Tangibles + Programming + Audio Stories = Fun

Varsha Koushik and Shaun K. Kane
Department of Computer Science
University of Colorado Boulder
Boulder, CO 80309 USA
{varsha.koushik, shaun.kane}@colorado.edu

ABSTRACT
Block-based programming languages enable novice programmers, including children, to learn the basics of programming. However, most block-based programming languages are not accessible to blind and visually impaired users because they rely upon visual drag-and-drop interaction, and because they typically create visual output. To improve access to block-based programming languages, we introduce Story Blocks, a programming toolkit that uses tangible blocks to represent story components, and which produces output in the form of accessible audio stories and games. Story Blocks provides an introductory programming environment that can be enjoyed by people of all abilities.

Keywords
Accessibility; Programming; Tangible Computing

1. INTRODUCTION
Block-based programming languages, such as Scratch and Blockly, provide novice programmers with support in learning programming languages. In particular, creating programs by combining blocks can show a novice how a program is structured, and can allow a novice to see whether a program is correctly assembled. Block-based programming languages often support the creation of simple animations and games so that young programmers can quickly create engaging content [2].

While block-based programming languages offer many benefits for novice programmers, most block-based languages are not accessible to blind and visually impaired users. This inaccessibility is caused primarily by two issues. First, block-based languages are programmed using drag-and-drop interactions on a graphical user interface, and these user interfaces are not easily translated for screen reader users. Second, the output of block-based languages often takes the form of graphic animation or games that are not accessible (and therefore not engaging) to blind learners.

To explore alternative input and output modes for block-based programming, we introduce the Story Blocks programming environment (Figure 1). In Story Blocks, users create programs by assembling physical blocks that represent characters and actions within an interactive story. The user can then scan this program using a mobile device camera. The program is executed as an audio story on the mobile device. Thus, Story Blocks provides both accessible tools for creating programs as well as universally accessible output. Creating more accessible programming tools may support children of all abilities in exploring computer science.

ASSETS '17, October 29-November 1, 2017, Baltimore, MD, USA

ACM ISBN 978-1-4503-4926-0/17/10.
https://doi.org/10.1145/3132525.3134769

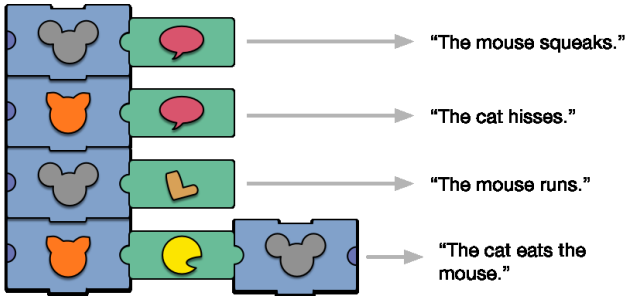

Figure 1. The Story Blocks programming language uses tangible blocks as the basis for programs. These blocks offer tactile features that allow them to be used by both blind and sighted users. The user assembles a program by connecting blocks, and captures an image of the program on a mobile device. The program is then executed on the phone in the form of an interactive audio story.

2. RELATED WORK
Several programming languages have been designed to support accessibility for blind and visually impaired users. Quorum [11] is a text-based programming language that provides explicit support for programming via a screen reader. Other researchers have explored ways to provide access to block-based languages by adding audio. Ludi [8] proposed extending the block-based language Blockly to provide access for screen readers. PseudoBlocks [7] is a nonvisual pseudo-spatial programming language that allows users to manipulate blocks using keyboard commands, and provides program output via speech. Blocks4All [9] allows users to explore a block-based program by touching a touch screen and receiving speech feedback. Story Blocks provides non-visual access to block-based code, but uses tangible input in addition to audio feedback.

Tangible programming tools also offer the potential to increase access to block-based programming. Strawbies [4] enables children to control an on-screen character by assembling a series of instruction blocks; blocks are scanned by a mobile device camera and are used to direct an on-screen character. Project Bloks [1] supports block-based programming by assembling a series of blocks with embedded sensors and actuators. However, these systems use blocks that are only identifiable visually. In this work, we are exploring how the tangible design of blocks can be used to create programs that are readable by a blind or visually impaired user. Project Torino [12] adopts a similar approach to Project Block, but is specifically designed to support both blind and sighted children. Users can assemble programs in Project Torino by connecting blocks with a cable, and can create programs that output music or spoken audio. However, Project Torino requires sophisticated hardware devices for each block, and thus may not scale well to large numbers of learners, such as in a large classroom. Story Blocks builds upon these tangible programming tools, but

focuses on the design of low-cost tangible blocks that can be used to create expressive audio output.

Enabling novice programmers to create stories and games can be effective in encouraging programming practice, especially for younger learners. Alice [3] and Storytelling Alice [6] are block-based programming tools that enable novice programmers to write programs that control on-screen characters. Story Blocks explores programming of audio stories and simple games as a way to provide compelling non-visual output for novice programmers.

3. STORY BLOCKS

Our ongoing work in the Story Blocks project is motivated by the following goals. First, we wish to provide tangible access to block-based programming languages that can be used by both blind and sighted users. Second, we wish to enable the creation of compelling non-visual programs for novice programmers. Finally, we wish to support scalability of the system, enabling multiple students in a classroom to create programs simultaneously. These goals have been embodied in the current Story Blocks prototype.

3.1 System Design

The Story Block programming language consists of a series of physical blocks that can be joined together to create a program (Figure 1). In the current prototype, these blocks are made from low-cost materials and fabricated using a 3D printer or laser cutter. Each block is designed to be identifiable by its shape and color so that blocks can be identified by both blind and sighted users. As in other block-based programming languages, the shape of each block indicates how it may connect to other blocks. Blocks are marked with a distinctive dot pattern, such as a reacTivision tag [5], so that they may be tracked by the system.

The current prototype supports three types of blocks: character blocks, which represent characters in the story such as the Cat, Mouse, and Princess; action blocks, which represent actions taken by the characters, such as talking, eating, and running; and control blocks, which support conditional branching, looping, and other aspects of program flow. Figure 1 shows a simple Story Blocks program in which a cat encounters a mouse.

The blocks used in this system are constructed from passive materials such as cardboard, wood, or plastic. Execution of the program is controlled by a companion application running on a PC or mobile device. A user can compile their program by capturing a photo of the blocks. Once the program is loaded into the application, the user can execute the story using standard screen reader commands. For each line of code, the companion application will read the next part of the story. The companion application can embellish each line by adjusting the text-to-speech voice to represent different characters, and by adding atmospheric music and sound effects. While the current prototype only supports static stories, future versions of Story Blocks will support the creation of interactive stories and audio games.

4. CURRENT PROGRESS

Our current prototype supports the creation of simple stories with a limited number of characters. We intend to extend the current system by adding additional characters and actions, and by providing additional support for playing atmospheric sound and music. We are also investigating alternative designs for the blocks themselves, exploring tradeoffs between different block sizes, materials, and tracking methods.

We are also beginning a study of how K-12 students and teachers use Story Blocks. In particular, we hope to gather feedback about the current prototype and to explore how students and teachers may integrate Story Blocks into their current activities.

5. CONCLUSION

While block-based programming languages offer many opportunities for novice programmers, many of the features of these languages remain inaccessible. By combining tangible input with engaging audio output, Story Blocks may offer a compelling approach to introducing programming concepts to students with a variety of abilities.

6. ACKNOWLEDGMENTS

We thank Clayton Lewis, Mike Horn, Erin Buehler, Lauren Milne, and Richard Ladner for their feedback on this work.

7. REFERENCES

[1] Blikstein, P., Sipitakiat, A., Goldstein, J., Wilbert, J., Johnson, M., Vranakis, S., Pedersen, Z. and Carey, W. (2016). Project Bloks: designing a development platform for tangible programming for children.

[2] Brennan, K., and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proc. AERA '12,* 1-25.

[3] Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges 15, 5,* pp. 107-116.

[4] Hu, F., Zekelman, A., Horn, M., and Judd, F. (2015). Strawbies: explorations in tangible programming. In *Proc. IDC '15,* 410-413.

[5] Jordà, S., Geiger, G., Alonso, M., and Kaltenbrunner, M. (2007). The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proc. TEI '07,* 139-146.

[6] Kelleher, C., Pausch, R., and Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. In *Proc. CHI '07,* 1455-1464.

[7] Koushik, V., and Lewis, C. (2016). An accessible blocks language: work in progress. In *Proc. ASSETS '16,* 317-318.

[8] Ludi, S. (2015). Position paper: Towards making block-based programming accessible to blind users. *IEEE Blocks and Beyond Workshop*, 67-69.

[9] Milne, L. R. (2017). Blocks4All: making block programming languages accessible for blind children. *ACM SIGACCESS Accessibility and Computing* 117, pp. 26-29.

[10] Schanzer, E., Fisler, K., Krishnamurthi, S., and Felleisen, M. (2015). Transferring skills at solving word problems from computing to algebra through Bootstrap. In *Proc. SIGCSE '15*, 616-621.

[11] Stefik, A., and Siebert, S. (2013). An empirical investigation into programming language syntax. *ACM Transactions on Computing Education 13*(4), Article 19, 40 pages.

[12] Thieme, A., Morrison, C., Villar, N., Grayson, M., and Lindley, S. (2017). Enabling collaboration in learning computer programing inclusive of children with vision impairments. In *Proc. DIS '17,* 739-752.