eyeGUI: A Novel Framework for Eye-Controlled User Interfaces

Raphael Menges
Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau
Koblenz, Germany
raphaelmenges@uni-koblenz.de

Korok Sengupta
Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau
Koblenz, Germany
koroksengupta@uni-koblenz.de

Chandan Kumar
Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau
Koblenz, Germany
kumar@uni-koblenz.de

Steffen Staab
Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau
Koblenz, Germany
staab@uni-koblenz.de

NordiCHI'16, October 23 - 27, 2016, Gothenburg, Sweden.

ACM 978-1-4503-4763-1/16/10...$15.00.
DOI: http://dx.doi.org/10.1145/2971485.2996756

Abstract
The user interfaces and input events are typically composed of mouse and keyboard interactions in generic applications. Eye-controlled applications need to revise these interactions to eye gestures, and hence design and optimization of interface elements becomes a substantial feature. In this work, we propose a novel eyeGUI framework, to support the development of such interactive eye-controlled applications with many significant aspects, like rendering, layout, dynamic modification of content, support of graphics and animation.

Author Keywords
Gaze input; eye tracking; interactive elements; visual feedback; eye-controlled interfaces.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]: User Interfaces

Introduction
To interact with computer applications having modern graphical user interfaces, users need to be able to perform point-and-select kind of operations. Interaction with interface elements such as icons and buttons is often performed with conventional input devices like mouse. The advantage of such conventional input mechanisms is

that there is almost no noise between the physical movement of the device and the on-screen object. This allows selection of small targets in a window environment to the very finest levels (to the pixel). This kind of interaction has been a popular phenomenon and the associated user interfaces has been well accepted by standard users. However, users who are not able to use their hands to interact with a computer (due to motor disabilities) need alternatives for performing such operations to gain access to the graphical user interface [4].

In that regard, real-time eye gaze signals from eye tracking systems provide a natural path to interact. Eye gaze signals has been often considered to provide natural ways to interact and assist people with disabilities to communicate with computers. Moreover, with the advancement in technology, able-bodied users would soon be accepting eye-based interaction as additional modality [6]. With the evolution of modern eye tracking devices, it is possible to record eye movements of users, which can be used to generate gaze events, and to analyze the data to deduce more high-level events [9]. Thus, there is a lot of potential in using eye gaze in human-computer interfaces, which is already evident with several applications of gaze-based input [5].

The methods for computer control by gaze interaction can be divided into two main categories: either the eye tracker is simply used to control the mouse in the normal graphical user interface [3], or the second approach of composing customized interfaces for eye gestures [6]. In the first case of mouse control, the major problem occurs due to no universal method for issuing mouse clicks, and the performance of such interaction highly depends on the eye tracking accuracy. Therefore, the second approach of eye-controlled customized interfaces, based on the direct link between the eye tracker and the application is very significant for the usability and performance.

Eye-controlled interfaces

The interfaces generally use the architecture that allows it to acclimatize to the type of input device used for operating the computer. The look and feel of the interface actually depends on the device that is selected as primary input. For example, when mouse and keyboard is the primary input device, the interface-controllable elements such as buttons, icons, menus, scroll bars, lists, and dialog boxes may appear as with a conventional interface. However, when a different physical input device like eye tracker is the primary source, the look and feel of these elements needs to be changed to be more appropriate for that device [7]. In that regard, eye gaze as input is a challenging phenomena due to the limitation of eye trackers like visual angle, calibration errors, drift, and inherent eye jitter. Hence the devices does not exactly correspond to where the user is looking, and therefore user not able to select small targets very precisely. Furthermore, Midas Touch is another major problem with eye-based interaction since it is difficult to discriminate between inspections and selections [9].

With regard to accuracy, interface adaptations techniques have often been explored, e.g., interface manipulations like enlarging targets to make them big enough is a kind of solution that addresses the accuracy issue of eye gaze interaction in the acquisition of small targets. Moreover, when the user looks at the screen, the area immediately around the gaze point can be linearly magnified and re-displayed in a zoom window [8]. The purpose of employing a magnified view is to accommodate the accuracy problem of the eye tracker so as to exactly map

Figure 1: Schau genau!: Interactive typing of nickname for high score table

gaze points to a desired target, consequently the desired actions can be easily and correctly performed. The area immediately around the gaze point can also be non-linearly distorted. Distorting the gaze area around the desired target could benefit the performance of user's gaze [2]. Another way to magnify a target is to temporarily expand the target itself rather than zooming the area around it. When the user's gaze falls on the vicinity of the desired target, the target size could increase enough to involve the gaze point into the enlarged target. With regard to Midas Touch problem, most errors are induced by lack of adequate feedback from the screen. Hence, providing feedback to users with respect to their gaze activity is a significant element of eye-controlled interfaces [10]. Especially since the slightest discrepancy between users eye movements and what they see/feel/hear can disrupt the experience they are engaged in. Hence the designers need to re-purpose the feedback mechanisms for sensory information from eyes. The usage of adequate visual graphics and animation as feedback could assist users to discriminate between inspections and activations and to reduce error in interactive operations, e.g., buttons get activated when the gaze hits them, and they shrink after activation to trigger the button. Colored overlay increasing in size would work as a visual representation of the remaining time until the trigger.

We have employed the above-mentioned heuristic of eye-controlled interfaces in an interactive game application: Schau genau! a game where eye tracking input is used to control a butterfly[1]. The player collects flowers and classifies photographs of flowers for gathering points [11]. In this immersive game environment several interaction elements were included with respect to the button size, shape, visual feedback, typing mechanism, fixation and dwell time, etc. For example figure 1 shows the game screen space for the player inserting a nickname for the high score table. The alphabet is shown on the top where the user can scroll horizontally through the letters. The fixated letter enlarges until a dwell time is over and the letter is selected. If the player fixates another letter in the meantime, the old one is scaled down again. On the bottom of the screen, the player can either confirm the input or delete the latest written later by individual selection buttons. All these interface and interaction components of Schau genau! were very well received by the participants yielding to high usage and explicit positive feedback. The game was installed in a horticultural garden show in Landau[2] in a stand-alone arcade cabinet. The game was played ∼2900 times during the exhibition period, which is a clear indication of the popularity and acceptability of adopted interaction elements.

The results validate the significance of gaze adapted interactive elements for enhanced usability of eye-based interaction. The need for such gaze enhanced user interface design has been discussed in the research community [7]. However there are no unifying framework to support such gaze-centered design and developments. In this work we propose eyeGUI to design and develop graphical user interfaces suitable for eye-based input control. eyeGUI specifically focuses on the concept of eye-controlled user interfaces to ease the development of interactive GUI elements, compared to other available open source approaches like OpenGazer[3], which addresses the generic problem of measuring eye gaze.

[1] https://github.com/raphaelmenges/schaugenau

[2] http://lgs-landau.de

[3] https://github.com/opengazer/OpenGazer

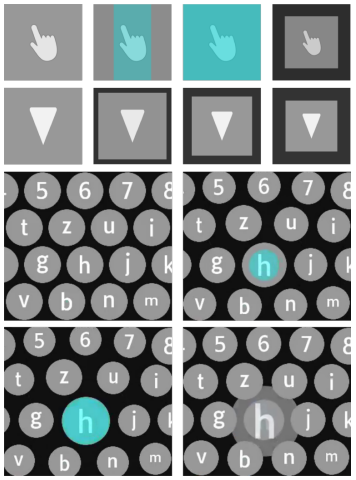

Figure 2: eyeGUI: Interactive elements like buttons (upper row), sensors (middle row) and character typing (bottom)

eyeGUI

The proposed eyeGUI framework enables the manipulation and rendering of user interfaces for eye tracking input. A variety of elements, like buttons, images and texts, to build a proper interface can be used from the framework. Most of them can be customized in their size, appearance or behavior, e.g., buttons can be given an arbitrary icon that scales automatically when the overall size of the interface changes, though the ratio of height and width of the image stays the same. eyeGUI layouts are supposed to be overlay for specific application interacting with the gaze input from the eye tracking device. Moreover stand-alone application can also be developed using eyeGUI layouts.

All elements in eyeGUI are designed especially for eye tracking in their size, appearance, and user interaction, e.g., buttons get activated when the gaze hits them, and they shrink after triggering to provide the user with interaction feedback. A colored overlay increasing in size works as a visual representation of the remaining time until the triggering. It also offers other eye tracking specific features like optionally showing the gaze path during usage. Figure 2, showcases some examples of these elements; the first row signifies a button triggering through eye-gaze interaction, where the animated highlighting over the icon shows the focus duration and the button is pressed after the duration is exceeded. The second row indicates a sensor like button, relevant for progressive elements like scrolling. The four images at bottom show different stages of eye-typing with magnifying effect of character selection.

Figure 3, outlines the architecture of eyeGUI framework. It is developed in C++ 11 and based on OpenGL. It offers to build user interfaces for eye tracking by adding XML-files as layouts and manipulating elements (e.g., buttons), within these layouts via listeners. The listeners can be accessed in the application environment to give every button an own functionality and also to interact with external APIs. Eye tracking device (e.g., SMI REDn Scientific, Tobii EyeX) would send raw gaze data to application, which implements a listener for that stream and receives data. Filtered gaze signal through application is piped to eyeGUI layouts, which handle interaction and call events in the interface (e.g. button pressed), and the application would react to those user interactions by call backs. The eyeGUI framework has been made available at a GitHub repository[4]. For development purposes, the control paradigm can be switched to mouse-control to emulate eye signals.

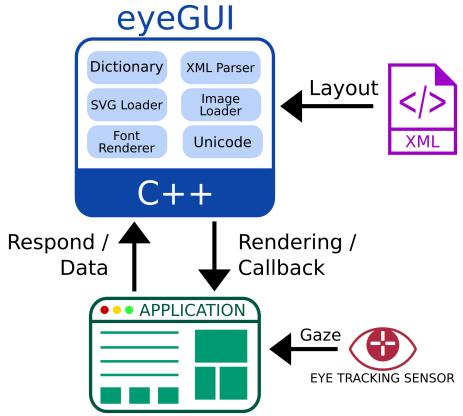

Figure 3: eyeGUI Architecture

The integration of eyeGUI is similar to other OpenGL user interface libraries[5]. The developer is free to choose how to create a window and on which way to initialize the

[4] https://github.com/raphaelmenges/eyeGUI
[5] ImGui: https://github.com/ocornut/imgui

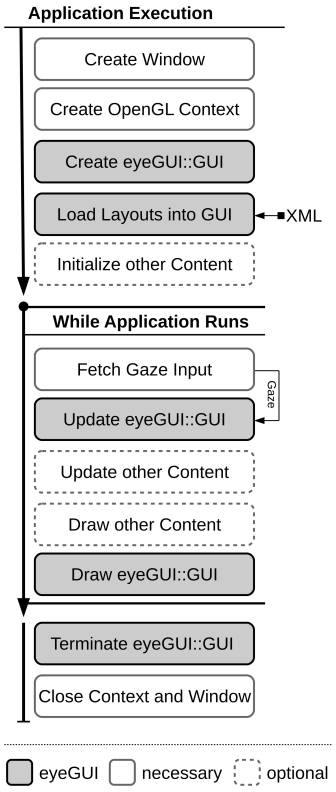

Figure 4: Sequence diagram of exemplary eyeGUI usage in custom application

OpenGL context. Before the render loop is entered, the GUI object for eyeGUI must be instantiated and an arbitrary number of layouts from XML files might be added. During the render loop, for every frame the most recent gaze input is used to update eyeGUI, which provides feedback whether the input has been used by any layout. Based on that feedback, the developer can decide how to update the application content. For interactions with the application, listeners might be employed to enable eyeGUI to call back into the application if a button is triggered or other events occurred. A detailed sequence of a minimal eyeGUI application is listed in Figure 4. All functions are accessible through a single header files with C++ functions and the memory allocation for displayed images and other is handled internally.

Figure 5: A Web browser application developed with eyeGUI framework

eyeGUI Applications

Adhering to the eyeGUI framework, we propose the development of customized eye-controlled applications for end-user needs. To demonstrate such applicability we showcase two prototype applications for Web and social media browsing using eye gaze signals. Figure 5 shows an adapted Web browsing interfaces for gaze interaction (developed using eyeGUI with Chromium Embedded Framework to realize the browser application capabilities). The custom eyeGUI layout at left and right indicates the options to select various browsing operations with gaze commands[6]. Figure 6 shows the Twitter application interface for gaze interaction (developed using eyeGUI with Twitter REST APIs), where user can perform all essential operations e.g., like, tweet, search, follow, discover etc. using eye gestures[7]. In the small scale lab studies these applications have achieved high usability and performances. In near future, we plan to conduct large scale studies for extensive evaluation of these applications.

Figure 6: A Twitter application developed with eyeGUI framework

[6]Detailed functionality of Browser prototype: https://youtu.be/zj1u6QTmk5k

[7]Detailed functionality of Twitter application: https://youtu.be/NQQfB7nf3qw

Conclusions and Future Work

In this paper we proposed a novel eyeGUI framework, to ease the developments of interactive applications based on eye tracking system. This framework today can be used to render interactive elements for several applications as demonstrated via Browser and Twitter applications. One of the most eminent future work that lies in the curve of eyeGUI is the inclusion of sensor module to its framework where in the application that hosts the eye tracking sensor code will send signal to the eyeGUI module, and all the filtering and processing of the signal will take place in it. This will add a big step in the advantage of using this framework as developers will no longer need to add on their integration of sensors to their main application and they can take help of the eyeGUI framework. Another work that lies in the future of eyeGUI is to adapt it for multimodal interaction. Multiple sensors can be used in conjunction with eye tracking system.

Acknowledgment

This work is part of project MAMEM [1] that has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement number: 644780.

References

[1] Multimedia authoring and management using your eyes and mind. http://www.mamem.eu/.

[2] Ashmore, M., Duchowski, A. T., and Shoemaker, G. Efficient eye pointing with a fisheye lens. In *Proceedings of Graphics interface 2005*, Canadian Human-Computer Communications Society (2005), 203–210.

[3] Bates, R., and Istance, H. O. Why are eye mice unpopular? a detailed comparison of head and eye controlled assistive technology pointing devices. *Universal Access in the Information Society 2*, 3 (2003), 280–290.

[4] Brodwin, M. G., Star, T., and Cardoso, E. Computer assistive technology for people who have disabilities: Computer adaptations and modifications. *Journal of rehabilitation 70*, 3 (2004), 28.

[5] Chandra, S., Sharma, G., Malhotra, S., Jha, D., and Mittal, A. P. Eye tracking based human computer interaction: Applications and their uses. In *2015 International Conference on Man and Machine Interfacing (MAMI)*, IEEE (2015), 1–5.

[6] Jacob, R., and Stellmach, S. What you look at is what you get: Gaze-based user interfaces. *interactions 23*, 5 (Aug. 2016), 62–65.

[7] Kumar, M., and Winograd, T. Guide: gaze-enhanced ui design. In *CHI'07 Extended Abstracts on Human Factors in Computing Systems*, ACM (2007), 1977–1982.

[8] Lankford, C. Effective eye-gaze input into windows. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, ACM (2000), 23–27.

[9] Majaranta, P., and Bulling, A. Eye tracking and eye-based human–computer interaction. In *Advances in physiological computing*. Springer, 2014, 39–65.

[10] Majaranta, P., MacKenzie, I. S., Aula, A., and Räihä, K.-J. Auditory and visual feedback during eye typing. In *CHI'03 Extended Abstracts on Human Factors in Computing Systems*, ACM (2003), 766–767.

[11] Schaefer, C., Menges, R., Schmidt, K., Kuich, M., and Walber, T. Schau genau! an eye tracking game with a purpose. In *Applications for Gaze in Games* (2014).