Inclusivity of Computing Education for Learners with Visual Impairments

Alex Hadwen-Bennett
King's College London
London, UK
alex.hadwen-bennett@kcl.ac.uk

Anja Thieme
Microsoft Research
Cambridge, UK
anthie@microsoft.com

Abstract

In this paper, we consider some of the barriers to computing education for learners with visual impairments (VI) and identify the inaccessibility of most existing block-based programming languages (BBLs) as a particular obstacle to the learning process. We suggest that physical programming languages (PPLs) could potentially be employed to overcome this obstacle. To this end, we present *Torino*, a physical programming language designed specifically to support collaborative learning experiences for children with mixed visual-abilities. Following a user-centric, iterative design process and initial pilot evaluation, over 30 sets of the technology are currently being trialed across the UK. In this paper, we discuss some of the challenges that we face in the development and evaluation of assistive technology for education, drawing on examples encountered during the development and evaluation of Torino.

Author Keywords

Visual impairments, computing education, accessibility, inclusive design; real-world technology deployment.

ACM Classification Keywords

K.4.2 [Social Issues]: Assistive technologies for persons with disabilities.

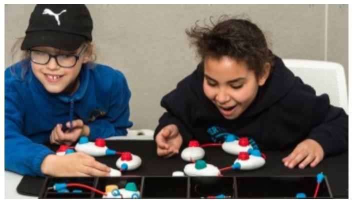
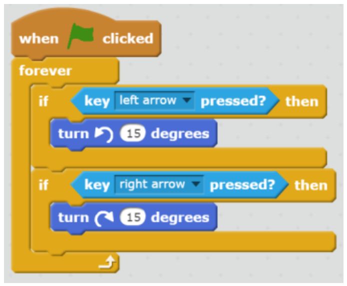


Figure 1. Torino in use.



```
when  clicked
forever
  if key left arrow pressed? then
    turn 15 degrees
  if key right arrow pressed? then
    turn 15 degrees
```

Figure 2. Example Scratch Program

Introduction

Technology has become ubiquitous, requiring a society that is proficient in its use of technology. It is estimated that over half of the workforce today require a high level of digital skills, which include amongst others financial modelling, content creation and social media analysis [8]. Currently, 50% of top paying jobs in the US require at least some coding skills [2]. In addition to coding, computational thinking skills are identified as one of the 10 skills that will be needed by workers in 2020 [3]. Despite the growing importance and potential value of technology, there is still a divide between those that have access to, and the skills to use it, and those that do not; and this gap especially pronounced for disabled populations (cf. [7]).

The recent introduction of computing into the national curriculum for England in 2014 [4], brought with it the requirement for children to be taught basic computing knowledge and skills from as early as the age of five. This is a great step forward, however there are many barriers to computing education for learners with visual impairments (VI). Next, we describe existing barriers to their learning of computing skills and discuss challenges that HCI researchers often face in addressing those.

Accessibility of Computing Education

Many modern programming environments are inaccessible to VI learners, as they are often impossible to interface with using a screen reader [1]. Currently the most popular languages for introductory programming in primary schools in the UK are block-based [5]. Block-based languages (BBLs) such as Scratch (see Figure 2) enable learners to develop programs by snapping virtual blocks together that are illustrated on a digital screen, removing the need for them to learn the complex syntax of a text-based language. However, BBLs are intrinsically visual, utilizing drag-and-drop interactions or graphical animations, that are inaccessible to most VI learners. Physical programming languages (PPL) present a viable alternative to BBLs. Here, commands are represented by physical objects which can be joined together to create programs. One recent example of an accessible PPL is Torino (see Figure 1), which we will describe in more detail below.

Torino

Torino is a physical programing language for teaching basic programing constructs and computational thinking to children aged 7-11, inclusive of children with VI. For the design of Torino, we partnered with two blind and two partially-sighted children to generate new ideas for, and prototypes of, technology over a period of 18 months. Next, we explicate some of the key decisions made in the design of Torino to support accessibility and collaborative learning experiences.

Design Rationale

To create programs with Torino, physical 'command pods' are connected to each other, which produce sound in the form of music, stories and poems [6]. Torino features four main types of command pods: *play*, *pause*, *loop* and *selection*, each of which represents a line of code in the program (see Figure 3). Adding a play pod instructs the program to play a sound that can be altered using a dial that rotates through a number of available sounds. The pause pod adds a delay between two commands in the program. Both pods also have dials to increase or decrease the duration of play or pause. The loop pod allows for those command pods that are added, to repeat, and can be

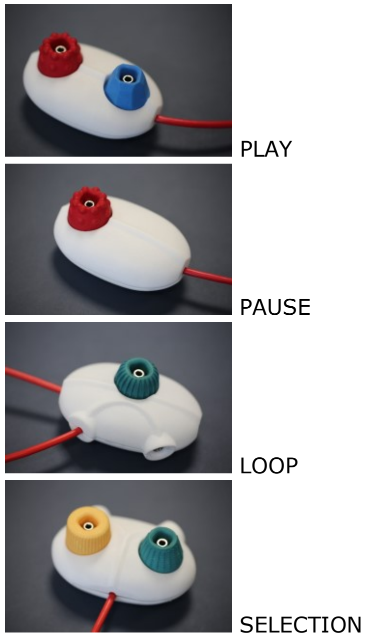


Figure 3. Torino pod types.

set to be ‘infinite’ or to cycle through a specific number of ‘iterations’. To facilitate their identification, each pod type is further distinct in size and shape, and their physical controls (dials) emphasized using high contrasting colors. Additionally, there is a selection pod that enables the program to take one of two paths, based on the result of a condition that is set via dials.

To create programs, command pods are connected to a hub, which features buttons to play and stop the program, along with in-built speakers so the output of the program can be heard. Each pod contains a custom circuit board, containing a microcontroller and connectors that provide power and communication to connected pods, allowing them to form a network. With the electronics and controls locally embedded in each pod, real-time audio feedback is played via a speaker in the hub in response to direct manipulations. Further, because it is necessary to connect pods to the hub, the hub acts as a *starting point* – a physical reference to the origin of the program; whilst the directionality of the *program flow* can be inferred by following the networked pods. Furthermore, each time a pod is added or removed from the network this is acoustically represented by a distinct ‘connect’ or ‘disconnect’ sound to support awareness and keeping track of both one’s own and other people’s interactions with the system.

Torino Evaluation

In an initial pilot study, two researchers (one of whom has a background in teaching), facilitated engagements with Torino for five pairs of children with mixed-visual abilities. The study showed how the specific multi-modal configuration of the technology, the programing tasks (e.g. the joint creation of a seven note piano scale) and the social interactions of using the tool with a peer, supported the children’s sense-making of Torino and their learning of computing skills. Building on these initial findings, we iterated the technology and deployed over 30 Torino sets, which are currently being used in a larger-scale trial across the UK.

A curriculum was developed for use in the trial, that aligns with the English National Curriculum, thus enabling teachers to address programming related learning objectives using Torino. The curriculum was written by an experienced computing teacher and reviewed by an expert in primary computing education, whose feedback fed into a second iteration of the curriculum. The content covered in the curriculum includes: sequence, selection, loops, debugging, decomposition and variables.

Over 25 parents and teachers are participating in the trial, most of whom are not computing specialists. This poses unique challenges as to how to best support them in using the tool in-the-wild. To this end, we provided them with extensive training materials, a comprehensive teachers’ guide that includes solutions to all the activities in the curriculum as well as example questions to ask to the children; and also video tutorials to help them get started with Torino.

Through this trial we aim to evaluate the effectiveness of different aspects of Torino. One particular challenge here is the development of valid research instruments which are suitable for VI children. These include the effectiveness of the technology, the curriculum, and users’ experiences of learning with Torino. The research instruments we will be employing include pre-post questionnaires and reflective teacher diaries. Part of the questionnaires are designed to assess how the children

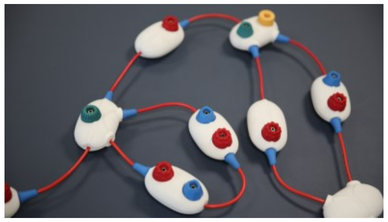


Figure 4. Example program: Shows a LOOP that repeats to PLAY pod instructions, followed by a PAUSE and then a SELECTION that either executes a PAUSE or a PLAY pod.

perceived the technology how its use impacted their coding ability and understanding of computer science more generally. To this end, we asked the children before they got started to describe to us a computer scientist and to rate their coding ability using a 5 star rating scale. Further, we are gathering reflections from the adult participants that are administering additional research instruments for the current study.

Additional research challenges and questions of interest include for example: To what extent is Torino inclusive of all learners? How feasible is its use for educating VI learners in mainstream schools? How can the transition from Torino to text-based languages be facilitated?

Conclusion

The development and on-going evaluation of Torino has highlighted challenges in the creation of assistive technologies for education, specifically for people with VI. In this workshop, we seek to discuss these challenges with other researchers and practitioners in this space, to share experiences from our journey with Torino, reflect on lessons learned, and to jointly work towards developing the agenda for future work.

Acknowledgements

Special thanks the Project Torino research team; and all teachers, parents and students supporting this work.

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 - CHI '15*, 3043–3052. https://doi.org/10.1145/2702123.2702589
2. Burning Glass Technologies. 2016. *Beyond Point and Click the Expanding Demand for Coding Skills*. Boston, MA.
3. Anna Davies, Devin Fidler, and Marina Gorbis. 2011. *Future work skills 2020*. Palo Alto, CA.
4. Department for Education. 2014. The national curriculum in England - Framework document. Retrieved November 20, 2017 from https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/381344/Master_final_national_curriculum_28_Nov.pdf
5. The Royal Society. 2017. *After the reboot: computing education in UK schools*. Retrieved November 20, 2017 from https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf
6. Anja Thieme, Cecily Morrison, Nicolas Villar, Martin Grayson, and Siân Lindley. 2017. Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems - DIS '17*, 739–752. https://doi.org/10.1145/3064663.3064689
7. UK Digital Skills Taskforce. 2014. *Digital Skills for Tomorrow's World*. Retrieved May 12, 2017 from http://www.ukdigitalskills.com/wp-content/uploads/2014/07/Binder-9-reduced.pdf
8. UK forum for Computing Education. 2015. *Digital Skills Taskforce call for evidence: Submission from the UK forum for Computing Education*. Retrieved from http://data.parliament.uk/writtenevidence/committeeevidence.svc/evidencedocument/digital-skills-committee/digital-skills/written/12323.pdf