12th IMEKO TC1 & TC7 Joint Symposium on
Man Science & Measurement
September, 3 – 5, 2008, Annecy, France

EVALUATION OF REAL-TIME EYE GAZE LOGGING BY A 3D GAME ENGINE

Charlotte Sennersten [1], *CraigLindley* [2]

[1] Blekinge Institute of Technology, Karlshamn, Sweden, charlotte.sennersten@bth.se

[2] Blekinge Institute of Technology, Karlshamn, Sweden, craig.lindley@bth.se

Abstract: Human Computer Interaction studies of visual attention in dynamic 3D computer gameplay can be greatly facilitated by automated gaze object logging implemented by integration of eye gaze tracking systems with game engines. This verification study reports the spatial and temporal accuracy of such an integrated system.

Keywords: Eye-tracking, gameplay, methodology.

1. INTRODUCTION

Empirical research on computer gameplay can provide scientific and empirically validated knowledge about the affects of game design features on the player and their experience, providing deeper foundations for game designers in choosing from design options to achieve specific motivational, emotional and cognitive affects.

For Serious Games there are specific issues of learning outcomes and how to create effective situated learning environments, while for entertainment games it is important to achieve a deeper understanding of game engagement and immersion, and how design decisions can shape the player experience. Studying games from a cognitive science perspective holds the promise of providing some answers to these kinds of questions. At the same time, computer games may reveal more about cognitive processes than, for example, traditional simplified cognitive stimuli [1].

3D computer games are visually rich, interactive virtual environments in which visual cognition has a primary role. Understanding how visual cognition functions in gameplay involves basic questions, such as how a player distributes their visual attention during gameplay, how visual attention is involved in cognitive task performance, how visual behavior and task performance interact with game design features, and how these factors lead to variations of player experience and cognitive learning outcomes. Studying visual cognition in 3D computer gameplay can also help in the investigation of broader questions such as: what is digital- and visual literacy?

Eye gaze tracking is a method for determining where on a computer screen a computer game player is looking. Eyetracking is a method that has been used commonly over the last 40 years to give insights into how attention is distributed when carrying out different vision-based tasks. The technique involves tracking eye movements and recording the distribution of gaze over time. Among a variety of eyetracking techniques (reviewed by [2, 3, 4]), video-based eyetracking, using combined pupil and corneal reflection, is currently the most commonly used method due to considerations of ecological validity and ease of use [5]. In this method, following a calibration process, the eyetracking system uses a camera to track variations in reflections of diode array patterns in the eye of a subject, and these variations are algorithmically decoded to calculate where upon a computer screen a person is looking at the moment when a gaze data sample is taken.

Stimuli used for gaze behaviour analysis to date have been predominantly 2- dimensional, limited to screens of 2D data with an overall static structure containing, for example, text, still images and 2D animations in fixed positions. A typical example is [6], investigating visual distribution while viewing newspapers. Recently eye gaze tracking has been used extensively to study visual attention patterns of users of web pages, e.g. to determine layout schemes and the effectiveness of advertising placement. Analysis of gaze behavior over 2D stimuli can be based upon *hotspot analysis*, where gaze density is shown by color maps laid over an image, by direct plotting of gaze traces, or by gathering statistics for specified Areas of Interest (AOIs) in a static 2D representation. AOIs are specified as polygons on the 2D surface mapping out zones within the perceptual field of the flat screen. Gaze behavior can be analyzed using statistics describing how gaze (x,y) positions fall within AOIs, providing 2D spatial distribution statistics summarizing extended visual times in specific screen areas.

There have been a number of studies where these analysis techniques have been applied to the dynamic, 3D stimuli provided by interactive computer games, e.g. [7]

analyse 3D gameplay using AOI´s. [8], [9] and [10] investigate correlations of AOI statistics with player immersion. [11] use eyetracking AOIs to investigate prediction of visual attention in complex 3D dynamic environments. [12] have developed a method for creating dynamic AOIs, based upon manual markup of AOIs on keyframes of the display of a captured gameplay session, with automated calculation of whether gaze data falls within those marked up AOIs. [13] describe a method for synchronization of gaze data with gameplay screen content by 2-d regions (not by automated 3d-object intersection from a gaze point, as described in this paper), functionality that is now available in commercial eyetracking systems. Eyetracking has been used extensively as an interaction technology, based upon gaze points falling within the bounds of 2d screen objects; [14] provides an early study of this technique. More recently, gaze tracking has been investigated as an interaction technology for immersive stereoscopic virtual reality systems [15][3], and also as an input method for commercial 3d computer games, substituting gaze position for mouse input in controlling a game cursor [e.g. 16, 17]. However, this technique applied to game interaction is not used for unmodified gameplay analysis per se, and has not gone further than providing an alternative interaction technology for existing game mechanics.

Eyetracking studies of dynamic visual stimuli have been generally regarded in the eyetracking community as being a very difficult problem since established analysis methods based upon statistics are only meaningful because they are collected from 2D areas within static screens for which the corresponding visual objects are also static and known. Since most of the features of a 3D game stimulus are not fixed within a 2D screen space, analysis of gaze behaviour during gameplay has required manual examination of a recorded video capture of a game session with gaze data superimposed on the game display and manual transcription of objects under gaze points for each frame. To analyse a First Person Shooter (FPS) game session manually in this way means to go through a number of video capture frames dependent upon the screen capture rate, and then manually transcribing the objects under the gaze points (e.g. fixations) [18]. For example, at a 10 fps (frames per second) capture rate, 600 frames for every minute of gameplay must be examined and transcribed. This is very time consuming, making statistically significant studies very difficult to conduct. For example, [18] reports that four hours of subject recordings took 3 months to analyse manually.

A solution to this analysis problem, described in this paper is the development of an integrated system, where data from an eyetracker is used by the game engine to automatically log objects of interest under the gaze point in real time. This paper describes this solution and the detailed methodology and summary of results of the verification study that was conducted, not only to verify correct operation of the system, but to obtain a characterization of its accuracy for use in future studies. The paper begins with a more detailed description of the problem. Specifications for an integrated system are then described, including details both of the game engine and the eyetracking system that are to be integrated. Details of the implementation of the system are then described. The paper then describes the detailed verification methodology and the resulting spatial and temporal accuracy model of the integrated system. The accuracy model is critical for being able to characterise the accuracy and hence validity of statistics obtained by using the system in ongoing studies; without such information, data obtained cannot be regarded as reliable - the accuracy model quantifies reliability. Implications of the verification study for the use of gaze-based virtual world interaction are also discussed.

2. PROBLEM TO SOLVE

In the initial study of gaze behaviour in 3D computer game play conducted by [18], eyetracking was already well established as a technology and a method primarily for studying 2D stimuli, such as text, static images and web pages. Methods used in these former studies come into question when studying gameplay in 3D environments: a simple 2D solution cannot easily be applied to dynamic stimuli like 3D game worlds. AOIs are useful when considering visual attention directed towards visually static[1] 2D areas of a computer game display, such as Head Up Displays (HUD´s) and static Graphical User Interface (GUI) elements, but this approach cannot capture gaze behaviour in a meaningful way for perceptual objects that change in their 2D display position. This means that when it comes to 3D games, we haven´t been able to log gaze data automatically using AOIs because the 360 degree dynamic view around the player's first-person point of view axis within the game world makes everything seen within the world dynamic. In 3D games there are also typically hundreds of objects the player has to move through, with the objects within the perceptual field constantly changing both in position and type, which means that automated methods of gathering gaze object statistics based upon static 2D AOIs are unusable. Analysis is possible, but only by extremely time-consuming frame-by-frame data collection. As reported by [18], a game stimulus (e.g. a commercial game level) can be played through by player participants while visual attention is recorded by the eyetracker and represented superimposed on top of frame capture data of the stimulus. The captured video file with the stimulus as background and the superimposed eye gaze targets on top of the stimulus then has to be analysed frame by frame in order to identify and record what the players attention is focussed upon while performing tasks to meet in-game challenges. This is a very time consuming procedure that becomes impractical for a statistically significant number of players playing for meaningful play session lengths of up to half an hour each: manual analysis of 0.5hrs of gameplay, corresponding to one player/play session, requires

[1] Where 'static' refers to 2D position, since these elements may change within position, e.g. graphical changes like boundaries and colours indicating activated buttons, animations in place, etc..

approximately 45hrs of transcription analysis in a controlled experiment.

To eliminate this impractical manual analysis time, the 2D (x,y) coordinates of each gaze point can be automatically mapped into the in-game space of the 3D environment and the object(s) of gaze picked and automatically logged (see Fig.1).

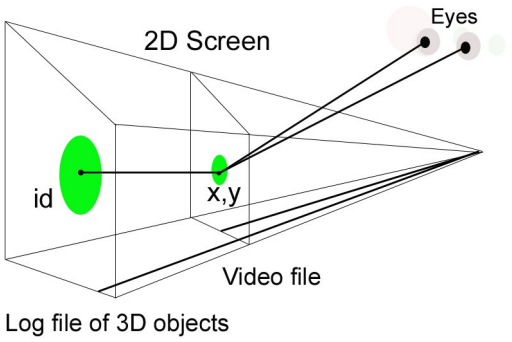

Fig.1. A ray trace from a gaze point to an object within a 3D virtual game space.

The solution developed for this purpose involves connecting two independent systems, an eye gaze tracker and a suitable game engine.

3. SOFTWARE REQUIREMENTS FOR INTEGRATING AN EYETRACKER AND A COMPUTER GAME ENGINE

3.1. Eye-Gaze Tracker Description

The eyetracker used for the work described in this paper is a Tobii Eye Tracker 1750. The 1750 runs on a PC running Windows. The system can be used either in single or double computer setups. In this study a game application is running at the same time as eye gaze tracking, which requires too much processing to do on one computer, hence we use the double computer and double screen setup.

The eye gaze tracking system detects and collects eye gaze data at a rate of 50 samples per second (50 Hz), with frame grabbing of the corresponding game display occurring at a lower but selectable frame rate. The eye gaze tracking system monitors the eyes of the subject/participant based upon reflected infrared light patterns and calculates gaze positions on the screen automatically, by hardware in combination with advanced software algorithms. The system has head motion compensation and low drift effects, with binocular tracking, which yields higher accuracy than a monocular method. Another advantage of this system is that if the participant happens to move, especially the head, tracking is not lost if one eye is occluded during the movement. (With other systems, the participants may have helmets, headrests or markers, with different respective freedoms and constraints of movement.) For ecological validity, the eye gaze tracking camera is implemented at the bottom part of a normal sized screen. The Tobii 1750 system can only track what is presented on the screen and not anything outside the screen area. The calibration time is 3 minutes if no problems occur due to eye disorders, blinking, etc.. The recommended eye distance to screen is approximately 60 cm. The accuracy of an eyetracker is measured in degrees. The *accuracy* or *bias error* of the Tobii 1750 system has been tested over a set of individuals and found to be about 0.5 degrees of visual angle when using standard accuracy measurement principles for eye trackers. One degree of accuracy corresponds to an average error range of about 1 cm between the measured and intended gaze point on the target screen at 60 cm distance from the user to the screen. The system tolerates head movements slower than about 10 cm sec^{-1}.

For each eye (left and right) the following data is available derived from the eye tracker data processed by the Clearview™ software and TET Server:

- Time, a timestamp in microseconds,
- Screen X, horizontal position,
- Screen Y, vertical position,
- Cam X, horizontal location of the pupil in the camera image,
- Cam Y, vertical location of the pupil in the camera image,
- Distance, from camera to eye,
- Pupil, size in mm
- Code, (validity of gaze data).

Validity codes ranging from 0-4 are logged for each eye, with every gaze data point. 0 is the highest validity value and means that all relevant data for the particular eye should be correct.

Fig.2. The Tobii 1750 eyetracker display with 2 infrared diode trackers and central camera at the bottom of the screen.

3.2. Computer Game Engine Description

A game engine is a software system that generates an interactive visualisation of a 3D virtual world. This is one of several 'layers' that constitute a complete game. Other layers include game logic (game rules, artificial intelligence, or AI, and interaction rules), which implement the logic of action within the game world in order to facilitate gameplay.

Game rules decide what the consequences of interaction(s) will be. Interaction with the system is via devices such as a mouse, keyboard and/or console, but could include other kinds of devices such as motion, sound and haptics. The third layer of a game is the game media layer, which includes such things as 3D models, 2D graphics, animation sequences, and audio components.

It is important to understand the difference between virtual worlds and games as such because virtual worlds (e.g. *Second Life*, secondlife.com, *ActiveWorlds*, www.activeworlds.com, *There*, www.there.com/, and *Metaverse*, metaverse.sourceforge.net) do not include instrinsic game rules by default. A computer game has certain tasks and challenges built/programmed/designed into the game and those particular tasks and challenges form the basis of game genres. Studies by Yarbus in the 1960s (summarized in [4]) demonstrate significant differences in gaze paths when viewers look at a scene with different objectives. Similar differences are apparent when investigating computer gameplay where the different objectives are associated with different game tasks and challenges.

The game platform used in this verification experiment is the HiFi Engine (HFE), a game/simulation engine developed at Man-System-Interaction, Swedish Defence Research Agency (FOI) in Linköping, Sweden. The high level operation of the game engine is a simulation loop that follows the steps shown on Figure 3. The `Start/Initialise´ phase is a gloss over what could include many metagame functions, such as forming and choosing characters, loading levels, setting control parameters, etc.. Software initialisation functions also occur here, such as instantiating classes for the GameWorld, GameObjects, ModelObjects, etc., depending upon the implementation.

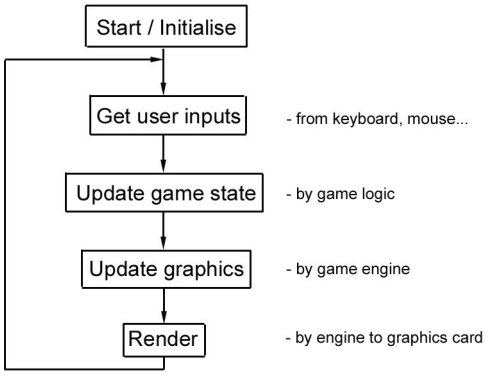

Fig.3. Typical Game Engine Algorithm per frame and/or simulation tick.

The simulation loop executes the game world simulation in which the player interacts with the world through keyboard, mouse or other devices. A simulation cycle might, for example, begin with getting user inputs. Then the game state is updated, based upon the game state in the last simulation cycle, together with the player inputs. Updates include updating simulations of physical events, actions of player characters, non-player characters and enemies (i.e. 'mobs'). Once the game state is updated, the graphics can be updated, rendered and displayed.

3.3. Requirements for an Integrated System

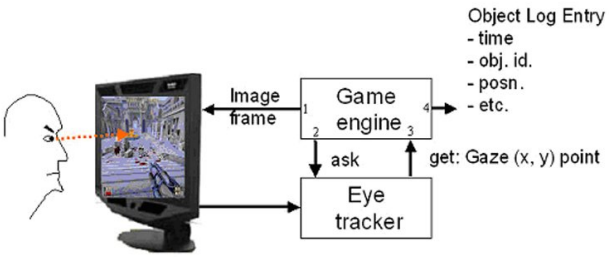

Fig.4. Eyetracker and game engine integration requirements.

The basic requirements for an integrated system are based upon the game engine being able to interrogate the eyetracking system to obtain the (x,y) screen coordinates of a gaze point, and then using that (x,y) coordinate to determine the underlying gaze object and generate a log entry for that object of gaze, as shown in Figure 4. On the game engine side, it is critical to be able to take the screen (x,y) coordinates from the eyetracker and trace back from these into the virtual 3D game world to identify the first object that is encountered, corresponding to the object of gaze, a process referred to as *picking*. This is very similar to the processing of a click at a mouse-controlled cursor position on the screen, e.g. for targeting weapons or other functions. However, the source in the current case is the eyetracker rather than a mouse, and processing of the interception point must address the generation of a gaze object log entry. The game engine must support the implementation of these functions in addition to any normal mouse (x,y) coordinate processing that occurs. (Note that for *gaze-based interaction*, the gaze point might *replace* the mouse function for implementing in-game actions, such as aiming a weapon.)

A further requirement for real time logging of objects within the game world is that each object must have understandable identity and position data for use in later analysis. Finally, it is conceptually and practically simple to implement object picking using the existing collision detection functions of an engine; then objects must have individual collision boxes so the inverted ray trace under an x, y screen coordinate can be logged in response to a specific type of collision event, in this case a collision with the ray traced from the gaze point. The veracity of object logging is then dependent upon the collision volumes around in-game objects; too large or too small a collision box for its associated graphical object can result in a misidentification of what is actually gazed upon within the game world.

4. TECHNICAL IMPLEMENTATION

The implementation of the integrated eyetracker/game engine system requires three additional steps in the basic

game algorithm described above (see Figure 5). After the rendering step in the initial algorithm, current gaze point (x,y) information is retrieved from the eyetracker. Then the object collision box under the x,y coordinates is identified, and finally the engine logs the gaze position and object data.

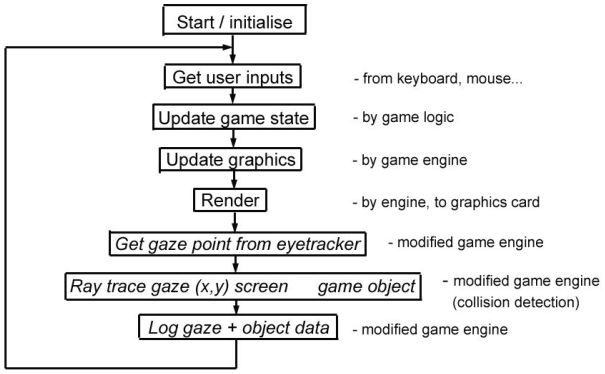

Fig.5. Modifications to the Game Engine Algorithm per frame and/ or simulation tick.

In the implementation of the integrated system, an operator computer runs the eyetracking software ClearView™ (2.6.3) and the TET eyetracking server, while a gaming computer runs the game application. The two computers are connected via TCP/IP. The TET eyetracking server gets the data from the eyetracker and sends it to the ClearView software. For calibration of the eyes of the player, the operator switches the eyetracker display over to the second screen of the operating computer that runs the Clearview software, for display of a calibration pattern (hence a dual head graphics card is needed on the operating computer for the double display setup). The output from the Game Computer goes through a splitter where two video outputs are created, one going to the video capture card on the operator computer, where the captured video data is input to ClearView, while the other output goes to a switch that selects either the game display or the operator computer second monitor display (with the calibration display) to send to the eyetracker display. The routing of the video display is illustrated in Figure 6.

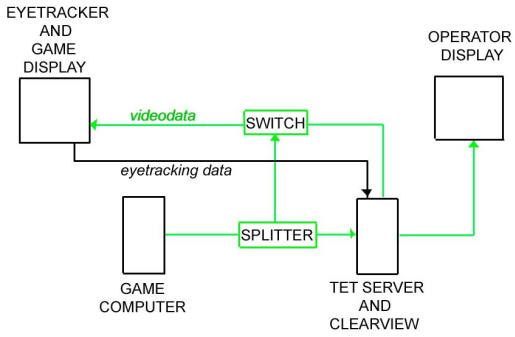

Fig.6. Data interconnection diagram showing for display routing in the dual-computer setup.

A typical play session using this system involves the game player playing the game in front of the eyetracker screen. After briefing the player with any instructions relevant to the study, the calibration process is run, calibration patterns being displayed on the eyetracker screen while Clearview™ uses the known positions of elements in the display to calibrate eyetracking for that specific player. Then the eyetracker screen is switched over to the game display via the routing switch. The operator starts game object logging within the game engine using a command line instruction, at which a synchronisation message is sent by the game engine via the TET server to Clearview™ to commence eyetracking. Synchronisation is necessary to simplify time correlation of object log entries on the game computer with video frames captured by the operating computer for the verification analysis. At an instruction from the operator, the player then commences playing while eyetracking and game engine object logging are running.

During game play, with each simulation tick or game display frame the HiFi (game) engine requests a gaze data sample from the Clearview™ software via the TET server. The game engine then traces a ray from the (x,y) coordinates of the gaze point to intercept the first collision mesh within the game world under that (x,y) point, corresponding with the object being looked upon by the player. Within the game engine, a log entry is generated that includes the name of the 3D object model appended with the (x,y) coordinates of the model instance on the horizontal plane of the 3D game world, ensuring that each object instance is uniquely identified by type and location, together with a time stamp generated by a trigger signal from the eyetracker collected with a high-resolution timer, based on Windows multimedia timer. The log record is then appended by the game engine to the object log file stored on the gaming computer. When the goals of the session have been met, the operator asks the player to stop playing. A command line instruction on the gaming computer is then used to turn off the object log and eyetracking.

5. VERIFICATION STUDY OF THE INTEGRATED EYETRACKER AND COMPUTER GAME ENGINE

The verification of the integrated system aims to establish correct functionality of the software and to characterise the accuracy of logged data based upon an investigation of the forms and degrees of error in the system. Such a study is critical for use of the system in experimental studies. Steps in the Verification Study include:

i) Prepare Stimulus

ii) Select Player Participant(s)

iii) Log Play Data (*generated by HiFi Engine)* and Game Capture Screen Data

iv) Create Transcription (*the old manual method)*

v) Synchronise record times from the Game Engine Object Log and Video Transcription

vi) Run Time Ordering Macro

vii) Conduct detailed analysis of time ordered data – *compare auto vs manually interpreted objects of gaze*

viii) Derive Accuracy Model

The verification study uses a game level stimulus consisting of a 3D virtual model of a World War II European town, with scattered enemy soldiers serving as combative game challenges. The aim for the player is to reach a central church in the town. Verification of automated gaze logging involves logging all gazed objects in real time during a play session, with parallel video capture of game play with superimposed eye gaze positions. To determine if the engine logged the gaze objects correctly, the object log data is manually compared with the video capture data. The video capture data with the superimposed eye gaze data has to be transcribed manually into text records, with a transcription entry stating for each log entry: the object identity number, the kind of object (suburban house, gun, fence, soldier, etc.) and also the duration of the eye gaze on the object.

When the recording and the transcription are done, the actual verification procedure starts. Verification of the real time logging by the game engine involves manual comparison of corresponding object identity numbers, object type and log record time from the engine log with the transcribed information obtained from the video capture data. Frame rates and timing are crucial to consider in analysing and verifying the data generated by this system. The game engine has its own tick-by-tick simulation update rate, the eye tracker has a different sampling rate and the real time video capture has yet another frame rate, as shown in Figure 7. The gaze point sampling rate of the 1750 eyetracker is a constant rate of 50Hz. The display frame update rate of the game engine varies, depending upon the complexity of the visual scene, within the range 25-35 ms. Video data capture was set at 10 fps (frames per second) for the present study. These different rates must be correlated during analysis in order to obtain the most accurate correspondences between gaze data, game frames and video frames, although, due to variable latencies in the system, the correlations are never perfect. Latency is defined as the time taken from the generation of a signal to its reception, in this case including time taken for eyetracking camera exposure, transfer to the TET server, calculation and delays in the server, transfer to Clearview™, request generation from the game engine, processing time in the TET server, and response propagation back to the game engine.

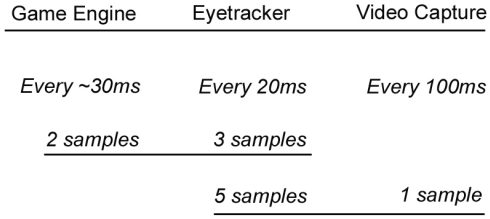

Fig.7. Sampling time of logging, gaze- and video capture.

The start time of the Game Engine also has to be synchronised with real time video capture in order to correlate log entries and video frames. Hence the starting time of one of the files must be offset and all subsequent times within the file adjusted accordingly to gain the best overall synchronisation between log record times and frame times for apparently corresponding gazed objects. The object logs and time-adjusted transcription records are then placed in a single spreadsheet and a sorting macro is run in order to sort all records of both kinds into a single time order.

The analysis then consists of a manual row-by-row comparison of object log records with what should be the corresponding video transcript data, noting: when objects correspond, time offsets if there appears to be a lead or lag of a corresponding object, and also object ambiguity, where a single object in the log corresponds with more than one candidate object in the video transcription. In general it is possible to take a subset of samples within these files to reduce the amount of data that needs to be analysed for a verification study, but in this initial study all records were used. This amounted to the manual investigation of 26,570 rows of log data and 7,560 captured video frames for the first run through the verification process. For the first attempt, the system could not be verified, and no plausible synchronisation between the object log and the video transcript could be found. It was then determined that the ray tracing did not work properly, actually invalidating the log data in a way that was not obvious by looking at the log file alone. The verification study had to be repeated once this fault was corrected, this time generating 13,285 game engine log entries and 3,780 transcribed video frames entries for a 5minute play session.

Analysis of the log data in comparison with the video transcript results in an accuracy model for the system. *Accuracy* is the degree of veracity (or closeness to the truth) of a measure, while *precision* is the reproducibility of the measure of accuracy. In the verification study, data was extracted from the sorted and analysed object log and video transcript table to characterise accuracy in both the spatial and the temporal performance of the system. *Spatial accuracy* is assessed by comparison with a known actual number of objects within the region of accuracy of the eyetracking system. The result can be expressed in terms of: for a given record for an object within the object log, what is the probability that there were actually *n* objects under the gaze point, for an arbitrary +ve integer *n*? *Temporal accuracy* is assessed by comparison of log entry time with the best estimate of the actual time when a specific object fell under the gaze point during one simulation time cycle of the game engine. The result can be expressed in terms of: for a given record for an object within the object log occurring at a time t_{log}, what is the probability that the actual time at which the gaze occurred is t_{log} +/- t_{offset}, for t_{offset} in some known real number range of seconds? Both temporal and spatial *precision* were not analysed in detail, since this would require a comparison across several studies, requiring an impractical amount of investigation time.

6. RESULTS

After analysing the data and categorized frequency count of time offsets, human and technical errors it was found that system performance in this study has an 81% time accuracy of total offsets of log entries, meaning that 81% of log entries fall within 1 standard deviation of the time noted in the log file (see Figure 8). The positive offset time error to the extreme right represents cases that are uninterpreted due to image occlusion by a gaze fixation indicator.

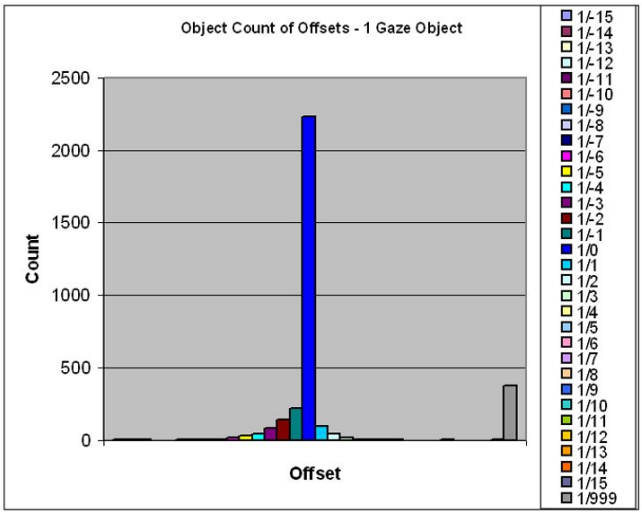

Fig.8. Frequency count of time offsets of object log entries from object visibility in the video transcript.

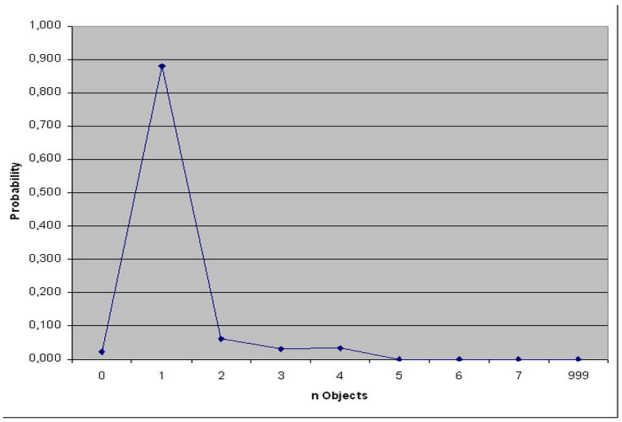

Fig.9. Probability of n Objects Under Gaze Point.

The verification study also resulted in a spatial accuracy characterization, expressed in terms of the probability of n objects falling within the spatial accuracy of the eyetracker, when a single object is logged under the gaze point by the game engine. As shown on Figure 9, one object logged corresponds with one object within the gaze accuracy area in 89% of cases.

7. DISCUSSION AND CONCLUSION

The verification study described in this report has shown that the integrated game engine and eyetracking systems can operate together with a quantifiable accuracy that is sufficient to conduct statistical studies of eye gaze behaviour in relation to dynamic, 3D stimuli. There are some changes that could improve the accuracy of the verification study, such as improving accuracy over time by higher frame/sampling rates in the game engine, synchronised start times for the game engine and video capture, and having a cursor on top of the video file that is transparent so verification objects are never obscured and can be more easily interpreted (e.g. as provided by Senso Motoric Instruments, SMI, http://www.smi.de/home/index.html, which uses an unfilled contour as a marker/cursor for representing gaze data superimposed on top of captured video). The accuracy of the integrated system could be increased by higher frame/sampling rates in the game engine and/or eyetracker, and by higher spatial resolution of the eyetracker. Spatial accuracy is highly dependent upon the design of the stimulus and can be increased by having collision volumes that closely bound graphical meshes, and minimising high spatial frequency areas of a level, especially in the form of reducing clustering of small objects or object parts and boundaries within the scale of accuracy of the eyetracker. There is also a fundamental question of how graphics are assimilated to the concept of an object of interest. E.g. for a human character, is an object of interest the whole character, a body part such as a head or a finger, or a pocket on clothing? The answer to this may depend upon the goals of a study, a particular study participant, their tasks, etc., and in this sense is a factor to be determined by a study rather than presumed. More flexibility in the system in this respect would be facilitated by using collision volumes organized as hierarchical object assemblies, e.g. representing a whole character and their perceivable subdivisions; this could, however, greatly complicate stimulus development. In any case, it is clear that the method can be used for investigating 3d gameplay with statistically significant participation group sizes. The basic gaze object analysis time then decreases from 45 hours per 30 minutes of play time [18] to 0.

Accuracy considerations are relevant to the use of gaze tracking for game and virtual world interaction, as well as for analysis. Gaze interaction has been used to provide access to common computer functions, such as email, word processing and web browsing for people who are unable to use mouse and keyboard interaction (e.g. http://www.tobii.com/assistive_technology/products/product_overview.aspx). Eye gaze tracking systems are expected to decrease in cost, eventually to a level of accessibility as mass market products. This raises the possibility that gaze-based interaction could become a common feature of mass market software systems, including computer games. There have been a number of experiments using gaze-based interaction for control of computer gameplay. For example, [19] describes a gaze-driven chess game, [16] compares several mouse and gaze interaction modes for the games *Sacrifice* and *Half-Life*, [20] compare gaze and mouse steering in the Breakout game, and [21] compares gaze and mouse control for the simple *Blob* game. The commercial game industry is also reported to have expressed interest in a controller that can simplify interaction [22].

The use of gaze tracking as an interaction technology requires design practices that minimize the impact of error sources, such as the mismatch of collision volumes with graphics, upon the player experience. The observed timing inaccuracies are not likely to have a major impact in computer game play, since they are within a small number of frames at a high frame rate. Spatial ambiguities can have an impact, and this should be minimized by minimizing spatial clustering and reducing requirements for game actions that occur at large (virtual) distances within the game design.

ACKNOWLEDGMENTS

This work has been conducted in the context of a joint project between Gotland University, Blekinge Institute of Technology and Swedish Defence Research Agency, Sweden, and also in the context of the EU FUGA (NEST- 28765) project.

REFERENCES

[1] C. A. Lindley and C. Sennersten, "A Cognitive Framework for the Analysis of Game Play: Tasks, Schemas and Attention Theory", Workshop on the Cognitive Science of Games and Game Play, The 28th Annual Conference of the Cognitive Science Society, Vancouver, Canada, July 26-29 2006.

[2] C. Sennersten, J. Alfredson, M. Castor, J. Hedström, B. Lindahl, C. Lindley, E. Svensson, "Verification of an Experimental Platform Integrating a Tobii Eyetracking System with the HiFi Engine", Methodology Report, Department of Man System Interaction, FOI -Swedish Defence Research Agency, Sweden, 2007.

[3] A.T. Duchowski, "Eye Tracking Methodology –Theory and Practice" Springer-Verlag London, UK, 2003.

[4] A.T. Duchowski, "Eye Tracking Methodology –Theory and Practice", 2nd edition, Springer-Verlag London, UK, 2007.

[5] Tobii Technology, "User Manual: Tobii Eye Tracker and ClearView analysis software", Sweden, 2005.

[6] J. Holsanova, H. Rahm, K. Holmqvist, "Entry points and reading paths on newspaper spreads: comparing a semiotic analysis with eye-tracking measurements", Visual Communication, Vol.5, No.1, 65-93, SAGE Publications, 2006.

[7] J.A. Renshaw, "Eye Tracking for Dynamic Scenes: enhancing the study of game playing behavior", ACM 2007, http://www.leedsmet.ac.uk/inn/RIP2007.htm, last accessed 23 January 2008, 2007.

[8] P.R. Kearney, M. Pivec, "Immersed and how? That is the question?", Games in Action, Gothenburgh, Sweden. http://www.learnit.org.gu.se/digitalAssets/862904_kearney_pivec.pdf, last accessed 23 January 2008, 2007.

[9] A.L. Cox, P.A. Cairns, N. Berthouze, and C. Jennett, "The Use of Eyetracking for Measuring Immersion", Workshop on What have eye movements told us so far, and what is next? 28th Annual Conference of the Cognitive Science Society (CogSci06), Vancouver, Canada, July 26-29 2006.

[10] T.J.W. Tijs, "Immersion in Games: An Eyetracking Study", Master's thesis. Department of Computer and Systems Science, Royal Institute of Technology, Stockholm, Sweden, 2006.

[11] R.J. Peters, L. Itti, "Computational mechanisms for gaze direction in interactive visual environments", In Proceedings of the 2006 Symposium on Eye tracking Research & Applications, San Diego, California, USA, 27 – 32, 2006.

[12] M.S. El-Nasr, S. Yan, "Visual Attention in 3D Video Games", In Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, Hollywood, USA, June 14-16, 2006.

[13] A. Kenny, H. Koesling, D. Delaney, S. McLoone, T. Ward, "A Preliminary Investigation into Eye Gaze Data in a First Person Shooter Game", 19th European Conference on Modelling and Simulation, ECMS 2005, Riga, Latvia, 2005.

[14] R.J. Jacob, "What you look at is what you get: eye movement-based interaction techniques", In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People, (Seattle, Washington, United States, April 01-05, 1990). J. C. Chew and J. Whiteside, Eds. CHI `90. ACM, New York, NY, USA, 11-18. DOI=http://doi.acm.org/10.1145/97243.97246, 1990.

[15] V. Tanriverdi, R.J.K. Jacob, "Interacting with Eye Movements in Virtual Environments", CHI´00, pp. 265-272, 2000.

[16] E. Jönsson, "If Looks Could Kill – An Evaluation of Eye Tracking in Computer Games", Masters Thesis in Human Computer Interaction. Department of Numerical Analysis and Computer Science. Royal Institute of Technology, Sweden, 2005.

[17] P. Isokoski, B. Martin, "Eye Tracker Input in First Person Shooter Games", In Proceedings of the 2nd Conference on Communication by Gaze Interaction, pages 76-79. Communication by Gaze Interaction (COGAIN), 2006.

[18] C. Sennersten, "Eye movements in an Action Game Tutorial", Masters Thesis, Department of Cognitive Science, Lund University, Sweden, 2004.

[19] O. Spakov, "EyeChess: the tutoring game with visual attentive interface", In G. Evreinov (ed.) In Proceedings of Alternative Access: Feelings & Games 2005, Department of Computer Sciences, University of Tampere, Finland, Report B-2005-2, pp. 81-86, 2005.

[20] M. Dorr, M. Böhme, T. Martinetz, E. Barth, "Gaze beats mouse: a case study", 3rd Conference on Communication by Gaze Interaction –COGAIN 2007: Gaze-based Creativity, Leicester, UK, 2007.

[21] D. Sasse, "A Framework for Psychophysiological Data Acquisition in Digital Games", Masters Thesis, Otto-Von-Guericke University, Magdeburg, Germany, 2008.

[22] J.D. Smith, T.C.N Graham, "Use of Eye Movements for Video Game Control", 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, Hollywood, California, USA, 2006.