Automating Camera Control in Games Using Gaze

Cameron Alston and **Arnav Jhala**
UC Santa Cruz
Computational Cinematics Studio
1156 High St, E2-262
Santa Cruz, California 95060
{calston, jhala}@soe.ucsc.edu

Abstract

This paper introduces a work in progress method for automating camera control in games using gaze tracking. Building on prior work by Burelli et al, we provide characterization of features for the gaze tracker and test a gaze-assisted camera control module in a first-person shooter game. We report results from a pilot study on the viability of gaze-assisted camera control. We also discuss challenges and viability of gaze-assisted camera controller for games and interactive applications. Finally, we present ongoing and future work in this area.

Introduction

In a 3D world the virtual camera serves as the window through which the user visually interacts with the digital environment, and as such has a very large influence on the user's experience with the environment. Poorly designed camera control schemes, even those that allow the user lots of freedom to move the camera however they choose, can have wholly detrimental effects on even the best designed game experiences. Another negative aspect of explicitly user controlled camera systems is that the burden of moving the camera is placed entirely on the user and is something that requires constant input and attention at all times during game play.

An automated camera control system frees the player form the burden of having to control the camera during gameplay and allows them to focus completely on their actions within the game. In this paper we summarize our in progress work on the development of a novel system for gaze based camera control. We introduce several software applications built for the project and a provide details on a preliminary user study conducted for analysis of the system. We then outline our proposed method and discuss possibilities for work beyond the initial goals of the system.

Related Work

Much work has been done in automating control of the virtual camera, using different methods and for different purposes. A lot of work in automating virtual camera control has been done for the purpose of creating improved dramatic experiences and computationally finding the optimal camera location to fit with a specific sequence of shots for a desired narrative effect. Many of these systems are constraint based and compute the best location for the camera by optimizing over the 3D environment for locations that best satisfy cinematic or dramatical constraints.

Burelli and Jhala utilized dynamic artificial potential fields for automating the camera's viewpoint in a 3D environment, optimizing the camera's ideal location and view parameters over constraints of visibility, projection size, and view angle (Burelli and Jhala 2009).

Yannakakis, et al. performed work in the area of player modeling with heart monitor biological feedback as a means for generating features for modeling which the affective states that a player experiences based on changing camera viewpoints in a game (Yannakakis, Martínez, and Jhala 2010). Their work utilized a neuro-evolutionary preference learning model with feature selection over heart rate and skin conductance bio-feedback data generated features and camera parameter information for the purpose of predicting a users emotional state while playing a game. The paper yielded positive results in that the neuro-evolutionary preference learning model was able to accurately predict the users emotional state

Work that is most closely related to the work being proposed in this paper was done by Picardi, et al (Picardi, Burelli, and Yannakakis 2011) and Burelli (Burelli 2013). In their paper the authors introduced a method for modeling the player's play styles and camera control preferences in a 3D platformer environment. The authors were able to conclude, using k-means clustering methods, that there were several distinct player and camera behaviors identified, based on the the gameplay area (fight, jump, collection). In (Burelli 2013), the author introduces a system for automating control of the camera in a 3D platforming game, with a general implementation that is quite similar to the work proposed in this paper, but differs in that is designed for a third person camera game environment and assumes that particular areas of the game will have distinct qualities for play styles, whereas this work hopes to identify camera behavior purely using models of visual attention, gaze, and gameplay in a first person shooter environment.

Characterizing Accuracy of the Gaze Tracker

In order to effectively utilize the gaze tracker in the experiments and data gathering for training the system, the accuracy of the gaze tracking device must first be verified. The bounds on the accuracy of the gaze tracker are also useful for improving the estimation of the users intended focal target object in acquiring data for training the neural network.

3 users were asked to use two different gaze tracking based applications, GazeCorners and GazeHunt. The purpose of both of the applications is for the user to follow a distinctive target moving around a 2D background with their gaze for a short period of time.

The experimental setup used in this study consisted of an Intel Core i7 laptop PC clocked at 2.8 GHz, which proved ample to run all applications sufficiently at 30 fps. The display used in the experiments is a 17" screen position 17" away from the user. The resolution of the monitor is 1920 x 1080, which results in a pixel density of 130 ppi. At the distance that the monitor was placed with the dimensions and pixel density mentioned, following the assumption that human foveal vision is within 2° (Fairchild 2005) and that the GazeTracker was calibrated to $\leq 0.5°$, the accuracy of gaze data measurement is within 2.5° which translates to a circular region with a radius of 0.43" surrounding approximately 76 pixels. The eye tracking device used in the experiments of this paper is an EyeTribe device which streams gaze data (focal location, pupil dilation, fixation/saccade) to the applications at a rate of 30Hz.

For each experiment, RMS error testing was performed, measuring the differences between the observed gaze locations and the target locations of the object in fixation in each application. The equation defining RMS error can be seen in Equation 1, where y_t represents the target (x, y) position and $\hat{y}_t$ represents the observed gaze location at each time sample.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n}} \tag{1}$$

GazeCorners

The GazeCorners application asks a user to attempt to focus on a red square that is 100 pixels on each side that remains stationary in each of the 4 corners of the screen for 5 seconds in each corner. A screenshot of the GazeCorners application can be seen in Figure 1. The point of focus is the center of the square, so the actual points that are supposed to fixated on are (50, 50), (1870, 0), (1870, 1030), and (0, 1030). The samples that are determined to be noise as a result of shifting gaze in between corners when the square changes position, and when the user must initially find the first corner are not considered in determining RMS error. The application runs for 20 seconds total and receives streaming gaze data at 30Hz. Of the expected 600 samples, the mean number of samples that were determined to be not noise was 530.

The results of the experiment are seen in Table 1. The mean RMS error for the 3 users is 49.21. A mean RMS value of 49.21 falls within the expected accuracy of the system, since the effective area of interest was 100 pixels on each side, or ±50 pixels from the target position.

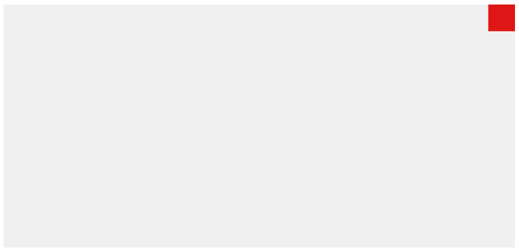

Figure 1: A screenshot from the GazeCorners application.

User	Calibration Score	Overall RMSE
1	5 Stars ($\leq 0.5°$)	48.40
2	5 Stars ($\leq 0.5°$)	55.15
3	5 Stars ($\leq 0.5°$)	44.08

Table 1: Accuracy results of GazeCorners.

GazeHunt

The GazeHunt application is similar to the GazeCorners application in that it asks a user to fixate of an object on the screen, but it differs in the fact that the object is constantly moving, and there is a secondary task of pressing a button when the target image changes color periodically. The same 3 users interacted with the GazeHunt application for a period of 180 seconds (the expected length of time that they will be playing the first person shooter game for data acquisition). The target point is the center of the image of the duck, whose dimensions are 122 x 90 pixels. An image from the GazeHunt application can be seen in Figure 2 below.

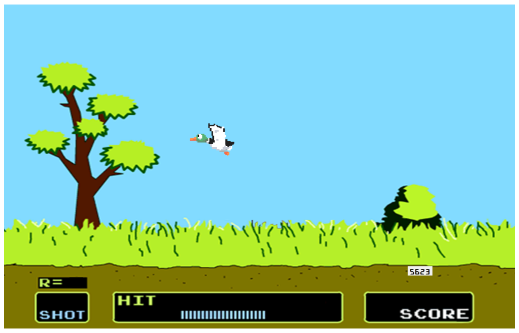

Figure 2: A screenshot from the GazeHunt application.

Upon running the application with the 3 different users, RMS errors were again calculated. The mean RMS error for the 3 users is 64.80. A mean RMS value of 64.80 represents the sample standard deviation of the distance between the target's position and the user's gaze position. The RMSE values falls within the expected accuracy of the system, with a small amount of discrepancy. Since the dimensions of the target are 122 x 90, an RMSE value of 64.80 lands most of the samples within the intended region on the x-axis, but a little bit outside of the intended region on the y-axis. This

User	Calibration Score	Overall RMSE	Q1 RMSE	Q2 RMSE	Q4 RMSE	Q4 RMSE
1	5 Stars ($\leq 0.5°$)	64.95	54.57	62.07	78.57	62.37
2	5 Stars ($\leq 0.5°$)	66.97	46.93	87.98	63.94	62.37
3	5 Stars ($\leq 0.5°$)	62.48	69.95	69.53	54.31	54.08

Table 2: Accuracy results of GazeHunt.

can likely be explained random motion of the object of interest in the y-direction, and the steady motion of the object along the x-axis. The user's gaze takes a small amount of time to refocus on the new trajectory of the object when it randomly changes direction. In addition to the error that can be attributed to the motion of the target, there is a distraction in that after the user completes the secondary task of identifying the color change in the target, the HUD element showing the score is updated. Even though samples that are determined to be noisy (more than 200 pixels outside of the target) are ignored, the motion of gaze away from the target and back to it will result in some increased error. The results of the experiment are shown in Table 2. In addition to the overall RMSE, the different RMS errors were also calculated for the first, second, third, and fourth quartiles of the data. Something to note about the results is that the data does not suggest that the device becomes less calibrated over time, so purity of the data can be assumed for longer play sessions.

User Study/Gathering Data

As a first step to the training of the learning algorithm for controlling the camera, a preliminary user study was conducted. The purpose for the user study was two-fold, to assess the usability and interest in such a system, and to acquire some initial gaze and gameplay based data for use in training the learning algorithm.

Methodology

A beta-testing methodology was utilized for conducting user tests. A playable version of the gaze controlled camera system available for users to play, and I conducted surveys both before and after they played two different versions of the same game for 3 minutes each. The questions that the users were asked upon completion of the experiment are listed in the appendix to the paper.

One version of the game used a standard WASD and mouse-look FPS control system, and the other was using gaze to control the camera with WASD movement. The initial study included 8 participants with ages ranging from 25 to 28, and of the 8 users, 6 were male and 2 were female. All of the users rated their level of expertise with the FPS game genre at between Amateur and Expert level. The testing setup is shown in Figure 3.

FPS Game

The game utilized in this study is a first person shooter game, built in Unity, and implementing a TCP client application for receiving gaze tracking data from the EyeTribe device. The game's environment is a realistic, canyonesque scene with

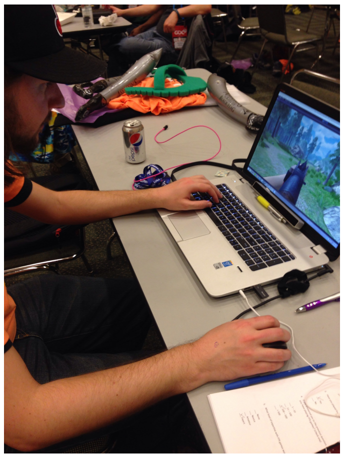

Figure 3: One of the users playing the FPS game.

rocky surroundings and several different rooms/areas for the player to explore. There are several different areas where enemies spawn, weapons are placed, and opportunities for platforming-like behavior where the player can climb and jump over rocks, ladders, roofs, and moving elevator platforms.

Controls The game used in the experiment uses a standard FPS game control scheme, with WASD player movement and camera movement using the mouse. Details of the control scheme can be found in Table 3.

Input	Action
WASD	Movement
Mouse	Move camera
Left click	Fire weapon
Space	Jump
Shift	Sprint
Ctrl	Crouch
E	Pickup item
R	Reload weapon

Table 3: Standard game controls.

For the modified version using gaze for moving the camera, all of the controls remained the same, even the mouse was given a very small amount of control over the camera to give users the ability to aim more precisely than the gaze tracking unit's expected accuracy, but the great majority of the camera's movement was driven by the gaze. The update function for the camera's view vector is shown in Equation 2. Essentially the point that the user is focused on is moved to the center of the screen on every subsequent frame, dampened to account for the speed of movement of the users focus.

$$camera(\rho,\theta)_{t+1} = camera(\rho,\theta)_t + (gaze(x,y)_t - center(x,y)) * dampening \tag{2}$$

Enemies There are 2 different types of enemies in the game and 4 different types of weapons in the game. The enemies are zombies and alien robots. The zombies move quickly and attack from close range while the alien robots carry shooting weapons and attack from a distance, but move very slowly. The enemies are shown in Figure 4.

(a) Zombie. (b) Alien robot.

Figure 4: Enemies in the FPS game.

Weapons The weapons in the game are a sword, a pistol, a shotgun, and a sniper rifle. The weapons are shown in Figure 5.

A top-down view of the entire game world is shown in Figure 6. The areas where enemies are spawn are marked on the map with red circles, and the areas where weapons can be picked up are marked with blue circles.

User Study Conclusions

All except for one of the participants expressed interest in using the gaze controlled technology again, if the bugs of the system were worked out and the control scheme fine tuned more. In particular, the majority of the users primary complaints were that were not able to accurately aim during gameplay and that they felt as though they were "chasing" the reticle around the screen with their gaze, causing them to get dizzy and experience eye fatigue. These effects were somewhat to be expected since the camera moves exactly as their eyes move with accuracy only verified within approximately 60 pixels from where the user is actually looking. As they attempted to focus on their intended target and the reticle only came within the 60 pixels, they tried to focus harder and harder to get the reticle to be exactly where they intended. Given the level of responses by the users toward the things that they liked and their interest in using the technology further, it seems as though it was a positive experience and their dissatisfaction with the accuracy of the camera and system will be ameliorated with further testing, development, and with the integration of a smarter camera control algorithm.

(a) Katana. (b) Pistol.

(c) Shotgun. (d) Rifle.

Figure 5: Weapons in the FPS game.

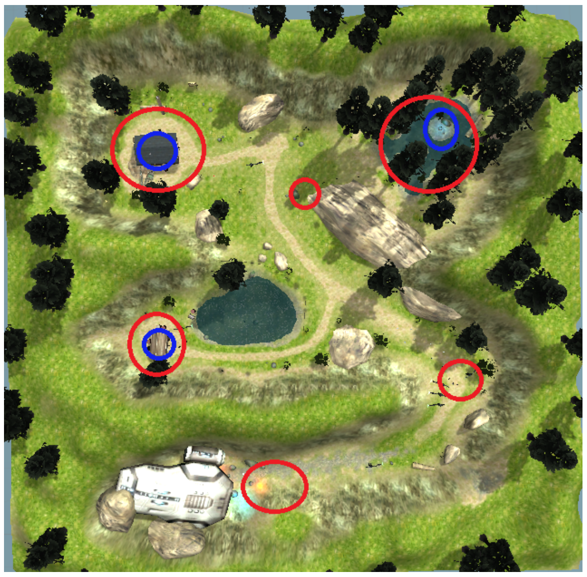
Figure 6: A top-down view of the game world.

Feature	Data Type	Description
Time stamp	Float	Time at which the events occurred.
Weapon	Enum	The weapon that the player is currently using.
Gaze point	Vector2	The (x, y) location of the user's gaze in the screen space.
Reticle vector	Vector3	The look vector of the camera down the reticle.
Player location	Vector3	The location of the player in the game world.
Normal to terrain	Vector3	Measuring the slope of the ground the player is on.
Player health	Int	Player's current health.
Reticle intersects enemy	Boolean	Whether or not the reticle is currently over an enemy target.
Gaze intersects weapon	Boolean	Whether or not the player is looking at a weapon.
Gaze intersects enemy	Boolean	Whether or not the player is currently looking at an enemy.
Gaze intersects HUD	Boolean	Whether or not the player is looking at the HUD.
Gaze intersects weapon	Boolean	Whether or not the player is looking at a weapon.
Num active enemies	Integer	Number of enemies in the area.
Num enemies in frame	Integer	Number of enemies visible to the player.
Average enemy distance	Float	How close the enemies are to the player.
Room size	Float	Represents the area of the region the player is in.
Kill score	Int	Total number of enemies killed up to this point.

Table 4: Features generated in the FPS game each frame.

Proposed System

The automated camera control system will utilize a neural network based approach to determine the best location and direction of the camera while the user is playing the game. The system is similar to the gaze controlled camera that the players played in the user study, except with a much more intelligent design for determining whether the location that the player is focusing on in the screen space is intended to be a target or not.

First person shooter games require extremely precise accuracy in order for the player to play the game effectively, which (without removing many of the the elements of challenge in FPS games) the automatic camera control system in it's current state cannot provide.

In traditional FPS games, the user controls the camera and bullets are fired down the look vector of the camera, in the automated camera control system, the reticle movement will be decoupled from the movement of the camera, so the player can aim away from the camera look vector while playing the game using the mouse. This will have the effect of addressing the issue that the players identified with not being able to aim precisely enough in that they will regain the pixel wise control that they are used to in FPS games, assuming that the automatic camera control system is able to effectively frame the most salient elements of the environment.

This type of control scheme is most similar to rail-shooter games, or shooting games that do not allow the user to control the camera, but do allow the user to aim within the viewing space. In rail-shooters the camera moves along a fixed path, whereas in the gaze-based version the camera's motion will be dynamically updated via the output of the neural network.

Generating Gaze Based Features

On each frame, the game records information about the player's gaze and various information about what is going on in the game at the time. The logged data is outlined in Table 4. Based on the responses and data gained in the initial user study, it is important that the features that are generated are able to address the issues that the players brought up when integrated with a learning-based camera control system, in particular, ensuring that the object(s) that the user is trying to focus on, attack, and acquire are present in the frame.

The logging of this data allows more features about the player's gameplay and camera control behavior to be calculated. The features provide a higher level view of the player's behavior and are necessary for modeling the player's camera control behaviors. Several of the features are adapted from the modeling setup described in Picardi, et al. and the details of the calculated features are outlined below.

- **Time watching enemies**: the amount of time the player focuses on enemies.
- **Time watching HUD**: the amount of time that the player was focused on elements of the HUD.
- **Time watching pickup items**: the amount of time the players looked at items they can interact with.
- **Time searching for targets/items**: the amount of time that the player spent looking around the environment/not focusing on anything for an extended period of time.
- **Camera speed**: the relative speed of the camera, $S = \Delta Camera(\theta, \phi)/T$ where S is speed and T is time. θ and ϕ are the polar and azimuthal angles. Relative speed is computed over the previous 1 second of game play.
- **Number of weapons picked up**: the amount and type of weapons acquired.
- **Number of health packs picked up**: the number of health recovery items picked up.

- **Average health**: the player's average health over the past 10 seconds.
- **Number of enemies spawned**: the number of enemies that the player triggered to appear.
- **Number of enemies killed**: the number of enemies that the player defeated.
- **Firing accuracy**: The number of targets hit versus the number of bullets fired.
- **Time with enemies in front**: the amount of time that players spent with enemies in vision.
- **Time with enemies in rear**: the amount of time that players had enemies primarily behind their front facing vector.
- **Time surrounded by enemies**: the amount of time that the players were surrounded (enemies evenly spaced around the player) by enemies.
- **Average enemy distance**: the distance that the player stayed away from active enemies.

Modeling Players

Similar to the work done by Picardi et al., the system will employ a k-means clustering methodology for determining the different types of player behaviors (Picardi, Burelli, and Yannakakis 2011).

The clustering will consist of gameplay features based on nearby enemies, nearby items, game objects in frame, game objects being focused on, amount of shots fired, amount of jumps, camera speed, and other features similar to the ones outlined above, and will serve to create a model that relates different camera behaviors to different types of players.

Automating Camera Control

Once the different camera profiles have been generated, an artificial neural network can be trained using the data from the users and the profiles of the players. The neural network will be trained using a structured prediction supervised learning methodology for determining the best location for the camera given the recently acquired input data from play of the game. In structured prediciton learning, the output of the A high-level visualization of the neural network can be seen in Figure 7.

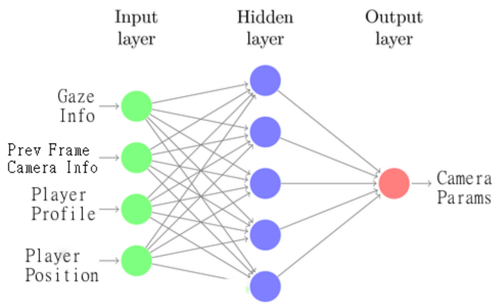

Figure 7: A high level view of the neural network to be used.

Future Work

Completing ANN Training

Currently, there is a good deal of work to be done in exploring the details of the neural network features, algorithm, and training.

Improving Features

The current system utilizes the gaze data streamed from the eye tracking device as an empirical location of focus in the game, but with more intelligent algorithms, such as one employing techniques similar to the one developed by Vidal et al. in their Pursuits application (Vidal et al. 2013), it could be possible to more accurately and intelligently identify what the player's focus of attention is on. This improvement will enable the learning algorithm to be more effectively tuned as to salient elements of the environment.

Gathering More Qualitative Data

Once the initial system is completed, a more comprehensive user study can be performed that focuses on analyzing the effectiveness of the system. Possible questions that could be asked to the users, in addition to the questions previously asked, upon playing game using the new, automated camera control system could include:

- What would you like to see changed in the automated camera control version of the game?
- What would make the automated camera control system feel more natural?

In addition to qualitative user data, quantitative user data as the the effectiveness of the camera system in allowing the user to perform well in the game when compared with the traditional camera control systems can be analyzed. Some examples of the quantitative features that could be analyzed are outlined below.

- What type of firing accuracy were the players able to achieve when compared to the user controlled camera scheme?
- How fast were they able to complete the level, comparatively?
- How effective were the players at killing the enemies and avoid damage?

The analysis of the above information, in addition to the previously analyzed features, would provide critical information in to what might be required for players to gain compentency with a new type of camera control system, as many of the users in the previous study expressed dissatisfaction with the gaze based system and said that they specifically felt like they were "too used" to the traditional methods.

Generalization for Other Games

Once this system has been developed and iterated on, further research in to a framework for generating a generalized feature set and learning algorithm for creating an automated camera control system in different types game environments, both 2D and 3D.

Belief Modeling

Some research has been done in to whether or not visual attention is actually a primary indicator of mental focus while playing a game. Ideally this study, and the following comprehensive user experience study, will provide a good idea about whether or not gaze data is a valid indicator of what the users are intending to see in the game, but further information about what the users have seen and believe to be true or find interesting about a game world could prove to be valuable in determining the optimal placement of the camera in an automated camera control system.

Conclusion

In this paper we have presented a new technique for automating the control of the camera in a game environment. We began by presenting initial results bounding the expected accuracy of the gaze tracking device used in the study and the results of an initial user study in gauging the interest that user's have for an automated camera control system that is based on gaze input. We followed by detailing a neural network based approach to automating the control of the camera in a first person shooter environment, using k-means clustering for generating features of areas of play and models of player camera control proles and features from gameplay and player gaze as inputs. We have further discussed the implications of such a system and outlined some possibilities for future work in the area of automated camera control using gaze input.

References

Burelli, P., and Jhala, A. 2009. *Dynamic Artificial Potential Fields for Autonomous Camera Control*. AAAI Press.

Burelli, P. 2013. *Adapting Virtual Camera Behaviour*. Foundations of Digital Games.

El-Nasr, M. S., and Yan, S. 2006. Visual attention in 3d video games. In *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '06. New York, NY, USA: ACM.

Fairchild, M. 2005. *Color Appearance Models*. The Wiley-IS&T Series in Imaging Science and Technology. Wiley.

Picardi, A.; Burelli, P.; and Yannakakis, G. N. 2011. Modelling virtual camera behaviour through player gaze. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, FDG '11, 107–114. New York, NY, USA: ACM.

Vidal, M.; Pfeuffer, K.; Bulling, A.; and Gellersen, H. W. 2013. Pursuits: Eye-based interaction with moving targets. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, 3147–3150. New York, NY, USA: ACM.

Yannakakis, G. N.; Martínez, H. P.; and Jhala, A. 2010. Towards affective camera control in games. *User Modeling and User-Adapted Interaction* 20(4):313–340.

Appendix

Questionnaire

After playing each version of the game for 5 minutes each, the users were asked the following questions:

1. Which camera control system did you prefer, the mouse controlled or the gaze controlled system?
2. Why did you prefer the one you preferred?
3. What did you like about the mouse controlled system?
4. What did you like about the gaze controlled system?
5. Please rate the level of enjoyment you got from playing each version of the game on a scale of 1 (not fun at all) to 5 (very fun).
6. What did you find enjoyable about the mouse controlled game experience?
7. What did you find enjoyable about the gaze controlled game experience?
8. Please rate the level of frustration you got from playing each version of the game on a scale of 1 (not frustrating) to 5 (very frustrating).
9. What did you find frustrating about the mouse controlled game experience?
10. What did you find frustrating about the gaze controlled game experience?
11. Please rate how easy it was to play each version of the game on a scale of 1 (not easy) to 5 (very easy).
12. What suggestions do you have for improving the gaze controlled game experience?
13. Would you be interested in seeing more of this type of game technology available?