Controlling Computer Applications Using an Eye Tracking Device

David Stark

BSc Computing (Industry)

Session 2007/2008

Acknowledgements

I would like to thank the following people for their contributions to the project:

David Hogg for his help with the report style, and guidance throughout the project.

Ardhendu Behera for the numerous times he answered the silly questions that I just couldn't find the answer to.

A big thank you to Christina Howard, and her mystery assistant, who dedicated her time to me when I needed to test the eye tracker.

My housemates; Luke, Kyle and Filip for the relief and advice they provided when the project was just about to tip me over the edge, helping me to get back on track.

And finally to Thomas, who even through a brand new job, a house move and other problems in his life, still dedicated his time to help me proof read this report.

Summary

The current computer to user discourse is one sided, a computer providing a far higher level of bandwidth to the user than from the user to the computer. The user is limited by the speed of the input devices he or she uses and not by the rate they would like to input to a computer.

One of the most agile and precise organs of the human body is the eye, with almost instant reaction speeds when transferring a current gaze location to another, it is a prime candidate for input to a computer system.

The project harnesses the high level of bandwidth the eye provides to control the cursor on screen and thus give the user an alternative more intuitive way of controlling a computer system. The project also devises methods to control specific applications such as PowerPoint and shows the benefits of a 'hands-free' control environment.

An iterative method is used to develop the eye tracker, beginning with simple tasks such as moving the cursor on screen, and with each iteration passed, increasing the functionality of the system. The end result is a set of prototype applications that show the great potential in using the eye as form of computer input.

Contents

1 Project Overview

1.1 Introduction

This project investigates the use of an eye as an input medium on a modern day operating system where the user gaze is calculated by using an eye tracking device. Emphasis is placed on the appropriateness of using such a medium and whether a modern day operating system is capable of supporting such an input.

1.2 Aim and Objectives

The aim of this project is to further previous work in the field of computer vision and attempt to translate previous success of interaction using the eye to the modern day operating system.

The objectives of this project are to:

- Develop a system that will translate the users gaze into a cursor movement on screen
- Emulate standard mouse controls using just a users eyes
- Learn the merits and drawbacks of a ‘mouse free’ environment
- Learn about computer vision, eye trackers and the potential of these

1.3 Minimum Requirements

The minimum requirements (all using the eye tracker) of the project are to:

- Develop a mouse driver that is able to move the cursor on screen and perform a simple click
- Be able to control a PowerPoint presentation moving slides back and forward
- Filter out erroneous movement (such as a blink)

Possible extensions include

- Add more control to the PowerPoint presentation
- Develop an additional method of left click
- Develop a novel application of the eye tracker (such as image extraction)
- Allow the eye tracker sensitivity to be configurable

The novel application is described in section 6.4 as the application was not entirely conceived until later in the development of the project.

1.4 Deliverables

The deliverables of this project are as follows:

7/12/2007 - Midterm report

14/03/2008 - Project Demonstration

23/04/2008 - Full Project Report

2 Project Management

2.1 Methodology Choice

The software development process chosen for this project is that of the Iterative methodology. This is a notable change from the original methodology proposed for this project (The Waterfall model). Very soon into the project it was noticeable that the Waterfall model was not appropriate for the project as its rigid nature prevented the developer from returning back to previous work and implementing any improvements found; If the developer did so, then there would have been little use in following the model in the first place.

Instead the developer decided that using an Iterative methodology is more appropriate for a project of this type as the iterative methodology uses a set of specific critical requirements to define the project, and then requires an order in which they are to be developed (Graham 1992) thus allowing the developer to create a modular like system by which the fundamental areas are built first, then additional functionality coded on top. This is useful for a project of this nature as the developer has not attempted any computer vision projects previous to this, nor has found any reliable approaches to a computer vision project. The first iteration can be used as a valuable learning experience where the developer can gain more knowledge of how the eye tracker functions and the additional work to create a functional system.

The initial iterative steps devised are as follows:

1. Background Reading
2. An Initial Design of the System
3. First Iteration – Movement of the Cursor
4. Analysis and possible Improvements
5. Second Iteration – Clicking Functions Added
6. Analysis and possible improvements
7. Third Iteration – Control PowerPoint
8. Analysis and Possible Improvements
9. Possible Further Iterations and Improvements

2.2 Schedule

2.2.1 Planned Schedule

The below schedule (fig 1) is the initial schedule included in the midterm report (see appendix XXX). As the project initially took a swift change in approach and limitations of the hardware, a new schedule was created to take into account a new methodology and other consequent changes.

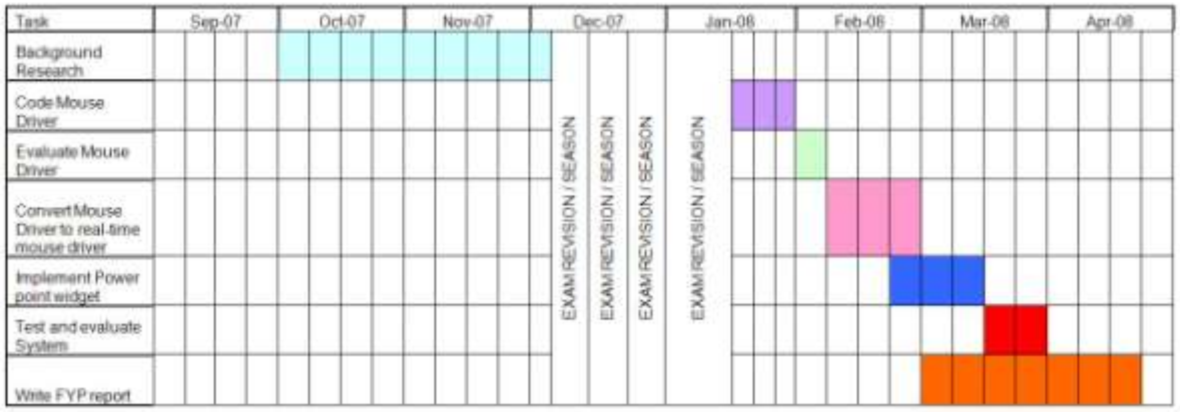


Fig 1. Original Project Schedule

2.2.2 Modified Schedule

Fig 2 depicts the schedule changed to reflect the change in Methodology and additional iterations added whilst creating the system.

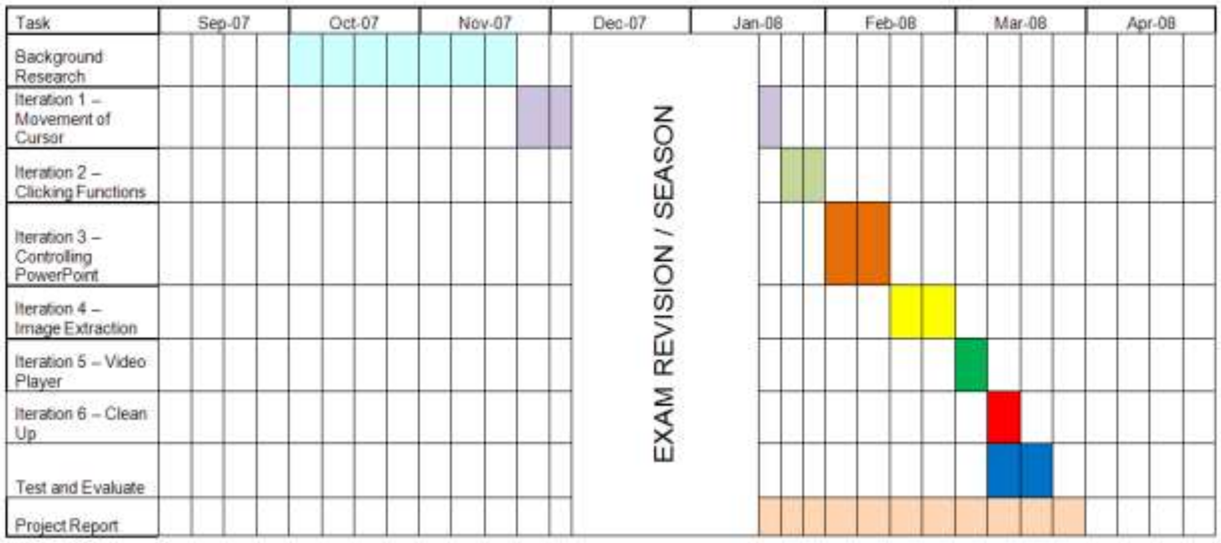


Fig 2. Final Project Schedule

3 Background Research

3.1 Problem Overview

Current computer to user discourse is one sided, the computer providing a far higher level of bandwidth to the user than from the user to the computer, with the current and most popular method of controlling actions upon a computer environment being that of the mouse.

The very first mouse was created in 1963 by Douglas Engelbart of the Stanford Research Institute (Barnes 1997), a bulky device that used the manual rotation of wheels to translate to x and y axis movements. As time has progressed, mice have come more ergonomic, responsive (and expensive) with the addition of roller balls, Light Emitting Diodes (LED's) and most recently, lasers.

For a standard mouse there are 4 actions needed to be performed to move the cursor on screen to a specified location:

1. Locate the object on screen (interchangeable with 2.)
2. Locate the mouse
3. Move the mouse to the location
4. Verify cursor is at correct location

Each point above represents an amount of time in terms of responsiveness, whether the time associated is seconds or milliseconds, there is a distinct delay between the users wish to do something and the time it takes to get there. The steps above can be generalised into two distinct action groups:

- eye movement
- hand movement

Eye movements can be almost classed as being an instinctive action, one that is beyond the necessity to think first. However hand movements are more deliberate, there is a definite thought process before an intentional action occurs.

This begs the question is it possible to make the dialogue between a human and a computer more efficient by removing the thought and physical movement process, and replace with a communication route, in which movement of the cursor on screen can be controlled with the instinctive and fast movements of the eye.

Many studies have tried to answer this question, (Ware 1987), (Jacobs 1991) and (Sibert 2000) however the majority of the studies are using self created applications to test the effectiveness. What hasn't been explored is if using an eye tracker is a good candidate to control a modern day operating system, such as Microsoft XP and the numerous applications that can run upon it. The report will seek to answer this question and expand further on the benefits of the eye as an input device.

3.2 Related Projects

'An Evaluation of an Eye Tracker as a Device for Computer Input' (Ware 1987) was a novel computer vision research paper to the extent that Ware was the first to use an eye tracker as an input device to a computer for uses other than disability access. In computing terms the paper is 'old' but the problems addressed and the design ideas are still relevant to today's eye tracking projects. Ware immediately highlights the potential and drawbacks of using the eye as a form of input focusing on the eyes inherently jittery nature, thus causing issues for accuracy. Ware analyses the eye tracker system in terms of Fitts's Law (a model for the act of pointing and usually incorrectly spelt Fitts' Law), noting that the main difference in speed between a mouse and eye tracker is the time taken for the user to physically grab the mouse. Upon testing the eye tracking system Ware notes that the size of the icon to be selected is important, stating that the targets should subtend at least one degree of visual angle else expect the system to lose accuracy and performance as the jittery nature of the eye will prevent a system being able to focus on the required object reliably.

Ware experimented using different implementations of an eye tracker input system ranging from a Gaze method, where the user simply gazes at the object to be selected for a fixed time to a hardware button where the user move the cursor with their eyes but is required to press a switch to click. The paper concludes that the Gaze method is most appropriate to those with physical disabilities that would prevent them pressing a hardware button, but if feasible the hardware button is preferred albeit it with sufficient practice time.

In 1991, Jacobs continued the research into using eye trackers for computer interaction with a paper entitled "The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At is What You Get" (Jacobs 1991). Jacob reiterates many of Ware's discoveries concerning the limitations of an eye tracker as a form of input, but highlights an important drawback of a simple ‘look and get’ (gaze clicking) approach, Jacobs naming the problem “The Midas Touch” after the Greek mythological story. In essence Jacobs is suggesting that it is at first empowering to look and then immediately get what you want simply by looking at it (in this case an icon), however it soon becomes an inconvenience as without preventative measures commands will occur at nearly everything a user looks at or concentrates on.

Jacobs’s paper furthers Ware’s investigations by delving into more day to day tasks that are performed on a modern day operating systems, such as menu selection and the scrolling of text in a window. Jacobs reports the system as a great success and notes that the system appears to be reading the users thoughts (if working well). The paper also discusses the accuracy of the eye tracking solution and draws comparison to that of a touch screen, though the report was written in 1991 and as with nearly all technologies, significant improvements have been made, such as the impressive Microsoft Surface technology that uses a multitude of different technologies for a fluent interactive experience (Microsoft, Microsoft Surface 2007). Finally Jacobs approach was different from previous solutions as Jacobs developed a management system that would control the eye tracker, keyboard and physical mouse. This took tokens (sets of coordinates / keyboard inputs) that were analysed, computed and sometimes modified before the action is performed on screen. This allowed for a greater ability to combine the 3 mediums and, with credit to Jacob, to great effect, there was however a reported small delay between the users eye movement and the action performed on screen.

The paper ‘Evaluation of Eye Gaze Interaction’ (Sibert 2000) is a follow up from the Jacobs paper, investigating further the strengths and weaknesses of the aforementioned system by performing greater controlled tests. These test included lengthy testing periods where the eye would be put under a greater level of strain than the tests in (Jacobs 1991). The results showed that the eye tracker was again faster than using a physical mouse and Sibert comments on the benefits of using the eye as an addition communication channel. Sibert does continue to state that although the benefits of an eye tracker can be seen the cost is still an issue with many eye tracking systems costing $15,000 upwards. The paper claims nothing but great success for the eye tracker, but Sibert does not mention if the selection algorithms were subject to false positives and false negatives, and the inherent costs to the user. Sibert closes the paper with a recommendation that interfaces in the future should and likely will be designed with using the eyes as a sole or part input medium.

The above studies have all focused on lab conditioned tests where an eye tracker is used to improve response times for those who have the appropriate motor skills to control a physical mouse, with the expectation of (Ware 1987) who continually references back to original studies that used eye tracking primarily to give greater communication opportunities for the disabled. Barreto published a paper on real-time assistive computer interfaces for users with motor disabilities (Barreto and Scargle 1999) researching various alternatives to a physical mouse for disabled users. Although the study focuses primarily on using electroencephalographic (EPG) waves that originate from the brain, a sizeable portion of the paper is dedicated to using eye trackers. Barreto takes quite a negative standpoint on the use of eye trackers, noting that whilst the improvement in speed for selection is of worth, there is a distinct lack of consistent accuracy and almost no ability for fine/small mouse movements. Barreto notes that positional placement of the head is important, as all but the most expensive eye trackers will lose their initial calibration values with just the slightest movement of the head; this in turn reduces the accuracy of the system calculating the users gaze location.

Although Barreto doesn’t take a favourable view of using an eye tracker as an alternative to a mouse, the negative points raised can be used to improve eye trackers and the software running behind them in the future.

3.2.1 Related Projects Summary

As time and technology progresses, those in the computing arena look to alternative inputs to control a computer, the majority of the research papers mentioned here believe that using the eye as an input can be the future, however (Barreto and Scargle 1999) take a more tentative approach claiming the benefit of a speed increase is counterbalanced by a distinct lack of control, accuracy and cost. Few reports have used an eye tracker to control a modern day operating system such as Windows XP and if using the eye were to become a serious input contender to replace a standard physical mouse more work has to be completed to find the suitableness of using eye trackers for this purpose.

3.3 The Eye

If the eye is to become a contender to replace the mouse as a form of input it is necessary to understand how the eye works, what limitations the eye may have and any problems that the eye could cause if it was used as an input medium.

The human eye is a complex sensory organ, providing the ability for a human to translate light waves into images via the brain (Atchison and Smith 2000). The eye works by directing focused light onto the retina, which will then turn will turn the light into electrical signals that are sent directly to the section of the brain that handles vision. Within the retina is an area named the Macula that specialises in high acuity vision capabilities (Ferkat and Weizer 2006). The high acuity of the Macula is in part to a small pit like structure called the fovea, that contains the highest density of cones located anywhere in the eye, giving it the ability to focus at a high resolution (Atchison and Smith 2000). Outside the fovea acuity reduces dramatically, up to 70% reduction in acuity can be had from only 10 degree's from the centre of the fovea, this means that for a person to fixate on an object, such as reading text in a book or tracking a moving object, the object in question has to be imaged on the fovea.

Focusing is not an energy free task; the eye eventually gets tired, the same way in which reading too long strains the eye. Ware discusses how long the eye can accurately fixate on a particular point, and concludes with approximately ten minutes (Ware 1987). The project will not expect a user to participate in a ten minute stare however there may be times where they eye will be in use for over ten minutes. Siberts' tests show that the users are able to use an eye tracker and focus on and off for over an hour, providing there are minute breaks between sessions (Sibert 2000).

Movement with the eye is very quick, with the most common method of moving the eye being a saccade, when a saccade occurs there is no correction for any errors in trajectory and this results in very fast response times from point to point.

3.4 Eye Tracking Devices

The eye tracking device to be used throughout the report is the Applied Science Laboratories Head Mounted Optic Mobile Eye product (shortened to Mobile Eye). The Mobile Eye determines the directions of the users gaze by simultaneously tracking the corneal reflection by shining an infrared light at the eye and the physical outline of the pupil. In addition a second camera attached to the Mobile Eye points in the same direction the user is facing. The visual line of gaze is computed from the relationship between the corneal reflection and the outline of the pupil (Applied-Science-Laboratories 2007).

The Mobile Eye has extensive output, recording the pupil radius and eye direction; however it is the coordinates of the users' gaze that will be the most important. The Mobile Eye has an accuracy of ±0.5 degrees and as such has the same accuracy as the system used in Ware's paper (Ware 1987).

3.5 Mouse Alternatives

3.5.1 Trackpad

The Trackpad (also known as touchpad) is the common input for laptop computers. The standard operation is to use a finger, as trackpads track the capacitance given from a users' finger, whilst other implements such as a stylus or pen, provide no capacitance for the touchpad to track.

The trackpad is similar in nature to that of a mouse, it is a relative motion device and will move the cursor on screen the same direction the finger is being dragged on the touchpad. Trackpads contain different methods for a user to click. Many have physical buttons (similar to a mouse) that give left, middle and right buttons, some others can use a physical tap to the keyboard that will imitate a left click.

Although trackpads aren't considered as easy to use as a mouse they have had a recent surge of popularity with the recent devices released by Apple that allow for multi finger tracking and simple gesture recognition to perform complex commands (Apple 2008).

3.5.2 Wii Remotes

The Nintendo Wii, released in November 2006, was a revolution in terms on next generation console units. The Wii, tried to focus on a new form of user interaction, attempting to move away from the standard controller based units (businessweek.com 2006). The Wii introduced a remote that can be used as a hand held pointing device, containing acceleration meters detecting the movement of the device in three dimensions.

The Wii remote has recently been adapted to work with a standard computer via the Bluetooth receiver contained within the remote. The mouse emulator has all the function of a standard 3 button mouse, allowing the user to scroll web pages, left click and right click (iHackEdge 2006). The preliminary results are promising, providing a fairly responsive alternative to a mouse. Others are following in iHackEdge's footsteps, Johnny Chung Lee providing alternate ways of using the remote as a sensor and not as a pointer, creating interfaces that mimic very portable whiteboards simply by pointing the remote and using LEDs that run at the same MHz as the Wii remote controller (J. C. Lee 2007).

The Wii remote was never intended for personal computer use, but it is swiftly becoming a viable alternative, especially with the increase of Home Media Centers that use personal computers to provide entertainment but still not provided a convenient method for controlling all aspects of the computer.

3.5.3 Joystick

The Joystick was a widely used computer peripheral, deriving its design and functionality from the flight controls of some planes (Herz 1997). The joystick is primarily a 2-D device, using the x and y axis for horizontal and vertical movement respectively. The joystick known today uses digital on/off signals to denote the direction currently being used, and will nearly always have one or more 'fire' buttons that are used to indicate an action.

Mouse control on a modern day Operating System is possible with a joystick but not without use of an additional application to translate joystick movement into on screen mouse movement. The use of which has been seen to improve mouse control of those with disabilities such as cerebral palsy (Rao and Seliktar 2000).

The bulkiness and lack of agility of a joystick prevents it being a real contender for a mouse replacement and instead is more suited to the gaming industry where it prevailed and of course for the continued accessibility for disabled users.

3.5.4 Touch Screen

The touch screen is an electronic device that can detect the location of touches within its display area, commonly used to replace the mouse as a computers primary input. Many industries use touch screen technology with a recent large uptake in the retail environment, more specifically in point of sale systems (Wilson and Inderrieden 1995).

An advantage of touch screen over a traditional mouse is its ability to use absolute positioning to control the location of the mouse cursor. Absolute positioning allows a user to simply press where they want the cursor to be and it will immediately move there, this is in comparison to a standard mouse where a user has to physically drag the cursor from its current location to the desired location.

Touch screen technology is by no means a complete product, Microsoft are soon to release their latest research product Microsoft Surface to their commercial partners. It is a new generation of touch screen devices that can allow multiple people to use the device simultaneously (up to 52 individual touches) by using a projector and five cameras recording infrared reflections from fingertips(Microsoft, Microsoft Surface 2007). The surface is also different from the standard touch screen as it possesses the ability to recognise specially "tagged" items to on screen (for example, mobile phones, credit cards, wine glasses) and provide item specific information.

In general touch screens have the potential to be very quick in selecting objects due to its absolute positioning, however, the speed of touch screens will be ultimately limited by the size of the touch screen and the time taken for the user to locate and then touch the object wanted.

3.6 Clicking

Today's modern day Operating Systems use an interaction style called, "Window, Icon, Menu, Pointer" (WIMP). Developed by Xerox PARC in 1973, and the name coined by Merzouga Wilberts in 1980(Dam 1997). This interaction style uses a physical input to control a cursor on screen, presenting information organised in windows and representing programs and documents by the means of graphical icons. To use and select the icons, a clicking method was introduced.

The standard clicking method appears in two forms:

3.6.1 Single Click

Single Click is the act of pressing the computer mouse (usually left mouse button) once. The single click often just selects an icon on screen and can be classed as the primary action of a mouse.

3.6.2 Double Click

The double click is the action of pressing the mouse button (usually left mouse button) twice in quick succession. Double click has the (almost) universal trait of executing the application/object that the icon is representing.

Other forms of clicking also include:

3.6.3 Right Click

The right click button when pressed, in most modern operating systems and applications invokes a context menu. This function is not included on all OS's but is increasingly becoming a standard.

3.6.4 Middle Click

The middle mouse button is a more recent inclusion in mouse ergonomics, originally providing an additional non-contextual menu, however as needs have changed with operating systems, the middle mouse button has become more synonymous with internet browsing, providing quick scrolling abilities.

Although users have a variety of methods to click, the single click and double click method are most prominent in daily use and are the only two forms of clicks that are needed to perform the standard actions on a modern OS, the other clicking methods are in essence shortcuts for users.

3.7 HCI Implications

Human to Computer Interaction is a fundamental aspect of any system that will interact with a graphical environment (such as Microsoft Windows XP). Consideration has to given to any variables that may impede a user in a smooth running experience with the computer as the basic goal of any HCI is to improve on the interaction of a user to a computer thus making the computer appear more receptive to a users' needs. Any barrier between the users cognitive model of what they are attempting to do is a concern and should be addressed.

The eye tracker poses immediate HCI issues, previous studies have found that the distance the user can reliably sit from the screen, the size of the icons and the clicking systems implemented can all affect how well a system that uses eyes as an input performs (Jacobs 1991).

3.7.1 Inaccuracy of Eye Tracker

The eye tracker being used (Mobile Eye) has an accuracy of ±0.5 degrees, and this on its own isn't particularly inaccurate, however combined with other HCI issues such as distance from screen and icon size the inaccuracy total can be a more worrying value. The inaccuracy of the eye tracker is out of the developers control however if measures can be taken to reduce the HCI implications of the other variables then the inaccuracy of the system can be kept to a minimum.

3.7.2 Distance from Screen

As discussed previously, to focus on an object, the object has to be focused on an area called the fovea; outside this area acuity drops rapidly. Studies have shown that the visual angle the fovea subtends is approximately 2 degrees (Ware 1987). This inaccuracy is not detrimental at close distance but becomes an issue the further away from the screen a user is located. A balance has to be struck between the size of the screen and the distance the user sits away from the screen.

Previous papers have considered standard sitting distance from a user to a computer desk (approx 25 inches) to be sufficient (Jacobs 1991) and (Sibert 2000), this distance should be seen as the minimum requirement for any system that is attempting to control a mouse using eye input alone as it is likely that any users who would require to use such a system using current eye tracking technology would be sat at a computer desk.

3.7.3 Resolution/ Icon Size

When taking into account the accuracies of the eye tracker and the potential inaccuracy of the fovea itself, caution has to be taken to ensure that the icons and buttons the user will be wanting to use are of a size large enough for the eye tracker to focus on. Microsoft Windows XP has extensive accessibility settings that are included for those who are of bad eye sight but can equally be used to increase icon size for the eye tracker.

3.7.4 Fitts's Law

Fitts's Law is a model of movement that predicts the time required to move to a target area given the distance to the target, size of the target, speed of the device using to point (Fitts 1992). It is commonly defined as

$$T = a + b \log_2 \left(\frac{D}{Y} + 1\right)$$

Although Fitts's Law was published in 1956, and many papers have mathematically proven that Fitts's law holds true, many of the findings are still ignored by GUI developers (Shneiderman 1993).

Although not directly related to the uses of an eye tracker on a modern day operating system, Fitts's Law can be used to assess how well the eye as an input medium is compared to that of a standard mouse or joystick controller and secondly if a modern day operating system lends itself to the user of an eye tracker.

4 Design Choices

This section is to discuss the possible architecture designs, and expand on ideas formed in the background reading, discussing the possible events to be implemented in a “mouse-free” environment.

Note, not all the design choices were made at the beginning of the project; many were created due to changes in specification following an iterations analysis.

4.1 Cursor Movement

Movement is an important aspect of using a modern operating system, with most systems using a mouse to control the cursor on screen. With the project employing the use an eye tracker as a form of input, the cursor positional movement will be absolute, providing the user with a ‘what you see (gaze at) is what you get’ system.

The eye tracker outputs the coordinates of the user current gaze in respect to the calibration area defined when the system is configured. The calibration coordinates received are approximately half of the screen resolution of an 800 x 600 pixel environment on a 19 inch monitor sitting at 25 inches away. Fig 3 in appendix B is a screen shot from a recorded session using the eye tracker.

Unfortunately, the approximation of the ratio is an unavoidable constant as further investigations of the eye tracker resulted in there being no direct way to access the calibration settings and as such unable to accurately know the exact configuration of the system.

Sibert notes in her paper that movement of the cursor whilst using an eye tracker is extremely quick (Sibert 2000), Sibert continues to say to point out that some of the speed gained in using the tracker was lost when additional methods filtered and modified the inputted coordinates so that the cursor would appear more smooth to the user.

Filters such as the Butterworth filter aims to capture important patterns in the data and leave/ignore noise (Butterworth 1930) similar to that of implemented in Siberts’ system, but for the scope of this project the developer is particularly interested in how the eye tracker will respond on a modern desktop, without filtering and smoothing. If there is extensive time remaining, the developer will implement a smoothing algorithm and compare with that of a non-filtered input.

4.2 Clicking

An integral part of the modern day Operating System and Graphical User Interface is the ability to click at an icon, an image, or any object using a mouse and movable cursor. Conversely, the eye tracker has no moving parts, no physical button to press to indicate a click, so an alternative system has to be designed that matches, or improves on, the manual point and click method.

Below will describe some techniques for clicking using an eye tracker as discovered in the background reading and devised by the developer.

4.2.1 Gaze-Clicking

As previously discussed in (Ware 1987) and (Sibert 2000), the gaze method involves gazing for a defined time at an object to invoke a click. The time taken to gaze is up for debate, on one hand, it is possible to hard code a gaze value based on previous studies and self research on gaze times; on the other, a configurable system by which a user can define the time of gazing at an object before a click occurs. Both methods have their advantages and disadvantages, one problem that is prominent in the user configurable method, is if the user configures the system so that the time taken to gaze at an object to click is very small, how can the system adapt so that the number of false positive clicks is reduced?

This question arises as if the glace time is too low, as it is possible an undistinguishable pattern is left for a click, one that could be too similar to the user attention briefly being taken away by a flashing advert, or if a user is simply taking slightly longer in reading a difficult word on screen. This would create a flurry of clicks and cause the system to fall into the "Midas Touch" scenario (Ware 1987), whereby the user is constantly found to be invoking an action from the system.

The user, when using the eye tracker will glance and focus on an object that they want to click, a similarity algorithm will run behind tracking the coordinates being fed from the eye tracker. When the algorithm notices that the user is specifically targeting an area, i.e. focusing, the algorithm will click on the target area. It is suggested in Ware's paper that a clutch like system is better used as to "activate" the clicking function when necessary. Therefore in the matters of self discovery and proof of Ware's theory, two Systems, one with a clutch and the other without will be developed.

4.2.2 Blink-Clicking

This method would require a user to physically blink for a set amount of time to invoke a click action. An average human blink takes approximately 30-40ms (Moses 1981), so consideration of this average has to be taken into account so that there can be a differentiation between a user blinking normally, and the user attempting to click. Similar to Gaze Clicking, a decision has to be made to whether the system is robust enough to allow a user to configure how long a blink should be before it is a click. The only reservations of a blink click method is that blinking is an almost autonomous bodily function, it is expected that a clicking system that requires a user to blink will be unnatural, tiring and possibly frustrating to the user, especially if there was a click intensive application being used.

4.2.3 Eye Movement with physical button press

This proposed method is probably closest to a "standard" mouse implementation. In addition to an eye tracker, the user will be provided with a hardware component that allows the user to click (similar, if not the same as mouse buttons); the user controls the location of the cursor using the eye tracking device and can physically press when they want to click. Although not as elegant a solution as using just an eye tracker, it is envisaged that the user will have a more perceived view of control; especially as the act of using a physical trigger instead of an algorithm based trigger will eliminate any false positives for a click.

In using this method, the user is likely to experience a greater selection speed, but is still using a hardware component to click an object. The developer is keen to break away from having to use a hardware component other than the eye tracker; this is to gage how successful using the eyes alone will be on a modern Operating System, therefore the physical button is unlikely to be developed further.

4.2.4 Eye Shift Clicking

The eye shifting approach to a click can draw comparisons to gesture based control systems. The user would in effect be using rapid movements of the eye to create a pattern of coordinates that the system would recognise. Fig. 4 describes the type of eye movement that would be required, in this case a flick of the eyes from left to right and back again.

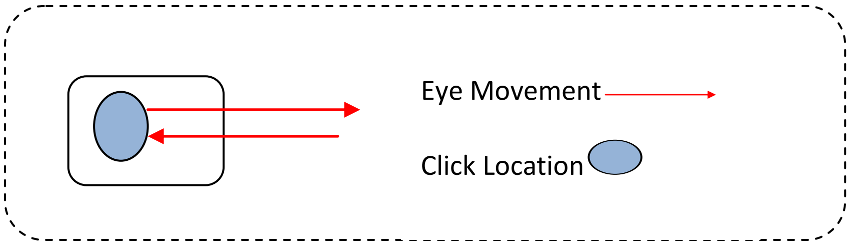


Fig. 4 Gesture based click system

Extracting meaningful patterns from data is called pattern spotting (Rose 1992) , locating gestures is an instance of pattern spotting whereby it is necessary to establish both the start and the end point of the gesture.

An immediate concern with the approach is the common issue in gesture recognition of segmentation ambiguity and Spatio-temporal variability(Lee and Kim 1999). Segmentation ambiguity concerns with when one gesture finishes and another starts, there will be an additional movement between the two gestures that may match another pattern and invoke an action. This is obviously unwanted and correct boundaries between the actions need to be discovered. The Spatio-temporal variability problem is that the same gesture being made can/will vary in shape and duration, even if the same gesturer is performing the gesture. Using models such as the Hidden Markov Model (Lee and Kim 1999) and (Eickeler and Kosmala 1998) have shown that the Spatio-temporal variability problem can be solved with a high accuracy rate.

Further concerns about this method of clicking relates to the accuracy of the eye tracker and the nature of the eye. When monitoring gestures, a reliable stream of data is preferred, the eye tracker will provide a constant stream of data but the actual coordinates given will be unpredictable and hard to map as the eye jitters nearly always.

4.2.5 What Click?

It is imperative that the system which is controlling the point and click system is intuitive to the user with a small learning curve, the more intuitive the system is, the less time the user has to think about what they are doing before it is actually done.

Using this ‘ethos’, the click method which has stood out in terms of intuitiveness is that of the Gaze Click method. The Gaze method has received many glittering reviews, (Ware 1987) and (Sibert 2000), both stating that the system reacted so quickly whilst gazing, that it was similar to the users thoughts being read.

As discussed previously, the Gaze Click will be implemented with a ‘clutch’ like system that will allow the user to specify when they would like the clicking function to be turned on, a simple key press on the keyboard will activate and deactivate the clutch. An alternate system will be developed that will not use a clutch like system and in essence will be, “always on”. It is expected that the always on system will produce many false positives clicks, however will show the need for a clutch like system in a reactive environment.

4.3 Moving a PowerPoint presentation

Whilst a user is giving a PowerPoint presentation there are many ways in which to navigate the slides; one method is to use the provided navigation buttons overlaid on the slide itself (see fig 5).

Fig. 5 – PowerPoint controls

Others methods include using the left mouse button (can only navigate forward), keyboard controls and specially crafted presentation devices such as Amazons Wireless presentation device[1].

For this level of control to be adapted to using an eye tracker, one of two methods can be employed:

1. Gesture Recognition
2. Define Hot Spot area's that perform an action (forward, backwards etc...)

As discussed in section 4.2.4 Gesture recognition although possible will be very difficult to implement, and the lack of a smoothing algorithm will likely heed the implementation further.

Hot spot areas, as defined in (Cao and Ofek 2005) attempt to mimic the presentation controls as in Fig. 5. Firstly, an area is defined on the screen that will act as a virtual button, if the cursor is moved into this area, a timer will start (see Fig 6 in Appendix B) , if the timer reaches a limiting number, for example 0.5 seconds, an action is performed. This action can be anything from moving the presentation forward, backward, to the beginning, to the end or one of the other various presentation controls defined in Microsoft PowerPoint guide (Microsoft, PowerPoint Shortcuts 2003).

The advantage of using hot spot areas is that the actions to be performed can be defined by the user and the hot spots can visually be completely hidden or appear as buttons on the screen. Cao reports positive results in using hot spot area's for PowerPoint control therefore the system will try to emulate the same success using the eye as an input medium as Cao did using fingers.

4.4 Extracting Images

When a designer wants to extracts an object from an image, they often use applications such as Adobe Photoshop, GIMP or even Microsoft Paint; these packages include a tool that allows a user to trace around a selected object and 'lift' the object from the image and then be available for further processing.

The standard motions required for extracting an image using a standard mouse and such applications is as follows:

[1] http://www.amazon.co.uk/Wireless-Control-Presentation-Pointer-Receiver/dp/B000Z2MTLQ/ref=sr_1_34?ie=UTF8&s=electronics&qid=1208866438&sr=8-34

1. Select a starting position on the section of image to be cut (Origin) then hold the left mouse button down
2. Trace around the image to be cut
3. Return to the Origin and release left mouse button

The proposed system is to mimic these actions, requiring that each of the individual steps used on a handheld mouse is repeated with the eye tracking system. It is proposed that an adaptation of the gaze method would be appropriate; allowing the system to recognise that the cursor is in a similar location and thus sending a command to the application that the mouse button is held down whilst simultaneously recording the origin location (refers to 1.). The user will then trace round the image with their eyes all the while a check being performed in the background to see if the cursor is back at the origin location(refers to 2). If the cursor is back at the origin a command is sent to the application that the mouse button has been released (refers to 3.).

The system proposed has two benefits, it will give an evaluation of using eyes to extract images but also present that an eye tracking system can be used to imitate a hold and drag method that could also be used in later iterations selecting multiple items.

4.5 Programming Environment

The goal is to decide upon a suitable programming environment to be used for the system development. The chosen language should be able to fulfil the necessary functions to interact with the eye tracker and be able to interface with the Microsoft Windows environment, more specifically Microsoft Windows XP.

The environment would also be required to be Object Oriented in nature in order to achieve a more component based system, where individual objects have their job and other components can be attached freely to use their inputs and outputs. Using Object Oriented languages also lends itself to a more “fluent” and elegant message passing system that may become useful as the project continues to grow and become more complex.

Portability at this time is not an issue; the system will only be expected to work on a Microsoft Windows XP environment that has been standardised, and beyond the individual applications installed on individual computers there will be no differences between computers.

There are many languages currently available that can fulfil these criteria; however the programmer has only extensive experience with two of the potential languages, these being Java and C++.

4.5.1 Java

Java, a development programming language, produced by Sun Microsystems, and released in 1995 [2] is a worldwide popular high-level programming language, and is portable across multiple platforms by compiling the written code to 'java bytecode' which in turn is executed using the hosts hardware java virtual machine.

Although not needed in the context of the initial design, Java has extensive GUI capabilities and would be suitable for work on a full system that would require some user input of physical buttons, text boxes and such. Java does have a Media Framework API, providing ways in which to extract frames from video files or camera's, suitable for testing and reviewing of the recorded data from the eye tracker. Many of the research community prefer to use more mature products such as OpenCV (C++), this isn't to say that no work has been completed using the Media Framework, Abraham successfully used the framework to control games using a webcam and a users hand (Abraham and Nath 2006).

4.5.2 C++

C++ has become the de-facto programming language for research and projects in the arena of computer vision over Java, due to its popularity in academia and its reported performance characteristics(Prechelt 1999). However one could argue with the introduction of JIT (Just in time) compilation with Java, and other optimisation techniques the gap isn't so large, and many studies have proved so (Bull, et al. 2001). Despite the debate of performance, tools such as Open CV have been built and allowed those using C/C++ to perform advanced vision projects. As C++ is an extension of the C programming language, one of the added extensions was the support for object oriented programming and a standard library (similar to Java) that included algorithms and containers. C++ also includes functionality to add Microsoft applications (and their respective capabilities) to a programmer's application; this includes PowerPoint, Word and Excel, all of which have a strong relevance to the project.

The languages and their respective toolkits are available to the developer so accessibility is not a concern while selecting a language. It is then left to confidence in the language and potential capability; both C++ and Java have extensive toolkits which would allow the developer to create a system by which the mouse will move "hands free", the developer has experience in both C++ and Java however it is the capability to interact easily with Microsoft products and the widely used OpenCV toolkit that edges out Java as a programming possibility. C++ has therefore been selected for development of the project.

[2] www.sun.com/java

5 Evaluation Plan

5.1.1 Measurable Factors

As previously stated in the background reading chapter and the design, when using a system that uses algorithms to calculate if the user wants to click (or not click), attention has to be paid to the amount of false-positives (unwanted clicks that occur) and false-negatives (clicks that should of occurred).

Evaluation of accuracy and the error rate can therefore be made using the total amount of clicks, total amount of false-positive clicks and the number of false-negative clicks

$$\textbf{Accuracy} = \frac{\text{Successful Clicks}}{\text{Planned number of clicks}}$$

$$\textbf{Error Rate} = \frac{\text{(False Positive Clicks + False Negatives)}}{\text{Planned number of Clicks}}$$

Consideration also has to be given to impact of the error types. False-Positive clicks are unwanted clicks, therefore clicks that occur that regardless of the users' intent at the time. These clicks are potentially more harmful than the False Negative Clicks, that are in essence, Clicks that should have occurred but didn't.

An environment such as Windows XP is very reactive to user input via clicking and is unforgiving to mistakes, partly as it was never designed to interact with a device that would have any noticeable error rate. The only noticeable preventative method from incorrect clicking is the double click feature, preventing users from activating anything on the desktop without physically clicking on the icon twice in quick succession (Microsoft, Double Clicking 2002).

5.1.2 Test Data

As the field of using an eye tracker to interact with a computer environment is relatively new, there are no standardised tests that can be used to draw comparisons and rate the effectiveness of the eye tracker against other interaction devices.

Therefore, to try and gain some insight to how the eye tracking system would perform in a standard environment, a simple test plan was created that would easily lend itself to comparison to other devices if needed. This comparison is the most effective with the image extractor test.

6 Implementation Iterations

6.1 First Iteration – Read and Move

6.1.1 Goal

The chief goal of the first iteration was to develop a method by which values can be passed (x and y coordinates), this is the most crucial of the requirements as this can be classed as the foundation of the project. Upon creating an interface in which coordinates can be passed, further analysis on the similarities between sets of coordinates and other actions can be done. In this iteration, a movement mechanism will be created that will require two coordinates to be passed (x,y) and will move the cursor to the location given.

6.1.2 Detail

The background reading provided insight to how C++ can interact on a low level with the mouse movement functions. Included in the header file **windows.h**, the function SetCursorPos(x,y) provdes such access to move the cursor to a given position given a set of coordinates. The simplicity of the movement function also allowed for easy testing.

The way in which values could be input into the system wasn't as straightforward. Due to the very limited time with the eye tracker itself, a temporary shift in priorities had to be made. A choice was made to concentrate on an alternative input system that would work using pre-recorded outputs from the eye-tracker. Fortunately the recorded output was in a recognised format, i.e. comma separated values (.csv), and this allowed the developer to use C++ string manipulation classes to extract the data from the file and thus the coordinates needed to pass to the mouse movement function.

The following is an excerpt of code that was used to read in and separate the values needed into x and y coordinates:

```
void getCoords(int coordx, int coordy)

while (getline( eyetrack, line )) { //While not at the end of the file

      if (line.empty()) continue;

            stringstream ss( line ); // interface to manipulate strings
{ string val;
getline( ss, val, ',' ); // array point, max char to store, delim char
stringstream( val ) >> coordx; //set first value found to coordx
}
}
```

Here, the **getline** function extracts all the characters provided in the string stream, up to the delimiting character(cplusplus.com 2008), providing the x coordinate. Similarly the code continues to extract the data after the delimiter and up to the white space character (carriage return).

When the x and y coordinate are extracted, they can be passed over to the mouse move method, that takes input of coordx and coordy, as follows.

```
 SetCursorPos(coordx,coordy); //Sets the absolute Cursor position
```

6.1.3 Testing

In order to test both the input coordinate's method and the cursor positioning method, a series of small tests were created using the main method.

Using an 800 x 600 pixels desktop, a .csv file was passed to the method **getCoords()** with the following coordinates:

X	**Y**
0	0
0	600
800	600
800	0

This would in essence cover all the four corners of the screen and give a visual aid to the developer of the cursor location. To verify the exact location of the cursor, the developer used the console out and the inbuilt GetCursorPos function in the **windows.h** header file as follows:

```
GetCursorPos(&pos);
int seex = pos.x;
int seey = pos.y;
```

```
cout << “I move x = “ << coordx << “ and y = “ << coordy “\n”
cout << "I see x = " << seex << " and Y = " << seey << "\n";
```

The output to the console was as follows:

```
I move x = 0 and y = 0
I see x = 0 and y = 0
I move x = 0 and y = 600
I see x = 0 and y = 600
...
```

Overall there was a 100% success rate in moving the mouse to the correct locations provided.

6.1.4 Evaluation

The system built at this stage is a very naive movement system with no error correction; for example, if a user blinked or a brief loss of pupil tracking occurred the mouse would erratically move (this will be addressed in a further iteration).

The system is at a basic level but provides a foundation for which other methods can be added that will work with the inputs and outputs provided. This system also provided the developer with some further insight into how the other proposed methods are going to have to interact with the system. Finally this initial Iteration confirmed that the project was feasible and the minimum requirements achievable.

The movement system is currently working for a manually created set of coordinates however when the system begins to use the coordinates provided by the eye tracker additional processing will need to occur to align the coordinates provided by the eye tracker and the intended location on screen.

6.2 Second Iteration – Click

6.2.1 Goal

The goal of this iteration is to advance on the current code base and add extra functionality in the form of clicking functionality. The clicking functionally used will be that of Gaze Clicking, in this case the user simply has to gaze at an object for a set amount of time for the system to click that object.

6.2.2 Detail

The previous iteration highlighted that the movement system currently didn't identify when there was a loss of connection with the pupil, this would be indicative of a blink (as the eyelid covers the pupil) or that the hardware and/or software were not correctly functioning to identify the location of the pupil. To overcome this, there needed to be further investigation on how the eye tracker deals with the loss of pupil recognition.

The investigation was very conclusive, showing that the eye tracker would resort to a default coordinate of (-2000, -2000) when no pupil could be recognised. Using this, the developer added extra functionality to the coordinate extraction method, that would search for an x and y coordinate of -2000. If the coordinate was found, the last previous "valid" coordinate was used, until a new valid coordinate (recognition of the pupil occurs).

```
if (tempx == -2000) {  //loss of pupil is indicated by a x/y coord of -2000.
      tempx = coordx);
}
else {
      (coordx = tempx);
}
```

The second implementation in this iteration is a method by which to click. As discussed in the design section the decided implementation was the Gaze Clicking method. The method as proposed in 'An evaluation of an eye tracker' (Ware 1987), stated that the user only had to gaze at the screen for approx 0.4 seconds to invoke a click. The developer will use this as an initial benchmark to gage performance of the eye tracker.

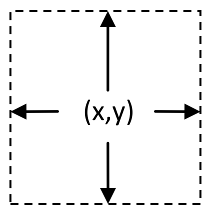
Despite thorough investigation of previous systems that employed an eye tracking system to control a computer, no previous examples or standard implementation code could be found or released; therefore the implementation has no attempted resemblance to others.

The Gaze Click System works as follows:

- Upon a valid coordinate being passed to the system and is classed as a previous coordinate
- Another coordinate is then read into the system classed as the current coordinate
- Using the previous coordinate, an active area is formed around the coordinate (see fig 7)

- A check is made to see if the current coordinate is in the boundaries of the active area
- If it **isn't**, the count is set back to 0 and the current coordinate is set as the previous coordinate
- If it **is**, a count is added to a counting system
- A check is made to see if the count has hit a set limit, if it is a click is invoked and the count set to 0.

The example code is located in Appendix B



- An active area is created by defining a square in terms of coordinates
- (x-5) to (x+5) wide
- (y-5) to (y+5) tall

Fig 7 – Definition of an active area

6.2.3 Testing

Similar to first Iteration, a .csv file was passed to the system with the Gaze Clicking method activated. The csv file had considerably more coordinates contained than the first iteration and tested the loss of pupil update as well as the new Gaze Click System.

If the system finds a coordinate value of -2000 a message was passed to the console that reads similar to this:

```
Found Coord -2000, previous valid coord is: 200, 145
```

If the new coordinate passed to the system is in the "active area" provided by the previous coordinate, the following message (or variance of) is passed to the console

```
Similarity Found! Adding 1 to the Count: Count = 3
```

A loss of similarity is indicated on the console as

```
Similarity Lost! Resetting the Count: Count = 0
```

Finally when the count had reached 10 (using Ware's example of 0.4 seconds) the following was displayed to the console

```
Click! Count Reached 10. Count Reset. Count = 0
```

6.2.4 Evaluation

The goal of this iteration was achieved in full. The system now is able to recognise when there is a loss of pupil recognition, be this for a blink or for a software/hardware failure and will try to compensate for this. The gaze click method works very well and early indications show that it is very responsive; when real data is used however the movement of the mouse is quicker than should be expected. This is due to the 0.04 second polling time of the eye tracker being significantly slower than the computing time taken to extract the coordinates and move the cursor. Using a delay similar to that of the polling time of the eye tracker should realign the movement of the cursor to real-time rather than computational speed.

6.3 Third Iteration – Move back and forth

6.3.1 Goal

The goal of this iteration is to add to the improvements made in the other iterations and to include functionality to the system that will allow a user to move a power point presentation back and forward.

6.3.2 Detail

Controlling a power point presentation without the use of a physical mouse isn't new; gesture control has been researched extensively (Baudel and Beaudouin-Lafon 1993) and even a system that analyses the speaker words to indicate when a slide has finished.

Effort was made to ensure the PowerPoint control system as easy to use as possible; a system was devised whereby hot spot areas were created. The hot spots are based on a Microsoft research paper that investigated alternate ways to control a PowerPoint presentation (Cao and Ofek 2005).

These hot spot areas are located at the far right and far left of the screen (see diagram XXX). The hot spot areas are activated by a user gazing into the defined area for a set amount of time, in this case 1 second.

To define a hot spot, a set of coordinates that set the boundaries for the x axis and y axis are needed. Then comparisons are made with the current location of the cursor to the location of the hot spot. If the cursor is contained within the hot spot area, the area is classed as being in a “depressed” state, similar to that of a holding a mouse button down on a standard button. If the user gaze's is contained within the area for a particular length of time, an action occurs, in this case the PowerPoint presentation is moved back or forwards.

Below is an excerpt of code that controls the depression of the hot spot:

```
if (i <= 25) {

      if (coordx >= pwrptxFW1 && coordx<= pwrptxFW2 && coordy >= pwrptyFW1 &&
      coordy <= pwrptyFW2){

      cout << " Mouse held in active area RIGHT for 1 second - Powerpoint
      slide + 1 \n" ;
i++;
      }
```

Another improvement made to the coordinate extractor method was to add a delay for every coordinate extracted. The poll time of the eye tracker is 0.04 seconds, so a method was added that paused the application for 0.04 seconds at the end of every coordinate extraction and brought it in line with the users' perception of real-time.

6.3.3 Testing

Similar to previous iterations, a csv file with preset coordinates was fed into the system with the PowerPoint presentation movement method enabled. Using the console for verification, the cursor was held in the active area's forward and then back for 1 second which resulted in the following being written to screen

```
      Mouse held in active area RIGHT for 1 second - Powerpoint slide + 1

      Mouse held in active area LEFT for 1 second - Powerpoint slide - 1
```

Additional code was then included to change the text written to screen with the action to move the PowerPoint the presentation; this was also tested with a PowerPoint presentation in full screen mode to great success.

6.3.4 Analysis

The goal here was completed in full and now the movement system instigated in phase 1 is fully complete. The PowerPoint presentation move works well, however the initial estimate of 1 second to move a presentation left or right seems a slight too long. In a further iteration, reducing the time to a level closer to the Gaze clicking method may give the user a better experience in using the system. What is noticeable is a lack of visual aid so that comparisons can be made to the eye trackers output and what the user was trying to do on screen, in line with the possible advance requirements a simple video player may be created that will run in tandem with the eye tracker movement code. The theory of hotspots is also a powerful tool and in further iterations more advanced hotspot areas can be created, such as return to the beginning of slide, or quit slide.

6.4 Fourth Iteration – Trace and Cut

6.4.1 Goal

The aim of the fourth iteration is to investigate the possibility that the eye tracker can be used with the cutting tool provided in packages similar to Microsoft Paint or Adobe's Photoshop. The cutting tool provides a method to cut a section of an image and can be modified, copied to clipboard and various other tasks performed upon it.

6.4.2 Detail

The developer considered that the best approach to this method is to mimic the actions of a mouse. When a user is cutting an image using a standard mouse, the approach is as follows:

4. Select a starting position on the section of image to be cut (Origin) then hold the left mouse button down
5. Trace around the image to be cut
6. Return to the Origin and release left mouse button

The new system that is attempting to mimic the actions of a standard mouse can thus be split up into four distinct tasks.

6.4.2.1 Selecting the Starting position

Influences from previous iterations can aid with step 1, the gaze click method as implemented in iteration one provides a stable way to recognise if the user is concentrating their gaze on a particular area. Using aspects from the Gaze method with the inclusion of some native mouse control code, in this case the command:

```
mouse_event(MOUSEEVENTF_LEFTDOWN, coordx, coordy, 0, 0);
startcoordX = coordx;
startcoordY = coordy;
depressed = true;
```

Initially the user has to select the cutting tool from the image editor or use a keyboard shortcut. Then, the user is to gaze at the area they want the Origin to be, the adapted gaze click method will then start to notice similarities between the coordinates and start the gaze timer. When the gaze technique reaches its time trigger (increased to 1 second to ensure that the user is definitely attempting to use the cut functionality), the system runs a command that simulates the user holding down the left mouse button and records the location of the last coordinate before the mouse button was held down. The Boolean value is used to verify that the mouse button is effectively held down.

6.4.2.2 Trace around the image

If the depressed value is set to true, the method cutimage() is run, the user is then to trace around the outline of the object or image they want to extract from the picture. As the user moves their gaze around the object they are extracting, a check is made on every coordinate passed to see if it is in the region of the origin. If it is in the region of the origin, the modified gaze method is repeated with a timer of 1 second as it is presumed that the user has traced around the entire image.

6.4.2.3 Return to the origin and release mouse button

The final stage of the method is the user gazing at the origin for one second and in doing so triggering another method that releases the mouse button and sets the Boolean depressed to false. The releasing of the mouse button will finalise the process of cutting an image and the user will be presented with a highlighted image.

6.4.3 Testing

As with the other iterations, a csv file was created that would test the methods recently created. In this case the csv file was set up to cut a square out of a coloured canvas using Microsoft Paint. See Appendix C for test image and resulting extracted image.

6.4.4 Analysis

The image extractor is a very crude method to extract an image, using the basic schematics that a standard hand held mouse would use; however the iteration was to not to create a perfect system but to research the feasibility of such a system. The early indications are that a system using an eye tracker to aid in image extraction is possible and likely quicker. What isn't clear is the accuracy of the system; the sporadic nature of the eye will cause some inaccuracies but the implications of which will be revealed when the full system testing occurs.

Another point to note is the starting and ending coordinate point is not highlighted and for a complicated image a user may lose their Origin point. A visual aid may be more appropriate for a task of this potential complexity.

6.5 Fifth Iteration – Vision

6.5.1 Goal

This goal is in addition to the minimum requirement of the project but felt necessary for completion. When the user records their actions using the eye tracker, the resulting outputs are of an video file in avi format of what and where the user is looking at (see fig. 13 in appendix B) and a csv file of the attributes of the recording, what would be a good validation of the accuracy of the system is to run both the csv file and the avi file in tandem to see if the physical cursor on screen, traces the crosshair that is overlaid on the avi file.

6.5.2 Details

The background reading and design uncovered that OpenCV was a popular method to implement a video stream into a computer vision project. Used mainly for the ability to stream a web camera, it can also be used to play videos frame by frame.

To begin a workspace frame has to be initialised and the size of the window specified, 800 x 600 is used as this was the resolution of the screens used for data collection and testing.

```
IplImage*        tmp_frame = NULL, *image;
CvCapture*       cap = NULL;

image =  cvCreateImage(cvSize(800, 600), 8, 3);
cap = cvCaptureFromFile("test_000032.avi");
tmp_frame = cvQueryFrame(cap);
```

```
image->origin = tmp_frame->origin; // used to flip video image around
cvNamedWindow("Eye", 1);
```

Then frame by frame, the image file is read in and displayed onto the window initialised previously

```
for( int fr = 1;tmp_frame; tmp_frame = cvQueryFrame(cap), fr++ )
{
	double t = (double)cvGetTickCount();
	cvResize(tmp_frame, image, 0);

	t = (double)cvGetTickCount() - t;
	cvShowImage("EYE  ",image);
	char k = cvWaitKey(40);
	if( k == 27 ) break;
}
```

6.5.3 Testing

To test this method was fairly none-trivial, a standard media player was first used to check the validity of the video file and then run using the Open CV player, a slight tweak of the cvWaitKey variable was needed to ensure that the videos ran at the same pace.

6.5.4 Analysis

The OpenCV video player was very successful in its task. The ability to extract the video frame by frame means that the video can be played in line with the extraction of the mouse coordinates, and the additional features of OpenCV will allow other images to be drawn on top of the video currently playing.

6.6 Sixth Iteration – Clean-up

6.6.1 Goal

The Goal of this final iteration is to complete any additional improvements to the system that are feasible in the remaining production time of the project.

6.6.2 Details

Tasks left to complete

- Add the eye tracker to screen ratio into the coordinate extraction method

- Clutch like system for gaze blink
- Integration of graphics with image cutter
- Synchronisation of video to code
- Blinking Click
- More Hot Spots for PowerPoint presentation

Due to time constraints, the following were implemented:

- Adding eye tracker to screen ratio
- Blinking Click – Using a similar count to the Gaze Click method, the time in which the user closes their eyes in monitored (a coordinate of -2000,-2000), a preliminary time of 0.5 seconds was used.
- Additional Hot Spots for PowerPoint Presentation – An extension to the previous work, providing methods for the user to quit the presentation and restart the presentation from slide 1.

6.6.3 Testing

Similar to the previous iterations, a .csv file was manually created for each of the new features included.

6.6.4 Analysis

Eye Tracker to Screen Ratio was a simple addition to the standard coordinate extraction method, simply doubling the coordinate value. Initial testing results proved to this be working, however the real proof of functionality will be when real data from the eye tracker can be passed to the extraction methods.

The Blinking Click method has comparative results to the Gaze Click method, however it is hard to manually model how a users blinks and the success of the method will be seen when real User data is passed through.

Finally the additional Hot Spots worked as well as expected, with the appropriate action occurring if the Hot Spot is activated.

7 Testing and Evaluation

7.1 Testing

As discussed in the methodology, after each iteration of the project and thus as each feature is implemented a reasonable level of testing is completed, similar to that of white box testing, tests are performed with the knowledge of testing the internal structure. For ever iteration a csv file was created by hand that would reflect the actions of a user whilst they were using an eye tracker. In using this method numerous bugs and improvement options were found. When the bugs and improvement options are considered to be at an acceptable level, the project is classed as complete.

Although not stated in the iterative and incremental development process, the developer set some time for full system test that would investigate the features implemented with more rugged, real data. Due to the scarcity of time with the eye tracker itself, help from the current possessor of the eye tracker (Bristol University) was enlisted to extract some data from the eye tracker. The data from which will be used to evaluate the system

7.1.1 Testing Results

7.1.1.1 Gaze Click

User / Results	Number of expected Clicks	False Positives	False Negative Clicks	Total Number of Clicks	Successful Clicks
User 1	16	10	2	28	12
User 2	16	6	6	16	4

Accuracy: 50%

Error Rate: 0.75

7.1.1.2 Blinking

User / Results	Number of expected Clicks	False Positives Clicks	False Negative Clicks	Total Number of Clicks	Successful Clicks
User 1	16	4	0	12	12
User 2	16	4	12	24	8

Accuracy: 62.5%

Error Rate: 0.62

7.1.1.3 PowerPoint Presentation Move

User /Results	Slides to Move	False Positives	False Negatives	Slides Moved
User 1	10	0	1	11
User 2	10	1	5	6

Accuracy: 85%

Error Rate: 0.35

7.1.1.4 Image Extraction

As the image extractor isn't easily quantifiable, the resulting cut images are below; the users participating in the tests were also asked to cut the same image out by hand to see if there was any improvement between the two.

User 1

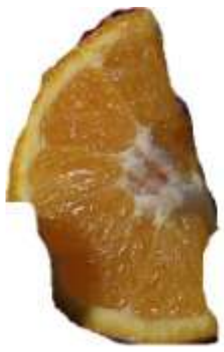


Fig. 8 - Image extracted with mouse

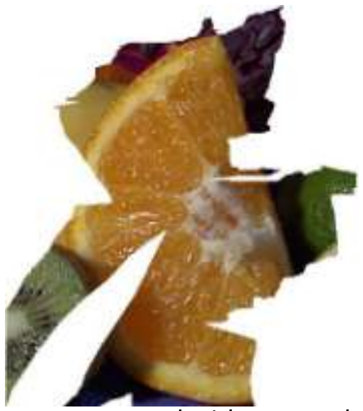


Fig. 9 - *Image extracted with eye tracker*

User 2

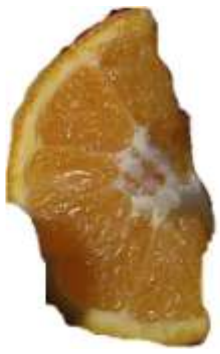


Fig. 10 - Image extracted with mouse

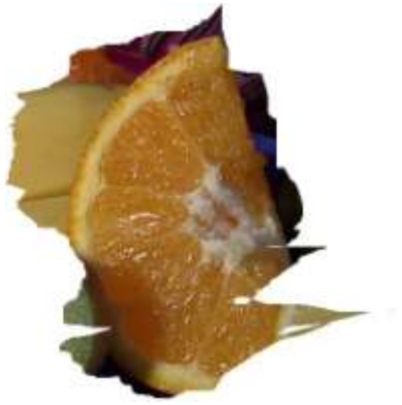


Fig. 11 - *Image extracted with eye tracker*

7.2 Evaluation

7.2.1 Requirements Evaluation

7.2.1.1 Minimum Requirements

Each of the minimum requirements, as stated at the beginning of the project has been fulfilled during the completion of the project. The cursor can move freely around a desktop using data provided by the eye tracking device. The system is able to distinguish between a small number of commands with the ability to mimic, single and double click using a Gaze based method. The project is also able to control simple PowerPoint commands that include moving the presentation forward and back one slide and back one slide. Finally the project is able to recognise when there is a loss of pupil recognition (such as a blink) and compensate accordingly.

7.2.1.2 Advanced Requirements

The advanced requirements were formed during the iteration process whereby the potential of the eye tracker was seen or by potential improvements that were beyond one line code fixes.

The first of the advanced requirements was to construct a video input stream that would allow the ability to read .avi files frame by frame at the same rate as the coordinate polling time of the eye tracker (approx 0.04 ms). The popular computer vision toolkit OpenCV was used due to its compatibility with C/C++ and the extensive tools it provides to handle vision projects. The aforementioned video stream player was used to help analyse the actual movement of the cursor on screen to the intentions of the user at the time.

The second advance requirement was an investigation testing the power of an eye tracker as an image extractor, exploring the feasibility of such a system and any improvements gained by using it. This requirement deemed a success and fig 9 and 11 were the results of a test using the extractor.

The third advanced requirement was an improvement noted in the PowerPoint presentation iteration; this being the inclusion of addition ‘hot spot’ areas for extra control of the presentation. The two included hot spots included reset the presentation to slide one and the other exited full screen mode. The two hot spot areas were completed successfully.

The final advanced requirement was to include a secondary clicking as an alternative to the Gaze method. This was successfully completed with the creation of a clicking system based solely on blinking.

7.2.1.3 Failed Requirements

Although the minimum requirements have been fulfilled, the developer was particularly keen to implement a system by which the clicking functionality of the Gaze method contained a clutch like system; this was because the developer felt that the always on click method made the system far too sensitive and prone to false positive clicks and as such was considered a failed requirement.

7.2.2 Methodology Evaluation

As described in the methodology section, an iterative development cycle was employed as the methodology to be followed through the project proving a defined structure for the application creation.

The initial requirements as defined at the beginning of this project were mapped onto the iteration steps of the methodology; this resulted in three initial iterations, movement, these being clicking, and presentation movement respectively.

The iterative and incremental development structure was found to be very beneficial and complimentary to the project, allowing the dev eloper to work on individual aspects of the project, test and then analyse the iteration completed, suggesting any improvements that may be included in further iterations. This allowed to developer to discover potential and real problems with the system early. The most notable example of this was during the first iteration whereby the developer found the eye tracking hardware provided no direct access to the real-time coordinates and that the only method to access the whereabouts of the users gaze is to access the output files from a recorded video session. With enough time the developer would have been able to translate the output from the serial port into meaningful values.

Using just the iterative and development cycle without a definitive framework caused the project to have a distinct lack of structure at times. The use of a complimentary framework such as the Dynamic Systems Development Method (DSDM) uses iterative development and would have provided a greater level of structure to the development; however DSDM was felt to be too complex for what was considered to be a research project and not so much a development of a tangible working product. The Unified Software Development Process (USDP) is most similar to the developer's approach of the project, as it primarily uses the iterative and incremental development process, but is use case driven (Bergström and Råberg 2003).

7.2.3 Implemented Features Evaluation

7.2.3.1 Cursor Movement

The movement of the cursor was the most fundamental aspect to the entire project. In having a robust movement system the project was allowed to expand and additional methods added onto the system. While the system was running, the actual on screen cursor movement was very jerky in nature, this wasn't unexpected as the eye is naturally jittery and the eye tracker is simply translating the movement on screen. This 'jittery' nature cold of been removed by using smoothing techniques as discussed in the background research however the functionality of the system didn't appear to be affected as the algorithms used to detect the gaze of a user took into account that they eye did not stay static for long.

A notable frustration for the developer was having no direct access to the calibration tools provided with the eye tracking software. The developer noted that although the background research and the design reported an approximate 2:1 ratio of real screen position to extracted coordinate it was by no means exact. This approximation is likely to have a detrimental effect on the methods that use the cursor movement position as a foundation; however limitations in the hardware and software prevented the developer from being able to do anything else but approximate.

7.2.3.2 Gaze Clicking

The gaze clicking method was the first of the two clicking methods implemented. The theory was that a quick glance at what the user wanted to interact with was all that was needed, the testing proved 'provisionally' otherwise, with an accuracy of 50%. The Gaze Click method proved to be rather sensitive, with a large amount of false positive clicks amounting to an error rate of 0.75. This developer considers the large number of false positive clicks to be attributable to the lack of clutch mechanism (described in the design) that wasn't implemented due to time constraints rather than the short time a user had to gaze before a click is invoked. This caused an always on system and experienced the "Midas Touch" problem (Ware 1987).

After reviewing the false negative rate it was found that in nearly all cases the false negative occurred due to the cursor sporadically moving across the screen, the likely cause to be a brief loss of pupil recognition or due to the algorithms that work with the eye tracker providing erroneous coordinates.

The Gaze clicking method does show that clicking via a gaze is possible and that it is extremely quick but that the speed gained is shadowed by the high number of clicks that the user did not what to happen (false positives). A clutch like system would have reduced if not removed the Midas touch issue however at more expense to the user who will constantly have to depress and release the clutch to switch on and off the clicking system.

7.2.3.3 Blink Clicking

The Blink Click method was essentially an extension to the standard clicking system whereby a user blinks for approximate half a second to invoke a single click. In comparison to the Gaze click system the blink system performed better, providing results that showed the system to be better performing in terms of accuracy (62.5%). This can be explained by the nature of how the click is triggered with the blink system, the user first has to locate the object they want to click and then physically blink, the blink being similar to a user physically pressing the left mouse button down. This physical intentional trigger is different from the gaze click whereby the gaze is interpreted and is difficult to distinguish between a user wanting to click and a user simply studying an image for an extended amount of time.

While studying the results, the number of false negative clicks between the two users is significantly different, user 2 doubling the total amount of clicks experiences whilst using the system. A review of the users session recording and a study of the coordinates extracted showed that User 2 was blinking significantly longer than User 1, this was triggering the blinking system 3 or 4 times and causing the error rate to increase heavily (0.62). The system was working to specification; however due to time constraints the user was not available for a retest.

The problem with a blink click system as described in (Jacobs 1991)is that making a user blink to click feels very unnatural and can become quite annoying. Despite the improvement in accuracy and the minimising of false positives, using the blink system would seem an inconvenience to the user and not align itself with good HCI practice.

7.2.3.4 PowerPoint Presentation Movement

The PowerPoint system was the first application specific tool that would use the eye tracker; the idea was inspired from current gesture based systems that can control PowerPoint (Baudel and Beaudouin-Lafon 1993)and (Lee and Kim 1999). Users were asked to gaze at “hot spot” areas that would invoke movements of the PowerPoint presentation. Overall the system was very responsive to the user with nothing more than a brief glance needed to shift the slide to the left or the right.

The analysis of the test results showed that User 2 hadn't moved all the slides as expected; when the raw data was examined it was found that the user had glanced at the active area for a very brief amount of time approximately 8ms when the trigger was 50ms.

The PowerPoint movement system appears to work very well, obtaining an accuracy of 85% and a very low false positive rate, meaning that during a standard presentation the slides are unlikely to sporadically move. The success of the presentation move system can be largely attributed to the large size of Hot Spots that compliments Fitts's Law, as the bigger and closer the target, the easier and less time it takes to point (Fitts 1992).

The problem envisaged with such a system is quite how it can be implemented in a real-life situation. The test of the system was a user sat in front of a computer screen approximately 25 inches away; in reality someone giving a presentation using PowerPoint is likely to be further away from the computer and not in a line of gaze to the computer screen. The system could be modified however and the system scaled to match the size of presentation screen only reacting when the speaker physically turns to look at the presentation screen itself, however this is likely to present other issues.

7.2.3.5 Image Extraction

The image extraction was the final feature to include use of the eye tracker, and was possibly the most intriguing of the applications developed. This idea was inspired by a colleague of the developers who had problems with using the cut tool from Adobe Photoshop application. They commented that holding the mouse button down while extracting complex images can be extremely tough and frustrating.

In testing the users were asked to trace their gaze around the orange piece at the bottom of the image (see fig 12). The resulting images are fig 9 and 11. The two images are dramatically different, this can be attributed to the strength of the pupil recognition by the eye tracker whilst the user traced the image. This can be seen within the recorded data files where User 1 had a large amount of -2000 coordinates recorded (indicative of loss of connection) and other sporadic coordinate values that made the cursor jump around the image, creating the very sharp edged image in fig 9. User 2 appeared to have more success with the eye tracker, with no sporadic movement of the cursor and the resulting image nearly tracing around the entire orange piece.

A notable point of this experiment was when the user cut out the image manually by using the mouse, the users were asked to do so whilst the eye tracker continued to log their gaze location. As a mini experiment, for nothing more than the developers' curiosity, the recorded coordinates were passed through the image extractor algorithm. The developer found that for both cases the users actually traced the image more accurately using both the mouse and the eye tracker than just with the eye tracker alone. The likely explanation for this phenomenon is whilst using the mouse, the user is presented with a moving object to focus their image on and is unlikely to be distracted by anything else in the image, whereby if the user is using the eye tracker alone then it is possible (unbeknownst to the user) that their attention is being grabbed.

The initial results of the image extractor is promising, there is still a large gap between the accuracy of the image extracted by hand and the one of the eye tracker but as a first of kind experiment there are a lot of improvements to be made.

7.2.4 Time Management and Schedule Evaluation

Despite a lapse in the initial weeks of the project due to problems concerning an initial hands-on session with the eye tracker, the development of the project was kept to schedule with each iteration, besides the first, completed and tested (provisionally). This was not the case for all aspects of the project; the full system testing was postponed for approximately 3 weeks as the eye tracker was currently located at Bristol University and there weren’t any plans for it to be returned to Leeds before the end of the project. With this in mind, a document containing a provisional ‘test plan’ was sent to Bristol University (see Appendix C) and the recorded data forwarded to the developer.

The report writing schedule started on schedule, however as the project continued it was clear that the expected time to write the report was underestimated. The report was due to be finished by the end of March however work continued on the report until mid April. University and External commitments contributed, to the extensive time to write the report in addition to the testing that occurred far later than planned, shifting the project schedule by about 2 weeks overall.

7.2.5 Testing Evaluation

After each iteration, there was a brief testing phase to ensure that iteration goals were achieved. This helped to minimise any severe problems with the system, and also to highlight any improvements that weren’t included in the original design. The sample files used to conduct these tests were created manually, and as such were not good representational data of what the real input to the system would be like. This sometimes gave the impression that the system was working to a higher level than it really would be using ‘real’ data.

The second phase of testing (full system test) focused on using data provided directly from the eye tracker, attempting to provide a more accurate representation of how the eye tracker mouse system would run if running real-time. However the data provided was not in ‘real-time’ and was instead was the product of a user recording what they would have done if the system was running on their machine. This recording is then passed through a system to check whether the users ‘potential’ actions occurred. This test results have to been taken subjectively, as the users had no immediate feedback on their use of the system, preventing the users from knowing when and where they were using the system wrongly. Yet the results aren’t meaningless and do show the potential of an eye tracked device as an input medium and as such the tests can be seen as a success.

7.3 Conclusion

The project has investigated if today’s eye tracking technology is capable of controlling a modern day operating system. The project has met varying results from the gaze clicking system performing at an undesirable level to an unexpected achievement with the image extractor. While the system as a whole is not performing up to a level that would be acceptable for a commercial product, it serves as a good proof of concept and as a firm grounded base for other projects.

It is an unfortunate event that the system doesn’t run in a live environment where coordinates are fed real time. This hindered the testing and evaluation as the users did not get any feedback from the system whilst using it and could only view their actions after they had recorded what they planned to do. This makes the system hard to evaluate in terms of replacing a mouse with the eye as an input medium but does show the potential of what a live system could achieve.

There is definite more room for improvement; accuracy is an issue with a system such as this and keeping the number of false clicks to a minimum is also imperative however with time and the ever falling cost of hardware, a system that is both affordable and accurate will be available to all thus allowing more people to investigate the advantages of using the eye as an input medium.

7.4 Further Work

As mentioned earlier, the system is by no means complete and this project could be improved and extended. Some examples are listed:

- Convert the current flat file coordinate reader into a real time system that uses coordinates passed directly from the eye tracker
- A clutch like mechanism for the Gaze Click system as described in section XXX
- A personalised system that can tailor the time needed to gaze at an object before it is invoked
- More PowerPoint hot spot area’s
- Gesture Control – Creating patterns with eyes to invoke actions on screen
- More studies on using a modern OS and an eye tracker – Browsing the internet, playing games
- Create on screen graphics and widgets that are accessible to the user and that can provide further help (as described in Iteration XXX)
- Explore the idea that using eye gaze could be adapted to work with more physical objects such as light control and switching on and off remote projectors similar to the manual controls used in many lecture theatres.

7.5 Bibliography

Abraham, A, and N Nath. “Computer Vision for Computer Games.” 2006.

Apple. *Apple - MacBook Air.* 2008. http://www.apple.com/macbookair/features.html (accessed March 05, 2008).

Applied-Science-Laboratories. *Applied Science Laboratories - Eye Tracking Expertise.* 2007. http://www.a-s-l.com/ (accessed November 2007).

Atchison, D. A., and G Smith. “Basic Optical Structure of the human eye.” In *Optics of the Human Eye*, by David A. Atchison, 1-7. 2000.

Barnes, S. B. “Douglas Carl Engelbart: Developing the Underlying Concepts for Contemporary Computing.” *IEEE Annals of the History of Computing*, 1997: 16-26.

Barreto, A. B., and S. D. Scargle. “A real-time assistive computer interface for users with motor disabilities.” *ACM SIGCAPH Computers and the Physically Handicapped*, 1999: 6-16.

Baudel, T, and M Beaudouin-Lafon. “CHARADE: Remote Control of Objects using Free-Hand Gestures.” *Communications of the ACM*, 1993: 28-35.

Bergström, S, and L Råberg. *Adopting the Rational Unified Process: Success with the RUP.* Redwood City: Addison Wesley Longman Publishing Co, 2003.

Bull, J. M., L. A. Smith, L. Pottage, and R. Freeman. “Benchmarking Java against C and Fortran for scientific applications.” *JGI '01: Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande*, 2001: 97-105.

businessweek.com. *The Big Ideas Behind Nintendo's Wii.* November 2006. http://www.businessweek.com/technology/content/nov2006/tc20061116_750580.htm (accessed February 26, 2008).

Butterworth, S. “On the Theory of Filter Amplifiers.” *Wireless Engineer*, 1930: 536-541.

Cao, X, and E Ofek. “Evaluation of alternative presentation control techniques.” *CHI '05 extended abstracts on Human factors in computing systems*, 2005: 1248-1251.

cplusplus.com. *istream::getline.* 2008. http://www.cplusplus.com/reference/iostream/istream/getline.html (accessed January 25, 2008).

Dam, A. “Post-WIMP user interfaces.” *Communications of the ACM*, 1997: 63-67.

Eickeler, S, and A Kosmala. “Hidden Markov Model Based Continuous Online Gesture Recognition.” *14th International Conference on Pattern Recognition (ICPR'98) - Volume 2*, 1998: 1206 - 1208.

Ferkat, S, and J S. Weizer. “Macular Hole.” In *All About Your Eyes*, 87-88. Duke University Press, 2006.

Fitts, P. “The information capacity of the human motor system in controlling the amplitude of movement.” *Journal of Experimental Psychology: General 121*, 1992: 262-269.

Graham, D.R. “ Incremental development and delivery for large software systems.” *IEE Colloquium on Software Prototyping and Evolutionary Development*, 1992: 1-9.

Herz, JC. In *Joystick Nation: How Videogames Ate Our Quarters, Won Our Hearts, and Rewired Our Minds*, 13-14. Boston: Little, Brown, and Co, 1997.

iHackEdge. *Wiimote as a mouse.* 6 Dec 2006. http://edge.i-hacked.com/wiimote-as-a-mouse-now-with-sensor-bar-control (accessed March 26, 2008).

Jacobs, R J. K. “The use of eye movements in human-computer interaction techniques: what you look at is what you get.” *ACM Transactions on Information Systems*, 1991: 152-169.

Katz, David H. Hutchens and Elizabeth E. “Using Iterative Enhancement In Undergraduate Software Engineering courses.” *ACM SIGCSE Bulletin*, 1996: 266 - 270.

Larman, C, and V Basili. “Iterative and incremental developments. a brief history.” *Computer*, 2003: 47-56.

Lee, H, and J.H. Kim. “An HMM-based threshold model approach for gesture recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999: 961-973.

Lee, J. C. *Wii Project.* 2007. http://www.cs.cmu.edu/~johnny/projects/wii/ (accessed December 20, 2008).

Microsoft. *Double Clicking.* 2002. http://www.microsoft.com/enable/training/windowsxp/doubleclickspeed.aspx (accessed Jan 28, 2008).

Microsoft. *Microsoft Surface.* 10 Dec 2007. http://www.microsoft.com/surface/index.html (accessed February 25, 2008).

Microsoft. *PowerPoint Shortcuts.* 2003. http://office.microsoft.com/en-us/powerpoint/HP051955191033.aspx (accessed March 13, 2008).

Moses, R. A. “Adler's Physiology of the eye clinical application.” In *Adler's Physiology of the eye clinical application*, 1-15. 1981.

Prechelt, L. “Technical opinion: comparing Java vs. C/C++ efficiency differences to interpersonal differences.” *Communications of the ACM*, 1999: 109-112.

QWERTY. “An HMM-Based Threshold Model Approach.” *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 21*, 1999: 961-973.

Rao, R. S., and R. Seliktar. “Evaluation of an isometric and a position joystick in a targetacquisition task for individuals with cerebral palsy.” *IEEE Transactions on Rehabilitation Engineering*, 2000: 118-125.

Rose, R. C. “Discriminant Wordspotting Techniques for Rejection Non-Vocabulary Utterances in Unconstrained Speech.” *IEEE International Conf. Acoustics, Speech and Signal Processing vol. II* , 1992: 105-108.

Shneiderman, B. “An empiricle comparison of pie vs linear menus.” In *Sparks of Innovation in Human-Computer Interaction*, 79-84. intellect, 1993.

Sibert, L. E. “Evaluation of eye gaze interaction.” *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000: 281-288.

Ware, C. “An evaluation of an eye tracker as a device for computer input2.” *Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, 1987: 183-188.

Wilson, K. S., and M Inderrieden. “A Comparison of Five User Interface Devices Designed for Point-of-Sale in the Retail Industry.” *Human Factors and Ergonomics Society Annual Meeting Proceedings,*, 1995: 237-277.

8 Appendices

8.1 Appendix A – Personal Reflections

With the project and the report now fully complete, I am now able to stand back from the cocoon that seemed the 'Final Year Project' and reflect openly and honestly about my thoughts and my recommendations to others who will be partaking in what seems at first a mammoth task.

When starting a project of this scale, it is important not to panic at the sheer amount of work that needs to be done, instead be systematic and choose an effective methodology that will allow your project to evolve and develop in stages. Be aware that you won't be able to remember every aspect of your code and the time in which you developed it, for this I recommend keep a project diary, in doing it allowed me to indicate where I had delays and comment on any particular problems I had on those days.

My personal recommendation to anyone who is thinking of a final year project is to choose wisely, it is easy to pick a project that sounds impressive or one that you think will have plenty of documentation on, but if you are not completely interested in your project topic you will soon find yourself neglecting the project and losing all enthusiasm. The project allowed me to stay enthusiastic as I am a great believer in trying to reinvent the wheel, the physical mouse has been such a prominent figure in modern computing to the point where it is such a commodity that many don't that there could be a better way. When I tell people the aims of my project however, many are shocked that such a system could be developed when in fact the technology to do so has been around for many years.

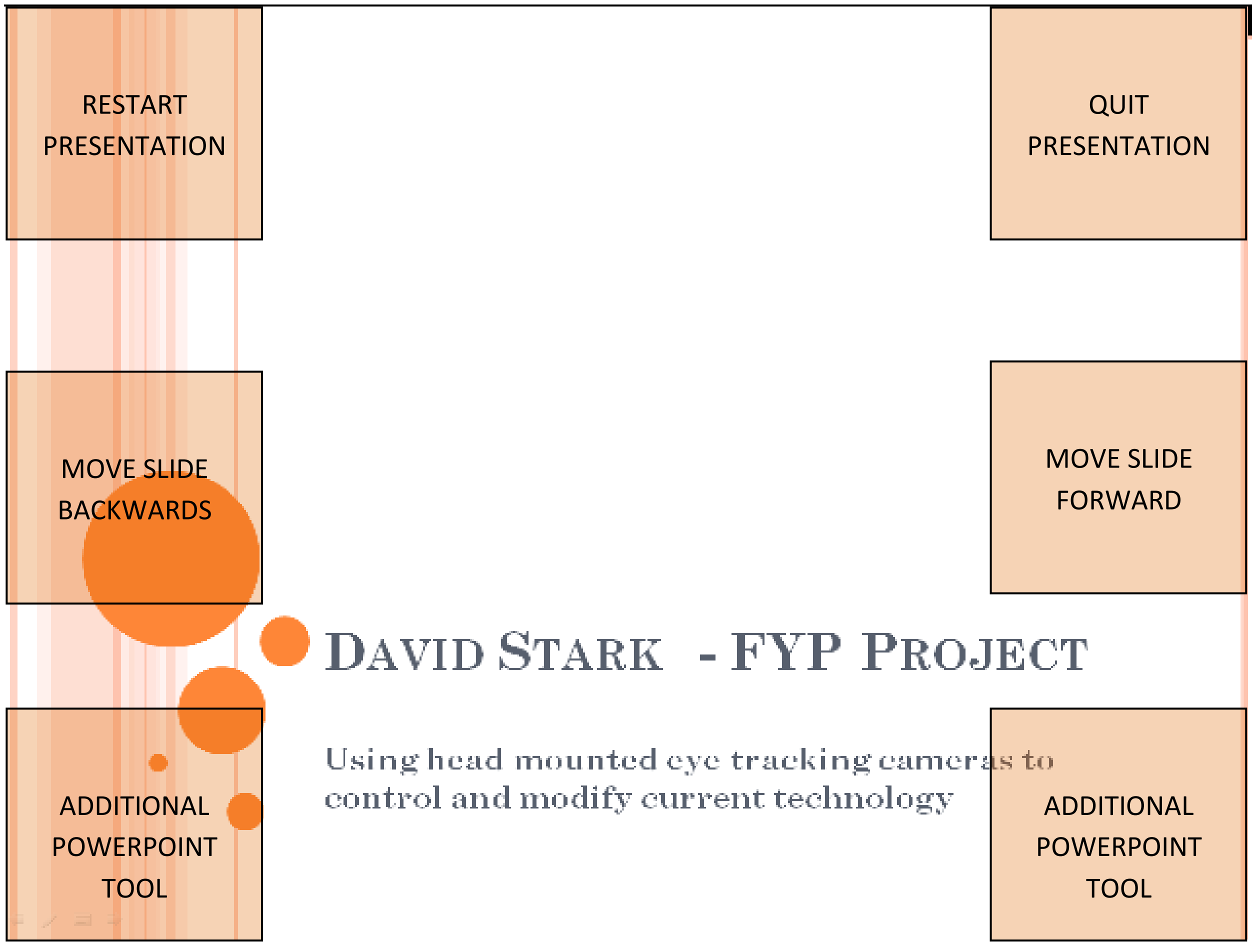
A year in industry, coupled with an effective methodology and schedule helped me to effectively manage my time in what was a tight schedule. I recommend to others to remember that your time will never be as evenly distributed as you may think; lectures and coursework do take up a half of your time and your personal life and interests will in many cases take up the majority of the other half. A good way to try and schedule in your project work is to treat the project as any other lecture, reserve some hours a week that will always be for the project and nothing else.

My final advice to students attempting a final year project is to create a plan for your project report early. Suggest the titles and subsections then review them with your tutor, when you have the skeleton of your report it is a lot easy to write individual sections about your project as and when you complete them.

Finally students should remember there is nothing better than after a year of hard work to sit back and be proud of an application that you have solely created and documented, I find its best to picture that moment and carry on with the hard work as that picture will come quicker than you think.

8.2 Appendix B – Images from project

Fig. 6



RESTART PRESENTATION

QUIT PRESENTATION

MOVE SLIDE BACKWARDS

MOVE SLIDE FORWARD

DAVID STARK - FYP PROJECT

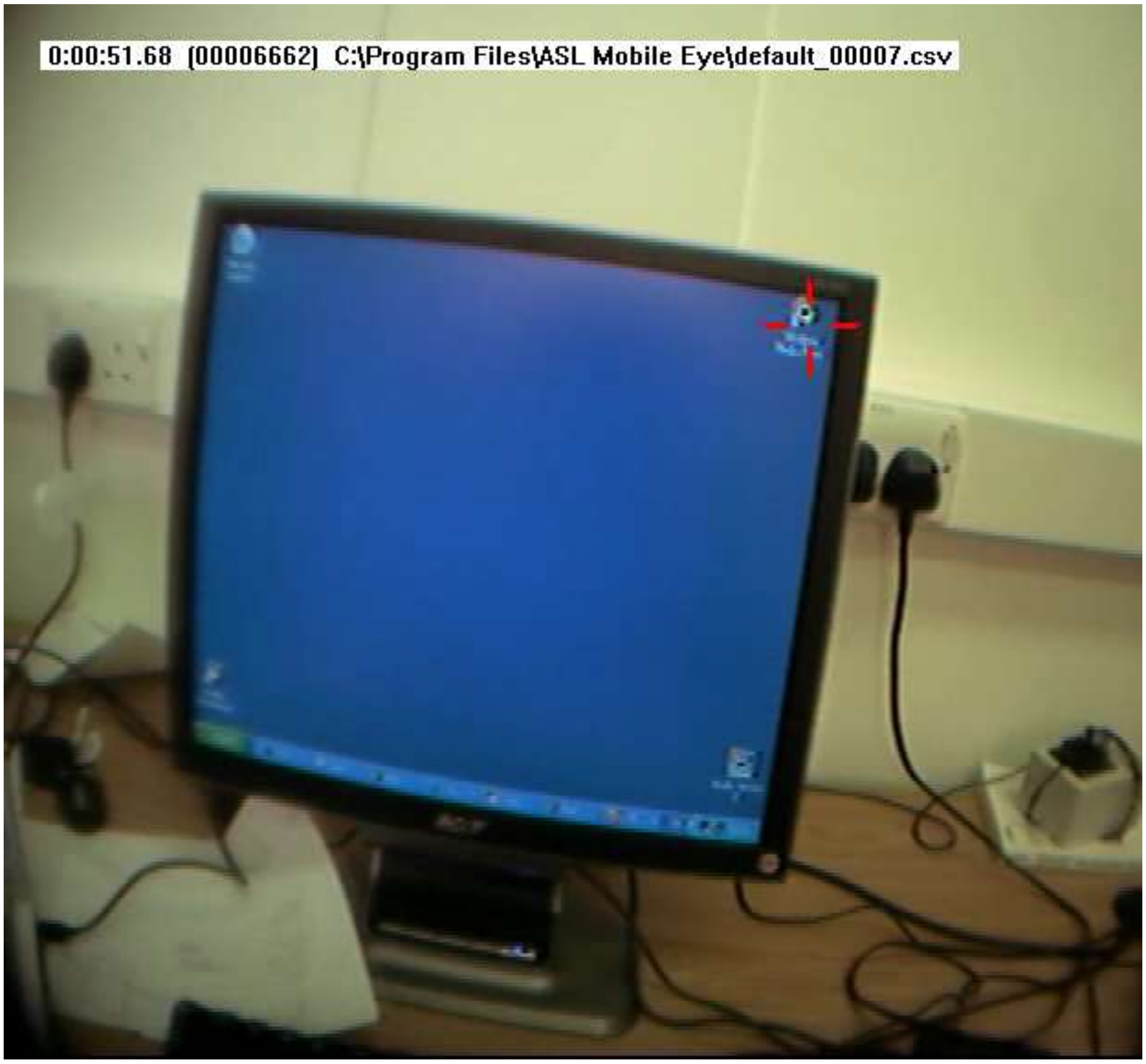
Using head mounted eye tracking cameras to control and modify current technology

ADDITIONAL POWERPOINT TOOL

ADDITIONAL POWERPOINT TOOL

Fig. 12

Fig. 13

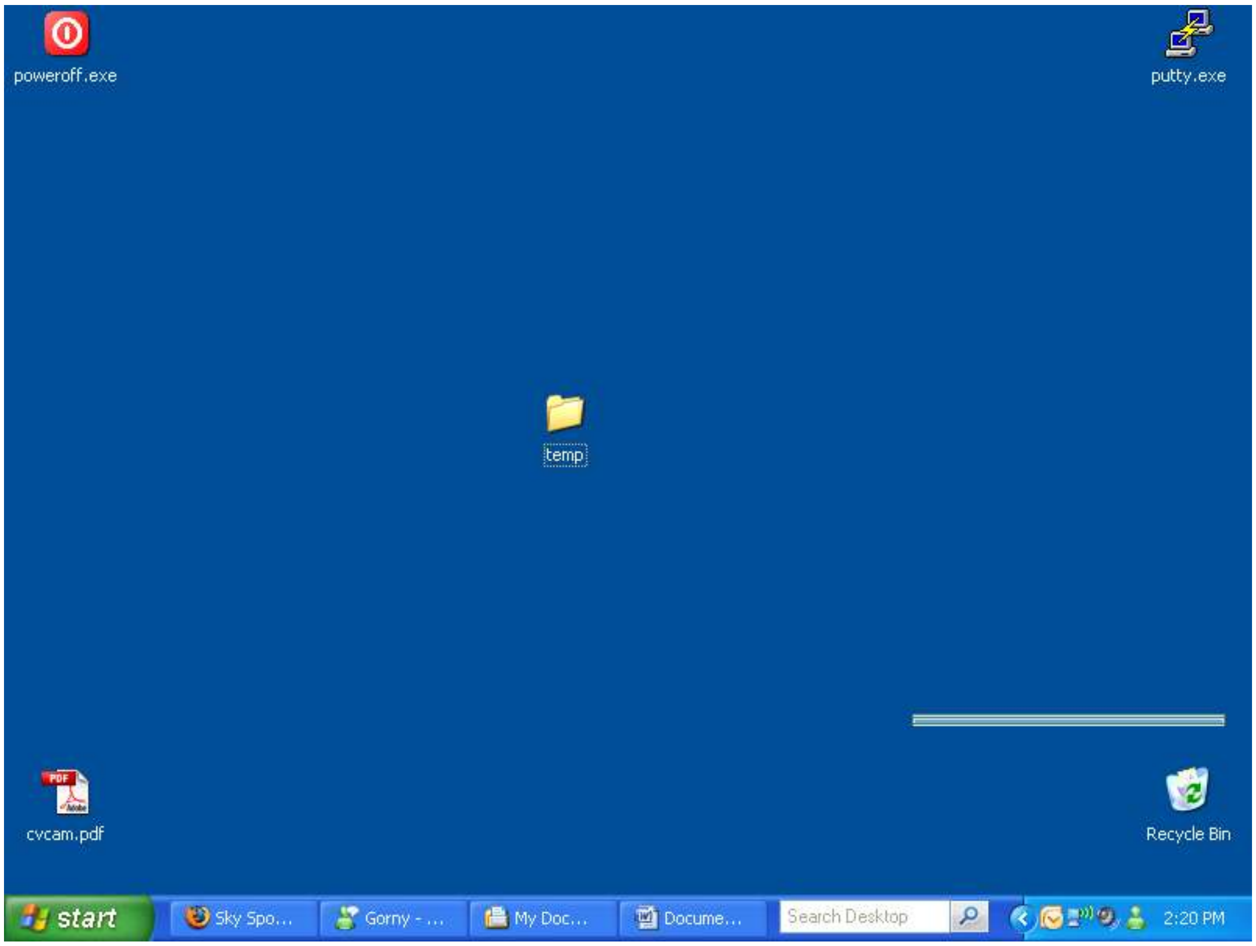

0:00:51.68 (00006662) C:\Program Files\ASL Mobile Eye\default_00007.csv

8.3 Appendix C – Test Data Document

Data Collection

Set up

- Screen Resolution 800 x 600 px
- A desktop icon in each corner of the screen



Training the eye tracker

I have implemented a training program, which will attempt to learn what a “click” is defined as, in terms of time.

I have implemented two different clicking scenarios, one using a gaze method and the other blinking. Thus requiring two different training sets.

If at least 2 users could perform all the tasks, I believe that would be sufficient.

Training Blinking

I would ask that a user performs the following actions, so that testing can be standardised:

- Focus on the centre of the screen for 2 seconds
- Look at the top left icon and blink for a time, they deem to be worthy of a click. (This maybe .5 of a second or a second etc...)
- Focus on centre of screen for 2 seconds
- Then go to top right, blink, back to centre.
- Repeat for all the corners of the icons twice

Training Gazing

- Similar to blinking, the user should focus at the centre of the screen for 2 seconds
- Then in turn, look at the icons, starting at top left, gazing for as long as the user believes they should gaze before a click is evoked
- Returning back to the centre of the screen before each icon is gazed at

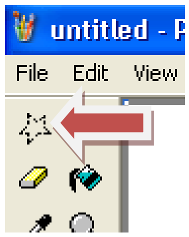
If possible, please can the training files be saved as, User1BlinkTrain.avi and User1BlinkTrain.csv or similar names that contain the User and the task at hand...

Cutting out image

While still in 800 x 600 resolution. If the users could open up the attached image in Windows Paint (same as below)

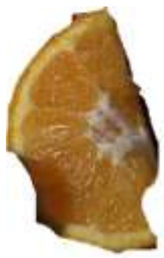
The aim of this test is to quantify how useful the eye tracker may be in cutting out images with pure vision alone

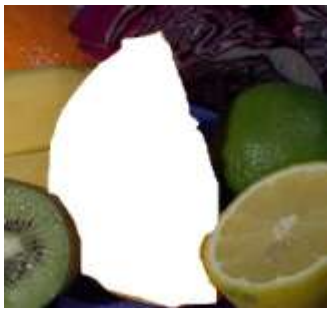
The algorithm works by allowing the user to focus on a certain point (for about a second). This will then imitate holding the mouse button down, the user should then start to trace around the image, returning to the initial point. This is course using the freehand cutting tool provided with paint.



Firstly I would like both users to cut out the image using the freehand cutting tool and the physical mouse.

Then cut the image, and paste into a new .jpg file, saving both the images, so that the result looks something like:





Then using the eye tracker I would like the users to

- focus at the top of the orange piece for approx 1 second
- then, anti-clockwise trace around the orange piece
- do so as smooth as possible and only tracing the orange, trying not to include any of the other fruit pieces
- Then return to the top of the orange and gaze for a further 1 second

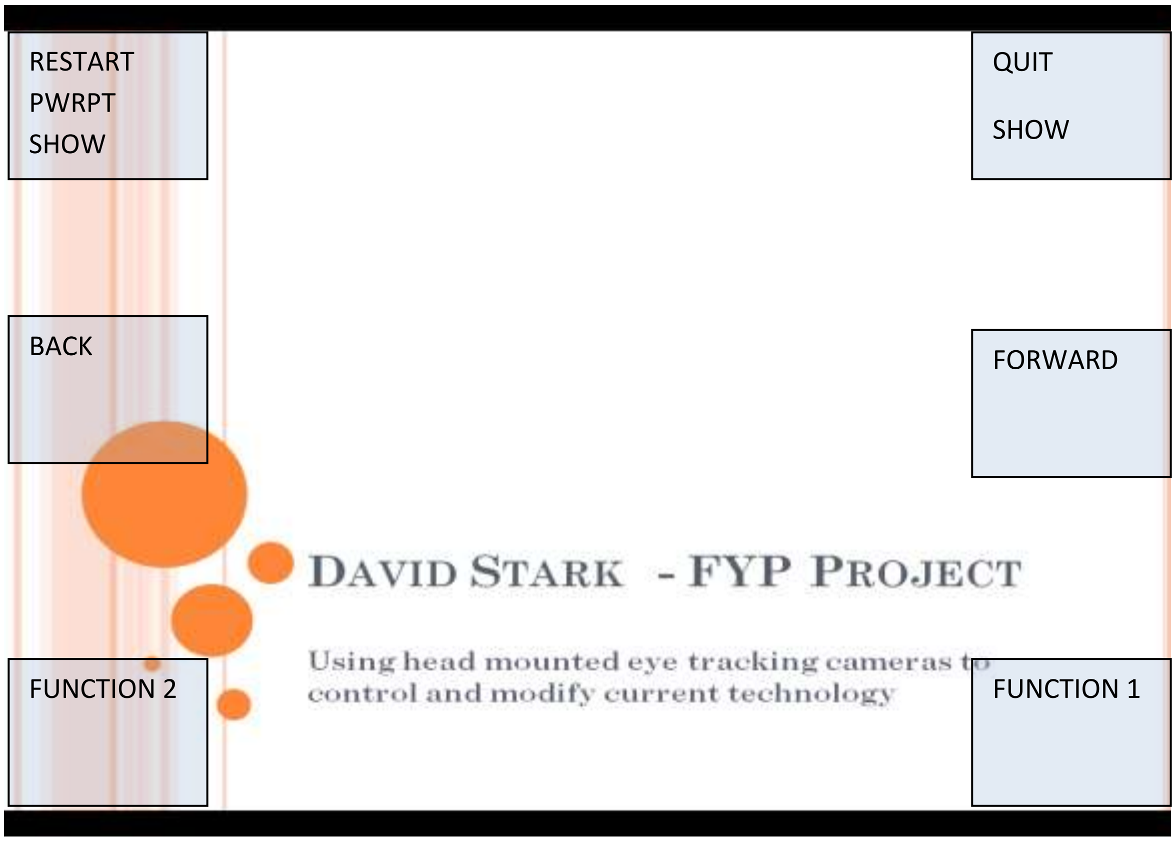
Following the test, can the users rename the files to suitable names for the test, with User1 and User2 within the title of the filenames.

PowerPoint

The next set of testing is to try and control a power point.

This will be similar to the gazing method as described previously.

To explain further, a system has been implemented by which active area's are defined when a PowerPoint is displayed in full screen. <see image below>


RESTART PWRPT SHOW
QUIT SHOW
BACK
FORWARD
DAVID STARK - FYP PROJECT
Using head mounted eye tracking cameras to control and modify current technology
FUNCTION 2
FUNCTION 1

The user can’t see the active area’s however they should have some knowledge of them being there before use.

To test this, I would like the Users to load up a PowerPoint show of more than 5 slides.

I would then like the users to perform the following

- Firstly make the PowerPoint show full screen by pressing F5
- Then gaze at the centre of the PowerPoint show
- Focus on the middle right of the screen for approx half a second
- Then focus on the center
- I would then like the user to manually skip a slide to signify that they have “moved” a powerpoint slide
- If they could do this for 5 slides forward.
- Then repeat for 5 slides back.

- For a quick test of the other 2 active area’s, if possible could the users skip to a random slide, then look at the center of the screen, then at the top left of the slide for approx half a second
- Finally, looking back at the centre for 2 seconds, can the users look at the top right hand of the screen. Then back at the center.

Again, can the users rename the files to suitable names for the test, with User1 and User2 within the title.