ISBN: 978-981-94-1156-6
2024 the 14th International Workshop on Computer Science and Engineering (WCSE 2024)
doi: 10.18178/wcse.2024.06.010

Computer Programming Education System for Visually Impaired Individuals

Takamatsu, Naoki [1] [+], Eiji Kamioka [1], Chanh Minh Tran [2], Phan Xuan Tan [1]

[1] Graduate School of Engineering and Science, Shibaura Institute of Technology, Japan
[2] College of Engineering, Shibaura Institute of Technology, Japan

Abstract. Block-based programming environments, widely used for teaching beginners, pose accessibility challenges for individuals with visual impairments due to limited support for screen readers and keyboard navigation. To address this issue, a previous study has proposed a novel user interface (UI) for navigating between code blocks using keyboards. However, this approach requires users to memorize block structures in advance, leading to cognitive strain. Additionally, visually impaired individuals often struggle to comprehend entire code sections at once due to the sequential nature of screen reader output, making debugging difficult. In response to these challenges, this paper proposes a novel UI for block-based programming that includes two key components to alleviate cognitive load and improve comprehension for users with visual impairments. Firstly, instead of navigating between code blocks, the proposed UI facilitates navigation between lines, reducing cognitive strain. Secondly, a debugging support method utilizing conversational AI is proposed to generate comprehensive code summaries in natural language, aiding in comprehension. The effectiveness of the proposed UI was evaluated through subjective assessment, demonstrating improvements in both task completion time and UI usability compared to existing methods.

Keywords: Block-based Programming, Accessibility, Visual Impairments, User Interface, Screen Reader, Keyboard Navigation, Debugging Support, Conversational AI

1. Introduction

The rise of information education has sparked growing interest in block-based languages in educational environments [1]. Block-based languages represent program syntax visually, using elements called blocks, with each block's color and shape indicating its syntax [2]. Notable examples include MIT's Scratch [3] and Google's Blockly [4]. These languages enforce syntactic correctness by allowing only valid block combinations, making them ideal for beginners in programming education. However, despite their widespread use, mainstream block-based languages like Scratch and Blockly lack full accessibility for individuals with visual impairments due to limited support for screen readers and keyboard navigation [5-6].

Furthermore, Potluri et al. [7] introduced the concept of "glanceability" to address accessibility challenges in programming for individuals with visual impairments. Glanceability highlights the challenge of comprehending an entire program when individuals with visual impairments must process information one step at a time through screen reader’s voice. Unlike sighted individuals, who can visually scan multiple lines of code simultaneously to grasp the entire program, individuals with visual impairments must rely on a screen reader to read each line separately. This sequential processing becomes increasingly challenging as programs become more complex, hindering full comprehension for those with visual impairments.

Given the widespread use of block-based languages in global education, these accessibility challenges can significantly restrict educational access for people with visual impairments. Especially within the framework of inclusive education advocated by the United Nations [8], it is crucial to integrate children with and without disabilities into the same learning environment without discrimination. Achieving such an educational environment requires the development of new UIs for block-based programming that serve both sighted individuals and those with visual impairments. This study proposes a novel UI to address these challenges and support individuals with visual impairments in programming education.

[+] Corresponding author. Tel.: +81 80 4800 3219.
E-mail address: af20019@shibaura-it.ac.jp.

2. Related Work

Koushik et al. [9] introduced "StoryBlocks," a programming language utilizing block-shaped tangible devices to create audible stories. However, its limited application to storytelling poses challenges in replacing general-purpose programming languages for education. Additionally, the high costs of specialized devices hinder widespread adoption. To address these issues, this study employs a web-based application, similar to Scratch and Blockly. Mountapmbeme et al. [10] proposed "Accessible Blockly," a web-based UI facilitating block navigation via keyboard commands. Each block features multiple connections, allowing for the simultaneous connection of multiple blocks. Consequently, when moving to adjacent blocks, users must explicitly select one of them. "Accessible Blockly" utilizes different keystrokes based on block relationships to facilitate navigation. For instance, to transition from a repeat block to an adjacent one, users utilize the keystrokes illustrated in Fig. 1. Pressing the W key navigates to the previous block, while the F key focuses on the block indicating repetition count. Moreover, the screen reader provides audible feedback on block content as users navigate. In this method, users navigate between blocks by pressing keys based on their relationships. This requires users to know the structure of the currently focused block beforehand. While sighted users can infer it from block shapes, visually impaired users must memorize it, which poses a burden, particularly for beginners with limited programming experience. Additionally, while this study focuses on enhancing keyboard operation and screen reader compatibility, it does not propose solutions for the glanceability issues.

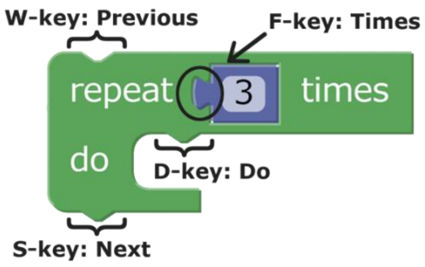


Fig. 1: Navigation method of Accessible Blockly

3. Proposed Method

3.1. Line-based Navigation UI

This study proposes a novel UI that navigates between lines of code instead of code blocks. The existing methods encounter challenges as they focus on navigating visually complex block elements, relying heavily on visual information during operation. In contrast, the lines proposed in this study have a simple structure, do not require visual information, and are compatible with sequential voice reading by screen readers. This approach enables us to provide visually impaired users with a more intuitive and less burdensome coding experience. Figure 2 illustrates the basic navigation method of the proposed system.

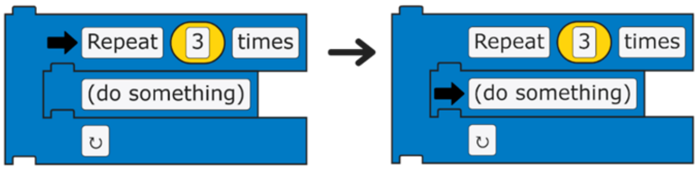


Fig. 2: Navigation from the 1st line to the 2nd line

The black arrow in the figure indicates the currently focused line. Users can navigate between lines by pressing the up and down arrow keys, with the screen reader reading aloud the content of the line they move to. In Accessible Blockly, the block has several different connections, requiring the user to memorize the complex structure of the blocks. However, in the proposed method, each line only connects to the previous and next lines. This allows users to understand the program content by continuously moving to the next line, without needing to consider the block structure. Furthermore, some blocks contain separate input fields within a single line. Users can select these input fields by pressing the left and right arrow keys, as illustrated in Fig. 3.

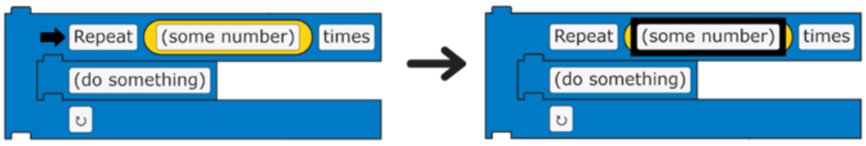
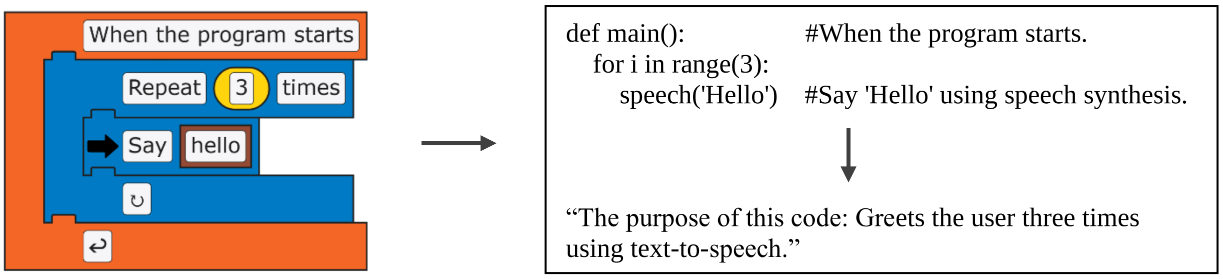


Fig. 3: Navigation from the beginning of the line to the input field

3.2. Automated Code Summarization Using Conversational AI

In this study, a novel debugging support method is proposed to assist visually impaired individuals through automated code summarization using conversational AI. Generating summaries that consider the overall context of the program using AI helps alleviate the unique difficulty faced by visually impaired individuals when reading programs line by line. In this study, Cohere's Chat API [11] is utilized as the language model in this study. When handling programs written in widely-used textual programming languages, such as Python or JavaScript, the language model has already been trained on sufficient data, allowing direct input of the program's content. However, in the case of block-based languages, simply inputting the displayed text on the screen into the language model does not guarantee reliable responses. Especially when developing a custom block-based language, creating teaching data and manually training the language model is impractical. Therefore, a novel method to generate summaries is proposed, involving the compilation of block-based languages into conventional textual languages, like Python, followed by inputting the results into the language model. With this method, it becomes possible to generate summaries of programming languages with high accuracy, provided an algorithm for compilation is prepared.

Figure 4 illustrates the process of compiling code written in a block-based language into a Python program, followed by the generation of summary sentences by conversational AI.



```
def main():              #When the program starts.
    for i in range(3):
        speech('Hello')  #Say 'Hello' using speech synthesis.
```

“The purpose of this code: Greets the user three times using text-to-speech.”

Fig. 4: The flow of summarization

4. Performance Evaluation

To verify the effectiveness of the proposed methods, a performance experiment was conducted in comparison with Accessible Blockly [10]. The experiment involved twenty participants in their twenties. Given the aim of achieving inclusive education, the proposed method had to be evaluated from both visually impaired and sighted perspectives. Consequently, the participants were divided into two groups: one wore blindfolds, and the other did not. Before commencing the experiment, participants received instructions on operating the systems. Once they felt confident, the experiment began. The task involved creating a program to print the phrase 'Hello' three times using both systems.

For evaluation, the time taken by each participant to complete the task was measured. Additionally, a 5-point survey was conducted to evaluate the usability of the system. In all items, "1" indicates discomfort, while "5" indicates comfort, with higher values indicating better results.

Figures 5 and 6 show the results of task completion times for blind and sighted participants, respectively. The results showed significant reductions in task completion times for both groups when using the proposed method. This reduction can be attributed to the intuitive operation of navigating between lines, which shortened the time participants spent figuring out how to operate the system. The experimental results confirm the effectiveness of the novel navigation UI for block-based programming.

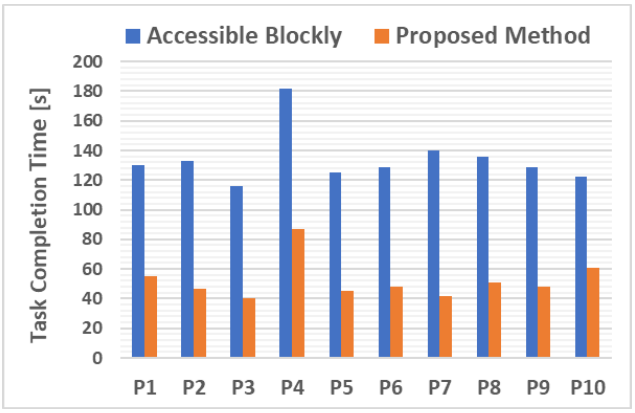


Fig. 5: Task completion time for blind participants

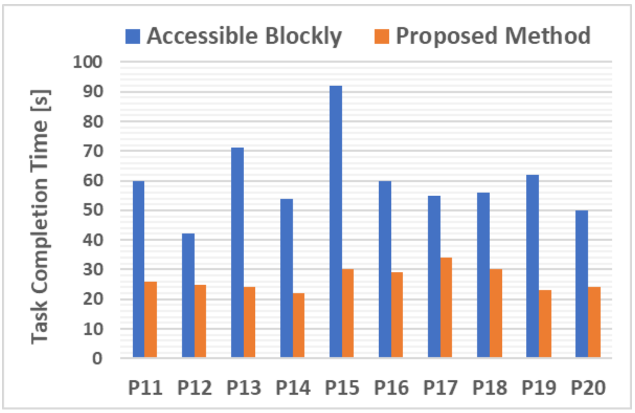


Fig. 6: Task completion time for sighted participants

Figures 7 and 8 display the evaluation results for usability among blind and sighted participants, respectively. The results show that both groups rated usability higher when using the proposed method. Interestingly, blind participants tended to rate the "Comfort Level" higher compared to sighted participants. This might be attributed to the challenging condition of blindness experienced during the experiment, which may have influenced the evaluation criteria for usability.

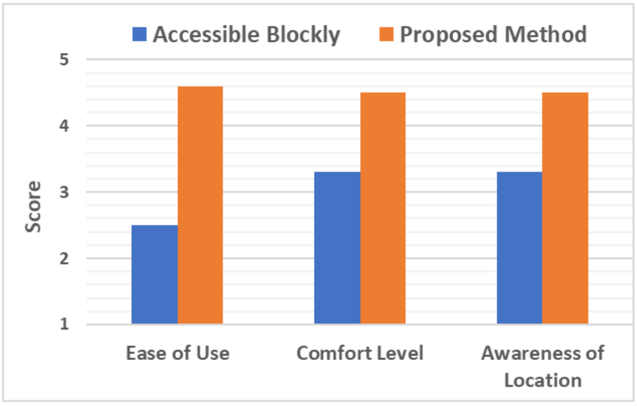


Fig. 7: Usability evaluation for blind participants

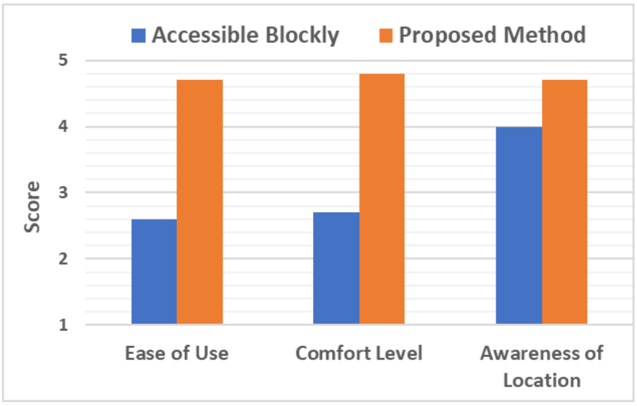


Fig. 8: Usability evaluation for sighted participants

Figures 9 and 10 present the evaluation results for the automated code summarization among blind and sighted participants, respectively. The results show that both groups rated the effectiveness of the automated code summarization highly. A comparison between the two figures reveals that blind participants rated the "Increase in Understanding" aspect higher, while sighted participants rated the "Reduction of Anxiety" aspect higher. These findings suggest that while blind participants have strong expectations regarding enhanced understanding of program content through summarization, they did not experience as significant effects in terms of "Reduction of Anxiety" compared to sighted participants.

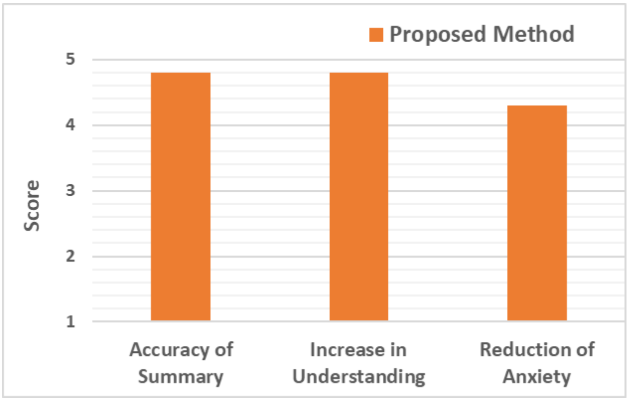


Fig. 9: Summarization evaluation for blind participants

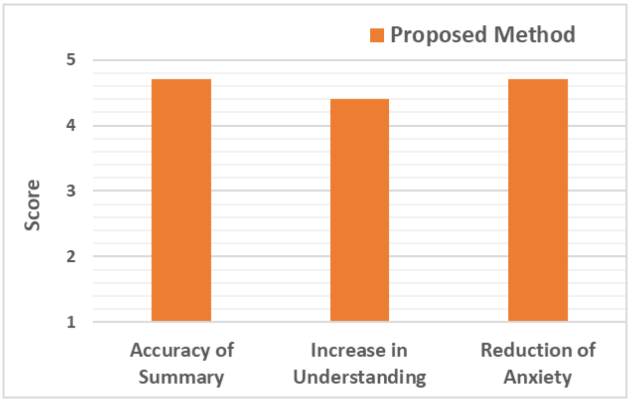


Fig. 10: Summarization evaluation for sighted participants

5. Conclusion

This study aimed to contribute to inclusive education by proposing a novel UI for block-based programming environments, specifically designed to alleviate the burdens faced by individuals with visual impairments. The previous study posed challenges where users were required to memorize block structures beforehand, resulting in significant operational difficulties. Additionally, it lacked effective proposals addressing the specific debugging challenges encountered by individuals with visual impairments. To address these gaps, two solutions were proposed: a novel UI that navigates between lines and automated code summarization using conversational AI. These proposals were implemented in a block-based programming environment, and an evaluation experiment was conducted. The results revealed reduced task completion times for all participants. Regarding the subjective evaluation results, participants found the proposed UI to be easier to use and less frustrating. Moreover, the automated code summarization feature received high ratings in terms of Accuracy of Summary, Increase in Understanding, and Reduction of Anxiety, highlighting its effectiveness in aiding comprehension and reducing cognitive load.

6. Future Work

Firstly, it is essential to refine the system further based on the user feedback gleaned from the evaluation experiment. Additionally, while the evaluation experiment focused on straightforward tasks, namely, printing "Hello" three times, exploring how the proposed method performs with more intricate programs is imperative. Lastly, although the evaluation experiment simulated blindness by having sighted individuals wear blindfolds, indicating effectiveness within that context, obtaining feedback from actual participants with visual impairments is paramount for a comprehensive evaluation of the proposed method. This is identified as a crucial future task in this study.

7. Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers JP23K11952.

8. References

[1] X. Weng et al., “Creativity Development With Problem-Based Digital Making and Block-Based Programming for Science, Technology, Engineering, Arts, and Mathematics Learning in Middle School Contexts,” Journal of Educational Computing Research, 2023, Vol. 61, Issue 2, pp. 304-328.

[2] D. Weintrop et al., “Comparing block-based and text-based programming in high school computer science classrooms,” ACM Transactions on Comp. Education, 2017, Vol. 18, No. 1, pp. 1-25.

[3] Scratch - Imagine, Program, Share. Retrieved Feb. 18, 2024 from: https://scratch.mit.edu/

[4] Blockly | Google for Developers. Retrieved Feb. 18, 2024 from: https://developers.google.com/blockly

[5] S. Riazy et al., “Evaluation of Low-threshold Programming Learning Environments for the Blind and Partially Sighted,” Proc. of the 12th Int. Conf. on Comp. Supported Education, 2020, Vol. 1, pp. 366-373.

[6] A. Mountapmbeme et al., “Investigating Challenges Faced by Learners with Visual Impairments using Block-Based Programming / Hybrid Environments,” Proc. of the 22nd Int. ACM SIGACCESS Conf. on Comp. and Acc., 2020, No. 73, pp. 1-4.

[7] V. Potluri et al., “CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers,” Proc. of the 2018 CHI Conf. on Human Factors in Comp. Systems, 2018, No.618, pp. 1-11.

[8] T. Buchner et al., “Same Progress for All? Inclusive Education, the United Nations Convention on the Rights of Persons With Disabilities and Students With Intellectual Disability in European Countries,” Journal of Policy and Practice in Intellectual Disabilities, 2021, Vol. 18, No. 1, pp. 7-22.

[9] V. Koushik et al., “StoryBlocks: A tangible programming game to create accessible audio stories,” Proc. of the 2019 CHI Conf. on Human Factors in Comp. Systems, 2019, pp. 1-12.

[10] A. Mountapmbeme et al., “Accessible Blockly: An Accessible Block-Based Programming Library for People with Visual Impairments,” Proc. of the 12th Int. ACM SIGACCESS Conf. on Comp. and Acc., 2022, No. 19, pp. 1-15.

[11] Cohere | The leading AI platform for enterprise. Retrieved Feb. 18, 2024 from: https://cohere.com/