Enhancing Gaming Interfaces via Gaze Tracking

Harsh Baid

Bronx High School of Science

Abstract

Human-Computer Interaction (HCI) has been on the rise ever since the inception of the first modern computing machines. The introduction of a Graphical User Interface (GUI) has allowed individuals to express all their decisions to a machine via a simple mouse/keyboard. As technology has evolved, so has our need for more sophisticated interfaces to support a wide spectrum of human activity. To this extent, Natural User Interfaces (NUI's) are suggested as a key next step in the process.

NUI's improve on current interfaces by removing the need to obtain user input through a physical utility (ie: a mouse), but rather by remaining effectively invisible and natural during interaction. Today many devices exist that allow for somewhat of a "natural" HCI such as the Microsoft's Kinect and Facebook's Oculus Rift. The problem with these devices, however, are that they involve some form of intentional/tethered physical communication with the device in order to interact with the machine.

To this extent, the advancement/development of eye tracking (also regarded as gaze tracking) is suggested to serve as a replacement for these devices to accomplish a NUI. Eye tracking offers several benefits including a completely un-tethered and effectively invisible input device, making the applications that incorporated it seem natural in operation. To demonstrate the impact of the technology as an NUI, a modification to a sandbox game called Minecraft was developed incorporating eye tracking. The EyeTribe Tracker was used to effectively replace the function of the mouse in-game, but rather the eyes to interact with the environment. This was done to illustrate a proof of concept of the impact, eye tracking technology can have.

1 Introduction

Human-computer interaction (HCI) has been on the rise ever since the inception of the first modern computing machines. With the introduction of a ubiquitous Graphical User Interface (or GUI), individuals have been able to express all of their decisions to a machine via a simple keyboard and mouse. Several decades later, computers have massively changed and so has our interaction with them. Even today, designers and programmers are encouraged to meet the needs of users which often involves more sophisticated interfaces for applications that support a wide spectrum of human activity. The introduction of technologies such as touch screen and voice recognition has allowed companies to offer more immersive user interfaces in their products.

Current Technologies

Natural User Interfaces (NUI's) have become a necessary next step in the development of immersive systems to improve HCI. While traditional interfaces usually involve the input of a user to any artificial control device (ie: a mouse), a NUI relies on a seamless and untethered method of obtaining user input. The word "natural" refers to the goal that is used to describe the overall user experience in that the design should work "naturally" while allowing users to quickly transition from novice to expert [1].

Given this model, many devices exist today that incorporates a multitude of sensors combined with software to detect gestures which allow for "natural" communication with the machine. Some of these devices include Microsoft's Kinect, Facebook's Oculus Rift, Google's Glass, and the Leap Motion. These devices, along with the programmers responsible for their various applications, have become revolutionary in advancing the progress towards a more NUI and ultimately a better user experience.

Microsoft's Kinect and the Leap Motion are such examples of devices that incorporate a Kinetic User Interface (KUI) which translates to interfaces that interact with computers through the motion of objects and bodies [2]. The Kinect, in particular, has become a very popular commercialized KUI that implements an RGB camera, depth sensor, multi-array microphone in conjunction with proprietary software that allows for 3D motion capture, facial/body recognition, and voice recognition capabilities [3]. In combination with the Kinect SDK (Source Development Kit), these features have allowed developers to create games that often involve gestures such as running and jumping to improve interaction/immersivity with the game itself. The Leap Motion, similarly, is another example of a KUI that incorporates two monochromatic infrared cameras and three infrared LEDs to generate a set of 3D position data effectively allowing hand/finger motion tracking in a 3D space [4]. This tracking has allowed developers to create applications that allow control of various computer functionalities through simple hand motions such as flicks and pinch-to-zoom gestures. While both the Leap Motion and the Kinect offer great benefits for a KUI through active motion tracking, the resulting physical bodily movement is intrusive and somewhat contradictive of a seamless and invisible NUI.

Facebook's Oculus Rift and Google's Glass are examples of a Reality-based User Interface (RUI) which often involves wearable computing to render real-world objects "clickable" and interaction occurs at real-time [5].The Oculus Virtual-Reality (VR) headset incorporates a flat 7-inch 720p LCD Display attached to two lenses (one for each eye) as well as an accelerometer, gyroscope, and magnetometer which is used in conjunction with the Oculus SDK to enable developers to incorporate 3D immersive environments in their applications. The data received from the three incorporated sensors is combined through a process called sensor fusion to enable fast and accurate tracking of your head orientation and synchronization effectively generating a virtual reality (VR) enviroment in real-time [6]. Google's Glass, similarly, is another example of an RUI and more specifically an Augmented Reality Interface (AR). The Glass features a field-sequential color, LED illuminated display which reflects light onto a Liquid Crystal on Silicon (LCoS) at a 45° angle creating the effect of a transparent display [7]. This display used in junction with a bone-conducting speaker and 720p camera simulates an overlay of information onto the physical world which allows for AR simulated by the Glass. While both the Glass and the Oculus Rift have effectively demonstrated the progress towards a seamless and natural NUI through the development of an RUI, the technology is still somewhat questionable. The resulting head strap required to put on with the Oculus Rift and the frame attached to the Glass results in a tethered method of communication contradictive to an invisible/untethered NUI.

Gaze Tracking

One such technology that has often missed the public eye as an effective solution to the NUI dilemma is eye tracking (also regarded as gaze tracking). Eye tracking is the process of determining the user's point of gaze on a computer screen relative to the motion of his eyes related to the head through the use of an eye tracker which measures eye movements and eye position. Currently, four methods of eye tracking exist today [8]:

1. Scleral Search Coils
2. Infrared occulography (IROG)
3. Electro-occulography (EOG)
4. Video occulography (VOG)

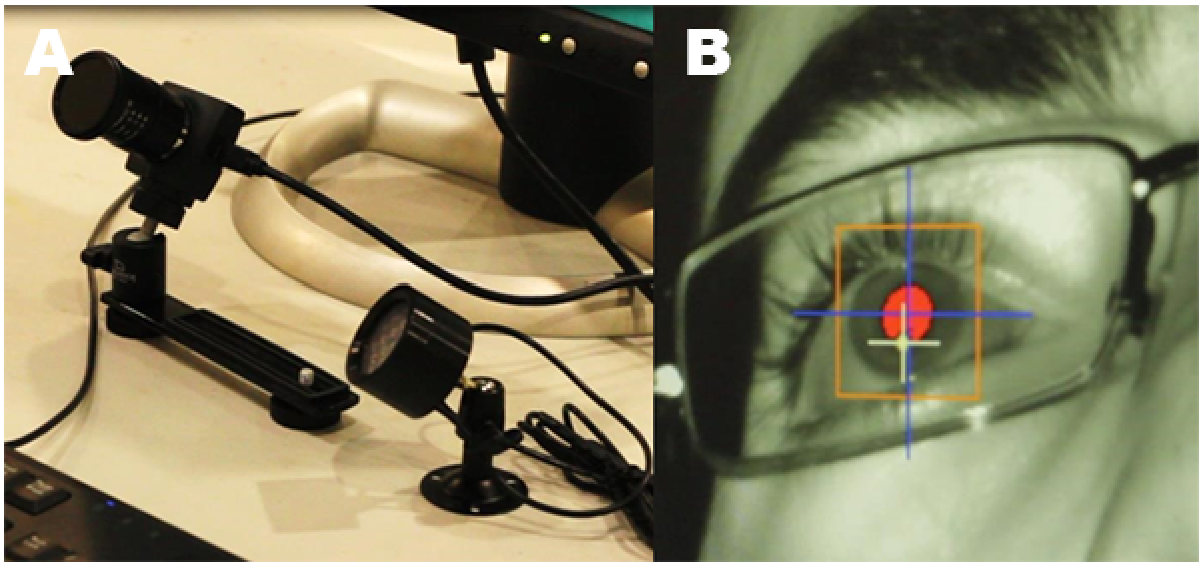

Figure 1: Eye tracker Setup [9]. (A) IR LED + Camera (B) Image from Gaze Tracker.

This paper, in particular, will specifically focus on VOG or video-based eye tracking. This method works on the properties of retro-reflectivity and corneal reflection to determine the users gaze. One or more infrared (IR) LED's are placed near the computer screen in conjunction with the camera that picks up on the corneal reflection or "glint" that the IR LED produces on the users pupil. IR LED's are used because they invisible to the human key preventing any unwanted distractions and can be easily picked up by cameras. The camera streams this video to the program which calculates a vector distance between the pupil center and corneal reflection to compute the gaze direction on a surface (see figure 1). This process usually involves a calibration mechanism beforehand which asks the user to fixate at certain points on the screen to determine the correct direction of gaze.

Currently there are two non-intrusive methods of corneal reflection and illumination via the IR LED's that are used in conjunction with eye tracking devices. Both of these methods involve the use of Pupil Center Corneal Reflection (PCCR) which causes visible reflections on the cornea (”glints”). Since a normal cornea has a near perfect sphere shape, the glint stays in the same position independent of the gaze direction [10]. The two methods are referred to as:

1. Bright Pupil Tracking
2. Dark Pupil Tracking

Bright Pupil Tracking involves placing the IR LED close to the optical axis of the camera which causes the pupil to appear bright/white similar to a phenomenon that causes red eye in photos [9]. Dark Pupil tracking, however, is directly the opposite and involves placing the IR RED away from the optical axis of the camera causing the pupil to appear darker than the Iris. Different factors can influence the method of pupil tracking used in remote eye tracking; the most common one being ethnicity. For Hispanics and Caucasians, the bright pupil detection method has been shown to works very well, while dark pupil method provides better tracking among the Asian population [11].

Eye tracking technology has the ability to serve as a good replacement to Microsoft's Kinect, Google's Glass, Facebook's Oculus Rift, and the leap motion in terms of an effective NUI. The technology itself is completely seamless while avoiding any additional external actions such as bodily movement and ”natural” in that it entirely operates on the direction of the user's gaze making it invisible. To this extent, the demonstration of the impact this technology can have on interfaces was shown through the implementation of eye tracking into a popular indie sandbox game, Minecraft. The modification was made as proof-of-concept to show the impact that eye tracking can have on technology.

Related Work

Interaction through gaze became a possibility with the advancements in the design of eye tracking systems and an increase in processing speeds of computers. Bolt, an MIT researcher, was the one of the first to foresee the applications of eye based interaction with computers. In a paper titled "Gaze-orchestrated dynamic windows" /citesource12, he described a media room with 15 to 20 big displays streaming dynamic content simultaneously, which he called "World of Windows". In his paper, he described two methods for eye-based interaction. His first method was to use dwell time, or how long the user looked at a particular display or window. His second method was to use an additional modality like a speech command to activate the zoom, based on the content the user is looking at.

In 1990, Robert J.K. Jacob introduced one of the first eye-based interactive systems, by demonstrating an intelligent gaze-based informational display [13]. In his system, a text window would scroll to show information on visually selected items. Jacob's system was one of the first to use eye tracking technology interactively, and he is well-known for the identification of an important problem in eye-based interactive systems: the Midas Touch problem [13]. This problem occurs if the eyes are used as a selection tool like the mouse, because eyes do not technically turn "off". Everywhere the user looks activates a command, and the user cannot look anywhere without issuing a command. In order to bypass the Midas Touch problem, Jacob, in his paper, discussed a few possible solutions including the use of blinks, and finally promoting the use of dwell time as a selection mechanism.

In 1999, Zhai, Morimoto, and Ihde explored another methodology to gaze-based interaction [13]. They figured out a way to use gaze as a sort of predictive pointing aid rather than a direct effector of selection, which helps to lessen the effects of the Midas Touch problem. In their paper, they present an acceleration technique where a cursor is warped to the vicinity of a fixated target. The acceleration would either occur immediately by eye movement, or delayed until the mouse moves first. Users of this system subjectively reported that it felt faster than just using a mouse.

In 2000, Isokoski did the first research on gaze gestures using off-screen targets for text input [14]. His system required the user to visit the off-screen targets with their gaze in a certain order to enter characters. His form of gaze gestures was not scalable and could not be performed in any location like other gesture methods, because the off-screen targets forced the gesture to be performed in a fixed location and fixed movement size.

Later in 2010, Istance and group defined gaze gesture as “a definable pattern of eye movements performed within a limited time period, which may or may not be constrained to a particular range or area, which can be identified in real-time, and used to signify a particular command or intent” [15].

2 Design and Implementation

Apparatus

Several pieces of software and hardware were used in the implementation and design of the gaze tracking system into the sandbox game, Minecraft. The program itself was written in Java using the Java SE Development Kit 8u25 and tested in the Java SE Runtime Environment 8u25. The application, however, has also been found to be fully operational in both Java SE Runtime Environment 6 and 7. Eclipse v4.4.1 (Luna) was the integrated development environment (IDE) used to develop the program and has been compiled and tested to work with Windows 7, 8, 8.1 in both 32-bit and 64-bit systems. The EyeTribe Tracker was the device used for the gaze tracking system running alongside the TheEyeTribe Source Development Kit. Certain hardware specifications are needed to meet the application requirements. Specifically, a USB3.0 Port, an i5/i7 (or similiar), and 1GB of dedicated RAM.

Gaze Implementation

The device operates by sending a stream of gaze data to the EyeTribe tracker API (Application Programming Interface). After an initial calibration involving fixating to several changing points on the screen, the eye tracker return a constants stream of x and y coordinates indicating where the user is look on the screen relative to the top left corner being the origin.

Using this information, the program was created to simulate mouse movement (looking around) inside Minecraft by translating the x and y coordinates from the tracker to a point of interest in Minecraft and then gradually bringing that point to focus in front of the user. To enhance the user interface of gaze tracker implementation inside Minecraft, it was also decided that in the process of bringing the users point of interest on the screen to focus it would also involve gradually moving the focus (point) to the center of the screen since the center is regarded as the most “focused” point on the screen as well.

Several design challenges were involved in accomplishing this goal of moving the point of interest into focus and towards the center of the screen. The initial challenge we encountered was the process of determining the user’s point of interest on the screen. According to Jacob, the Midas Touch Problem provided the answer by simply using dwell time to determine if the gaze coordinate was a point of interest to the user and then simply moving the perspective of the screen. This, however, proved to be nearly impossible since the application had not taken into account the eye saccades that had occurred when the user was looking around. Eye Saccades, a fast movement of the eye, resulted in constantly changing x and y coordinates and therefore the dwell time mechanism proved to be fundamentally useless since the eyes could not fixate on a certain point for an extended period of time. Our solution to this error then became to simply create a buffer inside the program that would constantly calculate the standard deviation of gaze coordinates at any given moment in the stream and send this data to the game which would then move the viewpoint of the player in-game to an average point resulting from the eye saccades. This again proved to be incompatible as the eye position still kept changing, except at a lower rate, and still didn’t give the program enough time to determine if the user was fixating on his point of interest or simply looking around. Eventually, we arrived at our final solution to this issue which became to simply take the first coordinate retrieved from the gaze tracker, calculate a circle around that point with a radius of 50 pixels. Then check every other coordinate to see if it was inside the range, which would then recalculate the circle if the coordinate was outside the range. This would allow the program to account for the eye saccades that occurred and determine if the user was fixating on a general area point or looking around.

3 Discussion/Conclusion

Eye tracking for human-computer interaction and user experience in the mass market consumer electronics is still in its infancy. The implementation of the game Minecraft was simply done to show as a proof-of-concept the potential ability this technology has to serve as a completely NUI. There is, however, still a lot of work to be done to the user interface, robustness of the technology, and the limitations it presents, before it can be implemented and accepted in actual consumer devices. Some of the limitations that eye tracking presents is that it can not track the user's peripheral vision and only records/displays foveal fixations in the sharpest part of our visual field. Although most of the time our fixations represent our interest in the target this may not always be the case which could be result in a misinterpretation by the eye tracler. Another potential limitation of the technology is that the IR LED responsibly for calculating gaze direction might start to irritate the eye after prolonged use which could result in an ultimately instrusive interface. These limitations need to be resolved in order to get eye tracking technology in cosnumer hands. The first step to get it accepted into the consumer market is lowering the cost, and that is exactly what is happening currently in the eye tracking field.

It was only 10 years ago that no one had smartphones with touch screens. Many companies made attempts to market touch screen technology, but it never really caught on until Apple introduced a pre- calibrated system for touch control that was sufficiently robust, accurate, and affordable. Eventually, touch control technology became mainstream. Eye tracking technology is the same and is still very early in its technology lifecycle, but it is currently poised to go through a transition or technological evolution.

References

[1] Steinberg, G. (2012, May 1). Natural User Interfaces. University of Auckland.

[2] Bruegger, P., & Hirsbrunner, B. (2009). Kinectic User Interface: Interaction through Motion for Pervasive Computing Systems.

[3] Totilo, S. (2010, July 10). Natal Recognizes 31 Body Parts, Uses Tenth Of Xbox 360 "Computing Resources" Retrieved from http://kotaku.com/5442775/natal-recognizes-31-body-parts-uses-tenth-of-xbox-360-computing-reso

[4] raver1975. (2013, Jan). Leap Motion Structured Light Pattern [Video file]. Retrieved from http://www.youtube.com/watch?v=UI5EBzU_QqM

[5] Poh, M. (n.d.). 8 Next-Generation User Interface That Are (Almost) Here. Retrieved from http://www.hongkiat.com/blog/next-gen-user-interface/

[6] Johnson, B. (n.d.). Technical Specifications: Cracking Open the Rift - How the Oculus Rift Works.

[7] Guttag, K. (2013, June 1). Proof That Google Glass Uses A Himax LCOS Microdisplay. Retrieved from http://seekingalpha.com/article/1504292-proof-that-google-glass-uses-a-himax-lcos-microdisplay

[8] Singh, H. & Singh, J. (2012). Human Eye Tracking and Related Issues: A Review. International Journal of Scientific and Research Publications.

[9] Ettikkalayil, J. (2013). Design, Implementation, and Performance Study of an Open Source Eye-Control System to Pilot a Parrot AR.Drone Quadrocopter.

[10] Drewes, H. (2010). Eye Gaze Tracking for Human Computer Interaction. A Dissertation submitted in the partial fulfillment of the Ph.D. Degree

[11] Tobii. (2010). Tobii Eye Tracking: An introduction to eye tracking and Tobii Eye Trackers. Retrieved from http://www.ehow.com/how_5829209_diagnose-eye-tracking-problems.html

[12] Bolt, R.A. (1981). Gaze-orchestrated dynamic windows. In Proceedings of the 8th annual conference on Computer graphics and interactive techniques (SIGGraph "81).

[13] Duchowski, A.T. (2002). A Breadth-First Survey of Eye Tracking Applications. Behavior Research Methods, Instruments, and Computers.

[14] Isokoski, P. (2000). Text Input Methods for Eye Trackers Using Off-Screen Targets. In Proceedings of the 2000 Symposium on Eye Tracking Research & Applications.

[15] Istance, H., Hyrskykari, A., Immonen, L., Mansikkamaa, S., & Vickers, S. (2010). Designing gaze gestures for gaming: an investigation of performance. Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications.