GUIDe: Gaze-enhanced UI Design

Manu Kumar
Stanford University, HCI Group
Gates Building, Room 382
353 Serra Mall
Stanford, CA 94305-9035
sneaker@cs.stanford.edu

Terry Winograd
Stanford University, HCI Group
Gates Building, Room 388
353 Serra Mall
Stanford, CA 94305-9035
winograd@cs.stanford.edu

Abstract

The GUIDe (Gaze-enhanced User Interface Design) project in the HCI Group at Stanford University explores how gaze information can be effectively used as an augmented input in addition to keyboard and mouse. We present three practical applications of gaze as an augmented input for pointing and selection, application switching, and scrolling. Our gaze-based interaction techniques do not overload the visual channel and present a natural, universally-accessible and general purpose use of gaze information to facilitate interaction with everyday computing devices.

Keywords

Pointing and Selection, Eye Pointing, Application Switching, Automatic Scrolling, Scrolling, Eye Tracking.

ACM Classification Keywords

H5.2. User Interfaces: Input devices and strategies, H5.2. User Interfaces: Windowing Systems, H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

Introduction

The keyboard and mouse have long been the dominant forms of input on computer systems. Eye gaze tracking as a form of input was primarily developed for disabled users who are unable to make normal use of a keyboard and pointing device. However, with the increasing accuracy and decreasing cost of eye gaze tracking systems [1, 2, 6] it will soon be practical for able-bodied users to use gaze as a form of input in addition to keyboard and mouse – provided the resulting interaction is an improvement over current techniques. The GUIDe (Gaze-enhanced User Interface Design) project [9] in the HCI Group at Stanford University explores how gaze information can be effectively used as an augmented input in addition to keyboard and mouse. In this paper, we present three practical applications of gaze as an augmented input for pointing and selection, application switching, and scrolling.

Motivation

Human beings look with their eyes. When they want to point (either on the computer or in real life) they look before they point [8]. Therefore, using eye gaze as a way of interacting with a computer seems like a natural extension of our human abilities. However, to date, research has suggested that using eye gaze for any active control task is not a good idea. In his paper on MAGIC pointing [18] Zhai states that "*to load the visual perception channel with a motor control task seems fundamentally at odds with users' natural mental model in which the eye searches for and takes in information and the hand produces output that manipulates external objects. Other than for disabled users, who have no alternative, using eye gaze for practical pointing does not appear to be very promising.*"

In his paper in 1990, Jacob [7] states that: "*what is needed is appropriate interaction techniques that incorporate eye movements into the user-computer dialogue in a convenient and natural way.*" In a later paper in 2000, Sibert and Jacob [15] conclude that: "*Eye gaze interaction is a useful source of additional input and should be considered when designing interfaces in the future.*"

For our research we chose to investigate how gaze-based interaction techniques can be made *simple*, *accurate* and *fast enough* to not only allow disabled users to use them for standard computing applications, but also make the threshold of use low enough that able-bodied users will actually prefer to use gaze-based interaction.

Related Work

Researchers have tried numerous approaches to gaze based pointing [3, 7, 10-13, 16-18]. However, its use outside of research circles has been limited to disabled users who are otherwise unable to use a keyboard and mouse. Disabled users are willing to tolerate customized interfaces, which provide large targets for gaze based pointing. The lack of accuracy in eye-based pointing and the performance issues of using dwell-based activation have created a high enough threshold that able-bodied users have preferred to use the keyboard and mouse over gaze based pointing [8].

The use of eye gaze for window management was discussed as far back as 1981 by Bolt [4]. In EyeWindows [5], Fono and Vertegaal explored two gaze-based window management techniques with the constraint that all windows be non-overlapping.

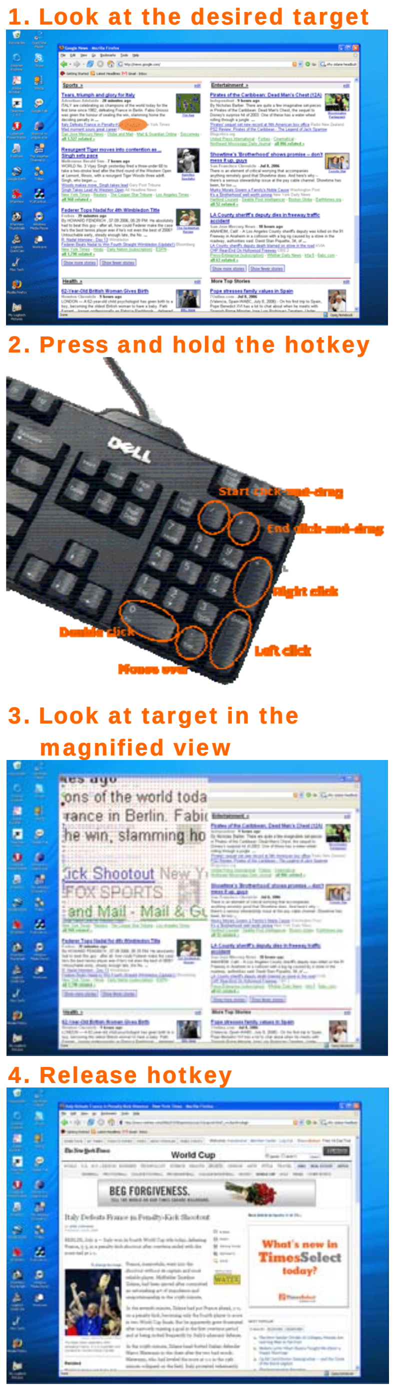

Figure 1. EyePoint fluidly combines a two-step progressive refinement of the target into a Look-Press-Look-Release action

Gaze-based scrolling is fraught with Midas Touch[1] [7] issues and therefore there hasn't been much in the literature regarding gaze-based scrolling to date.

EyePoint: Pointing and Selection

EyePoint provides a practical gaze-based solution for everyday pointing and selection using a combination of gaze and keyboard. EyePoint works by using a two-step, progressive refinement process that is fluidly stitched together in a look-press-look-release cycle.

To use EyePoint, the user simply looks at the target on the screen and presses a hotkey for the desired action - single click, double click, right click, mouse over, or start click-and-drag[2]. EyePoint displays a magnified view of the region the user was looking at. The user looks at the target again in the magnified view and releases the hotkey. This results in the appropriate action being performed on the target (sidebar).

To abort an action the user can look away or anywhere outside of the zoomed region and release the hotkey, or press the *Esc* key on the keyboard.

The region around the user's initial gaze point is presented in the magnified view with a grid of orange dots overlaid. These orange dots are called *focus points* and may aid in focusing the user's gaze at a point within the target. Focusing at a point reduces the jitter and improves the accuracy of the system.

Single click, double click and right click actions are performed as soon as the user releases the key. Click and drag is a two-step interaction. The user first selects the starting point for the click and drag with one hotkey and then the destination with another hotkey.

Technical Details

The eye tracker constantly tracks the user's eye-movements[3]. A modified version of Salvucci's Dispersion Threshold Identification fixation detection algorithm [14] is used along with our own smoothing algorithm to help filter the gaze data. When the user presses and holds one of the action specific hotkeys on the keyboard, the system uses the key press as a trigger to perform a screen capture in a *confidence interval* around the user's current eye-gaze. The default settings use a confidence interval of 120 pixels square (60 pixels in all four directions from the estimated gaze point). The system then applies a *magnification factor* (default 4x) to the captured region of the screen. The resulting image is shown to the user at a location centered at the previously estimated gaze point but offset to remain within screen boundaries.

The user then looks at the desired target in the magnified view and releases the hotkey. The user's eye gaze is recorded when the hotkey is released. Since the view has been magnified, the resulting eye-gaze is

[1] Unintentional activation of a command when a user is looking around.

[2] We provide hotkeys for all standard mouse actions in our research prototype; a real world implementation may use alternative activation techniques such as dedicated buttons located below the spacebar.

[3] If the eye tracker were fast enough, it would be possible to begin tracking when the hotkey is pressed, alleviating long-term use concerns for exposure to infra-red illumination.

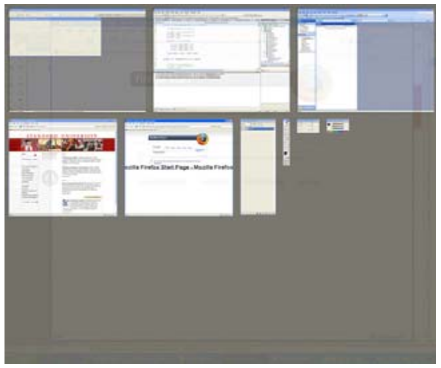
Figure 2. Pressing and holding the EyeExposé hotkey tiles all open applications on the screen. The user simply looks at the desired target application and releases the hotkey.

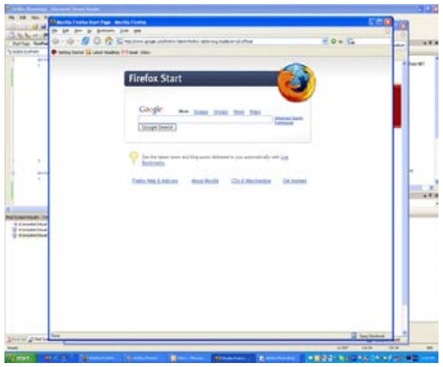
Figure 3. Releasing the hotkey restores the windows to their original size and brings the selected application to the foreground.

more accurate by a factor equal to the magnification. A transform is applied to determine the location of the desired target in screen coordinates. The cursor is then moved to this location and the action corresponding to the hotkey (single click, double click, right click etc.) is executed. EyePoint therefore uses a secondary gaze point in the magnified view to refine the location of the target.

Evaluation
A quantitative evaluation of EyePoint shows that the performance of EyePoint is similar to the performance of a mouse, though with slightly higher error rates. Users strongly preferred the experience of using gaze-based pointing over the mouse even though they had years of experience with the mouse.

EyeExposé: Application Switching

EyeExposé, combines a full-screen two-dimensional thumbnail view of the open applications with gaze-based selection for application switching.

Figure 2 and Figure 3 show how EyeExposé works - to switch to a different application, the user presses and holds down a hotkey. EyeExposé responds by showing a scaled down view of all the applications that are currently open on the desktop. The user simply looks at the desired target application and releases the hotkey.

The use of eye gaze instead of the mouse for pointing is a natural choice. The size of the tiled windows in Exposé is usually large enough for eye-tracking accuracy to not be an issue. Whether the user relies on eye gaze or the mouse for selecting the target, the visual search task to find the desired application in the tiled view is a prerequisite step. By using eye gaze with an explicit action (the release of the hotkey) we can leverage the user's natural visual search to point to the desired selection.

Evaluation
A quantitative evaluation showed that EyeExposé was significantly faster than using Alt-Tab when switching between twelve open applications. Error rates in application switching were minimal, with one error occurring in every twenty or more trials. In a qualitative evaluation, where subjects ranked four different application switching techniques (Alt-Tab, Task bar, Exposé w/mouse and EyeExposé), EyeExposé was the subjects choice for speed, ease of use, and the technique they said they would prefer to use if they had all four approaches available. Subjects felt that EyeExposé was more natural and faster than other approaches.

EyeScroll: Reading mode and Scrolling

EyeScroll allows computer users to automatically and adaptively scroll through content on their screen. When scrolling starts or stops and the speed of the scrolling is controlled by the user's eye gaze and the speed at which the user is reading. EyeScroll provides multiple modes for scrolling – specifically, a *reading mode* and off-screen dwell-based targets.

Reading Mode
The EyeScroll reading mode allows users to read a web page or a long document, without having to constantly scroll the page manually. The reading mode is toggled by using the Scroll Lock key – a key which has otherwise been relegated to having no function on the keyboard. Once the reading mode is enabled by pressing the Scroll Lock key, the system tracks the

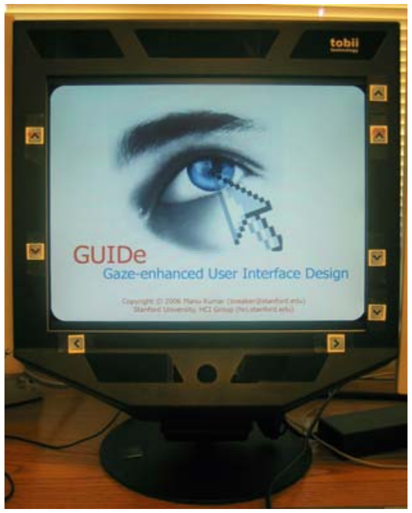

Figure 4. Eye-tracker (Tobii 1750) with off-screen targets for scrolling actions such as Home, End, Page Up, Page Down, Up/Down and Left/Right.

user's gaze. When the user's gaze location falls below a system-defined threshold (i.e. the user looks at the bottom part of the screen) EyeScroll starts to slowly scroll the page. The rate of the scrolling is determined by the speed at which the user is reading and is usually slow enough to allow the user to continue reading even as the text scrolls up. As the user's gaze slowly drifts up on the screen and passes an upper threshold, the scrolling is paused. This allows the reader to continue reading naturally, without being afraid that the text will run off the screen.

The reading speed is estimated by measuring the amount of time *t* it takes the user to complete one horizontal sweep from left to right and back. The delta in the number of vertical pixels the user's gaze moved (accounting for any existing scrolling rate) defines the distance *d* in number of vertical pixels. The reading speed can then be estimated as d/t.

In order to facilitate scanning the scrolling speed can also take into account the location of the user's gaze on the screen. If the user's gaze is closer to the lower edge of the screen, the system can speed up scrolling and if the gaze is in the center or the upper region of the screen the system can slow down scrolling accordingly. Since the scrolling speed and when the scrolling starts and stopped are both functions of the user's gaze, the system adapts to the reading style and patterns of each individual user.

By design the page is always scrolled only in one direction. This is because while reading top to bottom is a fairly natural activity that can be detected based on gaze patterns, attempting to reverse scrolling direction triggers too many false positives. Therefore, we chose to combine the reading mode with explicit dwell-based activation for scrolling up in the document (see below).

The user can disengage the reading mode by pressing the Scroll-Lock key on the keyboard.

Off-screen dwell-based targets
We placed multiple off-screen targets as seen in Figure 4 on the bezel of the screen. The eye-tracker's field of view is sufficient to detect when the user looks at these off-screen targets. A dwell duration of 400-450ms is used to trigger activation of the targets which are mapped to *Page Up*, *Page Down*, *Home* and *End* keys on the keyboard.

The off-screen dwell-based targets for document navigation complement the automated reading mode described above. Since the reading mode only provides scrolling in one direction, if the user wants to scroll up or navigate to the top of the document he/she can do that by using the off-screen targets.

Evaluation
Pilot studies showed that subjects found the EyeScroll reading mode to be natural and easy to use. Subjects particularly liked that the scrolling speed adapted to their reading speed. Additionally, they felt in control of the scrolling and did not feel that the text was running away at any point. Formal evaluation of EyeScroll is currently underway and results will be presented at a future conference.

Conclusion
We will demonstrate several practical examples of using gaze as an augmented input for interaction with computers. Our approach uses gaze as an input without overloading the visual channel and consequently enables interactions which feel natural and easy to use.

The true efficacy and ease of use of these techniques can only be experienced and description in words does not do them justice. We look forward to having CHI attendees try the techniques for themselves!

Acknowledgements

We would also like to acknowledge Stanford Media X and the Stanford School of Engineering matching funds grant for providing the funding for this work.

References

1. *IPRIZE: a $1,000,000 Grand Challenge designed to spark advances in eye-tracking technology through competition*, 2006. http://hcvl.hci.iastate.edu/IPRIZE/
2. Amir, A., L. Zimet, A. Sangiovanni-Vincentelli, and S. Kao. An Embedded System for an Eye-Detection Sensor. *Computer Vision and Image Understanding, CVIU Special Issue on Eye Detection and Tracking* 98(1). pp. 104-23, 2005.
3. Ashmore, M., A. T. Duchowski, and G. Shoemaker. Efficient Eye Pointing with a FishEye Lens. In Proceedings of *Graphics Interface*. pp. 203-10, 2005.
4. Bolt, R. A. Gaze-Orchestrated Dynamic Windows. *Computer Grahics* 15(3). pp. 109-19, 1981.
5. Fono, D. and R. Vertegaal. EyeWindows: Evaluation of Eye-Controlled Zooming Windows for Focus Selection. In Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 151-60, 2005.
6. Hansen, D. W., D. MacKay, and J. P. Hansen. Eye Tracking off the Shelf. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Antonio, Texas, USA: ACM Press. pp. 58, 2004.
7. Jacob, R. J. K. The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At is What You Get. In Proceedings of *ACM Transactions in Information Systems*. pp. 152-69, 1991.
8. Jacob, R. J. K. and K. S. Karn, Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises, in *The Mind's eye: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyona, R. Radach, and H. Deubel, Editors. Elsevier Science: Amsterdam. pp. 573-605, 2003.
9. Kumar, M., *GUIDe: Gaze-enhanced User Interface Design*, 2006. Stanford. http://hci.stanford.edu/research/GUIDe
10. Lankford, C. Effective Eye-Gaze Input into Windows. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. Palm Beach Gardens, Florida, USA: ACM Press. pp. 23-27, 2000.
11. Majaranta, P. and K.-J. Räihä. Twenty Years of Eye Typing: Systems and Design Issues. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. New Orleans, Louisiana, USA: ACM Press. pp. 15-22, 2002.
12. Miniotas, D., O. Špakov, I. Tugoy, and I. S. MacKenzie. Speech-Augmented Eye Gaze Interaction with Small Closely Spaced Targets. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Diego, California, USA: ACm Press. pp. 67-72, 2006.
13. Salvucci, D. D. Intelligent Gaze-Added Interfaces. In Proceedings of *CHI*. The Hague, Amsterdam: ACM Press. pp. 273-80, 2000.
14. Salvucci, D. D. and J. H. Goldberg. Identifying Fixations and Saccades in Eye-Tracking Protocols. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. Palm Bech Gardens, Florida, USA: ACM Press. pp. 71-78, 2000.
15. Sibert, L. E. and R. J. K. Jacob. Evaluation of Eye Gaze Interaction. In Proceedings of *CHI*. The Hague, Amsterdam: ACM Pres, 2000.
16. Wang, J., S. Zhai, and H. Su. Chinese Input with Keyboard and Eye-Tracking - An Anatomical Study. In Proceedings of *CHI*. Seattle, Washington, USA: ACM Press. pp. 349-56, 2001.
17. Yamato, M., A. Monden, K.-i. Matsumoto, K. Inoue, and K. Torii. Button Selection for General GUIs Using Eye and Hand Together. In Proceedings of *AVI*. Palermo, Italy: ACM Press. pp. 270-73, 2000.
18. Zhai, S., C. Morimoto, and S. Ihde. Manual and Gaze Input Cascaded (MAGIC) Pointing. In Proceedings of *CHI*. Pittsburgh, Pennsylvania, USA: ACM Press. pp. 246-53, 1999.