Generic Gaze Interaction Events for Web Browsers

Using the Eye Tracker as Input Device

Benjamin Wassermann
Media University Stuttgart
Mobile Media Group
Nobelstraße 10
Stuttgart, Germany
wassermann@hdm-stuttgart.de

Adrian Hardt
Eberhard Karls University Tübingen
Wilhelm-Schickard-Institute
Sand 14
Tübingen, Germany
a.hardt@student.uni-tuebingen.de

Gottfried Zimmermann
Media University Stuttgart
Mobile Media Group
Nobelstraße 10
Stuttgart, Germany
gzimmermann@hdm-stuttgart.de

ABSTRACT

In the last decade much research has been conducted on analyzing human eye and gaze movements using eye tracking technology, not only in the fields of neuroscience, psychology and marketing, but also in the field of human computer interaction. However, no flexbile framework exists to integrate eye tracking directly into web applications to easily create and test new interaction concepts. We have created a JavaScript library based on the latest HTML5 Web technology and the jQuery library to close this gap. Facilitated by HTML5 WebSocket, the browser directly receives gaze input samples from an eye tracker, generating events that are similar to those of a mouse input device. Events like *gazeOver/-Out* or *fixationStart/-End* can be attached to any HTML element in the DOM tree. New custom events derived from the eye tracking data, e.g. *blink* or *read*, can easily be added. Using this library we have successfully implemented a number of Web applications, allowing the users to interact with their eyes. This paper also describes our gaze enabled Web-based eLearning environment. Our JavaScript library is used within the eLearning environment to capture and interpret eye gaze events for the purpose to support users in the acquisition of new knowledge.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Input devices and strategies, Interaction styles - gaze; H.5.4 [**Hypertext/Hypermedia**]: Architectures, Navigation

General Terms

Human Factors, Design

Keywords

eye tracking, gaze events, gaze based interaction, gaze-enhanced web, gaze-enhanced eLearning

1. INTRODUCTION

Since the establishment of multi-touch devices in the mass market the acceptance and experimentation for new interaction designs has grown rapidly [24]. An upcoming approach is to model events by user intent in interaction (called *Indie UI* or former *Intentional Events*)[3, 4]. These technologies are envisioned for interacting with applications independent of input devices, giving the user the opportunity to choose their preferred input devices or allow people with disabilities to interact with assistive technologies. One of those technologies could be eye tracking, since it is continuously improving. In the near future this technology could reach the mass market, and eye gazing could be used as a means of user input. Based on this technology, using the gaze as input information, new interaction concepts are developed.

The *ScienceCampus* Tübingen [16] by the *Knowledge Media Research Center* [7] in Tübingen maintains a research line on the *Design of Interactive Informational Environments.* In the context of the pertaining project *Adaptable and Adaptive Multimedia Systems*, we are interested in new interaction designs that support users in the acquisition of new knowledge and the pertaining regulation processes. We have created a dynamic and adaptive learning platform based on the eLearning Web platform ILIAS [13] and the JavaScript-library jQuery [20], and have included eye tracking as an input device for the eLearning environment.

This approach allows us to acquire a user's eye gaze information in real-time that can be used to generate generic (i.e. not specific to a specific eye tracking device) gaze events, to analyse the data to deduce more higher-level events or to derive predictions about the user's condition, e.g. if the user is bored or overstrained by the learning content.

Before describing our concept of the generic gaze interaction events, we will give an overview of related work in the field of eye tracking technology. After that we will introduce our framework, enabling us to transfer the gaze data from the eye tracker to the Web browser, followed by our concept about gaze interaction events. We will then describe our eLearning environment, and discuss using eye tracking as an input device. Finally, we will give a short conclusion and an outlook on future work.

2. RELATED WORK

Recently, eye tracking technology has grown mature. It has not only been used for many years for studies in the area of image scanning [15], eye movement while driving [14], reading [18] and solving arithmetic tasks [27], but also to build interactive systems using eye tracking as an input device [9, 10, 26]. Jacob was one of the first in introducing gaze-based interaction techniques. He discussed gaze interactions such as object selection, object movement and eye-controlled scrolling for text [10]. He also described the "Midas Touch" problem which refers to a user's need to confirm their actions in order to interact with objects. To overcome this problem he introduced dwell-based activation. The gaze of the user has to remain over an object for a fixed time before it is activated. Since this dwell-time has an inherent latency, Jacob suggested allowing users to confirm their actions instantly using a keyboard as alternative.

A well-known problem is the accuracy of eye tracker systems. Jacob, for example, has overcome this problem by using sufficiently large targets for his experiments. He showed in his work that the eye can provide a pointing accuracy of approximately one degree [10]. Ware and Mikaelian [28] conducted studies to analyze different types of selection methods with eye tracking, also taking target size into account. Their results showed that eye selection can be faster than mouse selection, provided that the target has a moderate size. Since then researchers have come up with many different approaches for solving the problems of selecting and activating objects, even with low accuracy of the eye tracking device.

In classical approaches the eye movement is typically analyzed in terms of fixations and saccades. A fixation is a longer gaze over informative regions of interest while the saccades are rapid movements between fixations. Typical metrics of eye tracking analysis includes fixation or gaze durations, saccadic velocities and amplitudes and transition-based parameters, e.g. between fixations and regions of interest [23]. By tracing these fixations one can analyze cognitive protocols [21, 22], since it is generally agreed upon that visual and cognitive processing occur during fixations [11]. However, the determination of fixations is not clear, because there is space for interpretation about when a fixation starts and when it ends.

Eye tracking can be used as input device and connected to an application in two ways: First, by a link between the eye tracker system and the application, and second by having the eye tracker emulate the pointing device of the operating system. In the latter case, the cursor is controlled by the eye tracker, preventing in some cases the simultaneous usage of eye tracking and mouse. Drewes [5] addressed this problem by integrating eye tracking as additional interaction modality into a Web browser by using a special proxy, thus creating a powerful experimentation platform. Biedert [2] used a more flexible approach by embedding a Java applet to transfer eye movement data from the eye tracker into the Web browser. Drewes [6] also made first investigations for integrating eye tracking into eLearning environments. The eye movements bear more information than simple interaction events, and can therefore support personalization of the learning content and user interface.

3. FRAMEWORK

The framework consists of a communication adapter between Web browser and eye tracker as shown in figure 1. To acquire data from the eye tracker system, a proprietary communication API is provided. Since the Web browser does not support proprietary eye tracker APIs, an adapter is required. This raises the question: Which kind of communication technology, supported by the Web browser, should be used to transfer the eye tracker data to the Web browser?

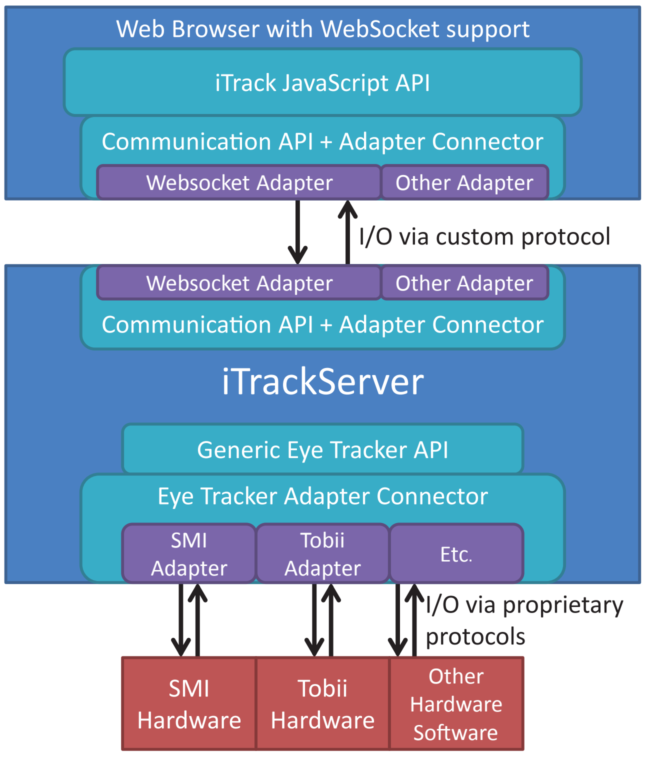


Figure 1: This graphic shows the architecture of our framework. The Web browser is connected to the iTrackServer using the Communication API with the WebSocket Adapter and the custom protocol to communicate with the iTrackServer. By using the JavaScript API iTrack, Web applications within the Web browser can access the data from the eye tracker. The iTrackServer uses the Generic Eye Tracker API to communicate with the eye tracker hardware. Miscellaneous eye tracker hardware can be used via the Eye Tracker Adapters.

To answer this question we first have to take into account that we want the eye tracker data in real-time, which implies an active communication from the server to the client with low latency. A traditional client-server request does not seem appropriate since a client always has to start a request to the server before receiving a reply from the server. There are some existing advanced techniques like reverse AJAX to circumvent this problem, but the latency is still quite high. A better solution could be the use of browser plugins, but this induces high implementation costs when multiple browsers have to be supported. Another possibility are embedded objects like flash or Java applets to communicate with external applications. This has been successfully demonstrated by the Text 2.0 Framework [2]. A drawback of this solution is the inclusion of alien objects as intermediates into the browser, thus limiting the flexibility in terms of configuration setups and security issues.

Therefore we propose to use HTML5 WebSockets [8] as communication means between an eye tracker and the Web browser. One reason for integrating WebSockets into browsers was to enable direct bidirectional communication between clients and servers with low latency. Optimal requirements for the transfer of data between an eye tracker and a browser. Another benefit is that the WebSocket protocol handles the security issues of network connections and data transfer. Since the HTML5 WebSocket API is a new web technology, it is not supported by old browsers. However, since many Web browsers are supporting WebSockets in their latest version [29], most people will have access to this technology.

An adapter is needed to bridge between the proprietary API of an eye tracker system and the WebSocket API as shown in the ITrackServer part of figure 1. Since we have an interprocess communication or even a network communication, we need to define a simple custom protocol for transferring eye movement data and properties from the eye tracker to the Web browser and vice versa remote commands in order to control the eye tracker from the Web browser. All data is passed on in the JSON format, because it can be efficiently parsed by Web browsers and in many programming languages. This protocol has been used as sub protocol for the WebSocket API, but it can also be used for any other communication technology between an application and the eye tracker. This keeps our approach flexible enough to adapt for future Web technologies, and allows other applications to connect to the eye tracker, e.g. by using a TCP network connection.

Given the custom protocol used as WebSocket sub protocol, adapters between a Web browser and any eye tracker system can be implemented. Our reference implementation is able to mediate between Web browsers and the SMI RED eye tracking system [25]. It consists of a separate application called iTrackServer. The iTrackServer was implemented in C++ and is using modular and object oriented concepts to easily extend the application by new eye tracker hardware.

4. GAZE INTERACTION EVENTS

Modern Web applications with dynamic user interfaces heavily rely on JavaScript and event driven design. Every HTML element in the source of a Website can be connected to event handlers. An event handler is a function that is called if a specific event is triggered. For example, one can connect a click handler to a button, so that the handler is called when the button is clicked with the mouse [17].

Every Web browser supports a standard set of possible events (with some variations between Web browsers though). They can be roughly classified into two groups: input events and system events. The input events are typically composed of mouse and keyboard interactions, while the system events are triggered from the Web browser, e.g. when the download of a Website has finished or the focus of an element has changed.

A mouse is an input device that transfers the movement of the hand of a user to the computer. Appropriate movements are applied by the computer to the mouse pointer on the screen. In combination with the mouse buttons, this enables the user to select or activate elements on the screen by moving the mouse pointer to a specific location and pressing a mouse button. A similar behavior can be considered for the eye gaze of a user. The movement of the pupils of the eyes of a user is recognized by the eye tracker and transformed into the screen position the user is looking at. It seems reasonable to consider both interactions as congruent in respect to the cursor movement.

Mouse events are among the most common events occurring in Web browsers, including *click*, *dbclick*, *mousedown*, *mouseup*, *mouseover*, *mousemove* and *mouseout* events and additionally the *mouseenter* and *mouseleave* events. The first four events are triggered if a user pushes a mouse button. The events *mousedown* and *mouseup* allow a more sophisticated way to recognize click events, e.g. for the support of Drag & Drop. The remaining events are triggered if the mouse pointer changes its position. The events *mouseenter* and *mouseleave* are separately listed, because they do not belong to the W3C Standard. We propose to use similar events for eye tracking and therefore we have defined following equivalent movement events: *gazeOver*, *gazeMove*, *gazeOut*, *gazeEnter* and *gazeLeave*. We have also added the *gazeEnter* and *gazeLeave* events for the sake of the Web developers' habits.

Despite all the similarities between the mouse input device and the eye tracker, there are still many fundamental differences. First, there is a difference in the accuracy between the two devices. A standard mouse device has a very high accuracy as opposed to the eye tracker, which is limited by the hardware used. There are additional factors influencing the accuracy of the eye tracker. One factor is the calibration of the system that varies from session to session. Without a calibration, only gross predictions of the gaze can be derived from the available data. Another problem is the position of the user: it changes over time, which also influences the accuracy of the tracked data. We also have to consider that the eye tracker produces a continuous stream of eye data samples, while the mouse device only produces data if the mouse has really been moved. This results in a continuous triggering of the *gazeMove* event and under some circumstances in a poor performance of Web applications. Finally, a mouse provides click events. The eye tracker does not have any immediate counterpart for these interactions. However, approaches exist, to use eye blinks or gaze fixations in combination with dwell-time to simulate a click event[10]. We have added fixation events (*fixationStart/-End*) as more higher level events that can be derived from the basic gaze samples to simulate click events.

4.1 Implementation

Since we are able to receive eye tracker data samples from an eye tracker within the browser in real-time by using the WebSocket technology, we can use the data to generate interaction and custom events. For accessing the data in the Web browser we have created the JavaScript library iTrack. The library is composed of three subparts. First, a communication interface similar to the one of the iTrackServer has been created. With this interface JSON messages can be sent and received via WebSocket.

The second subpart is the eye tracker controller. It can open a connection to an eye tracker via the communication interface and interpret the custom protocol. Via the protocol the controller can remotely control the eye tracker, activating the eye sample streaming or setting custom properties of the eye tracker. The controller provides an event interface to connect any function to a basic set of events: *eyeUpdate*, *fixationStart/-End* and *propertyChange*. The first two events are triggered when a new eye sample or a detected fixation from the eye tracker arrives. The remaining event is triggered if some configuration, state or property of the eye tracker has changed.

The last subpart is the event engine which generates the basic set of gaze events: *gazeOver*, *gazeMove*, *gazeOut*, *gazeEnter*, *gazeLeave* and *fixationStart/-End*. The event engine is connected to the eye tracker controller events and uses the received data to generate the events on object-level. To reduce the implementation overhead and to be cross-browser compatible, the event system of the jQuery JavaScript library was used to realize the generic eye tracker events. Thus we can easily trigger these events on any element in the DOM tree. The event handling is the same procedure as for all other native browser events using the jQuery event system.

Since the gaze positions provided by the eye tracker are global screen coordinates, an additional adaptation is required if the browser is not running in full screen mode to convert the eye tracker coordinates to the client coordinates. After the coordinate adaption is applied, the JavaScript function *elementFromPoint* can be used to find the underlying element of the given position. Using this process, the event engine is able to trigger the corresponding gaze events for the given element.

4.2 Client Coordinate Evaluation

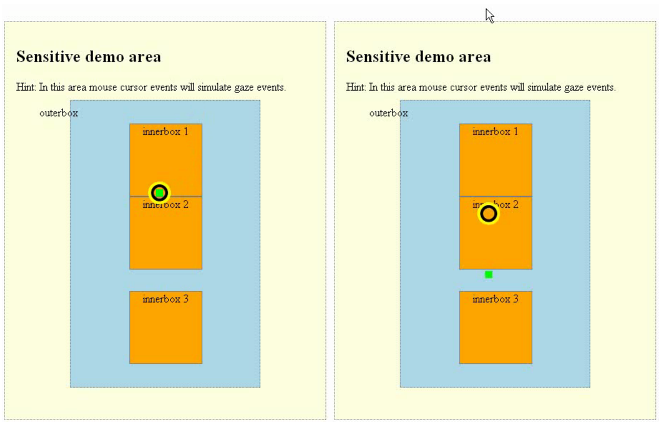

Sensitive demo area
Hint: In this area mouse cursor events will simulate gaze events.
outerbox
innerbox 1
innerbox 2
innerbox 3
Sensitive demo area
Hint: In this area mouse cursor events will simulate gaze events.
outerbox
innerbox 1
innerbox 2
innerbox 3

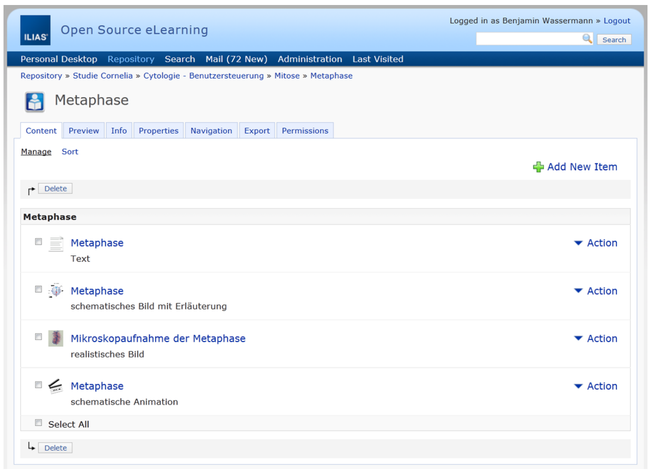
Figure 2: The left picture shows the gaze visualization with adapted screen coordinates, the right picture without. The green rectangle is visualized by the Web browser in client coordinates and the circle by the SMI Experiment Center in global screen coordinates.

To apply the client coordinate adaptation, the screen position of the client window, containing the content of the Webpage, has to be known. Until now, no native JavaScript function exists to acquire this information.To overcome this problem we used a mouse button click event. The mouse button click event provides as parameters the click position as both screen and client coordinates. Combining this information with the page offset (for scrolling), we can exactly compute the screen coordinates of the content window of the Website.

To evaluate the correctness of the client coordinate adaptation, we have visualized the position of the gaze in the browser and the screen position by using the Experiment Center tool from the SMI eye tracker system as shown in figure 2. The experiment has shown that the screen position visualized by Experiment Center exactly corresponds to the client position visualized by the Web browser.

5. ADAPTIVE LEARNING ENVIRONMENT

We have built an adaptive eLearning environment extension that is based on the open source eLearning platform ILIAS [13]. This extension allows us to conduct the planned empirical studies within the *Adaptable and Adaptive Multimedia Systems* project. The extension is divided into two parts: The authoring environment for the creation and editing of the content for the lessons and the learning environment for the presentation of the created lessons to the learner.



Repository » Studie Cornelia » Cytologie - Benutzersteuerung » Mitose » Metaphase

Metaphase

Metaphase

Metaphase
Text

Metaphase
schematisches Bild mit Erläuterung

Mikroskopaufnahme der Metaphase
realistisches Bild

Metaphase
schematische Animation

Figure 3: This picture shows the integrated authoring environment in ILIAS.

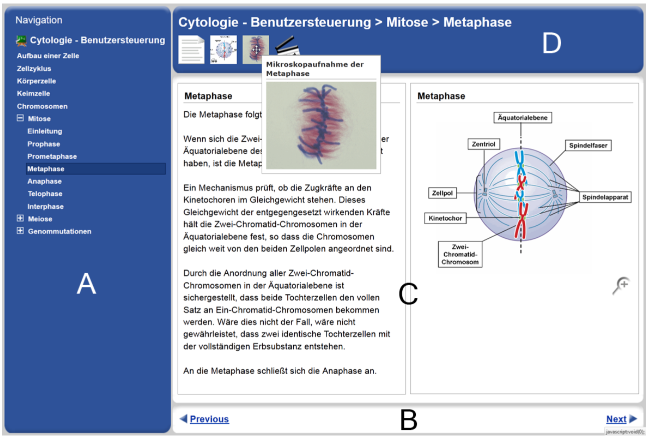
The authoring environment is directly built into the ILIAS platform. It uses the same concepts for creating and manipulating content as ILIAS for its own content objects. This reduces the barrier for authors that are familiar with the ILIAS platform. The authoring environment allows for the creation of modular lessons or learning units based on text, image, audio and video media types as shown in figure 3. A learning unit can contain an arbitrary number of content objects. Every learning unit additionally contains a set of properties for customization (e.g. choosing templates or selecting default content). To manage multiple lessons, the integrated administrative objects of ILIAS, such as categories, folders and courses, can be used. For exchange between local systems, import and export functionality has been added.

The learning environment or learning view presents the content to the user. The layout is subdivided into three parts:

Navigation area Depending on the chosen template and settings of the author, a linear navigation (see B of figure 4) and a navigation tree (see A of figure 4) are displayed.

Content area This area is composed of slots or subareas where the content objects of the learning unit are displayed (see C of figure 4). Depending on the author‘s settings, different kinds of layout with a varying number of slots can be chosen.

Media shelf If not deactivated by the author, the media shelf shows all content objects of a learning unit as small icons (see D of figure 4). The *mouseover* event will show a preview of the content. Using Drag & Drop technology, an icon can be dragged to any slot in the content area and dropped to dynamically place the content into the slot.



Die Metaphase folgt

Wenn sich die Zwei- er
Äquatorialebene de t
haben, ist die Metap

Ein Mechanismus prüft, ob die Zugkräfte an den Kinetochoren im Gleichgewicht stehen. Dieses Gleichgewicht der entgegengesetzt wirkenden Kräfte hält die Zwei-Chromatid-Chromosomen in der Äquatorialebene fest, so dass die Chromosomen gleich weit von den beiden Zellpolen angeordnet sind.

Durch die Anordnung aller Zwei-Chromatid-Chromosomen in der Äquatorialebene ist sichergestellt, dass beide Tochterzellen den vollen Satz an Ein-Chromatid-Chromosomen bekommen werden. Wäre dies nicht der Fall, wäre nicht gewährleistet, dass zwei identische Tochterzellen mit der vollständigen Erbsubstanz entstehen.

An die Metaphase schließt sich die Anaphase an.

Figure 4: This picture shows the learning environment of the ILIAS plugin. A: Navigation tree; B: Linear navigation; C: Content area; D: Media shelf

The look and feel of the learning view can be adapted by templates supplied by authors. The view is implemented as Web application, dynamically loading the content via AJAX technology from the server. We have added eye tracker support by including our iTrack JavaScript library to the presentation view. Additionally tools for creating areas of interest within images and the DOM tree have been implemented. In combination with some analytic tools complex interaction relationships can be easily modeled. Thus the project can now conduct its psychological experiments, analyzing to what extent the eye tracking can be used to give the user feedback on relationships between different content elements, and to provide explicit hints about information the user may have missed.

6. DISCUSSION

As mentioned ealier, there are justifications for the use of an eye tracker as an input device, at least for people with disabilities, because this technology is one of the few that can enable people with disabilities to interact with the computer. Since the eye tracker technology is currently still too expensive for the mass market (including most users with disabilities), the main application area is in (market) research. However, plans of companies producing an eye tracker system already exist to prepare this technology for the mass market by installing small eye tracker systems into normal notebooks [12, 19]. A further development can be found in free software that converts low cost hardware (e.g. Web cam) into eye tracker systems [1]. Unfortunately, the current low cost eye trackers suffer from low precision, much worse than the professional systems.

In combination with an eLearning environment, eye tracking can certainly offer some benefits [6]. It can be used not only as an alternative for the mouse input in terms of Indie UI, but can also provide information about the learning condition of a user. Such information may include information on the cognitive load, boredom or learning progress of a user. First attempts in this direction are currently being researched in parallel projects.

7. CONCLUSION AND FUTURE WORK

Using eye tracker systems as input devices provides new possibilities for the creation of new interaction designs for applications. Eye tracking can substitute classical pointing devices like a mouse to some extent, but also requires new interaction patterns. In this paper we have presented a framework that allows us to use an eye tracker system as input device for Web sites. Thus we can easily create new Web applications using new interaction design concepts. An adaptive eLearning environment was developed based on the eLearning platform ILIAS, including our framework. It is possible to let the user interact with the eLearning environment by eyes, and to derive more complex information about the user by analyzing the eye movement.

We are planning to extend our framework by supporting other eye tracker systems and adding a filter interface to the iTrackServer to apply and test different approaches of precision optimization algorithms for the eye tracker data and to search for patterns indicating special events. Additionally, we want to extend the iTrack JavaScript library to support calibration and validation within the Web browser. No external application for calibration will be required, and recalibration within a session can easily be performed without changing the application. As an additional side effect, the screen to client coordinate adaptation is not required anymore, since the eye tracker will be calibrated to the browser window and not to the whole screen.

In future studies we aim to find patterns and indicators in the eye tracker data to predict more sophisticated events like *read* or *search*, allowing new interactions, e.g. for auto scrolling while reading a document. We also plan to use machine learning techniques to find interaction patterns of users (also using mouse or keyboard input), which we could possibly use for user interface adaptations or for predicting some kind of user state, e.g. boredom or cognitive overload. Finally, we plan to analyze the usage of eye tracking for Indie UI.

8. ACKNOWLEDGEMENT

This work has been funded by the ScienceCampus of the Knowledge Media Research Center Tübingen [16].

References

[1] J. S. Agustin, H. Skovsgaard, E. Mollenbach, M. Barret, M. Tall, D. W. Hansen, and J. P. Hansen. Evaluation of a low-cost open-source gaze tracker. In *Symposium on Eye-Tracking Research & Applications*, pages 77–80, New York, 2010. ACM Press.

[2] R. Biedert, G. Buscher, S. Schwarz, M. Möller, A. Dengel, and T. Lottermann. The text 2.0 framework. In *Workshop on Eye Gaze in Intelligent Human Machine Interaction*, pages 114–117, New York, 2010. ACM Press.

[3] M. Cooper. Intentional events working group charter. `http://www.w3.org/2011/08/intentional-events-charter`, Nov. 2011.

[4] M. Cooper. Indie ui working group charter. `http://www.w3.org/2011/11/indie-ui-charter`, Feb. 2012.

[5] H. Drewes. *Eye Gaze Tracking for Human Computer Interaction.* Doctoral Dissertation, Media Informatics Group, Ludwig-Maximilians-Universität München, Germany, 2010.

[6] H. Drewes, R. Atterer, and A. Schmidt. Detailed monitoring of user's gaze and interaction to improve future e-learning. In *Conference on Universal access in Human-Computer Interaction: Ambient Interaction*, pages 802–811, Heidelberg, 2007. Springer-Verlag.

[7] F. Hesse and S. Schwan. Knowledge media research center. `http://www.iwm-kmrc.de/www/en/index.html`, Nov. 2011.

[8] I. Hickson. The websocket api. `http://dev.w3.org/html5/websockets/`, Nov. 2011.

[9] T. E. Hutchinson, K. P. White, W. N. Martin, K. C. Reichert, and L. A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:1527–1534, 1989.

[10] R. T. K. Jacob. The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Transactions on Information Systems (TOIS)*, 9:152–169, 1991.

[11] M. A. Just and P. A. Carpenter. A theory of reading: From eye fixations to comprehension. *Psychological Review 87*, pages 329–354, 1980.

[12] O. Kharif. Eye-tracking technology for the masses. *Bloomberg Businessweek*, Mar. 2011.

[13] M. Kunkel. Ilias open source e-learning. `http://www.ilias.de`, Nov. 2011.

[14] M. F. Land and D. N. Lee. Where we look when we steer. *Nature*, (369):742–744, 1994.

[15] D. Noton and L. Stark. Scanpaths in saccadic eye movements while viewing and recognizing patterns. *Vision Research*, 11:929–942, 1971.

[16] S. Pfeiffer. Sciencecampus. `http://www.wissenschaftscampus-tuebingen.de/www/en/index.html?ref=folder5`, Nov. 2011.

[17] T. Pixley. Document object model events. `http://www.w3.org/TR/DOM-Level-2-Events/events.html`, Nov. 2011.

[18] K. Rayner. Eye movements and cognitive processes in reading, visual search, and scene perception. *In J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), Eye Movement Research: Mechanisms, Processes, and Applications*, pages 3–21, 1995.

[19] redOrbit.com. Tobii unveils eye-tracking laptops. `http://www.redorbit.com/news/technology/2004501/tobii_unveils_eyetracking_laptops/index.html`, Mar. 2011.

[20] J. Resig. jquery. `http://jquery.com`, Nov. 2011.

[21] D. D. Salvucci. *Mapping eye movements to cognitive processes.* Doctoral Dissertation, Department of Computer Science, Carnegie Mellon University, 1999.

[22] D. D. Salvucci and J. R. Anderson. Tracing eye movement protocols with cognitive process models. In *Conference of the Cognitive Science Society*, pages 923–928, Hillsdale, N.J., 1998. L. Erlbaum.

[23] D. D. Salvucci and J. H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Symposium on Eye Tracking Research and Applications*, pages 71–78, New York, 2000. ACM Press.

[24] J. Schöning, A. Krüger, and P. Olivier. Multi-touch is dead, long live multi-touch. In *Workshop on Multi-touch and Surface Computing*, pages 1–5, New York, 2009. ACM Press.

[25] SMI. Red eye tracker. `http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/red-red250-red-500.html`, Nov. 2011.

[26] D. M. Stampe and E. M. Reingold. Selection by looking: A novel computer interface and its application to psychological research. *J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), Eye Movement Research: Mechanisms, Processes, and Applications*, 19:467–478, 1995.

[27] P. Suppes. Eye-movement models for arithmetic and reading performance. *E. Kowler (Ed.), Eye Movements and their Role in Visual and Cognitive Processes*, pages 455–477, 1990.

[28] C. Ware and H. H. Mikaelian. An evaluation of an eye tracker as a device for computer input. In *Conference on Human Factors in Computing Systems and Graphics Interface*, pages 183–188, New York, 1987. ACM Press.

[29] Wikipedia. Websocket. `http://en.wikipedia.org/w/index.php?title=WebSocket&oldid=459013040`, Nov. 2011.