TIP-Toy: a tactile, open-source computational toolkit to support learning across visual abilities

Giulia Barbareschi
UCL Interaction Centre & Global Disability Innovation Hub, London, UK
giulia.barbareschi.14@ucl.ac.uk

Enrico Costanza
UCL Interaction Centre, London, UK
e.costanza@ucl.ac.uk

Catherine Holloway
UCL Interaction Centre & Global Disability Innovation Hub, London, UK
c.holloway@ucl.ac.uk

ABSTRACT

Many computational toolkits to promote early learning of basic computational concepts and practices are inaccessible to learners with reduced visual abilities. We report on the design of TIP-Toy, a tactile and inclusive open-source toolkit, to allow children with different visual abilities to learn about computational topics through music by combining a series of physical blocks. TIP-Toy was developed through two design consultations with experts and potential users. The first round of consultations was conducted with 3 visually impaired adults with significant programming experience; the second one involved 9 children with mixed visual abilities. Through these design consultations we collected feedback on TIP-Toy, and observed children's interactions with the toolkit. We discuss appropriate features for future iterations of TIP-toy to maximise the opportunities for accessible and enjoyable learning experiences.

CCS CONCEPTS

• **Human-centered computing**; • **Accessibility**; • **Accessibility technologies**;

KEYWORDS

Tangible user interfaces, visual impairment, accessibility, children education

ACM Reference Format:
Giulia Barbareschi, Enrico Costanza, and Catherine Holloway. 2020. TIP-Toy: a tactile, open-source computational toolkit to support learning across visual abilities. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '20), October 26–28, 2020, Virtual Event, Greece.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3373625.3417005

1 INTRODUCTION

Introducing computing in education for all young children is now understood to be greatly beneficial for several reasons. Learning how to code doesn't just enable children to program and actively engage with computers but also teaches them strategies to break down, analyse and solve problems by being logical and creative [26]. Previous studies show that it is important for children to engage in computing education as early as possible as stereotypes about who can or cannot code tend to form at a young age [19, 21].

Resnick & Silverman developed the guiding principles of "*Low, floors, high ceilings and wide walls*" to support the design of new toolkits for children to learn and explore the basics concepts of computer programming [55]. Alper, Hourcade & Gilutz [1] re-elaborated these concepts suggesting that to increase the accessibility of toolkits for programming education, children with disabilities need ramps to access low floors, tall ladders to reach high ceilings, frames of interest to support exploration of wide walls and reinforced corners to build confidence through successful interactions.

Engaging in computing from a young age can be difficult for visually impaired (VI) children, as many mainstream toolkits for simplifying programming activities in school use Graphical User Interfaces (GUIs) [37]. Recently, researchers have developed novel computing education toolkits suitable for children with mixed visual abilities. One approach is to increase the accessibility of block-based GUIs through audio feedback or a simplified navigation interface based on consistent spatial organization [34, 45]. Another strategy is to use tangible user interfaces (TUIs) in the form of objects that children can manipulate and arrange to create programming sequences [33, 51]. None of these strategies is or aims to be universally superior to the others, but researchers recognize that a variety of approaches should be used to meet the diverse needs and preferences of young learners with mixed visual abilities [33].

In this paper we introduce the Tactile Inclusive Programming (TIP)-Toy: an open source educational toolkit using physical blocks to facilitate learning basic computing concepts through music in children with mixed visual abilities. This toolkit was developed using a modular and flexible approach that offers considerable opportunities for expansion and personalization while keeping the core system as simple as possible. This paper contributes: the design of TIP-toy, an accessible open-source educational programming toy for children with mixed visual abilities, see Figure 1; the report of three interviews with visually impaired experts which informed the initial design iteration, and of five sessions where nine children with mixed visual abilities interacted with TIP-Toy. Through a qualitative analysis of our data we also contribute reflections on how the construction of ramps, ladders, frames of reference and reinforced corners could improve TIP-Toy's design.

ASSETS '20, October 26–28, 2020, Virtual Event, Greece

ACM ISBN 978-1-4503-7103-2/20/10…$15.00
https://doi.org/10.1145/3373625.3417005

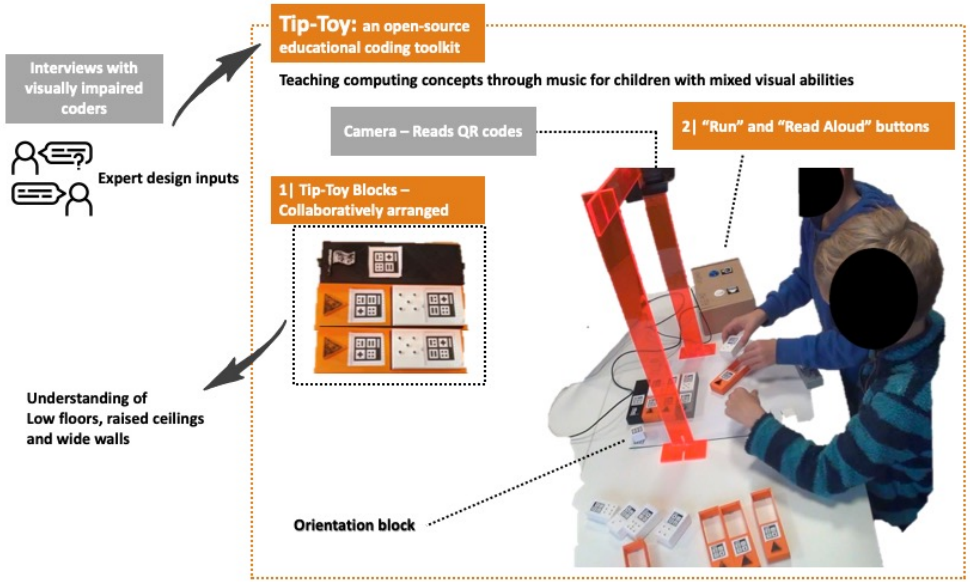

Figure 1: Overview of TIP Toy system

2 RELATED WORK

2.1 Low floors, raised ceilings and wide walls in computing education toolkits

The increase in popularity of computational activities among children has prompted researchers and designers to develop a plethora of new toolkits that are enjoyable and accessible to increasingly younger learners. Probably the most popular and salient characteristic of these toolkits is that they have "*low floors*", hence they are approachable for novices who can start using them regardless of their previous experience [55].

The most popular way to lower the entry threshold of toolkits for computational education is by using block-based structures that remove the need to learn complex syntax used by most traditional programming languages [31, 70]. Block-based programming languages generally feature GUIs where snippets of code are portrayed as pieces of puzzles that can be combined together to create a more complex set of instructions for a program. Scratch [54], Blockly [17] and Snap are some of the most widespread examples of block-based programming languages, but several others have been developed [2, 7, 23, 32].

Another strategy commonly used to facilitate learning basic computational concepts in children is to use TUIs [9, 48, 61, 69]. Leveraging the use of physical objects for programming builds on a combination of powerful concepts such as constructionism and embodiment by providing children with physical *objects-to-think-with* and enabling them to use their bodies to make sense of the world [15, 52]. TUIs can also facilitate collaboration between children, thus improving learning outcomes and are more enjoyable and easier to learn for younger children [59, 66, 72].

Computational toolkits with *high ceilings* enable children to explore increasingly complex problems and produce more sophisticated programs. Exploring more complex computational ideas, requires children to engage with a variety of computational concepts and thinking practices. Brennan & Resnick [11] developed a framework of seven computational concepts (sequence, loops, events, parallelism, conditionals, operators and data) and four computational practices (experimenting and iteration, testing and debugging, reusing and remixing, abstracting and modularizing) that are at the base of computing education. As shown in the review by Yu & Roque [73], concepts such as sequences, loops, events and data are supported by a variety of kits, whereas parallelism, operators and conditionals are less commonly explored. Similarly, being incremental and iterative, and testing and debugging are computational practices supported by a large number of toolkits, whereas reusing and remixing, and abstracting and modularizing are less common [73]. In short, the larger the number of computational concepts and practices supported by the toolkit, the higher the ceiling for the child.

Wide walls have been described as one of the most crucial features for computational toolkits as they enable diverse exploration and encourage creativity [55]. Computational toolkits have been widening the walls by increasing the number and type of outputs supported and by enabling children to move beyond a specific set of "recommended activities". Many computational toolkits, especially the ones based on GUIs, support visual outputs such as moving animations, blinking LEDs and simple game design [6, 16, 36]. Sounds is also common outputs for many computational toolkits. Sound can be combined with animation to produce multisensory effects [20, 54], or being used on his own to produce music or stories

[2, 63]. Finally, many computational toolkits focus on haptic outputs, especially through the use of robots that can be moved in space according to specific instructions. The Lego Mindstorms Kit is probably one of the most popular examples of computational toolkits using robotic outputs and have been used in many schools around the globe [35]. However, over the years many similar toolkits featuring different levels of complexity have emerged from both research and private enterprises [12, 62, 65, 74].

Another way to encourage diverse expressivity is to enable children to freely engage in different activities according to their own interests and abilities. Many kits set challenges or activities for children to complete. Challenges are usually constructed with increasing difficulty levels and to introduce new computational concepts as children progress through them [32, 35]. These challenges can come with or without a step-by-step set of instructions [9]. Moreover, they often provide scaffolding that enables the child to understand and correct mistakes so that the challenge can be successfully completed [36, 69]. Some kits also enable free exploration, inviting children to create their own stories, compose their own music, or modify and build robots to accomplish any task they wish [22, 23, 35]. Open ended projects are usually introduced at the end of a more scaffolded curriculum and are meant to provide greater challenge, and higher enjoyment due to the possibility to personalize outputs in a way that gives more meaning to the child's effort [30].

2.2 Ramps, ladders, frames of interest and reinforced corners for VI children

The strategies outlined above are usually sufficient for developing effective computational toolkits that only target non-disabled children. However, designing kits that are inclusive for children with VI requires consideration of their different abilities and preferences, which can only be obtained through direct engagement with VI children [8, 51]. Therefore, computational toolkits that aim to be accessible to VI children but have not been designed with them, will not be included in this section. Furthermore, the short review provided in this section focusses only on toolkits and languages using block-based structures, so we excluded other popular accessible programming languages such as Quorum.

R*amps* ensure that computational toolkits are accessible to children with VI [1]. Traditionally, accessibility was built into educational toolkits by providing screen reader compatibility [38], audio clues to aid block-based navigation [39], enabling the use of voice commands to enter coding instructions [58], or providing tactile outputs for data visualization [28]. However, these methods assume some proficiency with screen readers or a basic knowledge of coding structures, making them unsuitable for younger children [67]. Simplified navigation interfaces requiring the use of few keys to identify, select and position blocks within constrained geometries as suggested by Koushik & Lewis [34], could facilitate interaction, but still requires a level of abstraction that young children are yet to develop.

There are two successful strategies to build ramps that effectively support young children with VI. The first one, implemented by Block4All, leverages the use of accessible touchscreen interactions and a combination of audio and spatial clues to enable children to find and identify the blocks [46, 47]. This approach also uses touch-touch-drop interaction to simplify the action of selecting and combining blocks for VI children [47]. Second, and most popular, is through the use to TUIs that enable VI children to learn and explore computational concepts through manipulation and combination of physical objects.

A notable example of a TUI-based computational toolkit for VI children is Torino developed by Microsoft Research [51]. Torino features a series of instruction beads, which can be recognized by sight or touch. Beads represent different programming instructions such as *play, pause and repeat* that can be combined to explore computational concepts through the creation of music and stories [51]. Torino has shown great potential to stimulate collaboration between children with mixed visual abilities and, since its initial development, has been trailed in schools around the UK and India [25, 51, 67, 68].

More recent examples of computational toolkits for VI children using TUIs are StoryBlocks [33] and the music programming blocks developed by Sabuncuoglu [57]. StoryBlocks is an open-source system allowing VI novices to build audio stories by combining blocks with different features [33]. In contrast with Torino, StoryBlock's tactile blocks are passive (i.e. have no embedded electronics) and the system relies on a series of visual tags that are interpreted by a webcam mounted above the workspace [33]. On the other hand, Sabuncuoglu's music programming blocks feature NFC tags read by an Android app as the VI user scans the arranged blocks with a mobile phone [57].

Tall *ladders* that enable VI children to explore different computational concepts and practices are usually provided by an accurate scaffolding system of increasingly complex activities. New blocks are usually introduced gradually and children are invited to reflect on the implication of their use, understanding how introducing new functions and variables affects the program. Torino introduced a series of activities that are grounded on the UK and Australian primary school computing curricula [51]. Activities build on each other and the child's learning is assessed by having them explain the characteristics and functionality of the toolkit to their parents [51, 67]. Of primary importance when building ladders is the provision of appropriate training material to non-specialist teachers and parents so they can deploy activities correctly and assist children effectively when necessary [50].

Frames of interest are necessary for inclusive computational toolkits to accommodate a wide range of variation in relation to coding ability and also different levels of visual impairment (or the absence of it). Torino, music blocks and StoryBlocks have been shown to be effective and engaging for learners with different level of experience, from complete novices to children with some existing exposure to computers and programming [33, 51, 57]. Further, both Torino and StoryBlocks have been shown to enable collaborative play between learners with different visual abilities [33, 51, 67]. Frames of interest should also enable children to express their creativity and explore different possibilities without having to progress to more complex concepts if they are not ready or don't wish to do so. For example, StoryBlocks allows the learner to create custom blocks featuring new characters and actions to personalize stories [33]. Similarly, using Torino children can record their own sounds to compose new melodies and stories [51].

Finally, *reinforced corners* in inclusive computational toolkits can ensure that children receive appropriate support, while also rewarding success, reducing the possibility of errors to minimize frustration and remain motivated [1, 44]. For example, Blocks4All [46] included external elements, such as towers of wooden blocks that were knocked down as the robot completed different segments of the path, allowing children to monitor their success and provide them with a fun reward. Sabuncuoglu [57] provided children with a rack featuring 15 squares placed at regular intervals in a grid, so that children could keep track of the number and arrangement of the blocks, thus minimizing errors in positioning blocks and the resulting construction and debugging of code. Similarly Torino deliberately provided children with a limited number of beads to stimulate children to focus on conceptual complexity of the programs created rather than being hindered by issues of computational complexity that could arise when too many beads are used to create relatively simple programs [51].

Ramps, ladders, frames of interests and reinforced corners are all important to develop toolkits that can effectively support children with VI. Although some inclusive computational toolkits that embed these features have recently started to appear, children with VI have different needs and preferences that might not have been addressed by currently available solutions. To offer an alternative and complementary solution to the needs of learners with different visual abilities, we set out to design a modular and open-source toolkit that leverages the use of music and TUIs to support learning of basic computational concepts and practices.

3 DEVELOPING TIP-TOY

3.1 Initial design concept

TIP-Toy was created with an aim to be an open-source, low-cost computational toolkit, that children with mixed visual abilities could use alone or with friends, to learn basic concepts of programming while having fun. This aim meant some initial decisions were made before we engaged in the co-design process.

Our first decision was to utilize passive blocks to contain the cost, simplify the manufacturing process and enable more flexibility in shapes and sizes, inspired by prior work [13]; while active programming blocks are advantageous for interactivity as they enable direct manipulation of the beads to change parameters in the program, they are difficult to manufacture, bulkier, costlier and less robust than passive ones [41, 51]. Our second decision was to use a camera and a series of visual tags for the blocks similarly to StoryBlocks [33], as camera based systems for marker recognition are inexpensive, robust and easily implemented on mobile, online and desktop applications [14, 27]. Our final decision was to use music as a primary output for the toolkit. Music was preferred to other forms of audio output such as animal sounds or audio stories as previous studies have shown that it can be an effective tool to engage learners in computing activities, encouraging improvisation and experimentation, promoting accessible and creative learning [40, 49]. Furthermore, music involves features such as repetition, rhythm and melody which link to a variety of computational concepts and practices such as loops, parallelisms, remixing [56]. Researchers working on Torino have reported that children encounter increased difficulties composing and debugging programs with musical outputs, especially when activities become more complex or they involve the use of pauses which might be difficult to detect by listening to a melody [51, 67]. Despite these limitations, we decided to use music for the introductory activities illustrated in this publication because of their simple nature; more advanced activities might require different types of content such as stories.

3.2 Designing TIP-Toy with VI programmers

To inform the design of TIP-toy we interviewed three male VI adults who had extensive experience in training and mentoring VI novice programmers, both children and adults, and also worked as software developers and enjoyed programming for work and leisure. All three participants were involved in the design process for their knowledge of the initial difficulties that VI children face with coding, especially in light of their outreach and mentoring experiences. The interviews were semi-structured, and they explored different mechanisms for interaction, in order to identify design features for the tangible blocks and define suitable activities for subsequent sessions with children. Throughout each session, participants were presented with sets of blocks with different shapes, sizes, tactile features and interaction mechanisms. The importance of different programming concepts and practices was also discussed and participants were asked for suggestions for future developments.

Interviews were audio recorded and fully transcribed by the first author. Transcripts were analyzed by the first authors using Thematic Analysis [10] to extract design recommendations for the system (both in terms of the physical design of the blocks, and of the functionality) and for the series of programming activities for children. Themes were discussed with the other authors to ensure that design recommendations had been correctly extrapolated. Below we summarize the main insights we gained from these initial sessions.

3.2.1 Tactile features and block to block interaction. One of the first important questions was to identify a suitable way to enable children to combine instruction blocks such as play, pause and repeat with variable blocks (i.e. data containers) indicating different sounds or the number of loop iterations. Various ways to combine blocks using magnets, strings, matching concave and convex shapes or shapes that could slot into each other were presented to our participants. All of them preferred variable blocks slotting inside the command block, for several reasons. Firstly, this arrangement suggests a better conceptual representation of the relationship where the variable modifies the execution of the play, pause or repeat function. Secondly, it enables immediate tactile distinction between the smaller variable blocks and larger hollow function blocks. Finally, having blocks slot into each other would prevent them from accidentally separating, making it easier to manipulate sequences of code.

Next we had to decide how to present functionality, such as repeat loops, that could be expanded to contain a variable number of other blocks. In block-based GUIs this is usually represented by a larger block that stretches to contain instructions nested inside [24]. Stretchable elements such as coil wires or elastics were explored but these solutions were considered impractical as the stretchy element could feel either too loose or too tight. Participants suggested a

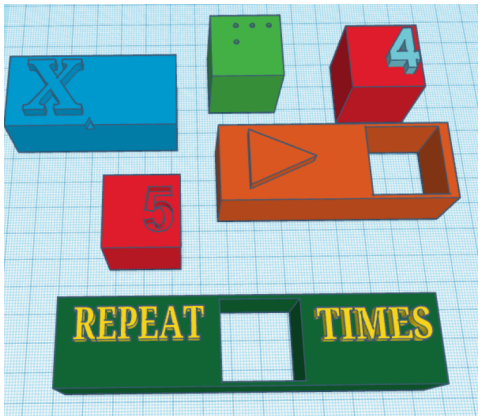

Figure 2: Design of blocks showing 3D printed features that were explored with VI developers during design consultation

simpler solution where loops have a start and an end block (end-of-the-loop statements are also used in a variety of scripting languages). It was suggested for such end blocks to be similar to the start-loop blocks to facilitate the conceptual link between the two blocks without causing confusion.

Blocks were designed with different tactile features including numbers, dots, symbols or letters to indicate the value of a variable or the meaning of a function, see Figure 2 for some examples. Symbols were unanimously preferred to letters and words, which were often found to be too difficult to read. Similarly, dots to invite counting were preferred to numbers as participants warned us that many young children who are fully blind might not be able to read numbers by tact. They also recommended spreading out dots where possible to avoid configurations that resembled braille cells, which could be confusing for some children. Embossed features were generally found easier to read compared to engraved ones, especially when counting the number of dots in a variable. As TIP-Toy was aimed at children with different visual abilities, participants suggested that we incorporate design features to support both tactile and visual learners. Specifically, it was decided that blocks with different functions would be printed in different colors and tactile features on the blocks would painted with high contrast to make them easy to recognize visually for children who had partial, or full sight.

Finally, as the orientation of the blocks was important to generate the correct code, participants suggested using a distinctive tactile feature on one of the block's corners. One of the developers suggested that this distinctive feature described as "*sort of a round bubble*" should be placed on the top of the block rather than along the edge where it might be difficult to feel when blocks were placed in a sequence.

3.2.2 Playing with others. TIP-Toy aims to be a computational toolkit inclusive of children with different visual abilities, rather than being specifically designed for fully blind children. Our expert participants recommended using different colors for different block types and high contrast to highlight the presence of dots and symbols, to make the system more usable by partially sighted children. One developer suggested that incorporating braille could support fully blind children to identify the blocks but the others felt that it could create confusion for sighted children, or the many VI children who were not braille literate. Additionally, one developer was concerned that the presence of braille could label TIP-Toy as "*another game only for blind children*", and would risk compounding isolation rather than promoting inclusion. On balance, no braille description was added to TIP-Toy blocks.

We initially chose symbols commonly found on consumer devices, such as a forward pointing triangle for 'play,' or two longitudinally placed bars for 'pause.' The interview revealed that two of our participants were not at all familiar with these symbols, as they either lost their sight at a young age or were born without. Yet, participants pointed out that it was easy to recognize the symbols, once they were explained their meaning. Further, the idea of using symbols commonly associated with the indicated functions was received positively as it would build on the affordances of the teachers, and sighted or partially sighted children, facilitating inclusion and shared play.

3.2.3 Designing the workspace. Once the features of the blocks had been defined it was important to decide on the size and characteristics of the workspace. Generally the developers felt that a toolkit which would be meant for use primarily in school could have a larger size, but at home "it would need to fit on a kid's desk... that's quite small". As the webcam (currently Logitech C920 HD) would need to be placed above the workspace and have full view of it, it was also important to control the height of the camera.

Considering these requirements we set the workspace size to that of an A3 sheet, so the block size was big enough to accommodate the presence of tactile features and symbols without cluttering them, but small enough for a child to hold one block in their hand without difficulty. The workspace can accommodate 10 blocks and the camera was fixed at an approximate height of 55cm and secured to a custom laser-cut acrylic U shaped stand. The presence of four static markers at the edges of the workspace allowed the camera to focus the field of vision and define the space where markers on the block could be translated into code.

Providing a fixed reference point can help VI individuals to create a mental map of the workspace supporting them when placing the blocks and performing debugging operations [29, 46]. For this, VI developers suggested that the start block should be fixed at the top of the page rather than allowing placement anywhere on the page so that block would always be placed with a similar layout.

Finally, to prevent the blocks from moving around on the workspace, small magnets were glued to the bottom of each block and a magnetic sheet was placed underneath the A3 sheet with the reference markers. Magnets were preferable to other proposed solutions (e.g., blue tack or a precut rack); they kept the blocks in place, but still allowed for easy movement in a group if necessary and provided pleasant haptic feedback when placing down or lifting the blocks.

3.2.4 Play, read and record. TIP-Toy featured a "Run" button to enable children to upload their code once completed; this ordered the system to translate the arranged sequence of blocks into a playable musical sequence. Participants felt that the system would

benefit from an additional button that read aloud the sequence created by the children, rather than just played. This could help children with debugging and allow them to identify any system failure (for example, if a block was placed too far away from others to be identified as part of the sequence). Moreover, hearing the pseudocode would help them to familiarize themselves with it and offer an experience similar to using a screen reader, thus supporting the transition effectively.

A second suggestion was to add another button to enable children to record their own sounds, stories or music. Although this function is important and could increase the possibilities for the children's expressivity, we could not add it in the initial iteration as it would considerably increase the system's complexity and would require further investigation on the best interaction mechanisms. A picture of the TIP-Toy setup showing all the features of the system is shown in Figure 3

3.2.5 Activities and goals for the toolkit. The final part of the interviews with VI developers was spent discussing activities for the testing session and longer term goals of TIP-Toy. For the initial design iteration it was agreed with all the participants that the aim of TIP-Toy would be to introduce four basic computational concepts (sequences, events, loops and data) and support the computational practices of being incremental and iterative; and testing and debugging.

More complex computational concepts such as parallelism, conditionals and operators and practices such as abstracting and modularizing or reusing and remixing were left out of the design of both the toolkit and the initial set of activities as they could increase the complexity of the system and they would require more practice than was feasible in a single session with children. It was also further reiterated that TIP-Toy would not be specifically targeted to visually impaired pupils or linked to classroom activities, but aim to be low-cost and usable by all students regardless of visual ability.

In the long term VI developers felt TIP-Toy should allow children to play independently, alone or with their friends, without the need for adult supervision. They all stressed that the main focus of the toolkit should be to facilitate progress beyond the use of TUIs, towards more traditional screen reader-based solutions. As observed by previous researchers due to their intrinsic nature TUIs are subject to limitations when creating programs over a certain complexity level; participants felt that rather than attempting to stretch these limits, the transition towards more flexible and advanced methods should be supported [51, 59, 68].

3.3 Final Design

The developed software and design files are available at https://bitbucket.org/ecostanza/tip-toy/. The final design of TIP-Toy incorporated the insights provided by VI developers with findings and learnings gathered from relevant literature. As mentioned before, TIP-Toy aims to provide a complementary alternative to other accessible TUI-based inclusive toolkits such as Torino [51] and Storyblocks [33] to support children and educators in finding the best solution to fit their unique learning goals. To help identify differences and similarity that might influence the suitability of different inclusive computational toolkits, in Table 1 we present a short comparison of the main features of the TIP-Toy, Torino and StoryBlocks.

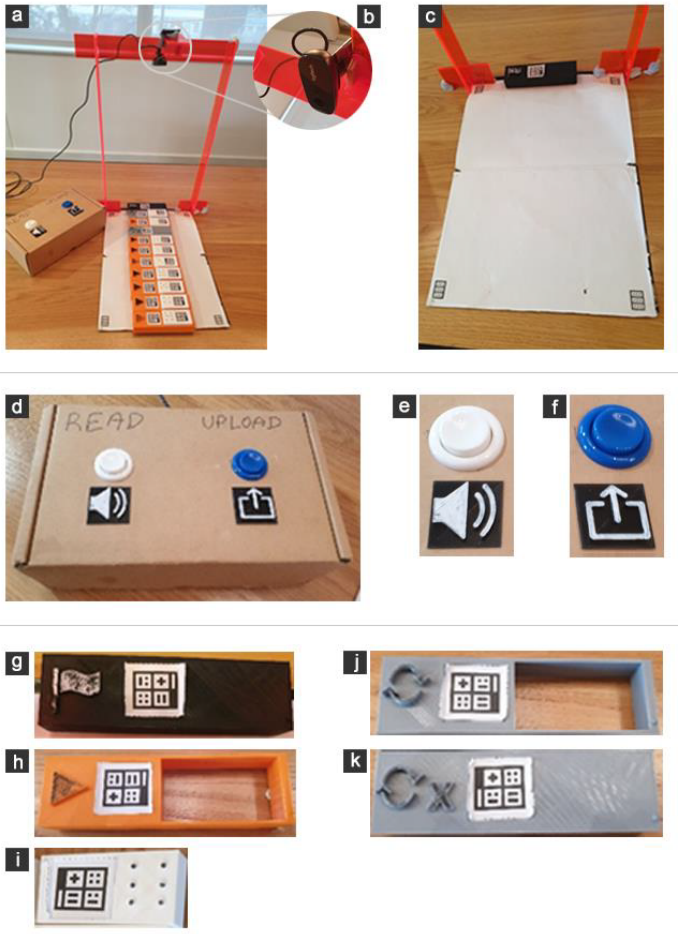

Figure 3: The setup of TIP-Toy and different features of the system: a) The complete set-up of TIP-Toy with a series of blocks positioned on the workspace; b) The webcam mounted on the custom made stand; c) The TIP-Toy workspace as seen by the webcam; d)The box featuring the "read aloud" and "upload" buttons; e) The "read aloud" button; f) The "upload" button; g) The Start block; h) A "Play" block; i) A "Variable block"; j) A "Start Loop" block; k) An "End Loop" block

4 TESTING TIP-TOY

Once we developed a working prototype of TIP-Toy according to the specifications and suggestions of VI developers, we wanted to explore its potential in supporting children with mixed visual abilities to learn basic computational concepts through music. Although we hope that TIP-Toy could appeal to both children and adolescents, our main target user group was primary school aged children with different visual abilities.

Table 1: Comparison of features: TIP-Toy, Torino and StoryBlocks

	Torino	StoryBlocks	TIP-Toy
Architecture of the system	Active programming beads with embedded electronic	Passive blocks with markers read by a webcam	Passive blocks with markers read by a webcam
Licensing, availability	Commercially available from Microsoft. Source code not available	Not available to the public	Open-source (GPL-v3)
Customization	Users can add sounds	Users can add sounds and create new blocks	Users can add sounds and create new blocks
Intended settings of use	Classroom	Classroom	Home and classroom
Target users	Children 7-11 years mixed visual abilities	Middle and high school VI users	Children 6-11 years, mixed visual abilities
Sound output	Music, stories, poetry	Stories	Music
Programming concepts supported	Sequences, loops, events, parallelism, conditionals, operators, data	Sequences, loops (limited), conditionals (limited)	Sequences, loops, events, data
Programming practices supported	Experimenting and iterating, testing and debugging, abstracting and modularizing	Experimenting and iterating, testing and debugging	Experimenting and iterating, testing and debugging

Table 2: Summary of participants in the TIP-Toy evaluation

Name	Age	Visual ability	Coding experience	Screen reader experience
Wayne	10	Sighted	Used microcontrollers at a summer school	None
Ben	10	Sighted	Used Scratch at school	None
Martin	10	Partially sighted (shapes and light)	None	Basic, learning NVDA
Donald	10	Blind	None	Basic NVDA and touch typing
Billy	9	Sighted	Scratch coding club	None
Bob	8	Sighted	Some Scratch and Raspberry Pi at home	
Harry	8	Partially sighted	None	None but can use computers with large fonts
Peter	6	Partially sighted	Scratch with support from an adult	None
Thomas	6	Sighted	Used Scratch at school	None

4.1 Participants

Nine children with mixed visual abilities took part in our exploratory evaluation; all children were male and between the age of 6 and 10 years old with visual abilities ranging from blindness to full sight. Despite our efforts to recruit girls, including using gender-neutral advertising material for the study, only boys volunteered to take part in the study. Table 2 shows an overview of the characteristics of the children. All names used in the paper are pseudonyms that the children chose to be used for the research project. All children attended mainstream schools. The aim of the study was to allow children to play with the toolkit and complete a series of activities to discover the various functionalities. As we wanted children to interact with the toolkit in a comfortable way, we offered them the opportunity to come to the session alone or with a friend. Except for the first pair of children, who were both sighted, the VI child was recruited first in the study. If the child was interested to bring a friend to the session they were reminded that their friend could either be sighted or visually impaired. All children but one, decided to bring a friend to play with; so, 8 of 9 participants, worked in pairs throughout the session, whereas one completed the session alone.

The sessions were carried out in our research laboratory and participants came to the session either accompanied by their parents or a specialist VI teacher. The study received ethics approval from the UCL Research Ethics Committee. Parents provided informed consent for children to take part in the study and for the use of video recording for research purposes; children consented to participating and video recording of sessions.

4.2 Procedure

The sessions for the exploratory evaluation lasted between 1.5 and 2 hours, depending on how long the children needed to complete the activities, and included a 15 minute resting break if necessary. Sessions were run by the first author who has previous experience

working with children with disabilities and delivering coding activities to children for outreach and education. The second author was present in all sessions to observe, take notes and provide technical support.

At the start of the session, the purpose of the research was explained to the children and accompanying adults. After acquiring informed consent background information was collected (children's visual abilities and previous experiences with computational activities) the play and variable blocks were introduced, and children were asked to try them out and describe their functionality. Children were then allowed to explore the workspace, identify the position of the camera and of the box with the "Run" and "Read Aloud" buttons, and had 5-10 minutes to familiarize themselves with the system.

After the familiarization, in the first activity children had to play a piano scale to understand the concept of sequences. Sounds were assigned to variables in the "correct order" (i.e. Variable 1 = C, Variable 2 = D...). The first activity introduced the concept of sequences and focused on the computing practice of being incremental and iterative. The concept of event was also introduced really early as children were encouraged to think about the effects of pressing the two different buttons on the outputs produced by the system. To support reflection during the activity children were asked to explain their strategies and every time they proposed a solution they were invited to explain their rationale. In the second activity the children were asked to complete a simple children's song ("Twinkle Twinkle Little Star"). For this activity, the variables contained audio snippets of the song and the order of the sounds was randomized. This strategy was used to introduce the concept of data and support practices of testing and debugging.

After completing the first two activities, the loop blocks were introduced to the children who were then asked to describe them. After explaining their use, children experimented with them freely for a few minutes. The third and fourth activity focused on learning the concept of loops and further practice computational practices of iteration, testing and debugging. In the third activity children had to recreate some simple melodies that were played to them using a sequence of piano notes. The final activity involved children playing another popular song that contained a loop ("Row Your Boat"). The researcher was available to support the children during all activities, offering help when needed.

To explore their understanding, after they completed the activities the children were asked to explain to their parents how TIP-Toy works, and the function of the various blocks and the strategies they used to complete various activities. At the end of the session the researcher asked children what they liked or disliked about the system and the activities they had tackled, if how, and where at home or in the classroom) they would like to use TIP-Toy and other activities or games they would like to play with it.

Session were video recorded, and field notes were taken by the second author. At the end of each session a short debrief helped to capture any immediate impressions or reflections among the authors. The video recordings were analyzed by the first author using thematic analysis [10] and findings were discussed and refined according to discussions with the other two authors. Thematic analysis was carried out using an inductive approach and the construction of themes was guided by our three research questions: (1) How did the characteristics of the TIP-Toy modulate how the children interacted with it and with each other? (2) How did TIP-Toy support children in building programs as they engaged with the activities? (3) Does TIP-Toy support children in gaining understanding of computational concepts and practices?

5 FINDINGS

5.1 Engaging with the toolkit

Children used different approaches to familiarize themselves with the system as they discovered the various blocks and components of the system. Some children spent more time on visual or tactile inspection of individual blocks, asking questions about features, examining symbols, establishing the size of the workspace and identifying the position of the start block before attempting to combine the blocks. For example, after placing a Play block on the workspace, Bob and Harry ensured they had the opportunity to examine all the variable blocks before proceeding to slot one in the dedicated hole; this allowed them to familiarize themselves with the blocks and to ensure they had all the blocks they needed available. Harry would bring each individual block close to his eyes to make sure that he could see the dots correctly; when looking for a specific block he leveraged Bob's sight to quickly find it "*Can you pass me the seven? She said we should have eight different ones...".* Examining the features of the blocks before placing them on the workspace helped some children to prevent mistakes; Ben was quick to catch Wayne as they started to combine blocks "*If the bubble is on that side it's not going to work, they have to stick together.*"

Other children preferred to straightaway combine blocks and construct code, figuring out how the system worked through trial and error, similar to a discovery learning approach [4]. Peter and Thomas started by laying down a few blocks on the spaces and used the buttons in the interface while listening to the output to understand how the system operated and if the blocks were laid down correctly. When Thomas pressed Run, they noticed that one of the two notes was being played. Peter asked "*Why didn't it play all? Press the other button".* Listening to the Read aloud the children figured out that the second play block was not being read. Peter repositioned the blocks to solve the problem. However, when asked if they understood why the block was not read and what changed when the blocks were replaced they were unsure. To help them understand, they were invited to place the blocks on the workspace with slight variations, purposefully introducing gaps and "wrong orientations", see Figure 4, using the Run and Read aloud buttons to understand the rules and limitations of the toolkit. This rigorous approach helped them to quickly grasp how different outputs, especially from the Read aloud button, could help to identify mistakes each time "*If there is a hole between the blocks it does not say play for the second time!*" (Thomas).

As they interacted with the system, adding notes, listening to the voice outputs and creating different melodies, the children displayed great enthusiasm and experimented and interacted with the toolkit in unanticipated ways. Before seeing the loop block, Wayne and Ben used the Run button as soon as the melody they created was finished, to create a repeat effect; Martin mixed outputs from the Run and Read aloud buttons by overlapping the voice and sound.

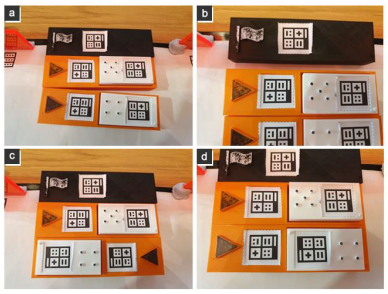

Figure 4: Different orientation mistakes shown in the collection of picture above (a) Gap between first and second play block, (b) Gap between start and play block, (c) Play block incorrect orientation, (d) Variable block incorrect orientation

He was proud of his discovery calling attention to his creative composition "*Look dad! I'm like a DJ*".

The introduction of new blocks required children to adjust their approaches. For example, all children except Bob and Harry built their first sequence outside the workspace area and then moved it under the camera only after they felt confident of how the blocks were arranged. New blocks also led to experimentation as children tried the outputs out of curiosity rather than need. For example, on completing their first successful sequence with a loop block Billy said to Donald "*Press the voice button I want to see what it says*".

5.2 Strategies for building code

Children adopted different strategies to construct their code. Martin used a structured approach, listening to the sound assigned to each variable individually before attempting to arrange them in a sequence. Bob and Harry, adopted a similar strategy but progressively allocated the blocks to the sequence as they found the ones that were meant to be played at the beginning ("*No! Don't move the first block! We need to find the bit after Twinkle now*"). Donald and Billy, and Peter and Thomas attempted to locate snippets of the songs as they listened to them, creating the proposed sequence on the side of the workspace as they discovered the sounds of more blocks. Finally, Wayne and Ben first attempted to place all the blocks on the workspace together in random order, and reconstruct the sequence through a process of "sound debugging". The first three examples showed how some children preferred to start from incremental and iterative practices, whereas Wayne and Ben relied more on testing and debugging practices from the start of the task.

Regardless of their strategy, all children encountered difficulties in completing the activities and adjusted their approach. Martin struggled to remember all the sounds associated with different variables and listened to the same blocks several times as he tried to compose the sequence in his head. In the end, he modified his strategy and started adding blocks to the sequence as the correct sounds were discovered in a similar way to Bob and Harry.

Both Donald and Billy and Peter and Thomas incorrectly positioned some blocks in the sequence. When the blocks were transferred to the workspace and the composed melodies were played, they had to sound debug their code to reconstruct the correct melody. Similarly, Bob and Harry incorrectly assumed which part of the song "Row your boat" contained the loop they had to use, leading to errors in the sequence. They requested help and to re-listened to the sounds associated with selected variables, enabling them to debug their own code and complete the activity.

Only Ben and Wayne completely abandoned their initial approach for a more iterative strategy; they found that playing the melody composed by a series of blocks arranged in random order produced a result that was far too confusing to be reversed engineered into the correct sequence. After listening to the output both children burst out laughing "*Oh my God, that was terrible!*" said Ben. "*It's too confusing*- agreed Wayne- *Maybe we just try one and see what it sounds*", leading them to use an approach similar to the one adopted by Martin for the rest of the activities.

The way in which all children ended up adjusting their approaches showed how, despite the tendency to privilege one computing practice over the other, both needed to be applied in order to succeed. Children like Martin who tried to be incremental to avoid mistakes that would need debugging found that testing combination of sounds was necessary to build correct sequences. Similarly, Ben and Wayne who attempted to rely solely on debugging practices discovered that their approach was too chaotic and incremental iterations were needed to tackle both simple and more complex tasks.

5.3 Collaborating and playing together

Children enjoyed tackling the activities and playing together with TIP-Toy. Overall, all children collaborated with their partners in completing the activities but the way they did it was qualitatively different across pairs.

Both Donald and Billy, and Bob and Harry divided tasks among themselves in a structured way, implicitly establishing roles for each other according to their different abilities and inclinations. Harry studied music and played the piano so he felt more confident than Bob in identifying notes when constructing or debugging sequences of blocks. On the other hand, Harry relied on his sight to quickly identify the blocks, arrange them on the workspace and spot errors cause by misalignment or wrong orientation of the blocks. To see if the position of the children in respect to that of the blocks and the workspace on the table had any influence on the role taken by the children we asked them to swap places at the end of each activity. However, the division of tasks remained consistent regardless of the position of the children.

Donald and Billy's different roles also remained mostly constant throughout the course of the session. Donald was a strong tactile learner, due to his braille experience, and was skilled at quickly identifying blocks by touch. He had a good musical ear and was often able to detect the pitch of the notes or the part of the song that was assigned to a particular variable ("*that is the highest note so it's the last one*" - Donald during the first activity). However, he preferred to rely on Billy for tasks requiring spatial awareness such as arranging blocks on the workspace. To facilitate collaboration and to better use Donald's help in debugging, Billy often stood up and moved the blocks sequence in front of Donald who used his hands to read the sequence asking Billy to play certain blocks so they could keep adding relevant parts to the song.

> *Donald: "What is this number two block?" Billy plays the block:*
>
> *Billy: "It says 'How I wonder' I think it goes at the end"*
>
> *Donald: "We found 'What you are' earlier, was it not number 5?"*

The other two pairs of children did not obviously assign set roles when completing the activities and were more likely to have a fluid form of collaboration taking turns to perform various tasks throughout the session. In some instances, this turn taking was quite explicit with children either asking for better access to some parts of the toolkit or offering their partner the option to take over. For example, between the first and the second activity Wayne asked Ben "*Swap with me so I can play with the buttons next*", but before the last activity he asked his friend to swap places again. During another activity, this transition was unspoken as Peter reached out across the table to grab some of the blocks on Thomas's side saying "*I know which one we need now!*".

An important part of collaboration was celebrating together when activities were completed or sharing fun discoveries made during activities or free play. When sharing success, children celebrated with each other either verbally "*We did it*" (Bob and Harry) "*Whoop!*" (Peter and Thomas) or with a high five (Wayne and Ben). To better share the satisfaction of successfully completing a task children would often press the Run button once each, to listen to the correct sequence twice. When successful, children would also draw the attention of the researcher ("*Look Miss it works now*" – Martin), their parents or teachers seeking positive feedback and praise. Interestingly, success sharing was important even when playing solo, as Martin called attention to his successes.

5.4 Understanding computational concepts and system rules

During the different activities children learned about basic computational concepts and practices, how TIP-Toy worked and its limits. For example, the concept of sequences was easily understood by all children ("*It's the list of things you want to play with the blocks*" Donald). However, children also understood the particular rules underlying the system's operation through play. Martin understood that with TIP-Toy "*You have to tell it 'Play this note and then play that note, you cannot tell it to play two notes together*". This shows that he knows that the system only allowed him to assign a single variable for each block and playing a sequence of two notes would need two different commands.

Similarly, Bob was very clear, "You have a start block at the top and then you have to put the play blocks under it to tell it what you want to play. You can't have holes or have the block the wrong way around otherwise it does not work, the two bumps go together. [. . .] You make a line of blocks and it plays them. But you also have another button so he can also read the blocks that are there". Here Bob is describing both the computational rule of sequences and the concept of events, as pressing one button would play the sequence, whereas pressing the other button would trigger the Read aloud. However, the placement of the sequence under the start block and its orientation is a system requirement linked to how the camera detects the markers on the blocks.

Through the various activities children became really familiar with the concept of variable. Unexpectedly, the toolkit also facilitate some more abstract reflections that we did not originally anticipate. When explaining the function of variable blocks Harry stated "*A variable block can be different sounds. In some of the activities they were notes but other times they were songs like in Twinkle Twinkle. . .. But three was also how many times you had to repeat Row. . .*". Harry's confusion was generated by the fact that in his initial mental model he imagined variables being assigned different sounds depending on the activity. However, in the final task, one block three was inserted within the loop to indicate the number of repetitions, whereas the other one was assigned one of the song's snippets that had to be incorporated in the sequence.

The interaction created by the blocks enabled children to easily understand the concept of loops. The presence of the two repeat blocks with the code to be repeated in the middle provided a clear metaphor allowing even the youngest children like Thomas to formulate a solid explanation: "*The loops block, you have two. One is start repeating and the second one is to tell it to stop repeating, then you have to use one of the small blocks to tell it how many time you want to repeat. . . And the play blocks you have to place them in the middle, between the start and finish*".

6 DISCUSSION

The results of our design consultation with VI developers and the testing session conducted with the children show that TIP-Toy has the potential to be effective and accessible in engaging children with mixed visual abilities in learning basic computing. These sessions also highlighted the limitations of the toolkit and helped to identify aspects to improve. Here, we elaborate on some of the insights that emerged from our studies and discuss how TIP-Toy, and other toolkits using a similar approach, could better incorporate ramps, ladders, frames of reference and reinforced corners to support children with different visual abilities.

6.1 Building ramps

Overall, TIP-Toy's accessibility was found to be good as all children, regardless of their visual abilities, could successfully interact with it, recognize blocks using either touch or sight, combine blocks to create musical sequences and listen to the outputs of the programs they built as part of the activities. However, throughout the sessions we noticed how some children experienced difficulties when

interacting with the system due to design features that did not fully cater to their needs and abilities.

Although all children found manipulating the blocks and assembling them into musical sequences easy, Donald, Harry and to a lesser extent Martin found it challenging to place the blocks correctly on the workspace. Over time Martin seemed to find the task easier as he progressed, instinctively reaching for the start block fixed at the top of the workspace. However, Donald and Harry relied on their sighted peers to place the blocks on the workspace as they could more easily locate the boundaries of the workspace and the location of the blocks already on it. The workspace area for TIP-Toy could be quite easily explored by children at the beginning of the session but the tactile exploration of a space that had poorly defined boundaries presented a challenge, especially as children were moving around the table and swapping places with each other to better access different areas. Similar issues were reported in other studies where it was observed that individuals with VI can find it difficult to locate objects or orient them in relation to each other in table top TUI toolkits [3, 42]. To facilitate identifying the boundaries of the workspace we'll design a new base with tactile borders similar to StoryBlocks [33]. Additionally, future iterations of the TIP-Toy could include accessories such as side guards, similar to rail bumpers in bowling alleys, to help children place blocks directly under the start and make it easier to achieve correct alignment.

Similarly, children with VI only interacted with the box featuring the Run and Read aloud buttons when it was located on their side of the table as the box was not fixed to the workspace and could be difficult to locate. The initial decision to not attach the box to a specific side of the workspace was made to allow children to move around freely and to arrange the elements of the toolkits based on the space they had available. To address the difficulty in locating the box, without restricting the position of the box to one side we'll design an attachment system that enables children to attach the box to the edge of the workspace on a preferred side, enabling them to create a fixed location for the buttons' box without constraining it to a fixed unsuitable location.

Finally, although tactilely recognizing the blocks individually was very easy for all children, locating a specific block from the pile was more challenging. When programming activities with TUIs are carried out in a setting involving teachers or other sighted adults, they often organize and sort the blocks, so that students can focus on the creative process [33]. However, when children are playing independently, it might be necessary to provide a storage box with dedicated compartments to help them locate spare components more easily [67].

6.2 Providing ladders

Similar to the design trade-offs in Torino [51], for the initial version of TIP-Toy we sacrificed extensibility in favor of simplicity and accessibility. In accordance with the suggestions of VI developers, we provided a limited number of blocks and a workspace that could not host more than 10 blocks at any time. The limited variety of blocks helped most children to quickly grasp the function of each type of block. Further, having shorter sequences of blocks made debugging more manageable allowing them have oversight of the program through touch, sight or audio. But some of the older and more experienced children expressed the desire for more complexity, like having a larger variety of blocks for different functions or the possibility to create longer and more complex musical sequences.

Increasing the number and types of blocks could allow children to create more complex programs and discover new computational concepts. But a large variety of blocks could overwhelm younger children and novices. A possible solution could be to provide children with an initial core set of blocks that could be expanded once basic functionalities have been mastered [33]. The open source nature of TIP-Toy could enable teachers and parents to create new blocks to support the expansion of the toolkit. However, contributing to open source projects can be difficult for newcomers, even when they have the right experience [64]. Thus, support is needed to enable novices, parents and teachers to contribute to further developments.

Although increasing TIP-Toy's complexity is possible to some extent, TUIs have objective limitations due to their physical characteristics that cannot be overcome [51, 59, 68]. VI developers engaged in the initial design consultation suggested that a better way to support children in engaging with more complex computational concepts and practices should be to facilitate the transition towards audio interfaces and text-based programming. In recent years, some researchers have focused on hybrid programming environments to facilitate transitions from block-based to text based programming for sighted children [71]. However, to our knowledge there has been no attempt to facilitate the transition between TUIs and text-based programming for VI children. To this end, some elements of TIP-toy could be leveraged to create the hierarchical organization of code and allow children to use the keyboard to enter simple commands, thus providing a simplified, constrained representation of the program structure. This would be a similar strategy to that offered by the Tangible Desktop for more basic navigation tasks [5].

6.3 Constructing frames of interest

Throughout the testing session we observed that all children, regardless of visual ability, age and previous level of coding experience were able to successfully engage and have fun with TIP-Toy. Children were able to play with TIP-Toy either alone or with their peers, highlighting how the toolkit could be potentially used both at home and in a classroom setting. It is worth noticing that all participants in the study, both children and adults, were male. Although this is a known issue in studies focusing on computing education [36], we acknowledge that this is a limitation of our study and we plan to expand our testing to include more female participants. We think this is particularly important in relation to the development of appropriate frames of interest as girls and boys have been shown to engage differently with TUIs express their creativity through different projects [18, 72].

Currently, TIP-Toy offers limited possibilities for enabling expressivity and lateral exploration but, due to its open nature, children could be supported in exploring their creativity in many ways without increasing the required level of expertise. Both VI developers and children suggested that a powerful way to do this would be to enable users to record their own sounds or melodies. While TIP-toy already offers the possibility to assign new sounds to variables, this

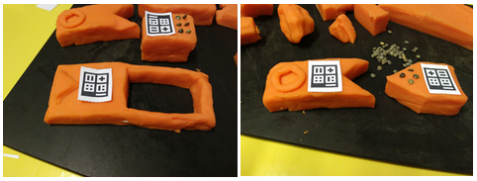

Figure 5: Two examples of salt dough "homemade" TIP-Toy blocks featuring different geometries

can only be done manually via the toolkit's software, which requires basic programming knowledge. Adding a third Record button on the interface could allow children to record their favorite sounds for one variable at the time, creating a custom set of variables for use in new stories and songs [33, 51]. We are also interested in investigating how making and crafting the toolkits could support a more engaging learning experience. For example, the children with special educational needs who took part in the study evaluating the Magic Cube [36] spent part of the initial session building and exploring the toolkit, without engaging in any programming activity. In contrast, toolkits such as Cubetto or Dr Wagon allow children to personalize their robots with craft material [73].

The tactile blocks used for our study were 3D printed in our laboratory using standard PLA FDM printers and their .stl files are available on the online repository, so anyone with a 3D printer can modify and print them. As many teachers and parents of VI children might lack access to the tools or the skills required to print existing blocks or create new ones, we explore fabrication of the blocks from easily available materials. This was made possible by the fact that the material and shape of the blocks do not influence the working of the system as long as the reciprocal distances between the markers are respected. Combining craft and computing is an effective way to increase engagement and improve learning experiences; it would help to reduce the cost and increase the accessibility of the toolkit [43, 53, 60]. To explore possible alternatives we fabricated some handmade blocks using salt dough and lentils and tested that they worked correctly with the system. Beyond trialing different materials, we also experimented with different block geometries which might be easier to reproduce by hand (Figure 5). At the end of the testing sessions we asked children, parents and teachers for their opinions on the crafted blocks and if they found the idea of making their own TIP-Toy blocks appealing. As a result of their suggestions, we are planning workshops to develop a crafting guide for TIP-Toy blocks so children can make their own toolkits at home or in the classroom.

6.4 Reinforcing corners

The design of TIP-Toy already includes reinforced corners for offering additional support to children. For example, the Read aloud button offers a second feedback modality to support debugging operations. Throughout the sessions some children used the Read aloud button to spot mistakes in their own code, and to identify systems errors due to misalignment or incorrect orientation of the blocks. We are conscious that TIP-Toy's reinforced corners could be improved and design improvements discussed above, such as rail bumpers to facilitate block positioning, would help children who need additional support.

Throughout the sessions we observed how sharing and celebrating success were an important part of the experience for all the children. Children understood when they had successfully completed the session's activities either because they were already familiar with the songs used or because they were able to compare their output with the required one. Yet, if children had to compose an unfamiliar musical sequence, the system should be able to recognize when the correct answer is given and provide positive feedback to reward their success [46]. Finally, one of the aims of TIP-Toy is to support children with mixed visual abilities to explore computing concepts and practices while playing independently. Longer studies should look at how the current design supports independently play, especially when one or both children are visually impaired. Particular focus should be placed not only to enable children to complete a specific activity, but also to be able to transition between different activities select tasks that are appropriate to their age and computing abilities.

7 CONCLUSION

It is crucially important to enable all children, regardless of visual ability, to learn basic computational concepts and practices. In this paper we presented TIP-Toy, a tactile and inclusive open-source programming toolkit that allows children with different visual abilities to combine plastic blocks to learn about computational topics through music. Initial design iterations with both adult VI developers and children with mixed visual abilities show that TIP-Toy has the potential to create engaging and effective learning experiences in an inclusive manner.

ACKNOWLEDGMENTS

We would like extend our deepest gratitude to all participants for their time, enthusiasm and support. None of the research presented in this paper would have been possible without them. Data access: the video recording data that this paper is based on cannot be shared to protect the privacy of our minor participants. The TIP-Toy prototype is available as open source software at https://bitbucket.org/ecostanza/tip-toy/

REFERENCES

[1] Meryl Alper, Juan Pablo Hourcade, and Shuli Gilutz. 2012. Adding reinforced corners: designing interactive technologies for children with disabilities. *Interactions* 19, 6: 72–75. https://doi.org/10.1145/2377783.2377798

[2] Jack Atherton and Paulo Blikstein. 2017. Sonification Blocks: A Block-Based Programming Environment For Embodied Data Sonification. In *Proceedings of the 2017 Conference on Interaction Design and Children* (IDC '17), 733–736. https://doi.org/10.1145/3078072.3091992

[3] Mauro Ávila-Soto, Elba Valderrama-Bahamóndez, and Albrecht Schmidt. 2017. TanMath: A Tangible Math Application to support children with visual impairment to learn basic Arithmetic. In *Proceedings of the 10th International Conference*

on PErvasive Technologies Related to Assistive Environments (PETRA '17), 244–245. https://doi.org/10.1145/3056540.3064964

[4] Doug Baldwin. 1996. Discovery learning in computer science. In *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education* (SIGCSE '96), 222–226. https://doi.org/10.1145/236452.236544

[5] Mark S. Baldwin, Gillian R. Hayes, Oliver L. Haimson, Jennifer Mankoff, and Scott E. Hudson. 2017. The Tangible Desktop: A Multimodal Approach to Nonvisual Computing. *ACM Transactions on Accessible Computing* 10, 3: 9:1–9:28. https://doi.org/10.1145/3075222

[6] Rahul Banerjee, Jason Yip, Kung Jin Lee, and Zoran Popović. 2016. Empowering Children To Rapidly Author Games and Animations Without Writing Code. In *Proceedings of the The 15th International Conference on Interaction Design and Children* (IDC '16), 230–237. https://doi.org/10.1145/2930674.2930688

[7] David Bau, D. Anthony Bau, Mathew Dawson, and C. Sydney Pickens. 2015. Pencil code: block code for a text world. In *Proceedings of the 14th International Conference on Interaction Design and Children* (IDC '15), 445–448. https://doi.org/10.1145/2771839.2771875

[8] Cynthia L. Bennett and Daniela K. Rosner. 2019. The Promise of Empathy: Design, Disability, and Knowing the" Other". In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–13.

[9] Marie Bodén, Bianca Pretorius, Ben Matthews, and Stephen Viller. 2018. DBugs: large-scale artefacts for collaborative computer programming. In *Proceedings of the 17th ACM Conference on Interaction Design and Children* (IDC '18), 545–550. https://doi.org/10.1145/3202185.3210773

[10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2: 77–101. https://doi.org/10.1191/1478088706qp063oa

[11] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, 25.

[12] Kunal Chawla, Megan Chiou, Alfredo Sandes, and Paulo Blikstein. 2013. Dr. Wagon: a "stretchable" toolkit for tangible computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (IDC '13), 561–564. https://doi.org/10.1145/2485760.2485865

[13] Enrico Costanza, Matteo Giaccone, Olivier Kueng, Simon Shelley, and Jeffrey Huang. 2010. Ubicomp to the masses: a large-scale study of two tangible interfaces for download. In *Proceedings of the 12th ACM international conference on Ubiquitous computing* (UbiComp '10), 173–182. https://doi.org/10.1145/1864349.1864388

[14] Enrico Costanza and Jeffrey Huang. 2009. Designable visual markers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), 1879–1888. https://doi.org/10.1145/1518701.1518990

[15] Paul Dourish. 2004. *Where the action is: the foundations of embodied interaction*. MIT press.

[16] Louise P. Flannery, Brian Silverman, Elizabeth R. Kazakoff, Marina Umaschi Bers, Paula Bontá, and Mitchel Resnick. 2013. Designing ScratchJr: support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (IDC '13), 1–10. https://doi.org/10.1145/2485760.2485785

[17] Neil Fraser. 2015. Ten things we've learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, 49–50. https://doi.org/10.1109/BLOCKS.2015.7369000

[18] Alexandra Funke and Katharina Geldreich. 2017. Gender Differences in Scratch Programs of Primary School Children. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (WiPSCE '17), 57–64. https://doi.org/10.1145/3137065.3137067

[19] Francisco J. Gutierrez, Jocelyn Simmonds, Cecilia Casanova, Cecilia Sotomayor, and Nancy Hitschfeld. 2018. Coding or Hacking? Exploring Inaccurate Views on Computing and Computer Scientists among K-6 Learners in Chile. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (SIGCSE '18), 993–998. https://doi.org/10.1145/3159450.3159598

[20] Alexandria K. Hansen, Hilary A. Dwyer, Charlotte Hill, Ashley Iveland, Timothy Martinez, Danielle Harlow, and Diana Franklin. 2015. Interactive design by children: a construct map for programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (IDC '15), 267–270. https://doi.org/10.1145/2771839.2771893

[21] Alexandria K. Hansen, Hilary A. Dwyer, Ashley Iveland, Mia Talesfore, Lacy Wright, Danielle B. Harlow, and Diana Franklin. 2017. Assessing Children's Understanding of the Work of Computer Scientists: The Draw-a-Computer-Scientist Test. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17), 279–284. https://doi.org/10.1145/3017680.3017769

[22] Brian Harvey and Jens Mönig. 2010. Bringing "no ceiling" to Scratch: Can one language serve kids and computer scientists. *Proc. Constructionism*: 1–10.

[23] Charlotte Hill, Hilary A. Dwyer, Tim Martinez, Danielle Harlow, and Diana Franklin. 2015. Floors and Flexibility: Designing a Programming Environment for 4th-6th Grade Classrooms. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (SIGCSE '15), 546–551. https://doi.org/10.1145/2676723.2677275

[24] Michael S. Horn and Robert J. K. Jacob. 2007. Tangible programming in the classroom with tern. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '07), 1965–1970. https://doi.org/10.1145/1240866.1240933

[25] Gesu India, Geetha Ramakrishna, Jyoti Bisht, and Manohar Swaminathan. 2019. Computational Thinking as Play: Experiences of Children who are Blind or Low Vision in India. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '19), 519–522. https://doi.org/10.1145/3308561.3354608

[26] Yasmin B. Kafai and Quinn Burke. 2014. *Connected Code: Why Children Need to Learn Programming*. MIT Press.

[27] Martin Kaltenbrunner and Ross Bencina. 2007. reacTIVision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (TEI '07), 69–74. https://doi.org/10.1145/1226969.1226983

[28] Shaun K. Kane and Jeffrey P. Bigham. 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), 247–252. https://doi.org/10.1145/2538862.2538975

[29] Shaun K. Kane, Jacob O. Wobbrock, and Richard E. Ladner. 2011. Usable gestures for blind people: understanding preference and performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11), 413–422. https://doi.org/10.1145/1978942.1979001

[30] Majeed Kazemitabaar, Jason McPeak, Alexander Jiao, Liang He, Thomas Outing, and Jon E. Froehlich. 2017. MakerWear: A Tangible Approach to Interactive Wearable Creation for Children. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17), 133–145. https://doi.org/10.1145/3025453.3025887

[31] Michael Kölling, Neil C. C. Brown, and Amjad Altadmri. 2015. Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (WiPSCE '15), 29–38. https://doi.org/10.1145/2818314.2818331

[32] Chonnuttida Koracharkornradt. 2017. Tuk Tuk: A Block-Based Programming Game. In *Proceedings of the 2017 Conference on Interaction Design and Children* (IDC '17), 725–728. https://doi.org/10.1145/3078072.3091990

[33] Varsha Koushik, Darren Guinness, and Shaun K. Kane. 2019. StoryBlocks: A Tangible Programming Game To Create Accessible Audio Stories. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI '19), 1–12. https://doi.org/10.1145/3290605.3300722

[34] Varsha Koushik and Clayton Lewis. 2016. An Accessible Blocks Language: Work in Progress. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '16), 317–318. https://doi.org/10.1145/2982142.2982150

[35] Pamela B. Lawhead, Michaele E. Duncan, Constance G. Bland, Michael Goldweber, Madeleine Schep, David J. Barnes, and Ralph G. Hollingsworth. 2002. A road map for teaching introductory programming using LEGO©mindstorms robots. *ACM SIGCSE Bulletin* 35, 2: 191–201. https://doi.org/10.1145/782941.783002

[36] Zuzanna Lechelt, Yvonne Rogers, Nicola Yuill, Lena Nagl, Grazia Ragone, and Nicolai Marquardt. 2018. Inclusive Computing in Special Needs Classrooms: Designing for All. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18), 1–12. https://doi.org/10.1145/3173574.3174091

[37] Stephanie Ludi. 2015. Position paper: Towards making block-based programming accessible for blind users. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, 67–69. https://doi.org/10.1109/BLOCKS.2015.7369005

[38] Stephanie Ludi, Mohammed Abadi, Yuji Fujiki, Priya Sankaran, and Spencer Herzberg. 2010. JBrick: accessible lego mindstorm programming tool for users who are visually impaired. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility* (ASSETS '10), 271–272. https://doi.org/10.1145/1878803.1878866

[39] Stephanie Ludi, Jamie Simpson, and Wil Merchant. 2016. Exploration of the Use of Auditory Cues in Code Comprehension and Navigation for Individuals with Visual Impairments in a Visual Programming Environment. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '16), 279–280. https://doi.org/10.1145/2982142.2982206

[40] Brian Magerko, Jason Freeman, Tom McKlin, Scott McCoid, Tom Jenkins, and Elise Livingston. 2013. Tackling engagement in computing with computational music remixing. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13), 657–662. https://doi.org/10.1145/2445196.2445390

[41] Mark Marshall, Thomas Carter, Jason Alexander, and Sriram Subramanian. 2012. Ultra-tangibles: creating movable tangible objects on interactive tables. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 2185–2188. https://doi.org/10.1145/2207676.2208370

[42] David McGookin, Euan Robertson, and Stephen Brewster. 2010. Clutching at straws: using tangible interaction to provide non-visual access to graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10), 1715–1724. https://doi.org/10.1145/1753326.1753583

[43] David A. Mellis, Sam Jacoby, Leah Buechley, Hannah Perner-Wilson, and Jie Qi. 2013. Microcontrollers as material: crafting circuits with paper, conductive ink,

electronic components, and an "untoolkit." In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction* (TEI '13), 83–90. https://doi.org/10.1145/2460625.2460638

[44] Oussama Metatla, Marcos Serrano, Christophe Jouffrais, Anja Thieme, Shaun Kane, Stacy Branham, Émeline Brulé, and Cynthia L. Bennett. 2018. Inclusive Education Technologies: Emerging Opportunities for People with Visual Impairments. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI EA '18), 1–8. https://doi.org/10.1145/3170427.3170633

[45] Lauren R. Milne. 2017. Blocks4All: making block programming languages accessible for blind children. *ACM SIGACCESS Accessibility and Computing*, 117: 26–29. https://doi.org/10.1145/3051519.3051525

[46] Lauren R. Milne and Richard E. Ladner. 2018. Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18), 1–10. https://doi.org/10.1145/3173574.3173643

[47] Lauren R. Milne and Richard E. Ladner. 2019. Blocks4All: Making Blocks-Based Programming Languages Accessible for Children with Visual Impairments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (SIGCSE '19), 1290. https://doi.org/10.1145/3287324.3293755

[48] Kritphong Mongkhonvanit, Claire Jia Yi Zau, Chris Proctor, and Paulo Blikstein. 2018. Testudinata: a tangible interface for exploring functional programming. In *Proceedings of the 17th ACM Conference on Interaction Design and Children* (IDC '18), 493–496. https://doi.org/10.1145/3202185.3210762

[49] Calkin Suero Montero and Kaisa Pihlainen. 2017. Let's play! music improvisation as a medium to facilitate computational thinking. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research* (Koli Calling '17), 199–200. https://doi.org/10.1145/3141880.3141910

[50] Cecily Morrison, Nicolas Villar, Alex Hadwen-Bennett, Tim Regan, Daniel Cletheroe, Anja Thieme, and Sue Sentance. 2019. Physical Programming for Blind and Low Vision Children at Scale. *Human–Computer Interaction* 0, 0: 1–35. https://doi.org/10.1080/07370024.2019.1621175

[51] Cecily Morrison, Nicolas Villar, Anja Thieme, Zahra Ashktorab, Eloise Taysom, Oscar Salandin, Daniel Cletheroe, Greg Saul, Alan F. Blackwell, Darren Edge, Martin Grayson, and Haiyan Zhang. 2018. Torino: A Tangible Programming Language Inclusive of Children with Visual Disabilities. *Human–Computer Interaction* 35, 3: 191–239. https://doi.org/10.1080/07370024.2018.1512413

[52] Seymour Papert. 1990. *Children, computers and powerful ideas*. New York: Basic Books.

[53] Hannah Hannah Rosamonde Perner-Wilson. 2011. A kit-of-no-parts. Massachusetts Institute of Technology.

[54] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: programming for all. *Communications of the ACM* 52, 11: 60–67. https://doi.org/10.1145/1592761.1592779

[55] Mitchel Resnick and Brian Silverman. 2005. Some reflections on designing construction kits for kids. In *Proceedings of the 2005 conference on Interaction design and children* (IDC '05), 117–122. https://doi.org/10.1145/1109540.1109556

[56] Alex Ruthmann, Jesse M. Heines, Gena R. Greher, Paul Laidler, and Charles Saulters. 2010. Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (SIGCSE '10), 351–355. https://doi.org/10.1145/1734263.1734384

[57] Alpay Sabuncuoglu. 2020. Tangible Music Programming Blocks for Visually Impaired Children. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction* (TEI '20), 423–429. https://doi.org/10.1145/3374920.3374939

[58] Jaime Sánchez and Fernando Aguayo. 2005. Blind learners programming through audio. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05), 1769–1772. https://doi.org/10.1145/1056808.1057018

[59] Theodosios Sapounidis and Stavros Demetriadis. 2013. Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. *Personal and Ubiquitous Computing* 17, 8: 1775–1786. https://doi.org/10.1007/s00779-013-0641-7

[60] Sarah Schoemann and Michael Nitsche. 2017. Needle as Input: Exploring Practice and Materiality When Crafting Becomes Computing. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction* (TEI '17), 299–308. https://doi.org/10.1145/3024969.3024999

[61] Eric Schweikardt and Mark D. Gross. 2006. roBlocks: a robotic construction kit for mathematics and science education. In *Proceedings of the 8th international conference on Multimodal interfaces* (ICMI '06), 72–75. https://doi.org/10.1145/1180995.1181010

[62] Yasaman S. Sefidgar, Prerna Agarwal, and Maya Cakmak. 2017. Situated Tangible Robot Programming. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction* (HRI '17), 473–482. https://doi.org/10.1145/2909824.3020240

[63] Andy Smith, Bradford Mott, Sandra Taylor, Aleata Hubbard Cheuoua, James Minogue, Kevin Oliver, and Cathy Ringstaff. 2020. Designing Block-Based Programming Language Features to Support Upper Elementary Students in Creating Interactive Science Narratives. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (SIGCSE '20), 1327. https://doi.org/10.1145/3328778.3372653

[64] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming open source project entry barriers with a portal for newcomers. In *Proceedings of the 38th International Conference on Software Engineering* (ICSE '16), 273–284. https://doi.org/10.1145/2884781.2884806

[65] Amanda Sullivan, Mollie Elkin, and Marina Umaschi Bers. 2015. KIBO robot demo: engaging young children in programming and engineering. In *Proceedings of the 14th International Conference on Interaction Design and Children* (IDC '15), 418–421. https://doi.org/10.1145/2771839.2771868

[66] Hideyuki Suzuki and Hiroshi Kato. 1995. Interaction-level support for collaborative learning: AlgoBlock—an open programming language. In *The first international conference on Computer support for collaborative learning*, 349–355.

[67] Anja Thieme, Cecily Morrison, Nicolas Villar, Martin Grayson, and Siân Lindley. 2017. Enabling Collaboration in Learning Computer Programing Inclusive of Children with Vision Impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (DIS '17), 739–752. https://doi.org/10.1145/3064663.3064689

[68] Nicolas Villar, Cecily Morrison, Daniel Cletheroe, Tim Regan, Anja Thieme, and Greg Saul. 2019. Physical Programming for Blind and Low Vision Children at Scale. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI EA '19), 1–4. https://doi.org/10.1145/3290607.3313241

[69] Danli Wang, Yang Zhang, Tianyuan Gu, Liang He, and Hongan Wang. 2012. E-Block: a tangible programming tool for children. In *Adjunct proceedings of the 25th annual ACM symposium on User interface software and technology*, 71–72.

[70] David Weintrop and Uri Wilensky. 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* (IDC '15), 199–208. https://doi.org/10.1145/2771839.2771860

[71] David Weintrop and Uri Wilensky. 2017. Between a Block and a Typeface: Designing and Evaluating Hybrid Programming Environments. In *Proceedings of the 2017 Conference on Interaction Design and Children* (IDC '17), 183–192. https://doi.org/10.1145/3078072.3079715

[72] Lesley Xie, Alissa N. Antle, and Nima Motamedi. 2008. Are tangibles more fun? comparing children's enjoyment and engagement using physical, graphical and tangible user interfaces. In *Proceedings of the 2nd international conference on Tangible and embedded interaction* (TEI '08), 191–198. https://doi.org/10.1145/1347390.1347433

[73] Junnan Yu and Ricarose Roque. 2018. A survey of computational kits for young children. In *Proceedings of the 17th ACM Conference on Interaction Design and Children* (IDC '18), 289–299. https://doi.org/10.1145/3202185.3202738

[74] Meet Cubetto - Primo Toys Cubetto: A toy robot teaching kids code & computer programming. *Primo Toys*. Retrieved April 24, 2020 from https://www.primotoys.com/