browsing. Furthermore, we found that eye input using spacebar activation performed equivalently to the mouse with click activation. One of the caveats with the use of eye tracking as parallel input during browsing is that manual input might not in fact be overloaded. Instead, when asking users to search for a movie segment, the eye's role in providing visual input to the user becomes overloaded with the role of providing output to the computer [25]. One advantage of manual selection in this scenario is that the hands can move toward the next anticipated focus window in parallel, while the eyes remain on a segment playing in the current focus window. These observations indicate that eye tracking may perform superiorly to the mouse when manual input is overloaded by content-related activities.

Prototype 2: Desktop EyeWindows

For our second prototype, we adapted elastic windowing to function in a normal desktop environment with arbitrary positioning and resizing of windows. This allowed us to provide the user with multiple large windows placed anywhere on the screen, in a way that minimized the number of zooming operations required. Windows may contain any application or document normally available in Mac OS X. Figures 7 and 8 show many applications on-screen at once, with a single zoomed focus window. We again deployed eye input for focus selection, thus freeing up the hands for content-related activities such as typing.

Modified Elastic Windowing Algorithm

Our adaptation of the Elastic Window algorithm first tries to allocate any unused space to the focus window. 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 a neighboring window prevents full expansion of the focus window, the neighbor is first moved laterally to use up empty space between windows. Further operation of the algorithm is similar to that in our first technique. Figure 6 shows how adjustments are propagated recursively upon collision of borders. A buffer gap of at least 5 pixels is maintained between windows at all times. Upon activation, focus window A is expanded to the right by a total of 50 pixels. There are 15 pixels of unused space between A's right border and window B's left border. Maintaining a buffer of 5 pixels between the windows, B's left border is moved to the right by 40 pixels. This movement is propagated proportionally to B's right border. Since B's right border is located halfway between B's left border and the edge of the screen, B's right border accommodates 50% of the 40 pixels, moving 20 pixels to the right. If a window C were placed at the right of B, the movement of B's right border would similarly propagate to the borders of C. As in our first prototype, the expansion is animated linearly

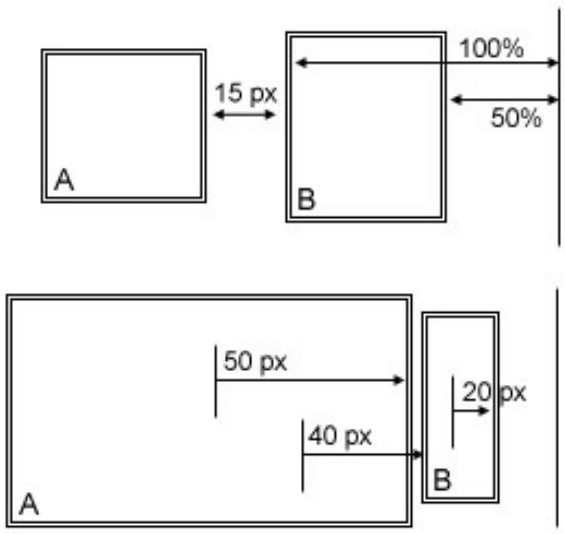

Figure 6. Example of an Irregular Elastic Window zooming operation. The bottom diagram shows the results of window A's expansion from the top diagram. The right border of A is moved to the right by 50 pixels, which pushes the left border of B to the right by 40 pixels. Since B's right border is located halfway between B's left border and the edge of the screen, the adjustment is proportionally propagated to the right border of B, which moves 20 pixels. This effectively removes unused space between affected windows, but limits the effect of the operation on the rest of the screen.

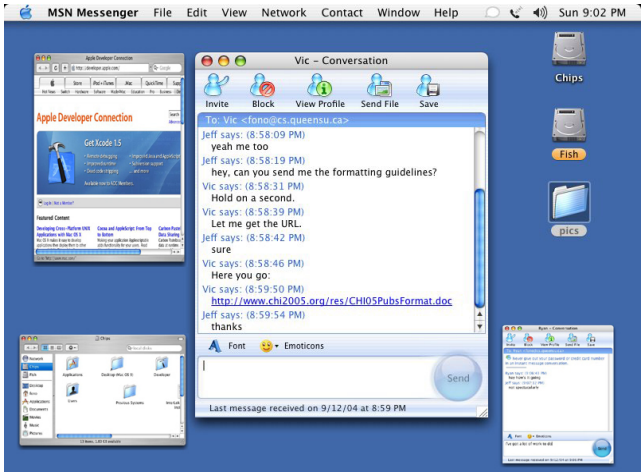

Figure 7. The user concentrates on a single focus window with an active conversation. Other windows are distorted but remain visible.

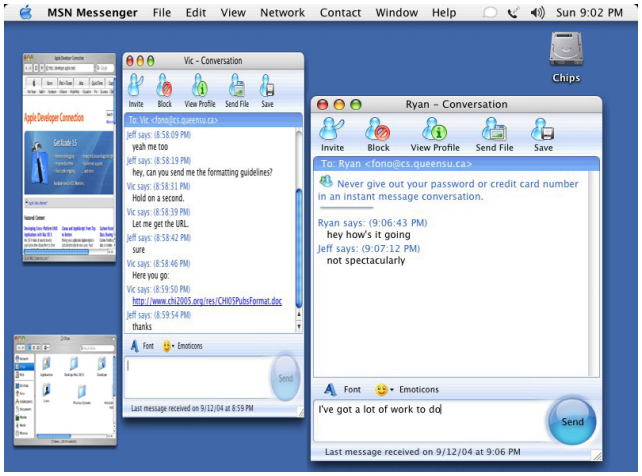

Figure 8. The user selects the lower right window as the new focus window. That window zooms to full resolution, while the other windows move aside.

with a 1 *s* completion time. The rates of expansion for horizontal and vertical axes of the focus window are calculated independently, such that expansion in both directions always completes simultaneously. As before, all windows have a minimum horizontal and vertical size. If a border adjustment requires that a window is reduced below this size, the adjustment propagates to the next border. Failing this, the original adjustment is not allowed.

Scenario 2: Visual Turn Taking in Instant Messaging

The following scenario illustrates how our second prototype supports parallel input, with the eye specifying context and the hands specifying content. We chose instant messaging, as it is a good example of a common task with manual overloading (where typing and focus window selection is normally performed using sequential hand movements), and rapid shifts of attentional focus.

Figure 7 shows user Jeff in an instant messaging session with user Vic. Jeff notices the online arrival of his friend Ryan through a notification alert in the periphery of his vision. He looks at the alert and presses the activation key. The alert is dismissed and a new instant messaging window opens. Jeff looks back at Vic's window, and continues typing a response to Vic without having removed his hands from the keyboard. In his peripheral vision he notices Ryan posting a response. He looks at Ryan's window while pressing the activation key and immediately starts typing. Ryan's window zooms (see Figure 8). A third person, Jake, arrives. After opening a session, Jeff wants to copy one of Vic's responses to Jake. He looks at Vic's window while pressing the activation key. Jeff keeps his hands on the keyboard to scroll back and copy the line. He looks at Jake's window, hits the activation key and pastes the response.

EXPERIMENT 1: EYE INPUT FOR FOCUS WINDOW SELECTION

We designed an experiment to evaluate the efficiency of eye input for focus window selection when manual input is overloaded by content-related tasks. The experiment compared the performance of four selection techniques during a simple transcription task. Since we wished to evaluate eye-controlled focus selection separately from zooming windows, the experiment was performed on a simplified version of our second prototype. We later conducted a second experiment using the second prototype itself, in order to evaluate eye-controlled focus selection in unison with zooming windows.

Participants and Design

Twelve volunteers participated in the experiment. All were expert mouse users, seven had previous experience with eye tracking, and six were touch typists. We used a within-subjects design, where each participant used each of the four selection techniques. Participants performed 3 trials with each selection technique, using 4, 8, and 12 windows at a time. The orders of presentation for selection technique and number of windows were counterbalanced between subjects. After completion of all twelve trials, participants were asked to fill out a questionnaire evaluating the various selection techniques.

Selection Techniques

Participants used four different techniques for focus window selection, across three different input devices:

- *Eye tracker with key activated selection (Eye +key).* The current coordinate of on-screen gaze was used to determine the target window. This target window became selected after pressing the spacebar.
- *Eye tracker with automatic selection (Eye + auto).* The coordinate of on-screen gaze was used to determine the target window. The target window became selected immediately upon eye fixation.
- *Mouse with click activated selection.* A two-button mouse was used to position a visible cursor over the target window. The target window became selected after pressing the left mouse button.

- *Hotkeys.* The F1-F12 function keys were used to select corresponding windows. During the 4-window condition, windows corresponded to keys F1-F4. During the 8-window condition, the first row of windows corresponded to keys F1-F4, and the second row corresponded to keys F5-F8. During the 12-window condition, keys were the same as with 8 windows, with keys F9-12 added for the third row.

Window Arrangements

Participants performed three trials using each selection method, with a different number of windows each time:

- *Four windows:* A 2x2 grid.
- *Eight windows:* A 4x2 grid.
- *Twelve windows:* A 4x3 grid.

Apparatus

The experimental task was displayed on a 1024x768 LCD screen. We used an LC Technologies Eyegaze system to implement the two eye input conditions for the experiment. Users were calibrated with the system using a standard 15-point calibration procedure. We used a Logitech optical mouse for the mouse condition. To allow for optimal performance with the mouse, control-to-display gain was set to medium using standard Mac OS X acceleration. We used a 108-key Apple Extended keyboard for the hotkey condition.

Task

Participants were presented with a grid of 4, 8 or 12 windows. Each window contained an upper panel and a lower panel. At the start of the trial, the lower panel of a random window would turn red. The participant would then select the window, and we measured the time from stimulus to selection. Upon selection, the lower panel would turn into a text entry field, and the upper panel would show a 6-character string. Participants would then type the given string into the lower panel (see Figure 9). Upon completion, the string would disappear and another window would turn red. This process was repeated sequentially 20 times for each trial. We chose this task because it is a representative abstraction of activities that require both continuous manual input and frequent window switching.

Results

Table 1 shows a ranked list of the mean window selection times and standard errors for each combination of selection technique and number of windows. Selection times varied significantly with selection method ($F_{3,33}=104.54,\ p<0.01$) and with number of windows ($F_{2,22}=17.00,\ p<0.01$). The interaction between method and number of windows was also significant ($F_{3,102}=2.58,\ p=0.027$).

Post-hoc comparisons show that differences between eye with key activation and mouse were significant in all windowing conditions (after Bonferroni corrections: $t_{11}=-5.08,\ p<0.01,\ t_{11}=-6.38,\ p<0.01,\ t_{11}=-9.42,\ p<0.01$ for 4, 8, 12 windows respectively). At 4 windows, eye with key activation was 33% faster than mouse. At 8 windows, eye with key activation was 36% faster, and at 12 windows, it was 34% faster.

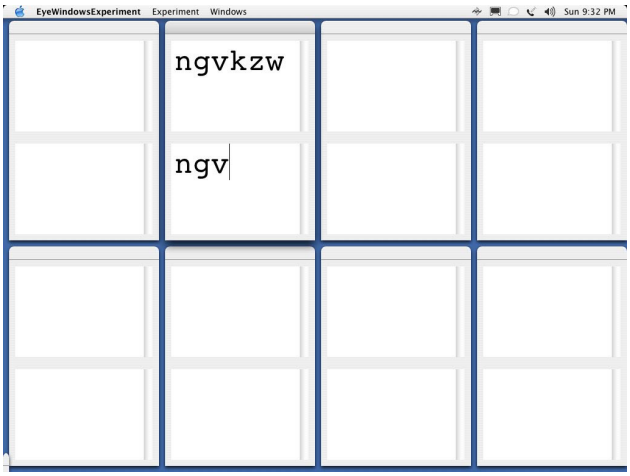

Figure 9. Screenshot of the experimental task from our first experiment. The user has selected a window, and is now transcribing the string that appeared.

Post-hoc comparisons also show that eye with automatic activation was significantly faster than eye with key activation (after Bonferonni corrections: $t_{11}=-8.26,\ p<0.01,\ t_{11}=-6.01,\ p<0.01,\ t_{11}=-4.22,\ p=0.06$ for 4, 8, 12 windows respectively). The difference in speed between eye with automatic activation and eye with key activation was approximately 23% in all cases.

User Satisfaction

Table 2 shows the mean scores for responses to the four 5-point Likert-type questions on our questionnaire. For each selection technique, the questions evaluated speed of window selection, difficulty of window selection, distraction caused by selection, and overall impression. A non-parametric Friedman Test showed differences between selection methods were significant for perceived speed ($\chi^2(3)=14.04,\ p=.003$) and difficulty ($\chi^2(3)=8.63,\ p=.035$). Pairwise comparisons suggest that the differences between eye with key activation and eye with automatic activation were not significant. Similarly, the differences between mouse and hotkeys were not significant. These results suggest that observed differences were *between* the two pairs of selection methods. Differences between input methods were not significant for distraction and overall satisfaction ($\chi^2(3)=5.66,\ p=0.129$ and $\chi^2(3)=5.65,\ p=0.231$, respectively).

Participants frequently reported unintentional focus window selection and fatigue using eye input with automatic activation. Although participants appreciated the way in which the focus of their keyboard input seemed to "magically" flow across the screen with their eyes, they also reported frustration with their inability to look away from the target window while typing. This limitation caused significant difficulties for participants who were not efficient touch typists, since they could not look at the keyboard while typing.

Mean Selection Time (milliseconds) *(standard error)*			
Method	**4 Windows**	**8 Windows**	**12 Windows**
Eye + auto	448 *(14.5)*	461 *(11.2)*	475 *(14.3)*
Eye + key	580 *(23.4)*	591 *(22.2)*	618 *(32.0)*
Mouse	868 *(65.6)*	927 *(60.8)*	941 *(55.0)*
Hotkeys	1035 *(59.5)*	1186 *(69.7)*	1200 *(60.4)*

Table 1. Ranked mean selection times and standard errors for each of our four selection methods at each window arrangement.

Discussion

Results indicate that for focus window selection during tasks that require manual input, eye input is more efficient than either a mouse or hotkeys. Eye input with key activation provided a performance increase over mouse and hotkeys by a minimum of 33%. Eye input with automatic activation provided an additional speed increase over key activation of approximately 135 ms, in line with typical keystroke selection times of about 200 ms. The 65 ms difference is likely due to lag from the eye tracker, which was more noticeable when using automatic activation, and may have mitigated its speed advantage over key activation.

The primary reason for the poor performance of the mouse in this task appeared to be manual overloading. In order to select a new window after typing a string, participants had to move their dominant hand from the keyboard to the mouse. They then had to move the hand back to the keyboard to continue typing. Conversely, both eye input methods allowed users to keep their hands on the keyboard at all times. Furthermore, the eye input methods appeared to reduce cognitive load while selecting windows. During training trials, hotkeys often outperformed the mouse. We believe this was because during training, participants were able to consistently concentrate on the key-to-window mapping. However, during the actual task, participants had to swap between typing and window selection. Concentration switched continuously to and from the key-to-window mapping, increasing the mental load placed on participants during the task. These observations indicate that hotkeys may be appropriate for tasks involving selection alone, but inappropriate for tasks that involve other content-related manual activity.

Despite the difference in speed between automatic activation and key activation for eye input, eye input with key activation appeared to be the more effective method overall. When using eye input with automatic activation, participants were unable to glance away from the target window while typing. This restriction would severely limit the efficacy of eye input with automatic activation in real world usage, since users often consult off-screen documents or other information sources while typing. Moreover, users would be unable to look at their input devices while using them. Thus, eye input with key activation appears to provide the most appropriate compromise between efficiency and effectiveness.

Mean Score				
	Eye + key	**Eye + auto**	**Mouse**	**Hotkeys**
Fast	*4.2*	*4.7*	*3.5*	*2.9*
Difficult	*2.4*	*2.2*	*3.3*	*3.8*
Distracting	*2.4*	*2.7*	*3.8*	*3.6*
Overall Satisfaction	*3.7*	*3.9*	*3.0*	*2.8*

Table 2. Mean responses to each of the questions about our four selection methods.

EXPERIMENT 2: ZOOMING WINDOWS

One of the stipulations of using eye input for focus window selection is that windows may not overlap. Overlapping windows would result in certain windows becoming partially or completely obscured, thus making them inaccessible to the eye. While some partially overlapped windows in an overlapping environment could still be selected, there would be no clear way to limit overlap to within the acceptable threshold. Eye input as a focus window selection method is thus best applied in a non-overlapping environment. In EyeWindows, we attempted to limit the repercussion of this stipulation on resolution by implementing windows that zoom upon focus selection.

We conducted a second experiment to evaluate the efficiency of zooming windows compared to regular static windows. Since this evaluation was performed within the context of EyeWindows, eye input with key activation was used as the focus window selection technique.

Design

Ten of the participants from the first experiment participated in this experiment as well. The same apparatus was used. Each participant engaged in four trials with the following conditions: 4 zooming windows, 12 zooming windows, 4 static windows, and 12 static windows. Once again, the order of conditions was counterbalanced between participants.

Task

As in the first experiment, participants were presented with a grid of 4 or 12 windows. Each window had an upper and a lower panel. Each upper panel contained a random 500-character string spread across multiple lines, with a scrollbar to the right of the text. At the start of the trial, the lower panel of a random window would turn red. Upon selecting this window, the lower panel would turn into a text entry field. The user then typed the first 20 characters of the string (see Figure 10). Upon completion, no more input into the current window was allowed, and another window would turn red. This process was repeated 10 times. When a participant returned to a previously selected window, the previously typed text would remain, and the participant would enter the next 20 characters of the string. Thus, participants often had to scroll down to reveal more of the string in the

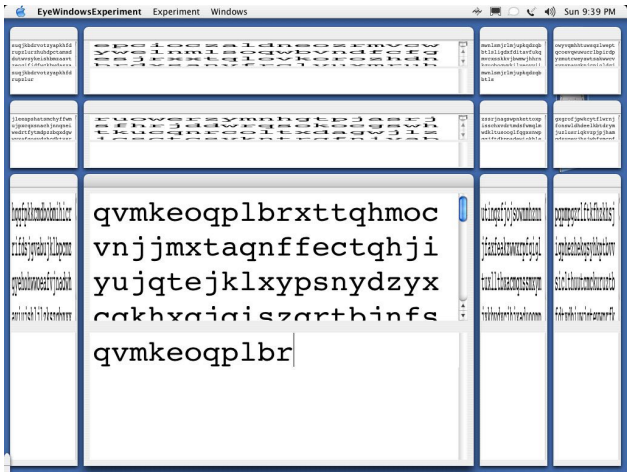

Figure 10. Screenshot of the experimental task from our second experiment, using zooming windows. The user has zoomed a window by selecting it, and is now transcribing a portion of the string within. Once the text within view has been transcribed, the user must scroll down to reveal more of the string.

upper panel. We measured time from stimulus to completion of the 20-character text entry.

During the static window conditions, window sizes never changed and text was displayed at a constant resolution. During the zooming window conditions, windows were initially shrunk and would zoom to a larger size using the Elastic Window algorithm described in our first prototype. Text was displayed at normal font size when the window containing the text was zoomed. Text was reduced in size when the window containing the text was shrunk. Thus, zooming windows displayed more text per line than static windows.

Results

Table 3 shows a list of the mean text entry times and standard errors for each condition. Times varied significantly with windowing style ($F_{1,9}=96.51$, $p<0.01$) and with number of windows ($F_{1,9}=27.42$, $p=0.01$). The interaction between windowing style and number of windows was also significant ($F_{q,9}=48.89$, $p<0.01$).

Post-hoc comparisons show that the difference between static and zooming windows was significant at both 4 and 12 windows (after Bonferroni corrections: $t_9=6.76$, $p<0.01$ and $t_9=9.59$, $p<0.01$, respectively). At 4 windows, working with zooming windows was 14% faster than working with static windows. At 12 windows, the difference was 30%. These results strongly suggest that the performance gain afforded by zooming windows increases with the number of windows in use.

After the experiment, each participant was asked which windowing style they preferred. Participants appeared to adapt quickly to the sudden changes on the screen caused by zooming windows, unanimously preferring this technique.

Discussion

Zooming windows led to noticeably better user performance than static windows throughout the experiment. Furthermore, the difference was much greater with 12 windows on-screen than it was with 4 windows. The primary difficulty with tiled, static windows is that window size is limited by the sharing of display space: the more windows there are, the less content can be displayed in each. This problem manifested itself in the experiment with a tendency of subjects to scroll more when windows were smaller. Since zooming windows increase in size when selected, they require fewer scrolling actions. Given that participants worked with only one window at a time, the distortion of peripheral windows did not appear to pose any problems. We conclude that zooming windows are an effective alternative to static windows in a non-overlapping environment when content is greater than the available display space. This observation is in line with the results of the Elastic Windows evaluation in [14]. Zooming windows are particularly effective when many windows are open simultaneously.

Mean Text Entry Time (seconds) *(standard error)*		
Style	**4 Windows**	**12 Windows**
Static	17.0 *(0.9)*	21.4 *(1.3)*
Zooming	14.6 *(1.2)*	14.9 *(1.4)*

Table 3. Mean text entry times and standard errors for each windowing style at each window arrangement.

CONCLUSIONS

We have presented an attentive windowing system that uses eye tracking, rather than manual pointing, for focus window selection. We evaluated the performance of 4 focus selection techniques: eye tracking with key activation, eye tracking with automatic activation, mouse and hotkeys in a typing task with many open windows. We also evaluated a zooming windowing technique designed specifically for eye-based control, comparing its performance to that of a standard tiled windowing environment. Results indicated that eye tracking with automatic activation was, on average, about twice as fast as mouse and hotkeys. Eye tracking with key activation was about 72% faster than manual conditions, and preferred by most participants. We believe eye input performed well because it allows manual input to be provided in parallel to focus selection tasks. Results also suggested that zooming windows outperform static tiled windows by about 30%. Furthermore, this performance gain scaled with the number of windows used. We conclude that eye-controlled zooming windows provides an efficient and effective alternative to current focus window selection techniques.

REFERENCES

1. Apple Computers, Inc. Mac OS X Exposé. http://www.apple.com/macosx/features/expose/, 2003.

2. Baudisch, P., DeCarlo, D., Duchowski, A., and Geisler. W. Focusing on the Essential: Considering Attention in Display Design. In Special Issue on Attentive User Interfaces, Communications of ACM Vol. 46, No. 3, 2003, pp. 60-66.

3. Baudisch, P., Good, N., Belotti, V., and Schraedley, P. Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming. In Proceedings of CHI'02 Conference on Human Factors in Computing Systems. Minneapolis: ACM Press, 2002, pp. 259-266.
4. Bly, S. and Rosenberg, J.K. A Comparison of Tiled and Overlapping Windows. In Proceedings of CHI'86 Conference on Human Factors in Computing Systems, Boston: ACM Press, 1986, pp. 101-106.
5. Bolt, R. A. Gaze-Orchestrated Dynamic Windows. In Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques. Dallas: ACM Press, 1981, pp. 109-119.
6. Cadiz, J., Venolia, G., Jancke, G. Gupta, A. All Ways Aware: Designing and Deploying an Information Awareness Interface. In Proceedings of ACM CSCW'02 Conference on Computer Supported Cooperative Work. New Orleans: ACM Press, 2002, pp. 314-323.
7. Duchowski, A. *Eye Tracking Methodology: Theory & Practice*. London, UK: Springer Verlag, 2003.
8. 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.
9. Gutwin, C. Improving Focus Targeting in Interactive Fisheye Views. In Proceedings of CHI'02, Conference on Human Factors in Computing Systems. Minneapolis: ACM Press, 2002, pp. 267-274.
10. Horvitz, E., Jacobs, A., and Hovel, D. Attention-Sensitive Alerting. In Proceedings of UAI'99 Conference on Uncertainty and Artificial Intelligence, 1999, pp. 305–313.
11. Jacob, R. The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At is What You Get. In ACM Transactions on Information Systems, Vol. 9, No 3, 1991, pp. 152-169.
12. Jacob, R. What You Look At Is What You Get: Eye Movement-Based Interaction Techniques. In Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems. Seattle, ACM Press, 1990, pp. 11-18.
13. Kandogan, E. and Shneiderman, B. Elastic Windows: A Hierarchical Multi-window World-Wide Web Browser. In Proceedings of ACM UIST'97 Symposium on User Interface Software and Technology. Banff, Canada: ACM Press, 1997, pp. 169-177.
14. 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.
15. Sarkar, M., and Brown, M. Graphical Fisheye Views of Graphs. In Proceedings of CHI'92 Conference on Human Factors in Computing Systems. Monterey: ACM Press, 1992, pp. 83-91.
16. Shell, J., Vertegaal, R, and Skaburskis, A. EyePliances: Attention-Seeking Devices that Respond to Visual Attention. In Extended Abstracts of ACM CHI '03 Conference on Human Factors in Computing Systems, 2003, pp. 770-771.
17. Shell, J., Selker, T., and Vertegaal, R. Interacting with Groups of Computers. In Special Issue on Attentive User Interfaces, Communications of ACM Vol. 46 No. 3, 2003, pp. 40-46.
18. Sibert L., and Jacob, J. Evaluation of Eye Gaze Interaction. In Proceedings of CHI'00 Conference on Human Factors in Computing Systems. The Hague: ACM Press, 2000, pp. 281-288.
19. Smith, D., Irby, C., et al. Designing the Star User Interface. BYTE 7(4), 1982.
20. Starker, I., and Bolt, R. A Gaze-Responsive Self-Disclosing Display. In Proceedings of CHI'90 Conference on Human Factors in Computing Systems. Boston: ACM Press, 1990, pp. 3-9.
21. Tobii Systems. http://www.tobii.se, 2003.
22. Vertegaal, R. Attentive User Interfaces. Editorial, Special Issue on Attentive User Interfaces, Communications of ACM Vol. 46, No. 3, 2003, pp. 31-33.
23. Wang, J., Zhai, S. and Su, H. Chinese Input with Keyboard and Eye-Tracking - An Anatomical Study. In Proceedings of ACM CHI'01 Conference on Human Factors in Computing Systems, 2001, pp. 349-356.
24. Ware, C., and Mikaelian, H.T. An Evaluation of an Eye Tracker as a Device for Computer Input. In Proceedings of the ACM CHI + GI'87 Human Factors in Computing Systems Conference. Toronto, Canada: ACM Press, 1987, pp. 183-188.
25. 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.
26. Zhai, S., Morimoto, C., and Ihde, S. Manual and gaze input cascaded (MAGIC) pointing. In Proceedings of the ACM CHI'99 Conference on Human Factors in Computing Systems, 1999, pp. 246-253.