

DIPLOMARBEIT / DIPLOMA THESIS

VISUAL ANALYSIS OF GAZE DATA  
IN VIRTUAL ENVIRONMENTS

SOPHIE STELLMACH



Department of Simulation and Graphics  
Faculty of Computer Science  
Otto-von-Guericke University Magdeburg

April, 2009

Sophie Stellmach: *Visual Analysis of Gaze Data in Virtual Environments* Diploma Thesis © April, 2009

COURSE OF STUDIES:  
Computational Visualistics

MATRICULATION NUMBER:  
169351

EMAIL:  
SStellmach@gmail.com

SUPERVISORS:  
Prof. Raimund Dachsel  
Prof. Craig Lindley

PROCESSING PERIOD:  
November 15, 2008 - April 15, 2009

LOCATION:  
Magdeburg, Germany / Karlshamn, Sweden

SOPHIE STELLMACH

VISUAL ANALYSIS OF GAZE DATA IN  
VIRTUAL ENVIRONMENTS

Why are things as they are and not otherwise?

— Johannes Kepler

## ABSTRACT

---

Eye tracking technology allows the recording of eye movements for post-hoc analysis of visual attention. One of its major strengths and a reason for its popularity is presenting easily comprehensible, superimposed visualizations for two-dimensional stimuli - for example heat maps and gaze plots. However, such gaze visualization techniques do not adequately support interactive analysis of three-dimensional (3D) virtual environments (VEs).

This work intends to facilitate eye tracking studies in 3D VEs by enhancing visual gaze analysis techniques and methodologies. For this purpose, a survey was conducted for investigating improvements of existing gaze analysis techniques. This led to the development of a taxonomy of gaze visualizations to determine strengths and weaknesses of current techniques. Based on feature requests from the survey and findings from the classification of gaze visualizations, a gaze analysis tool was developed that integrates visualizations for static 3D VEs. The implementation is based on the *Tobii 1750 Eye Tracker*, the *XNA Framework*, and *Windows Forms*.

The impact of this thesis is twofold: Firstly it helps eye tracking mature into a technology that can be used to easily investigate visual attention in simulation and, for example, serious gaming environments. For this purpose, a software solution is provided to generate and explore gaze visualizations in VEs. Secondly it serves as a foundation for future research into 3D visualizations of human gaze by providing the methodological and taxonomical framing for these visualization techniques.

## ZUSAMMENFASSUNG

---

Eye Tracking Technologie erlaubt die Aufnahme von Blickbewegungen zur Analyse visueller Aufmerksamkeit. Eine ihrer größten Stärken und damit ein Grund für ihre Beliebtheit ist die Darstellung einfach verständlicher, übereinander gelagerter Visualisierungen für zweidimensionale Reize - zum Beispiel Hitzekarten (heat maps) und Blickpunktdiagramme (gaze plots). Allerdings unterstützen vorherrschende Blickvisualisierungstechniken die interaktive Analyse von dreidimensionalen (3D) virtuellen Umgebungen (VEs) nicht angemessen.

Die vorliegende Arbeit beabsichtigt Eye Tracking Studien in 3D VEs zu erleichtern, indem visuelle Blickanalysetechniken und -methoden erweitert werden. Zu diesem Zweck wurde eine Umfrage mit Eye Tracking Experten und Wissenschaftlern durchgeführt, um Verbesserungsmöglichkeiten existierender Blickanalysetechniken zu eruieren. Dies führte zur Entwicklung einer Taxonomie von Blickvisualisierungen, um Stärken und Schwächen aktueller Techniken zu bestimmen. Basierend auf den Verbesserungsvorschlägen aus der genannten Umfrage und Schlussfolgerungen aus der Klassifikation von Blickvisualisierungen wurde ein Blickanalysewerkzeug entwickelt, das Visualisierungen für statische 3D VEs integriert. Die Implementierung basiert auf dem *Tobii 1750 Eye Tracker*, dem *XNA System* und *Windows Forms*.

Die Bedeutung dieser Diplomarbeit ist zweifältig. Erstens hilft sie Eye Tracking zu einer Technologie heranzureifen, die benutzt werden kann, um einfach visuelle Aufmerksamkeit in Simulationen und bspw. Spieleanwendungen zu erforschen. Für diesen Zweck wird eine Softwarelösung zum Erstellen und Erkundensichten von Blickvisualisierungen in 3D Umgebungen beschrieben. Zweitens dient die Arbeit als Fundament für zukünftige Forschung im Bereich von Blickvisualisierungen in virtuellen Welten durch die Bereitstellung des methodischen und taxonomischen Rahmens für solche Visualisierungstechniken.

*Really great people make you feel that you, too, can become great.*

— Mark Twain

## ACKNOWLEDGMENTS

---

This thesis would not have been possible if it wasn't for my friends, family, and colleagues. Thank you very much!

Ein großes Dankeschön geht an meine Familie, für die ich diesen Absatz extra auf Deutsch schreibe: *Monika, Cuno, Eva, und Rocco*. Ihr habt mich über die Jahre, wo ihr nur konntet unterstützt. Mit dieser Diplomarbeit schließe ich nun mein Studium der Computervisualistik ab. Auch ihr habt euren Anteil daran geleistet und dafür und für vieles mehr möchte ich euch danken!

I want to thank *Lennart Nacke* for patiently bearing with me, cheering me up during long working hours, giving constructive feedback, inspiring, and motivating me. Thank you for being such a wonderful partner!

Huge gratitude goes to *Prof. Raimund Dachsel* for supervising my work, for profoundly interesting discussions, and constructive feedback. In addition, I would like to thank my colleagues at GAMALab, *Prof. Craig Lindley* and *Charlotte Sennersten*, for providing a work environment for me and for giving feedback on my work. Additionally, I'd like to thank the European Commission for funding this research under the 6th Framework Programme - NEST, FUGA (FP6-NEST-28765).

I thank my friends for making me laugh even when I didn't feel like it. The best motivation ever! Thank you! Special thanks goes to *Alexander Kuhn* for providing 3D models, which I could use for my scenarios, and *Ulrike Zenner* for diligently proof-reading this thesis.

Finally, I'd like to thank all eye tracking professionals who took time off their busy schedules to participate in my survey. Insights from this survey have significantly helped to identify needs for improved gaze analysis techniques.

# CONTENTS

---

|       |   |    |
|-------|---|----|
| 1     | INTRODUCTION                                | 4  |
| 1.1   | Motivation                                  | 4  |
| 1.2   | Research Objectives                         | 5  |
| 1.3   | Structure of this Thesis                    | 6  |
| 2     | BASICS OF EYE TRACKING                      | 7  |
| 2.1   | Eye Movements                               | 7  |
| 2.2   | The Eye Tracker System                      | 8  |
| 2.2.1 | Tobii 1750 Eye Tracker                      | 8  |
| 2.3   | Eye Tracking in Virtual Environments        | 11 |
| 2.3.1 | Eye Tracking as an Interaction Technique    | 12 |
| 2.3.2 | Diagnostic Use of Eye Tracking              | 13 |
| 2.3.3 | Logging of Gaze Data                        | 14 |
| 2.4   | Survey on Improved Gaze Analysis Techniques | 16 |
| 2.4.1 | Method                                      | 16 |
| 2.4.2 | Evaluation                                  | 17 |
| 2.4.3 | Discussion                                  | 19 |
| 3     | GAZE VISUALIZATION TAXONOMY                 | 20 |
| 3.1   | Taxonomy Formulation                        | 20 |
| 3.1.1 | Input Space                                 | 21 |
| 3.1.2 | Mapping                                     | 23 |
| 3.1.3 | Output Space                                | 24 |
| 3.2   | Gaze Data Analysis Tools                    | 24 |
| 3.2.1 | General Analysis Tools                      | 25 |
| 3.2.2 | Tools for Behavioral Studies                | 26 |
| 3.2.3 | Specialized Gaze Data Analysis Tools        | 26 |
| 3.3   | Gaze Visualizations                         | 28 |
| 3.3.1 | One-dimensional                             | 28 |
| 3.3.2 | Temporal                                    | 29 |
| 3.3.3 | Two-dimensional                             | 30 |
| 3.3.4 | Three-dimensional                           | 35 |
| 3.3.5 | Multi-dimensional                           | 37 |
| 3.3.6 | Tree  | 37 |
| 3.3.7 | Network                                     | 38 |
| 3.4   | Conclusion                                  | 39 |

|       |  |    |
|-------|--|----|
| 4     | CONCEPT FOR ENHANCED VISUAL GAZE ANALYSIS TECHNIQUES | 41 |
| 4.1   | Requirements Analysis                                | 41 |
| 4.1.1 | Scenarios using Personas                             | 41 |
| 4.1.2 | Input Space Specification                            | 43 |
| 4.1.3 | Further Requirements                                 | 49 |
| 4.2   | Design of Gaze Visualizations                        | 51 |
| 4.2.1 | Models of Interest Timeline                          | 51 |
| 4.2.2 | Three-Dimensional Scan Paths                         | 54 |
| 4.2.3 | Aggregated Representations                           | 57 |
| 4.2.4 | Summary  | 61 |
| 4.3   | Coherent Application Design                          | 61 |
| 4.3.1 | Screen Layout  | 61 |
| 4.3.2 | Linked Views   | 62 |
| 4.3.3 | Features   | 65 |
| 4.3.4 | Tasks  | 66 |
| 4.4   | Summary  | 68 |
| 5     | IMPLEMENTATION                                       | 69 |
| 5.1   | Development Environment                              | 69 |
| 5.1.1 | Eye Tracker Integration with XNA                     | 71 |
| 5.1.2 | XNA and Windows Forms Integration                    | 73 |
| 5.2   | Data Acquisition                                     | 75 |
| 5.2.1 | Object Tracking with Triangle Accuracy               | 75 |
| 5.2.2 | Data Filtering and Processing                        | 77 |
| 5.2.3 | Data Logging   | 77 |
| 5.3   | Implemented Visualizations                           | 78 |
| 5.3.1 | Models of Interest Timeline                          | 78 |
| 5.3.2 | Scan and Camera Paths                                | 79 |
| 5.3.3 | Aggregated Representations                           | 81 |
| 5.4   | Implementation of a Coherent Application             | 84 |
| 5.5   | Summary  | 85 |
| 6     | CONCLUSION AND FURTHER WORK                          | 87 |
| 6.1   | Summary  | 87 |
| 6.2   | Discussion   | 88 |
| 6.3   | Further Work   | 90 |
| 7     | APPENDIX   | 91 |
|       | BIBLIOGRAPHY   | 93 |

## ACRONYMS

---

|           |  |
|-----------|--|
| AOI       | Area of Interest   |
| API       | Application Programming Interface  |
| COM       | Component Object Model   |
| ID        | Identification   |
| IP        | Internet Protocol  |
| LOD       | Level of Detail  |
| NPC       | Non-Player Character   |
| MOI       | Model of Interest  |
| ROI       | Region of Interest   |
| SDK       | Software Development Kit   |
| SVEETER   | Sophie's Visualization Environment for Eye<br>Tracking & Experimental Research |
| TetComp   | Tobii Eye Tracking Components <a href="#">API</a>                              |
| TETServer | Tobii Eye Tracker Server   |
| VOI       | Volume of Interest   |
| VE        | Virtual Environment  |
| VR        | Virtual Reality  |
| XML       | Extensible Markup Language   |
| 1D        | One-dimensional  |
| 2D        | Two-dimensional  |
| 3D        | Three-dimensional  |

## LIST OF FIGURES

---

|           |   |    |
|-----------|---|----|
| Figure 1  | The Tobii 1750 Eye Tracker.   | 9  |
| Figure 2  | Overview of the proposed gaze visualization taxonomy.   | 21 |
| Figure 3  | Enhanced stimulus classification (including examples).  | 23 |
| Figure 4  | Screenshot from Spotfire’s online sample application ( <a href="http://spotfire.tibco.com/">http://spotfire.tibco.com/</a> ).                     | 26 |
| Figure 5  | AOI and heat map visualizations in Tobii Studio ( <a href="http://www.tobii.com/">http://www.tobii.com/</a> ).                                    | 27 |
| Figure 6  | AOIs vs. time graph by Lessing and Linge [LLo2].  | 29 |
| Figure 7  | Two examples of scan path visualizations are shown.   | 31 |
| Figure 8  | Gaze visualization on flattened object [RTSB04].  | 32 |
| Figure 9  | Example for pure scan paths from free examination of the picture for 2 minutes [Yar67].   | 32 |
| Figure 10 | Scan path in Tobii Studio ( <a href="http://www.tobii.com/">http://www.tobii.com/</a> ).  | 33 |
| Figure 11 | Heat map in BeGaze [SMIo8].   | 33 |
| Figure 12 | Example from an eye tracking study by Wooding [Woo02].  | 34 |
| Figure 13 | Attentional map variations as implemented by NYAN 2.0 <sup>X†</sup> [@in09].  | 35 |
| Figure 14 | Gaze depth paths within a VR training scenario [DMC <sup>+</sup> 02].   | 36 |
| Figure 15 | VOIs as described by Duchowski [DM98].  | 36 |
| Figure 16 | Superimposed fixation maps.   | 37 |
| Figure 17 | Transitional graphs by Lessing and Linge [LLo2].  | 38 |
| Figure 18 | Overview of the applicability of the presented gaze visualizations to different stimuli. Users are assumed to be able to change their viewpoints. | 40 |

|           |   |
|-----------|---|
| Figure 19 | Gaze ray from the viewer to the screen and into the VE. 47  |
| Figure 20 | Multiple selections and magnification factors in the MOI timeline. 52   |
| Figure 21 | Semantic fixation zooming is illustrated. If a user zooms out, fixation sizes remain the same compared to the screen size. In order to decrease clutter, fixations may also be combined in a clustered representation. 56 |
| Figure 22 | Alternatives for representing fixations in 3D VEs are shown. Besides fixation locations and durations, viewing directions, and distances are incorporated. 57   |
| Figure 23 | Concept for different levels of detail for aggregated gaze visualizations. 58   |
| Figure 24 | Gaze data is projected onto the top and side view planes. 59  |
| Figure 25 | A cut-out of a model mesh is displayed to illustrate the triangle-based attentional representation. 60  |
| Figure 26 | Concept of the screen layout. 62  |
| Figure 27 | Different views are linked by consistent coloring. Each object receives a specified color, which can be changed by the user. 63   |
| Figure 28 | Different alternatives of how to represent objects and visual attention with colors are presented. The blue color represents the cube entity, while the red color is assigned to the cube's visual attractiveness. 64     |
| Figure 29 | Overview of system environments required for different stages of an eye tracking study lifecycle, including preparation (scenario design), testing, and analysis (gaze visualization). 70                                 |
| Figure 30 | Dependencies and categories of the Tet-Comp objects 71  |
| Figure 31 | Dual computer, dual screen setup and communication between the different systems. 72  |

|           |   |
|-----------|---|
| Figure 32 | Structure for the integration of the Tobii ET-1750 with the XNA Framework. 72   |
| Figure 33 | Overview of the implemented system for the gaze viewer application. 73  |
| Figure 34 | XML syntax for the different log file types 78  |
| Figure 35 | An example of the implemented MOI timeline. 78  |
| Figure 36 | The data collection can be navigated by using additional sliders in the MOI timeline. 79  |
| Figure 37 | Screenshots from the implemented 3D scan paths. 80  |
| Figure 38 | Fixations can be represented by cones. The magnification factors from fixations can be dynamically adjusted. 80   |
| Figure 39 | Two examples for camera paths and viewing traces. 81  |
| Figure 40 | Projected attentional map superimposed over the top view. 82  |
| Figure 41 | Example of an object-based attentional representation (with filtered textures). 82  |
| Figure 42 | Gaze distribution is represented as an attentional heat map skinned to the model's surface. Examples for differing radius values are presented, while transparency is 80% and $\sigma^2$ is 0,5. 83 |
| Figure 43 | Different color gradients can be assigned to visualize gaze distribution. 84  |
| Figure 44 | Screenshot of SVEETER. 85   |

## INTRODUCTION

---

### 1.1 MOTIVATION

The eye is a truly amazing organ providing us with the ability to detect light, colors, and even shapes. Their behavior can give a suggestion about cognitive processes. Whether this is about the words we concentrate on while reading, how we observe a scene, or how our next move in a chess game could look like. Our eyes are one of the (if not even the) most vital of your sensory inputs.

This has led to the study of eye movements, which has been pursued for over a century. At the beginning techniques were quite obtrusive. Subjects had to use bite boards to hold their heads motionless or mechanical contact lenses that were attached to the eyes [Jac91]. Modern methods manage without any physical contact, which is much more comfortable for the user. The constantly increasing quality, accuracy, and convenience of eye tracking devices promote their use in different application areas. This includes user experience studies and the deployment of eye tracking as input modality for human-computer interaction. Recorded eye movements can be analyzed by using gaze visualizations. In general, information visualization can help researchers to understand and interpret critical relationships in multivariate and multidimensional data [CMS99]. Thereby, such visualizations play an essential role for revealing phenomena hidden in extensive numerical datasets.

Over the years the majority of diagnostic eye tracking studies have focused on the evaluation of visual attention on images [Woo02, BPF03], advertisements, and websites [SDTF03, SNo8, YJSHo8]. For this reason, two-dimensional (2D) stimuli are commonly deployed for which gaze data are visualized as superimposed graphics (e.g., heat maps and gaze plots). Such visualizations provide quick insights into areas observers have looked at. Existing gaze visualization techniques are, however, not well adapted to support interactive visual analysis of three-dimensional (3D) virtual environments (VEs).

Current visual gaze analysis techniques for dynamic and 3D stimuli usually include frame-by-frame evaluations of video data grabbed from the video device on which the stimulus was rendered. As a result experiments include recording video files along with additional data, such as gaze points. During the subsequent analysis, video replays are incorporated with superimposed gaze graphs for each frame. Although this procedure is effective, it is rather inefficient due to its time-consuming procedure. Solutions to improve this issue were done within game research by integrating game engines and eye tracking systems to automatically log viewed game objects [SAC<sup>+</sup>07, Steo7, SLo8]. However, a simple and intuitive way to visualize gaze data for dynamic stimuli is still missing [RTSB04, Š08].

There is a great potential for interactive, 3D gaze visualizations, since they allow researchers to quickly grasp which details of a certain object have received high visual attention. This becomes more and more important for industry and research. Hence, techniques supporting quick and convenient visual gaze analysis in 3D VEs need to be developed. The development (including design, implementation, and evaluation) of such interactive techniques to study major surrounding phenomena is classified within the field of human-computer interaction (HCI) [HBC<sup>+</sup>96].

## 1.2 RESEARCH OBJECTIVES

The objective of this work is to investigate ways to facilitate eye tracking studies in 3D VEs by enhancing visual gaze analysis. In this context appropriate and coherent visualizations of gaze data need to be designed and evaluated. Interaction techniques for operating on such multivariate data sets need to be developed. Before designing software that accurately incorporates the aforementioned features, a taxonomy of gaze visualizations is established. This way important strengths and weaknesses of current gaze visualization techniques can be identified. Based on this taxonomy appropriate and novel gaze visualizations are devised and implemented as prototypes.

### 1.3 STRUCTURE OF THIS THESIS

This thesis investigates the enhancement of visual gaze analysis techniques for *three-dimensional (3D) virtual environments (VEs)*. Before starting to devise such techniques, basic knowledge about eye movements, eye tracking, and its application in *3D VEs* has to be established (see *Chapter 2*). In addition, requirements for improved gaze analysis have to be specified. For this purpose a survey has been conducted with eye tracking professionals and researchers asking about features they would desire for convenient eye tracking analysis.

An important aspect of visual gaze analysis are visualizations. In order to learn from existing approaches, a gaze taxonomy describing and classifying gaze visualizations is presented in *Chapter 3*. This will help to identify strengths and weaknesses of current techniques.

The insights gained from the survey and from classifying gaze visualizations provide a great foundation for developing enhanced visual analysis techniques. Concepts for visualizations applicable in *3D VEs* and supporting features are described in *Chapter 4*.

Based on the devised concepts the implementation of a gaze analysis tool, *Sophie's Visualization Environment for Eye Tracking & Experimental Research (SVEETER)*, is addressed in *Chapter 5*. An integration of the *Tobii 1750 Eye Tracker* with the *XNA Framework* is used for scenario creation and data logging. Reported data can be analyzed with *SVEETER*, which is based on the *XNA Framework* and *Windows Forms*.

*Chapter 6* concludes this thesis by summarizing the accomplished work, evaluating the results, and looking at future research tasks.

## BASICS OF EYE TRACKING

---

In order to improve gaze analysis in *three-dimensional (3D) virtual environments (VEs)* it is important to learn about characteristic eye movements (Section 2.1) and eye tracking (Section 2.2) first. In this context the *Tobii 1750 Eye Tracker* is introduced, which will be used for testing implemented concepts due to its availability in our local laboratory. After basic knowledge about eye tracking has been established, possible applications in VEs will be examined in Section 2.3. Finally, possibilities of how to improve gaze analysis techniques will be discussed. For this purpose, a survey with eye tracking professionals asking about desired features has been conducted and is described in Section 2.4.

### 2.1 EYE MOVEMENTS

Although we are generally not consciously aware of it, but our eyes are in constant movement. This movement is, however, not steady and smooth across the visual field. Both eyes often make quick simultaneous movements in the same direction, which are called *eye saccades* [Cas01]. The visual system is suppressed to a high extent during saccades [Jö5]. A saccade is usually followed by a *visual fixation*. During a fixation, which usually lasts about 200 to 600 milliseconds [Jac91], items can be observed. However, even during fixations the human eye is vibrating constantly causing small involuntary eye movements called *microsaccades*, which are usually completely imperceptible [Jac91].

In order to see an object clearly the eyes have to be moved in the gaze line of the item using saccades and fixations. Hence, they provide good indications of what parts of a scene are examined [Jac91]. Microsaccades on the other hand are involuntary eye movements. Eye tracking researchers are usually not interested in examining what eye muscle reflexes dictate subjects to do. Therefore, microsaccades are usually filtered out in eye tracking studies.

In a nutshell, the most important eye movements with re-

spect to eye tracking analysis (concerning the study of visual line of gaze) are *fixations* and *saccades*.

## 2.2 THE EYE TRACKER SYSTEM

The study of eye movements has been pursued for over a century (e.g., Javal in 1879 [Hue68]). Except for mere visual observations, eye tracking methods have long been quite obtrusive. This involved, for instance, techniques where mechanical contact lenses were placed directly on a subject's cornea<sup>1</sup> [JK03]. Already at the beginning of the twentieth century Dodge and Cline developed a first non-invasive eye tracking technique using reflected light from the cornea [DC01]. However, this procedure demanded participants to keep their heads motionless and only recorded horizontal eye movements.

Over time techniques have improved and become less obtrusive. Advanced computer performance enables procedures such as video-based corneal reflection eye tracking to deliver high accuracy gaze data in real-time. For this technique, near-infrared light is reflected by the cornea and recorded by a camera. If the light source is positioned coaxial with the camera, the pupil will appear as a bright circle [Jö5]. Image processing enables eye detection and calculation of gaze positions. This unobtrusive technique is practical also for non-laboratory settings, since the eye tracker can be integrated in a computer screen several feet away from the user. The user is not limited by additional equipment; however, extensive movements should be avoided to maintain good eye tracking results. The Tobii 1750 Eye Tracker, which is used in the scope of this thesis, integrates this procedure in a table-mounted screen application.

### 2.2.1 Tobii 1750 Eye Tracker

In this thesis, the development of enhanced gaze analysis techniques is based on the Tobii 1750 Eye Tracker (or Tobii ET-1750). Hence, it is important to learn about the system's characteristics, which include logged data types and system limitations.

---

<sup>1</sup> The cornea is the transparent front part of the eye that covers the iris, pupil, and anterior chamber, and provides most of an eye's optical power [Cas01].



Figure 1. The Tobii 1750 Eye Tracker.

As previously mentioned, the Tobii ET-1750 is a table-mounted eye tracker based on corneal reflection (see Figure 1). It has a stable frame rate of 50 Hz; this means that gaze data are collected 50 times per second. Tracking of both eyes simultaneously, which is called *binocular eye tracking*, is possible. The system automatically determines which eye is left and which one is right. This increases robustness against head movements and blinking.

#### *Eye Tracking Data Types*

Since the development is based on the Tobii 1750 Eye Tracker supplied data types by the system have to be specified. Data allocated by the Tobii *Application Programming Interface (API)* are listed in Table 1 (see Tobii's user's guide [Tob05] for a complete list). Ramloll et al. identified typical data types of interest to eye tracking researchers (independent from a particular eye tracking device) [RTSB04]. They distinguish between four orders of gaze data. From first order data, as provided by the Tobii API, additional values can be derived. This includes the calculation of fixations and saccades (second order), the total fixation times in certain areas (third order) as well as the shape of scan paths (fourth order).

#### *Eye Tracking Accuracy*

Different conditions can affect eye tracking quality. In general, there is a distinction between persistent and variable

| <i>Data type</i>     | <i>Description</i>  |
|----------------------|---|
| Counter              | <i>Log entry enumerator</i>   |
| Time stamp           | <i>Time at which gaze was sampled in milliseconds</i>   |
| Gaze target position | <i>Screen-based gaze position ranging from (0, 0) upper left to (1, 1) lower right screen corner</i>  |
| Eye position         | <i>Position of the eye as seen by the eye tracker ranging from (0, 0) upper left corner of the eye tracking sensor and (1, 1) lower right sensor</i>                        |
| Distance             | <i>Shortest distance between tracking sensor and eye</i>  |
| Pupil size           | <i>Pupil dilation in millimeter</i>   |
| Validity             | <i>Certainty of correct data correlation - Values range between 0 (certain) and 4 (uncertain). It is recommended to remove entries with validity codes above 2 [Tob05].</i> |

Table 1. Data allocated by the Tobii [API](#).

influences. Some factors are (more or less) persistent over the course of an eye tracking study. This includes certain characteristics which relate to the particular eye tracking system. For example, eye tracking systems have to operate at a certain threshold and interpolate gaze positions due to microsaccades. This frame-to-frame variation of measured gaze points is called *spatial resolution* (or noise) and is approximately 0.25 degrees for the Tobii ET-1750. In addition, there is an *average error* of 0.5 cm between the measured and the actual gaze point (assuming a distance of 50 cm between the user and the viewed object). Further information about accuracy of the Tobii ET-1750 can be found in its product description [Tob04].

Besides persistent factors, variable factors may change from one session to another within an eye tracking study. Such conditions include [Tob05]:

- Calibration quality

- Similarity between current light and environment conditions to conditions during calibration
- Distance of gaze target to calibration areas
- Head and pupil motion
- Glasses
- Diversion of the cornea of a subject's eye from the assumed average human cornea

Since the calibration is an important aspect affecting accuracy, it is briefly discussed in the following paragraph.

#### *Calibration Process*

Everybody is unique and so are their eyes. This results in the need to adapt eye trackers to individual users in a calibration procedure. In the case of the Tobii ET-1750 the calibration is long-lasting. This means that calibration data can be saved in personal profiles to be reused for continued studies with corresponding subjects [Tob04]. The calibration window has a plain-colored background, on which one circle at a time is displayed. The observer has to follow the circle, which first appears in the upper left corner from where it moves to the first calibration location. There it changes size to catch the user's attention and collects calibration data. Once enough data has been gathered the point moves on to the next calibration location. The calibration works with as low as 2 locations or as high as 9 [Tob05]. The higher the amount of calibration locations the longer the procedure takes. This duration can range from 5 to 60 seconds.

In summary, this section briefly introduced the Tobii 1750 Eye Tracker, which the development of enhanced gaze analysis techniques is based on. For this reason, available data types and system constraints concerning eye tracking accuracy have been specified.

### 2.3 EYE TRACKING IN VIRTUAL ENVIRONMENTS

After discussing basic information about eye movements and eye tracking, it is important to investigate the application of eye tracking in *virtual environments* (VEs) to understand its

usefulness in such contexts. For this reason, this section addresses eye tracking studies in 3D VEs. In accordance with this thesis, VEs generally represent computer-based simulations.

In general, Duchowski proposes two main application areas of eye tracking: *Interactive* and *diagnostic* application [Duc02]. While in Section 2.3.1 eye tracking is discussed as an interaction technique, reasons and approaches for diagnostic use are examined in Section 2.3.2.

### 2.3.1 Eye Tracking as an Interaction Technique

Gaze-based interactions are a prevalent topic in the discussion concerning 3D VEs and eye tracking [WM87, Jac90]. While Jacob discusses the design of appropriate interaction techniques that use eye movements as an input device for 2D environments [Jac90], Tanriverdi and Jacob work with 3D VEs, in which interaction based on eye movement may provide “an easy, natural, and fast way of interacting” [TJ00]. This claim is supported by their comparison of gaze-based and conventional 3D pointing. Tanriverdi and Jacob noticed that eye movements show a significant speed advantage for selecting distant virtual items [TJ00]. Cournia, Smith, and Duchowski continued this work and actually showed that, when adjusting the conventional 3D pointing, no clear performance advantage could be measured anymore [CSD03].

Eye tracking has also been successfully used in studies of gaze control as an interaction technique in digital games [Jö5, SGo6, Gow07, IJvM09]. Gowases presents problem solving games, which are controllable via gaze and mouse [Gow07]. A study focusing on the use of a player’s pupil size for controlling digital games is presented by Ekman et al. [EPM08b]. In a more detailed publication Ekman et al. investigate techniques for voluntarily changing pupil size as an input modality [EPM<sup>+</sup>08a].

Striving for a more natural way of remote social interaction, Steptoe et al. apply subjects’ gaze to the animation of their corresponding virtual avatars [SWM<sup>+</sup>08]. This supports non-face-to-face communications by including additional non-verbal cues.

Gaze data may also be used to improve development issues, for example, by means of increased performance. Several studies engage in improving video compression rates based on

gaze position information [DM98, NH07]. Regions receiving high gaze density are assigned a higher level-of-detail (LOD) before encoding. This concept is also applicable to 3D environments. Itti, Koch and Niebur propose an attentional model for predicting human fixations for view-dependent enhancement of VEs [IKN98]. Danforth et al. present a platform for gaze-contingent 3D VEs allowing dynamic LOD changes to render scenes containing significant topological detail based on user position and gaze direction [DDGM00].

### 2.3.2 Diagnostic Use of Eye Tracking

Using eye tracking as a diagnostic tool provides objective and quantitative evidence of a viewer’s visual and attentional processes [Duc02]. For this purpose, gaze data is usually recorded for post-hoc, offline assessment. Accordingly, the stimulus does not need to provide immediate (e.g., visual) feedback on a user’s gaze. Gaze data logging methods are briefly discussed in Section 2.3.3. The desire to analyze gaze data can be motivated by several objectives. This may include the desire to investigate mental processes (cognition), visual perception, and user behaviors. Some examples for such motivations are presented in Table 2.

| <i>Focus on...</i> | <i>Example for arising questions</i>  |
|--------------------|---|
| Cognition          | <i>How can gaze patterns reveal insights into processes of thought?</i>                       |
| Perception         | <i>Do prevalent gaze patterns exist? Do certain areas provoke increased visual attention?</i> |
| Development        | <i>Can areas of low visual attention be produced at a lower cost?</i>                         |
| User               | <i>Can an application evoke greater user satisfaction based on gaze patterns?</i>             |

Table 2. Examples for different motivations for gaze data analysis

An example for cognition-centered gaze analysis is presented by Yarbus [Yar67]. He described several eye tracking studies in which subjects had to accomplish certain tasks while looking at images [Yar67]. Subjects were, for example, asked to examine a given drawing and estimate the age of

depicted people. Yarbus noticed that gaze patterns change depending on the given task. Therewith, Yarbus suggested that an observer's thoughts are conveyed (at least to some extent) by their gaze behavior. In addition, gaze patterns may give information about a person's expertise. Reingold et al. investigated chess players' gaze behavior, which indicated that chess experts have a perceptual encoding advantage [RCPS01]. Law et al. found differences between eye movements of novices and experts in performing minimally invasive surgery [LAKLo4], which provides additional evidence for a link between gaze behavior and professional expertise.

Perception-centered gaze analysis focuses on finding out if certain stimulus regions provoke increased visual attention. This information can be used to derive prevalent gaze patterns. Skrba, O'Connell and O'Sullivan investigated, for example, which parts of quadrupeds receive high visual attention. For this purpose, they showed short video clips of quadrupeds (mainly farm animals) to participants [SOO08]. They identified a typical eye-movement pattern, starting from an animal's head, along its torso and ending with its hips.

A user-centered approach focuses on adapting to the needs of the user to improve their user experience (including satisfaction). Jie and Clark present a model of attention allocation for dynamically adjusting a game's difficulty [JCo8]. Based on this model, the game difficulty is increased by placing game-relevant items in areas with low visual attention.

Regardless of the specific motivation for using eye tracking for diagnostics, they all have one thing in common: gaze data has to be logged for post-hoc analysis. However, there may be significant differences between the logging of gaze data for 2D and 3D stimuli. Therefore, gaze data logging methods will be addressed more closely in the following.

### 2.3.3 *Logging of Gaze Data*

Both interactive and diagnostic eye tracking applications depend on gaze data. While gaze interaction only requires temporary access, data need to be logged to files or databases for post-hoc analysis. As previously mentioned, such data include gaze positions and pupil sizes. These data are independent from the particular stimulus. That means that the eye tracker provides the same basic data types no matter if images, videos,

or 3D VEs are used. On the one hand this provides high applicability, since any stimulus can be employed. On the other hand it complicates easy data comprehension. For example, if an eye tracking study is conducted using digital games, the analyst will have problems to draw quick conclusions only by inspecting logged gaze points. Instead, a common approach is to examine video recordings that contain superimposed gaze visualizations. This procedure can be very time-consuming, since video frames have to be inspected individually.

The use of eye tracking in VEs allows for an advanced approach by integrating eye tracking frameworks with software systems (e.g., a game engine). That way viewed objects can be logged with descriptive identifiers. Automated gaze object logging may greatly facilitate the analysis of visual attention in dynamic computer gameplay [SLo8]. In this respect, Sennersten et al. presented an automated logging method for viewed objects in a First-Person Shooter for the HiFi game engine [SAC<sup>+</sup>07]. This included a verification study assuring quantifiable accuracy for conducting statistical analysis and a thorough investigation of object logs. Two related projects describe the integration of the Tobii ET-1750 with Half Life 2's Source SDK [Ste07] and with the TorqueX game engine [Saso8] for automated object logging.

The automated logging of viewed object identifications can facilitate answering questions like: *How often has a particular item been looked at? Are certain objects often or always viewed consecutively?* Object identifiers may describe complex entities, omitting detailed information about which parts (e.g., head, eyes, or torso of a humanoid model) actually attracted attention. Using the aforementioned object logging method to evaluate a single but highly complex 3D model is therefore insufficient, because questions concerning detailed visual attention patterns on 3D objects cannot be answered. Since 3D models are usually represented as triangle meshes, viewed triangles could be logged to provide detailed information. For the development of enhanced gaze analysis techniques such a triangle-precise logging method will be used. The procedure is described in Section 4.1.2 and its implementation is addressed in Section 5.2.1.

In summary, various examples for the application of eye tracking in VEs have been mentioned. Allocating gaze data plays a vital role for both interactive and diagnostic eye tracking

applications. The post-hoc analysis of 3D VEs usually includes a time-consuming frame-by-frame analysis. The integration of eye tracking systems with software environments may facilitate such studies by logging related data such as identifiers of viewed objects. In the scope of this thesis, a triangle-precise eye tracking method is used to provide detailed information about gaze distributions. Features to further enhance gaze analysis are addressed in the next section.

## 2.4 SURVEY ON IMPROVED GAZE ANALYSIS TECHNIQUES

In order to facilitate gaze analysis an interview with eye tracking professionals has been conducted to examine their particular desires. The complete questionnaire is listed in the Appendix, page 91.

### 2.4.1 Method

#### *Participants*

Ten eye tracking professionals and researchers aged between 28 and 57 years ( $M = 37.7$ ,  $SD = 9.38$ ) participated in this online survey. Of the total, 50% ( $N = 5$ ) were female and 50% ( $N = 5$ ) were male. Participants had been working with eye trackers for 2 to 15 years with an average of 7.2 years ( $SD = 4.49$ ). When asked how many studies incorporating eye tracking analysis they had worked on, participants' experience ranged between 3 and 50 studies ( $M = 15.9$ ,  $SD = 14.71$ ).

#### *Survey Design, Apparatus, and Procedure*

The survey consisted of demographic, mixed-method (eight quantitative and four qualitative) questions. Some quantitative questions were aimed at evaluating the importance of visualizations for eye tracking analysis (“How important were visualizations for the analysis of your eye tracking studies?”, “How would you assess the importance of sophisticated gaze visualization techniques for dynamic virtual environments?”). These evaluations were done on a scale ranging from 1, not important, to 5, very important. Other quantitative questions were aimed at uncovering the stimuli types used in these studies (static [2D, 3D], dynamic [2D, 3D], interactive [2D, 3D]). The qualitative questions were to inquire about personal experiences (“What are your experiences in designing and analyzing eye tracking ex-

*periments employing dynamic interactive stimuli?”), weaknesses of current gaze visualizations (“Where do you see weaknesses in current gaze visualization techniques?”), and desirable features for gaze analysis (“What features would you desire for a simple and intuitive gaze analysis?”). The survey was implemented online using the tool LimeSurvey<sup>2</sup> (Version 1.70+). Thirty-two researchers were selected from searches on major eye tracking publication venues (COGAIN, ETRA and ECEM) and together with staff from Tobii Technology AB. Anonymous identifiers were assigned to each participant and they were then invited via email to participate in the online survey. No financial incentives were offered for participation.*

#### 2.4.2 Evaluation

##### *Quantitative results*

Visualizations for eye tracking analysis were assessed as important ( $M = 3.7$ ,  $SD = 0.95$ ). Gaze visualizations for dynamic VEs were estimated to be even more important ( $M = 3.9$ ,  $SD = 0.88$ ). Ninety percent of the participants had already used static 2D stimuli in their experiments. Following this, 70% had drawn on interactive 2D stimuli (user interfaces) and 40% had deployed 2D dynamic stimuli (videos). Only 20% had already used 3D stimuli of all kinds (static, dynamic, and interactive).

##### *Qualitative results*

Among the ten people interviewed, six people made a statement about their application areas (multiple selections were possible): four employed eye tracking for diagnosis and four for interactive applications. Only two respondents had conducted studies using virtual 3D stimuli.

Although gaze visualizations are generally considered important ( $M = 3.7$ ,  $SD = 0.95$ ), researchers using eye tracking for interaction studies declared that they “almost never did any visualizations” (Male, 10 years work experience with eye tracking). For diagnostic purposes, however, visualizations provide “the only way to quickly review and analyze what participants saw/did not see” (Female, 8 years work experience). Thus, gaze visualizations can play an essential role in detecting gaze patterns. This is especially beneficial when trying to commu-

---

<sup>2</sup> Open source survey application mainly developed by Carsten Schmitz and available for download at [www.limesurvey.org](http://www.limesurvey.org).

nicate findings to customers. Nevertheless, one respondent argued that visualizations *“do not prove anything”* (Female, 8 years work experience).

In spite of the importance granted to gaze visualizations by most interviewees, several drawbacks were also specified. An important issue is the lack of effective dynamic visualization methods. In fact, eight people agreed with the following statement: *“So far, nobody has come across a simple and intuitive way to visualize gaze data for dynamic stimuli.”* [RTSB04, Š08]. One person disagreed, claiming that gaze replays are an intuitive way to visualize gaze positions. Another participant named several approaches for improving gaze visualizations for videos [DM98, NNB<sup>+</sup>04, SGGD05], but granted the difficulty of considering these techniques simple or intuitive.

A general problem with gaze visualizations is the danger of misinterpretation, which was mentioned by one interviewee. Reports on weaknesses of current visual gaze analyses mainly targeted two areas: features of available tools and of visualizations. Gaze analysis tools were criticized for their lack of multiple views and properly formatted data. In addition, further improvements of fixation identification algorithms and the conversion of processed visualizations into table data was called for. It was also mentioned that gaze visualizations for 3D contexts need to be improved and should provide aggregated comparisons (i.e., 3D heat maps). Some participants warned that scan paths from multiple participants (especially in 3D) may result in visual clutter as well as problems with temporal synchronization. Another respondent advised that *“simple gaze path animations may not help in seeing repeating patterns”* (Female, 8 years work experience).

Concerning eye tracking studies within VEs, one interviewee requested the possibility to *“compare multiple scan paths in the VE while still allowing the person viewing this to control their own position/orientation in the VE”* (Male, 15 years work experience). Another respondent (Male, 10 years work experience) provided the following recommendations: 1. Easy way to specify ‘variables of interest’ (in contrast to areas of interest); 2. Multiple visualizations and easy-to-follow links between them; 3. Ability to select and edit gaze data in a very flexible way (in tables, interactively), having the ‘variables of interest’ recalculating, and multiple visualization repainting ‘on-the-fly’.

Further requests included:

- Integration of signal processing methods such as frequency analysis for automatic pattern detection
- Automatic relocation of [AOIs](#) for moving objects (videos)
- Visualization of transitions between different views
- Integration of external data sources
- Multiple views, overview-and-detail, backtracking
- Data format that allows the use of statistical programs
- Annotation tool (allowing researchers to attach and share comments in the playback)
- Not being time-consuming to set up and analyze (quick deployment)

#### 2.4.3 *Discussion*

The presented results clearly imply the importance of gaze visualizations for diagnostic eye tracking studies. An interesting result is that a majority of eye tracking researchers have not used [3D](#) stimuli so far. Reasons for this may include a higher complexity to develop sophisticated [3D](#) scenarios and the use of frame-by-frame evaluation of session videos. Enhanced gaze visualizations for virtual environments can facilitate the analysis of such studies. This includes aggregated representations, such as heat maps. Moreover, visualizations may help in detecting gaze patterns. Beside the development of adapted visualizations, several features concerning usability of available analysis tools have been mentioned, including techniques such as overview and details-on-demand.

In conclusion, identifying what eye tracking professionals and researchers desire for improved gaze analysis provides the basis for the development of enhanced techniques. In the context of this thesis, a gaze analysis tool is developed for eye tracking studies in virtual environments. This tool shall implement novel gaze visualizations as well as additional features, for example, multiple views for enhanced post-hoc evaluation.

## GAZE VISUALIZATION TAXONOMY

---

For the development of novel gaze visualization techniques, a thorough investigation of existing approaches, their strengths and weaknesses needs to be done. The presented taxonomy helps to identify prevailing characteristics and facilitates understanding existing challenges for gaze visualizations.

### 3.1 TAXONOMY FORMULATION

A comprehensive classification of gaze visualization techniques has been described by Špakov [Šo8]. He distinguishes one-, two-, and three-dimensional illustrations in context with static and dynamic stimuli. However, this approach does not describe visualizations sufficiently for stimuli that vary in dimensions and dynamics. Building on Špakov's approach a more complex taxonomy is devised that integrates other fundamental visualization taxonomies, which are discussed in the following.

Guzdial et al. have presented a general approach for the analysis and visualization of log files [GSB<sup>+</sup>94]. Based on their work, three components for the gaze visualization taxonomy are distinguished: input space, mapping, and output space. For a more precise description of these spaces, Shneiderman's task-by-data-type taxonomy for information visualizations [Shn96] is taken into account. This taxonomy incorporates seven data types (1D, 2D, 3D, temporal, multidimensional, tree, and network data) and seven tasks (overview, zoom, filter, details-on-demand, relate, history, and extract). The specification of data types is particularly important for the input and output space description, while tasks and additional processing procedures relate to the mapping category. Figure 2 presents an outline of the gaze visualization taxonomy, while respective spaces are discussed in more detail in Section 3.1.1 through 3.1.3.

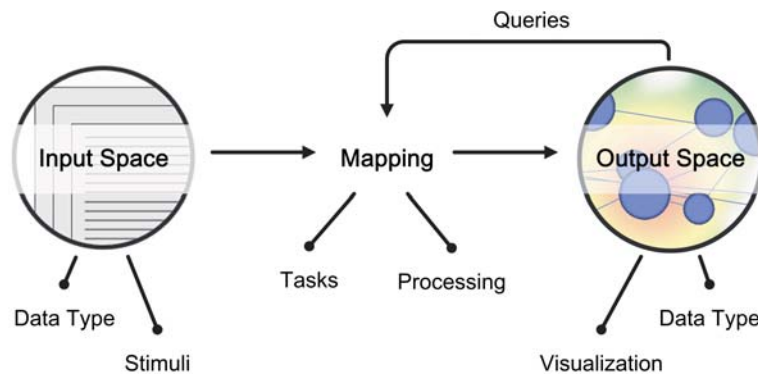


Figure 2. Overview of the proposed gaze visualization taxonomy.

### 3.1.1 *Input Space*

The input space describes conditions for an eye tracking study. This includes data types of interest, deployed stimuli, and the underlying motivation (research question). These aspects may influence the selection of appropriate visualization techniques greatly. Depending on the eye tracking study, collected data, which are usually stored in log files or databases for further analysis, may belong to various types (see Section 2.2.1, page 9). In general, gaze studies comprise the following data:

- Temporal      *Time stamps*
- Spatial        *Gaze position, eye position, pupil size*
- Semantic      *Description of occurred events and viewed items*

The underlying stimulus can be classified by its dimension, dynamic, and interactivity. Table 3 shows eye tracking studies organized by these aspects. However, it should be pointed out that dynamic stimuli may be described by static means. A dynamic website, for example, can be specified by static parts, such as a banner, a sidebar, or a main content area. If an analyst is only interested in the areas an observer has looked at, the stimulus can be described as static (due to the fixed layout). However, if the analyst wants to know what precisely has been viewed in the dynamic banner, a static description will not be sufficient. Also a combination of different dynamics is possible. An example are digital games, which usually contain static content (e.g., infrastructure) but also dynamic items (e.g., player character).

|             |    |                                |   |
|-------------|----|--------------------------------|---|
| Static      | 2D | <i>Still images</i>            | [LLo2, MTNKo2, Wooo2]                       |
|             |    | <i>Static web sites</i>        | [SDTFo3, SNo8]                              |
|             |    | <i>Data sheets</i>             | [Xu00]                                      |
|             | 3D | <i>Static 3D model</i>         | [RTSB04]                                    |
| Dynamic     | 2D | <i>Slideshows and videos</i>   | [BPFo3, SWF <sup>+</sup> o4, Spao6, VNKJo8] |
|             | 3D | <i>3D simulations</i>          | [BMS <sup>+</sup> o6]                       |
| Interactive | 2D | <i>2D user interfaces</i>      | [AHR98]                                     |
|             | 3D | <i>Virtual environments</i>    | [DMC <sup>+</sup> o2, SGGDo5]               |
|             |    | <i>3D games</i>                | [Sen04, EPMo8b]                             |
|             |    | <i>Real-world observations</i> | [BBIo7]                                     |

Table 3. General stimulus classification.

Although the presented classification is a good starting point, it cannot account for different kinds of VEs appropriately. Let us assume, for example, a virtual urban scenario in which static models of houses, trees, and streets are incorporated, and which a user can freely explore by altering their viewpoint. Based on this scenario, various aspects are not considered by the applied criteria in Table 3, such as:

- Dynamics of the user (e.g., changing viewpoint) and of virtual content (e.g., moving models, which are not user-controlled) are not distinguished.
- Although a user being able to change their viewpoint can be regarded as interactive, no interaction with the virtual environment is happening.

In addition, the applied criteria do not clearly state whether the underlying stimulus is the same for each subject (e.g., video, predefined sequence of images) or not (e.g., random images). Thus, an enhanced stimulus classification is devised, distinguishing between the behavior of users (or more accurately users' viewpoints) and the virtual context. For this reason, three distinct behaviors are identified:

- Static *Inactive, motionless*
- Predefined *Same sequence of reproducible processing steps*
- Unique *Unique and therefore not reproducible behavior*

Considering different user and VE behaviors resulted in an enhanced stimulus description matrix, which is presented in Figure 3. It includes some examples for understanding the combination of these states. Based on the resulting stimulus matrix, three overall stimuli groups can be distinguished: identical static, identical dynamic and unique stimuli.

In conclusion, the input space describes data of interest (depending on researchers' motives) and stimuli to help identifying appropriate gaze visualizations.

|             |            | Experimental context                               |   |   |
|-------------|------------|--|---|---|
|             |            | Static   | Predefined                                      | Unique                                      |
| User's view | Static     | Looking at a picture                               | Watching a movie                                | Looking at random images                    |
|             | Predefined | Scheduled camera flight through a static scenery   | Predefined rotation around an animated 3D model | Set camera flight through a 3D online world |
|             | Unique     | Freely moving around in a static urban environment | Freely exploring an animated scenario           | Playing a First-Person Shooter              |

Each subject has...

- the same **static** stimulus
- the same **dynamic** stimulus
- a **unique** stimulus

Figure 3. Enhanced stimulus classification (including examples).

### 3.1.2 Mapping

In general, mapping can be described as the dependence between two quantities. While one quantity is given (input), the other one needs to be formulated (output). Thereby, mapping describes the transformation of data into other data types and how they are associated with each other. For this purpose, the examined data (input space) and desired representations (output space) have to be questioned [GSB<sup>+</sup>94]. This means, that depending on a given input space and desired visualization type, specific mapping techniques have to be performed.

Eye tracking studies can result in extensive amounts of data, whereas usually only subsets are observed based on formulated hypotheses. Tasks for aiding data management are

proposed by Shneiderman [Shn96], which include: overview, zoom, filter, and details-on-demand. Thus, user queries need to be understood and processed for dynamically adjusting visualizations. Further processing may include data clustering and statistical calculations (e.g., number of times a certain item has been looked at). Clustered representations may contribute to a more compact data view and enable quick visual analysis.

Moreover, mapping describes the possibility to depict multiple trials (for example, different gaze log files from a particular stimulus). This, however, depends on the deployed stimulus and desired visualization technique. A scenario which is unique for each subject is not suitable for this purpose.

In summary, mapping describes the transformation of a given input space to desired visualizations. Depending on applied mapping techniques, the output space may contain various representations for a given data set.

### 3.1.3 *Output Space*

The output space describes visualizations based on a specified input space and applied mapping techniques. The resulting representations can be distinguished by their data type, dynamics, and interactivity. Shneiderman's task-by-data-type taxonomy [Shn96] is used to describe data types and interactivity (tasks) of such visualizations.

Gaze visualizations are usually implemented by gaze data analysis tools. Therefore, different gaze data analysis tools are described in the following section to find out prevalent techniques. In Section 3.3 several gaze visualizations will be described arranged according to their data type.

## 3.2 GAZE DATA ANALYSIS TOOLS

Before addressing common visualization techniques for depicting eye tracking data in Section 3.3, tools used for the analysis of gaze data are discussed. Gaze analysis tools differ in their degree of specialization. For this reason, distinct advantages and disadvantages of general and specialized solutions are considered. While a general approach is more flexible in handling various data, it requires a higher level of knowledge and precaution from the user to work with them correctly. Otherwise misleading visualizations are cre-

ated, conveying a wrong conclusion. In contrast, specialized solutions are limited to certain application areas, but may allow handling and interpreting data more appropriately. This leads to the identification of three types of analysis tools:

- *General analysis tools*, usually spreadsheet applications such as Microsoft Excel, Spotfire, and Tableau
- Specialized tools for *observational studies* (including eye tracking), as provided by Noldus and Mangold
- Tools explicitly specializing in *gaze analysis* such as Tobii Studio, BeGaze, GazeTracker, NYAN 2.0<sup>XI</sup>, and OGAMA

### 3.2.1 General Analysis Tools

Spreadsheets and databases containing any type of data can easily be imported into spreadsheet applications such as Microsoft Excel, TIBCO Spotfire, and Tableau. The user can choose from a set of visualizations to depict the desired data, including pie, 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 within the visualizations provided by both tools.

Using spreadsheet applications is common for gaze analysis (e.g., plotting fixations [AHR98] and statistical analysis [SAC<sup>+</sup>07]). The provided visualizations are first and foremost one- (e.g., scattergrams) and two-dimensional (e.g., pie charts) diagrams. Besides having a difficulty to represent context-dependent data (e.g., the relationship between gaze coordinates and displayed items), these tools lack video and audio support.

An example from Spotfire is shown in Figure 4 where a scattergram for football players is displayed, with each circle representing one football player in a certain team. Details are provided when hovering over a particular circle with the mouse pointer or when clicking on it. Additional information on selected items are presented in the lower window.

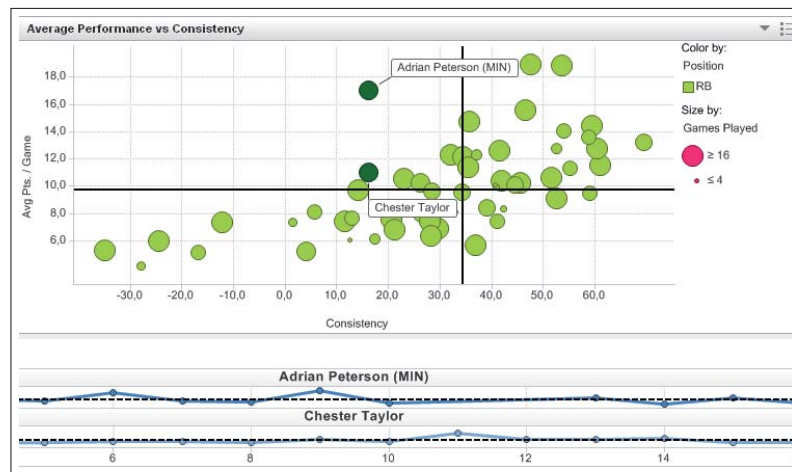


Figure 4. Screenshot from Spotfire's online sample application (<http://spotfire.tibco.com/>).

### 3.2.2 Tools for Behavioral Studies

Noldus and Mangold provide software suites for the acquisition, analysis, and presentation of video, audio, and sensor data (including gaze data) from behavioral studies. Multiple video views and the ability to assign event markers linking to the video are implemented. The integrated eye tracking visualizations only incorporate static graphs superimposed on static stimuli representations<sup>1</sup>. Common gaze visualizations like attentional maps (see Section 3.3 for more details) are included.

Compared to Spotfire and Tableau, the visualizations are not as sophisticated and easy-to-use. Behavioral and physiological data can be visualized in (static) plots. In contrast to Spotfire, which provides the possibility to dynamically filter elements within the visualization, the user has to create an additional subset profile in Noldus Observer XT, for example.

### 3.2.3 Specialized Gaze Data Analysis Tools

Various tools for the analysis of gaze data exist. Such tools are usually designed for particular hardware devices, as for example by Tobii Technologies and SensoMotoric Instruments (SMI). Since features are similar, the following five gaze analy-

<sup>1</sup> This information has been provided by sale representatives from Mangold and the official website from Noldus.



Although Tobii Studio is advertised as “ideal for evaluating interactive media such as websites, [...], computer games, [...]”<sup>2</sup>, Tobii Studio and the other mentioned tools do not contain specialized visualization techniques for 3D VEs.

In summary, ten analysis tools differing in their degree of specialization were presented. Spreadsheet applications are convenient for managing data represented in charts, yet lack support of video and audio recordings. However, analyzing session recordings are a prevailing method for behavioral studies deploying dynamic stimuli in particular. Drawbacks of the reviewed behavioral analysis tools are limited flexibility and diversity of gaze visualizations. The gaze analysis tools provide both video support and gaze visualizations. Gaze visualizations are addressed in more detail in the following section.

### 3.3 GAZE VISUALIZATIONS

Various visualization techniques exist for depicting eye tracking data. Most gaze visualizations focus on illustrating projected screen positions to investigate what attracts visual attention and in what way it catches the attention. Pupillary dilation is used for assessing mental workload [Bea82, GPMSo4, IZBo4, BIo8] and is usually visualized in timelines [BBIo7, KKHo8].

For the description of gaze representations the input space has to be clearly defined. This includes data types of interest and a hypothesis or research question, which the visualizations shall help to answer. In the following classification gaze visualizations are presented to gain insights into the distribution of subjects’ visual attention. Therefore, data types of interest are gaze positions and corresponding timestamps. Details concerning suitable stimuli are closely related to the respective visualization techniques and thus have to be individually addressed.

#### 3.3.1 *One-dimensional*

Sequentially arranged textual documents, mainly including log files, are regarded as linear data types [Shn96]. Design

---

<sup>2</sup> The entire advertisement can be found on Tobii’s official website <http://www.tobii.com/>.

issues include the selection of appropriate font attributes (for instance color and size). Thus, one-dimensional visualizations include the illustration of log files as a collection of text lines. For this purpose, spreadsheet applications (see Section 3.2.1, page 25) are usually used.

### 3.3.2 Temporal

Temporal visualizations map data against time. Common temporal representations are timelines, often considered a subcategory of one-dimensional visualizations, where items have specific start and finish times and may overlap [Shn96].

Timelines are not limited to a certain stimulus type. In this respect, the independence from stimuli provides high applicability, but may impede a simple interpretation. Additionally, multiple trials can be compared within one graph. However, arranging multiple plots has to be handled with care. Displaying various trials at once may hinder easy understanding of a graph, due to stacked plots and therefore hidden information. Visibility and color adjustments can help to solve this problem. Keogh et al., for example, use *timeboxes* for directly manipulating time series to filter data of interest [KHS02]. Desired data can be selected via *timeboxes* and individually displayed in vertically arranged distinct layers.

Lessing and Linge present a temporal visualization depicting *Areas of Interest* (AOIs) [LL02]. It provides information about what, in what order, and for how long a subject has been looking at certain locations. As shown in Figure 6, layers

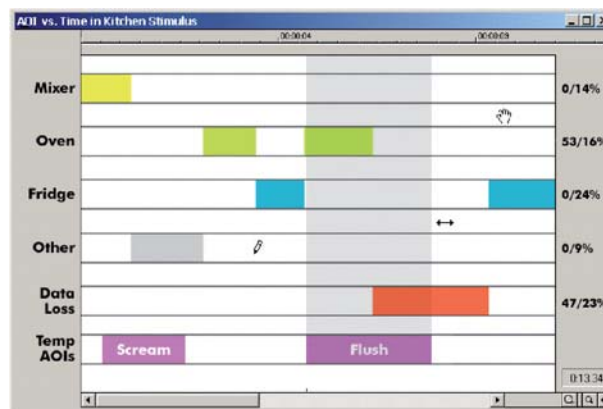


Figure 6. AOIs vs. time graph by Lessing and Linge [LL02].

for additional events to describe subjects' behavior can be added. Obviously, this technique is most suitable for stimuli in which **AOIs** can be defined, which is usually the case for static images and web sites (for further information see Section 3.3.3, page 35). A disadvantage of this technique is that with an increasing number and decreasing size of **AOIs** it becomes more and more difficult to obtain a suitable understanding. Further groupings of small **AOIs** may help avoid this problem. In addition, it seems that this technique is designed for displaying data from only one subject at a time. Thus, multiple trials cannot be compared in one representation.

### 3.3.3 *Two-dimensional*

Two-dimensional (**2D**) gaze visualizations are the most commonly used techniques for visually representing gaze data. Typical **2D** gaze visualizations comprise the depiction of gaze positions (scan path), aggregated representations of visual attention including average dwell times (attentional maps), and the illustration of specified **AOIs**.

#### *Scan paths*

A common approach for representing eye tracking data is the creation of scan path visualizations superimposed on a stimulus image. Scan paths depict the order of eye movements by drawing connected lines (saccades) between adjacent gaze positions (fixations).

Scan paths are appropriate for the analysis of all identified stimuli types (see Figure 3, page 23). However, dynamic and unique stimuli imply a frame-by-frame analysis, since each video frame is associated with certain gaze positions. This representation often incorporates preceding gaze positions.

Since **3D** data (gaze position and time) are mapped onto **2D** screens, scan paths may be hard to spot and follow due to overlapping graphics (i.e., parts of the scanpath may be occluded, see Figure 7(a)). This makes it difficult to find out in which order certain items have been looked at. One approach for solving this issue is presented by R ih a et al., who conducted a study on visual scannings of web search result listings [RAM<sup>+</sup>05]. A non-overlapping scan path is used for gaze analysis, in which the horizontal values from gaze po-

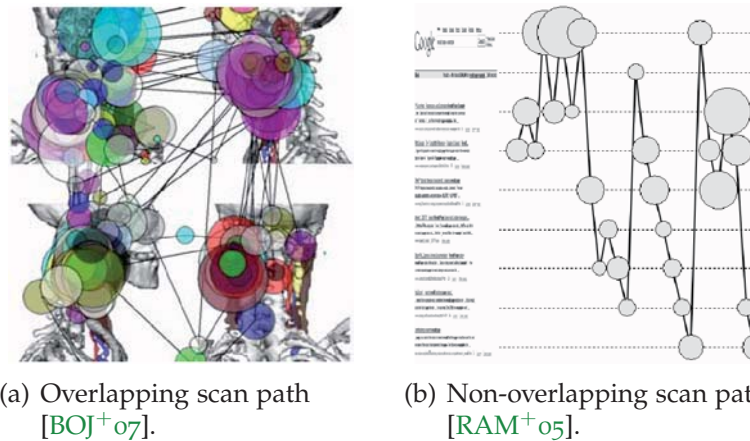


Figure 7. Two examples of scan path visualizations are shown.

sitions are disregarded. This results in a graph where the vertical values are mapped against time (see Figure 7(b)). Another approach for an improved overview is data containment by confining the research question. Mello-Thoms et al., for example, analyze gaze behavior for improved cancer detection by just displaying fixations [MTNK02]. In addition, depicted data can usually be reduced by defining time intervals of interest. A third possibility is the observation of data subsets from shorter time segments.

Replaying scan paths is very common and considered dynamic in that manner. It is possible to display gaze data from several subjects. In case of an *identical dynamic stimulus* such as videos, gaze data from multiple observers (representable by different colors) can be mapped to corresponding video frames. Tobii Studio’s *Bee Swarm* implements this technique [Š08]. In addition, Tobii Studio integrates fixation tool tips [Tob08] to provide details-on-demand for each fixation, including the fixation duration and precise position.

Ramloll et al. present an approach for replacing tedious frame-by-frame gaze analysis of dynamic 3D non-stereoscopic content [RTSB04]. A fixation net is created by translating gaze positions to the flattened object resulting in a 2D representation (see Figure 8). In addition, they present an example for using interactive scan paths that include filtering features (e.g., transparency and zooming level).

Two categories of scan paths can be distinguished based on the degree of preprocessing:

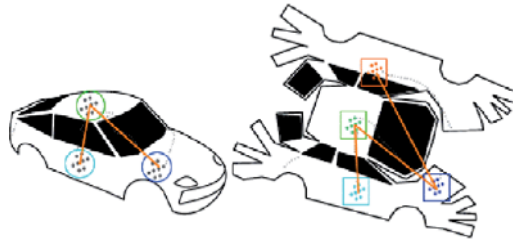


Figure 8. Gaze visualization on flattened object [RTSBo4].

- *Pure scan paths* depict raw gaze positions
- *Fixation and saccade plots* imply data clustering

*Pure scan paths* are probably the oldest gaze visualization technique, which was first described by Javal in 1879 [Hue68]. Extensive studies utilizing pure scan paths have been conducted by Yarbus during the 1950s [Yar67] (see Figure 9). Due to low processing effort, this gaze visualization technique is often considered one of the easiest to implement and is still widely used by researchers today [Šo8]. The main disadvantage of pure scan paths is the lack of clarity about how long a certain position has been looked at.

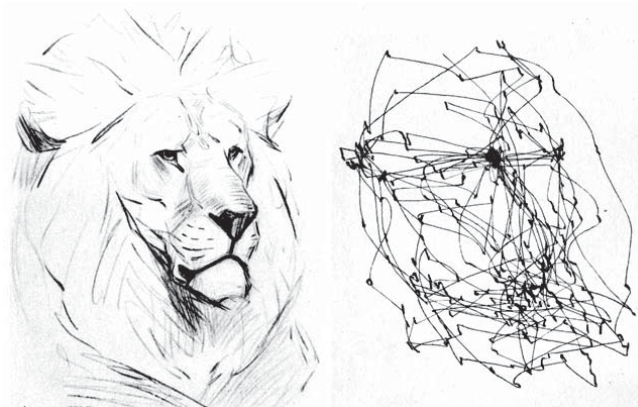


Figure 9. Example for pure scan paths from free examination of the picture for 2 minutes [Yar67].

*Fixation and saccade plots* (see Figure 10) imply data clustering by grouping information into meaningful chunks (fixations). This provides more clarity about how often or for how long (dwell times) a certain location has been looked at. Thus, Rähkä et al. [RAM<sup>+</sup>05] have noted that this is more useful than

displaying individual data points (pure scan paths). While fixations indicate areas attracting the observer's attention, saccades provide information about how fixations are related to each other. Fixations are usually displayed as circles varying in size depending on the fixation duration. However, surveys on using different shapes have also been conducted [LL02]. Saccades are commonly depicted as straight lines.



Figure 10. Scan path in Tobii Studio (<http://www.tobii.com/>).

### *Attentional Maps*

An attentional map (also commonly referred to as heat map or attentional landscape [VH96]) is an aggregated representation, depicting areas of visual attention (see Figure 11). Thus, it constitutes a probability graph describing what content is most likely to be looked at. Areas receiving no interest can be discovered as well.

Attentional maps are convenient for gaining a general idea of the overall distribution of visual attention for primarily static 2D stimuli. This is substantiated by the fact that an attentional landscape usually has the same dimensions (width and height) as the underlying stimulus [Woo02]. Each pixel in the attentional map is assigned a value for describing its degree of visual attraction over a certain period of time. Creating a 2D graph representing 3D gaze positions would lead to data loss.



Figure 11. Heat map in BeGaze [SMI08].

Moreover, attentional maps are well-suited for representing multiple trials. This benefited Wooding, who used this technique for the evaluation of an eye tracking experiment involving more than 5,500 subjects [Woo02]. An example from Wooding's studies is presented in Figure 12.

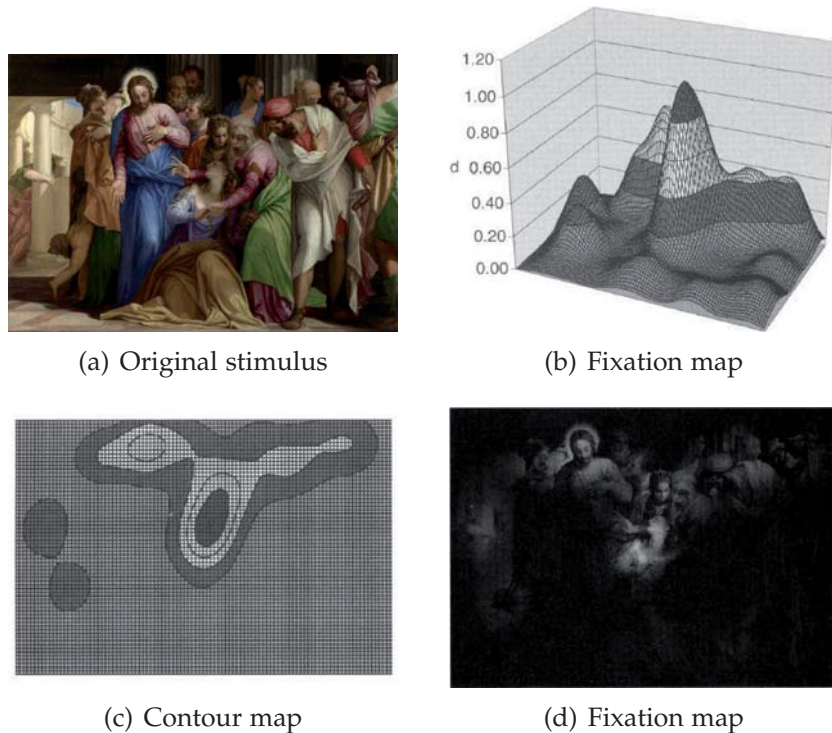


Figure 12. Example from an eye tracking study by Wooding [Woo02].

Since the number of views at a particular gaze position are merely accumulated, attentional maps do not provide any indications about the viewing order or specific fixations. This means that we cannot find out from attentional maps whether a certain location has been looked at once for a longer period of time or frequently but for shorter durations. The time component is disregarded, complicating the assessment of how long a subject actually has been looking at a certain location.

Different types of attentional maps exist. Wooding presents fixation maps actually illustrating a 3D plot from which a 2D contour map can be derived (see Figure 12). A contour map ranging from red (high visual attention) to green or transparent is often described as a heat map. However, depending on

the assigned color and transparency spectrum alternatives can be characterized. In this regard, NYAN 2.0<sup>XT</sup> adopts its so-called *Sinn Builder* including sinnbild, white filter, blur filter, and heat map (see Figure 13).

An example for the customization of attentional maps is presented by Lessing and Linge [LL02]. A property menu is provided for adapting the landscape type, opacity, and time interval. Schießl et al. use a combination of different attentional foci by distinguishing between the gaze behavior of male and female participants [SDTF03].

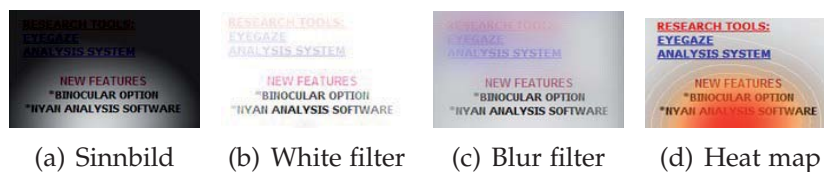


Figure 13. Attentional map variations as implemented by NYAN 2.0<sup>XT</sup> [in09].

### *Areas of Interest*

Another common gaze visualization and analysis technique is the definition of areas of interest (AOIs), which are 2D regions clustering gaze positions for one or more subjects. AOIs have been successfully used for the analysis, segmentation, and interpretation of static 2D stimuli [RAM<sup>+</sup>05, SNo8]. Either regions are manually defined by drawing customized shapes [LL02, RAM<sup>+</sup>05] or dynamic AOIs can be retrieved depending on the degree of visual attention [Woo02] and distinct decision rules [NH07]. Thereby, the manual definition of AOIs is highly subjective [LL02]. Approaches also have been undertaken for dynamically adjusting AOIs to follow moving objects (for instance, implemented by GazeTracker as so-called “Look Zones”).

### 3.3.4 *Three-dimensional*

While 3D visualizations provide the advantage of an additional dimension and a higher degree of freedom, they also cause problems of occlusion. Some techniques proposed by Shneiderman’s *Information Seeking Mantra* [Shn96] assist in re-

solving this issue, for example, by applying overview and filtering methods.

Three-dimensional gaze visualizations are still rather uncommon compared to the previously mentioned 2D techniques. An example of a visualization based on gaze position and depth in a 3D scene (on a 2D display) is introduced by Duchowski et al. [DSR<sup>+</sup>00, DMC<sup>+</sup>02, SGGD05]. A diagnostic Virtual Reality (VR) system is presented for tracking head and eye movements (using a binocular eye tracker). It measures user performance during scenario execution in a VR aircraft inspection simulator. The used gaze visualizations include *gaze depth paths*, which are scan paths with an additional depth cue (see Figure 14).

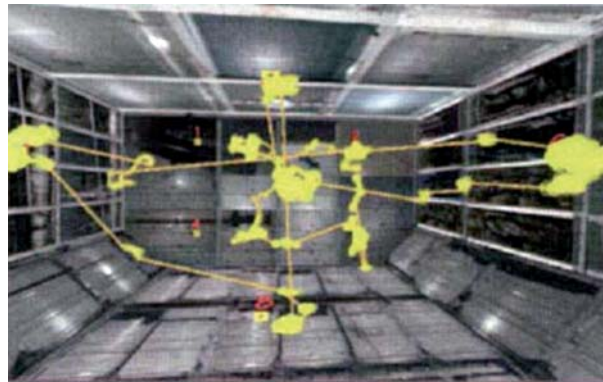


Figure 14. Gaze depth paths within a VR training scenario [DMC<sup>+</sup>02].

In addition, Duchowski introduces the term *Volumes of Interest (VOIs)*, representing 2D screen positions that are translated into 3D space-time over time [DM98] (see Figure 15). For this purpose, AOIs from consecutive video frames are merged to visualize foveal VOIs.

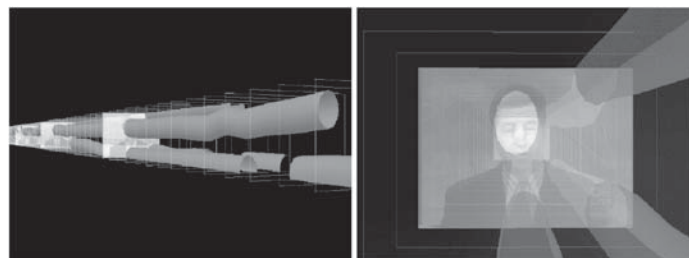


Figure 15. VOIs as described by Duchowski [DM98].

Fixation maps [Woo02], which have been described before, can also be considered 3D gaze visualizations (see Figure 12(b)). Babcock et al. use fixation maps to compare visual attention on observed images with different given evaluation tasks [BPF03]. For this purpose a stimulus representation is projected onto the horizontal plane (see Figure 16(a)). A similar approach is used by Burgert et al. [BOJ<sup>+</sup>07], displaying the stimulus image on the back plane, however, causing overlaps (see Figure 16(b)).

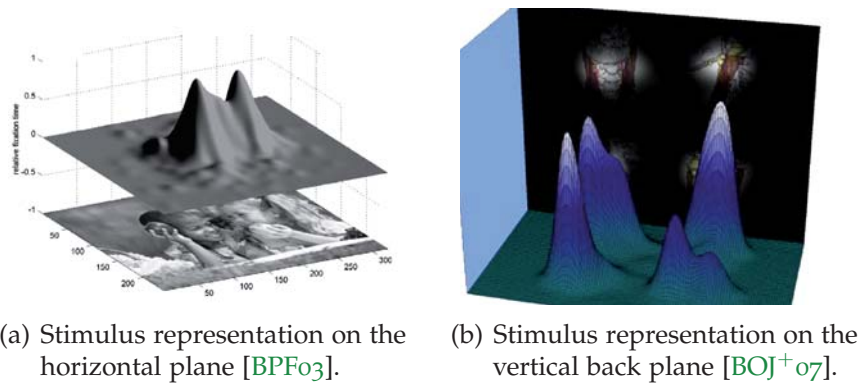


Figure 16. Superimposed fixation maps.

### 3.3.5 Multi-dimensional

The visualization of diversified data sets demands using additional dimensions besides spatial ones. This includes among other things color, transparency and form coding. Fixation and saccade plots, although considered 2D in general, can also be classified among multi-dimensional visualizations. For example, a fixation's radius provides an additional dimension by indicating the size of a fixation instead of merely identifying its position. Therefore, multi-dimensional visualizations are not considered as a stand-alone category in the scope of this thesis.

### 3.3.6 Tree

Tree visualizations are particularly suitable for depicting hierarchical data structures. Each element has one corresponding parent node (except for the root node) and any number of

children. So far, tree visualizations are uncommon in visual gaze analysis. The reason for this might be that gaze data are usually not arranged in hierarchical groupings and therefore cannot be displayed as such. However, with respect to **AOIs** it would be feasible to define different discrimination levels (or levels of detail). For example, if the headline in Figure 10 (page 33) is defined as one **AOI**, we could further distinguish different symbols. Thus, hierarchical visualizations for visual gaze analysis are feasible, but so far specialized tree visualizations of gaze data have not been published.

### 3.3.7 Network

Network visualizations are convenient for conveying relationships between items [Shn96]. Lessing and Linge [LLo2] use transitional graphs to depict the relationship between and the percentage of fixations in **AOIs**. Thereby, two types can be distinguished: transitions superimposed on the stimulus (see Figure 17(a)) and stand-alone diagrams (see Figure 17(b)). Stand-alone visualizations have the advantage of being stimulus-independent, whereas superimposed graphs require some sort of static stimulus representation. On the other hand, superimposed graphs may be better for comprehending how **AOIs** are related to each other by seeing the stimulus. Transitions can integrate data from multiple subjects.

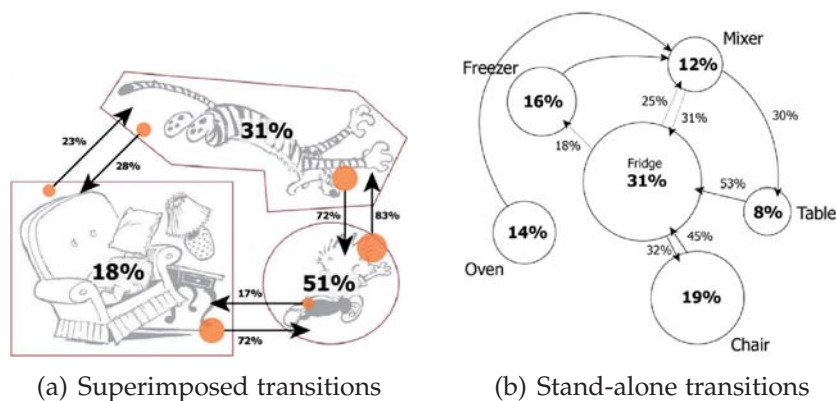


Figure 17. Transitional graphs by Lessing and Linge [LLo2].

### 3.4 CONCLUSION

A taxonomy for advanced descriptions of gaze visualizations has been introduced. With regard to the proposed gaze taxonomy, common gaze visualizations have been presented. The taxonomy include three components: input space, mapping and output space. An overview of the taxonomy can be found in Figure 2 (page 21).

Concluding from the examined gaze visualizations, data of interest include time stamps, gaze positions, and gaze dilations. Possible stimuli have been roughly classified into three categories: static identical, dynamic identical, and unique stimuli (see Figure 3, page 23). Most visualizations apply well to 2D stimuli, which is shown by their high applicability to 2D stimuli (see Figure 18). Data from dynamic or unique stimuli usually have to be examined with a frame-by-frame analysis of recorded session videos. An exception is the *Gaze depth* path, which can be used for 3D VE scenarios.

Mapping includes managing user queries. Although visualizations can be customized to a certain degree (for example, by assigning desired colors) they are usually not explorative in an interactive manner. Details-on-demand are, however, a preferable technique to provide further insights to specific data of interest. Data processing typically includes clustering gaze positions as fixations and performing calculations for AOIs. This processing can be done in real-time, allowing researchers to interactively explore the data.

Visualizations resulting from specific inputs and mapping techniques have been introduced according to their underlying data types. In general, stimulus-independent visualizations provide high applicability, but meanwhile may impede data interpretation. Such visualizations included (stand-alone) transitional, one-dimensional, and temporal representations. Superimposed graphs on the other hand enable a better understanding of spatial relationships. In this regard, several 2D and 3D visualizations have been discussed.

No examples for the application of tree visualizations for gaze analysis have been found. Multi-dimensional visualizations have not been examined as a stand-alone category since most visualizations can be enhanced by representing additional information (e.g., by assigning color and transparency values).

|               |                  | 2D       |         |        | 3D     |         |             |
|---------------|------------------|----------|---------|--------|--------|---------|-------------|
|               |                  | Stimulus |         |        |        |         |             |
|               |                  | Static   | Dynamic | Unique | Static | Dynamic | Interactive |
| 1D / Temporal | Time line        | ✓        | ✓       | ✓      | ✓      | ✓       | ✓           |
|               | Scan paths       | ✓        | ○       | ○      | ✗      | ✗       | ✗           |
| 2D            | Attentional maps | ✓        | ✗       | ✗      | ○      | ✗       | ✗           |
|               | AOIs             | ✓        | ○       | ○      | ✗      | ✗       | ✗           |
| 3D            | Gaze depth       | ✗        | ✗       | ✗      | ✓      | ○       | ○           |
|               | VOIs             | ✓        | ✓       | ✗      | ✗      | ✗       | ✗           |
| Network       | Transitions      | ✓        | ○       | ○      | ○      | ○       | ○           |
| Tree          | Hierarchies      | ?        | ?       | ?      | ?      | ?       | ?           |

✓ Feasible      ○ Restricted utility      ✗ Inappropriate      ? No information

Figure 18. Overview of the applicability of the presented gaze visualizations to different stimuli. Users are assumed to be able to change their viewpoints.

An overview of the applicability of the presented visualizations to screen-based 2D and 3D stimuli is presented in Figure 18. The dynamics *static*, *dynamic*, and *interactive* relate to the behavior of the stimulus. Concluding from this comparison, there is a profound lack of enhanced gaze visualizations for 3D stimuli. This further encourages the aim to develop novel visualizations for managing simplified gaze analysis within digital 3D environments.

## CONCEPT FOR ENHANCED VISUAL GAZE ANALYSIS TECHNIQUES

---

Based on the findings from the initial survey (see Section 2.4) and from the classification of gaze visualizations (see Chapter 3) eye tracking analysis techniques for the application in *three-dimensional (3D) virtual environments (VEs)* are devised in this chapter. This includes the development of enhanced gaze visualizations, as well as the design of a coherent application context. However, before developing such concepts we need to consider requirements and constraints for such techniques. For this purpose, a requirements analysis is carried out in Section 4.1. Based on this investigation and conclusions from Chapter 3, concepts for novel gaze visualizations are presented in Section 4.2. A discussion about how to integrate these visualizations in a coherent application environment are done in Section 4.3. This includes the description of required tasks and features for improved system usability.

### 4.1 REQUIREMENTS ANALYSIS

The requirements analysis provides the basis for the development of novel gaze visualizations and a corresponding system to integrate them. In order to design enhanced techniques, the intended context of use has to be specified. For this purpose, personas are described in Section 4.1.1 to explore future applications. Based on the described personas specifications for the anticipated input space are derived in Section 4.1.2. This includes suitable stimuli and, according to research questions, required data types. Further requirements that need to be considered for the intended system design are described in Section 4.1.3.

#### 4.1.1 *Scenarios using Personas*

Defining user scenarios is a common technique to explore future applications and design solutions from an early stage in the design process [Nieto4]. A special way of doing this is

to use fictional characters called personas to represent distinct user types (introduced by Cooper [Co09]) in a scenario. It is important to provide a detailed description of the imaginary user so that the reader can establish a personal relationship with otherwise abstract customer data [PA06]. This technique helps to determine necessary features and interactions by regarding a persona's situation and goals. In the following paragraphs, three personas are introduced to describe their respective desires for novel gaze visualizations better.

### *Virtual 3D Shopping with Emma-Zone*

Our first persona is young Penelope Widmore, who works in the marketing department of *Emma-Zone*, a rather moderate successful online retailer offering a virtual 3D shopping experience. Penelope loves the possibility to reach out to people all over the world and offer items just as she could do in a local store. But then one gloomy day, her boss delivers the message that due to decreasing sales they would have to shut down the website. Young and eager Ms. Widmore cannot resign herself to let that happen and thinks about how to increase sales revenues. She goes on the internet, looking for helpful clues, and comes across eye tracking studies that aid in increasing usability of websites. Immediately she plans a study for *Emma-Zone* with 50 customers to find out more about their buying behavior and how they would wander through her virtual store. With the new visualization techniques Penelope is able to conduct and evaluate her studies within a few hours. The gained insights are used for improved and customized product placement. Penelope goes to her boss, presents her findings, and is eventually able to triple sales in no time.

### *Company Advertisements in Virtual Communities*

Sheldon Cooper is often described as a serious and highly ambitious person. He recently founded a company *GeeksInTheory* with his friends Leonard and Howard. The other night he awakes all of a sudden with a flash of inspiration to promote their company in their favorite MMORPG (Massively multi-player on-line role-playing game). Leonard and Howard love his idea. However, when Sheldon goes on about extensive user studies and proposes a work plan that would keep everybody busy for the next six months to analyze the recorded data,

Leonard and Howard take off to look for a simpler solution. An hour, a few mouse clicks, and a little chitchat with their neighbor later, they present Sheldon with the novel gaze visualizations. Using these techniques enables them to quickly correlate data from various participants and to get an idea where people are most likely to look at within the game world. Based on these findings they can place the advertisement for *GeeksInTheory* much more effectively.

#### *Evaluation of Product Designs*

Susan Mayer works at the product design department of *Norniel Group* and focuses on the development of new products. The *Norniel Group* pursues a user-centered approach by integrating periodic feedback from test customers throughout the development process. For this reason, Susan prepares several design alternatives to present to the customers in each phase. Besides simply asking which design the participants favor most the *Norniel Group* has an eye tracker to find out whether customers would actually look at those parts of the models that were assumed to be focus parts. One day, when Susan presented a new design for a personal digital assistant (PDA) she noticed that participants were distracted from the display and instead were looking at the outline of the PDA again and again. Surprisingly, when questioned about this issue most participants were not even aware that they had done this. But after thinking about it they mentioned that the PDA's edges seemed too beveled. That way, Susan was able to detect design issues which otherwise may have remained unnoticed.

#### 4.1.2 *Input Space Specification*

The aforementioned personas will help to specify the input space which defines the initial conditions for the development of visualizations. As described in Section 3.1.1 a comprehensive description includes specifications of *stimuli* and *research questions*. In addition, *required data types* need to be specified. An *eye tracking procedure* for 3D VEs is introduced to provide desired data.

### *Stimulus Description*

Based on the described personas, gaze visualizations are used for diagnostic eye tracking studies in 3D VEs. Reports about the application of gaze visualizations within such contexts are very limited (except for 3D scan paths, see Section 3.3.4). This means that well-established methods for representing gaze data in such environments are lacking. Therefore, a foundation of effective and efficient techniques has to be established. For this reason supported stimuli by the herein presented system are limited to static 3D VEs, that omit transparent phenomena, such as fog or smoke. Users can freely explore the 3D scenes that contain a variety of different static models by moving their viewpoints via mouse and keyboard controls.

Transparency can be a problem for identifying viewed objects. If transparent objects are aligned behind each other, either a real-time decision algorithm is needed to log only actual viewed items (which is ambiguous) or all intersecting objects. The latter is probably the better solution for sophisticated post-analysis. However, both solutions may reduce system performance drastically. Thus, tracking transparent phenomena, such as fog or smoke, will not be supported by the herein presented system.

Another important issue is the scene structure. A scene can be described by a scene graph that arranges the scene's spatial and logical representation. Each node in this structure represents objects in the scene. Logical relationships, for example, between a *table* and *items* placed on it can be established. This results in a scene graph containing a *table* node with attached *item* nodes. This is useful for applying transformations or other actions to object hierarchies. For example, if the *table* is moved, the *items* will change their locations as well. Although supported stimuli only include static scenarios scene graphs may still be of assistance since they can also be used to describe different levels of detail (LODs) by defining semantic hierarchies. Penelope Widmore, for example, might be interested in finding out how often a certain shelf (containing a number of different items) has been looked at. So, instead of listing precisely which products have been looked at gaze data from certain objects could be combined. For this purpose, scene descriptions for analysis should include scene graphs that specify semantic relationships between models.

After all, supported stimuli include static 3D VEs without

transparent phenomena. Scenes are described by a hierarchical scene graph, which can be used to provide different LODs. The stimuli can be described in the taxonomical group “unique user behavior, static stimulus context” (see Figure 3, page 23).

### *Research Question*

Research questions influence the choice of visualization techniques. Based on the introduced personas we can observe that each persona conducts a study for various reasons. Depending on the specific purpose of a study, certain visualizations are more suitable than others. As we have learned in Section 3.3, for example, aggregated representations provide insights into overall gaze distributions and dwell times. Questions about gaze sequences, however, cannot be answered by this technique. In this case, scan paths are more suited.

Instead of having a specific research question Jacob and Karn note that usability researchers do not always have a strong theory driving their studies [JK03]. Instead, observers may notice that users take longer to perform a certain task than anticipated and are looking for reasons causing this. Susan Mayer might observe that her product design receives high visual attention in unexpected areas and she wants to find out why. In this respect, Goldberg et al. state that eye tracking data analysis usually proceeds either top-down or bottom-up [GSL<sup>+</sup>02]. A top-down approach is built on cognitive theories or design hypotheses, whereas bottom-up analysis advances without predefined theories.

In conclusion, the novel gaze visualizations intend to facilitate eye tracking analysis in 3D VEs for both top-down and bottom-up approaches. The visualizations focus on the representation of gaze behavior in contrast to pupil size. Further details about questions which visualizations can help to answer are discussed in Section 4.2 when addressing the individual visualization techniques.

### *Required Data Types*

As we have learned from the previous section reasons for conducting eye tracking studies may differ as do the required data types for those studies. It is, however, common to log more data than actually studied. This provides the possibility to investigate disregarded data later on, if desired. Penelope,

| <i>Properties</i>                    | Log file types |      |        |        |
|--------------------------------------|----------------|------|--------|--------|
|                                      | Tobii          | Gaze | Object | Camera |
| Counter                              | X              | X    | X      | X      |
| Time stamp                           | X              | X    | X      | X      |
| 2D gaze position                     | X              |      |        |        |
| 3D gaze position                     |                | X    |        |        |
| Object ID                            |                |      | X      |        |
| Viewed triangle                      |                |      | X      |        |
| Camera position                      |                |      |        | X      |
| Viewing direction                    |                |      |        | X      |
| Distance from camera to intersection |                |      |        | X      |

Table 4. Four log file types are distinguished containing the required data types.

Sheldon, and Susan are all interested in the distribution of visual attention in 3D VEs. Moreover, for Susan’s evaluation of product designs, it is essential to find out which details of a product have been observed more closely. Therefore, 3D gaze coordinates, viewed models, and mesh triangles need to be determined based on gaze target positions received from the Tobii Application Programming Interface (API) (see Section 2.2.1, page 9). For this purpose, a special eye tracking procedure is needed to deliver these data and is described in Section 4.1.2.

In addition to gaze data logging it is also essential to record camera properties because of the variable viewpoint of a user. This includes camera positions and viewing directions. All in all, three additional log files have to be created by the new system to collect the specified data types: *gaze*, *object*, and *camera* log files. Data stored in these files are listed in Table 4.

## Eye Tracking in Virtual Environments

In order to log the specified data types an adapted eye tracking procedure is described in this section. The description is based on the assumption that the Tobii ET-1750 (see Section 2.2.1) is used. Since this is a binocular eye tracker there are different ways to proceed:

1. Determine gaze depths [DSR<sup>+</sup>00, DMG<sup>+</sup>01]
2. Calculate the intersection of a gaze ray with virtual objects [SAC<sup>+</sup>07, Ste07]

While the first technique is object-independent the latter approach provides precise information about which objects have been looked at. Since required data types include viewed objects the latter approach will be used for this thesis. Figure 19 illustrates the tracking procedure along with required data types. The logging of viewed objects is described by Sennersten et al. [SAC<sup>+</sup>07] and Stellmach [Ste07] (see page 14), who compute gaze ray intersections with models' bounding volumes. Each model is characterized by a unique identification (*Object ID*). In contrast, the approach presented here works with triangle accuracy, which means that intersections with the triangle mesh itself are calculated. The gaze ray intersection with a viewed model is determined and the corresponding triangle is identified (see Figure 19). As a result, the higher the polygon count of a viewed model is the more precisely a logged triangle can represent the viewed location. Let us assume, for example, that a virtual cuboid room is defined.

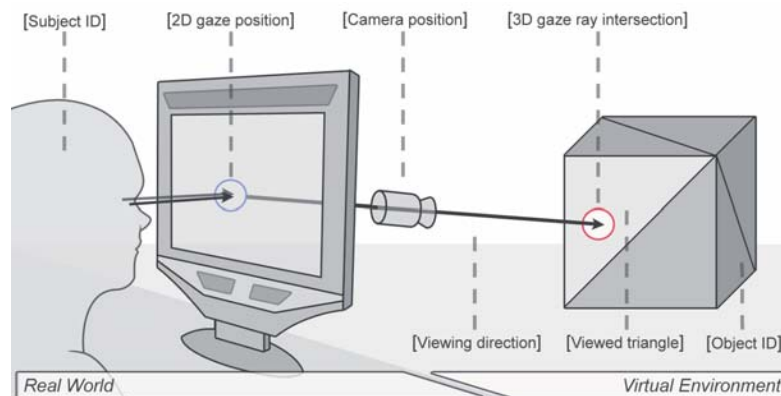


Figure 19. Gaze ray from the viewer to the screen and into the VE.

Each side is only defined by two triangles. Although the particular side that is looked at can be determined precisely, the logged triangle representation is rather imprecise. We cannot determine from the viewed triangle whether the observer looked at the top or bottom of a certain wall. While this is important for people such as Susan Mayer who are interested in detailed representations of visual attention scattered over a model's surface, this may not be essential for others (e.g., Penelope Widmore) who may strive for more general answers (e.g., which products have been looked at).

Compared to other implementations the presented approach has several advantages. On the one hand intersections with bounding boxes introduce a precision error. Although a user could be looking over the shoulder of a *Non-Player Character* (NPC) to a distant item, a gaze ray intersects with the NPC's bounding box first. Thus, the log file would inadequately indicate that the user has looked at an NPC instead of the targeted item. In this respect, the triangle-precise calculation decreases the number of occurrences of this type of error, since the gaze ray directly intersects with a model's surface. However, general errors due to spatial inaccuracy of the eye tracking instrumentation and spatial ambiguity remain (see Section 2.2.1, page 9).

The triangle-precise approach is beneficial to gain detailed information about where a user has been looking exactly. On the other hand due to higher complexity triangle-precise tracking may lead to decreased performance. It could be necessary that every triangle from a model has to be checked to determine which triangle was hit by the gaze ray. This may cause critical performance problems if high-resolution models are used that contain millions of triangles.

All in all, both approaches, the intersection with bounding volumes and with triangle meshes, have distinct advantages. Depending on research questions the analyst has to decide which approach is most suitable. For the purpose of this thesis the triangle-precise tracking is used to be able to visualize detailed gaze behavior in 3D VEs. The following logging procedure of viewed items and mesh triangles in VEs will be used:

1. Receive 2D screen-based gaze positions from the eye tracker

2. Determine 3D collision ray based on the 2D gaze position, camera projection, and view
3. Locate the intersection of a gaze ray with objects triangle-precisely
4. Write necessary data to log files

The particular data types which need to be logged have been specified in Section 4.1.2. Further details about individual processing steps are explained in Chapter 5 where the system's implementation is addressed. For the logging of gaze data several limitations have to be considered, which are described in the following paragraph.

#### *Eye Tracking Limitations*

Based on the aforementioned tracking procedure general constraints have to be considered. Sennersten et al. report problems for eye tracking in a 3D VE (a First-Person Shooter) [SAC<sup>+</sup>07]. One of the most challenging aspects which also seems relevant for the here described tracking procedure is spatial ambiguity. Problems of spatial ambiguity may result from small model sizes, closely arranged objects, and high virtual distances between a gaze target and a user's viewpoint. The error margin can be decreased by confined VEs. A scene could, for example, be limited by walls to minimize the maximal distance between a user's viewpoint to virtual objects in sight. Moreover, model locations and sizes could be specified with respect to adequate spacing. Due to inaccuracies of the eye tracking system (see page 9) the overall accuracy of the tracking procedure in context with respective scenarios should be investigated in future verification studies. However, due to timely constraints, this thesis does not include such a study.

#### 4.1.3 *Further Requirements*

By specifying the anticipated input space the basic conditions for novel gaze visualizations have been established. However, further requirements for the intended system need to be discussed, which result from the survey about improved visual eye tracking analysis (see Section 2.4, page 16) and the introduced personas. A coherent context for future application of the to-be-developed visualizations has to be designed. The

tool shall integrate 3D VEs, in which analysts can explore logged data. The analyst should be able to load different scenarios and corresponding log files. Also, being able to load data from various participants belonging to a certain experiment is desired. This facilitates comparison between different subjects.

The interviewed eye tracking professionals stated some additional features to improve eye tracking analysis. Several respondents requested multiple views to look at the scene from different angles. In this respect, defining customized viewpoints (i.e., save and load particular camera settings) is beneficial. Another interviewee asked for “multiple visualizations and easy-to-follow links between them”. Thus, relationships between various gaze visualizations and viewed objects have to be established. Requested features also include overview and detail methods as well as improved aggregated comparisons in 3D contexts. Moreover, it is desired to devise methods that help in detecting repeating patterns.

The introduced personas have distinct motives for conducting eye tracking studies. On the one hand Susan Mayer desires detailed representations of gaze distribution to evaluate product designs. For this purpose, aggregated gaze visualizations over model surfaces are required. Sheldon Cooper, on the other hand, is interested in game areas, which receive considerable visual attention, to place his advertisement more effectively. Thus, a representation indicating distinct objects or areas of high interest is needed. Finally, Penelope Widmore would like to find out how customers wander through her virtual store and which items have been looked at. Visualizing camera and scan paths as well as the aforementioned representations is necessary.

After all, the mentioned requirements can be met by a coherent system that integrates novel visualizations, specified features (e.g., multiple views of a scene), and also supports tasks such as overview, zoom, filter, and details-on-demand.

In summary, this section carried out a requirements analysis for identifying needs and challenges for the development of techniques for improved gaze analysis system. Different personas have been introduced to explore future applications for the intended system. Supported stimuli and required data types have been specified. An adapted eye tracking procedure

to determine required data in 3D VEs has been discussed. Finally, desired features for a coherent application environment have been mentioned.

## 4.2 DESIGN OF GAZE VISUALIZATIONS

After establishing a set of requirements, the next step is to proceed with the description of the design of novel gaze visualizations. In this section gaze visualizations, some derived from commonly used 2D techniques, are discussed. In Section 4.2.1 a space-filling<sup>1</sup> timeline for displaying viewed objects and for navigating data is introduced. For example, certain parts of scan paths which are described in Section 4.2.2 can be displayed by using the timeline. The advantage of superimposed visualizations, including scan paths and attentional landscapes, is that they provide quick insights about gaze patterns (see Chapter 3.3). This benefit can be used for 3D contexts as well. Therefore, different possibilities to integrate aggregated representations for 3D VEs is discussed in Section 4.2.3.

### 4.2.1 Models of Interest Timeline

Timelines are a means to order events by time and thus facilitate finding events before, after, or during a given time interval. More precisely, MacEachren [Mac95] as well as Müller and Schumann [MS03] have posed several questions which can be answered by using well-designed visualizations for unknown temporal data. Based on these questions, the following problems can be answered with regard to viewed models enlisted in a timeline:

- *Existence of a data element:* Has object  $x$  been observed at time  $t$ ?
- *Temporal location:* When has object  $x$  been looked at? Has object  $x$  been observed repetitively (any cyclic behavior)?
- *Temporal interval:* For how long has object  $x$  been looked at?

---

<sup>1</sup> *Space-filling* visualizations are compact representations that effectively utilize assigned screen real estate [Ahl96] omitting voids.

- *Temporal texture*: How often has object x been looked at?
- *Rate of change*: For how long and how frequently are objects examined?
- *Sequence*: In what order are objects looked at?
- *Synchronization*: Are certain objects usually viewed shortly after one another?

Similar to the *Object vs. Time View* [LLo2] (see Section 3.3.2, page 29), *Model of Interest*<sup>2</sup> (MOI) transitions shall be depicted in an axis-based temporal visualization. In contrast to the *Object vs. Time View*, a space-filling MOI timeline is proposed to represent viewed models over the duration of an experiment. That way, a compact overview of the distribution of visual attention can be given. Each model is represented by a specific color (see Figure 20). Except for using colors to distinguish between represented models, icons or small model previews can be displayed for each timeline entry. If the entry size is large enough to show a preview it will be displayed, otherwise omitted (see Figure 20). The user should have the possibility to change colors and even assign the same value to different objects to manually define semantic groups (e.g., similar looking models or objects placed close to each other).

The MOI timeline should support different tasks, including techniques for overview, zoom, filter, details-on-demand, and

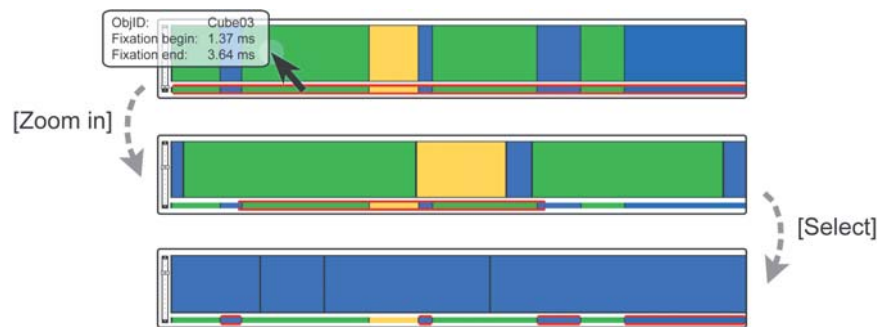


Figure 20. Multiple selections and magnification factors in the MOI timeline.

<sup>2</sup> The term *Volume of Interest* (VOI) is also frequently used for 3D objects in eye tracking studies. However, several studies use the term to describe generic volumes based on gaze data for video resolution degradation [DM98, NH07]. In order not to confuse such generic volumes with distinct 3D models the term *Model of Interest* (MOI) is used throughout this thesis.

data navigation. These tasks are addressed in the following paragraphs.

#### *Overview*

In general, the **MOI** timeline provides an overview about the general gaze distribution (based on viewed models). Based on related work such as the *Simile Timeline* [Sio9] the viewing area is divided into a large detail area showing an extract of the data set and a small overview displaying the entire collection (see Figure 20).

#### *Details-on-Demand*

Details about an object could be displayed when hovering over it with the mouse pointer. Such details include the current time stamp, object **ID**, and description. Moreover, context menus are triggered by right-clicking on items. The context menu includes functionalities such as centering the camera on the respective model in the **3D** view and changing its color.

#### *Zoom and Filter*

The longer an experiment lasts the more data has to be mapped to the timeline. This may lead to data loss due to limited interface dimensions and may result in the need for specific zooming and selection techniques. While the *Simile Timeline* [Sio9] only implements drag interaction, *TimeZoom* [DW06] and *FacetZoom* [DFW08] additionally allow for continuous zooming. Based on these approaches zooming can be accomplished by rotating the mouse wheel or by using the zoom scale on the left of the timeline (see Figure 20). While dragging the scale's slider allows continuous zooming, the + and - buttons enable small-scale incremental zooms.

If the mouse is used for zooming the focal point will be equal to the current mouse pointer location. On the other hand, if the slider is used instead the center of the details screen is applied. Based on aforementioned approaches, the timeline can be dragged to a desired position to offer horizontal scrolling.

#### *Data Navigation*

The **MOI** timeline can also be used to navigate through data. For example, all gaze paths corresponding to one particular **MOI** can be displayed. As a result, an approach for select-

ing various regions is required instead of just selecting one continuous time interval. In this respect, Keogh, Hocheiser and Shneiderman present time boxes for specifying queries on temporal data sets [KHS02]. Based on this approach, users shall be able to select time intervals by drawing rectangles within both views (i.e., detail and overview areas) of the MOI timeline. Moreover it should be possible to do an object-based selection by picking out MOIs in the 3D view. Playback of scan and camera paths is also feasible. For this purpose recorded camera data can be used to automatically set viewpoints. Typical commands such as *Play*, *Pause*, *Stop*, *Speed up*, and *Slow down* should be implemented.

#### *Semantic Hierarchies*

As mentioned before, semantic hierarchies can be established by manually assigning similar colors. In context with scene graphs (see page 44), the possibility to define logical relationships between objects was discussed. If such a scene graph is provided items belonging to a certain group could automatically receive the same color.

Alternatively, the MOI timeline could be extended to a space-filling hierarchy visualization such as treemaps [JS91, Shn92], which show an arbitrarily-sized tree in a fixed rectangular space. Based on such space-filling techniques *TimeZoom* [DW06] and *FacetZoom* [DFW08] implement hierarchical facets that can be browsed with seamless continuous navigation techniques and a quick tap-and-center interaction. These approaches may also apply well for semantic hierarchies of viewed objects.

In summary, an overview of the described properties of the MOI timeline are presented in Table 5. It is important that changes within the MOI timeline will affect the other views as well. This includes color changes and subset selections.

#### 4.2.2 *Three-Dimensional Scan Paths*

Scan paths have already been successfully applied in eye tracking studies in VEs by Duchowski et al. [DSR<sup>+</sup>00, DMC<sup>+</sup>02, SGGD05] (see Section 3.3.4, page 35). While fixations are displayed as circles for 2D stimuli they are represented as spheres in 3D. Saccades are depicted as straight lines. Duchowski et al.

| <i>Property</i>           | <i>Description/Example</i>                                      |
|---------------------------|---|
| Space-filling design      | <i>Compact layout</i>   |
| Overview of viewed models | <i>Quick conclusions about the general gaze distribution</i>    |
| Zooming                   | <i>Zoom in on various items while still maintaining context</i> |
| Multiple selections       | <i>Filter items of interest</i>                                 |
| Details-on-demand         | <i>Display model ID, fixation duration, etc.</i>                |
| Customizable entries      | <i>Change object colors</i>                                     |

Table 5. Properties of the **MOI** timeline.

report this technique only for one specific example: a *Virtual Reality* scenario for aircraft inspection training.

Generic applications, in which scenario sizes may differ drastically, could benefit from adjustable magnification factors of fixations (and saccades). For example, if we look at a large scene from above, we may not be able to detect any fixations at all, because they are too small. Instead, the user would have to zoom in and go through the scenario, which, however, impedes a quick overview about the spatial distribution of gaze points.

One solution is the ability to change line widths (saccades) and sphere diameters (fixations) proportionally. This implies the need to be able to handle dynamic user queries. The dynamic change of a scan path's magnification factor should, however, be independent from the zoom factor in the **3D** environment. This means, while a user zooms out (by rotating the mouse wheel or altering their viewpoints with the cursor keys) the fixation size relative to the screen size may remain the same. This is similar to virtual geographic maps, in which city names remain legible until the magnification factor becomes too small and city labels are removed<sup>3</sup>. Instead of simply removing small fixation representations from a certain point on,

<sup>3</sup> For an application example of this technique in virtual geographic maps see *Google Maps*: [maps.google.com](https://maps.google.com) (last visited April, 2009).

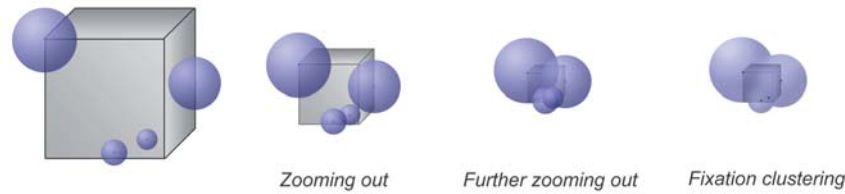


Figure 21. Semantic fixation zooming is illustrated. If a user zooms out, fixation sizes remain the same compared to the screen size. In order to decrease clutter, fixations may also be combined in a clustered representation.

visualizations may also be clustered to provide a more compact view while still offering information about fixations. An example for zooming and clustering of fixations is illustrated in Figure 21.

Displaying the scan path of a complete examination session (or even the data from different subjects) may hinder easy comprehension due to overlapping graphics (as discussed in Section 3.3.3, page 30). Therefore, it is desired to filter fixations (resulting in pure scan paths) or to cluster them. In addition, the ability to select time intervals of interest is essential. For this purpose the aforementioned MOI timeline can be used to navigate through the entire collection. To filter certain data by their corresponding time stamps also provides the possibility to animate visualizations by replaying gaze paths.

As mentioned before, fixations can be represented as spheres in 3D VEs. With an increasing diameter size it becomes more difficult to identify the corresponding gaze point though. This is especially a problem with overlapping spheres. One way to solve the problem is to use reference lines (each parallel to one of the coordinate axes) to facilitate orientation and identification of corresponding gaze points.

Alternatives for displaying fixations are considered and briefly discussed in the following. First of all, it should be examined if 2D representations are sufficient to represent fixations instead of introducing an additional dimension. For this purpose, 3D sprites<sup>4</sup> could be deployed. However, both circles and spheres do not integrate the circumstance that positions may have been observed from different angles. Instead

<sup>4</sup> Three-dimensional sprites are textures mapped to 3D facets whose surface normals always face the camera.

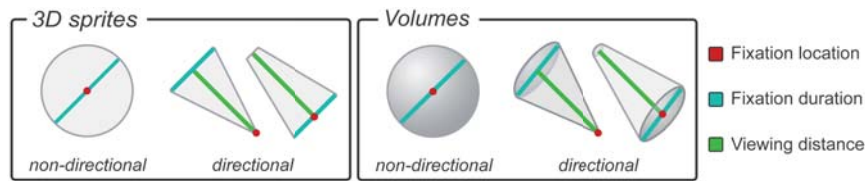


Figure 22. Alternatives for representing fixations in 3D VEs are shown. Besides fixation locations and durations, viewing directions, and distances are incorporated.

of using directionless representations, objects (e.g., cylinders, cones) or sprites (e.g., triangles) whose shapes point at a corresponding gaze position could be used. Moreover, the distance from where an object has been looked at could be integrated. Thus, four variables could be represented by the fixation illustration: *gaze position*, *fixation duration*, *viewing direction*, and *viewing distance*. Figure 22 shows several possibilities of how to map these variables to different shapes. In addition, the user should have the possibility to decide whether to display a counter for each fixation or to represent time via a color gradient. However, this could cause confusion because certain colors are already assigned to particular models and may also represent the intensity of visual attention (see next section).

#### 4.2.3 Aggregated Representations

As presented in Chapter 3.3 superimposed aggregated gaze visualizations work well with gaze positions in static 2D stimuli. They depict the overall distribution of visual attention (from one or more participants) for a given stimulus.

Based on the descriptions provided in Chapter 3.3 customization of features and the selection of alternative color gradients shall be incorporated. The user can, for example, adapt the opacity, visualization type, and time interval. Based on the *Sinn Builder* from NYAN 2.0<sup>XT</sup> (see page 35) different color gradients can be applied. A *white filter*, ranging from a transparent value (high interest) to white (low interest), provides a general idea about centers of visual attention. Whereas a gradient ranging from red (high interest) and yellow over green to blue offers a broader spectrum, allowing finer perception of differences in visual attention.

In the following sections three different techniques are de-

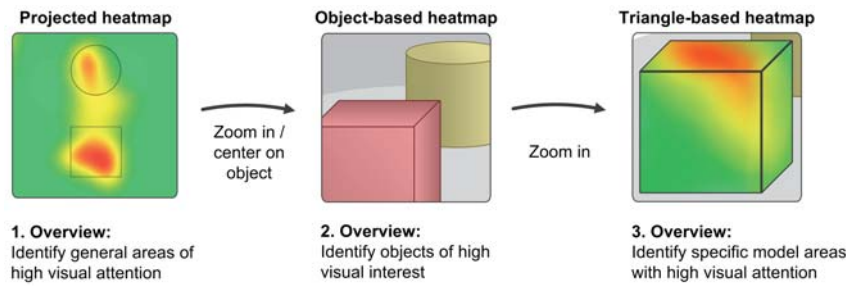


Figure 23. Concept for different levels of detail for aggregated gaze visualizations.

scribed for superimposing aggregated visualizations over virtual 3D stimuli: *projected*, *object-based*, and *triangle-based* attentional representations. A combination of these techniques can be used to implement different levels of detail. A projected attentional map can help identifying areas of high interest if a scene is scrutinized from afar. When zoomed in, distinct objects are colored based on how often they have been observed (unicolored models). Further amplification shows a detailed representation of how the visual attention is distributed over models' surfaces (colorized triangle mesh). Figure 23 shows the described course of action. The combination of different techniques contributes to a better orientation (overview) and improved performance especially with regard to large scenes.

#### *Projected Attentional Representation.*

The projected attentional map is a 2D representation of gaze data aggregated for specific views. It helps to get quick insights in the data and supports formulating hypotheses bottom-up. This technique accumulates 3D gaze points located on rays orthogonal to a given plane. Three views (and planes respectively) can be distinguished: *front* (x-y plane), *top* (x-z plane), and *side* (y-z plane).

An example of a projected attentional map for the top and side view is illustrated in Figure 24. In principle, the projected attentional map is similar to a *contour plot* [Woo02], in which the value  $y$  (Wooding refers to it as variable 'd') in the 3D fixation map is represented with a color in a 2D image. However, instead of simply adding the number of views for a certain position  $(x, y)$ , values from several layers need to be aggregated. If we look at a scene from above, all values  $(x, y, z)$  for position  $(x, z)$  have to be accumulated in 3D VEs. For

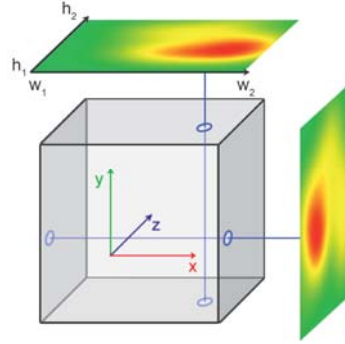


Figure 24. Gaze data is projected onto the top and side view planes.

this purpose the list of logged gaze points has to be passed through. The calculation for the top view can be done with equation 4.1. Variables  $w_1$ ,  $w_2$ ,  $h_1$ , and  $h_2$  are shown in Figure 24.

$$\text{Attention}(x, z) = \sum_{x=w_1}^{w_2} \sum_{z=h_1}^{h_2} \sum_{y=-\infty}^{\infty} \#\text{views}(x, y, z) \quad (4.1)$$

To compensate for peripheral vision a centered normal distribution (Gaussian) is assigned to marked entries. First, the *distance*  $d$  from a marked position to surrounding points is calculated and then applied to the distribution function. This procedure can be performed with the equations 4.2 for the top view. The *variance*  $\sigma^2$  can be altered by the user. A high variance anticipates a broad peripheral vision, whereas lower values indicate a more confined view. Finally, the accumulated values are mapped to a value between zero and one for assigning colors from the selected color gradient.

$$d = \frac{\sqrt{(d_x)^2 + (d_z)^2}}{\sigma}$$

$$f(x + d_x, y + d_z) = \frac{1}{\sigma \cdot \sqrt{2\pi}} e^{-\frac{d^2}{2}} \quad (4.2)$$

Burgert et al. [BOJ<sup>+</sup>07] show an example where a heat map is presented on the back plane of a room (see Figure 16(b), page 37) with a fixation map in front of it. This leads to overlapping layers, making it difficult to observe the entire visualization. Thus, it is more advisable to use superimposed representations similar to attentional maps for 2D stimuli.

### *Object-based Attentional Representations.*

This representation colorizes objects (unicolored) based on received visual interest. Since the log files contain all observed models, the number of views can be counted for each object. These values can then be represented by the selected color gradient.

Since each object is assigned a specific value the gradient is not continuous. This may impede identification of small attentional differences among objects. As a response to this problem details are displayed when selecting objects or when hovering over them with the mouse pointer. In addition, models are enumerated in a listing (see Section 4.3) which can be arranged according to the number of received views.

### *Triangle-based Attentional Representations.*

The triangle- (or vertex-) based attentional map visualizes gaze behavior across a model's surface. Similar to the approach by Wooding [Woo02] a 3D Gaussian is applied to each viewed mesh triangle. That way, adjacent triangles are granted a certain amount of visual attention. This amount decreases the further a triangle is away from the viewed one. This technique provides great insights into the visual gaze distribution over the surface of a 3D model.

Object logs provide information about viewed models and mesh triangles, with each triangle being described by three vertices. In order to determine the visual attractiveness of each triangle they have to be located in the model's triangle collection first. Once it is found the visual attraction level of the target triangle and its surrounding vertices are increased. This procedure is illustrated in Figure 25.

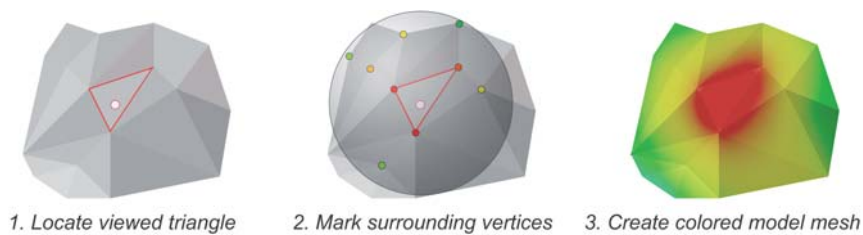


Figure 25. A cut-out of a model mesh is displayed to illustrate the triangle-based attentional representation.

#### 4.2.4 Summary

In this section the design of novel visualizations for gaze analysis was described. This included the [MOI](#) timeline, [3D](#) scan and camera paths, and aggregated representations. The [MOI](#) timeline is a space-filling visualization for displaying viewed objects and can be used to navigate the data collection. This enables replay of studies while superimposing additional information. Although the [3D](#) scan path is not a novel technique improvements for the application in [3D VEs](#) have been discussed such as adapted zooming and fixation representations. Finally, three options for aggregated gaze representations have been described, which can be combined to provide different levels of detail: *projected*, *object-based*, and *triangle-based attentional representations*.

### 4.3 COHERENT APPLICATION DESIGN

As described in the requirements analysis (Section [4.1.3](#)) a coherent application context has to be developed which integrates the presented visualizations, specified features, and supports several tasks such as overview, zoom, filter, and details-on-demand. For this purpose, the screen layout and basic functionalities for the application are described in Section [4.3.1](#). Desired features mentioned in the requirements analysis include easy-to-follow links between multiple visualizations. This issue is discussed in Section [4.3.2](#). Further features are described in Section [4.3.3](#). Finally, possibilities of how to apply the aforementioned tasks to the visualizations and entire system are discussed in Section [4.3.4](#).

#### 4.3.1 Screen Layout

A concept for the screen layout for the coherent application context is shown in [Figure 26](#). The screen contains six main areas: *menubar*, *toolbar*, *side panel*, *viewing area*, *timeline panel*, and *statusbar*. The *menubar* provides functions for loading scenes and subjects, and creating and arranging views. Each entry possesses shortcut commands for faster access. Visualization commands are represented by image buttons in the *toolbar*. Selected visualizations from the *toolbar* are applied to all active views. On the right side of the screen is the *viewing area* that

contains all subviews. Each subview has a context menu to select features (e.g., visualizations and filters) that will only be applied to the corresponding view. The *timeline panel* at the bottom integrates the **MOI** timeline, which can be used to navigate the data collection. The *statusbar* provides immediate feedback about the current status of the system.

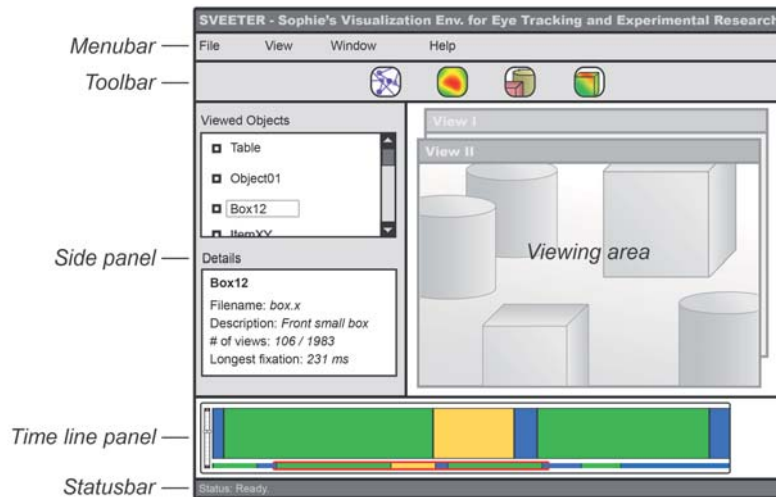


Figure 26. Concept of the screen layout.

In the *side panel* several listings are contained, describing scene objects, and details of selected items. Each object is described by a unique **ID**, filename, and description. In addition, assembled data (e.g., number of views, average fixation duration, and specified colors) are listed. Sorting items according to these different parameters is desired. In case of defined object hierarchies, the simple object listing in the side panel has to be replaced by tree diagrams. A common data tree visualization includes indented labels used in tables of content [CS94]. To better identify which entries belong to which models, a preview or icon of the corresponding model could be displayed.

#### 4.3.2 *Linked Views*

Models (or groups of models) can be represented by unique colors in different views (including object listings, the **MOI** timeline, and subviews). This concept is illustrated in Figure 27. Information about an item (e.g., model **ID** and description)

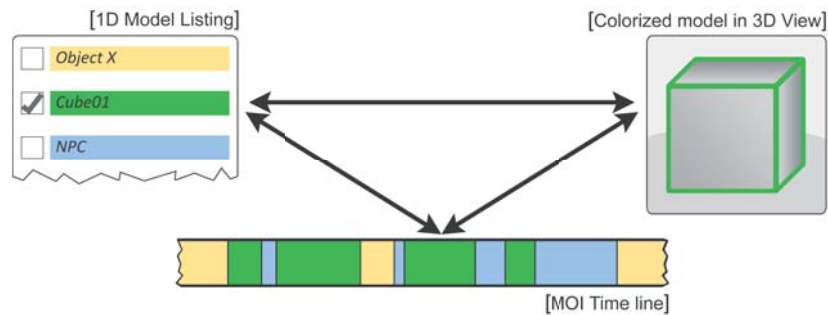


Figure 27. Different views are linked by consistent coloring. Each object receives a specified color, which can be changed by the user.

can be accessed and adjusted (e.g., color changes) in each view.

However, color coding objects as identification on the one hand, and as object-based attentional representation on the other hand is conflicting. Hence, alternatives for a clearer overview need to be discussed. In the **MOI** timeline, viewed models are represented by colored rectangles. Since rectangle widths may become very small and colors still allow good differentiation, it may be assumed that objects are by any means represented by colors due to their applicability in the **MOI** timeline. This results in two remaining possibilities to improve clarity: 1. Specify how to represent both objects and attention with colors or 2. Find an alternative to represent visual attention in the aggregated visualizations.

#### *Multivariate Color Coding*

There are several possibilities of how to use colors to represent both objects and attention and still maintain clarity. First of all, it is not advisable to colorize model surfaces for identification, since this is also done for the object-based attentional representation. Some alternatives are illustrated in Figure 28, namely colorized shadows, shells, name tags, model outlines, and polygon meshes. These techniques are appropriate for simple stand-alone models. However, they might be difficult to apply to complex entities, which are assembled by several small models. Techniques like displaying shadows on the 'virtual floor' are unsuitable, because a floor may not even be defined (or too far away from the designated item to display its shadow on it). Shells introduce additional artifacts, which

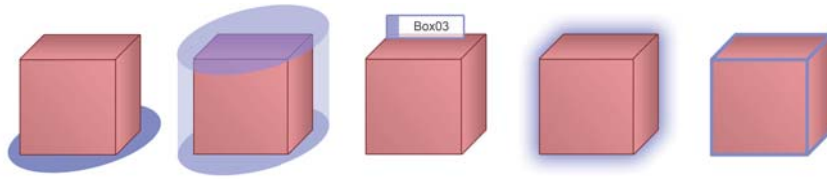


Figure 28. Different alternatives of how to represent objects and visual attention with colors are presented. The blue color represents the cube entity, while the red color is assigned to the cube's visual attractiveness.

may clutter the view. If items are tightly embedded into a complex construction, it may be difficult to position name tags or display their colorized outline. Therefore, the most feasible technique seems colorizing model meshes. To further improve clarity different (disjunct) color palettes can be defined.

#### *Alternative Representation of Visual Attention*

Aside from using colors to depict a model's visual attractiveness, alternatives should also be considered. Especially in huge scenes containing hundreds of items the difficulty of distinguishing between unique colors increases. Instead of using color gradients for aggregated visualizations, gradients based on transparency or luminance could be used. Areas of high interest could be displayed as opaque (or bright colored), while areas of low visual attention could be transparent (or dark colored). In this regard, implemented attentional map variations by NYAN 2.0<sup>XT</sup> [ino9] are relevant (see Figure 13, page 35). Even more elaborate techniques such as adapted bump mapping, non-photorealistic rendering, or vector field visualizations could be used to depict visual attention.

After all, for the course of this thesis a similar approach to NYAN 2.0<sup>XT</sup>'s Sinnbuilder [ino9] is used. The user can choose from color, transparency, and luminance gradients. For improved clarity, colors used for attentional maps should not be used for object representations. Moreover, it should be possible to colorize entire models or only model meshes corresponding to assigned colors in the object listing (in the side panel) and MOI timeline.

### 4.3.3 Features

Since the intended stimuli are VEs the analyst should be able to load scenes and freely explore the 3D VE. For this purpose, different 3D scenes and corresponding log data need to be loaded.

#### *Load scenes*

For loading scenes it is necessary to describe their setup. Since the 3D scenes are static, meaning that model properties do not change, they can be specified in XML files. Examples for such model properties are listed in Table 6. Necessary data for post-analysis can be retrieved from these files to rebuild defined scenes.

| Property          | Example                                |
|-------------------|--|
| Filename          | <i>box.x</i>                           |
| Model ID          | <i>Box05</i>                           |
| Model description | <i>Small box with texture of a dog</i> |
| Position          | <i>(-1.4, 0.7, 1.0)</i>                |
| Scale             | <i>(0.5, 0.5, 0.5)</i>                 |
| Rotation          | <i>(0.0, 0.5, 0.0)</i>                 |

Table 6. Static 3D VEs can be described by model properties.

#### *Load subject data*

An eye tracking study may include several participants. Each subject may have certain personal data, which are important for the evaluation of a study, including visual acuity and demographic information (e.g., age and gender). In addition, each subject corresponds to a certain set of log files. Thus, when loading a particular subject, associated log files must automatically be referenced as well. Subject data may also be described in XML files.

After setting up the foundation for the system by loading necessary data some features requested by eye tracking professionals are examined (see Section 2.4, page 16). This includes multiple views to look at the scene from different angles.

### *Multiple views*

For the analysis of 3D VEs it is a common standard to provide different views at a scene simultaneously. Scene editors (e.g., *Hammer Editor* by Valve) and modeling tools (e.g., *3ds Max* and *Maya* by Autodesk or *Cinema4D* by Maxon) usually provide orthographic top, side, and front, as well as isometric views. The herein presented instrumentation shall use a looser arrangement. The user can define as many additional views as desired and appoint individual viewpoints to each (either orthographic or perspective). Several default viewpoints are available, including *top*, *front*, *side left*, *side right*, and *object centered views*. Furthermore, users can save and load their own viewpoints via a context menu. Commands from the menu and toolbar (e.g., apply visualizations), side panel (e.g., filter models), and MOI timeline (e.g., data navigation) are applied to all subviews. However, if desired, gaze visualizations and the filtering of properties (e.g., displaying textures and model grids) can be assigned to individual views by configuring the settings in the context menu of a particular view. That way, it is possible, for example, to compare aggregated visualizations with scan paths.

### 4.3.4 *Tasks*

Aspects for improved gaze data exploration need to be considered. Shneiderman's *Information Seeking Mantra*: "Overview first, zoom and filter, then details-on-demand" [Shn96] has been mentioned previously in Chapter 3. In the following paragraphs, possibilities to incorporate these basic principles in the presented system are discussed.

#### *Overview*

Overviews help to get a general idea of a given data collection. It also serves to facilitate orientation. Based on common geographic maps, a view from above is often used for this purpose in 3D VEs. Projected heat maps can be used to offer such an overview for gaze data acquired throughout a scene. Moreover, item listings can be used to give an overview of observed objects and models contained in a scene. For this purpose, the MOI timeline offers a general idea about gaze behavior. The listing of scene models in the side panel (see page 62) provides an overview of the general scene setup.

### *Zoom*

In general, two types of zooming can be distinguished: *physical (geometric)* and *semantic* [Mod97]. While a geometric zoom adapts the size and visible detail of objects, a semantic zoom changes the appearance and meaning of displayed information [Mod97]. An example for semantic zooming is displaying city and street names on a digital map. Zooming in will make street names and even additional information (e.g., hotel names and local sights) available. Zooming out will cause this information to disappear from the screen. Geometric zooming in a 3D VE can be accomplished by simply altering the camera's position (i.e., moving closer/further away from an object by rotating the mouse wheel or using the cursor keys). With semantic zooming detailed information such as fixations or icons (for the MOI timeline) can be provided if an item is examined more closely; otherwise that information will be clustered to a more compact representation. With respect to 3D scan paths, fixations may retain the same size relative to the screen size while zooming.

### *Filter*

Filtering out unessential data helps assessing a given situation better; it aids in concentrating on items of interest. There are several options to support filtering in the described system. On the one hand, this includes filtering objects, textures, and model meshes. Reasons for hiding particular models are occlusions or the desire for a more limited view at the scene. The mentioned filters can either be applied to a certain view only via corresponding context menus or to all views via commands in the side panel. On the other hand, further filtering in a spatial and temporal context can be done. The MOI timeline can be used to limit data temporally.

Items can be spatially filtered by selecting objects in the 3D view. That way, it is possible to display only objects positioned in a certain area. In addition, the object listing (in the side panel) supports sorting items according to different filter criteria such as number of views, average fixation length, and total fixations.

### *Details-on-Demand*

This technique provides information about selected items or groups if needed. When hovering over items with the mouse

pointer brief information is presented close to the mouse position. In the 3D views this may only include the model ID, whereas in the MOI timeline it also contains the corresponding time stamp. Moreover, the context menus integrate a “Show properties” entry to display more detailed information in the designated textbox in the side panel.

#### 4.4 SUMMARY

This chapter described the conception of a system that integrates novel visualizations for improved eye tracking analysis in 3D VEs. In Section 4.1 a requirements analysis has been carried out, describing user scenarios, development considerations, the anticipated input space, and desired features for a coherent application context. This also included the specification of a triangle-precise eye tracking procedure. The presented approach supports static 3D VEs avoiding transparent phenomena.

Based on the investigated requirements several ways how to depict gaze data for 3D VEs have been devised in Section 4.2. In Table 7 the discussed visualizations are summarized.

In Section 4.3 the design of a coherent application has been described. For this reason, trends and needs demanded by eye tracking professionals (see Section 2.4, page 16) and Shneiderman’s *Information Seeking Mantra* [Shn96] have been considered.

| Dimension | Visualization                         |
|-----------|---------------------------------------|
| Temporal  | Space-filling MOI timeline            |
| 2D        | Projected attentional map             |
| 3D        | Scan and camera paths                 |
|           | Object-based attentional distribution |
|           | Triangle-based attentional map        |

Table 7. Overview of presented visualizations in Chapter 4.

## IMPLEMENTATION

---

After developing concepts for enhanced gaze analysis techniques for *three-dimensional (3D) virtual environments (VEs)* in Chapter 4, this chapter describes their software implementation. Deployed development environments are introduced in Section 5.1. This includes the integration of the Tobii 1750 Eye Tracker (Tobii ET-1750) system with the XNA Framework. As prerequisite for gaze visualizations required data have to be supplied. For this purpose, data acquisition (including logging, processing, and storing data) is discussed in Section 5.2. The logged data can then be loaded with *Sophie's Visualization Environment for Eye Tracking & Experimental Research (SVEETER)*, which is the gaze analysis tool developed in the practical part of this thesis that integrates novel gaze visualizations. The implemented system is described in detail in Section 5.3.

### 5.1 DEVELOPMENT ENVIRONMENT

Based on the requirements analysis in Chapter 4, a development environment is needed for creating 3D VEs, in which a user's gaze can be logged with triangle accuracy. For this purpose, the *Tobii ET-1750* is integrated with Microsoft's *XNA Game Studio 3.0*<sup>1</sup>. XNA is a toolset for facilitating digital game development and management and is based on the native implementation of the *.NET Framework 2.0*. Supported programming languages include C#, which is an object-oriented language developed by Microsoft. As recommended for XNA developers, *Visual C# 2008 Express Edition* is used as an integrated development environment.

The decision to use XNA and the .NET Framework was motivated by several reasons, which include the following:

- Necessary tools are freely available
- Prior programming experience with C#

---

<sup>1</sup> See <http://www.xna.com/>. Last accessed March, 2009.

- Large online developer community<sup>2</sup> providing tutorials and feedback
- Simplified development of virtual 3D (game) scenarios
- Possibility to integrate existing interface elements from *Windows Forms* for the application design

The main drawback using XNA for conducting studies is the higher development effort due to additional content creation. However, with regard to preceding eye tracker integrations using a commercial game engine [Jö5, Seno4, SAC<sup>+</sup>07, Steo7], those approaches are limited to a certain context (such as First-Person Shooters). Thus, in exchange to increased development costs, the XNA framework provides high flexibility.

For the course of conducting an eye tracking study different system environments are mandatory (see Figure 29). The first stage, the design of static 3D VEs, only requires XNA. Thus, an eye tracker is not needed for scenario creation, which includes specification and positioning of models. For preliminary testing, gaze target points can be supplemented by mouse pointer coordinates. The second step (testing) requires the *Tobii SDK*. This is a collection of *Application Programming Interfaces (APIs)* for communicating with the eye tracker and thus is essential for conducting an eye tracking study with participants. After collecting necessary data, log files can be loaded by the herein described gaze analysis tool (*SVEETER*). *SVEETER* is based on XNA to provide 3D views and on *Windows Forms* to gain from existing interface elements (e.g., menus, buttons, file dialogs, and list views). In the following sections the integrations of XNA with the *Tobii SDK* and with *Windows Forms* are described.

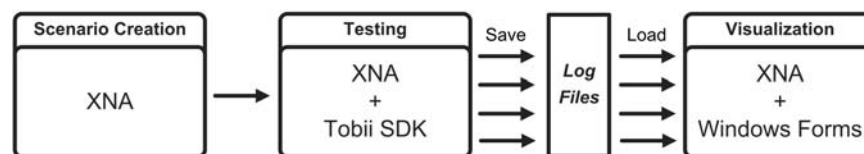


Figure 29. Overview of system environments required for different stages of an eye tracking study lifecycle, including preparation (scenario design), testing, and analysis (gaze visualization).

<sup>2</sup> See <http://creators.xna.com/>. Last accessed March, 2009.

### 5.1.1 Eye Tracker Integration with XNA

For the integration of the Tobii ET-1750 with the XNA Framework, the *Tobii Eye Tracker Server* (TETServer) and the *Tobii SDK* are fundamental. The Tobii SDK is a collection of APIs to access the TETServer controlling the eye tracker hardware. This includes the *Tobii Eye Tracking Components API* (TetComp), which is an interface containing all required methods to get real-time gaze data from the eye tracker [Tobo5]. Figure 30 shows TetComp objects and their internal dependencies and illustrates that TetComp communicates with the TETServer indirectly via the low level API. The *TetClient* (one of the TetComp objects) handles communication to the lower software abstraction levels. Thus, it is possible to develop a complete eye tracking application by using this object only. At a maximal frequency (for the Tobii ET-1750 this is 50 Hz, see Section 2.2.1 for more information) gaze data is triggered as regular COM events. These events have to be handled within the XNA implementation to complete the integration. However, before going into detail about the XNA implementation, the systems setup is specified first.

At the *Game and Media Arts Laboratory* (GAMALab), we use a dual computer, dual screen configuration as described in Tobii's user manual [Tobo6]. The eye tracker may reside on any host as long as it has IP connectivity to the host running the application that uses TetComp. Figure 31 describes this setup and the systems' communications. The application computer runs the scenario and is responsible for the automated logging of viewed models. The eye tracking computer accesses the TETServer which provides a control interface to the eye tracking hardware.

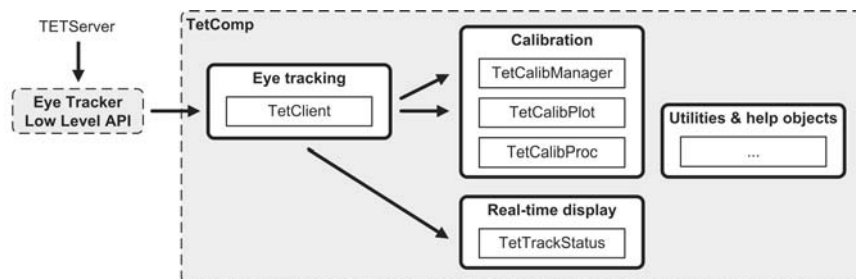


Figure 30. Dependencies and categories of the TetComp objects

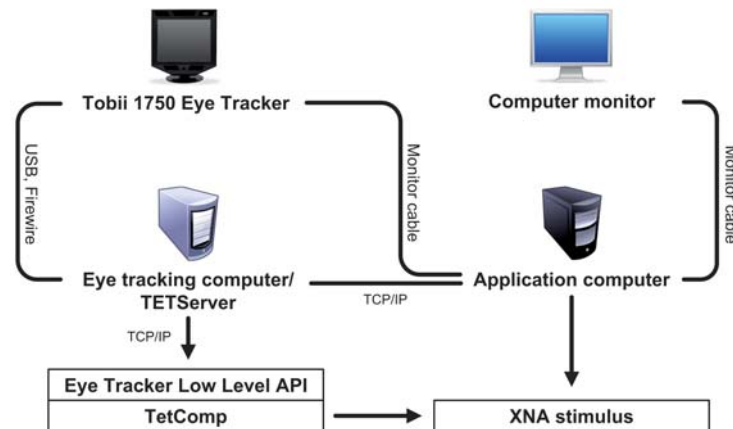


Figure 31. Dual computer, dual screen setup and communication between the different systems.

A mediator between XNA and `TetComp` is established by the `GazePointer()` class, which implements a listener function to wait for input from the `TETServer` (or eye tracking computer respectively). Figure 32 illustrates the structure for the integration of both systems, while the mentioned classes are briefly described in the following paragraphs.

#### *GazeTracker*

The `GazeTracker()` class is responsible for accessing the `TETServer`. This includes eye tracking initiation, calibration execution, and gaze data event handling.

#### *GazePointer*

This class manages 3D gaze positions, including the transformation of received 2D gaze target positions into the virtual

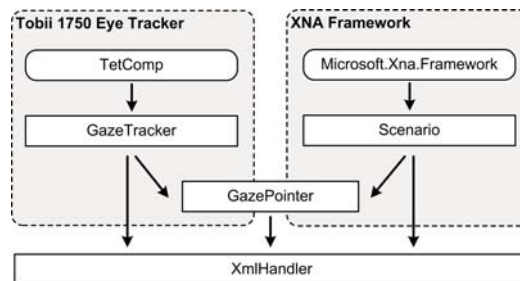


Figure 32. Structure for the integration of the Tobii ET-1750 with the XNA Framework.

3D space. It also handles the identification of viewed objects.

### Scenario

The Scenario() (or by default Game()) class manages the 3D VE. This includes loading and drawing content (i.e., textures and models). This class implements GazePointer().

### XmlHandler

The XmlHandler() is a simple logging class for reporting desired information to XML files. As described in Section 4.1.2 (page 45), gaze, object, and camera log files are distinguished. Further details about the structure of the different XML log files are discussed in Section 5.2.

The described integration of the Tobii ET-1750 with XNA enables the creation of log files with required data, such as 3D gaze positions, viewed models and triangles, as well as camera positions and viewing directions.

#### 5.1.2 XNA and Windows Forms Integration

As mentioned in Chapter 4 a coherent application context is desired, which integrates novel gaze visualizations. For this purpose, SVEETER builds on XNA and Windows Forms to benefit from Windows interface elements.

The integration of XNA with Windows Forms is based on an article by Frederic My [My07]. An overview of the implemented system is shown in Figure 33. The individual classes will be briefly described in the following.

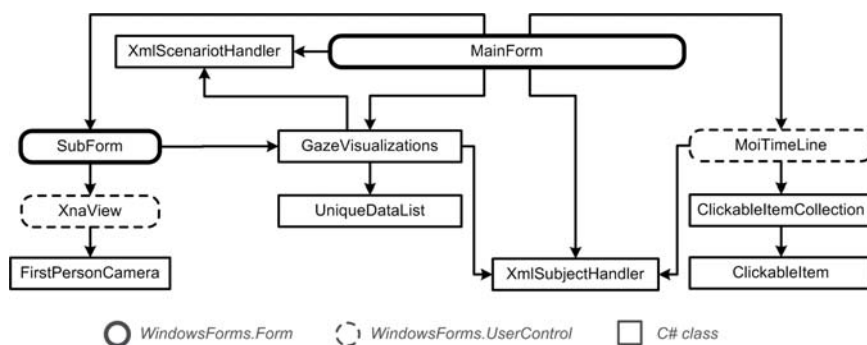


Figure 33. Overview of the implemented system for the gaze viewer application.

WindowsForm MainForm() is SVEETER's basis class and main window. It is used as a framework for integrating the desired interface elements which have been specified in Section 4.3.1. This includes the *menubar*, *toolbar*, *side panel*, *statusbar*, *viewing area*, and *timeline panel*. The first four interface elements are implemented by MainForm(). The viewing area contains multiple views which are handled by the WindowsForm SubForm() and UserControl XnaView() classes. Each SubForm() manages one XnaView(), which is associated with a render target and a depth buffer to render respective 3D views. Resulting renderings are integrated in the subviews as textures. This is done by the RenderToTexture function, which is called by a SubForm()'s Draw function.

An aspect which was not described by My [My07] was the support of multiple views for looking at one specific scene from different angles. For this purpose a class managing camera settings for each view (and SubForm() respectively) is defined: FirstPersonCamera(). The user can move with the cursor and up/down keys in a subview if the mouse pointer is hovering over it. In addition rotating the mouse wheel results in zooming in/out. The FirstPersonCamera() classes are responsible for handling input concerning camera movements.

The MOI timeline is implemented by the MoiTimeLine() class. A Windows.Forms.Panel is used as basis for the timeline. The panel acts as a container for the ClickableItemCollection() in which each log entry is represented as one ClickableItem(). Each ClickableItem() contains five properties to represent log entries in the timeline, which are described in Table 8. This way, object-related details and menu options can be displayed.

The superimposed visualizations, such as scan and camera paths as well as aggregated representations, are implemented by the GazeVisualizations() class. The UniqueDataList() class is used for storing unique items of an array, for example viewed models. For this purpose, each entry in the UniqueDataList() has a variable specifying the number of occurrences.

The XmlSubjectHandler() and XmlScenarioHandler() implement methods for loading XML files that describe particular scenes and subjects (as described in Section 4.3.3). The XmlScenarioHandler() manages displayed models and corre-

| <i>Property</i>              | <i>Description</i>  |
|------------------------------|---|
| <code>int</code> ColorIndex  | <i>Item's color index in a predefined color array</i>                                   |
| <code>int</code> ObjectIndex | <i>Index of the current item in an array containing all log entries</i>                 |
| <code>string</code> ObjectID | <i>Object identifier</i>  |
| <code>Point</code> Location  | <i>Location of a <code>ClickableItem()</code> in the timeline panel</i>                 |
| <code>Size</code> Size       | <i>Size of a <code>ClickableItem()</code> depending on the corresponding time stamp</i> |

Table 8. Properties of *ClickableItem*.

sponding data, such as model descriptions. The `XmlSubjectHandler()` manages subject information, such as a subject **ID**, age, gender, and associated log file names and data. Therefore, the classes `GazeVisualizations()` and `MoiTimeLine()` need to access these classes to visualize gaze data. The acquisition of such data is described in the following section.

## 5.2 DATA ACQUISITION

As prerequisite for the visualization of gaze behavior, techniques to acquire and store necessary data have to be implemented. Section 5.2.1 describes how objects are tracked with triangle accuracy. Before saving data in log files (see Section 5.2.3), data is processed to filter out objectionable entries (see Section 5.2.2).

### 5.2.1 Object Tracking with Triangle Accuracy

Tracking viewed objects with triangle accuracy has been accomplished by adapting the “Picking with Triangle Accuracy” sample application by Microsoft<sup>3</sup>, which implements an algorithm for fast, minimum storage ray/triangle intersections

<sup>3</sup> See <http://creators.xna.com/en-US/sample/pickingtriangle>. Last accessed March, 2009.

by Möller and Trumbore [MT05]. This sample provides a custom content processor, `TrianglePickingProcessor`, to access model vertex positions at runtime. It overrides the built-in `ModelProcessor.Process` method. After scanning over a model's geometry tree, it stores all vertex positions into a list and attaches it to the `Tag` property of the output `ModelContent`. Thus, by accessing a model's `Tag` property, we can retrieve a list of all mesh triangles, represented by three entries per triangle. However, for this purpose the *ContentProcessor* for each model has to be set to `TrianglePickingProcessor`. Otherwise an exception will be thrown due to missing vertex position data.

After discussing the basic requirements for the triangle picking, the different steps for gaining the required data logs are briefly described (see also Section 4.1.2 for a general description of the tracking procedure). Four major steps have to be performed by `GazePointer()` (described on page 72) ranging from getting gaze data, transforming them, to performing collision detection and finally to the point of logging required data:

1. `GazePointer()` gets gaze target positions from `TetComp`.
2. A gaze ray is calculated depending on the current projection and view matrices.
3. *Repeat for each model:* Perform collision detection based on the calculated gaze ray, the current model, and its transformations. The method returns:
  - the camera's distance to the intersection,
  - a boolean value indicating, whether the camera is located within the model's bounding box, and
  - the intersected mesh triangle.
4. Log the determined model and additional data by accessing `XmlHandler()`.

While collecting data, objectionable data are automatically sorted out. This filtering procedure is described in the next section.

### 5.2.2 Data Filtering and Processing

During data acquisition some processing measures have to be taken to remove objectionable data on-the-fly. Thereby, two cases are distinguished:

1. The eye tracker is not able to locate both eyes
2. The gaze target points are not within the valid coordinate range (e.g., the user looks at something outside the screen).

The validity values provided by [TetComp](#) give information about the certainty of reported gaze data. Thus, the first issue can be resolved by filtering gaze data with validity values above 2 due to their high inaccuracy [[Tob05](#)]. In case of a user's gaze not directed at the screen, the values (-1, -1) are returned. This facilitates sorting out corresponding values.

In a nutshell, data with negative gaze target positions (-1, -1) or validity codes above 2 are removed. This processing can be performed during runtime.

### 5.2.3 Data Logging

For logging data the [XML](#) format is used. [XML](#) works well for structuring data in a self-documenting format (assuming that the names of [XML](#) elements are well chosen) [[@Wio9](#)]. This makes [XML](#) files more readable for both humans and machines. The advantage of [XML](#) files compared to simple text log files is [XML](#)'s strict syntax. This facilitates parsing algorithms and makes them simpler, more efficient, and consistent [[@Wio9](#)]. The `System.Xml.XmlReader` in the .NET Framework even allows using and accessing parsed results as local variables within the methods performing the parsing. As described in Section [4.1.2](#), three additional log files have to be created. Each log type has its distinct structure, which is shown in [Figure 34](#).

Although logging data to [XML](#) files worked well for initial studies testing [SVEETER](#), it may lead to verbosity and thus increased file sizes for more extensive studies. Assuming that 50 gaze points per second are collected (due to Tobii ET-1750's sampling rate of 50 Hz), the size of a gaze log file is approximately 5 megabytes for a 10 minutes session. The camera log

```

// Gaze log           // Object log           // Camera Log
<Entry>              <Entry>              <Entry>
  <Timestamp/>        <Timestamp/>        <Timestamp/>
  <GazeData>         <ObjectID/>         <CameraPosition>
    <x/>              <PickedTriangle>    <x/>
    <y/>              <Vertex1/>          <y/>
    <z/>              <Vertex2/>          <z/>
  </GazeData>        <Vertex3/>          </CameraPosition>
</Entry>            </PickedTriangle>  <CameraDirection>
                    </Entry>              <x/>
                                        <y/>
                                        <z/>
                                        </CameraDirection>
                                        </Entry>

```

Figure 34. XML syntax for the different log file types

even amounts to twice that size. Thereby, the data volume is independent from the scene size, while the eye tracker’s sampling rate is crucial. Increased file sizes could affect application efficiency due to higher storage, transmission, and processing costs [Wio9]. Therefore, it is recommended to implement a database communication for more efficient data storage while still providing fast access to individual data entries.

### 5.3 IMPLEMENTED VISUALIZATIONS

After establishing a system for delivering required log files, the next step is to visualize these data. In the following sections, implemented visualizations based on the concepts discussed in Section 4.2 are presented.

#### 5.3.1 Models of Interest Timeline

The *Models of Interest (MOI)* timeline is created automatically after loading subject data to provide a session overview. As shown in Figure 35 details about items are presented when

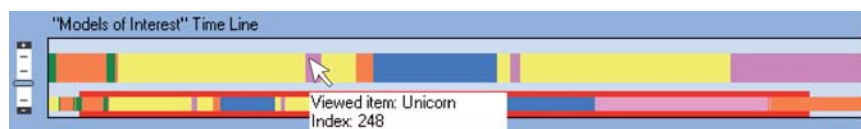


Figure 35. An example of the implemented MOI timeline.

hovering over them. The upper view in the timeline can be zoomed by rotating the mouse wheel and can be moved by clicking and dragging. Time stamps are mapped between zero and the panel's width. Mapped time stamps are used to specify the width of the timeline entries.

The MOI timeline can be used to limit depicted scan and camera paths. For this purpose, additional navigation sliders will appear if these visualizations are selected. Otherwise the sliders are hidden. The user can simply click and drag the respective slider to define intervals/objects of interest. Figure 36 shows an example.

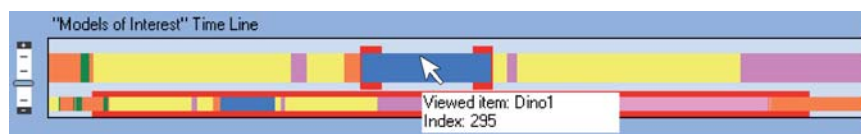


Figure 36. The data collection can be navigated by using additional sliders in the MOI timeline.

### 5.3.2 Scan and Camera Paths

Based on 2D gaze plots, two types of 3D scan paths are distinguished: *pure scan paths* and *fixations and saccades*. For the first possibility, the logged 3D gaze positions are simply represented by a list of connected lines. The second choice requires the identification of fixations. Each fixation is represented depending on its logged position and fixation duration. Moreover, the user can specify whether fixations should be depicted as spheres or cones. Figure 37 shows an example for fixations and saccades using spheres.

Fixations' magnification factors can dynamically be adjusted to get a better overview. Figure 38 shows an example for different magnification factors, where fixations are represented as cones. In case of huge scenarios the possibility to adapt fixations' magnification factors may facilitate getting quick overviews about gaze paths.

Based on related work in this area (see Section 3.3.3 and Section 3.3.4), saccades are represented as straight lines connecting fixations. This, however, causes problems in 3D due to saccade lines crossing through surfaces. One approach to respond to this issue would be to adapt lines to run on a

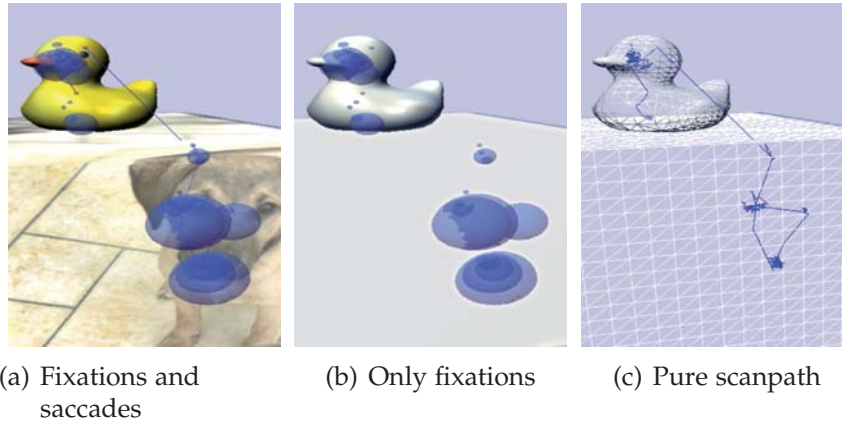


Figure 37. Screenshots from the implemented 3D scan paths.

model's surface (or above) in order to be visible at all times. However, this would not accurately reflect that saccades are simple connections between fixations. Moreover, this approach would lead to highly increased processing costs. Another possibility is to find an alternative representation for saccades compared to thin lines connecting fixations from center to center. Fixations and saccades could, for example, be represented by a gaze tube or gaze veil. However, such techniques have to be handled with care, because they usually result in more overlapping graphics and may complicate detecting what has been looked at. A third possibility, besides adapted lines and substitute representations, is the simplest technique to implement: the traditional saccade representation remains, but the models' transparency values can be adapted to be able to see which fixations are connected. For this purpose, models can also be represented by their wireframes (see Figure 37(c)).

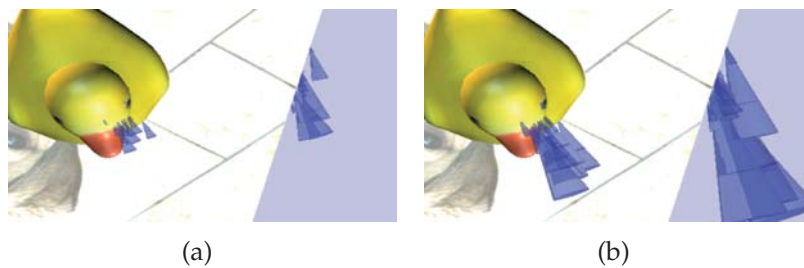


Figure 38. Fixations can be represented by cones. The magnification factors from fixations can be dynamically adjusted.

In addition to depicting scan paths, it is also important to visualize how the camera locations and viewing directions have changed during a session. For this purpose, the camera path is visualized with traces pointing at the respective gaze positions. In Figure 39(a) an example is shown, where the scene has been observed from one viewpoint. In contrast, Figure 39(b) depicts changing camera locations.

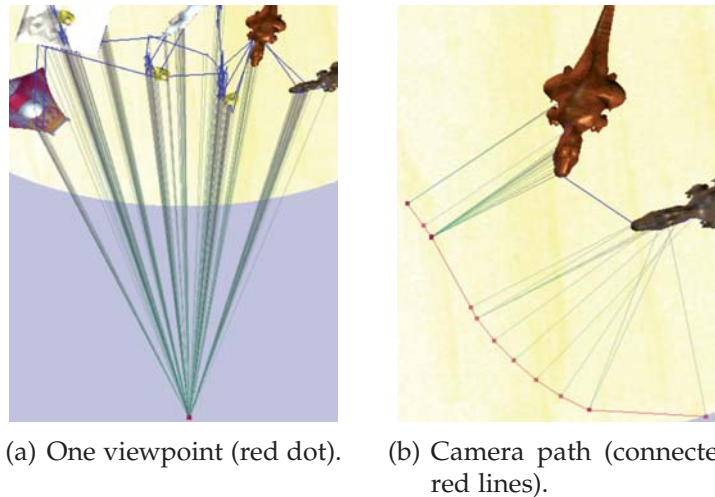


Figure 39. Two examples for camera paths and viewing traces.

### 5.3.3 Aggregated Representations

Aggregated representations can help to understand the general distribution of visual attention in a scene. Based on the concept in Section 4.2.3, three visualizations of this type have been implemented: *projected*, *object-* and *triangle-based* representations. The visualizations use color gradients, which can be selected from a drop-down menu.

#### *Projected Attentional Representation*

The projected attentional representation provides an overview of the gaze distribution in a scene by creating heat maps for predefined views (i.e., top, front, and side views). Projected heat maps are presented in an additional overview window. Figure 40 shows an example where a projected heat map is applied to the top view. The user can zoom by rotating

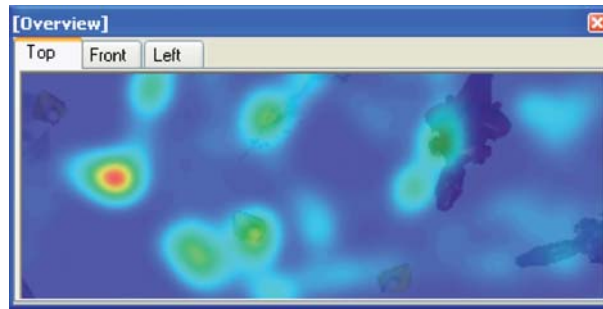


Figure 40. Projected attentional map superimposed over the top view.

the mouse wheel, while hovering over the overview window. Another option is to use the add (subtract) keyboard key to zoom in (out). The user can also pan the heat map by using the cursor keys or by clicking and dragging.

Additional menus to customize the heat map are available. Variables such as the variance  $\sigma^2$ , the attentional radius  $r$  (how far items are allowed to be to still be considered important), transparency, and color gradient can be adjusted.

From the implementation perspective, a 3D sprite is used to depict the projected attentional map. It is then superimposed over a rendered image of the 3D VE.

#### *Object-based Attentional Representation*

The object-based attentional representation gives a quick overview about how often objects have been looked at. Each model is painted in a certain color depending on how often it has been viewed and depending on the selected color gradient (see Figure 41).



Figure 41. Example of an object-based attentional representation (with filtered textures).

When first loading the logged data, it is automatically determined how often each model has been looked at. The amount of views has to be normalized to assign color values from the selected color gradient. Finally, the models' diffuse and specular colors are set to the identified color values.

### *Triangle-based Attentional Representation*

The triangle-based attentional map can be used for more detailed information about the gaze distribution. The visualization depends on how often a certain triangle has been looked at, the specified variance  $\sigma^2$ , and radius  $r$  (or gaze tolerance). The radius indicates whether vertices at a certain distance from a given gaze point should still be considered *visually attractive*. An example of the deployed visualization and differing radius values is shown in Figure 42.

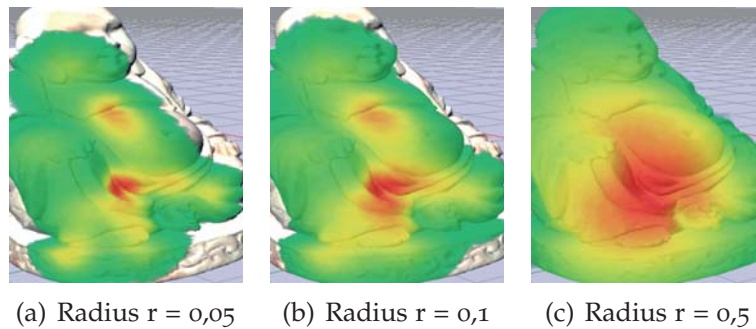


Figure 42. Gaze distribution is represented as an attentional heat map skinned to the model's surface. Examples for differing radius values are presented, while transparency is 80% and  $\sigma^2$  is 0,5.

An adjustable radius plays an important role considering peripheral sight and the compensation of inaccuracies (see Section 2.2.1). However, it may also be important if model sizes differ significantly between scenes. Assume, for example, that we have one specific scene stored with different scales. The user may not notice a difference when viewing the different representations. However, for assigning tolerance values the differing dimensions have to be considered.

Besides manually adjusting the radius it should ideally change automatically depending on the corresponding viewing distance. The closer a user is to a gaze target, the lower the

tolerance should be. However, if an object has been viewed from a further distance so that the whole object is in the observer's sight, the higher the radius should be.

Similar to projected heat maps, various properties for this visualization can be customized. This includes the variance  $\sigma^2$ , attentional radius  $r$ , transparency, and color gradient. An example for the application of different color gradients by the triangle-based representation is shown in Figure 43. Additional context menus will be displayed for customization if right-clicking on a subview.

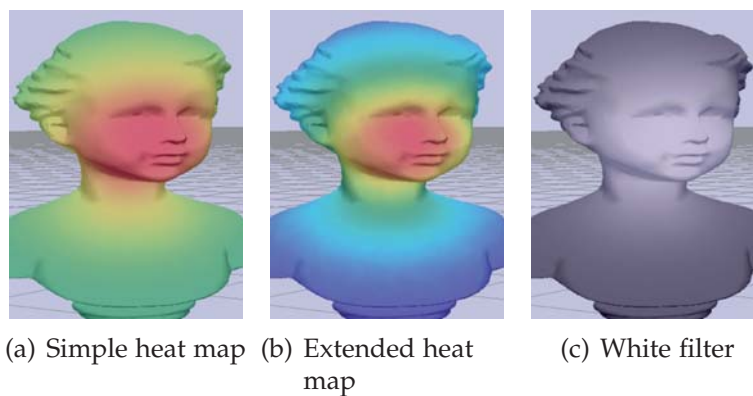


Figure 43. Different color gradients can be assigned to visualize gaze distribution.

#### 5.4 IMPLEMENTATION OF A COHERENT APPLICATION

The implemented gaze analysis tool, called *Sophie's Visualization Environment for Eye Tracking & Experimental Research (SVEETER)*, combines a coherent framework for loading different 3D scenes and corresponding subjects with convenient gaze visualization techniques. It is based on *XNA* and *Windows Forms*. The user interface is based on the screen layout devised in Section 4.3.1. Figure 44 shows a screenshot of the implemented system.

Commands are quickly accessible via short-cuts and descriptive buttons (showing the anticipated visualization). Multiple views can be defined by the user. Each view has its own context menu to provide options applicable for a respective view only. Additional tasks are available in the left sidebar, where actions concerning all views can be issued, for example filter-

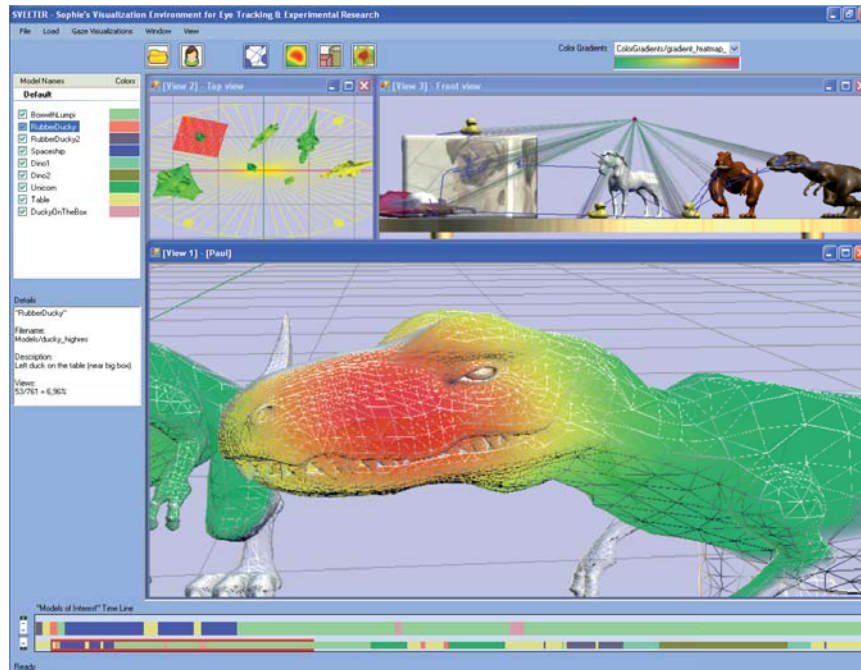


Figure 44. Screenshot of SVEETER.

ing certain models. In the statusbar at the bottom immediate feedback is provided.

## 5.5 SUMMARY

In this chapter, the implementation of modules managing different stages of an eye tracking study lifecycle for *three-dimensional (3D) virtual environments (VEs)* were discussed. These ranged from preparation (scenario creation) and testing to analysis (gaze visualizations). For scenario creation and testing *XNA* and the *Tobii SDK* have been deployed. Procedures for logging, processing, and storing necessary data were addressed.

These procedures provide the basis for the third stage, the gaze analysis, for which *Sophie's Visualization Environment for Eye Tracking & Experimental Research (SVEETER)* was introduced. *SVEETER* is a gaze analysis tool for *3D VEs* based on *XNA* and *Windows Forms*. The user can deploy *SVEETER* for exploring scenes with superimposed gaze visualizations, such as *3D scan* and camera paths and aggregated visualizations. Three different types of aggregated visualizations

were described: the projected, object- and triangle-based attentional representations. In addition, the implementation of the *Models of Interest* (MOI) timeline was described, which displays viewed models in a one-dimensional timeline. This timeline provides a comprehensive and accessible overview of time-related questions concerning gaze distribution.

## CONCLUSION AND FURTHER WORK

---

This chapter summarizes the research and development carried out in the scope of this thesis. Furthermore, prospects for further development of the presented techniques are described.

### 6.1 SUMMARY

This thesis addressed the improvement of current gaze analysis techniques, in particular gaze visualizations for *three-dimensional (3D) virtual environments (VEs)*. For this purpose, a gaze visualization tool, *Sophie's Visualization Environment for Eye Tracking & Experimental Research (SVEETER)*, was developed, which uses novel visualization techniques to display gaze data. It currently supports the analysis of static *3D VEs*, which the user can freely explore. Features for improved gaze analysis techniques were identified in a survey conducted among eye tracking professionals worldwide.

In order to enhance visual gaze analysis, a gaze visualization taxonomy was established to classify visualization techniques and emphasize their particular strengths (and weaknesses). In general, visualizations can help researchers to understand and interpret gaze data and to reveal gaze patterns. While *two-dimensional (2D) gaze plots* perform this task well for static *2D* stimuli, they cannot be efficiently applied to *3D* or dynamic stimuli. A frame-by-frame analysis of video recordings usually has to be done, which is very time-consuming if done manually. Based on widely used *2D* gaze visualizations, such as scan paths and attentional maps, enhanced techniques and alternatives for the application in *3D* environments have been discussed. In addition to that, camera paths and viewing directions were visualized, since users can change their viewpoint.

Furthermore, different ways to integrate attentional maps in *3D VEs* have been developed: *projected*, *object-based*, and *triangle-based* attentional maps. These aggregated visualizations may provide different levels of detail for exploring gaze

distributions. While projected attentional maps may provide an overview of gaze data acquired throughout a scene for a specific view (top, front, side), the triangle-based representation presents detailed gaze distributions across a model's surface. The object-based attentional map assigns each model with one color depending on the received visual attention.

Besides these superimposed visualizations, a timeline of viewed models was devised. This *Models of Interest (MOI)* timeline provides an overview of gaze orders and dwell times. It can help to answer questions, such as whether an object has been looked at during a certain time interval or whether some items are usually viewed one after another. The timeline can also be used to navigate the data collection to limit depicted scan and camera paths.

The aforementioned visualizations were integrated in a software application: *SVEETER*. This tool provides multiple views, links between different visualizations, and various techniques to filter properties in the different views (e.g., filter textures, models, and transparencies).

After describing the accomplished work, the following sections discuss the developed system, especially concerning further improvements. In addition, future research tasks are specified.

## 6.2 DISCUSSION

The developed framework provides techniques to evaluate eye tracking studies in 3D VEs. Questions, such as *"Which parts of an object received high visual attention?"* or *"Has a certain object been looked at frequently?"* can quickly be answered with the implemented visualizations. However, the presented techniques are still prototypes needing further refinement before being applicable in an industry context.

The system currently supports static 3D VEs without transparencies. Before enhancing the system and adapting it to dynamic or even unique stimuli, it is essential to verify the effectiveness and efficiency of the presented approach. For this reason, a verification study assuring the system's accuracy will have to be conducted as one of the logical next steps. In addition, empirical user studies should be done to evaluate the software and the visualizations.

This thesis provided techniques for managing different

stages of an eye tracking study lifecycle, including preparation (scenario creation), testing (data logging), and visual analysis. With respect to these different stages considerations and improvements are discussed in the following paragraphs.

#### *Scenario Creation*

While XNA provides a flexible and well-working environment for scenario creation, the user (the person who designs the scene) has to manually define model positions in configuration files. The handling could be improved by utilizing a graphical user interface to facilitate simple model positioning, a feature quite common in game-specific editing and modding tools. For this purpose 3D graphics applications, such as *Blender*<sup>1</sup>, or solutions based on XNA, such as the *Torque 3D game engine*<sup>2</sup> could be used.

#### *Data Logging*

As mentioned before, a verification study is necessary to assure data and software accuracy. In addition, data should be logged to databases instead of XML files. This is especially important for more extensive studies or for eye trackers with higher recording frequencies, since the use of XML quickly results in substantial log file sizes.

Another issue how to improve data logging is to shift laborious calculations, such as the identification of viewed mesh triangles, to a post-hoc processing stage. Since stimuli are static, this is a feasible approach to keep delays during data recording low.

Once the system's reliability is assured, additional data could be recorded. The system could, for example, log if the user moves into a certain area or presses particular buttons. Further data may even include psychophysiological input, similar to the psychophysiological game event logging presented by Stellmach [Ste07] and Nacke et al. [NLS08].

#### *Gaze Analysis Tool*

SVEETER shows high potential for facilitating eye tracking studies in 3D VEs. In scope of this thesis, it was not possi-

---

<sup>1</sup> *Blender* is an open source 3D content creation suite, available for all major operating systems under the GNU General Public License: <http://www.blender.org/> (last visited April, 2009).

<sup>2</sup> The *Torque 3D* game engine is published by GarageGames: <http://www.garagegames.com/products/torque-3d/> (last visited April, 2009).

ble to implement all of the requested features, which were mentioned by eye tracking professionals and researchers in the survey. Therefore **SVEETER** could be further elaborated based on the reported improvements, for example by supporting data export to statistical analysis software and annotation functionalities. However, before continuing on the development, it is advisable to conduct empirical user studies to find out how useful the developed techniques are in practice so far.

### 6.3 FURTHER WORK

Further work includes tackling the aforementioned improvements, but also further development on the tool in general to improve its usability and documentation for first-time users. This includes quality assurance by conducting empirical verification studies of the eye tracking procedure and user studies about the usefulness and efficiency of this tool. After establishing a stable and reliable software system, additional features and visualizations can be integrated in an iterative manner.

## APPENDIX

## A.1 INTERVIEW QUESTIONS ON EYE TRACKING EXPERIENCES IN RESEARCH

## 1. Quantitative Questions

QI\_0 What is your ID?

*Has been sent to you via email*

QI\_1 What is your age in years?

QI\_2 Are you male or female?

QI\_3 How would you grade your knowledge concerning eye tracking?

*On a scale from [1-no knowledge] to [5-extensive knowledge]*

QI\_4 For how many years have you been working with eye trackers?

QI\_5 How many studies have you worked on incorporating eye tracking for analysis?

QI\_6 How important were visualizations for the analysis of such studies?

*On a scale from [1-useless] to [5-essential]*

QI\_7 What kind of stimuli were used for these studies?

*Please check all that apply!*

- 2D Static: Images, static websites

- 2D Dynamic: Videos

- 2D Interactive: 2D user interfaces

- 3D Static: Static view on 3D model or virtual scene

- 3D Dynamic: Animated 3D scenery

- 3D Interactive: Virtual environments (including 3D games)

- Others: ...

QI\_8 How would you assess the importance of sophisticated gaze visualization techniques for dynamic virtual environments?

*On a scale from [1-unimportant] to [5-fundamental]*

## 2. Qualitative Questions

- QII\_0 What are your experiences in designing and analyzing eye tracking experiments employing dynamic interactive stimuli?
- QII\_1 Based on the question QI\_6 about the importance of gaze visualizations, can you please elaborate in more detail, why you have this opinion?  
*Reference to QI\_6: "How important were visualizations for the analysis of such studies?"*
- QII\_2 Where do you see weaknesses in current gaze visualization techniques?
- QII\_3 Would you agree with the following comment: "So far, nobody has come across a simple and intuitive way to visualize gaze data for dynamic stimuli" [Šo8]? Please explain, why you would agree/disagree!
- QII\_4 What features would you desire for a simple and intuitive gaze analysis?

## BIBLIOGRAPHY

---

- [Ahl96] C. Ahlberg. “Cocktailmaps: a space-filling visualization method for complex communicating systems”. In *AVI '96: Proceedings of the workshop on Advanced visual interfaces*, pages 175–183. ACM, 1996. (Cited on page 51.)
- [AHR98] A. Aaltonen, A. Hyrskykari, and K.-J. Räihä. “101 spots, or how do users read menus?”. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 132–139. ACM Press/Addison-Wesley Publishing Co., 1998. (Cited on pages 22 and 25.)
- [BBIo7] B. P. Bailey, C. W. Busbey, and S. T. Iqbal. “TAPRAV: An interactive analysis tool for exploring workload aligned to models of task execution”. *Interact. Comput.*, 19(3):314–329, 2007. (Cited on pages 22 and 28.)
- [Bea82] J. Beatty. “Task-evoked pupillary responses, processing load, and the structure of processing resources”. *Psychological Bulletin*, 91(2):276–92, 1982. (Cited on page 28.)
- [BIo8] B. P. Bailey and S. T. Iqbal. “Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management”. *ACM Trans. Comput.-Hum. Interact.*, 14(4):1–28, 2008. (Cited on page 28.)
- [BMS<sup>+</sup>06] S. A. Balk, K. S. Moore, J. E. Steele, W. J. Spearman, and A. T. Duchowski. “Mobile phone use in a driving simulation task: Differences in eye movements”. *Vision Sciences Society (Posters)*, 6:872, 6 2006. (Cited on page 22.)
- [BOJ<sup>+</sup>07] O. Burgert, V. Örn, M. Joos, G. Strauß, C. Tietjen, B. Preim, and I. Hertel. “Evaluation of perception performance in neck dissection planning using eye-tracking”. In *SPIE Conference on Medical Image Computing*. SPIE, 2007. (Cited on pages 31, 37, and 59.)
- [BPF03] J. S. Babcock, J. B. Pelz, and M. D. Fairchild. “Eye tracking observers during rank order, paired comparison, and graphical rating tasks”. In *PICS*, pages 10–15, 2003. (Cited on pages 4, 22, and 37.)
- [Cas01] B. Cassin. “*Dictionary of Eye Terminology*”. Triad Publishing Company (FL), 4th spiral edition, July 2001. (Cited on pages 7 and 8.)

- [CMS99] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *“Readings in information visualization: using vision to think”*. Morgan Kaufmann Publishers Inc., 1999. (Cited on page 4.)
- [Co099] A. Cooper. *“The Inmates Are Running the Asylum”*. Macmillan Publishing Co., Inc., 1999. (Cited on page 42.)
- [CS94] R. Chimera and B. Shneiderman. “An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents”. *ACM Trans. Inf. Syst.*, 12(4):383–406, 1994. (Cited on page 62.)
- [CSD03] N. Cournia, J. D. Smith, and A. T. Duchowski. “Gaze- vs. hand-based pointing in virtual environments”. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 772–773. ACM, 2003. (Cited on page 12.)
- [DC01] Dodge and Cline. “The angle velocity of eye movements”. *Psychological Review*, 8:145–157, 1901. (Cited on page 8.)
- [DDGM00] R. Danforth, A.T. Duchowski, R. Geist, and E. McAliley. “A platform for gaze-contingent virtual environments”. In *In Smart Graphics (2000 AAAI Spring Symposium, Technical Report SS-00-04)*, (Menlo Park, CA, 2000), AAAI, pages 66–70, 2000. (Cited on page 13.)
- [DFW08] R. Dachsel, M. Frisch, and M. Weiland. “FacetZoom: a continuous multi-scale widget for navigating hierarchical metadata”. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1353–1356. ACM, 2008. (Cited on pages 53 and 54.)
- [DM98] A. T. Duchowski and B. McCormick. “Gaze-contingent video resolution degradation”. In *Human Vision and Electronic Imaging III*. SPIE, 1998. (Cited on pages 1, 13, 18, 36, and 52.)
- [DMC<sup>+</sup>02] A. T. Duchowski, E. Medlin, N. Cournia, A. Gramopadhye, B. Melloy, and S. Nair. “3d eye movement analysis for vr visual inspection training”. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 103–110. ACM, 2002. (Cited on pages 1, 22, 36, and 54.)
- [DMG<sup>+</sup>01] A. T. Duchowski, E. Medlin, A. Gramopadhye, B. Melloy, and S. Nair. “Binocular eye tracking in vr for visual inspection training”. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 1–8. ACM, 2001. (Cited on page 47.)

- [DSR<sup>+</sup>00] A. T. Duchowski, V. Shivashankaraiah, T. Rawls, A. K. Gramopadhye, B. Melloy, and B. Kanki. “Binocular eye tracking in virtual reality for inspection training”. In *ETRA*, pages 89–96, 2000. (Cited on pages 36, 47, and 54.)
- [Duco2] A.T. Duchowski. “A breadth-first survey of eye-tracking applications”. *Behavior Research Methods, Instruments, and Computers*, 34(4):455–470, November 2002. (Cited on pages 12 and 13.)
- [DWo6] R. Dachsel and M. Weiland. “TimeZoom: a flexible detail and context timeline. In *CHI '06: Extended abstracts on Human factors in computing systems*, pages 682–687. ACM, 2006. (Cited on pages 53 and 54.)
- [EPM<sup>+</sup>08a] I. Ekman, A. Poikola, M. Mäkäräinen, T. Takala, and P. Hämäläinen. “Voluntary pupil size change as control in eyes only interaction”. In *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 115–118. ACM, 2008. (Cited on page 12.)
- [EPMo8b] I. M. Ekman, A. W. Poikola, and M. K. Mäkäräinen. “Invisible eni: Using gaze and pupil size to control a game”. In *CHI '08: Extended abstracts on Human factors in computing systems*, pages 3135–3140. ACM, 2008. (Cited on pages 12 and 22.)
- [Gow07] T. Gowases. “Gaze vs. mouse: An evaluation of user experience and planning in problem solving games”. Department of computer science. Master’s thesis. University of joensuu, finland, 2007. (Cited on page 12.)
- [GPMS04] P. W. M. Van Gerven, F. Paas, J. J. G. Van Merriënboer, and H. G. Schmidt. “Memory load and the cognitive pupillary response in aging”. *Psychophysiology*, 41(2):167–174, 2004. (Cited on page 28.)
- [GSB<sup>+</sup>94] M. Guzdial, P. Santos, A. Badre, S. Hudson, and M. Gray. “Analyzing and visualizing log files: A computational science of usability”. Technical report, Presented at HCI Consortium Workshop, 1994. (Cited on pages 20 and 23.)
- [GSL<sup>+</sup>02] J. H. Goldberg, M. J. Stimson, M. Lewenstein, N. Scott, and A. M. Wichansky. “Eye tracking in web search tasks: design implications”. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 51–58. ACM, 2002. (Cited on page 45.)
- [HBC<sup>+</sup>96] T. T. Hewett, R. Baecker, S. Card, T. Carey, J. Gasen, M. Mantei, G. Perlman, G. Strong, and W. Verplank. “ACM SIGCHI Cur-

- ricula for human-computer interaction". <http://www.sigchi.org/cdg/cdg2.html>. (Last accessed Oct., 2008), 1996. (Cited on page 5.)
- [Hue68] E. Huey. *"The Psychology and Pedagogy of Reading"*. Cambridge, MA: MIT Press (originally published in 1908), 1968. (Cited on pages 8 and 32.)
- [IJvM09] P. Isokoski, M. Joos, O. Špakov, and B. Martin. "Gaze controlled games". *Information Society Journal*, 2009. (Cited on page 12.)
- [IKN98] L. Itti, C. Koch, and E. Niebur. "A model of saliency-based visual attention for rapid scene analysis". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998. (Cited on page 13.)
- [@in09] @interactive minds GmbH. <http://www.interactive-minds.com/de/eyetracking-software/sinn-builder>. Last visited January, 2009. (Cited on pages 1, 35, and 64.)
- [IZB04] S. T. Iqbal, X. S. Zheng, and B. P. Bailey. "Task-evoked pupillary response to mental workload in human-computer interaction". In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1477–1480. ACM, 2004. (Cited on page 28.)
- [Jö5] Erika Jönsson. "If looks could kill - An evaluation of eye tracking in computer games". Department of Numerical Analysis and Computer science, Master's thesis. Royal Institute of Technology, Stockholm, Sweden, 2005. (Cited on pages 7, 8, 12, and 70.)
- [Jac90] R. J. K. Jacob. "What you look at is what you get: eye movement-based interaction techniques". In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 11–18. ACM, 1990. (Cited on page 12.)
- [Jac91] R. J. K. Jacob. "The use of eye movements in human-computer interaction techniques: what you look at is what you get". *ACM Trans. Inf. Syst.*, 9(2):152–169, 1991. (Cited on pages 4 and 7.)
- [JCo8] L. Jie and J. J. Clark. "Video game design using an eye-movement-dependent model of visual attention". *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(3):1–16, 2008. (Cited on page 14.)
- [JK03] R. J. K. Jacob and K. S. Karn. "Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises" (Section Commentary). In book: *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, pages 573–605. Amsterdam, Elsevier Science, 2003. (Cited on pages 8 and 45.)

- [JS91] B. Johnson and B. Shneiderman. “Tree-Maps: a space-filling approach to the visualization of hierarchical information structures”. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 284–291. IEEE Computer Society Press, 1991. (Cited on page 54.)
- [KHS02] E. J. Keogh, H. Hochheiser, and B. Shneiderman. “An augmented visual query mechanism for finding patterns in time series data”. In *FQAS '02: Proceedings of the 5th International Conference on Flexible Query Answering Systems*, pages 240–250. Springer-Verlag, 2002. (Cited on pages 29 and 54.)
- [KKH08] J. Klingner, R. Kumar, and P. Hanrahan. “Measuring the task-evoked pupillary response with a remote eye tracker”. In *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 69–72. ACM, 2008. (Cited on page 28.)
- [LAKL04] B. Law, M. S. Atkins, A. E. Kirkpatrick, and A. J. Lomax. “Eye gaze patterns differentiate novice and experts in a virtual laparoscopic surgery training environment”. In *ETRA '04: Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 41–48. ACM, 2004. (Cited on page 14.)
- [LL02] S. Lessing and L. Linge. “IICap: A new environment for eye tracking data analysis”. Master’s thesis. University of Lund, Sweden, 2002. (Cited on pages 1, 22, 29, 33, 35, 38, and 52.)
- [Mac95] A. M. MacEachren. *How Maps Work: Representation, Visualization and Design*. Guilford Press, 1995. (Cited on page 51.)
- [Mod97] D. Modjeska. “Navigation in electronic worlds: A research review”. Technical report, Computer Science Institute of the University of Toronto, 1997. (Cited on page 67.)
- [MS03] W. Müller and H. Schumann. “Visualization for modeling and simulation: visualization methods for time-dependent data - an overview”. In *WSC '03: Proceedings of the 35th conference on Winter simulation*, pages 737–745. Winter Simulation Conference, 2003. (Cited on page 51.)
- [MT05] T. Möller and B. Trumbore. “Fast, minimum storage ray/triangle intersection”. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 7. ACM, 2005. (Cited on page 76.)
- [MTNK02] C. Mello-Thoms, C. F. Nodine, and H. L. Kundel. “What attracts the eye to the location of missed and reported breast cancers”. In *ETRA*

- '02: *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 111–117. ACM, 2002. (Cited on pages 22 and 31.)
- [@My07] F. @My. <http://www.fairyengine.com/articles/xnainform.htm>. Last visited March, 2009, 2007. (Cited on pages 73 and 74.)
- [NH07] M. Nyström and K. Holmqvist. “Deriving and evaluating eye-tracking controlled volumes of interest for variable-resolution video compression”. *J. Electron. Imaging*, 16(1), 2007. (Cited on pages 13, 35, and 52.)
- [Nie04] L. Nielsen. “*Engaging Personas and Narrative Scenarios*”. PhD thesis, Department of Informatics, Copenhagen Business School, October 2004. (Cited on page 41.)
- [NLS08] L. Nacke, C. Lindley, and S. Stellmach. “Log who’s playing: Psychophysiological game analysis made easy through event logging”. In *Second International Conference on Fun and Games, Eindhoven, The Netherlands*, 2008. (Cited on page 89.)
- [NNB<sup>+</sup>04] S. G. Nikolov, T. D. Newman, D. R. Bull, N. C. Canagarajah, M. G. Jones, and I. D. Gilchrist. “Gaze-contingent display using texture mapping and opengl: system and applications”. In *ETRA '04: Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 11–18. ACM, 2004. (Cited on page 18.)
- [PA06] J. Pruitt and T. Adlin. “*The Persona Lifecycle*”. Morgan Kaufman, Elsevier, 2006. (Cited on page 42.)
- [RAM<sup>+</sup>05] K.-J. Räihä, A. Aula, P. Majaranta, H. Rantala, and K. Koivunen. “Static visualization of temporal eye-tracking data”. In *Proceedings of IFIP INTERACT05: Human-Computer Interaction*, pages 946–949. Springer Verlag, 2005. (Cited on pages 30, 31, 32, and 35.)
- [RCPS01] E. M. Reingold, N. Charness, M. Pomplun, and D. M. Stampe. “Visual span in expert chess players: evidence from eye movements”. *Psychological Science*, 12:48–55, 2001. (Cited on page 14.)
- [RTSB04] R. Ramloll, C. Trepagnier, M. Sebrechts, and J. Beedasy. “Gaze data visualization tools: opportunities and challenges”. *Eighth International Conference on Information Visualisation*, pages 173–180, July 2004. (Cited on pages 1, 5, 9, 18, 22, 31, and 32.)
- [SAC<sup>+</sup>07] C. Sennersten, J. Alfredson, M. Castor, J. Hedström, B. Lindahl, C. Lindley, and E. Svensson. “Verification of an experimental platform integrating a Tobii eyetracking system with the HiFi game

- engine”, 2007. Methodology Report. (Cited on pages 5, 15, 25, 47, 49, and 70.)
- [Saso8] D. Sasse. “A framework for psychophysiological data acquisition in digital games”. Department of Simulation and Graphics, Master’s thesis. Otto-von-Guericke-University Magdeburg, Germany, 2008. (Cited on page 15.)
- [SDTF03] M. Schießl, S. Duda, A. Thölke, and R. Fischer. “Eye tracking and its application in usability and media research”. *Sonderheft: Blickbewegung in MMI-interaktiv Journal*, 6, 2003. (Cited on pages 4, 22, and 35.)
- [Sen04] C. Sennersten. “Eye Movements in an Action Game Tutorial”. Department of Cognitive Science, Master’s thesis. Lund University, Sweden, 2004. (Cited on pages 22 and 70.)
- [SG06] J. David Smith and T. C. Nicholas Graham. “Use of eye movements for video game control”. In *ACE ’06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 20. ACM, 2006. (Cited on page 12.)
- [SGGD05] S. Sadasivan, J. S. Greenstein, A. K. Gramopadhye, and A. T. Duchowski. “Use of eye movements as feedforward training for a synthetic aircraft inspection task”. In *CHI ’05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 141–149. ACM, 2005. (Cited on pages 18, 22, 36, and 54.)
- [Shn92] B. Shneiderman. “Tree visualization with tree-maps: 2-d space-filling approach”. *ACM Trans. Graph.*, 11(1):92–99, 1992. (Cited on page 54.)
- [Shn96] B. Shneiderman. “The eyes have it: A task by data type taxonomy for information visualizations”. *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343, Sept 1996. (Cited on pages 20, 24, 28, 29, 35, 38, 66, and 68.)
- [@Sio9] @SimileTimeline. <http://simile.mit.edu/timeline/>. Last visited March, 2009. (Cited on page 53.)
- [SL08] C. Sennersten and C. Lindley. “Evaluation of real-time eye gaze logging by a 3D game engine”. In *12th IMEKO TC1 & TC7 Joint Symposium on Man Science & Measurement*, 2008. (Cited on pages 5 and 15.)

- [SMIo8] SMI. “SMI BeGaze Flyer”. Technical report, SensoMotoric Instruments GmbH (SMI), 2008. (Cited on pages 1 and 33.)
- [SNo8] A. Sutcliffe and A. Namoune. “Getting the message across: visual attention, aesthetic design and what users remember”. In *DIS '08: Proceedings of the 7th ACM conference on Designing interactive systems*, pages 11–20. ACM, 2008. (Cited on pages 4, 22, and 35.)
- [SOOo8] L. Skrba, I. O’Connell, and C. O’Sullivan. “Eye-tracking dynamic scenes with humans and animals”. In *APGV '08: Proceedings of the 5th symposium on Applied perception in graphics and visualization*, pages 199–199. ACM, 2008. (Cited on page 14.)
- [Spa06] J. G. Spanne. “Task Impact on Cognitive Processing of Narrative Fiction Film”. Department of Cognitive Science, Master’s thesis. Lund University, Sweden, 2006. (Cited on page 22.)
- [Steo7] S. Stellmach. “A psychophysiological logging system for a digital game modification”. Department of Simulation and Graphics, Bachelor thesis. Otto-von-Guericke-University Magdeburg, Germany, 2007. (Cited on pages 5, 15, 47, 70, and 89.)
- [SWF<sup>+</sup>04] B. Spence, M. Witkowski, C. Fawcett, B. Craft, and O. de Bruijn. “Image presentation in space and time: errors, preferences and eye-gaze activity”. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 141–149. ACM, 2004. (Cited on page 22.)
- [SWM<sup>+</sup>08] W. Steptoe, R. Wolff, A. Murgia, E. Guimaraes, J. Rae, P. Sharkey, D. Roberts, and A. Steed. “Eye-tracking for avatar eye-gaze and interactional analysis in immersive collaborative virtual environments”. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 197–200. ACM, 2008. (Cited on page 12.)
- [TJ00] V. Tanriverdi and R. J. K. Jacob. “Interacting with eye movements in virtual environments”. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM, 2000. (Cited on page 12.)
- [Tob04] Tobii Technology AB. “Product Description, Tobii 50 Series”, 2004. (Cited on pages 10 and 11.)
- [Tob05] Tobii Technology AB. “User’s Guide to Tobii Programming Interfaces”, 1. edition, 2005. (Cited on pages 9, 10, 11, 71, and 77.)

- [Tobo6] Tobii Technology AB. “*User Manual: Tobii Eye Tracker - ClearView analysis software*”, 2006. (Cited on page 71.)
- [Tobo8] TobiiStudio. “Tobii studio 1.2 release notes”. Technical report, Tobii Technology AB, 5 2008. (Cited on page 31.)
- [VH96] B. M. Velichkovsky and J. P. Hansen. “New technological windows into mind: there is more in eyes and brains for human-computer interaction”. In *CHI '96*, pages 496–503. ACM, 1996. (Cited on page 33.)
- [VNKJo8] A. Voßkuehler, V. Nordmeier, L. Kuchinke, and A. M. Jacobs. “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):1150–1162, 2008. (Cited on page 22.)
- [Šo8] O. Špakov. “iComponent - Device-independent platform for analyzing eye movement data and developing eye-based applications”. Department of Computer Sciences, Dissertation. University of Tampere, Finland, 2008. (Cited on pages 5, 18, 20, 31, 32, and 92.)
- [@Wio9] @Wikipedia. <http://en.wikipedia.org/wiki/XML>. Last visited March, 2009. (Cited on pages 77 and 78.)
- [WM87] C. Ware and H.T. Mikaelian. “An evaluation of an eye tracker as a device for computer input”. In *ACM CHI + GI'87 Human Factors in Computing Systems Conference*, pages 183–188. ACM, 1987. (Cited on page 12.)
- [Woo02] D. S. Wooding. Fixation maps: quantifying eye-movement traces. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 31–36. ACM, 2002. (Cited on pages 1, 4, 22, 33, 34, 35, 37, 58, and 60.)
- [Xu00] A. Xu. “Eye tracking study on identifying and analyzing user behavior”. Master’s thesis. University of North Carolina at Chapel Hill, USA, 2000. (Cited on page 22.)
- [Yar67] A. L. Yarbus. “*Eye movements and vision*”. New York Plenum, 1967. (Cited on pages 1, 13, and 32.)
- [YJSHo8] Y. Yesilada, C. Jay, R. Stevens, and S. Harper. “Validating the use and role of visual elements of web pages in navigation with an eye-tracking study”. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 11–20. ACM, 2008. (Cited on page 4.)

## DECLARATION

---

This is to certify that I alone compiled and completed this thesis using only the resources allowed. Quotations and references are indicated accordingly.

*Magdeburg, Germany / Karlshamn, Sweden, April, 2009*

---

Sophie Stellmach