

Tangible Music Programming Blocks for Visually Impaired Children

Alpay Sabuncuoglu

asabuncuoglu13@ku.edu.tr

Koc University Intelligent User Interface Lab

Twin Science and Robotics



Figure 1: Music blocks in action: A shot from workshop to test our tangible coding platform.

ABSTRACT

Programming can benefit children on learning science, math, and creative thinking, and has become a part of the primary school curriculum. However, programming tools for visually impaired children are still scarce. We developed an affordable and accessible tangible music platform for visually impaired children that aims to teach the basics of programming through music creation. By ordering the tangible blocks in an algorithmic structure, the children can create a melody. The physical and conceptual design of the system was developed with the help of visually impaired developers. We conducted

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
TEI '20, February 9–12, 2020, Sydney, NSW, Australia

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6107-1/20/02...\$15.00

<https://doi.org/10.1145/3374920.3374939>

a user study with fourteen visually impaired middle school children to observe their interactions with the prototype. In this paper, we present our design, provide several TUI design considerations for students with low to zero sight, and discuss the results of our user study and future directions.

CCS CONCEPTS

• **Human-centered computing** → *Empirical studies in accessibility*; • **Social and professional topics** → *K-12 education*; • **Hardware** → *Tactile and hand-based interfaces*.

KEYWORDS

Tangible programming blocks, Music programming, Visually-impaired student education

ACM Reference Format:

Alpay Sabuncuoglu. 2020. Tangible Music Programming Blocks for Visually Impaired Children. In *Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '20)*, February 9–12, 2020, Sydney, NSW, Australia. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3374920.3374939>

1 INTRODUCTION

Programming is integrated into the primary education curriculum as it becomes a crucial 21st century need for various employment fields [8]. Hence, through programming, pupils acquire many vital skills such as creative and computational thinking, gain hands-on abilities such as building electronics, animations and integrate technology into daily life. However, commonly used programming tools depend on visual elements, which makes them rather inaccessible for visually impaired children. This can lead to an inequality in the development of the aforementioned skills for the visually impaired community. To that end, we developed an affordable tangible programming tool for visually impaired children to create music with algorithms.

Most popular programming platforms for early education, such as Scratch and Alice, use drag and drop blocks in a visual interface to create programs [5, 12]. Similarly, electronic development boards such as Arduino and Micro:bit are also coded with drag and drop blocks or writing text scripts [2, 9]. In this regard, all these popular tools highly depend on visual graphics and are inaccessible for students with vision impairment.

Current research on early programming education for visually impaired students is limited, but the interest grows due to the prevalence of programming in the primary school curriculum. Project Torino [13] and Blocks4All [11] develop accessible platforms to introduce programming in early education. With these platforms, children create stories or code robot movements. However, these platforms rely on costly electronic parts, which lowers their accessibility in terms of affordability.

In light of these, we developed an affordable tangible programming tool to create music with algorithms. Our block set consists of twenty-five coding blocks varying in levels of perceptual information (shape and surface detail). These design decisions were made with the guidance of visually impaired developers. The music blocks only require NFC stickers for fast and reliable recognition, and we synthesize the music on Android phones. We chose music creation as the subject because it is one of the most preferred and accessible creative subjects for the visually impaired community.

In this paper, we share our research process; (1) workshop with visually impaired developers to derive design considerations for tangible user interface for children with visual impairment, (2) development of the conceptual prototype, and (3) user study with 14 visually impaired children ($Age=12.4$). We discuss our observations from the user study and the next step for the system.

We contribute to the literature 1) as the first algorithmic style- tangible music creation platform for visually impaired children and 2) provide several TUI design considerations

for students with low to zero sight. Our research can inspire researchers aiming to develop tools for the visually impaired community.

2 THEORY AND RELATED WORK

In this section, we give background of programming education and tangible programming tools. We keep the focus on theory and related work about visually impaired students.

Programming Education

According to the 2018 estimates by the World Health Organization, 217 million people have moderate to severe vision impairment, and 36 million people are blind [3]. Stackoverflow's recent survey reports 1.5% of the developer community is visually impaired [16]. Developers with a severe visual disability use high contrast colors, screen magnifiers and readers to write scripts on development environments [15]. Additionally, projects like CodeTalks explore potential applications for integrated development environment (IDE) accessibility [10].

Visually impaired students usually learn to program via text-based coding with the help of a mentor. If the students struggle with these kinds of environments, their motivation levels are likely to drop. Normally, to maintain motivation, teachers make use of children's interests, such as creating games and animations in Scratch, but these platforms depend on graphics.

To lower the barrier for coding education, Blocks4All makes block programming accessible via new touch screen interactions and tactile-robot movement outputs. Since most of the visually impaired children can use touch screen interaction, interfaces that allow a similar interaction enables them to code rapidly and easily. However, Blocks4All's touch interaction is limited for the long and nested codes and the platform requires a robot to code, which makes it costly.

Microsoft's Project Torino allow children to create some stories and add music with fundamental programming elements. Their project also serves one of the most detailed research for introductory inclusive programming. However, developing such an interface that relies on many electronic parts is still an expensive solution for most of the families.

Tangible Programming Tools

Tangible User Interfaces (TUI) allow hands-on interaction and control over digital features with physical artifacts [7]. Most technologies developed for visually impaired students rely on audio and touch interaction. On the other hand, the literature points at the various benefits of tangible interaction on learning.

Using tangible elements at the classroom keeps students physically active, supports collaboration and development of motor skills and spatial understanding [17]. Zuckerman

et.al. list the advantages of tangible interfaces as a teaching tool for abstract problem domains under three categories: (1) Tangibles provide the natural environment for learning with activating multiple senses such as touch, sound or vision. (2) They offer a great variety of interaction, so increase the accessibility for mental and physical disabilities. (3) Offering a multi-hand interface facilitates natural group interaction, and increase collaboration. These advantages led to the development of various tangible programming tools to teach computational thinking [6][18].

Moreover, tangible technology supports collaboration between people of different abilities, alluding to the consequent learning that can then take place [4]. Therefore physical computing is especially desirable between children with mixed visual abilities.

3 DESIGN OF THE MUSIC BLOCKS

Workshop with Visually Impaired Developers

This step of our research was formulated to identify visually impaired children's needs in a learning environment and deriving design considerations. We conducted a workshop session with two visually impaired developers. These developers work as a team for a children-oriented technology company and have been coding for more than two years. They work with visually impaired children to teach coding and know about the obstacles they encounter.

In the session, we went through the elements of Tangible Learning Design Framework. In this framework, Antle and Wise provide an extensive guideline based on theories of cognition [1]. The principles allowed us to distribute the roles of tangibles to control different outputs, set up actions and decide on informational relations. Upon this introduction, the developers and authors started to generate design considerations.

The TUI design considerations we formulated for visually impaired children are as follows:

- (1) *Avoid the extensive use of Braille alphabet.* Currently, the visually impaired developers teach children programming with electronic blocks that have Braille code stickers. However, the developers stated that the students spend a lot of time trying to read and remember the codes. So, in our design, we only used Braille alphabet for communicating the musical notes (Table 1).
- (2) *Do not use various feedback modalities for the recognition of the blocks.* Using several feedback types (i.e. audio, haptics) to help children recognize the function of the blocks might overwhelm and confuse the students. We decided to use only shapes as the distinctive feature between the blocks (Table 1). Yet, we created an extra *info* block to identify the blocks with audio,

as some students might be more used to having audio feedback.

- (3) *Use abstract and basic shapes rather than detailed icons.* Shape differentiation provides only one level of information (i.e. family a, family b). If the tangibles need to be more informative (i.e. family b, member x) using the surface information is available. However, surface engravings need to be easily discernible by touch and conceptually understandable at the same time. For example, a piano icon can be engraved on a tangibles surface, but the students might not know what a piano looks like, or might need extra time to comprehend the engraving itself. Therefore, we adopted simple icons and explored surface information alternatively (Table 1).

Conceptual Prototype

With the principles of tangible learning framework and the guidance of the first design session, we designed twenty-five tangible block pieces and developed an Android application. The blocks allow students to design an algorithmic melody with different octave, rhythm, and sound.

Form & Material: Our block set consists of five interface (circular blocks) and twenty coding blocks that contain two levels of perceptual information which are shape and surface details.

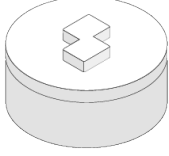
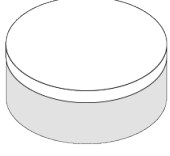
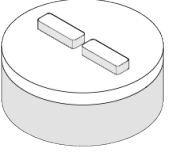
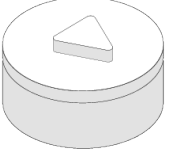
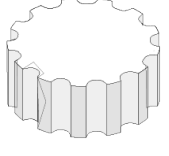
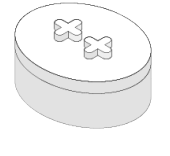

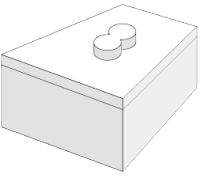
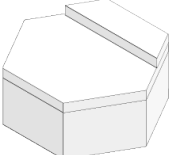
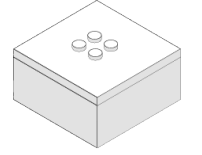
On the first level, there are 7 shape families; The interface blocks are all circular and the coding blocks have the rest of the shapes shown in Table 1. Amongst the shape families, the members are distinguished via their surface properties. For example, out of the coding blocks, the sound block family is hexagonal. A hexagonal member that has a topographic height covering less than half of its volume is the piano block.

We used cardboard as the tangible material due to its affordability and ease in shape giving (3). The 2mm width cardboard blocks were glued on top of each other. All the forms were within the boundary dimensions of 8 mm height and 4 mm in diameter.

Interaction: Tangible blocks are laser-cut card-boards with NFC stickers inside. We use card-board as an affordable and fast solution, but it can be 3D printed or built by any other material as long as students are comfortable with the material. To read the NFC stickers and run the output we developed an Android application. We didn't use a camera system to recognize the blocks because the system would constrain the working space due to the camera's viewing region. Hence, students would not be able to understand if the block is in viewing the region or identify such a problem easily.

To keep track of the blocks on the surface we used a rack as seen in Figure 4. Additionally, they can check the program with switching to the info area and then tap onto the

Table 1: Sample Tangible Blocks and Explanations

Blocks	Explanation	Blocks	Explanation
	Start block initiates the code. It is the only block without audio help. When the student is at info area, if this block is recognized, application returns to code area.		Info block opens the info area. When the application is at info mode, it gives audio hints about blocks.
	Clear All block clears all the created code. Students use them after the start block to be sure about they start with an empty code.		Run block executes the code and start the melody. If the device supports the NFC auto-start, students can open the application with tapping to this block
	Loop block starts a loop with the code part introduced afterwards. (e.g. A > B > Loop > C > D will run A and B once then starts a loop of C and D.)		Frequency block changes the frequency of the following blocks. We designed three levels of frequency: low (+), medium (++) and high (+++) We placed (+) sign to link with the level of frequency.
	Wave Type block changes the type of the wave. We put two options: Square or Sine.		Beat block changes the beat of the music. For now, it affects the whole melody. So, they can place it anywhere on the code. Similar to frequency blocks, has three options: slow, moderate and fast
	Sound block changes the instrument sample of the music. Three options are provided: Cello, Piano and Harmonium.		Note block have 8 options: 7 Notes and 1 Stop. They have braille alphabet on the blocks.

RUN block to hear all the script. The Android application recognizes the blocks with NFC stickers as the coder moves the phone over each block one at a time. As the last step, the play block is scanned and the melody plays from the speakers. The application has two modes: code and info. To create the code and play the melody, students need to be at code mode. If students switch to info mode, they can listen to the functionality of the blocks.

Language: The design of the blocks programming language has a very similar style with Processing [14]¹. The programmer needs to specify the attributes first, then define the object that will carry the properties. For example, the Figure 2 shows a sample code and its output from Processing. Attribute-oriented design allow us to run programs without inputs and variables for this prototype, yet it is a powerful concept to introduce children to develop their algorithmic thinking skills.

¹The kit is open source, the code is available at <https://github.com/asabuncuoglu13/budgie-tangible-music-kit>

```

fill(255, 0, 0);
rect(60,100, 100, 100);
ellipse(20, 40, 30, 30);
noStroke();
fill(0, 255, 0);
rect(60, 20, 100, 40);

```

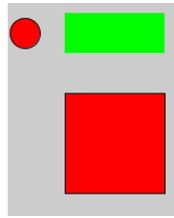


Figure 2: Processing Example and Output

In the code, the **fill** command affects both **rect** and **ellipse** objects. Similarly, our system takes attributes first and until recognizing an alternative attribute, code blocks will affect from the attribute. So, students do not need to recognize the same block to add the same attribute for the notes. For example,

Start > Add Note C > High Frequency > Add Note D > Add Note A > Run

Will run C4 - D3 - A3 melody and play a melody of normal frequency C, then low frequency D and A. The standard frequency is 4, so if you do not specify the frequency, all the notes will play at 4 octaves.²

We utilized Processing language as the core of our tangible kit for three reasons: (1) The language has an active community of creative coders, which allow novice programmers to discuss their problems about coding and music. (2) Processing has a sound library, if the learner would like to continue with a text-based programming platform, it will be familiar. (3) The language can be deployed to multiple platforms and converted to different programming languages such as JavaScript or Python.

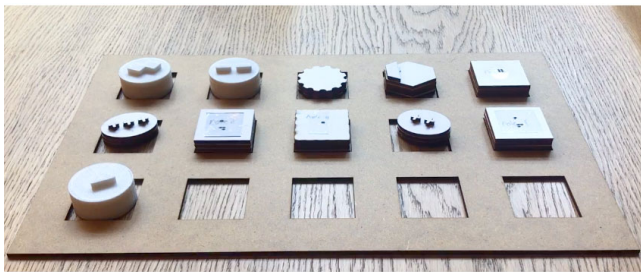


Figure 3: Blocks and rack used during the workshop. The rack eases the ordering of music blocks for the novice programmers.

²The demo of the system can be accessed at <https://budgi.es/demo.html>, people can experiment with the system from online platform before building tangibles.

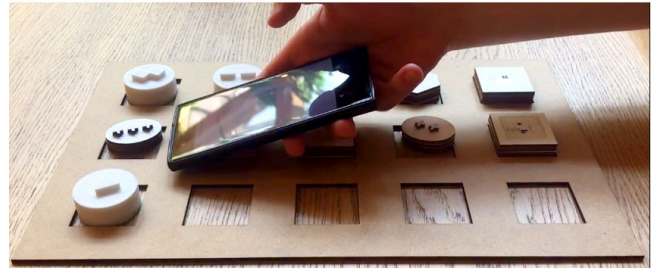


Figure 4: NFC reading of a music block. Students read the blocks one by one with their mentor's phones.

4 USER STUDY

We conducted a user study to observe children's interaction with the tangible blocks and the mobile device. Fourteen visually impaired students (7 boys, 7 girls, *Age*=12.5) with various socio-economic-statuses participated in the study.

In the first five minutes of the workshop, we had a quick chat about their interests and current knowledge of programming. All the participants have an interest in technology and programming, but have limited or no knowledge of programming. Only two of them have previous experience on text-based coding and three of them have experience with music. Overall, students use computers or mobile phones regularly.

For the user study, students were randomly selected to create groups of two or three. This grouping was made to observe the collaboration between group members. Every group had a mentor to keep them informed and to help them find the blocks for the first time.

We designed this user study structure to make participants experience sounds, learn the logic of creating algorithms and learn musical terms such as notes, octave, rhythm, melody, tempo, and sound. The user study had the following structure:

- The physical properties of sound (20 minutes),
- Historical context (10 minutes),
- Fundamental Music Theory (10 minutes),
- Introduction to Algorithm (20 minutes)
- Introducing Tangible Blocks and Creating Different Programs (60 minutes)

Analysis

We had five points of inquiry for the user study:

(RQ1: Motivation) Did the user study affect children's motivation to continue learning programming?

(RQ2: Challenges) Which challenges did the children encounter while using the system?

The rest of our questions aimed to observe the advantages of tangibles, as Zuckerman suggests.

(RQ3: Collaboration) How was the collaboration between the mentor and the group members?

(RQ4: Sensory Evaluation) Did they struggle to recognize the blocks?

(RQ5: Learning) How many of the blocks were covered during the session? Were they able to construct a melody by themselves?

To answer the questions, we observed the children during the study and also conducted a semi-structured interview both with the participants and the mentors after the study.

5 RESULTS AND DISCUSSION

One of the most prominent findings of our user study was participants enjoyment and engagement with the system. Throughout the study, the participants were smiling, talking excitedly and working very carefully with the system.

RQ1, Motivation: Upon finishing the user study, most participants asked about how they could continue coding, what would be the next updates to this music coding language and how they could learn new programming languages. During the interviews, all participants stated that they felt encouraged to start learning programming thanks to their pleasant experience with our platform. Even the two participants that knew text-based coding, stated that using a language especially designed for themselves was very exciting.

RQ2, Challenges: We encountered a challenge related to distraction during the study. To explain, the groups need to run their code occasionally to hear the creation. Throughout the study, a cacophony of musical sounds emerged which was slightly distracting. Regardless, the participants were able to maintain their focus on the task. Ideally, to scale the project, children will work in pairs and can use headphones to overcome this issue. In this way, many children will be still working in the same room but only their conversations would create a slight noise.

RQ3, Collaboration: Introducing the rack eased the collaboration between group members. To illustrate, placing the tangible blocks onto the rack allowed each group member to keep tabs on the constructed algorithm. This ease allowed them to apply a 'pair programming style' coding. Pair programming is a method for agile development in which one developer writes the code and another developer reviews the code. While the development, they switch the roles. Our system enabled children to practice this style intuitively.

RQ4, Sensory Evaluation: We designed our prototypes with simple forms that had two levels of information via shape and surface properties. All in all, the forms were easily distinguishable and recognizable for the participants. We had refrained from adding different features to our blocks based on our TUI design considerations that advocates simplicity.

On the other hand, we should highlight that there is an opportunity to discover other types of block distinctions than shape and surface information. For example, particular sounds can emerge when the students pick up the blocks. Another example would be creating blocks with different masses or different textures. Yet, implementing the extra feedback might cause a cognitive load to students, increase the cost of blocks and contradict with the simplicity principle. These assumptions have to be tested nevertheless.

RQ5, Learning: We designed a two-hour user study to cover the fundamentals of music definitions and introduce music programming. The students then experimented freely as a group to create different melodies. At the end of the workshop, all groups were able to arrange the tangible blocks correctly and were able to express their musical ideas via algorithms.

Limitations: We did not conduct the user study in a real-classroom setting, which we plan to do so in the future. Also, the fact that the children started the activity right after they met their mentors might have affected their interactions and self-expressions. In an ideal scenario, the children will work in a place and with a mentor that they know.

6 FUTURE WORK

After the encouraging findings of our user study, we will pursue developing our system with several next steps. First, we have to expand our design to meet the programming curriculum standards. For example, we will alter our design so that it allows tinkering and remixing the projects. Second, to fully develop the computational thinking skill, we have to introduce additional blocks to use abstraction, decomposition or pattern recognition skills while creating the algorithm. To fulfill these needs, we will further develop the project in two branches, (1) expanding the capabilities of the interface and (2) creating new physical blocks. We will add one more area to interface to allow students to share their projects. In this area, children will see their own and friends' creation. Sharing will be supported with additional *save* and *share* blocks. To support computational thinking development, we will introduce two blocks in the next step: creating functions and build parallel structures.

7 CONCLUSION

In this paper we presented our affordable tangible programming tool for visually impaired children that enables creating music with algorithms. We undertook a three-step research process which was 1) extracting TUI design considerations for visually impaired children with the contribution of two visually impaired developers, 2) development of the prototype and 3) user study with 14 visually impaired children ($Mage=12.4$). The findings of our user study indicate that our system has the potential to support children's engagement

and collaboration with each other. We further saw that the design of the blocks was easy to differentiate and use. After the user study, all the participants stated that they would like to pursue programming education, which indicates our design considerations ability to support their needs. Future research can apply these considerations and explore it's efficacy further. We contribute to the literature 1) as the first algorithmic style- tangible music creation platform for visually impaired children and 2) provide several TUI design considerations for students with low to zero sight. We believe that our research can inspire researchers aiming to develop tools for the visually impaired community.

ACKNOWLEDGMENTS

I would like to thank Ceylan Besevli for her great support in the writing process. Ezgi Cevik from Twin Science and Robotics, modelled the 3D blocks and coordinated the session with me. Melis Kahyaoglu from Twin Science and Robotics, drew the illustrations for the model files. T. Metin Sezgin gave ideas to improve the kit. I would also thank all YGA volunteers, who gave their own time to spread their knowledge and confidence to young minds of Turkey.

REFERENCES

- [1] A. N. Antle and A. F. Wise. 2013. Getting Down to Details: Using Theories of Cognition and Learning to Inform Tangible User Interface Design. *Interacting with Computers* 25, 1 (Jan 2013), 1–20. <https://doi.org/10.1093/iwc/iws007>
- [2] Ardublock. [n. d.]. Ardublock | A Graphical Programming Language for Arduino. <http://blog.ardublock.com/>
- [3] Rupert R A Bourne, Seth R Flaxman, Tasanee Braithwaite, Maria V Cicinelli, Aditi Das, Jost B Jonas, Jill Keffe, John H Kempen, Janet Leasher, Hans Limburg, Kovin Naidoo, Konrad Pesudovs, Serge Resnikoff, Alex Silvester, Gretchen A Stevens, Nina Tahhan, Tien Y Wong, Hugh R Taylor, Rupert Bourne, Peter Ackland, Aries Arditi, Yaniv Barkana, Banu Bozkurt, TASANEE BRAITHWAITE, Alain Bron, Donald Budenz, Feng Cai, Robert Casson, Usha Chakravarthy, Jaewan Choi, Maria Vittoria Cicinelli, Nathan Congdon, Reza Dana, Rakhi Dandona, Lalit Dandona, Aditi Das, Iva Dekaris, Monte Del Monte, Jenny Deva, Laura Dreer, Leon Ellwein, Marcela Frazier, Kevin Frick, David Friedman, Joao Furtado, Hua Gao, Gus Gazzard, Ronnie George, Stephen Gichuhi, Victor Gonzalez, Billy Hammond, Mary Elizabeth Hartnett, Minguang He, James Hejtmancik, Flavio Hirai, John Huang, April Ingram, Jonathan Javitt, Jost Jonas, Charlotte Joslin, Jill Keffe, John Kempen, Moncef Khairallah, Rohit Khanna, Judy Kim, George Lambrou, Van Charles Lansingh, Paolo Lanzetta, Janet Leasher, Jennifer Lim, Hans LIMBURG, Kaweh Mansouri, Anu Mathew, Alan Morse, Beatriz Munoz, David Musch, Kovin Naidoo, Vinay Nangia, MARIA PALAIYOU, Maurizio Battaglia Parodi, Fernando Yaacov Pena, Konrad Pesudovs, Tunde Peto, Harry Quigley, Murugesan Raju, Pradeep Ramulu, Serge Resnikoff, Alan Robin, Luca Rossetti, Jinan Saaddine, MYA SANDAR, Janet Serle, Tueng Shen, Rajesh Shetty, Pamela Sieving, Juan Carlos Silva, Alex Silvester, Rita S Sitorus, Dwight Stambolian, Gretchen Stevens, Hugh Taylor, Jaime Tejedor, James Tielsch, Miltiadis Tsilimbaris, Jan van Meurs, Rohit Varma, Gianni Virgili, Jimmy Volmink, Ya Xing Wang, Ning-Li Wang, Sheila West, Peter Wiedemann, Tien Wong, Richard Wormald, and Yingfeng Zheng. 2017. Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. *The Lancet Global Health* 5, 9 (sep 2017), e888–e897. [https://doi.org/10.1016/S2214-109X\(17\)30293-0](https://doi.org/10.1016/S2214-109X(17)30293-0)
- [4] Anke M. Brock. 2017. Tangible Interaction for Visually Impaired People: why and how. *World Haptics Conference - Workshop on Haptic Interfaces for Accessibility* (2017). <https://hal.inria.fr/hal-01523745v1>
- [5] Carnegie Mellon University. [n. d.]. Alice. Tell Stories. Build Games. Learn to Program. <https://www.alice.org/>
- [6] Michael S Horn and Robert JK Jacob. 2007. Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction*. ACM, 159–162.
- [7] H. Ishii and B. Ullmer. 1997. Tangible bits: towards seamless interfaces between people, bits, and atoms. *Proceedings of the 8th international conference on Intelligent user interfaces* March (1997), 3–3. <https://doi.org/10.1145/604045.604048> arXiv:arXiv:1011.1669v3
- [8] Yasmin B Kafai. 2016. From computational thinking to computational participation in K–12 education. *Commun. ACM* 59, 8 (2016), 26–27. <https://doi.org/10.1145/2955114>
- [9] Microsoft. [n. d.]. Microsoft MakeCode for micro:bit. <https://makecode.microbit.org/>
- [10] Microsoft. [n. d.]. Project CodeTalk - Microsoft Research. <https://www.microsoft.com/en-us/research/project/codetalk/>
- [11] Lauren R. Milne and Richard E. Ladner. 2018. Blocks4All. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18* (2018), 1–10. <https://doi.org/10.1145/3173574.3173643>
- [12] Scratch Team MIT Media Lab. [n. d.]. Scratch - Imagine, Program, Share. <https://scratch.mit.edu/>
- [13] 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* 00, 00 (2018), 1–49. <https://doi.org/10.1080/07370024.2018.1512413>
- [14] Processing Foundation. [n. d.]. Processing.org. <https://processing.org/>
- [15] Stackoverflow. [n. d.]. Stackoverflow - How can you program if you're blind? - Stack Overflow. <https://stackoverflow.com/questions/118984/how-can-you-program-if-youre-blind>
- [16] Stackoverflow Insights. [n. d.]. Stack Overflow Developer Survey 2018. <https://insights.stackoverflow.com/survey/2018>
- [17] Oren Zuckerman, Saeed Arida, and Mitchel Resnick. 2005. Extending tangible interfaces for education. June 2014 (2005), 859. <https://doi.org/10.1145/1054972.1055093>
- [18] Oren Zuckerman, Tina Grotzer, and Kelly Leahy. 2006. Flow blocks as a conceptual bridge between understanding the structure and behavior of a complex causal system. (06 2006), 880–886.