REyeker: Remote Eye Tracker

Jonas Mucke
jonas.mucke@s2018.tu-chemnitz.de
Chemnitz University of Technology
Germany

Marc Schwarzkopf
marc.schwarzkopf@mb.tu-chemnitz.de
Chemnitz University of Technology
Germany

Janet Siegmund
siegj@hrz.tu-chemnitz.de
Chemnitz University of Technology
Germany

ABSTRACT

Eye tracking allows us to shed light on how developers read and understand source code and how that is linked to cognitive processes. However, studies with eye trackers are usually tied to a laboratory, requiring to observe participants one at a time, which is especially challenging in the current pandemic. To allow for safe and parallel observation, we present our tool REyeker, which allows researchers to observe developers remotely while they understand source code from their own computer without having to directly interact with the experimenter. The original image is blurred to distort text regions and disable legibility, requiring participants to click on areas of interest to deblur them to make them readable. While REyeker naturally can only track eye movements to a limited degree, it allows researchers to get a basic understanding of developers' reading behavior.

CCS CONCEPTS

• **Human-centered computing** → *User studies.*

KEYWORDS

Eye tracking, Visual attention, Comprehension, Program learning, Mouse-contingent interface

ACM Reference Format:
Jonas Mucke, Marc Schwarzkopf, and Janet Siegmund. 2021. REyeker: Remote Eye Tracker. In *ETRA '21: 2021 Symposium on Eye Tracking Research and Applications (ETRA '21 Short Papers), May 25–27, 2021, Virtual Event, Germany*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3448018.3457423

1 INTRODUCTION

Eye tracking has successfully made its way into programming research, for example, to understand how code style can affect programmers' comprehension [Bauer et al. 2019; Sharif and Maletic 2010], how expertise modulates reading behavior [Peitek et al. 2020], or how eye movements can shed light on underlying cognitive processes [Fakhoury et al. 2018].

However, eye tracking experiments are typically time-consuming and require to measure one participant at a time, with the experimenter and the participant being in the same room to set everything up. Especially in the current pandemic, it is difficult to find volunteers and conduct safe measurements.

To nevertheless conduct safe studies, a remote approach is necessary, for example, by using a use webcam-based eye tracking or a restricted focus viewer paradigm [Jansen et al. 2003].

First, with webcam-based eye tracking, we would use the webcam of participants' devices (e.g., personal computer or smartphone), but it comes with several challenges. From the technical side, gathering reliable data is difficult, because several factors influence data quality, such as the resolution of the webcam, the lighting conditions, the distance to the webcam, or the angle to the webcam. Furthermore, protecting the private data of participants is a delicate matter and can potentially lead to conflicts.

Second, a restricted focus viewer shows a participant a blurred image that is deblurred only at the location of visual attention. Thus, a participant can only perceive one certain area of interest at a time. Naturally, this has less similarity to actual eye-tracking data than a webcam-based approach, but is less intrusive for participants. Furthermore, it is quicker to implement, which is an important factor in the current pandemic situation. Thus, we decided to implement a restricted focus viewer paradigm.

Following this approach, we decided to develop REyeker, a remote eye tracking tool for conducting safe research with a larger group of participants. It is a cursor-based technique, which relates visual focus and cursor position by using a viewing-window, blurring all areas around this viewing window. The restricted focus viewer was the first system to use this method [Jansen et al. 2003]. It blurs visual stimuli, and only a small area is clear and can be perceived by participants. Participants can move this area in the image by moving the mouse to deblur other image sections. An extension of the restricted focus viewer is the BubbleView [Kim et al. 2017]. Instead of deblurring visual information by moving the mouse, the BubbleView deblurs only a section on which was clicked. This process more accurately represents the way visual information is received, as the background tends to be blocked out by peripheral vision and only points of visual focus are viewed more closely, which in this case is achieved by deblurring. This approach can reasonably approximate fixations, compared to eye tracking system [Kim et al. 2017].

However, these alternatives have their limits when tracking the eye movements of programmers when reading the source code, because the BubbleView only supports a circular deblurred area. For reading source code, programmers typically read entire lines, which can be better approximated with an oval deblurred area.

ETRA '21 Short Papers, May 25–27, 2021, Virtual Event, Germany

ACM ISBN 978-1-4503-8345-5/21/05…$15.00
https://doi.org/10.1145/3448018.3457423

To let us remotely observe the reading behavior of programmers, we adapted the underlying idea of the BubbleView in our tool REyeker, which is available at the project's Web site [1]. Specifically, we offer different shapes for the deblurred area and a customizable blur, which both can be adjusted depending a concrete study.

In this paper, we give an overview of how REyeker works, including important details from the implementation and how experimenters can adjust settings according to their needs.

2 REYEKER - AN OVERVIEW

In Figure 1, we show an example of how participants view source code. The deblurred area in Figure 1(a) almost entails one entire line of source code, which approximates where the participant is looking at. By clicking on another area, the current area will be blurred, and the clicked area will be deblurred.

The source code snippet is shown as an image, so REyeker currently does not support experiments in which participants need to change source code. The main reason is that, as an image, we had the best experience with fast load time and tracking accuracy, so that we can have the best approximation to an on-site eye tracker.

To adapt the deblurred area to a concrete study, experimenters can make several adjustments to REyeker. Specifically, the following parameters for the deblurred area can be adjusted:

- Shape of the deblurred area: rectangle, circle, or ellipse.
- Size of the shape: The shape can have the size 0 (i.e., everything is blurred) or up to the size of the image (i.e., nothing is blurred).
- Transition: the transition between the blurred and deblurred area can be adjusted to be rather abrupt or smooth.
- Filter: the kernel can also be adjusted as needed in either the x or y dimension, adding a smear to the blurred areas.

In Figure 1, we illustrate different uses of the tool with different parameters. In Figure 1(a), a weak but even blur is used. The deblurred area is a rectangle, showing almost the entire line. This is particularly suitable for understanding program comprehension, as the code structure and the instructions are the key focus. By contrast, the settings in Figure 1(b) allow participants to read only a rough code structure due to the strong blur. Furthermore, the shape is a small circle with a transition area, which collects data that is more similar to that of an on-site eye trackers. An intermediate solution can be achieved by parameters as defined in Figure 1(c), where code instructions can still be displayed within a small ellipse, which, however, does not reveal as much of the code as the rectangle in Figure 1(a). Another adjustment is the strong blur in the x-axis. This can be particularly interesting when the structure of a block or the syntax highlighting of a line should be recognizable.

(a)

(b)

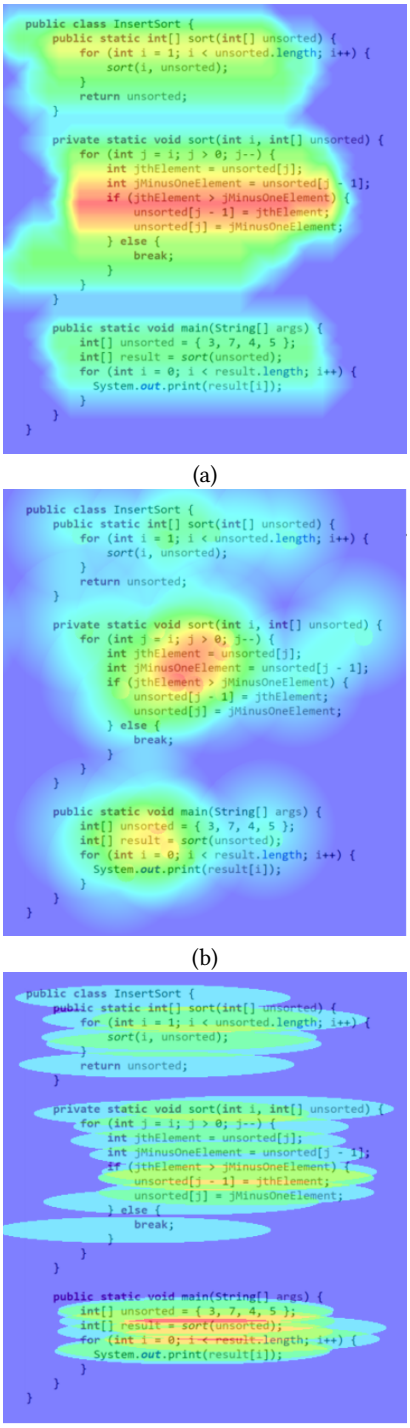
(c)

Figure 1: Illustration of different settings for REyeker. (a) A rectangle, with a medium-soft transition. (b) A circle with smooth transitions. (c) An ellipse, with abrupt transition.

2.1 Data

The data of the reading behavior of a participant are stored as x/y coordinates with the according time stamps in ms (e.g., 187-27, 1604). The origin of the coordinates (i.e., 0-0) are defined to be the upper left corner. The maximum x and y values depend on the resolution of the image that participants see (e.g., the source code). Thus, for an image of resolution 900x1000, the maximum values are 900x1000, depending on the resolution of the monitor, and scrolling might be required. The time starts with the beginning of the experiment,

[1] https://github.com/brains-on-code/REyeker



(a)

(b)

(c)

Figure 2: Heatmaps based on the click data and on the selected parameter values. Each heat map is based on the settings illustrated in Figure 1.

so the data describes that, after 1604 milli seconds, the participant clicked at the position 187-27.

Based on these data, REyeker can produce several visualizations. First, REyeker can create heat maps (Figure 2). In contrast to on-site eye trackers, the heat maps have a coarser approximation at which a participant looked at, which depends on the adjustable parameters. For example, the heat map in Figure 2(a) corresponds to the setting in Figure 1(a), so the shape of the areas of interest are rectangular (size: x: 134, y: 6) and cover almost an entire line, with medium-soft transitions (transition range: 34) between the areas of interest. This map illustrates that a participant often looked at one specific line, but it cannot show what within that line was of interest. For smaller areas of interest, different shapes need to be selected and adjusted. The elliptical shape in Figure 2(c) (size: x: 146, y: 13, transition:1) gives us more detail what part of a line was interesting, and the circle in Figure 2(b) (size: radius: 10, transition:77) resembles most closely the heat map of on-site eye trackers.

A suitable setting of the parameters is important, because the parameter settings also affect the reading behavior of participants. Specifically, a small circle requires more clicks than a large ellipse, because only a smaller part surrounding the center of the click gets deblurred and is readable for participants. However, smaller readable areas for participants also make it more difficult for participants to read source code, because they may need to keep more source code in their working memory (or click more often). Thus, experimenters need to select a suitable trade off between detailed areas of interest and resemblance to natural reading behavior of participants. To thoroughly understand how the settings affect the behavior of participants, comparative studies with on-site eye trackers are necessary, which we will conduct once the corona regulations at our institution allow it.

In addition to heat maps, REyeker can also visualize the chronological sequence of the click data, shown in Figure 3. These visualizations do not depend on any of the adjustable parameters, but only on the behavior of participants. In conjunction with the heat maps, it can give a useful approximation of participants' reading behavior.

For example, in Figure 3, a participant searched all methods from top to bottom, jumping quickly between the methods. As soon as they found the "main" method, their behavior changed. Clicking was done more frequently in this method. Finally, the participant left this method and briefly looked at all other methods. Thus, we get a basic understanding of participants' strategy to understand source code.

Further options for data analysis are also possible with REyeker. For example, we are currently exploring how to define areas of interest and transitions between these areas.

2.2 Integration into Survey Tools

To be flexible, REyeker can be integrated into different survey tools or be run on stand-alone web servers. We have integrated it into SoSciSurvey[2] to run it with our surveys. This happened by adjusting small parts of the code of REyeker and by setting up some boilerplate code in SoSciSurvey.

For adjusting the code in REyeker, we made three changes:

[2] https://www.soscisurvey.de/

Figure 3: Chronological sequence of click data. The black, stepped vertical line illustrates the click sequence, starting from top left. With each click, the vertical line moves one step to the right.

- We defined a use case by creating a new variable in the class *UseCases*. This acts as a flag to indicate which code fragments need to be executed, which depends on the concrete integration.
- We defined an adapter to connect REyeker to SoSciSurvey. The adapter retrieves HTML elements that are defined in SoSciSurvey, which refers to the HTML elements as internal variables, relative paths of media files, or parameters.
- Last, the click data that are stored during runtime in REyeker need to be serialized to be stored permanently. To this end, we need another adapter as connection that stores the data of the previously retrieved HTML elements. On the project's Web site, we provide an example of all the adjustments and its according effects.

On the SoSciSurvey page, we made the following changes:

- We uploaded the compiled code of REyeker and media files (e.g., images of the source code).
- When creating the questionnaire, HTML variables must be created to store the click data of participants.
- These HTML variables then are accessed to store the click data permanently.

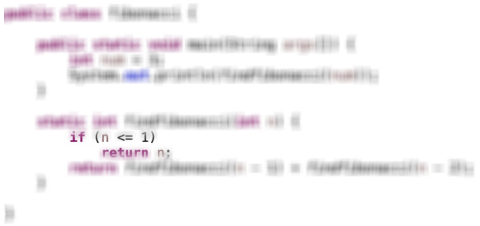
In order to integrate REyeker into other tools, similar steps must be carried out. Certain concepts, such as the SoSciSurvey variables, differ from tool to tool, but it is also possible not to use these variables at all and instead send the data to a separate server with a database API. On the project's Web site, there are more details on integrating REyeker or running it on a stand-alone server.

3 PROGRAMMING LANGUAGE

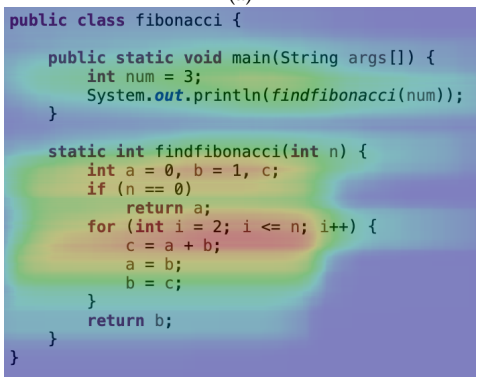
Originally, we planned to use JavaScript, so that REyeker can be executed natively in browsers. However, since different web browsers do not always implement the latest JavaScript features specified by ECMAScript standard, writing compatible source code is difficult. In a first version of REyeker, we implemented a work-around by using WebAssembly compiled from Rust, which also runs natively in most browsers. This had the advantage of writing type-safe code, but the disadvantage that a JavaScript implementation was still needed, at least for loading the code.

Eventually, the choice fell on TypeScript. TypeScript can be compiled into different versions of JavaScript, which makes it easy to ensure versatile support. Furthermore, it is a statically typed language, so supports writing type-safe code. Additionally, the code can easily be split into different files and later compiled into one. This supports modularization in development, so that writing maintainable code is more straightforward than with JavaScript.

4 EVALUATION



(a)



(b)

Figure 4: Figure (a) visualizes the user view of the ongoing evaluation study, using the following parameters: The selected shape is a rectangle with a width of 200 pixels and a height of 1 pixel. The transition range amounts to 30 pixels. The filter was set to 8 pixels in the x and y axis. Figure (b) shows a heatmap of the same code snippet. It illustrates the weighted average of all participants' data.

Currently, we are conducting a study with REyeker to evaluate the data that it collects and whether participants feel comfortable enough with it during the studies. So far, the data themselves look promising, as illustrated in Figure 4. We used the following parameters for this study:

- Shape of the deblurred area: rectangle
- Size of shape: [width: 200px, height: 1px]
- Transition: 30px
- Filter: [x: 8px, y: 8px]

The heatmap shows that participants concentrate on parts of the source code that appear relevant for the computation of Fibonacci

numbers, that is, the conditions and computations of the for loop, the starting condition, and the call with the actual value. Furthermore, none of the participants mentioned to be too much disturbed by the blurred areas and the necessity to click on certain spots for deblurring. Thus, we are confident that REyeker is a safe and reliable way to conduct remote eye tracking studies.

5 CONCLUSION AND FUTURE WORK

To allow us to continue our research in a safe and efficient way, we have developed REyeker, a tool that simulates an eye tracking environment by restricting the readable area for participants. It can be adjusted to provide a compromise between fine-grained data and undisturbed reading behavior of participants.

Currently, we can only conduct remote studies with REyeker to evaluate the robustness of the data. Once possible, we will compare the data that REyeker collects with that of an on-site eye tracker. Although the motivation to implement REyeker was strongly driven by the pandemic, we hope that in the future, REyeker will become a useful tool to conduct large-scale, remote eye tracking studies.

ACKNOWLEDGMENTS

This work is supported by DFG grant SI 2045/2$\bar{7}$2.

REFERENCES

J. Bauer, J. Siegmund, N. Peitek, J. C. Hofmeister, and S. Apel. 2019. Indentation: Simply a Matter of Style or Support for Program Comprehension?. In *Proc. Int'l Conf. Program Comprehension (ICPC)*. IEEE CS, 154–164.

Sarah Fakhoury, Yuzhan Ma, Venera Arnaoudova, and Olusola Adesope. 2018. The Effect of Poor Source Code Lexicon and Readability on Developers' Cognitive Load. In *Proc. Int'l Conf. Program Comprehension (ICPC)*.

Anthony R. Jansen, Alan F. Blackwell, and Kim Marriott. 2003. A Tool for Tracking Visual Attention: The Restricted Focus Viewer. *Behavior Research Methods, Instruments, & Computers* (2003), 57–69.

Nam Wook Kim, Zoya Bylinskii, Michelle A Borkin, Krzysztof Z Gajos, Aude Oliva, Fredo Durand, and Hanspeter Pfister. 2017. BubbleView: An Interface for Crowdsourcing Image Importance Maps and Tracking Visual Attention. *ACM Transactions on Computer-Human Interaction (TOCHI)* (2017), 36.

Norman Peitek, Janet Siegmund, and Sven Apel. 2020. What Drives the Reading Order of Programmers? An Eye Tracking Study. In *Proc. Int'l Conf. Program Comprehension (ICPC)*. ACM, 342–353.

Bonita Sharif and Johnathon Maletic. 2010. An Eye Tracking Study on camelCase and under_score Identifier Styles. In *Proc. Int'l Conf. Program Comprehension (ICPC)*. IEEE CS, 196–205.