Physical Programming for Blind and Low Vision Children at Scale

Nicolas Villar
Microsoft Research
Redmond, WA
nvillar@microsoft.com

Cecily Morrison
Microsoft Research
Cambridge, UK
cecilym@microsoft.com

Daniel Cletheroe
Microsoft Research
Cambridge, UK
daclethe@microsoft.com

Tim Regan
Microsoft Research
Cambridge, UK
timregan@microsoft.com

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

Greg Saul
Microsoft Research
Cambridge, UK
gregsaul@gmail.com

ABSTRACT

There is a dearth of appropriate tools for young learners with mixed visual abilities to engage with computational learning. Addressing this gap, we present Project Torino, a physical programming language for teaching computational learning to children ages 7-11 regardless of level of vision. To create code, children connect and manipulate tactile objects to create music, audio stories, or poetry. Designed to be made and deployed at scale, Project Torino (along with a scheme of work) has been successfully used by 30 non-specialist teachers with 75 children across the UK over three months.

CHI'19 Extended Abstracts, May 4-9, 2019, Glasgow, Scotland, UK.

ACM ISBN 978-1-4503-5971-9/19/05.
DOI: https://doi.org/10.1145/3290607.3313241


Figure 1: Two learners with mix visual abilities using Torino: a physical programming language for teaching computational learning to children ages 7-11 regardless of level of vision.

KEYWORDS

Accessibility; tangible education technology; inclusive learning; mixed visual abilities; blind; computational learning; evaluation at scale; computer science education (CSE).

CONTRIBUTION

- Presents a novel physical programming language for children ages 7 – 11 with mixed visual abilities that has been successfully manufactured and used at scale with more than 75 children and 30 teachers to support computational learning.
- Illustrates design features of *persistent program overview* and *liveness* that support non-specialist teachers to co-produce learning using Torino with children of different ages, visual, and cognitive abilities.

1 INTRODUCTION

Policy initiatives throughout the world are including programming, and computational learning more broadly [2], into education and national curricula. Addressing these initiatives, a range of specialist teaching tools have been created to encourage the development of computational learning in school children [3]. Yet, the formalization of computational learning into schools and curricula makes more apparent the lack of appropriate tools for many children with disabilities [1].

While inclusive computing (see Sidebar for references) is a growing area of research, there remains a dearth of tools for teaching basic computational learning to young children with mixed visual abilities. Common languages used by their sighted peers, such as Scratch or Alice [9] are visual both in manipulating the code (e.g. drag and drop) and in the effect that the code has (e.g. animation). Existing physical programming languages also rely heavily on visual properties for: distinguishing pieces, connecting them correctly, and experiencing the outcome of the program.

Addressing this gap, Project Torino is a physical programming language for teaching computational learning to children ages 7-11 regardless of level of vision (Figure 1). To create code, children connect and manipulate tactile objects to create music, audio stories, or poetry. A scheme of work supports non-specialist teachers in delivering lessons with the system. The design of the initial prototype and early evaluations are reported in [5,8]. Multiple design iterations have resulted in a new, full-featured, manufacturable version of Project Torino that can be deployed at scale.

We have validated Project Torino with 75 children and 30 teachers situated across 24 localities in the UK for a full three-month academic term. This stands in strong contrast to other physical (or tangible) technologies that have been developed for teaching computational learning, for which empirical validation is rare [11]. Our findings showed that children were highly engaged and that teachers reported age- and ability-appropriate learning across the cohort.

2. PROJECT TORINO SYSTEM

A Project Torino kit is made up of three kinds of physical entities: one hub, 15 pods, and 12 plugs. These can be connected together in various ways to create programs.

Program Structure: Each pod represents an action to be executed sequentially when the program is run. In this latest version of the system, in addition to *play*, *rest*, and *loop* pods, there are also *conditional* (if then), and *merge* (end if) pods. Each pod has a number of connectors and cables that are plugged together to define the program structure. Pods plug into one of four connectors on the hub, which represents the logical starting point of the program, with each connection point representing the start of a thread. Figure 2 shows an example program with three threads. The first thread will play a single sound, and then will enter a loop that will repeat the action of playing a rest (silence) followed by a sound. A second thread will play three sounds in sequence. A third thread will play either a sound or a rest, depending on the outcome of a conditional statement.

Parameters and Values: Pods have knobs that represent configurable parameters, which can be rotated to specify the value of the parameter. Play pods have two knobs, one to specify the sound

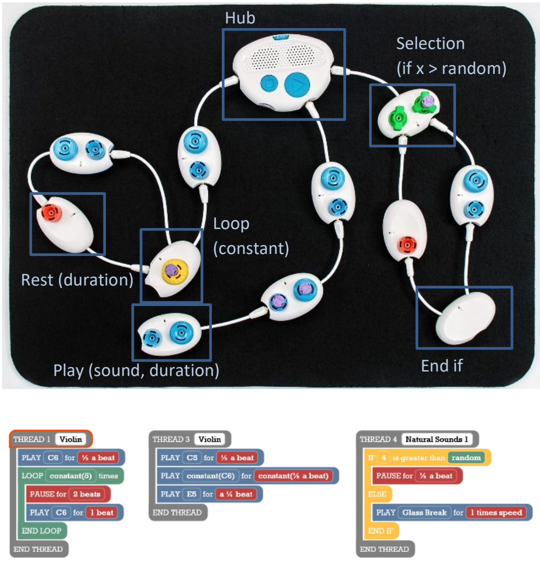

Figure 2: Torino system pieces connected in a multi-threaded program with text-based code below.

INCLUSIVE COMPUTING

Hadwen-Bennett et al. 2018. Making Programming Accessible to Learners with Visual Impairments: A Literature Review. *International Journal of Computer Science Education in Schools*, 2, 2.

Kane et al. 2018. Bonk: accessible programming for accessible audio games. *In Proceedings of IDC Conference on Interaction Design and Children*, 132–142.

Lechelt et al. 2018. Inclusive Computing in Special Needs Classrooms: Designing for All. *In Proceedings of CHI Conference on Human Factors in Computing Systems*, 517.

Milne and Ladner. 2018. Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments. *In Proceedings of CHI Conference on Human Factors in Computing Systems*, 69.

Stefik et al. 2011. On the design of an educational infrastructure for the blind and visually impaired in computer science. *Proceedings of CSE Technical Symposium on Computer Science.*

and one duration; rest pods have a single knob for the duration of silence; the knob on a loop pod specifies the number of loop iterations; and the pair of knobs on a conditional pod represent the values of the conditional statement x > y. New to this version of Torino are plugs that can be inserted into the knobs to *programmatically* change their values, enabling taught concepts to include constants and variables. The set of plugs include: constant values 1-8, random, infinity, increment and decrement counters as well as variables.

Multisensory Feedback: Each physical entity was designed to be tactually and visually distinct. Each pod has different slopes and textures along with differentiated placement and number of dials. Dials each have a distinct texture and are colored to support those using visual information. This includes sighted children and teachers, as well as many blind and low vision children. We were careful to avoid disparity between tactual and visual information to ensure unimpeded interaction between those of different visual abilities. Manipulating knobs and plugs will result in an immediate audio response that indicates their value. This *liveness* was a specific design feature used to mimic the liveness of ILEs [7]. It was derived from our iterative design process that highlighted the ways children engaged with the world through their hands [5].

Reviewing Code: Students are expected to read and understand their code physically. They are particularly encouraged to follow their program as it executes, precisely touching or pointing to each pod as the program progresses. Research has noted that incorrect mental models can form when there is an inadequate understanding of the 'hidden' processes that are not directly observable from the program [6]. As a result, Torino followed the design construct to provide a *persistent program overview* of the program at all times. Combined with the physicality of the program, this design approach encourages computational learning through planning and prediction (algorithmic design [10]), and by following program execution (tracing and debugging [4]). Physical program-following has the added benefit of supporting shared attention between learners and can assist in debugging as the learner's hand is already in position to fix the bug when spotted.

As students get ready to transition from Torino to a text-based language, they can listen to an audio description of the code, similarly to how a text-based program can be understood with a screen reader. Those with vision can view their code in software in an appropriate visual medium. We found teachers and adults often used the visual code (see Figure 2).

Low Threshold, High Ceiling: Project Torino was deliberately designed to accommodate a broad range of capabilities. Younger children (or those with additional learning needs) can start with very simple sequential programs, manipulating parameters manually. Complex concepts, such as nested looping, conditional logic and programmatic parameter manipulation can be introduced gradually as the learner progresses.

Implementation: Each pod contains a custom-designed circuit board, a microcontroller and connectors which power connected pods and enable them to communicate. Control messages, including type of pod and current state, are propagated through the network until they reach the Hub. From these messages a network graph is constructed, where a node is a pod and the edges are the connections between them.

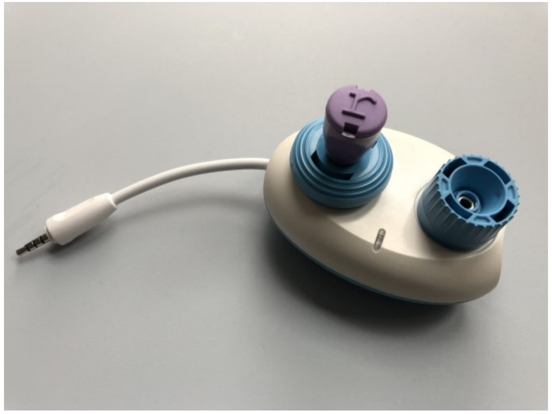

Figure 3: An illustration of a plug to programmatically change a parameter value.

PHYSICAL PROGRAMMING LANGUAGES

Horn and Jacob. 2007. Tangible Programming in the Classroom with Tern. *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1965-1970.

Lechelt et al. 2016. ConnectUs: A new toolkit for teaching about the Internet of Things. *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 3711–3714.

Sullivan et al. 2015. KIBO robot demo: Engaging young children in programming and engineering. *Proceedings of the IDC Conference on Interaction Design and Children*, 418–421.

Zuckerman et al. 2006. Flow blocks as a conceptual bridge between understanding the structure and behavior of a complex causal system. *Proceedings of International Conference on Learning Sciences*, 880–886.

Sentence et al. 2017. Teaching with physical computing devices: the BBC micro: bit initiative. *Proceedings of Workshop on Primary and Secondary Computing Education*, 87–96.

The audio processing and visual output is done on a linked device, such as a tablet. The design is intended to be extensible, allowing for the introduction new pods and plugs to extend the program syntax.

Demonstration: This will be the first time that Torino is demonstrated at a scientific conference. Attendees will be able to explore all design features described as well as to build their own computer programs using Project Torino. We will further showcase companion experiences and the teaching guides that were developed, and discuss with the audience lessons learned in designing and deploying educational technology with, and for, children of different ages, visual and cognitive abilities.

ACKNOWLEDGMENTS

We would like to acknowledge all the learners and teachers who have participated in Project Torino, its design and evaluation. We would like to acknowledge the broader research team who participated in this project at different stages.

REFERENCES

[1] Sheryl E. Burgstahler and Richard E. Ladner. 2007. Increasing the participation of people with disabilities in computing fields. *Computer* 40, 5: 94–97.
[2] Stephen Cooper, Lance C. Pérez, and Daphne Rainey. 2010. K--12 computational learning. *Communications of the ACM* 53, 11: 27–29.
[3] Caitlin Duncan, Tim Bell, and Steve Tanimoto. 2014. Should your 8-year-old learn coding? *Proceedings of the 2014 WiPSCE Workshop on Primary and Secondary Computing Education.*, 60–69.
[4] Raymond Lister, Elizabeth S. Adams, Sue Fitzgerald, et al. 2004. A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin* 36, 4: 119–150.
[5] Cecily Morrison, Nicolas Villar, A Thieme, et al. 2018. Torino: A tangible programming language inclusive of children with visual disabilities. *Human Computer Interaction.*
[6] Juha Sorva. 2013. Notional Machines and Introductory Programming Education. *ACM Transactions of Computing Education* 13, 2.
[7] S. Tanimoto. 1990. VIVA: A visual language for image processing. *J. Vis. Languages Computing*: 127–139.
[8] Anja Thieme, Cecily Morrison, Nicolas Villar, Martin Grayson, and Siân Lindley. 2017. Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments. *Proceedings of the 2017 DIS Conference on Designing Interactive Systems*, 739–752.
[9] Ian Utting, Stephen Cooper, Michael Kölling, John Maloney, and Mitchel Resnick. 2010. Alice, Greenfoot, and Scratch – A Discussion. *Trans. Comput. Educ.* 10, 4: 17:1–17:11.
[10] Jane Waite, Paul Curzon, William Marsh, Sue Sentance, and Alex Hawden-Bennett. 2018. Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal of Computer Science Education In Schools.*
[11] Bieke Zaman, Vero Vanden Abeele, Panos Markopoulos, and Paul Marshall. 2012. The evolving field of tangible interaction for children: the challenge of empirical validation. *Personal and Ubiquitous Computing* 16, 4: 367–378.