Abstract
Navigating source code is at the core of software development, consuming a significant amount of time and effort. Navigation is typically done using the mouse, which in many cases may not be efficient and usable, as it causes context switching and interruptions.

EyeNav brings eye tracking to code editors. It allows for a more natural source code navigation, controlled by the developer's gaze and keyboard shortcuts. It aims to improve usability and efficiency over the mouse on code navigation tasks. EyeNav provides a smoother user experience by letting developers keep their hands on the keyboard at all times. It is a production-ready Brackets.io plugin that allows anyone with an eye tracker to start using it immediately.

Demo video: https://youtu.be/AkDyx2l-YGk
Source code: https://github.com/sradevski/eyenav

Author Keywords
Eye tracking; Gaze tracking; Source code navigation; Code editors; IDE

ACM Classification Keywords
H.5.2 [Information interfaces and presentation (e.g., HCI)]: Input devices and strategies
Introduction

Navigating source code is at the core of software development. In fact, developers spend around 35% of the time navigating within and between source code files, and another 20% on reading code during maintenance tasks. Reading code is identified by text caret movement, mouse cursor hovering, text selection, and scroll bars hovering [7]. This signifies the importance of the efficiency and usability of code navigation.

Many existing approaches to improve navigation focus on coarse-grained navigation [2,11], such as moving across files and locating artifact definitions. Although this is one approach to navigation, fine-grained navigation, also referred to as reading code in [7], should not be neglected, as shown earlier. Since our research focuses on fine-grained navigation activities, this paper will refer to fine-grained navigation as navigation.

While coding, the typical resting place for the palms is on the keyboard. Therefore, the movement required in order to use the mouse, as well as the mouse being an abstract interface, can hurt usability and efficiency. Consequently, there have been several attempts to bring natural interfaces to software development, such as touchscreens [1] and hand gestures [12].

As an attempt to improve usability and efficiency of code navigation, we propose EyeNav. EyeNav uses eye (gaze) tracking as coordinate input in combination with keyboard shortcuts to execute commands. Since navigation is done while keeping the hands on the keyboard, EyeNav requires no additional movements. EyeNav is an open-source project and it is delivered as a Brackets.io$^1$ editor plugin.

EyeNav

In this section, the functionalities and architecture of EyeNav will be described. The eye trackers used for this research were EyeTribe Tracker (first version)$^2$ and Tobii EyeX$^3$. EyeNav has full support for both devices.

While building EyeNav we aimed at making it as flexible and extensible to different eye trackers as possible. In order to achieve flexible and loosely coupled architecture, we based our architecture on WebSockets, separating device-specific logic to a thin server wrapping over the provided SDK, while keeping all the logic as an editor plugin, as described in Figure 1. This means EyeNav can be extended so it works with essentially any eye tracker on the market with minimal effort.

---

$^1$http://brackets.io/ (accessed on 22 June, 2016)

$^2$https://theeyetribe.com (accessed on 22 June, 2016)

$^3$http://www.tobii.com (accessed on 22 June, 2016)
As of now, EyeNav supports the following functionalities:

- **Clicking**: By pressing the designated shortcut, the caret moves where the gaze is centered, synonymous to a mouse click at a certain location.

- **Code Scrolling**: Code scrolling is moving the caret along only one axis - horizontally along a line, or vertically along a column. This functionality eliminates the error along one of the axes, resulting in a more precise caret movement.

- **Page Scrolling**: While the shortcut is held down the page is scrolled in the direction of the gaze. The speed varies with the change in distance between the center of the editor and the gaze location. Scrolling is a good candidate to be extended by automatic scrolling and haptic feedback [6].

- **Single Character Movement**: This functionality represents a 1 to 1 mapping between the arrow keys and WASD keys for easier access while coding. The user can also do one character manual calibration while making a single character movement by holding the designated shortcut.

- **Code Selection**: Selecting code is possible with any of the commands (except for page scrolling) just by holding an additional key before executing them.

Each functionality is triggered by a dedicated keyboard shortcut. There are three types of keys based on the function they do. The first one is the trigger key, the main key that toggles all functionalities of EyeNav when pressed and held down. The second type is command keys, keys that execute a functionality bound to a certain key. The third type is modifier keys, keys that apply some additional functionality to the command keys, and have no functionality on their own.

---

**Related Work**

Eye tracking has been used in software engineering in ways that bring a better understanding of software engineering in various areas, but almost not at all in a manner that utilizes eye tracking as an input device [10], with the exceptions of EyeDE.

EyeDE, which is closely related to EyeNav, is another research that tries to apply eye tracking to code editing in order to aid with navigation [4]. EyeDE focuses on hands-free navigation for reading and understanding code. EyeDE has a completely distinct set of functionalities, usage, and deployment model, making it difficult to compare it to EyeNav. That being said, some of the capabilities of EyeDE are well designed and could be merged into future versions of EyeNav.

iTrace is an eclipse plugin, implemented with the purpose of collecting gaze data for further analysis [9]. Although the aim of EyeNav and iTrace are different, there are some similarities in terms of deployment and capabilities.

It is important to mention that research concerned with the application of eye tracking for typing and computer operation for people with disabilities is very important as a lot of ideas and knowledge can be drawn from it and applied to EyeNav.

---

**Challenges and Future Work**

Before we expect a wider adoption by software developers, it is important to show the benefits of EyeNav as a novel tool with no previous usage history. This is why our next goal is to do an empirical evaluation of the tool with software developers.

Furthermore, the inherent accuracy problems may be a challenge in convincing developers to adopt EyeNav. In
order to mitigate them to some extent, one of our goals is to do data smoothing [8] and to implement movement prediction and approximation, where applicable.

Another goal is to separate the dependency of EyeNav from Brackets.io, and have an implementation for both Atom.io and Visual Studio Code. We hope this will increase the availability and utilization of EyeNav.

**Conclusion**

In this paper we introduced EyeNav, a novel approach to navigating source code by using an eye tracker. Its flexible architecture makes it very easy to extend it to work with any eye tracker. It is a production-ready Brackets.io editor plugin that allows anyone with an eye tracker to utilize it as part of their software development workflow.

**Acknowledgment**

This work has been supported by JSPS KAKENHI Grant Number 16H05857 and JSPS Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers: Interdisciplinary Global Networks for Accelerating Theory and Practice in Software Ecosystem (G2603).

**References**


