LAIF: A Logging and Interaction Framework for Gaze-Based Interfaces in Virtual Entertainment Environments

Lennart Nacke
University of Saskatchewan, Canada
lennart.nacke@acm.org

Sophie Stellmach
Otto-von-Guericke University Magdeburg, Germany
stellmach@acm.org

Dennis Sasse
Microsoft Deutschland GmbH, Germany
dennis.sasse@microsoft.com

Jörg Niesenhaus
University of Duisburg-Essen, Germany
joerg.niesenhaus@uni-due.de

Raimund Dachselt
Otto-von-Guericke University Magdeburg, Germany
dachselt@acm.org

ABSTRACT

Eye tracking is a fascinating technology that is starting to be used for evaluation of and for interacting in virtual environments. Especially digital games can benefit from an integrated (i.e., evaluation and interaction) approach, harnessing eye tracking technology for analysis and interaction. Such benefits include faster development of innovative games which can be automatically evaluated in an iterative fashion. For this purpose, we present a framework that enables rapid game development and gameplay analysis within an experimental research environment. The framework presented here is extensible for different kinds of logging (e.g., psychophysiological and in-game behavioral data) and facilitates studies using eye-tracking technology in digital entertainment environments. An experimental study using gaze-only interaction in a digital game is also presented and highlights the framework's capacity to create and evaluate novel entertainment interfaces.

Keywords

digital games, eye tracking, interactive techniques, gameplay logging, software tool, XNA

INTRODUCTION

Eye tracking is a valuable technology to provide analytical insights for studying human behavior and visual attention [6]. Besides that it is an intuitive human-computer interface that especially enables users with disabilities to interact with a computer. The most common applications for eye tracking today are either in marketing (e.g., [18]) or in usability research (e.g., [23]). Yet, using eye trackers as devices for human computer interaction (HCI) has started to become a focus of research in recent years and the field is slowly starting to come of age [13] [2]. However, the use of eye tracking in digital games is still new [10], in the same way it is new for gaze interaction in virtual worlds [12] and for gaze visualizations in three-dimensional (3D) environments [30].

Gaze interaction through eye tracking is an interface technology that has great potential. While it is essentially a human-computer interface that can support[7] traditional input devices to improve efficiency, it can also be used to gather interaction data for post-usage evaluation. Interesting psychological data are for example the position and movement of gaze along the screen and pupil dilation. All eye tracking hardware allows to record patterns and distribution of fixations and saccadic eye motion with different levels of precision.

A problem of using eye tracking technology in game development is the lack of common frameworks that would simplify producing gaze games (e.g., as stimuli for psychological experiments). Usually, researchers have to fall back to developing custom software for each game and experiment. When eye tracking research was done for input and analysis of experiments, this was essentially the same problem. In general, there exist only few frameworks for developing small-scale games in an academic setting [19], let alone analysis tools within game engines that are able to support eye tracking technology (for a discussion see [29] and [24]). Our approach aims at filling this gap and addressing the above-mentioned problems. We contribute a framework, which was especially designed with the goal of encouraging rapid development and allowing easy access to a database with eye tracking data.

For rapid software development it is also essential to provide the possibility to reuse already existing code, which can be established by a modular approach using object-oriented paradigms. This way it is also possible to

[7] Eye trackers are limited in completely replacing mouse input. For example, if the mouse click is substituted with an eye blink, it may result in inaccuracies or – if done via dwell time measurement – in longer task times.

apply several modules for the integration of different input and output devices, such as eye trackers and other psychophysiological equipment.

We begin this paper with a review of related work in gaze interaction for games and gaze evaluation using game technologies. In addition, we discuss a few existing commercial logging solutions that can also be used to evaluate virtual environments. Next, we discuss the development and features of a logging and interaction framework (i.e., LAIF) implemented with the .NET-based XNA framework and the Torque X engine. Some main components of the framework are discussed in detail. The framework is then put to use in developing a gaze-only digital game that demonstrates its capabilities. For a better understanding of the evaluative questionnaire employed in the following user study, we also briefly touch on the concept of presence in games. The user study evaluated a digital game created for gaze-only input in terms of general spatial presence and gameplay experience and also compared it to mouse input. The study demonstrates the flexibility and extensibility of the framework. Finally, we close with a concluding discussion and possible future applications, suggesting that more studies can benefit from game creation using LAIF.

RELATED WORK

Our review of related work will focus on approaches that have successfully used eye tracking in combination with digital games, especially as an input mechanism. We will then also briefly review instrumentation tools that allow logging and evaluating data in digital games.

Gaze Interaction and Eye Tracking for Games

Jönsson [14] evaluated eye tracking for digital games and tested gaze input (vs. mouse input) for two commercial 3D games. For the first game, aiming at enemies was done using the eyes and firing of a bullet via clicking the mouse. In addition, gaze steering was evaluated with a second game, one time just for aiming, one time for changing the view and one time for doing both together. The following gaze interaction characteristics for games were created following qualitative observations in that study: (1) control was subjectively better, (2) game experience was more fun and committing, and (3) eye control felt natural, easy, and fast (see also [1][13]).

Sennersten et al. [24] conducted a verification study for the integration of an eye tracker with a 3D game engine. The implementation was done in proprietary software (i.e., HiFi engine from Swedish Defence Research Agency FOI), but the game engine used the *Battlefield 1942* file format, so that experimental stimuli could be developed using the level editor Battlecraft. Therefore, her work primarily demonstrates using game design tools for prototyping of digital games that can be used in psychological experiments. However, this approach falls short of implementing a gaze interaction modality for the evaluated game engine and provides no empirical support for the efficiency of the implemented system.

Istance et al. [12] investigated the use of eye tracking input for special use cases within the Massively Multiplayer Online Game *World of Warcraft* and the virtual environment *Second Life*. By using both bottom-up and top-down approaches specific tasks within the role-playing game were selected and implemented. It was then evaluated how well they can be carried out using eye tracking as the only input device to solve locomotion, fighting, equipment exchange and communication tasks. Compared to the standard keyboard and mouse controls the task completion times of the gaze input are very similar. However, a potential in optimizing gaze input is discussed by solving some distraction errors during the locomotion tasks which lead to path deviations.

Smith et al. [26] presented a study using eye-based input for game modifications of *Quake 2*, *Neverwinter Nights*, and *Lunar Command*. For the first game, eye tracking was used to control player orientation, but movement and firing was still done using keyboard and mouse input. In *Neverwinter Nights*, gaze replaced mouse movement for locomotion, however, with confirmatory clicks to control pointing. In *Lunar Command*, the only 2D game, players similarly replaced mouse movement with gaze and mouse clicks. In this game, players performed significantly better with the mouse, whereas no significant difference was found for the first two 3D games. The subjective results of Smith's study showed participants felt more immersed when using the eye tracker for input.

In another study, Kenny et al. [15] created a first-person shooter (FPS) game and used a 3D engine to log eye tracking data, video data and game internal data that was correlated with each other. The gaze data were used for fixation analysis of their game and a result was that players fixate the center of the screen for a majority of the time.

Wilcox et al. [33] created a third-person adventure puzzle game with gaze input and voice recognition for disabled kids who are not able to control the game with the standard combination of mouse and keyboard inputs. The gaze input works by focusing a game object and selecting or activating it via voice command or blinking. In order to solve the problem of users looking at new targets while giving the voice commands a time lag for selecting items was implemented.

In one of the studies of Nacke et al. [22] a game modification level of *Half-Life 2* was used to test the navigation of users in a 3D environment. The mouse input for the camera view control was substituted with a gaze input and combined with the common keyboard controls for character movement. The navigational challenges consisted of a labyrinthine structure of a catwalk with obstacles placed in between. The results of a questionnaire indicated a very positive gaming experience, where the challenge of controlling the game by gaze (supported by

keyboard) input results in positive affection and feelings of flow and immersion.

Isokoski et al. describe the advantage of gaze pointing in FPS games as making alignment of the player camera to the target become obsolete, when aiming is decoupled from player view [9]. Their results show gaze input for FPS games can compete with "killing efficiency" of gamepad input, but leads to problems with targeting accuracy.

A preliminary, short investigation was also conducted by Isokoski and Martin [11], where they examined efficiency of eye trackers compared to game controllers. They also designed a game with focus on moving and aiming, where gaze was used for aiming at moving targets. Again, shooting was performed here with mouse button clicks.

Other gaze-only interaction games include the development of a gaze-only 2D eye chess game [27] and a game that used dwell times and pupil dilation to create an innovative digital game mechanic [7]. Moreover, Isokoski et al. [10] give an extensive overview of research focused on games supported by eye tracking input devices and discuss the implications and possible future developments of gaze input for different game genres. They present a taxonomy based on *user input requirements*, *technical limitations* and *gaming contexts* to classify computer games into groups offering different opportunities for eye tracking technology. For example, they argue that the current generation of eye tracking hardware is not capable of competing with the high accuracy of the state-of-the-art gaming mice needed for fast-paced FPS games that have high demands on precise aiming.

The literature on gaze interaction games shows that there are a few approaches using eye tracking technology to support and extend the interaction possibilities with digital games. However, no previous approach has integrated logging (using eye tracking hardware) and interaction functionality, because the application of logging instrumentation to gameplay evaluation is a rather new approach [28][20]. Next, we will discuss existing logging instrumentations that could be used for evaluation of game interaction data.

Analysis Tools

The automatic logging of events to better understand user behavior within an interactive system is tied to the history of research in psychology and usability. Traditional automated logging solutions first kept track of animal interactions to analyze the behavior of for example rats in a maze [25]. By analyzing the response rate logs, the theory of reinforcement schedules was incepted. Automated logging of human behavior is still common today. Game metrics (e.g., time to complete a task, accuracy of input, user satisfaction) along with survey and observation measurements are common approaches to the analysis of gameplay behavior [3][20][16].

For example, the TRUE system presented by Kim et al. [16] combines the advantages from different research approaches, such as the collection of behavioral evaluation data, qualitative survey data, and other tracking data. Such event-related data sets can also be compared to video recordings to provide contextual information.

Commonly, game instrumentation or metrics data [4] are collected into spreadsheets or databases, which can contain various amounts of interesting behavioral player data. Spreadsheet applications such as *Excel*, *Spotfire*, and *Tableau* allow the fast visualization of massive data sets to quickly explore meaning and relations in the data. Here, users can choose from visualizations, including pie or bar charts, and scattergrams. Spotfire and Tableau impress with well-designed user interfaces and graphics, for example, simple data integration via drag-and-drop. In addition, displayed data elements can be selected and filtered dynamically.

Noldus and Mangold provide software suites for the acquisition, analysis, and presentation of video, audio, and sensor data (including gaze and psychophysiological data) from behavioral studies. Multiple video views and the functionality to assign event markers (linking back to the video) are implemented. Behavioral and physiological data can be visualized in (static) plots. What these data analysis suites are lacking is an integrated functionality that allows the design of a behavioral stimulus, such as a game so that during the design of a game, the events of interest can already be defined in the tool, so that subsequent analysis is almost completely automated. One such solution was discussed in [21] and it is a common approach in the game industry to automate user testing with behavioral data [5]. However, not much emphasis is currently given to automated collection of sensor data, such as eye tracking or psychophysiological data. Out framework attempts to fill this gap by providing a flexible solution for sensor data analysis, especially gaze data.

Various special tools for the analysis of gaze data exist. Such tools are often custom-designed for particular hardware devices (e.g., *Tobii Studio*). In general, these tools support texts, still images, animations, software, videos, and web content as psychological stimuli. The tools typically handle gaze data synchronization with video recordings. This is especially important for the visual analysis of dynamic stimuli [29]. One example for an open source freeware deploying slide shows as stimuli is the *Open Gaze And Mouse Analyzer* (OGAMA) [32]. None of the gaze analysis tools are explicitly focused on the integration with digital games or virtual environments. Nevertheless, as our review of related literature has shown, there is a huge research interest in developing gaze interaction games and the development of a framework combining gaze interaction with evaluative logging functionality is worthy research effort. We will proceed to discuss how we created this integrated logging and interaction framework: LAIF.

FROM CONCEPT TO DEVELOPMENT

The idea behind LAIF was to create a tool equally usable for researchers from psychology and computer science (who commonly work together in human-computer interaction). This system should support rapid prototyping of game levels that could be used as digital game stimuli in psychological experiments or as demonstrators of new game interaction technology, ideally combining the best of both worlds.

During the conception phase of the framework many game development technologies were evaluated. We broke down game development into two distinct parts: (1) developing core gameplay mechanics with programming code and (2) creating game content usually with an editing tool (and saved as map or level files).

For users without technical knowledge functionality should be accessible through an editor, while coders should also have the possibility of using a high-level programming language. After interviewing game designers and psychologists with only little experience in game development, we formed the following criteria for usability of a level editing and scripting tool:

- Browse, preview, and place existing content in a game
- Make adding game logic easy (e.g. using triggers)
- Graphic integration and access to gaze logging
- Code completion with extensive suggestions
- Several code samples and templates

We looked at graphical game editors and development tools used commonly in research settings as well as reviews of these different tools regarding their suitability for game development (e.g., [19]). Tools such as *GameMaker*, *Flash* and *Torque* X were all evaluated with the computer science and psychology researchers of an HCI laboratory. Torque X was chosen in the end, because it was judged to have the highest future potential in terms of code support and continuing development. The establishment of the XNA[8] Creator's Club and other Microsoft initiatives were all factors contributing to the success of Torque X and influenced our initial choice. The improved continuous support of XNA and Torque X to the day of this writing supports and validates our initial choice.

The Torque X game engine is largely component based, using an aggregation model. Instead of putting common functionality in a base class from which it is then inherited, game objects share common components. This component system is also tightly integrated in the Torque X Builder application, which can dynamically create editors for all properties of custom engine components.

Requirement Analysis

We decided to create LAIF as a set of modular Torque X components. These components are objects derived from the `TorqueComponent` class adding functionality to a game object. The target users of the tool were researchers with little game design and programming experience. Thus, they need to be able to configure logging settings for an experimental game stimulus through a graphical user interface (in the Torque X Builder). Our design objectives here were in detail:

- Creating new or existing log files in text format and direct export of logged gaze data into a database
- Saving configurations and settings in an XML file
- Integration of logging in a graphical editor tool, which also provides drag-and-drop functionality

Preliminary Target Hardware

The hardware the framework was developed for and tested with consisted of a *Tobii 1750* eye tracker. It features an integrated camera in a 17" TFT monitor and tracks the eyes with two infrared diodes, which produce reflection patterns on the eyes' corneas. These patterns make it possible to extract the pupil locations and dilations through digital real-time image processing. It has to be kept in mind that this hardware was only chosen due to its availability, but while our framework makes use of this hardware it can also be extended to include other types of eye tracking (or general sensor recording) hardware.

FRAMEWORK IMPLEMENTATION

The above-mentioned eye tracker ships with the Tobii Eye Tracker Components API, which is a type library implemented as a set of COM objects, allowing access to the software abstraction layer provided by Tobii. It can be accessed by some high-level programming languages for Microsoft platforms, such as C#. The API unit itself is dependent on a few driver libraries and the .NET framework. The eye tracker may remain on any host as long as the API component is installed on that host. There must be TCP/IP and UDP/IP connectivity between an application and the host running eye tracker server.

Since we had extensive experience in using and writing queries for *MySQL* databases, we chose MySQL over the Microsoft solution *SQL Server 2005 Express*. In addition, the SQL Server Express is limited to 4 GB in database size, whereas MySQL is only limited by the capabilities of hardware. Since gaze logging can quickly amass huge amounts of data over longer periods of time, we were convinced that MySQL was the best solution for our purposes. To complete data connectivity to MySQL databases, the *MySQL Connector/NET ADO.NET* driver was used.

[8] A high-level programming environment specifically targeted at the hobby and academic development market.

The logging framework

The logging framework is designed as a set of Torque X components, derived from the default Torque Component class[9].

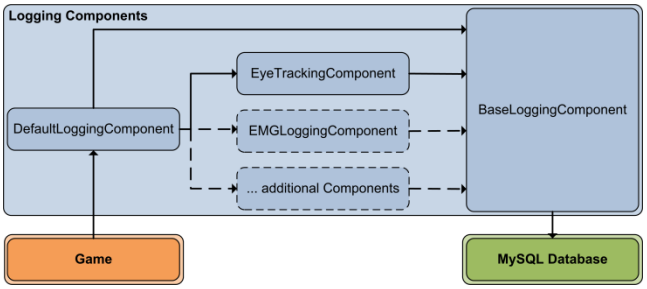

Figure 1. Component Schema of the Logging Framework

Each component provides individual key functionality, such as writing to a log file, performing queries to a MySQL database, or accessing a piece of data acquisition hardware. We did this to keep the logging framework extensible and reusable for other applications and apparel that we might use in the future on top of gaze logging (for example psychophysiological data logging).

The key components of the framework are:

- The *Basic Logging Component* (BLC) handles all access to the log files, as well as to the MySQL database.
- The *Eye Tracking Component* (ETC) is responsible for all access to the eye tracker and the functions provided by the TET Components API.
- The *Default Logging Component* (DLC) is a basic template component that provides a unified interface to the BLC and to the ETC. Following the concept of aggregated functionality suggested by the Torque X Engine, a game using the logging framework should never need to access the other components directly.

One of the requirements for the logging framework was to allow easy reconfiguration and automated management of many experiments. Therefore, all settings and configurations of the logging components are stored in XML configuration files. The only property that has to be set in the editor is the name of the XML configuration file in which all other properties are stored.

Basic Logging Component (BLC)

The BLC is responsible for creating and writing to a predefined log file in .txt or .csv format. It also loads all necessary settings are from an XML file. Properties for XML file and log file need to be set. The log file defined serves as fallback log and is initialized together with the component. Therefore, it also stores basic error and exception messages, even if loading the configuration settings from the XML file failed.

[9] The framework code can be requested from dennis.sasse@gmail.com

Once initialized, BLC provides a method to write new lines of data to the log file. It also provides methods for read and write access to a MySQL database, whose database address, name, access identification and password are read from the XML configuration file. If the connection to the MySQL database defined in the configuration file cannot be established, data will be written to a log file instead. The BLC is required by all other components and will automatically be added by the Torque X Builder to any object to which other logging components are added.

Eye Tracking Component (ETC)

The ETC provides access to the eye tracker. It reads the IP address of the required server from an XML defined in an editor property. The XML configuration file has a flag that can switch whether eye tracking is enabled, in which case the ETC tries to connect automatically to the eye tracking server on initialization.

Once the connection is established, an event listener is registered that handles all data coming from the eye tracker automatically. The ETC provides methods to access gaze positions of the left and right eye separately, as well as the median position of both eyes.

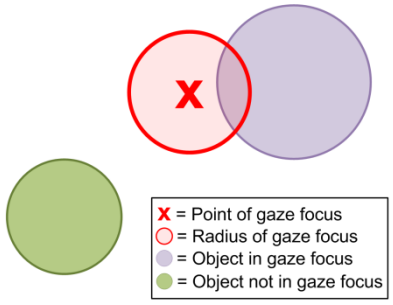

Figure 2. Distance-based detection of objects in gaze focus

A method returning all objects in a set radius around the player's point of gaze focus is also provided (objects in gaze focus are detected based on their distance to the point of gaze focus; see Figure 2). The radius of gaze focus is defined by a property in the XML configuration file. The Torque X Engine allows organizing game objects on different layers, therefore detecting objects in the player's gaze focus can be limited to certain layers, if required. Since all coordinates returned are given in screen positions, more methods are provided to transform screen coordinates into world coordinates and to transform world coordinates back into screen coordinates.

The logging framework was developed for a version of Torque X, which could only display and handle 2D graphics and game worlds. Therefore, the trivial distance-based detection of objects in the player's gaze focus was sufficient for development (cf. Figure 2). During the writing of this article, the most recent Torque X version also features a 3D world editor and full 3D rendering functionality, so that extending the logging framework for 3D games and integrating it into a graphical 3D game

editor is a promising future development. How the integrated components look in the Torque X Builder (i.e., the level editor) is shown in Figure 3.

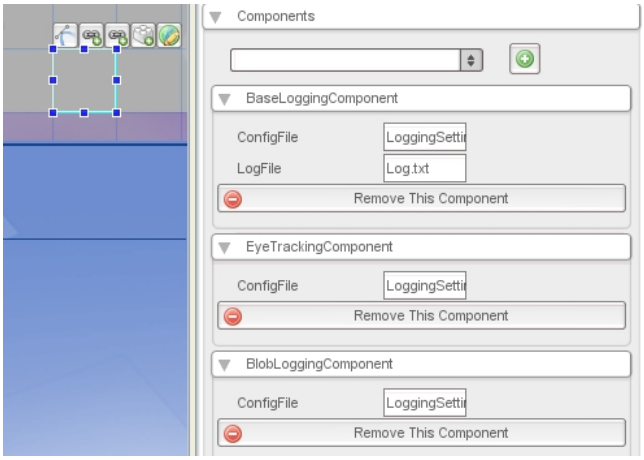

Figure 3. Screenshot from the Torque X Builder application, depicting the integrated logging components and their properties.

Default Logging Component (DLC)

DLC is a basic template component, using BLC and ETC. It shows how components should be aggregated and accessed: Instead of accessing every component from the game, it is advised to use a custom component based on DLC as a wrapper for the functionality of the other components. This also simplifies access to the required logging functionality, because only the specialized implementation of DLC must be included into the game, which then automatically loads and initializes all following required components.

The logging framework only provides methods to track a player's gaze positions and to write data to a MySQL database or to a local file. It does not dictate what data should be logged or how the log files or entries in the database should be organized. It is up to the researcher to decide about these details, because they generally vary from experiment to experiment, depending on the hypothesis or game created. In addition, the logging framework is not just limited to data acquisition. ETC can also be used to integrate gaze-based interaction into a digital game, as we will show in the next section.

It is advised to implement all changes or additions to existing components in a new component derived from DLC to fully embrace the concept of aggregating streamlined components. This feature also allows the easy extension of the logging framework to include custom components.

An Example Game Using Eye Gaze Logging

To show the flexibility of this framework and to evaluate its workflow, a digital game was designed for an experiment studying gaze interaction, which we called *Blob* (see Figure 4)[10].

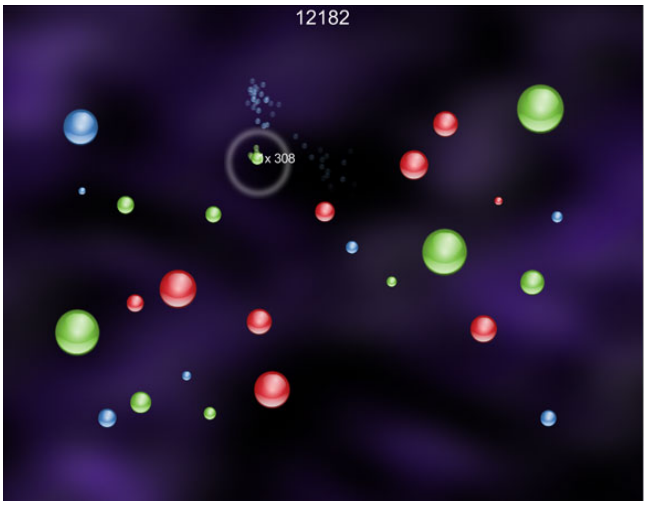

Figure 4. Screenshot from the Blob example game, showing Blobs of different size and color as well as a high score on top of the screen.

In Blob players have to pop as many colored bubbles (or *Blobs*, thus the name) as possible by touching them with their own smaller Blob, which we wanted to be controllable by mouse or by gaze. New Blobs spawn in one of three colors (red, green, or blue) at random positions on the screen and gradually grow in size. If two or more Blobs collide with one another, the game is over.

The larger a Blob is when collected, the higher the score rewarded for it. Players are also rewarded with a score multiplier for consecutively collecting multiple Blobs of the same color. The more Blobs of the same color are collected in a row, the higher the multiplier is. Final goal of the game is to achieve the highest score, which is kept in a highscore database (using database access in BLC).

[10] A mouse version of the Blob game can be downloaded from www.dennissasse.com

Entry Label	Value	Example Data
`SessionID`	Internal `SessionID` of the particular play session	`bg_03`
`GameTime`	Time at which the particular event happened in millisceconds after the beginning of the game.	`00:03:49.2`
`Event`	Code to identify a log event, such as `PlayerCollectBlob` if the player collected a blob or `BlobsCollide` if two blobs collided.	`BlobsCollide`
`Score`	Current score of the player at event.	`133753`
`ComboMultiplier`	Current combo multiplier at event.	`5`
`BlobsCollected`	Number of blobs the player has collected at event.	`42`
`PlayerBlobColor`	Current color of the player's blob at event.	`Red`
`BlobColor`	Color of the blob the player collected or the blob that is colliding. *Note: This value depends on the event. If two blobs were colliding, there are two entries: One for each of the colliding blobs.*	`Green`
`PlayerFocusObject`	`ObjectID` of the object the player is looking at. *Note: This entry is only written if the player looks at an object. Object(s) in the player's gaze focus are detected by a region-based approach (see Figure 2). If there is more than one object in gaze focus, this event will be written to the database for each object in gaze focus.*	`BlobGreen_64`

Table 1. Overview of logged data written to the database in the example game Blob.

We created the Blob demonstrator game in five simple steps:

1. Creating a new XNA game project in Microsoft Game Studio Express based on one of the Torque X starter kits (provides the project structure, the basic game loop, components for collision detection and basic player input)
2. Programming the game logic and components (all components, whether the AI that controls Blob generation, the control of the player Blob, or the logging components had to be programmed)
3. Building the game world which can be created with the graphical user interface in the Torque X Builder (including materials, animations, and particle systems). Components with game logic scripts were also attached to game objects.
4. Integrating logging functionality. BLC is required by any other logging component and thus automatically added by the Torque X Builder to any object a logging component is attached to. Property panels for each logging component are then automatically created.
5. Building and deployment of the game was done by compiling the Torque X Builder files in Microsoft Game Studio Express.

The Blob Logging Component (BlobLC)

As suggested before, the core components of the logging framework should not be edited or accessed directly from the game. Instead, a new component should be derived from DLC and edited to contain all logging functionality that may be required. The number of components is not limited; therefore it is possible to create as many new components as needed for a particular game.

In the case of Blob, BlobLC initializes both BLC and ETC. It provides methods to identify players currently playing by their identification number (ID), to initialize a new play session and to write logging events to the database. It also handles storage and display of the highscore list, once the game is over.

Similar to all other logging components, BlobLC needs an XML configuration file, from which all settings are read on initialization. In addition to the ID required to identify a player, these settings contain a property to enable or disable logging.

The logging framework itself does not regulate what data is written to the database or how the database is organized. To provide maximum flexibility, type, and format of the logged data has to be defined when developing the particular game. Table 1 gives an outline of what data is logged by BlobLC and how log events are organized in the database.

A STUDY OF THE GAZE-BASED GAME INTERFACE

After the gaze-only game was developed, we conducted an evaluative user study with several people testing the gaze-only interaction.

Design

In a first test, we used two evaluative questionnaires of gameplay experience [8] and spatial presence [31] that people who played the gaze-only game had to answer. In a second test, we developed another version of the game that uses mouse input and we asked a few comparative questions concerning the differences in this interaction. For a better understanding of the evaluative questionnaires in the first test, we will briefly explain the concept of presence and gameplay experience in games.

Presence and Gameplay Experience Concepts

Spatial presence is a two-dimensional construct in which the core dimension is the sensation of physical location in a virtual spatial environment and the second dimension entails the perceived action possibilities (i.e., individuals only perceive possible actions relevant to the virtual mediated space) [34]. Regarding gameplay experience in general, IJsselsteijn, Poels, and de Kort theorized that *immersion*, *tension*, *competence*, *flow*, *negative affect*, *positive affect*, and *challenge* are all important elements of gameplay experience and developed a game experience questionnaire (GEQ) to assess these elements [8].

Participants

For our study, 25 male higher education students participated, aged between 19 and 38 years (M = 24 years, SD = 5). Most of the participants were right-handed (88% right-handed) and a little more than half of them were not wearing glasses or contact lenses (56%). All the participants owned a personal computer (PC) and 96% also rated this as their preferred gaming platform.

Game Material

The digital game blob was created as described in the section before. For the second evaluative test in the study, we developed a version of Blob that can be played with mouse input. Blob itself was consciously designed so it would allow for both types of interaction mechanics, classic mouse input, and gaze input from the *Tobii 1750* eye tracker.

Measures

We used the game experience questionnaire (GEQ) [8] as well as the short scale spatial presence questionnaire (MEC SPQ) [31] to get an overview of gameplay experience with the gaze-only game. The GEQ combines several experiential measures. It was developed based on investigations of frequent game players. It has seven dimensions: flow, challenge, competence, tension, negative affect, positive affect and sensory and imaginative immersion. Each dimension is measured, each using 5 items in the full version. Each item consists of a statement on a five-point scale ranging from 0 (not agreeing with the statement) to 4 (completely agreeing with the statement). The MEC SPQ scales *self-location* and *possible action* are constructed in a similar fashion (except: scale from 1 to 5).

Six preference questions were used in the second test, when comparing gaze with mouse input (Q1: *Which interaction mode did you enjoy playing with more?* Q2: *Which interaction mode was easier to learn?* Q3: *Which interaction mode was easier to use?* Q4: *With which interaction mode did you feel more immersion in the game?* Q5: *For which interaction mode did the controls feel more natural?* Q6: *Which interaction mode would you prefer to use in the future?*). Participants had to answer with either gaze or mouse depending on their preference.

Procedure

Figure 5. A person playing the Blob game using only eye tracker input.

In the first test, participants only played the gaze version of the game and answered the GEQ and SPQ. Later, participants returned to play Blob with once with mouse and one time using gaze interaction (see Figure 5). The sequence in which the game interaction modes were used in the second test was randomized. Each interaction mode was played for 5 minutes and then switched. All participants had to fill out an evaluative questionnaire and a presence questionnaire for the gaze interaction part [31], where they had to rate their feeling of presence (SPQ) on a scale from 1 to 5 (GEQ was on a scale from 0 to 4).

Results

The results from the GEQ are shown in Figure 6. General appraisal of gaze-only interaction was positive.

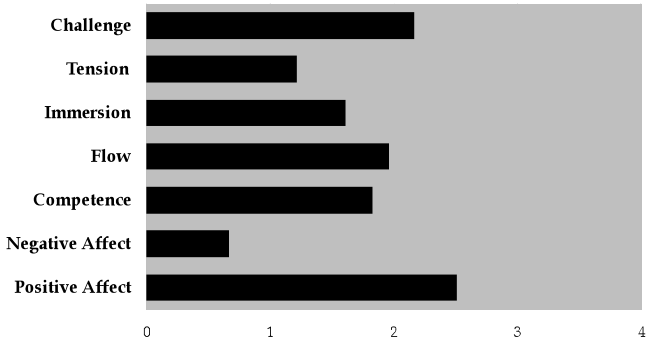

Figure 6. GEQ results for gaze-only interaction.

The low ratings for negative affect and tension, together with higher ratings of flow, challenge, and positive affect indicate a more positive than negative gameplay experience in line with previous research findings [22]. However, it needs to be noted that–except for positive affect and challenge–the result are below the median of the measurement scale (2), indicating a generally not very expressive or extreme response to the game stimulus. The reliability of the GEQ was acceptable in this test (Cronbach's $\alpha = .72$).

For the component *spatial presence self location* reliability was very high (Cronbach's $\alpha = .95$). The mean for items was below the median value ($M = 2.1$, $SD = 1.1$). The other component used was *spatial presence possible actions*, which was also reliable (Cronbach's $\alpha = .88$). Its item mean value was higher but still just below the median value ($M = 2.9$, $SD = 1.4$). Thus, gaze only interaction did not seem to especially facilitate feelings of presence in the game.

For the comparative part of the study, when asked which interaction mode they preferred, 17 people answered that they preferred steering the game with their eyes. However, only 3 found this interaction mode easy to learn and 21 found the mouse interaction easier to use for this game. Additionally, 13 people found the mouse control felt more natural, but 18 people would prefer to use gaze interaction in the future if they had more time to train for it. The reliability of the questionnaire for comparison between mouse and gaze was acceptable (Cronbach's $\alpha = .76$).

Observations and Brief Discussion

A major complaint of participants was the immediate responsiveness of the system when under eye tracking control. This may cause discomfort due to the natural 600ms delay between a person making a decision (pointed out by eye movement) and the execution of a task [17]. In general, we have to note the positive gameplay experience indicated by the GEQ results when playing the gaze-only game. This supports that players highly appreciate novel forms of game interaction although they might be challenging to learn.

Our study shows that the logging framework cannot only be used to implement gaze input and logging, but also that users appreciated a gaze-only game, which was created with our framework and they even preferred the novel experience of steering the Blob with their eyes (although their feeling of presence was rather low). However, the accuracy of the eye logging data needs to be tested in a larger scale verification study (for an example see [28]).

CONCLUSIONS AND FUTURE WORK

Eye tracking technology has great potential for studying player behavior on the one hand and to be used as an input device for games on the other hand. The presented work needs to be understood as a first effort in the direction of integrated logging and interaction solutions for digital games. A more comprehensive framework with support for more components, including various novel input and output devices shall be created in the future. First approaches into visual analysis within a game engine, also using XNA technology have already been undertaken [29] and promise interesting application scenarios.

With the framework presented in this paper, we can facilitate experiments using eye tracking for analysis of and interaction with digital games. Our study also showed how easy a game with this functionality can be implemented and how gaze steering is appreciated as a novel input method for digital games.

The logging framework will be extended to 3D games in the future since we are already investigating integration with *Half-Life 2*. The integration into a graphical 3D editor integrated with Torque X (also visualizing gaze data in virtual environments) will be a next step [30].

We can conclude that integrating logging into games in general is a valuable method to analyze game design. Using these data to evaluate novel player-game interaction holds great research potential for the future. Thus, we have only begun to see the tip of the iceberg for research in the area of game analysis by logging and of gaze interaction.

ACKNOWLEDGMENTS

The development of this framework was partially supported by the European Community FP6 FUGA research project (NEST-PATH-028765), as well as the BMBF (Bundesministerium für Bildung und Forschung) funded ViERforES project. We thank our FUGA colleagues at BTH, especially Craig A. Lindley for great support and stimulating discussions. In addition, we would like to extend our heartfelt thanks to the volunteer participants in the user study.

REFERENCES

1. Castellina, E. and Corno, F. Multimodal Gaze Interaction in 3D Virtual Environments. *Proceedings of COGAIN 2008: Communication, Environment and Mobility Control by Gaze*, COGAIN (2008), 33-37.
2. Cournia, N., Smith, J.D., and Duchowski, A.T. Gaze- vs. hand-based pointing in virtual environments. *CHI '03 extended abstracts*, ACM (2003), 772-773.
3. Drachen, A. and Canossa, A. Analyzing Spatial user Behavior in Computer Games using Geographic Information Systems. *Proc. MindTrek*, ACM (2009).
4. Drachen, A. and Canossa, A. Towards gameplay analysis via gameplay metrics. *Proc. MindTrek*, ACM (2009), 202-209.
5. Drachen, A., Canossa, A., and Yannakakis, G.N. Player Modeling using Self-Organization in Tomb Raider: Underworld. *Proc. of CIG2009*, IEEE (2009).
6. Duchowski, A.T. *Eye tracking methodology: Theory and practice*. Springer, New York, 2007.
7. Ekman, I.M., Poikola, A.W., and Mäkäräinen, M.K. Invisible eni: using gaze and pupil size to control a game. *CHI '08 extended abstracts*, ACM (2008), 3135-3140.

8. IJsselsteijn, W., Poels, K., and de Kort, Y. *The Game Experience Questionnaire: Development of a self-report measure to assess player experiences of digital games.* Report. TU Eindhoven, Eindhoven, 2008.
9. Isokoski, P., Hyrskykari, A., Kotkaluoto, S., and Martin, B. Gamepad and eye tracker input in first person shooter games: Data for the first 50 minutes. *Proceedings of* COGAIN (2007), 11-15.
10. Isokoski, P., Joos, M., Spakov, O., and Martin, B. Gaze controlled games. *Universal Access in the Information Society 8*, 4 (2009), 323-337.
11. Isokoski, P. and Martin, B. Eye Tracker Input in First Person Shooter Games. *Proceedings of the 2nd Conference on Communication by Gaze Interaction: Communication by Gaze Interaction - COGAIN 2006*, (2006), 78-81.
12. Istance, H., Vickers, S., and Hyrskykari, A. Gaze-based interaction with massively multiplayer on-line games. *CHI09 extended abstracts*, ACM (2009), 4381-4386.
13. Jacob, R.J.K. What you look at is what you get: eye movement-based interaction techniques. *Proc. of CHI 1990*, ACM (1990), 11-18.
14. Jönsson, E. If Looks Could Kill: An Evaluation of Eye Tracking in Computer Games. 2005. Master's Thesis. Royal Institute of Technology. Stockholm, Sweden.
15. Kenny, A., Koesling, H., Delenay, D., McLoone, S., and Ward, T. A preliminary investigation into eye gaze data in a first person shooter game. *Proceedings of the 19th European Conference on Modelling and Simulation*, (2005).
16. Kim, J.H., Gunn, D.V., Schuh, E., Phillips, B., Pagulayan, R.J., and Wixon, D. Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems. *Proc. of CHI 2008*, ACM (2008), 443-452.
17. Koesling, H. and Höner, O. Bend it like Beckham - Mindsets and Visual Attention in Decision-Making in Soccer. *Proc. of ECEM 12*, (2003).
18. Maughan, L., Gutnikov, S., and Stevens, R. Like more, look more. Look more, like more: The evidence from eye-tracking. *The Journal of Brand Management 14*, (2007), 335-342.
19. Nacke, L. Facilitating the Education of Game Development. 2005. Diplomarbeit. Otto-von-Guericke University. Magdeburg, Germany.
20. Nacke, L. Affective Ludology: Scientific Measurement of User Experience in Interactive Entertainment. 2009. Ph.D. Thesis. Blekinge Institute of Technology. Karlskrona, Sweden.
21. Nacke, L., Lindley, C., and Stellmach, S. Log Who's Playing: Psychophysiological Game Analysis Made Easy through Event Logging. *Proc. of Fun and Games 08*, Springer (2008), 150-157.
22. Nacke, L., Stellmach, S., Sasse, D., and Lindley, C.A. Gameplay experience in a gaze interaction game. *Proceedings of COGAIN 2009: Gaze Interaction For Those Who Want It Most*, The COGAIN Association (2009), 49-54.
23. Schießl, M., Duda, S., Thölke, A., and Fischer, R. Eye tracking and its application in usability and media research. *MMI-interaktiv Journal 6*, Sonderheft: Blickbewegung (2003).
24. Sennersten, C., Alfredson, J., Castor, M., et al. *Verification of an experimental platform integrating a Tobii eyetracking system with the HiFi game engine*. Research Report. FOI, Linköping, 2007.
25. Skinner, B.F. *The Behavior of Organisms: An Experimental Analysis*. D. Appleton-Century Company, 1938.
26. Smith, J.D. and Graham, T.C.N. Use of eye movements for video game control. *Proc. of ACE 2006*, ACM (2006), 20.
27. Špakov, O. EyeChess: the tutoring game with visual attentive interface. *Alternative Access: Feelings & Games*, University of Tampere (2005), 81-86.
28. Stellmach, S. A psychophysiological logging system for a digital game modification. 2007. Research Report. Otto-von-Guericke University. Magdeburg, Germany.
29. Stellmach, S. Visual Analysis of Gaze Data in Virtual Environments. 2009. Diplomarbeit. Otto-von-Guericke University. Magdeburg, Germany.
30. Stellmach, S., Nacke, L., and Dachselt, R. Advanced Gaze Visualizations for Three-dimensional Virtual Environments. *Proc. Eye Tracking Research & Applications (ETRA 2010)*, ACM (2010).
31. Vorderer, P., Wirth, W., Gouveia, F.R., et al. *MEC Spatial Presence Questionnaire (MECSPQ): Short Documentation and Instructions for Application*. 2004. Report to the European Community, Project Presence: MEC (IST-2001-37661).
32. Vosskühler, A., Nordmeier, V., Kuchinke, L., and Jacobs, A.M. OGAMA (Open Gaze and Mouse Analyzer): open-source software designed to analyze eye and mouse movements in slideshow study designs. *Behavior Research Methods 40*, 4 (2008), 1150-1162.
33. Wilcox, T., Evans, M., Pearce, C., Pollard, N., and Sundstedt, V. Gaze and voice based game interaction: the revenge of the killer penguins. *ACM SIGGRAPH 2008 posters*, ACM (2008), 1-1.
34. Wirth, W., Hartmann, T., Böcking, S., et al. A Process Model of the Formation of Spatial Presence Experiences. *Media Psychology 9*, 3 (2007), 493-493.