TORINO: A TANGIBLE PROGRAMMING LANGUAGE INCLUSIVE OF CHILDREN WITH VISUAL DISABILITIES

Running Head: Torino

ABSTRACT

Across the world, policy initiatives are being developed to engage children with computer programming and computational thinking. Diversity and inclusion has been a strong force in this agenda, but children with disabilities have largely been omitted from the conversation. Currently, there are no age appropriate tools for teaching programming concepts and computational thinking to primary school children with visual disabilities. We address this gap through presenting the design and implementation of Torino, a tangible programming language for teaching programming concepts to children age 7-11 regardless of level of vision. In this paper, we: 1) describe the design process done in conjunction with children with visual disabilities; 2) articulate the design decisions made; and 3) report insights generated from an evaluation with 10 children with mixed visual abilities that considers how children are able to trace (read) and create (write) programs with Torino. We discuss key design trade-offs: 1) readability versus extensibility; and 2) size versus liveness. We conclude by reflecting upon how an inclusive design approach shaped the final result.

CONTENTS

1 INTRODUCTION

Technology now plays a widespread role in our daily lives, necessitating a society literate in the computational aspects of digital media. This has resulted in policy initiatives throughout the world to engage children with computer programming (Langdon, McKittrick, Beede, Khan, & Doms, 2011; Peyton Jones, 2013). To support this educational aim, a range of specialist teaching tools have been designed and developed to encourage the learning process. Scratch (Resnick et al., 2009) is one of the best known, but the momentum continues with the development and uptake of new physical technologies, such as Microbit (Rogers et al., 2017).

Diversity and inclusion has been a key part of the movement to encourage children to program. The biggest emphasis has been on getting girls involved (Kelleher, Pausch, & Kiesler, 2007), including the development of specific tools. LilyPad Arduino e-textiles, for example, aim to integrate computing into activities and communities in which women traditionally engage (Buechley, Eisenberg, Catchen, & Crockett, 2008). More general efforts have been made to engage children across underrepresented categories using physical technologies which emphasize making e.g. (Johnson, Shum, Rogers, & Marquardt, 2016). Recent efforts however, have begun to highlight a largely unsupported demographic in computer science education, students with disabilities (Burgstahler & Ladner, 2007).

Currently, there is a dearth of tools for teaching programming concepts to young school children ages 7 – 11 with visual disabilities. Common languages used by their sighted peers, such as Scratch (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010) or Alice (Cooper, Dann, & Pausch, 2000), are visual both in manipulating the code (e.g. drag and drop) and in the effect that the code has (e.g. animation). Although tangible or physical programming languages may be better suited to teaching programming to children with visual disabilities, existing systems also rely heavily on visual properties (Horn & Jacob, 2007). Distinguishing pieces, connecting them correctly, and experiencing the outcome of the program are all visual activities.

Addressing this gap, *Torino* is a tangible programming language for teaching programming concepts to children age 7-11 regardless of level of vision. To create code, children connect physical instruction beads and tune their parameters to create music or stories using the Sonic Pi language as shown in Figure 1.

In presenting Torino, we contribute to the HCI research literature the:

> The design and implementation of a novel tangible programming language that can be used by children ages 7 -11 regardless of their level of vision.

The contribution includes design insights generated from formative workshops with children with visual disabilities as well as reflections on usage of the completed prototype. An overview video is provided in Video Figure 1.

[INSERT FIGURE 1 Here]

We begin with a summary of insights from a broad set of literatures relevant to the design of a new tangible programming language inclusive of children with visual disabilities. We then describe a series of workshops held with children with visual disabilities to explore non-visual approaches to tangible interface design. Building on these insights, we introduce the design and implementation of Torino. Finally, we present a preliminary evaluation of Torino with 10 children who have a range of visual abilities. We conclude with how the inclusive design approach taken fundamentally altered how we thought about tangible programming languages.

2 LITERATURE REVIEW

2.1 Programming Tools for Visually Disabled Students

A number of other tools have been developed specifically to support blind programmers. One of the more recent ones is *StructJumper,* which creates a hierarchical tree of the nesting structure of the program to enable programmers using screen readers to more easily get an overview of their code (Baker, Milne, & Ladner, 2015). Other examples focus more specifically on educational settings. *JavaSpeak* (A. C. Smith, Francioni, & Matzek, 2000) provides structural and semantic information for a Java program; *Audio Programming Language* (APL) (Sánchez & Aguayo, 2005) intends to simplify program logic and allows the writing of programs by choosing one of five commands and following step-by-step instructions. *Quorum* is perhaps the most used language among students with visual disabilities, developed as part of a project to develop a blind-centric computing curriculum (Stefik, Hundhausen, & Smith, 2011). All these systems highlight the challenges of getting an overview of a program and understanding scope.

A number of researchers have also written on curated teaching experiences for engaging visually disabled students with computer science at the high school level. Kane & Bigham discuss a week long workshop to teach programming to teenagers through writing code to 3d-print tactile graphics visualizing crisis data (Kane & Bigham, 2014). Bigham et al. also documents the use of chat bots to engage students with computer science in a similar week long workshop (Bigham et al., 2008). In both papers, the authors promote the idea of programming from the start. They propose that while programming with a screen reader is challenging, enabling students to program in a supportive environment is more likely to give them the confidence to take a computing class in a mainstream environment during which they will be expected to program.

While the majority of systems focus on screen experiences, there are several tangible examples. *Jbrick* was created to enable the programming of Lego Mindstorm Robots for teenagers. It is compatible with screen readers, refreshable braille displays, and magnification software with additional markers for navigating the code (Ludi, Ellis, & Jordan, 2014). Work has also been done on using 3D printing to enable the teaching of computer science concepts such as data structures (Papazafiropulos, Fanucci, Leporini, Pelagatti, & Roncella, 2016). Both of these try to address engagement of tactile learners as well as accessibility issues. Yet, all of these systems are focused at secondary education (11-16), with no viable alternative for younger children with visual disabilities.

2.2 Initial Learning Environments

Young children are more commonly taught with Initial Learning Environments (ILE), designed to teach programming by offering abstractions of important concepts found in conventional languages. They aim to support a child's understanding of the epistemology of programming languages, introducing new constructs such as data structures and algorithms (D. C. Smith, Cypher, & Schmucker, 1996) in an engaging environment. *Scratch*, alongside *Alice* and *Greenfoot*, is a widely-used example (Fincher, Cooper, Kölling, & Maloney, 2010; Utting, Cooper, Kölling, Maloney, & Resnick, 2010). A recent paper reviewed more than 47 different tools available for children between the ages of 2 to 18 years (Duncan, Bell, & Tanimoto, 2014).

The majority of ILEs use block-syntax (i.e. graphical representations) with drag-and-drop program composition rather than text-syntax that is typed. Controlled comparisons between block-syntax editors and text-editing environments demonstrate the value of visual and tangible interaction cues to scaffold syntax learning (Weintrop & Wilensky, 2015). For example, syntax errors, such as synching brackets or type mismatches, are usually prevented by design, such as not allowing blocks of the wrong type to snap together (Fincher et al., 2010). Programs are *failsoft*; if blocks do click together, they attempt to do something. Although block-based environments are not accessible to visually disabled learners, the thinking behind ILEs can be applied to Torino.

One notable characteristic of ILEs is that they exhibit higher levels of *liveness* (Tanimoto, 1990), the speed at which the program display updates when code is change. More formally, liveness can be defined as a fundamental property of the user experience of programming environments that refers to the temporal and causal relationship between modifications that the programmer makes to the program source code and the changes in the behaviour of the executing program that result from those actions. Enhanced levels of liveness are considered a valuable attribute of educational programming languages (Burg, Kuhn, & Parnin, 2013; Burnard, Florack, Blackwell, Aaron, & Philbin, 2017) .

2.3 Tangible Programming Languages

Tangible programming languages which use physical objects to represent or interact with programming constructs may be a more obvious choice for children with visual disabilities. A number of tangible programming languages have been explored in the research literature: *Algoblocks* provides large blocks that could be connected to make a command sequence (Suzuki & Kato, 1995); *Tern* consists of puzzle-like pieces containing machine readable patterns that are snapped together to drive a robot (Horn & Jacob, 2007); *Flow blocks* demonstrate causality in complex systems through lighting up blocks that have been magnetically connected to show a chain reaction, e.g. a linear versus probabilistic chain reaction (Zuckerman, Grotzer, & Leahy, 2006).

Recently, tangible programming languages have found their way out of research laboratories and are now available commercially. *Osmo Coding* lets children snap together wooden tiles (with some buttons/dials) to manipulate on-screen games (Hu, Zekelman, Horn, & Judd, 2015). *Kibo* is a stand-alone kit that enables children to build a

robot with a variety of sensors, and to program it using a set of wooden blocks that can be affixed together. The blocks are read through a camera in the robot (Sullivan, Elkin, & Bers, 2015). There is also *Cubetto* for the pre-literate child, a robot that can follow a course through putting shapes on a board.[1] All these focus on the sequencing aspect of code.

The most recent explorations in tangibles is around *making* using Internet of Things (IoT) toolkits. The *ConnectUs Toolkit* encourages children to make cubes that help them understand IoT concepts (Lechelt, Rogers, Marquardt, & Shum, 2016). Other recent work has focused on measuring the role of making in the learning process. (Johnson et al., 2016) showing that making plays a substantial role in children's later understanding of the functionality of what they made. The authors potentially attribute this to the ability to create appropriate schemas that prepare them for thinking about how technology works. This work starts to flesh out how assembly may be functioning in tangible languages as well as broaden out notions of computing.

A key similarity of tangible languages to date is the notion of blocks or buildable pieces, emphasizing that tangible languages enable a clear physical mapping of action to program. Indeed, sometimes a very close mapping can be achieved. For example, in the Tern language, a fork in the road denotes a conditional statement where the robot follows one of two paths; whilst others use analogies such as falling dominos and bicycle chains to explain program behaviours. In taking a strong building metaphor, previous designs have limited the form factor to block-like items that are connected together on a surface. We reconsider this assumption as part of the Torino design process.

A key trade-off between tangible systems is whether they use active and passive pieces. Flow blocks are an example of active blocks with embedded electronics that operate as a standalone system. In contrast, Tern has passive pieces, such that the program is constructed, then later executed through the use of a camera scanner. Arguments have been made that active pieces can make the technology responsive in ways that encourages playful and exploratory interaction (Horn, Solovey, & Jacob, 2008). Whereas, a passive approach encourages a design workflow, moving from the initial design to testing and then reflection on, and revision of, the initial program. Passive pieces are more robust and cheaper.

We see a range of solutions to the trade-off of active versus passive pieces in commercial systems. Osmo utilizes an iPad to give the experience of active pieces, while reducing the electronics embedded in the tiles. Kibo embeds their electronics in the robot to give an engaging experience without the need for active pieces or a screen-device. Cubetto puts the electronics in the board to which its pieces attach, avoiding either active pieces, or a screen device. These decisions lead to a different balance between the cost of the pieces (ones with electronics being substantially more expensive), the liveness of the interaction

[1] https://www.primotoys.com/

as a program or piece responds, and the need for a screen. In Torino, we focus on the liveness of the experience, choosing active pieces.

Tangible languages, both in their proposition and evaluation, have highlighted how their use can support human interaction. Across systems, the tangible nature of the languages seems to invite and foster collaborative engagement. The creators of Tern, for example, point to the ability to work on programs away from a computer as a substantial advantage for classroom management and child interaction (Horn & Jacob, 2007). They also highlight how the tangibility of the language draws people into participation in both classroom and museum settings, especially girls (Horn, Solovey, Crouser, & Jacob, 2009). This is in line with a broader literature on tangible technology that suggests that its physical nature invites sharing (Hornecker, Marshall, & Rogers, 2007). This constellation of references suggests that a tangible programming language has the potential to support the kind of collaborative interaction we strived for between visually disabled and sighted children.

2.4 Teaching Programming

In imagining a new tangible language it is also necessary to consider *what* will be taught. The Computing at School initiative in the UK, for example, regards computer science as a discipline; one that *"encompasses foundational principles (such as the theory of computation) and widely applicable ideas and concepts (such as the use of relational models to capture structure in data). It suggests incorporating techniques and methods for solving problems and advancing knowledge (such as abstraction and logical reasoning)"* (Peyton Jones, 2013). This broad view of computing influenced us to address concepts related to writing code, such as iteration, as well as computing-related problem solving, such as dealing with resource constraints.

More specifically, the learning goals of the UK Curricula for primary school children were used to anchor the design of Torino (Department for Education, 2013). To ensure that the design would not be UK specific, learning goals were cross-checked with the Australian curriculum[2], the other major English language curriculum available at the time, and found to be very similar. Learners ages 7 – 11 "design, write and debug programs that accomplish specific goals" and solve problems through decomposition. Learners are expected to do this by using sequence, selection, repetition, variables, and inputs and outputs. There is also a focus on the use of logic to detect and correct errors in algorithms and programs.

There is also a growing literature on *how* children might be taught to program. Along with an increasing understanding of pedagogic strategies for introducing key concepts, some attention has been given to ameliorating misconceptions. This has been discussed in relationship to an inadequate understanding of the 'notional machine', and especially to the 'hidden' processes that are not directly apparent from the program (Sorva, 2013). Consequently, learners of programming may form an incorrect mental model of how the

[2] http://www.australiancurriculum.edu.au/technologies/digital-technologies/curriculum/f-10?layout=1

program works, impeding understanding. This prompted us to consider how we make apparent, and support, an understanding of program execution.

A number of strategies have been developed to support understanding of code and its execution that translate well into the classroom. *Use-modify-create*, suggests that users learn by using and modifying existing programs before creating their own (Lee et al., 2011). Own programs then follow a cycle of testing, analysis, and refinement. Another similar strategy is *read-before-you-write* (Lister et al., 2004). This technique is based on developing the ability to trace through programs as they execute. A third strategy that is gaining attention from teachers of computing is *pair programming* (Sentance & Csizmadia, 2016). Students are asked to tackle difficult programming problems together, in line with a constructivist position on how children learn (Williams, Wiebe, Yang, Ferzli, & Miller, 2002). These pedagogical techniques have influenced the design trade-offs that we make in Torino.

2.5 Teaching STEM to Visually Disabled Students

Not least, we specifically consider teaching practices from Science, Technology, Engineering & Maths (STEM) subjects for students with visual disabilities. A recent literature review covering all disabilities (Moon, Todd, Morton, & Ivey, 2012), distinguishes between two philosophically different approaches: inclusive teaching and assistance (either human or technological). Inclusive teaching uses multiple means of presentation and has been demonstrated as an efficacious method for accommodating students with disabilities. Yet, it can be challenging to implement in STEM subjects as it requires the development of appropriate tools to engage senses other than sight.

The use of assistants is more common. This may include helpers who do the laboratory work while a disabled student watches. Or, the pairing of disabled and non-disabled students and dividing the work by abilities. This report argues that these kinds of assistance run counter to the culture of STEM education, that treats 'hands-on' participation as a key part of the learning. Beyond the challenges of finding assistants with appropriate skill, there remains the issue that these approaches miss the visceral aspects of STEM, and promote 'partial participation'. As result, research is being put into ways that STEM can be made hands-on to disabled students. It is the explicit approach in desiging Torino to focus on inclusive teaching rather than assistance.

A recent article has reviewed tactile and auditory teaching materials to support inclusive primary math education (Leuders, 2016). It suggests that much of the math teaching content is represented by spatial structure, such as a number line. While it is possible to gather spatial structure through touch, mathematical manipulatives often require processing of the exact spatial structure. This differs from the perception of outstanding features needed for object identification. This article proposes that children need to learn strategies of active touch: 1) preliminary scanning – deliberate scan of structure before counting; 2) count organizing – learn to follow given structures like dot lines or circles; and 3) partitioning – develop strategies to keep track of elements already counted by moving them aside or by using one hand to indicate a partition. The article emphasises that spatial structure should clearly emphasise features representing mathematical content. The use of auditory information with touch can be particularly beneficial.

Technology has been developed to facilitate STEM learning for those with visual disabilities. It has, for the most part, been focused on tactile graphics and more recently 3D models. For example, Petit et al. focus on capturing images that are in school books through the design of specialized hardware that creates a refreshable tactile display (Petit, Dufresne, & Levesque, 2008). Baker et al. (2016) focus on the challenge of the label information that surrounds tactile graphics such as charts, which can be become unwieldly in braille. They propose a solution that uses audio labels provided through QR codes. Most recently, with the growing availability of 3D printing, Shi, Zelzer, Feng, & Azenkot (2016) present a toolkit that enables users to add and access audio labels on 3D printed models. These provide a precedent for developing simple tools to make accessible mainstream learning. Less emphasis is put on creating tools that are inclusive for the whole classroom.

2.6 Summary

We've drawn upon a diverse range of literatures to inform the design of Torino. The literature review evidenced that despite the large number of programming languages already available for children, there are none that are age appropriate for children 7 – 11 years old with visual disabilities. Both visual and tangible programming languages require vision to manipulate, whether coloured blocks in the world or on the screen, and vision to observe the program output, whether animations, physical movement, or lights. Below we summarise key insights we drew from the literature that directly influenced the design:

1. Address a broad view of programming for children ages 7 – 11 guided by, but not exclusive to, the UK primary computing curriculum.
2. Make apparent and support an understanding of program execution.
3. Reduce syntax errors through prevention by design.
4. Utilise *liveness,* or immediate update of how changes to the code influence the program, to support understandability and engagement.
5. Utilise the physical mapping provided by the tangible technology to reveal structure.
6. Consider in depth the trade-offs between passive and active technology pieces within the language.
7. Acknowledge the different philosophies to inclusion and focus on inclusive teaching.

3 TORINO YOUNG DESIGN TEAM WORKSHOPS

In addition to the insights we gained from investigating existing research systems and technologies in the market place, the design of Torino has been informed by a series of workshops that involved four children with mixed visual abilities. These were conducted through the lens of inclusive design. In this section, we first describe the inclusive design methodology used, followed by the method, materials, and findings of the workshops.

3.1 Inclusive Design Methodology

Inclusive design is a design methodology that draws on the full range of human diversity. Many of the voices that have defined this methodology are captured in (Clarkson, Coleman, Keates, & Lebbon, 2013), but we have taken the specific perspective of the Microsoft Inclusive Design toolkit and materials (Shum et al., 2016). This approach emphasises three tenets of inclusive design: 1) recognise exclusion; 2) learn from diversity; and 3) design for one, and extend to many.

The first design tenet re-defines disability as something that arises when there is a mismatch between a person's body and the features of the technology, rather than being something inherent to the person. This puts the onus on designers to be mindful of how they might create disability with their designs. The second tenet emphasizes how working with the constraints of a particular disabled person(s) can help generate unexpected designs by making design constraints explicit rather relying on implicit design ideas shaped by the experiences of the design team. The third tenet suggests that while it is important to maintain consistency of design through designing 'for one', designers should actively consider how their designs can be extended to many people.

Motivated by the first tenet, we put substantial design effort in Torino towards the second and third. Specifically, we involved children with visual disabilities in our design process to learn from them, first-hand, how they were perceiving and engaging with the world in a tactile manner; and how we could create an experience that might feel exciting to all children. We were mindful however, to not create anything that would be exclusively used by blind children, e.g. using braille labels. Imagining how we might extend our design to many has helped us be inclusive of the very diverse visual abilities that children labelled as disabled have. In the UK, of the 25.000 visually disabled children, less than 1.000 are braillists who learn exclusively through tactile means[3]. Surprisingly, the vast majority of visually disabled children, including those registered blind, and some exclusively tactile learners, can perceive color, and rely heavily on it to orient themselves in a world that they otherwise cannot see. As a result, we wanted a design inspired by tactile learning, but available to those who learn visually.

Thinking of the broader population also enabled us to keep a philosophy of inclusion in mainstream classrooms. While the majority of children with a visual disability are in mainstream classrooms, the challenges of inclusion are substantial and rarely achieved well across the board (Metatla, 2017). As a result, children with visual disabilities are often segregated to use tools designed specifically for them. We aimed to create a technology that did not create such a barrier to inclusion. Indeed, we specifically adopted the perspective of inclusive teaching rather than assistance.

3.2 Design Workshop Method

We formed a Young Design Team, modelled after the cooperative inquiry approach (Yip et al., 2013), to gain the expertise of children with visual disabilities on tactility. Four

[3] http://www.rnib.org.uk/professionals/knowledge-and-research-hub/key-information-and-statistics

children with visual disabilities (three girls), ranging in age from 8 – 12 with a wide variety of visual abilities attended four out-of-school workshops over a period of three months. See Figure 6 for details. The children were recruited from attendance at a computing outreach workshop for children with visual disabilities run in conjunction with the local charity for sight loss. They were selected to ensure we captured the perspectives of children with a range of visual abilities and ages.

Each workshop was designed to help our Young Design Team articulate, or demonstrate how they interacted with the world through tactile objects. Each workshop further refined this question as we moved from a general understanding of how the Team engaged with objects to a specific focus on how we might design 'blocks' that connect together to represent code. The research received ethical approval by our local research board. Informed consent for the children to participate in the research, and for the sessions to be audio and video recorded, was obtained by the parents prior to the workshops. We also asked the children if they were happy to partake in the project and explained to them why we were doing these workshops.

The workshops were attended by at least four researchers as well as video recorded. Some of the activities and materials created were also photographed. After each session, the researchers took extensive notes to record their main impressions and observations. One researcher took the lead in prompting team design discussions through the use of salient video clips. For the scope of this paper, we will focus on the learnings from these design sessions which directly impacted the design. For each workshop, we describe the focus and materials used, followed by findings and numbered key learnings.

3.3 Workshop 1

3.3.1 Materials

Our first workshop focused on contrasting children's interactions with a non-tactile approach to coding, such as a screen reader, with a tactile approach using blocks. We augmented Sonic Pi[4], a live coding language for digital music, with a screen reader and enlarged text. We also attempted to connect Sonic Pi to the commercially available Cubelets[5] modular robotics system. As we had difficulty with the robustness of the Bluetooth connection, the workshop was conducting in two halves instead: coding with Sonic Pi and playing with the Cubelets.

In the first part of the workshop, each child sat with a researcher at a laptop computer around a large table. The researcher oriented the child to a simple program in Sonic Pi containing play statements and a loop statement which played a catchy tune. First, the child listened to the output of the program, and then was taught how to change the parameters in the statement to get a different tune, using the use-modify-create teaching approach discussed in the literature review. The child could experiment as much

[4] http://sonic-pi.net/

[5] https://www.modrobotics.com/cubelets/

as they liked. The researchers were prepared to introduce new types of statements if the child showed interest in developing their program.

In the second part of the workshop, the researcher asked the child to build something fun, introducing the different aspects of the Cubelets (such as the motor or light sensor). The researchers answered questions of the child about how to achieve a specific behaviour, but otherwise did not intervene.

3.3.2 Findings

The children struggled to interpret the code presented to them through enlarged text or audio (non-tactile). Those children who used enlarged text could see only the line of code they were working on. They had to scroll through the code to see all of the lines, losing a sense of the meaningful groupings, such as code in a loop, which is emphasised visually through indentation. This was only exacerbated for the children who used a screen reader, which read out the lines of code in sequence, giving no impression of their relationship. This is best illustrated by the following instance:

> One child using the screen reader version of Sonic Pi was trying to identify what a piece of code did by changing any numeric value that he encountered, and then playing the program. He did not realise that the code was in a loop and was unable to interpret how his changes affected the code. Visibly agitated, he started frantically moving through the code to try and get an overview, but he was unable to remember it from beginning-to-end. Eventually, his mother brailled out what she could see on the screen. Only after reading the braille did this child understand the overall shape of the program and which lines of code were in the loop. He continued to use the braille version to maintain his overview of the program while he changed the parameter values.

In contrast to the constant nudging of the researchers to encourage the children to try different parameters in their Sonic Pi programs, the children engaged immediately with the Cubelets. All of the children took the same approach of grabbing a few blocks, snapping them together, observing what happens, and iterating. While the children were not particularly interested in form, neither commenting or feeling it, they were interested in eliciting different behaviours, such as getting a creation to move, or responding to their hand motion. Soon children started to form groups of two, playing together, using their creations as a central focus for conversation and ideation.

We derived the following learnings from this workshop:

1. *To support coding in this age group, children need: 1) a tactile representation of code that helps them understand how code relates; 2) a consistent overview of the code at all times to decrease memory demands.* Typical accessibility approaches, such as enlargement or screen reader, did not work for this age group as they assume conceptual understanding and have substantial memory demands.
2. The *immediacy of the response,* similar to the notion of liveness discussed in the literature review, of Cubelets invited experimentation and manipulation that was central to the engagement with them.

3.4 Workshop 2

3.4.1 Materials

The first workshop helped identify the need for a tactile way to express code with objects that invited manipulation. In the second workshop, we explored in more depth experiences of tactility around objects. We asked the children to bring in two objects: one that they enjoyed touching, and one that they liked to connect things to, or things with. This distinction was made as we predicted that the eventual design would require both tactile elements that can be touched and are distinguishable, and that can also be connected. The first part of the workshop involved the children introducing and sharing these objects with the rest of the team, passing them around while explaining their liking. The facilitating researcher encouraged the other children to play with them and to also share their thoughts on each object. In the second half of the workshop, the children were given different types of blocks that could be connected in various ways to build, such as Magnuts. All block materials connected together with magnets; something that the children were familiar and needed no instruction with. Straight away, they started to play and make.

3.4.2 Findings

The children brought four objects that they liked to feel: a small rubber stretchy monkey that stuck to surfaces; a special spoon handle, a "water" ball in which you could hear water when shaken; and a squashed coin. The stretchy monkey was particularly liked by all the children. They repeatedly pulled its limbs in different directions, experiencing the difference between pulling both arms as opposed to pulling an arm and a leg. They tried fast pulls and slow pulls feeling the visceral continuous feedback of the monkey responding to their actions. The special spoon handle had a particularly comfortable grip and was familiar to the child who brought it; the water ball had an engaging sound for all, inviting experimenting with multiple ways of shaking; while the coin had an intricate pattern with a complexity that was interesting to explore for the tactile-learners but not those who learned visually.

As for the connector objects, the children referred to the stretchy monkey, Lego bricks, and poppers (a stick with a suction cup on one end). All the connector objects were liked because of the auditory and tactile feedback they provided. The stretchy monkey sticks to surfaces when water is applied, adding another response that the children could explore with different amounts of water or by applying force. The child who had brought that object liked the continuous sensation of it peeling off the table. The Lego bricks were appreciated because they gave a nice click sound when snapped together and did not fall apart easily if knocked. The poppers seemed to make a particularly satisfying noise. The owner said, "I use them to play because it makes a funny sound." Feedback, both tactile and auditory, were central to the preferences expressed here by the children.

In the second activity, the children enjoyed building and quickly set themselves challenges as they did in the first workshop: Can I use all of the pieces? Can I build a vending machine? Figure 2 shows a number of built pieces, including a geometric design, two houses, and a vending machine. The children had no difficulty disambiguating similar pieces, however, in contrast to the objects brought in by the children, these square

building pieces did not seem to entice the children to explore their surfaces with their hands, or to experiment with connecting them. Indeed, the magnetic connections proved particularly frustrating as the blind children frequently knock over what they had built accidently in an attempt to experience the shape they created, or identify what was missing. Further discussion of magnets illustrated a range of challenges for the children, including difficulty moving a creation to show to another child.

In summary, we learned that:

1. Tactile elements need to *connect securely* in a way that provides strong auditory and tactile feedback. Magnetic connectors, so commonly used for building block like structures, are not appropriate.
2. While blocks were distinguishable, they did not invite exploratory play in the same way as the children's objects or the Cubelets.

[Insert Figure 2 Here]

3.5 Workshop 3

3.5.1 Materials

To identify secure, yet tactile connectors that did not rely on magnets, and easy-to-distinguish shapes that would invite children to pick them up and manipulate them, the third workshop investigated the concept of a bead metaphor: a set of commands in a program represented as interestingly shaped beads along a thread. Following this metaphor, we explored how the children would connect objects with flexible wires, and whether bead like objects would encourage manipulation. The first activity invited children to create a program through connecting 3D printed blocks with wires using 2.5mm audio jacks. The children could change the value of the blocks by placing a piece mounted on a 3.5m audio jack into pre-configured holes at the top of each block. The sounds were then Wizard of Oz-ed by the facilitating researcher typing commands into Sonic Pi and playing the program when requested.

[Insert Figure 3 Here]

The second activity focused on a series of manipulative beads. These were a series of round 3D printed, bead-like shapes. Each piece had a means of physical interaction to invite handling and manipulation. These included: blowing, pushing, twisting, and squeeze clicking as shown in Figure 3. The facilitating researcher introduced the children to each one of these shapes and mechanisms, and then asked them to choose the one they liked most, and least. We were interested to explore if using such physical mechanisms to change parameter values on the beads would invite their handling and iterative play. Pairs of children alternated between the first and second activities, working in separate spaces.

3.5.2 Findings

The children found it easy to connect the blocks with wires. Those with more vision brought their heads close to the blocks to locate the connectors, while others with less vision used one hand and a finger to locate the jack, using it as a reference for plugging

in the wires. They expressed their liking of the feel and click interactions when inserting an audio jack pin into a plug, often smiling and mimicking the sound. One child commented that it was like Lego, "clicks and stays." We used 2.5mm jacks for the wires and 3.5mm jacks for the top parameter plugs. This distinction was confusing as children could not feel the difference and got frustrated when they tried to connect the wrong pin.

The wires also played a role in communicating program flow, enabling the children with the least vision to easily follow the chain of blocks from one side to the other, similar to uses of table edges or a braille line. Interestingly, both groups attempted to connect the wire of one block into the other socket of the same block. In both cases, they surmised that this would lead to repetition (or more formally, iteration). While both groups wondered how they should connect this contraption to their program, they had both unwittingly stumbled on the idea of a loop, before we had introduced the concept. Although the length of the wires made the programs a bit unwieldly, both the concept of a bead metaphor and the use of an audio jack pin as connector for creating programs were well received and understood.

In the second activity, the children immediately started to manipulate the beads. The first child started by blowing vigorously into one to get the inner propeller to turn, producing a whistling sound that she liked. The second child was attracted to the accordion mechanism pushing and pulling in a similar way to how the children used the squeezy monkey in the first workshop. While they explored each of the mechanism beads offered, they returned to these as their favorites. The second group of children with less vision, spent much time tapping the beads on the table or bringing them up to the ear to listen to their sounds, such as those made by the squeeze click. They seemed more interested in the auditory response then the tactile one. In both groups, we observed how the ability to manipulate the pieces, and getting an immediate response from this, encouraged repeated exploration, which is reminiscent of the Cubelets in workshop 1.

The intensive interaction with these manipulative beads contrasted significantly to the way that the children handled the square blocks. The square blocks were lined up in a row, but rarely handled. In contrast, the roundly-shaped mechanism pieces were constantly held and rotated in their hands. The children spent time feeling them in the palm of their hand and with their fingertips. They felt the sensation of rolling the pieces on the table or brought them up to the ears for listening. While the square blocks had the advantage that they did not roll away out of reach of a blind child who could not find them, they were less interesting in terms of their tactile qualities.

Learnings included:

1. Audio jack pins and plugs make excellent connectors, but need refinement to be practical.
2. Rounded shapes invite holding, but need to be designed in a way that would not roll out of reach of a child who may not be able to relocate them.
3. The manipulative beads encourage iterative play and exploration, but need to be robust to substantial handling, and banging against other surfaces.

3.6 Workshop 4

3.6.1 Materials

In our final workshop, we presented the Young Design Team with a full set of beads. Each bead had a rounded, but distinct shape with a wire at one end, and a dial or push button mechanism to adjust the parameter. The design was a direct outcome of the previous workshop, however, the original mechanisms were adjusted for robustness. This set of beads is very similar to the final shapes of the Torino design, described in depth in Section 4. Here, we wanted to investigate if the design enabled the children to effectively trace their programs, and identify the different bead types. In pairs, the children were invited to create their own programs by connecting the beads. These programs were then Wizard of Oz-ed by the facilitating researcher.

3.6.2 Findings

We observed a number of ways in which the children tactilely followed their programs. The children with the least vision frequently kept one hand on the starting bead or hub and followed the wire lines outwards to find the beads. Their hands did not "stop" to feel each bead, but smoothly shifted from wire to bead and then again to wire, recognizing the shape as they moved their hands. The beads were distinct enough such that the children did not have any difficulty naming them as they read out the program instructions: play, play, rest, etc. When asked to add a certain component in between two beads, they could locate the correct bead, and insert it. These children could also identify the bead shapes when following the hand movements of their programming partner, without actually touching the beads.

Children with more vision were also able to easily distinguish the beads through touch. When asked to describe the differences, they spoke about shape (e.g. 'the long one') or referred to the mechanism on the bead (e.g. 'this one has buttons'). They had more difficulty distinguishing the beads on the table as their tactile skills were not as developed as those of the children with less vision, and their visual skills could not be utilised as the white 3D printed beads blended into the table without a dark surface to contrast, or colour to support visual distinction.

We learned:

1. To support the full spectrum of visual abilities, *we needed to include visual elements that mapped to tactile elements, without leading to discrepancies of information provision* (e.g. as would happen if visual symbols were used).

4 TORINO DESIGN & IMPLEMENTATION

Drawing on the design workshops with the Torino Young Design team, we designed and built a working version of Torino. To create code, the children connect physical instruction beads with wires and tune their parameters using on-bead mechanisms to create music or stories that are rendered to audio using the Sonic Pi language as shown in Figure 1. A key design focus for Torino was a "bead" metaphor. Like bracelets or rosary beads, we wanted to design something that invited a tactile experience, a picking

up and exploring with the hands that creates a 3-dimensional experience of form. This stands in contrast to the blocks or bricks metaphor used in other tangible programming languages that are most often experienced flat on a table.

[INSERT Figure 4 Here]

4.1 Bead Design

Torino is composed of a central hub and a suite of beads (play, pause, loop, and variable) shown in Figure 4. The hub is the central component of the program to which multiple bead 'threads' can be attached. Threads are a series of connected beads which, following the conventional metaphorical terminology of computing, represent a concurrent execution model in terms of threads in the program. The Sonic Pi execution model that underlies Torino is an inherently concurrent language, in which simultaneous musical parts or voices are expressed as concurrent threads. The hub houses a Raspberry Pi which powers all of the beads and carries out the computation to identify the topology of the beads and translate the physical program into Sonic Pi code. It also houses the speaker, which plays the resulting audio program. The start button allows the child to control initiation of the program.

Play, pause, and loop act as instruction beads, adding a single instruction, or line of code, to the program. They play a sound, create a pause, or enable iteration of attached beads, respectively. The design of the bead shells is intended to enable rapid tactile distinction, while giving the impression that the beads are part of a single family. For example, they all have a rounded shape, but the pause is almost spherical, whereas the play is oblong, and the loop has flatter sides. Color is also used to distinguish beads for those with some color perception or sight, which is quite common among children with visual disabilities.

[Insert Figure 5 Here]

Interaction mechanisms on the play and pause beads can be used to increment or decrement a parameter value. A dial enables choice of sound, and a two-way button, duration. This provides both an inviting physical interaction and an immediacy of hearing change. Variables (in the loosest sense), plug into loops to set the number of iterations, otherwise an indefinite loop is assumed. Variables could also be used with play and pause beads to set a particular parameter value. To avoid syntax errors, as commonly done within the 2D jigsaw metaphor of many on-screen ILEs, variables' connections have a square port, in contrast to program flow connections which are round. They also have been left uncolored. Further, we provide caps for the variable plugs as a 'training wheels' experience for first-time use to avoid children getting confused about what plugs fulfill which purpose.

The beads are connected together with a 5cm long wire which have a 2.5 mm audio headphone jack on the end. Each bead has a wire that extends from it, inviting it to be connected onto an existing chain of beads. Connections give satisfying auditory and tactile feedback. Wires have blue color wraps which match blue circular ports. A ridged wrap is used for the loop to distinguish which wire 'closes the loop' and which 'connects to the program'. The beads each have their own embedded electronics to support

liveness embodied as real-time response to bead manipulation. The tangible system is entirely standalone, with no need for a separate host computer.

Figure 5 illustrates Torino code alongside pseudocode communicating what the program does. The threads execute concurrently.

4.2 Software and Hardware Implementation

Each bead contains a custom-designed circuit board, containing a microcontroller and a number of connectors that provide power and communication to connected beads, allowing them to form a network. When a bead is connected, it becomes powered and transmits data to its neighbours, including details such as the type of bead and its current state. Messages are propagated through the network, keeping track of the route of the message until it reaches a central bead, which is connected to a Raspberry Pi device. From these messages it is possible to construct a graph of the network, where a node is a bead and the edges are the connections between them.

A Python script running on the Raspberry Pi translates this graph into Sonic Pi code. Sonic Pi plays an important role in managing the auditory experience, such as synced threads, and provided inspiration for Torino. However, Torino is not an exact copy of Sonic Pi. For example, we automatically included the *Sleep* command into each bead, to ensure that sounds are sequential. While the Sleep command gives important flexibility to the professional user of Sonic Pi, its necessity is not obvious to the student nor is it manageable in a physical programming language which would otherwise become very unwieldly.

4.3 Design Decisions and Trade-Offs

Programming languages always have trade-offs that optimise interaction for a certain set of activities. To aid our systematic thinking about these trade-offs, we considered them using the Tangible Correlates (TC) to Cognitive Dimensions (CD) framework (Edge & Blackwell, 2006). This is a heuristic evaluation tool for the analysis and design of physical notations (e.g. tangible user interfaces). The TCs (and selected CDs), their definitions, and their relevance to design trade-offs are presented in Appendix 1. Below we discuss three key design choices and the trade-offs made in achieving these: bead metaphor, persistent program overview, and liveness.

4.3.1 Bead Metaphor

Torino is built on a bead metaphor in which beads are strung together to create a program. This stands in contrast to arranging a set of blocks or tactile elements on a table as commonly done in the design of tangible languages. A key advantage is that a string of beads is robust to accidental bumps that may change the semantic meaning ($Shakiness_{<TC>}$) when being read by a tactile learner. It also enables the connected beads to be picked up and experienced in three dimensions, possibly deforming the physical arrangement, while preserving connectivity of the information structure ($Rootedness_{<TC>}$). This is achieved through the use of audio jack and wire connections.

Wired connections have a further advantage as they allow for rapid tactile exploration of the program as they are easy to follow, by tracing them with one's hands. There is a

trade-off between shorter connections being faster to follow with fewer tangles, and longer connections allowing clearer arrangement of beads, but potentially leading to overly large programs (Unwieldy Operations$_{<TC>}$). To keep the number of operations manageable, we chose to connect one end of the wire to the bead. In doing so, we defined an intermediate length, meaning a loss of opportunity to use wire length as a source of information that expressed program semantics, such as duration (Structural Correspondence$_{<TC>}$).

A number of our design decisions addressed the trade-offs of using wires. The hub is used to create a beginning, as a string of beads does not have an obvious start point. The hub also helped the children locate where the beads might be on the table given that these could be moved (Rootedness$_{<TC>}$). While there is an opportunity to move the program around, system state might be modified if the rotary and button controls are unintentionally knocked (Shakiness$_{<TC>}$). Not least, tangible languages take up more space than virtual ones, often mitigated through flat pieces that can be stacked for storage and laid flat on a table. Our graspable 3D beads take up more storage space (Bulkiness$_{<TC>}$).

4.3.2 Persistent Program Overview

The most important design decision was to ensure that a child could maintain a tactile overview of their program at all times (Permanence$_{<TC>}$). To create a clear correspondence between the tactile experience of the code and its meaning (Structural Correspondence$_{<TC>}$), each bead represents one line of code. This was necessary to account for the age range of our target user group, considering the perplexing and negative experience of having access to only one line of code that they had previously with alternatives such as screen readers and enlarged text, which removed the context of the code. However, this design choice limits the complexity that can be achieved with programs, for example, we did not introduce functions, although this is part of the curriculum.

Despite an emphasis on a persistent overview, we chose not to make the parameters inspectable on the beads as it could lead to accidental change if the program in knocked (Shakiness$_{<TC>}$). We considered an additional device, *The Inspector*, as a way to inspect parameters, but thought this unnecessary in the first stage of evaluation. Without such a device it would be challenging to set values over the available eight octaves, incrementing between values 21 and 108. Instead, we provided eight sounds and four durations to the children that would cycle continuously. Duration is made more palpable by providing a marker at the beginning and ending of a duration, enabling pauses to be more easily distinguishable (Hidden Augmentations$_{<TC>}$).

4.3.3 Liveness

The Torino design was intended to be responsive to encourage manipulating the beads in a way that invited iterative exploratory play. To achieve this, the design included electronics in every bead. This had substantial implications for size, complexity and cost of design (Bulkiness$_{<TC>}$). A minimal set of beads is needed to reduce cost of building and maintaining such a system. We decided on 14 beads: seven plays, two pauses, two loops, and two variables. This number of beads was sufficient to create musically interesting, multi-thread programs. However, to reduce complexity, we omitted the use of an instrument bead which would allow setting the sample or synth used (i.e. sets of

sounds or instrument choice). Instead, this choice was implemented and set within a thread. Further, in favour of simplicity of progam structure and the size, availability and functional robustness of the beads deliberately did not add electronics that would vibrate into a block to indicate that an instruction was being executed by the program.

5 EXPLORATORY DESIGN EVALUATION

We carried out an initial evaluative study with a single prototype to investigate how the overall design worked in practice with children with a range of visual abilities. The motivation for a tactile approach (as opposed to common assistive technologies such as screen readers) emerged from challenges faced by children in the first workshop to get an overview of their code such that they could understand, or 'read' it. This is often referred to as *tracing* in the computer science education literature and considered to be a core skill predictive of the ability to code when teaching undergraduates, e.g. (Lister et al., 2004; Lopez, Whalley, Robbins, & Lister, 2008).

As a first step in understanding the effectiveness of the developed "bead metaphor" and related design decisions, we address how Torino supports tracing, or the read-before-you-write educational strategy. We specifically ask the following three research questions:

RQ1: Are there any usability issues that stop children from creating code?

RQ2: How do children with mixed visual abilities trace their programs using Torino?

RQ3: How do children with mixed visual abilities create programs using Torino?

At this early stage, we did not assess any of these quantitatively. Instead we took an in-depth qualitative approach to support comparison across a wide variety of potential users to help identify areas for improvement in the design. We use the discussion to reflect upon the design decisions made. Although we touch briefly on the role of collaboration in this process, a detailed analysis of how Torino supports collaboration can be found in (Thieme, Morrison, Villar, Grayson, & Lindley, 2017).

5.1 Method

5.1.1 Procedure

Over a two month period, three consecutive sessions were carried out for each participant. Consecutive sessions were chosen to see how usage evolved over time without asking teachers to commit large amounts of teaching time to a new prototype. The first two sessions were attended by the pairs of learners. This mirrors our expected usage of Torino in educational settings and addresses our aim of inclusive teaching which does not segregate disabled learners. In the third session, learners were asked to attend alone with a parent or teacher. This final session gave an appropriate motivational structure for young learners to think aloud while teaching Torino to their parent or teacher, illustrating their understanding of Torino. Think aloud is considered a key approach in assessing conceptual understanding in computer education (Lye & Koh, 2014).

The sessions were attended by two researchers. One researcher (CM), who is experienced as a teacher and in working with children with visual disabilities, led the programing activities. The second researcher (AT) oversaw the audio-visual equipment and addressed any prototype failures such that the flow of the session was not interrupted. Parents or teaching staff were present to oversee and assist in the activities. This enabled better tailoring of the sessions to the learners, whose vision levels and academic capabilities were unknown to the researchers. With exception of our meetings with two pairs at their schools, all sessions were held in a meeting room in our research lab. Scheduling issues at the school meant that only two sessions were attended by these children.

Sessions were designed to gradually introduce programming concepts: sequences, threads, iteration, and variables; as well as broader programming concepts: debugging, breaking problems down, and dealing with resource constraints. Sessions were comprised of exercises, such as translating code written on paper (in form of large prints or Braille) into Torino, and challenges, in which the children had to create a program to match a particular audio recording. Children were also given time to create their own programs. In early sessions, this was a matter of changing the parameters for a given set of instructional beads following the use-modify-create method. In later sessions, children were invited to record their own sounds and create programs with those. The general lesson plans for the three sessions can be found in Appendix 1. These were adapted depending on age and speed of learning for each pair by researcher CM.

5.1.2 Participants

Ten learners age 7 – 12 (5 male) were opportunistically recruited to participate in this study. Four of the children (Cat & Grace2455, Penelope & Reuben) had taken part in the research workshops described above. This meant they had already some experience with the shapes of the play and pause instructions. Yet, as this was the first time that the technology was implemented, they still – like all other children – had to learn and make sense of how to assemble and manipulate these beads into computer programs. We further collaborated with a QTVI (Qualified Teacher of the Visually Impairment), who organized the participation of two blind children at their respective schools. Each of these children chose a sighted peer to attend the sessions with them. A pair of sighted children was recruited through personal contacts.

Torino was designed for use across the sight spectrum so that it was inclusive of, rather than exclusive to, a child with visual disabilities. As such, our study involved varying degrees of visual ability. Five participants were users of a white cane. Two of these, *Reuben* and *Penelope*, were 'tactile learners', meaning that they were very good at reading Braille; *Grace2455*, who was progressively losing her sight still learnt visually, and *Fin* and *David*, two of our youngest participants, were only beginning to learn to read braille. *Cat* was partially-sighted and learned visually as did all of the sighted children. The details of the learners of captured in the table in Figure 6.

We did not try to control for the visual abilities in the pairs that the children chose. We felt strongly that Torino should work in all possible combinations of vision if it was to be an inclusive teaching technique, rather than an assistive one (see literature review). Our

study included a strong variety of pairs: one pair of visually disabled visual learners (partial-sight, registered blind); one pair of blind learners; one pair of sighted learners; and two pairs of mixed blind-sighted learners. Such variation is representative of the education of children with visual disabilities and appropriate for edge-case understanding in qualitative analysis.

To support understanding in the analysis the use of tactile or visual skills, we have marked four categories of visual ability: 1) blind (B) – tactile learners; 2) registered blind (RB) – predominantly visual learners with some tactile skills; 3) partially-sighted (PS) – visual learners with adaptive techniques; 4) sighted (S) – visual learners. Only two of the sighted children, *Hairy* and *HTBP*, had some prior experiences of programing, using Scratch in school.

Appropriate ethical guidance was sought from our internal ethics board and health & safety regulations were met. All parents agreed to the use of photographs and video for research purposes. The study was also explained to the learners. As part of this explanation, we described the writing of research reports. To engage the children in understanding this process, we asked them to provide their own pseudonym names, which are used throughout the analysis.

[Figure 6 Here]

5.1.3 Data Capture & Analysis

All sessions were video and audio recorded and transferred to secure data storage (17 hours in total). We use qualitative methods to understand usage and its variations, drawing methodologically on discussions presented in Fieldwork for Design (Randall, Harper, & Rouncefield, 2007). Episodes of poor usability were coded during review of the video data to answer RQ1. These video snippets were then shared with the wider team for discussion.

To address RQ2, we identified episodes of tracing for each pair. Tracing has been noted in the literature as a key skill in developing programming skills. The conclusions of a large multi-national study suggest that "an early emphasis on program comprehension and tracing, with the aim of automating basic skills, might then free the minds of students to concentrate on problem-solving" (Lister et al., 2004). Other work more explicitly makes the link between the ability to trace and the ability to code (Lopez et al., 2008). We defined tracing to be any physical, visual, or verbal way of describing the execution of the program. We considered whether there were any changes in how tracing was executed over time for each pair. We then looked at how tracing differed between pairs.

For both RQ2 and RQ3, we coded the data based on observable engagement with computing concepts. A large number of frameworks have been developed to capture what is learned in computing; many of the early ones are reviewed in (Robins, Rountree, & Rountree, 2003). We choose to use (Lye & Koh, 2014) as a more recent piece of work directly oriented towards school children. It presents a framework for computational thinking that we adapted to make appropriate to the activities covered in the lessons. Specifically, we coded for: 1) computational concepts – sequences, loops, threads; and

2) Computational Practices -- being iterative and incremental, testing and debugging. These were then used to address either RQ2 or RQ3 depending on whether activities of tracing (reading) or creating (writing) were involved.

The findings section presents descriptions of selected vignettes of observed interactions that exemplify the phenomena coded as per the three research questions. The cases were selected to reflect both common observations of each pair as well as key differences between them. We have found that computational concepts and practices are often intertwined in the data and therefore do not use these as strict categories of presentation.

5.2 Findings

5.2.1 Usability

All children were able to distinguish between the beads as well as to connect them together to make a program by the end of the first session. For most, this was mastered in a matter of minutes. However, we also identified a few usability issues that we will describe first to provide an appropriate frame for the more detailed descriptions of use in the following two sections.

The most substantial interaction issue related to the hub. While distributing connection ports to the four corners of the hub, made them easy to locate tactilely and to distinguish threads by pushing them in various directions; the children struggled to plug beads into the ports as these were angled downwards, making it particularly difficult for those who needed to feel the connection rather than look at it.

Further, there was quite a bit of confusion for the blind children about which end of connected beads to unplug, as both sides – the plug side and the wire side – felt the same when beads were connected to each other. Some tactile indication, such as a ridge, would enable this distinction and protect the beads from being pulled in directions unintended.

To make the electrical connection between beads robust presented another challenge. The children, particularly the blind ones, often plugged beads in more slowly, or did so at more of an angle, compared to any technology testing done by the researchers. This led to beads not providing enough of a power surge to be recognised. Moreover, while the sounds for plugging and unplugging beads were helpful in isolation (e.g. one bead being plugged in), the children frequently unplugged and replugged entire threads in another port. The sound feedback, containing all of the connection sounds as well as the sounds of the beads, was cacophonous and confusing. A more nuanced approach to feedback of system state is needed to distinguish more clearly between different kinds of interactions.

5.2.2 Tracing Programs

5.2.2.1 Tracing Sequences

We consistently encouraged children to physically trace their programs as they played in order to develop an understanding of how their program was executing. All of the

children could correctly trace a sequence by the end of their sessions, but there was substantial diversity in how this was done. Below are three short vignettes, also captured in Video Figure 2 that illustrates the variation that we saw across visual abilities.

> *Reuben(B)* and *Penelope(B)* are about to trace a sequence with 8 beads that they have created together at the end of their first lesson. Reuben has his right hand on the hub to press play and left hand on the first play bead. Penelope has her right hand resting above Reuben's hand on the first bead and her left hand of the second bead. When the program starts Penelope immediately grazes her right hand along the wire to the next beads, Reuben catching up. Penelope then lands with hands curled around two beads before her hand darts to the next rest. Reuben then catches her hand and picks up the rest with both hands, stating "and that's the rest". They continue to the end, Penelope slightly behind the program as she needs to stand to reach it. She gets confused associating the rest with the last note and then wondering what the last bead should do.
>
> *Grace2455(RB)* explains a program to her mother in her last session. Grace2455 hits the play button and then jumps her hand to curl around each bead as it plays. She is slightly hesitant at the rests to make sure she has moved at the right time. She switches hands from left to right to finish the program before looking at her mother. Continuing to rest her right hand on the last bead. She repeats the tracing, singing the program as she goes. Repeating a bit in the middle that she didn't get quite right.
>
> *Ginny(S)* and *Fin(B)* are tracing a program at the end of their first session. Ginny starts the program off and Fin brings his hand to lay on the first bead open and flat. He continues to land on each bead as they quickly play while Ginny points at them from whatever angle she can around Fin. Towards the end, Fin verbalises the instructions, "Oh no. Pause." When the program finishes, Ginny looks to the teacher and says 'yeah', while Fin remains focused on the last bead until the sound stops, nodding his head.

We see that visual ability plays a substantial role in how the tracing is physically accomplished. Those with less sight relying more heavily on the wired connections, while those with sight are able to point from afar. We can also surmise that those with intermediate levels of vision (e.g. Grace2455 and Fin) were relying heavily on the colour in conjunction with the tactile experience to follow their program as their hands jumped between beads. Interestingly, verbalisation happened alongside the visual and tactile experience in all three of these accounts regardless of level of vision.

Tracing was not immediately achieved by any child. We consistently saw that when first asked to trace, children would randomly run their hands over the beads as the program played without a recognisable relationship between the bead and the program. Unlike the behaviours described above, the hands did not pause on each bead, but ran continuously, usually finishing before or after the program did. As demonstrated to them, all participants, with the exception of one sighted child, physically touched the beads in their initial traces. The time it took to be able to trace a program varied substantially between children, some mastering it after a few run throughs, others taking all of the lessons.

There are several coded episodes in the data that are revealing about the learning process while tracing. *David(B)*, our youngest participant, found tracing extremely difficult in the first session. He often started tracing from the end of the program and had to be continuously reminded that 'all programs start at the hub'. He would often jump between non-adjacent beads, not seemingly making the connection that each bead played in sequence despite his ability to follow the string a beads when the program was

not playing. To address this challenge, the facilitator laid the beads out in a line to be read left to right similar to the way in which this child would read braille. This improved the sequential nature of the tracing when each bead was executing. Clarity of execution came when *David,* a very musical child, created a Twinkle, Twinkle program. He was able to apply his understanding of sequence in music to the physical beads on the table.

Cat(PS), who quickly learned to trace sequences, found tracing more difficult in her third session when the programs became more complicated. In this session, she was carrying out the challenge of creating a multi-threaded program that played Jingle Bells. On one thread was a bell and on the other thread the words to the song. Each play bead played a sample from the song. Some samples were single words, e.g. 'jingle' and others were multiple words, e.g. 'all the way'. These could be heard by rotating through the dial on the play bead. Cat struggled to move beyond a single sound per bead. Indeed, to achieve tracing in this task required understanding the beads as containers, and not just direct representations of sound. It is an increased level of abstraction to attain as children's thinking matures.

Penelope(B), our oldest and most mathematically advanced child, quickly mastered tracing, but had given it up by the third session. In her third session, Penelope was working on a debugging task. She had created a program herself and then the facilitator had created two bugs that she was asked to find. While creating the program, Penelope would trace the program when prompted. However, when finding bugs, she did not follow the prompts to trace. Yet, she was able to verbally identify where the bug was and quickly find the bead: – 'it's the fourth one' her hands reaching out and finding the fourth bead without error or needing to start from the beginning of the program. Her behaviour suggests that she had both a mental and physical map of the program and had related these to program execution. This indicates the extent to which tracing can be mastered and divorced from the physical.

5.2.2.2 Tracing Loops and Threads

While tracing loops and threads is similar to tracing sequences, they also present their own challenges. Six of the ten children were introduced to loops and threads. We consistently observed across this cohort that large loops of more than three beads were difficult to trace. *Hairy(S)* and *HTBP(S)* for example, created a loop from a sequence in their first lesson and Hairy chose to follow it while HTBP pressed start. Hairy was uncertain of which was the first bead in the loop, saying 'is it this way?' as he chose a direction. Whether right or wrong, he quickly lost track of the different drum beats happening and started running his hand over the loop waiting for it to end.

This challenge of tracing demonstrated in this vignette became an opportunity for learning in sessions with multiple children. The facilitator focused children's attention on how they could work out the behaviour of a loop. One child proposed the use of markers, ie. a high pitch to start the loop. Other children decided to break down the problem, building up the loop one bead at a time in order to understand its behaviour. The direction of the loop, while explicitly taught, seemed particularly challenging for the blind children, who wanted to follow the wire coming out of the end of the loop bead in direction of flow of the program, loop around to the loop bead and then exit the loop

continuing their hand in the same direction of travel. This contrasted with the design which required changing the direction of their hand on meeting the loop bead in order to start the loop on the first port reached.

Threading posed different challenges for the children. *David(B)* and his father tackled threads in their second lesson. David created a melody line on one thread and a drum line on the other. He began by following the melody line which he mastered, despite the challenge of following sequences in his first lesson. However, he found it very difficult to follow two sequences at once as this requires either being able to hear two independent lines, possible for some blind people but notoriously difficult, or understanding how the threads interact, as one might use the left and right hand on a piano score. Indeed, this difficulty was seen across all of the children.

To build up the ability to trace threads, the facilitator suggested to David that he ask his father to follow the second thread. While successful in tracing the program as a pair, David's lack of vision meant that this two-person approach was not experienced by David in any way. This contrasted to sighted participants Hairy and HTBP, who could watch the other's movements while following their own thread, understanding how they interacted. To try and achieve the same experience for David, the facilitator suggested that David changes the threads such that only one sound was heard at a time, ie. the melody and the drum did not execute at the same time. This was eventually achieved by David and his father by shortening the threads and then using trial and error. David was then able to follow both threads of the program himself.

5.2.2.3 Tracing to Debugging

Tracing was an excellent starting point to support debugging activities. *Penelope*(B), for example, was creating a sequence as part of an exercise on threads in her second lesson with Reuben. The program did not play as she intended, so she traced it physically. Her hand went across the beads stopping on the forth one as she waited for it to play. Her hand then scooted back to the third bead as she announced, 'there is a misconnection here'. She unplugged and replugged the forth bead and then her program worked. We observed many such examples across all of our participants. Physically tracing the program meant that their hands were in the right place to fix a program, whether the problem was a connection issue, or most likely, the need to adjust the parameters of a bead through the dials or buttons.

In pairs in which there were no sighted children, e.g. *Penelope(B)* and *Reuben(B)* and *Cat(PS)* and *Grace2455(RB)*, the majority of tracing was done hand over hand. That is, both children would physically trace the program together. With Penelope and Reuben, this meant that there was a shared awareness of what the other was doing, enabling talk such as 'there is a bug here'. For Cat and Grace2455 it led to shared fixing of bugs, one holding the bead while the other adjusted the parameters – a four-handed interaction. This contrasted with pairs with sighted children who often pointed to rather than touched the beads. It was unclear whether this was simply visual preference or whether the intimacy of touching hands was less familiar to those children with sight.

5.2.2.4 Reflections on Design

The physical design of rounded beads and wires made it comfortable for blind children to quickly scan their program without the need for careful consideration of the beads' shape. Those with colour vision made use of the visual cues to locate the beads as evidenced by not touching the wires in between the beads. We did note that a full sequence of eight beads required children either to stand up to reach the end or for it to be wound around in a s-shape, requiring more dedicated attention from the children to follow the wires. If non-connected beads got too close, they could be confused as adjacent by children not meticulously following the wires.

The numerous examples in this section suggest that Torino supports tracing of programs, both sequences and more complex programs, by the children across the visual spectrum using both tactile and visual processes. In the read-before-you-write approach to programming (Lister et al., 2004), tracing is the first step in the learning process. The second step is to be able to write, or create, programs. We discuss in the next section, how children created programs. We begin with a focus on how the children engaged with the physical affordances of Torino, before discussing some of the vignettes that relate.

5.2.3 Creating Programs

5.2.3.1 Creating with a physical language

Immediately, the children started plugging beads together to create sequences, not requiring any instructions or additional motivation. The children then started pressing the dials and buttons, spinning or clicking them very quickly at first, before recognizing that slow movement allowed them to choose a sound. This process was iterative and incremental, as suggested in the vignettes below.

> *Reuben(B)* and *Penelope(B)* are being introduced to Torino in their first lesson. The facilitator hands Reuben a play bead and Penelope a pause. They each describe their bead to the other while Reuben impatiently asks several times: "Does it plug into the holes on this [hub]?" The facilitator allows Reuben to plug his bead in and then encourages Penelope to plug hers in, while Reuben hands are rapidly patting the mat to see what else is there. Reuben then starts rapidly spinning the dial while Penelope patiently awaits her turn.
>
> *Cat(PS)* and *Grace2455(RB)* are putting together a sequence with all the beads in their second session. The two girls take out all the beads from the tray and split them on the mat in front of them. Cat volunteers to go first, grabbing the hub and quickly plugging in all her beads. She then sets about setting the dials, while Grace2455 comments on whether she likes the sound. Grace2455 decides to hand out her beads to people in the room, including the facilitator and two other observing researchers. She then directs them to plug them in. They have completed their sequence.
>
> *David(B)* and *Charlotte(S)* work together to create their first sequence with sounds of their choice at the end of the first lesson. David's teacher encourages him to go first. Ginny hands David a bead as David gets distracted by the hum of the speaker in the hub and the vibrations that it affords. David finds it very difficult to find the hole on the hub. Ginny lifts the hub up for him to position it in a way that makes David's plugging successful. Ginny then quickly plugs in the next one.

While the children were encouraged to plan their programs during certain exercises, there was a strong preference for plugging in one bead at a time and seeing what happened. Our observations, as captured in the above vignettes, suggest that the affordances of the design encourage interaction with the beads, either plugging or

setting the parameters through the dials/buttons. The way the children achieved creation jointly depended on the pair, something we explore in the next several paragraphs.

We saw a number of examples of how creating with a physical language facilitated creation between pairs of children. For example, *Penelope(B)* asked *Reuben(B)* to help her identify the type of bead in her hand: '*I'm not sure the one I've got is a play*.'; reaching out and feeling its shape, Reuben responds: '*No, it is not, it's a pause*' (see Figure 7 (top). In pairs with mixed visual ability, handing beads became a way round the complex verbal interactions required when vision is not a shared sense. For example, *Ginny(S)* pointed out to *Fin(B)* that there was another play bead next to him, saying 'There is one there'. Fin paused but did not respond as he could not interpret 'there'. Ginny then reached over and handed him the play bead that he needed to add to the program next, avoiding the lengthy (and complicated) description of: there is a play bead at 10 o'clock to your left arm.

Audio feedback played another, less expected role, in facilitating collaboration. The hub generated almost in-audible vibrations of its built-in speaker that were immediately noticeable to each of the blind children. While the hub could be repositioned by someone else to be brought closer to a child, each child without sight could quickly locate it on the table by sound. Similarly, the subtle clicks of the mechanisms made it possible for the blind children to locate the bead that their partner was working on. Whilst an unintended side-effect of the hardware set-up, in the absence of vision, these sounds presented a useful design feature.

Our sighted pair, *Hairy(S)* and *HTBP(S)* also benefited from the physicality of the language. They spent much time, with some prompting from the facilitator, in 'laying-out' their program and discussing it before plugging it in. For example, these boys alternatively grabbed beads and put them in a line, with the other partner rearranging if they felt a mistake had been made. While laying out beads is a highly visual approach to discussion, a contrasting way of using the physicality of the beads in communication was seen with *Cat(PS)* and *Grace2455(RB)*. Debating about what to do about a non-working bead, Grace2455 'plugged' herself in, by attaching her pinky finger to the connector jack, and performed the final note of the program (Figure 7 (bottom)).

We also observed most of the children persistently 'fiddling' with the program. This came in various guises. *Reuben(B)*, for example, frequently twiddled an unattached bead in his hands (Figure 7 (middle)). His mother referred to this as a useful 'fiddle thing', allowing Reuben to engage without the typical 'fiddle things' in his pocket, such as magnets. Others fiddled through repeatedly making small adjustments, and experimenting with minor changes to the sounds they were making. *Grace2455(RB)*, had to find the perfect sequence of comedy kiss sounds to make her mother laugh. *Fin(B)*, like many blind children with a love of buttons, engaged repeatedly in moving through the different natural sounds, e.g. ambulance, glass breaking etc. This type of persistent, in-the-moment behaviour is often correlated with 'flow' states (Csikszentmihalyi, 1996).

[Figure 7 about here]

5.2.3.2 Sequences, Loops and Threads

Soon after creating sequences, many of the pairs (unless discouraged by the facilitator) experimented with threads wondering what would happen if beads were plugged into different ports. Here is a short vignette of pair experimenting with threads:

> Session 1 (8 minutes):
> Grace2455: Can I plug them all into different corners?
> Facilitator: What do you think would happen if you plugged them into different corners?
> Cat: All the different ones would go into one.
> [Cat tries]
> Facilitator: What is happening?
> Cat: They are playing at the same time!
>
> Session 1 (22 minutes):
> A multi-threaded program has been created after the pair have handed out beads and allowed people to plug them anywhere.
>
> Facilitator: What happened?
> Grace2455: It played at the same time.
> Facilitator: How do we make them play one after the other?
> Cat: Put them all into the different plugs.
> Grace2455: I think we need to join them together.
> Facilitator: How are we going to do that?
> Grace2455: …Two more plays … No. Doesn't work. Can't connect them.
> ….
> Grace2455: We could put them into all one. All piano.

In their third sessions, *Grace2455(RB)* and *Cat(PS)* both independently described correctly the behaviour of threads. It is interesting here how much repetition of experimenting with threads was needed before the understanding solidified.

Loops, unlike threads, had a specific bead associated with them. However, several of the younger children, e.g. *Charlotte(S)* and *Fin(B)* surmised that bending a sequence into a 'circle' would make it repeat before the loop bead was introduced. This was similar to what we saw with the pairs in the design workshops. This suggests that being able to physically represent the loop achieves a useful mapping between form and function.

Deeper understanding of loops came through challenges that forced children to use loops to avoid running out of beads. For example, *Cat(PS)* worked in her third session with her mother to recreate a Jingle Bells program (see Video Figure 3). She begins by plugging in all the play beads she has until she runs out. She looks ponderously at her mother who responds, 'that's not going to work'. Cat then proposes pauses (rest beads). While this would increase the number of beads, it would not play the program. Her mother responds with the question 'pauses?' followed by a question of 'how many times do you need to say Jingle?'. Cat ponders that she has to say Jingle three times, but doesn't have enough beads, suddenly realising that she needs to use a loop. This example illustrates how the constraints of a physical language can encourage exploration of new computational concepts.

5.2.3.3 Parameters

Torino enables parameters to be physically set on the beads through manipulating dials or buttons. While many children first enjoy the fiddling that these provided, we also observed several examples that suggested a more abstract understanding of parameters

as a mechanism to specify state. For example, *Cat(PS)* realised that she could plug her bead into another port to set the state, in this case duration, while her partner was working on their sequence. *Penelope(B)*, for example, counted the number of pitch parameters and duration parameters available and then created her program to be a pattern of these – 1, 5, 8 for pitch, 1, 3, 1 for duration. *Reuben(B)* grasped the idea of how duration parameters related and used it to ensure his threads did not play at the same time by adding up durations.

We also saw notions of how children began to understand scope in their programs. *Reuben(B)* for example correctly predicted that nothing would play after an infinite loop, but he did get confused that one loop could be infinite, while another loop, within a separate thread in the program, could have a specified number of iterations. *Grace2455(RB)* realised that while loops could be constructed in the same way, they could have different instruments by plugging them into different ports. Other children struggled with scope as well, some wondered if bits outside the loop will loop, even when they were able to correctly demonstrate their functionality by pointing which bits will loop. These examples show how the children are beginning to grasp abstractions associated with variables, which is described as a significant determinant of success in programming education (Bornat, Dehnadi, & Barton, 2012).

5.2.3.4 Computational Language

An important aspect of working with a physical language that does not use text is reinforcing relevant computing language. *Cat(PS)*, for example describes a program to her mother as: *"A program is a set of instructions that tells something what to do"*. *Grace2455(RB)* began her second session by drawing out the blocks on the whiteboard and describing their function: "*A loop repeats something that you play. Play plays a note. Pause makes a break". David*(B) was able to get past his initial anger if a bead wasn't right, by saying: *"Oh. It needs debugging*". He could therefore turn what could be a very upsetting moment into a comforting plan of action with language. Putting names to the concepts not only helped solidify them, but gave form to how to think computationally. It wasn't a matter of plugging things together, but thinking about problems in a particular way, and with specific words.

5.2.3.5 Reflections on Design

The children were able to create programs and benefitted from the preservation of state when they were moved about the table to accommodate reach. There were numerous examples that suggested that the form of the program (e.g. loop) usefully gave information about function. The close mapping that we tried to achieve in Torino worked mostly, even if it constrained the complexity of the program. Indeed, we also saw examples, in which the constraints of the physical language could encourage the use of computational concepts, such as loops. Effort was needed, however, to map the physical to appropriate verbal language, such as 'sequence' or 'debug', as it was not embedded in the code in the same way it would be in a visual- or text-focused language.

Most noticeable in the design however, was the amount of fiddling that took place. We saw examples of that fiddling being a way to engage the hands through twiddling knobs as well as a means to iterative and incremental learning through slowly building up a

picture of a program through a trying out of different options (Lye & Koh, 2014). Children, for example, plugged beads into multiple ports to see what happened. They also learned about the duration parameter through continuously clicking through the buttons and listening to the outcome. For some children, this experience gave them the concept of an array suggesting how fiddling led to abstract concepts not directly portrayed by the form.

We also see examples of collaboration taking place between children utilising the physical nature of the programming language. While this differs depending on the visual abilities present in the pair, it was observed in all pairs. This is discussed in more detail in a different paper (Thieme et al., 2017).

6 DISCUSSION

In this paper, we presented the design and implementation of Torino, a novel tangible programming language for teaching computer programming to children ages 7-11 regardless of level of vision. Taking an inclusive design approach, the design is derived from a series of workshops with children with visual disabilities. During these workshops, we learned that common design elements of tangible languages, such as blocks connected by magnets, were problematic for this group. Some issues were practical, such as tactile learning requires information structures to be maintained while being manipulated. Others were more subtle, for example, the ability to pick up and manipulate objects is central to a tactile experience. We developed a design based on the concept of a 'bead metaphor' whereaby each program command is represented by a different shaped bead on a string, and developed novel hardware to achieve this.

To further iterate on this design, we carried out an exploratory evaluation with a small group of children with mixed visual ability to understand how Torino supports key computational thinking and programming practices. Taking inspiration from the read-before-you-write approach (Lister et al., 2004), we considered first how children read or 'traced' their programs and then how they created them. Aside from the usability issues identified, we described how all children could trace sequences by the end of the sessions, and the ways that this differed between children depending on their use of visual or tactile skills. In either case, children verbalised their program while tracing its execution.

We found that Torino supported an iterative and incremental approach to creating programs which presents an important part of computational learning as identified in an overview of the literature for school aged learners by (Lye & Koh, 2014). We noted that the affordances of the physical language encouraged experimentation, whether it be plugging beads together, turning the dials, or plugging beads into multiple ports on the hub to discover threads. We illustrated a number of ways in which the children were able to collaborate when vision is not a shared sense through the physical language, a topic explored more deeply in (Thieme et al., 2017). Building on the design reflections in Section 5, we now reflect on two major design decisions with substantial trade-offs: 1) Readability versus extensibility; 2) Liveness versus bulkiness.

We put significant emphasis on readability of the program over extensibility. That is, one of our primary design considerations was that a child has a persistent overview of their program at all times. That meant leaving out functions or more complex notions of variables that would be hidden inside the physical state. Such a choice, while possibly most useful for learning in the early stages, could potentially limit the longevity of Torino in a classroom and hence its viability. This is particularly true given the limited number of beads available and the expense of the hardware to create the 'liveness' of the experience. We had created a low floor, but was the ceiling high enough? – to borrow the phrase from (Maloney et al., 2010).

While the children were limited in the programming constructs that they were introduced to: sequences, loops, and threads, we found that the limited number of beads and emphasis on tracing opened up opportunity for substantial focus on areas of computational thinking. We showed a number of examples of how children's understanding of program execution was pushed by exercises with loops and threads in which the ability to trace became part of a larger activity, e.g. figuring out how to identify the first bead in a loop, or being able to follow two threads simultaneously by understanding how they executed together. On reflection, the one programming construct that was missing was conditional statements (or more formally called selection in the UK curriculum).

Our analysis also pointed to the importance of only being able to create short programs with Torino. As the computational challenges increased, the limited number of beads was essential in allowing focus on these programming constructs and computational tasks. This focus helped the children to work out the computational challenges without the need for extra support. Further, on reflection, we found that scoping the tracing problem such that children can work it out is more valuable educationally than showing the children how a program executes. This stands in contrast to approaches in ILEs (e.g. (Meerbaum-Salant, Armoni, & Ben-Ari, 2010)) in which children focus on how to make the end product, rather than how to use programming constructs effectively.

We further observed how engagement with the parameters prompted an understanding of abstraction in some children, particularly as it related to variables. We described examples of understanding: the maintenance of state, implicit understanding of arrays, as well as concepts of scope. That said, given that grasping abstractions associated with variables is believed to be a determinant of success in programming education (Bornat et al., 2012), more could be done to expand the notion of pegs used to set loop iteration to other kinds of 'variables'. Moving beyond constants, future work should also include notions of random, infinity, and counters.

Increasing the number of variable types coupled with the simplicity of the language could open up many opportunities for exploring computation. For example, using random with a conditional could be used not only to build lessons around probability, but also to enable children to create dynamic stories with different endings. Indeed, functions could be addressed by allowing children to record their own sounds and thereby allowing them to record a program already 'created' and to re-use it. While this would cover all aspects of the primary curriculum, our reflections on how important the physical embodiment of

the language for tracing would discourage us from actively encouraging children to create functions this way.

Another way to 'raise the ceiling' is to think about the transitional experience to a text - based language. We touch on this briefly in the discussion of how we encourage appropriate verbal language when using a tangible language. However, the ability to read out, or view in an appropriately adapted way, the code in some form could help children connect the concepts presented tangibly with the way linear code is written. Just as we encouraged children to trace a program as it executes, they could trace the program when listening to the code. The literature suggests that transitioning from block-based coding to text-based coding is challenging and we would expect the same for a tangible language (Weintrop & Wilensky, 2015). Yet, supporting text-based coding literacy may also be a valuable conduit in increasing children's confidence in their coding abilities (Stead & Blackwell, 2014).

The other substantial trade-off considered was the bulkiness of programs versus the liveness of the beads. Including electronics in each bead to make each one individually responsive increased the size of the beads. Even with just eight beads in a program, most children needed to stand to trace to the end or curl their program around, making tracing more difficult. Storage also presents a challenge. We've already pointed to the importance of keeping programs simple leaving us to conclude that eight beads might be enough. Despite reservations that the beads were too big, they were a good size for four-handed interactions and much time was spent manipulating or fiddling with them suggesting that size was not an insurmountable problem.

More importantly, we regard the liveness of the beads as a key factor in the children's iterative development of programs, and their understanding of them. The ability to approach the problems from iterative attempts with the materials can be thought to provide positive task feedback, reinforcing satisfaction with new experiences, and motivation to continue experimenting (Nash & Blackwell, 2014). It also has positive educational outcomes, as documented in the literature on 'tinkering' in computer science education. Tinkering – technical manipulation that is driven by curiosity rather than a specified goal – is strongly associated with the development of self-efficacy in educational contexts. Design strategies to encourage tinkering result in improved educational outcomes for girls and other groups with low self-efficacy (Beckwith et al., 2006). Taking these observations in line with the literature, suggests that the persistent 'fiddling' with Torino could demonstrate longer-term benefits in building self-efficacy and result in higher educational attainments.

Last, we reflect upon the role of Torino as a means of inclusive teaching. We demonstrate that creating and tracing programs can be done regardless of level of vision. There is nothing inherent in Torino that requires assistance. That said, facilitation played an important role in the use of Torino. On the one hand, focusing on computational thinking tasks (e.g. where does the loop start) rather than on just getting to an end goal, requires someone with the skills to ask such questions. This suggests that Torino requires a scheme of work to successfully be used by non-specialist teachers.

On the other hand, integrating blind and sighted students together is also a challenge in any educational context and requires facilitation (Metatla, 2017). While we saw examples of children using the physical beads to overcome the challenges of interacting when vision is not a shared sense, such skills only come with practice and support from facilitators. However, despite some extremely fast fingers, it is more time consuming to 'see with your hands'. This often meant that sighted partners needed to wait, or help out. These issues are instantiations of existing challenges of integrating blind children into mainstream environments and cannot be solved by the technology alone.

An important next step, following a design iteration based on this initial evaluation, is the evaluation of the technology on a larger scale. Such an evaluation can explore in more depth measures of engagement and learning as well as explore the issues that arise when Torino is being used in a mainstream classroom.

7 CONCLUSION

There is a strong diversity agenda behind the push to get more children fluent with computational thinking and able to use it for the benefit of society. This agenda has mainly focused on gender and ethnicity, but disability is an important segment that should not be ignored. To this end, appropriate tools are needed to support learning. While efforts have been made for older blind students, there has been little attention to computing education for blind learners in early years and primary schooling. This is particularly problematic given the recent mandatory introduction of computing for the primary school age group in several countries, including the UK. Torino aims to address this gap by providing a tangible language for teaching programming and computational thinking concepts to children ages 7 – 11 inclusive of those with visual disabilities.

Having taken an inclusive design approach to the development of Torino we created a learning tool that is appropriate for use by children with visual disabilities without its design being 'exclusive to them'. The development of a 'bead' metaphor to provide a 3-dimensional experience that prioritises tactile interaction over visual understanding sets Torino apart from many tangible programming languages and interactive blocks available in the research literature, and on the market. Existing designs are all block shaped, relying on the metaphor of puzzle pieces. Working with children with visual disabilities in the Torino Young Design Team helped us reimagine this space and explore novel ways of interactions that would not rely on 'vision for interpretation'. As a result, taking an inclusive design approach enabled us to innovate both in form and underlying technology.

9 REFERENCES

Baker, C. M., Milne, L. R., Drapeau, R., Scofield, J., Bennett, C. L., Ladner, R. E., … Richard, E. (2016). Tactile Graphics with a Voice ACM Reference Format : *ACM Transactions on Accessible Computing (TACCESS)*, *8*(1), 1–22.

Baker, C. M., Milne, L. R., & Ladner, R. E. (2015). StructJumper. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15* (pp. 3043–3052). New York, New York, USA: ACM Press.

Beckwith, L., Kissinger, C., Burnett, B., Wiedenbeck, S., Lawrance, J., Blackwell, A., & Cook, C. (2006). Tinkering and gender in end-user programmers' debugging. In *Proceedings of CHI* (pp. 231–240).

Bigham, J. P., Aller, M. B., Brudvik, J. T., Leung, J. O., Yazzolino, L. a., & Ladner, R. E. (2008). Inspiring blind high school students to pursue computer science with instant messaging chatbots. *ACM SIGCSE Bulletin*, *40*(1), 449. http://doi.org/10.1145/1352322.1352287

Bornat, R., Dehnadi, S., & Barton, D. (2012). Observing Mental Models in Novice Programmers. In *24th Annual Workshop of the Psychology of Programming Interest Group*.

Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008). The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 423–432). ACM.

Burg, B., Kuhn, A., & Parnin, C. (2013). 1st international workshop on live programming (LIVE 2013). In *International Conference on Software Engineering (ICSE)* (pp. 1529–1530).

Burgstahler, S. E., & Ladner, R. E. (2007). Increasing the participation of people with disabilities in computing fields. *Computer*, *40*(5), 94–97. http://doi.org/10.1109/MC.2007.175

Burnard, P. A., Florack, F., Blackwell, A. ., Aaron, S., & Philbin, C. A. (2017). Learning from Live Coding. In *The Routledge Companion to Music, Technology, and Education.* (pp. 37–48).

Clarkson, P. J., Coleman, R., Keates, S., & Lebbon, C. (2013). *Inclusive design: Design for the whole population*. Springer Science & Business Media.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: A 3-D Tool for Introductory Programming Concepts. In *Proceedings of the Fifth Annual CCSC Northeastern Conference on The Journal of Computing in Small Colleges* (pp. 107–116). USA: Consortium for Computing Sciences in Colleges.

Csikszentmihalyi, M. (1996). *Creativity: Flow and the Psychology of Discovery and Invention.* Harper Perennial.

Department for Education. (2013). Computing programmes of study : key stages 1 and 2 National curriculum in England. Retrieved from http://www.computingatschool.org.uk/data/uploads/primary_national_curriculum_-_computing.pdf

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn coding? In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education.*

Edge, D., & Blackwell, A. (2006). Correlates of the cognitive dimensions for tangible user

interface. *Journal of Visual Languages & Computing*, *17*(4), 366–394.

Fincher, S., Cooper, S., Kölling, M., & Maloney, J. (2010). Comparing Alice, Greenfoot & Scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 192–193). New York, NY, USA: ACM. http://doi.org/10.1145/1734263.1734327

Horn, M. S., Crouser, R. J., & Bers, M. U. (2012). Tangible interaction and learning: the case for a hybrid approach. *Personal and Ubiquitous Computing*, *16*(4), 379–389. http://doi.org/10.1007/s00779-011-0404-2

Horn, M. S., & Jacob, R. J. K. (2007). Designing Tangible Programming Languages for Classroom Use. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction* (pp. 159–162). New York, NY, USA: ACM. http://doi.org/10.1145/1226969.1227003

Horn, M. S., Solovey, E. T., Crouser, R. J., & Jacob, R. J. K. (2009). Comparing the use of tangible and graphical programming languages for informal science education. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.*, *32*, 975. http://doi.org/10.1145/1518701.1518851

Horn, M. S., Solovey, E. T., & Jacob, R. J. K. (2008). Tangible programming and informal science learning. In *Proceedings of the 7th international conference on Interaction design and children - IDC '08* (p. 194). New York, New York, USA: ACM Press.

Hornecker, E., Marshall, P., & Rogers, Y. (2007). From entry to access: How shareability comes about. In *Designing Pleasurable Products and Interfaces* (pp. 22–25). http://doi.org/10.1145/1314161.1314191

Hu, F., Zekelman, A., Horn, M., & Judd, F. (2015). Strawbies: explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 410–413).

Johnson, R., Shum, V., Rogers, Y., & Marquardt, N. (2016). Make or shake: An empirical study of the value of making in learning about computing technology. In *The 15th International Conference on Interaction Design and Children* (pp. 440–451).

Kane, S. K., & Bigham, J. P. (2014). Tracking @stemxcomet: teaching programming to blind students via 3D printing, crisis management, and twitter. In *Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14* (pp. 247–252). New York, New York, USA: ACM Press.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 1455–1464).

Langdon, D., McKittrick, G., Beede, D., Khan, B., & Doms, M. (2011). *STEM: Good Jobs Now and for the Future. ESA Issue Brief #03-11*. US Department of Commerce.

Lechelt, Z., Rogers, Y., Marquardt, N., & Shum, V. (2016). ConnectUs: A new toolkit for teaching about the Internet of Things. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 3711–3714).

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., … Werner, L. (2011).

Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32. http://doi.org/10.1145/1929887.1929902

Leuders, J. (2016). Tactile and acoustic teaching material in inclusive mathematics classrooms. *British Journal of Visual Impairment*, *34*(1), 42–53. http://doi.org/10.1177/0264619615610160

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., … Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, *36*(4), 119–150. http://doi.org/10.1145/1044550.1041673

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the fourth international workshop on computing education research* (pp. 101–112).

Ludi, S. L., Ellis, L., & Jordan, S. (2014). An Accessible Robotics Programming Environment for Visually Impaired Users. *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility*, 237–238. http://doi.org/10.1145/2661334.2661385

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *Trans. Comput. Educ.*, *10*(4), 16:1–16:15. http://doi.org/10.1145/1868358.1868363

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (Moti). (2010). Learning computer science concepts with scratch. In *Proceedings of the Sixth international workshop on Computing education research - ICER '10* (p. 69). New York, New York, USA: ACM Press.

Metatla, O. (2017). Uncovering Challenges and Opportunities of Including Children with Visual Impairments in Mainstream Schools. In *British HCI 2017* (pp. 1–7).

Moon, N. W., Todd, R. L., Morton, D. L., & Ivey, E. (2012). Accommodating students with disabilities in science, technology, engineering, and mathematics (STEM): Findings from research and practice for middle grades through university education.

Nash, C., & Blackwell, A. F. (2014). Flow of creative interaction with digital music notations. In *The Oxford Handbook of Interactive Audio* (pp. 387–404).

Papazafiropulos, N., Fanucci, L., Leporini, B., Pelagatti, S., & Roncella, R. (2016). Haptic Models of Arrays Through 3D Printing for Computer Science Education. *International Conference on Computers Helping People with Special Needs*, 491–498. http://doi.org/10.1007/978-3-642-31522-0

Petit, G., Dufresne, A., & Levesque, V. (2008). Refreshable tactile graphics applied to schoolbook illustrations for students with visual impairment. *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, 89. http://doi.org/10.1145/1414471.1414489

Peyton Jones, S. (2013). Computing at school in the UK : from guerrilla to gorilla.

Communications of the ACM, (April), 1–13.

Randall, D., Harper, R., & Rouncefield, M. (2007). *Fieldwork for Design*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172. http://doi.org/10.1076/csed.13.2.137.14200

Rogers, Y., Shum, V., Marquardt, N., Lechelt, S., Johnson, R., Baker, H., & Davies, M. (2017). From the BBC micro to micro: bit and beyond: a British innovation. *Interactions*, *24*(2), 74–77.

Sánchez, J., & Aguayo, F. (2005). Blind learners programming through audio. In *CHI '05 extended abstracts on Human factors in computing systems - CHI '05* (p. 1769). New York, New York, USA: ACM Press. http://doi.org/10.1145/1056808.1057018

Sentance, S., & Csizmadia, A. (2016). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 1–27. http://doi.org/10.1007/s10639-016-9482-0

Shi, L., Zelzer, I., Feng, C., & Azenkot, S. (2016). Tickers and Talker: An Accessible Labeling Toolkit for 3D Printed Models. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 4896–4907. http://doi.org/10.1145/2858036.2858507

Shum, A., Holmes, K., Woolery, K., Price, M., Kim, D., Dvorkina, E., … Malekzadeh, S. (2016). *Inclusive: a Microsoft design toolkit*.

Smith, A. C., Francioni, J. M., & Matzek, S. D. (2000). A Java programming tool for students with visual disabilities. In *Proceedings of the fourth international ACM conference on Assistive technologies - Assets '00* (pp. 142–148). New York, New York, USA: ACM Press.

Smith, D. C., Cypher, A., & Schmucker, K. (1996). Making Programming Easier for Children. *Interactions*, *3*(5), 58–67. http://doi.org/10.1145/234757.234764

Sorva, J. (2013). Notional Machines and Introductory Programming Education. *ACM Transactions of Computing Education*, *13*(2).

Stead, A., & Blackwell, A. F. (2014). Learning Syntax as Notational Expertise when using DrawBridge. In *Proceedings of the Psychology of Programming Interest Group Annual Conference(PPIG 2014)* (pp. 41–52).

Stefik, A. M., Hundhausen, C., & Smith, D. (2011). On the design of an educational infrastructure for the blind and visually impaired in computer science. In *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11* (p. 571). New York, New York, USA: ACM Press.

Sullivan, A., Elkin, M., & Bers, M. U. (2015). KIBO robot demo: engaging young children in programming and engineering. In *Proceedings of the 14th international*

conference on interaction design and children (pp. 418–421).

Suzuki, H., & Kato, H. (1995). Interaction-level support for collaborative learning. *The First International Conference on Computer Support for Collaborative Learning - CSCL '95*. http://doi.org/10.3115/222020.222828

Tanimoto, S. (1990). VIVA: A visual language for image processing. *J. Vis. Languages Computing*, 127–139.

Thieme, A., Morrison, C., Villar, N., Grayson, M., & Lindley, S. (2017). Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (pp. 739–752).

Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, Greenfoot, and Scratch – A Discussion. *Trans. Comput. Educ.*, *10*(4), 17:1–17:11. http://doi.org/10.1145/1868358.1868364

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 199–208).

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education*, *12*(3), 197–212.

Yip, J. C., Foss, E., Bonsignore, E., Guha, M. L., Norooz, L., Rhodes, E., … Druin, A. (2013). Children Initiating and Leading Cooperative Inquiry Sessions. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 293–296). New York, NY, USA: ACM. http://doi.org/10.1145/2485760.2485796

Zuckerman, O., Grotzer, T., & Leahy, K. (2006). Flow blocks as a conceptual bridge between understanding the structure and behavior of a complex causal system, 880–886.

10APPENDIX

10.1 Cognitive Dimensions and Tangible Correlates Table

Dimension	Definition	Analysis of Torino Design
Consistency<CD>	Similar semantics are expressed in similar syntactic forms.	Play and pause beads have the same control mechanism for duration and different port shapes for variables indicates a semantic difference (+), but loop does not have a physical control mechanism (-).
Provisionality<CD>	Actions or marks can be reversed or removed.	Physical controls support sound experimentation (+), but sounds cannot be unheard and programs must be created and debugged in a shared audio space even before they are 'ready' for sharing (-).
Secondary Notation<CD>	Information can be expressed outside the formal syntax.	The arrangement of beads on a tabletop surface can be appropriated in meaningful ways (+), but these might not be apparent to collaborators (-).
Progressive Evaluation<CD>	Progress-to-date can be checked at any time.	The program audio can be played at any time (+) but the program state must otherwise be inferred from handling or inspection (-).
Premature Commitment<CD>	The order of doing things is unnatural or overly constrained.	Beads can be plugged in any order (+), but the instrument of each thread is fixed and predetermined (-).
Bulkiness<TC>	Physical objects or representations occupy space in three dimensions.	Beads are palm-sized for easy handling (and large enough for the embedded electronics and physical controls) (+), but their connection to form networks could be constrained by the available workspace (-).
Permanence<TC>	Physical representations and control mappings can be preserved for future use.	Physical representations are preserved (+), but pressure on scarce beads (which may arise due to cost and sharing requirements) could require disassembly and recycling of parts (-).
Shakiness<TC>	Physical representations are prone to accidental or irreversible damage.	Snap-lock wired connections are robust (+), but the physical control mechanisms have the potential to be 'knocked' accidentally (-).
Juxtamodality<TC>	Multiple interaction modalities are coordinated across different physical spaces or objects.	A blind child can simultaneously manipulate bead controls by hand and listen to the resulting sounds (+), but must rely on tactile feedback alone when in the process of connecting beads (-).
Rigidity<TC>	Manipulation of objects or their arrangement is resisted.	Manipulation of bead controls is fast and direct (+), but inserting or reordering beads requires multiple disconnection and reconnection actions (-).
Rootedness<TC>	Movement of objects or their arrangement is resisted.	The hub is not physically tied to any particular location (+), but growing bead networks may make it hard to move (as well find, for blind children) (-)
Automation<TC>	New behavior can be programmed and redefined.	Physical state embodies replayable audio output (+), but playback much be invoked manually (-).
Adaptability<TC>	New states can be specified and redefined.	Adults can re-program the ports with different sound sets, including children's own sounds (+), but the meanings of beads are fixed (-).
Purposeful Affordances<TC>	Possible physical actions have a clear and meaningful purpose.	Physical mechanisms have clear affordances (+), but the use of up and down buttons to cycle through four duration options is not obvious (-).
Hidden Augmentations<TC>	Physical objects are digitally augmented in a non-obvious manner.	Use of connecting wires gives a clear indication of how the physical beads are digitally augmented (+), but injection of variables is less obvious (-).
Unwieldy Operations<TC>	Demand on physical resources because of the actions required on objects.	Bead controls can be manipulated with the hand holding the bead (+), but holding or manipulating more than one bead per hand is difficult (-).
Structural Correspondence<TC>	Physical structure matches the underlying digital representation.	Each bead represents a single program instruction (+), but variables have different use syntax (multiple references in code vs actions on objects) (-).

10.2 Lesson Plan 1

1. Introductions
    a. Introduce people at the table
    b. Introduce Torino, a programming language that you can touch
    c. Introduce the idea of a prototype, it breaks and needs to be restarted often.
    d. Demonstrate Sonic Pi

2. Introduce the Torino Beads
    a. Vocabulary:
        i. Programs are sets of instructions that tell the computer precisely what to do.
    b. Orientation
        i. Invite each blind child to physically explore the programming space
    c. Introduce Beads
        i. Give a bead to each child and ask them to describe it to their partner
    d. Make a simple program
        i. Invite children to plug them together in a program and manipulate the parameters
3. Create a Program
    a. One child reads out a prepared program from paper and the other puts it together
    b. Each child takes a turn at reading the program
    c. The children can then set the parameters to enjoy their program.

4. Race to Program
    a. Tell children a program (which they memorise) and time them in putting it together
    b. Discuss strategies for doing it faster. Get them to try at least two strategies and compare which one is faster.

5. Create a scale

6. Create the first line of Twinkle Twinkle

10.3 Lesson Plan 2

1. Sequence Review
    a. Each child is asked to take 3 plays and a pause bead
    b. The children decide who will create a melody and who will create a rhythm
    c. Children make their own thread and listen to the whole program
    d. Children are challenged to make sure that the piano and drum never play at the same time.

2. Loops
    a. Introduce loops
    b. Ask children to create a loop
    c. Ask children what they think the loop will do and then test their idea
    d. Ask them to adjust their program to something they like

3. Programming with Loops
    a. Introduce Iteration
    b. Provide program on paper and ask children to build it (contains loops, and different entry exits, pauses, threads)
    c. Follow the program when listening

4. Debugging
    a. Vocabulary: Debugging is the word we use to describe how we identify and fix a problem with our program
    b. The children tune the program to something that they like
    c. Record it.
    d. Adults make two changes and children must identify verbally and then fix (x2)
        i. Loop parameter
        ii. Bead swapping

5. Story Challenge or Jingle Challenge
    a. Give children a recorded story that can only be created on two threads and see how they try to solve
    b. Do both if time allows

10.4 Lesson Plan 3

1) Check understanding of basic concepts
   a) Explain to your parent/teacher what a program is and how to make one with Torino?

2) Can you make a program together?
   a) What is it that you want to make?
   b) What instruction beads do you need?

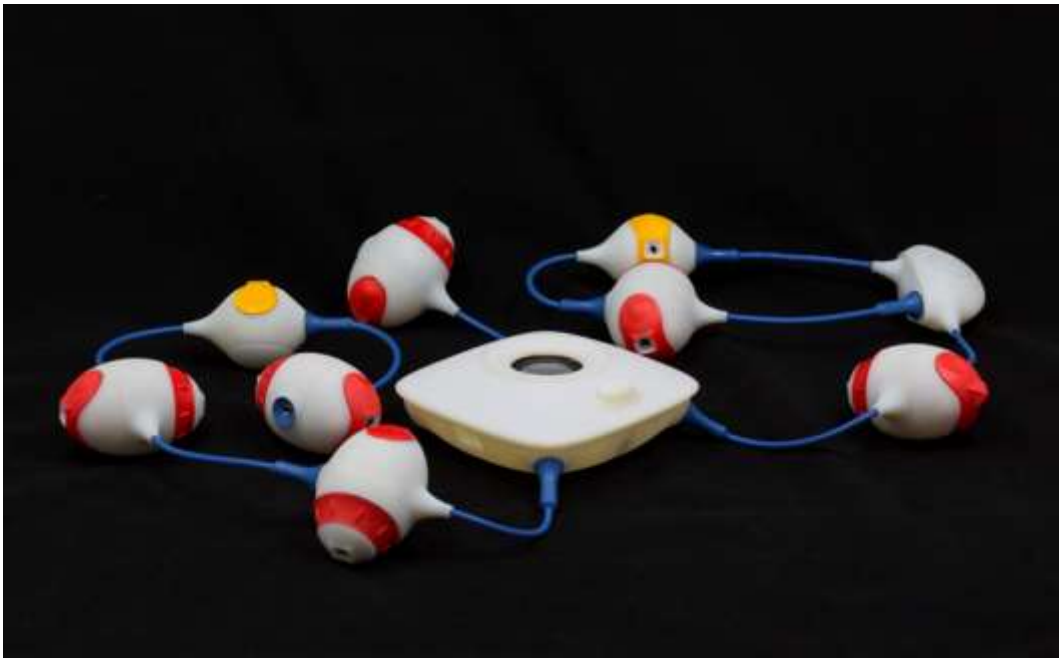
3) Parent Debugging
   a) Can you explain to your parent what debugging is?
   b) Parent leaves and child changes two things

4) Challenge

11 FIGURES

11.1 FIGURE 1

Figure 1. A Torino program demonstrating sequences, loops, and threads.



11.2 FIGURE 2

Figure 2. Objects built by the Torino Young Design Team.

11.3 Figure 3

Figure 3: Torino Yong Design Team playing with mechanism blocks.

11.4 Figure 4

Figure 4: Torino beads (left to right): Pause (yellow), Loop (white), Play (red), Hub (square).

11.5 Figure 5

```
Thread 1 Percussion
    Loop for infinity
            Play Ding for 1
            Pause for 1
    End Loop
```

```
Thread 2 Jingle Samples
    Loop for 2
            Play Jingle for 1
            Play Bells for 1
            Pause for 1
    End Loop
    Play Jingle for 1
    Play All for 1
    Play The for 1
    Play Way for 1
```

Figure 5: Sample Torino program with pseudocode

11.6 Figure 6

Figure 6: Participant characteristics

Pseudonym	Age	Gender	Visual Ability	Mobility aid	Participation in Design
Cat (PS)	8	F	Partially sighted	No	Yes
Grace2455 (RB)	8	F	Registered blind	Yes	Yes
Reuben (B)	8	M	Blind from birth (color vision)	Yes	Yes
Penelope (B)	12	F	Blind from birth (no vision)	Yes	Yes
HTBP (S)	10	M	Fully sighted	No	No
Hairy (S)	10	M	Fully sighted	No	No
Charlotte (S)	7	F	Fully sighted	No	No
David (B)	7	M	Blind from birth (no vision)	Yes	No
Ginny (S)	7	F	Fully sighted	No	No
Fin (B)	7	M	Blind from birth (color vision)	Yes	No

11.7 Figure 7

Figure 7: (top) Checking the bead; (middle) Twiddling; (bottom) Partaking in the program.