Tactile Code Skimmer: A Tool to Help Blind Programmers Feel the Structure of Code

Olutayo Falase
Stanford University
Stanford, California
tfalase@stanford.edu

Alexa F. Siu
Stanford University
Stanford, California
afsiu@stanford.edu

Sean Follmer
Stanford University
Stanford, California
sfollmer@stanford.edu

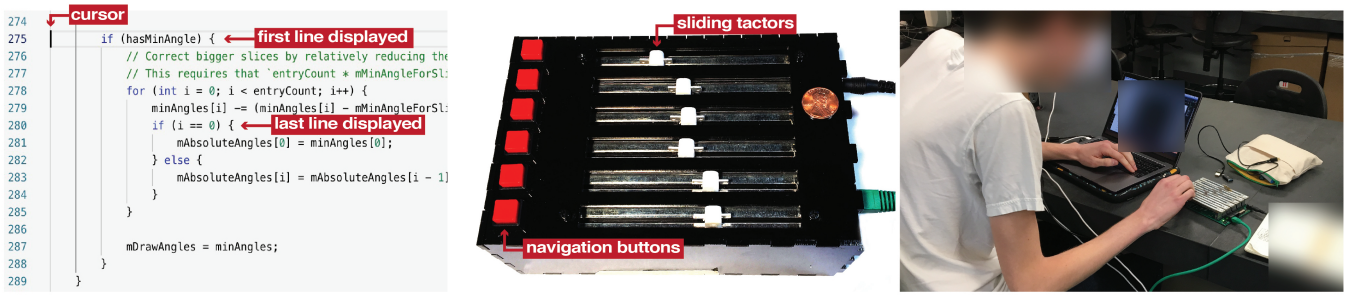

Figure 1. Indented code (left), its representation on the Tactile Code Skimmer (middle), and a participant interacting with TCS (right)

ABSTRACT

Skimming new code with a screen reader can be a time-consuming task for blind and visually impaired programmers. Screen readers aid with code navigation, but dictate code line-by-line and read spaces and tabs individually. This often provides more information than is needed. In this work, we present the Tactile Code Skimmer (TCS), a tool to aid blind and visually impaired programmers with skimming code. The device physically reflects the indentation levels of code with actuated slide potentiometers, thus helping reduce the "hearing load" that often accompanies screen readers. We describe the TCS design and implementation. Based on feedback from participants obtained through demos and an in-depth session, we discuss some considerations for tactile tools that aid with code skimming.

CCS Concepts

•**Human-centered computing** → **Accessibility technologies; Accessibility systems and tools;**

Author Keywords

Accessibility; Blind Programmers; Code Structure; Tactile Aids; Visually Impaired

INTRODUCTION

Familiarizing oneself with code that someone else has written can be a daunting task. While sighted programmers can skim new code by observing the indentation to get an overview, blind and visually impaired (BVI) programmers often use screen readers, which dictate code line-by-line. The linearity of screen readers makes it difficult to navigate blocks of code without reading the whole document [6]. Furthermore, screen readers dictate indentation as individual spaces or tabs as opposed to single units. Some BVI programmers resort to skipping code blocks to avoid such exhaustive dictation, and others have written custom scripts as a workaround [1].

It has been documented that nested sections within code are often a point of confusion for BVI programmers, and code structure in general is difficult to grasp [5]. Albusays et. all found that code nested beyond three levels is difficult to understand. A majority of the BVI programmers they interviewed agreed that "nesting and scope level indicators" would be helpful [1].

Baker et. all designed StructJumper, an Eclipse extension that digitally represents code as a hierarchical tree structure [3]. Participants were able to complete tasks to find particular sections of code faster with StructJumper. We propose the Tactile Code Skimmer (TCS), a variation of this tool that displays levels of indentation in code with slide potentiometers (Figure 1a-b). The programmer can feel with their hand how the code is indented, which reduces the need for memorization that often accompanies code represented by a tree structure or auditory cues. In conjunction with a screen reader, this device could have potential to expedite the code skimming process for BVI programmers.

Other prior work has explored auditory cues for code skimming and navigation [2, 4]. While researching such tools, we noticed an opportunity for a tactile aid. Braille displays can accomplish something similar to our concept; they mitigate a "hearing load" that accompanies screen readers and provide

ASSETS '19, October 28–30, 2019, Pittsburgh, PA, USA.

ACM ISBN 978-1-4503-6676-2/19/10.
http://dx.doi.org/10.1145/3308561.3354616

"assistance in determining the level of whitespaces, through touch" [1]. However, Braille displays are often too expensive for the average programmer [1]. Our tool could offer a more affordable solution.

TECHNICAL IMPLEMENTATION & INTERACTION

The TCS consists of six actuated slide potentiometers that respond to serial input. A Visual Studio Code extension sends information about the code indentation of the selected line and the following 5 lines via HTTP to a Python webserver, which translates those indentation levels into a slider position and sends those positions over serial connection to the potentiometers.

TCS has two modes: regular mode and folded mode. Regular mode displays each line in succession, and folded mode collapses lines with the same indentation level into one (the folding is done on the device and not within the editor itself). Folded mode is intended to give a bird's eye view of the code and aid with navigation. Navigation buttons on the device move the cursor to the different lines of code displayed.

USER FEEDBACK AND DESIGN CONSIDERATIONS

Feedback was gathered through an in-depth session with one participant and informally through demos at blind programming meetups. Figure 1c shows a user interacting with the TCS to skim a code repository.

In the in-depth session, the participant, who has peripheral vision only and 5 years of programming experience, uses assistive technologies such as VoiceOver (the screen reader for MAC OS), built-in-zoom, and high-contrast features. The participant was given time to familiarize himself with TCS and was asked to navigate code from the MPAndroidChart[1] and zxing[2] repositories on Github. Based on the user feedback, we formulate the following design considerations:

Context Provision

In both regular and folded mode, the participant found that there wasn't enough information about the lines displayed on TCS without navigating to them. The participant wanted more context about the control structure of the lines, i.e. whether they were part of loops, conditionals, or other types of blocks. A tactile indentation display should provide such context when needed, perhaps through auditory cues. In folded mode, information about the number of lines that are folded is also valuable.

Currently, the line with the cursor in the editor is the top line of TCS. Instead, a reasonable default should be chosen with the option to change it. Displaying the cursor line in a position apart from the first can provide context about the previous lines.

Indentation and Line Representation

The tactors of the code skimmer rest at the far side of the device when there are large levels of indentation. Scaling (manipulating the indentation width) and/or shifting (displaying a deeper level of indentation as zero indentation and its nested levels relative to the new base) could be a solution for indentation beyond the reach of the device. We also intend to incorporate the just-noticeable difference of tactile devices, the error of the hardware, and the average maximum levels of indentation in code to determine how many indentation levels should be displayed at a time.

We also need to consider how to abstract the code. TCS is currently sensitive to spaces that don't correspond to indentation, such as a space before a comment symbol. It might make sense to ignore such whitespace. Furthermore, TCS treats a line without whitespace as a line without indentation, whereas it might make more sense to give such lines the same indentation as the surrounding block.

Lastly, we were given feedback that six lines might provide too much information—the study participant was mostly concerned with the indentation differences between two lines at a time. In our implementation, users had to move their whole hand to feel each line. A tactile code indentation display might be a more effective device if users can feel all the lines at the same time or adjust the amount of information displayed.

Interaction

The study participant found using the same hand for the keyboard and TCS awkward, emphasizing the need for a tactile code skimming device to fit into a user's workflow in a non-disruptive way.

Similarly, several commenters had established their own ways of navigating code and would have difficulty incorporating TCS into their workflow. It would be interesting to explore if a tool like TCS would be more beneficial for new programmers.

Additional feedback on interaction revealed another application for the TCS as an indentation editor, as BVI programmers often create sections of code that have locally appropriate indentation but are globally off.

Notably, a majority of the BVI programmers that offered feedback agreed that TCS should work with any text editor of the operating system. Thus, a tactile code skimming device should be editor agnostic.

CONCLUSION

We have presented TCS, a tactile tool to help blind and visually impaired (BVI) programmers navigate new code and understand its structure. We propose that such a tool could help offload the memorization that often accompanies understanding new code as a BVI programmer, as well as the "hearing load" from screen readers. Feedback from an informal user study and suggestions for improvements on such a tactile device offer considerations for future tactile code skimming tools.

ACKNOWLEDGMENTS

We thank all participants for their time and feedback.

REFERENCES

[1] Khaled Albusays, Stephanie Ludi, and Matt Huenerfauth. 2017. Interviews and Observation of Blind Software Developers at Work to Understand Code

[1] https://github.com/PhilJay/MPAndroidChart

[2] https://github.com/zxing/zxing

Navigation Challenges. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '17)*. ACM, New York, NY, USA, 91–100. DOI: http://dx.doi.org/10.1145/3132525.3132550

[2] A. Armaly, P. Rodeghero, and C. McMillan. 2018. AudioHighlight: Code Skimming for Blind Programmers. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Piscataway, NJ, USA, 206–216. DOI: http://dx.doi.org/10.1109/ICSME.2018.00030

[3] Catherine M. Baker, Lauren R. Milne, and Richard E. Ladner. 2015. StructJumper: A Tool to Help Blind Programmers Navigate and Understand the Structure of Code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3043–3052. DOI:http://dx.doi.org/10.1145/2702123.2702589

[4] Joe Hutchinson and Oussama Metatla. 2018. An Initial Investigation into Non-visual Code Structure Overview Through Speech, Non-speech and Spearcons. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems (CHI EA '18)*. ACM, New York, NY, USA, Article LBW562, 6 pages. DOI:http://dx.doi.org/10.1145/3170427.3188696

[5] Stephanie Ludi, Lindsey Ellis, and Scott Jordan. 2014. An Accessible Robotics Programming Environment for Visually Impaired Users. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '14)*. ACM, New York, NY, USA, 237–238. DOI: http://dx.doi.org/10.1145/2661334.2661385

[6] S. Mealin and E. Murphy-Hill. 2012. An exploratory study of blind software developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Piscataway, NJ, USA, 71–74. DOI: http://dx.doi.org/10.1109/VLHCC.2012.6344485