A Programming System for Blind and Low-Vision Novice Programmers

Joshua G. Lock
joshua.lock@kcl.ac.uk
King's College London
London, UK

Abstract

Most programming systems offer little or no support for blind and low-vision programmers beyond the bare minimum offered by screen readers—the translation of a spatial, visually rich, and graphical user interface into a linear and ephemeral stream of narrated speech.

This research aims to improve the educational experiences of blind and low-vision novice programmers by designing a novel "born accessible" programming system with auditory outputs and keyboard inputs as the primary interaction model.

Keywords

Accessibility, Program editing, Novice programming

ACM Reference Format:
Joshua G. Lock. 2025. A Programming System for Blind and Low-Vision Novice Programmers. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2025), June 27-July 2, 2025, Nijmegen, Netherlands.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3724389.3731293

1 Context and Motivation

Learning to program is difficult. Novices must learn how to translate their thoughts and ideas into program semantics, and then map those program semantics into the programming system's representation—usually code in blocks or text. This is cognitively demanding work at all levels of experience. Consequently, a significant focus in the development of educational and professional programming systems has been on using rich graphical interfaces to improve program comprehension, navigation, and manipulation outcomes to reduce the cognitive load on programmers.

These visual affordances are inaccessible to blind and low-vision programmers who use screen readers to narrate computer interfaces and their content. The rich visual cues of graphical programming systems are lost in a screen reader's translation to a linear audio stream, and mouse-based direct manipulation is unworkable for many of these users.

This linearisation of spatial graphical interfaces and program code into a single, continuous and ephemeral audio stream—often with confusing pronunciation that differs from how programmers speak the same syntax—compounds the cognitively demanding nature of programming for blind and low-vision programmers.

Our work aims to reduce the difficulties faced by blind and low-vision programmers learning to program in order to enable more equitable participation in programming education.

2 Background and Literature Review

Several researchers have tackled the challenge of making programming systems more accessible to blind and low-vision novice programmers.

Work focused on improving text-based programming systems for undergraduate and adult novices has been underway for several years. Smith et al. [6] presented a system for use by undergraduate novices which provides multiple levels of narration targeting different levels of learner understanding and uses tone and emphasis in the synthesised speech to alert the programmer to special tokens, similar to syntax colouring in visual editors. Stefik et al. [8] proposed a programming language, compiler and debugging architecture which narrates the logical, rather than spatial, context of program code and provides additional context during narration to help blind and low-vision programmers better understand code and the runtime state of the program during debugging. Baker et al. [1] demonstrate a system which helps blind and low-vision programmers more easily navigate the structure of a program to more quickly gain an understanding of it, a design intended to replicate a sighted programmer's skim-reading and provide an alternative to the typical end-to-end linear audio narration of screen readers. Similar techniques were generalised to a programming language independent toolkit by Schanzer et al. [5]

More recently, work has been done to make block-based programming systems more accessible to blind and low-vision novices. Milne and Ladner [3] identified challenges using block-based programming environments on touchscreen devices, then implemented and evaluated a design which uses audio and touch-based spatial cues to improve code understanding. Mountapmbeme et al. [4] modified the popular Blockly library, used to create browser-based block-based programming environments, so that it uses the screen reader as an alternative output channel and a keyboard as an alternative input channel. The keyboard serves double duty as a navigation and editing tool and consequently has separate modes, with a virtual cursor to support moving the navigational focus of the screen readers auditory narration without also changing the edit location. Work by Stefik et al. [7] highlighted the need for both inputs and outputs to be non-visually accessible and describes implementing a block-based editor for the Quorum programming system. The editor is designed to feel like a text-editor when operated purely with a keyboard and favours simple defaults, with more

ITiCSE 2025, Nijmegen, Netherlands

ACM ISBN 979-8-4007-1569-3/2025/06
https://doi.org/10.1145/3724389.3731293

advanced features available as novices become more comfortable with the system.

3 Problem Statement

A gap remains in accessible programming systems for blind and low-vision programmers at the secondary school or middle school age. Children at this age are typically making the transition from block-based programming systems to text-based programming systems, or beginning their programming education with text-based programming systems. Children learning to program at this stage of their education may still be getting used to using a computer with keyboard and mouse, a task which becomes more challenging for blind and low-vision children who are also learning how to use accessibility technologies at the same time as learning to use a computer and learning to program.

The frame-based editing paradigm introduced by Kölling et al. [2] claims to ease the transition from blocks to text by offering a highly visual, yet keyboard first, editing system. In the frame-based paradigm, code writing is simplified by reducing the amount of syntax programmers have to type, minimising errors, while code comprehension is enhanced by the use of visual cues, including: font formatting; semantic colouring of program text and interface elements; and automatic white-space management to emphasise keywords, scope, and structure.

Frame-based systems use a richer graphical vocabulary than text-based systems. Initially, this seems to pose a higher risk of being unusable to blind and low-vision programmers because there are more elements to linearise. However, the underlying model of frame-based editors—which separates program structure from its representation—provides an opportunity to work on a programming system's structure and representation separately. Therefore, we can improve the auditory representation independently of the graphical representation. Instead of a linear representation of a spatial graphical model, we will design an auditory first representation which provides auditory equivalents of the visual navigation and manipulation interactions of the frame-based paradigm in order to bring equivalently simplified code writing to blind and low-vision programmers.

Note that our goal of an auditory first representation does not mean that we intend for blind and low-vision students to use alternative tools to their sighted peers. Programming is predominantly a collaborative task, in both educational and professional settings, and enabling equitable participation in mixed ability settings is necessary to support collaboration and help seeking.

In short, the problem is that the gap in programming systems at secondary school or middle school age is not bridged for blind and low-vision programmers.

4 Research Goals

Through a participatory design process, our research aims to design a user interface for a novice programming system with first-class speech support, based on the frame-based paradigm, with improved code navigation and manipulation interactions for blind and low-vision programmers. We intend to implement core aspects of the design within an existing frame-based educational programming system to enable more equitable collaboration in mixed-ability programming environments.

5 Research Methods

We will follow a participatory design process to achieve our research goals. We have begun by using semi-structured interviews to better understand the lived experiences of blind and low-vision programmers during their programming education and in their present programming practice. We will utilise the insights gained through these interviews and a thorough review of the literature to design an initial prototype, which we will discuss with willing participants in follow-on interviews.

After a series of iterative design and refinement loops, with prototyping to validate key aspects of the design, we will move onto design and implementation within an existing educational programming system. At this stage we intend to more formally evaluate the implemented design with members of the target audience: blind and low-vision programmers at secondary school or middle school who are learning to program with text-based programming systems.

6 Contributions

We expect to make the following contributions:

- add to the understanding of challenges faced by blind and low-vision novice programmers as they learn to program
- design solutions to address the challenges
- propose designs for mixed-ability settings that allow for equitable vision-first and auditory-first collaboration
- validate the designs with a prototype implementation and evaluation with blind and low-vision novice programmers.

References

[1] 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*. Seoul Republic of Korea, 3043–3052. doi:10.1145/2702123.2702589

[2] Michael Kölling, Neil C. C. Brown, and Amjad Altadmri. 2015. Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming. In *Proceedings of the Workshop in Primary and Secondary Computing Education*. London United Kingdom, 29–38. doi:10.1145/2818314.2818331

[3] Lauren R. Milne and Richard E. Ladner. 2018. Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Montreal QC Canada, 1–10. doi:10.1145/3173574.3173643

[4] Aboubakar Mountapmbeme, Obianuju Okafor, and Stephanie Ludi. 2022. Accessible Blockly: An Accessible Block-Based Programming Library for People with Visual Impairments. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*. Athens Greece, 1–15. doi:10.1145/3517428.3544806

[5] Emmanuel Schanzer, Sina Bahram, and Shriram Krishnamurthi. 2019. Accessible AST-Based Programming for Visually-Impaired Programmers. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. Minneapolis MN USA, 773–779. doi:10.1145/3287324.3287499

[6] Ann C. Smith, Joan M. Francioni, and Sam D. Matzek. 2000. A Java programming tool for students with visual disabilities. In *Proceedings of the fourth international ACM conference on Assistive technologies*. Arlington Virginia USA, 142–148. doi:10.1145/354324.354356

[7] Andreas Stefik, Willliam Allee, Gabriel Contreras, Timothy Kluthe, Alex Hoffman, Brianna Blaser, and Richard Ladner. 2024. Accessible to Whom? Bringing Accessibility to Blocks. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. Portland OR USA, 1286–1292. doi:10.1145/3626252.3630770

[8] Andreas Stefik, Andrew Haywood, Shahzada Mansoor, Brock Dunda, and Daniel Garcia. 2009. SODBeans. In *2009 IEEE 17th International Conference on Program Comprehension*. Vancouver, BC, Canada, 293–294. doi:10.1109/ICPC.2009.5090064