Using Robots to Make Computational Thinking Accessible to Children With Visual Impairments

Filipa Rocha
Instituto Superior Técnico

Hugo Nicolau
Instituto Superior Técnico

Ana Cristina Pires
Faculdade de Ciências da Universidade de Lisboa

ABSTRACT
Computational Thinking is becoming a fundamental skill, such as reading and writing, and expected to be used worldwide by the middle of the century. Visually impaired students find barriers to learning computer science. These start with inaccessible tools, such as language with graphical outputs [29], for example Scratch [26].

Growing up, children progress from a stage of reaction to external stimuli to actively engaging with the environment surrounding them and learning by interactive play [3]. Physical artifacts such as robotic kits and toys can immerse children in a playful learning environment becoming a powerful tool that promotes the development of computational thinking skills, for example problem-solving skills and thinking at different levels of abstraction [10].

Since robots can have a multi-modal interface, users can interact with the robot with no need for peripherals [3]. Navigating in the real world with the robot is a good starting point to prepare for future programming tasks [17], and contribute to the development of spatial cognition [38].

PROBLEM
VI students can find many barriers in learning computer science. These barriers start with inaccessible tools, and as any other child, a high level of abstraction is needed to understand what they are coding [21]. Graphical programming environments create accessibility barriers to blind children by heavily relying on visual elements and output [28].

Existing solutions are visually demanding and inaccessible to VI children [28, 30], placing them at risk of being excluded from learning CT. Coding kits often require children to create a set of instructions that will help a character (e.g., digital avatar or robotic device) overcome a series of spatial challenges by following a given path, avoiding obstacles, and collecting rewards. We refer to these activities as spatial programming activities as they contribute to the development of spatial cognition, a critical skill for VI children [38]. Ironically, previous research efforts to make CT accessible to VI children are largely limited to sequential audio-based actions, preventing VI children from engaging in spatial programming activities as their sighted peers.

Previous approaches, such as Storyblocks [22, 21], and Project Torino [37], use tangibles to program. However, the output of the activities consists of audio stories or music. Blocks4all uses a touchscreen device compatible with VoiceOver\(^1\) to move the robot spatially. Their studies showed that VI children

\(^1\)https://en.wikipedia.org/wiki/VoiceOver

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.

© 2020 Copyright held by the owner(s). Publication rights licensed to ACM.
DOI: 10.1145/1235

INTRODUCTION
Computational Thinking (CT) is becoming a fundamental literacy skill, such as reading and writing, and expected to be used worldwide by the middle of the century [31, ?]. CT “is the thought process involved in formulating a problem and expressing it in a way that a computer - human or machine - can effectively carry out” [1]. It borrows concepts from computer science [8], such as sequences, operators, and iteration as well as practices like being incremental and iterative, testing, debugging, abstracting, and reusing. The value of CT goes beyond computing contexts and promises to impact the social, emotional, and cognitive development of children[5, 12], while fostering personal and career growth [2].

There is not a direct and defined strategy to teach young children these computational concepts and practices, however most research scenarios include block-based programming languages with graphical outputs [29], for example Scratch [26].

Growing up, children progress from a stage of reaction to external stimuli to actively engage with the environment surrounding them and learn by interactive play [3]. Physical artifacts such as robotic kits and toys can immerse children in a playful learning environment becoming a powerful tool that promotes the development of computational thinking skills, for example problem-solving skills and thinking at different levels of abstraction [10].

Since robots can have a multi-modal interface, users can interact with the robot with no need for peripherals [3]. Navigating in the real world with the robot is a good starting point to prepare for future programming tasks [17], and contribute to the development of spatial cognition [38].
struggled with the activities, and additional work was required to reduce the demand associated with these challenges. We sought to bridge this gap of a fully tangible solution accessible to VI children.

**APPRAOCH**

In this dissertation, we propose the use of tangible blocks that build computer programs and a robotic device to render an accessible multimodal output. In our work, we started by conducting interviews with computer science teachers to understand with which methods and concepts should children initiate their learning. Following this, we sought to understand the qualities and flaws present in current programming environments for children regarding accessibility. First, we conducted a focus group with IT and SNEs (Special Needs Educators), where we presented four environments with at least one tangible component. The educators interacted with each and gave feedback. For the second study, we adapted blocks with additional tactile cues and audio feedback, and the Dash robot with augmented physicality, feedback, and feed-forward mechanisms. We used a Wizard of Oz methodology, where the children tinkered with the blocks, and an investigator would use the Wonder Blockly2 app to program the robot accordingly.

Following the formative user studies, we considered the results and created a fully tangible system. The proposed solution, takes into consideration the benefits of block-based languages, and is composed of tangible blocks with embossed pictograms to represent actions, magnets and saliences to facilitate coupling and TopCode markers. An app in an Android mobile device recognizes the markers with the camera and translates to actions for the robot to execute. The robot moves on top of a tactile map and gives feedback on its actions, allowing the user to follow it by touch. We took into account that most children find a robot engaging.

**GOALS**

Our work aims to allow VI children to learn CT and computer programming in an accessible way. We expect to meet the following goals:

- Understand the main difficulties children have when interacting with technology and programming tools, by interviewing computer science teachers and special need educators;
- Understand the necessary qualities of a spatial programming environment by with special needs educators and VI students;
- Design tangible blocks that allow VI children to overcome the existing barriers while programming;
- Determine a robotic output with audio feedback that will allow VI children to understand instruction abstraction and perform spatial tasks;
- Validate our solution by performing user tests with VI children.

2https://www.makewonder.com/apps/blockly/

**LEARNING COMPUTATIONAL THINKING**

CT is a discipline that uses computerized devices to create solutions and automate processes. CT is defined by the Computer Science Teachers Association and International Society of Technology in Education[13] as a “problem-solving process that includes formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analyzing, and implementing possible solutions to achieve the most efficient and effective combinations of steps and resources; and generalizing and transferring this problem-solving process to a wide variety of problems.”

There are four essential methods in adopting CT to start solving a complex problem: decomposing a problem or system into smaller parts, recognizing patterns and similarities among different problems, abstraction from irrelevant details and focusing on the crucial information, and develop a straightforward solution by steps or develop an algorithm. Finally, these simple steps can build a computer program and solve the problem in the most appropriate way.

To adopt the previous methods is crucial to consider the existing computational concepts that are highly useful in programming contexts. Sequences are a series of individual instructions or steps that specify the behavior or action to be produced. Loops are a mechanism to repeat a sequence of instructions multiple times. Events are the component that makes one action cause another to happen. Parallelism is the ability to have different instructions executing at the same time. Conditionals are components that allow distinct outcomes based on a defined condition. Arithmetic operators provide support to perform numeric and string manipulations. Data is the information to be stored, received, and sent. While adopting computational methods and concepts, the developers may follow CT practices even if they are not aware. CT empowers the development of creativity, planning, and problem-solving skills.[36].

**Block-based Tools for Learning**

Computer programming can be a way of teaching computational thinking, and the majority of research scenarios include block-based languages [3]. Blocks programming languages [4] can allow beginners to construct programs without struggling with the syntax. These graphical programming languages have proven to be more engaging and understandable for new learners. There are learning barriers that block programming languages that attempt to minimize: selection, use, and coordination.

When constructing a program with a blocks programming language, users can rely on recognition of the categories and shape of the block instead of recalling vocabulary. It reduces the cognitive load, since blocks can only be assembled in a certain way or with other specific types of blocks, the code is less likely to be syntactically incorrect. These characteristics and designs simplify the discovery and exploration of building programs for new learners that can concentrate on the meaning of the code instead of the writing notation.
Robots for Learning
There is research regarding children robot interaction in different areas, such as children suffering from anxiety [2]. Although robots encourage interest in science and technology, there is minimal research regarding the interaction between VI people and robots.

Researchers are exploring how blind people perceive robots and how these devices fit in their expectations and fears regarding the increase of dependency in them [6]. The researchers could infer from the answers that the participants preferred to feel in control. If they were to be in contact with a robot, they preferred that it respond to commands and questions. While interacting with a robot, it was helpful to the users that it made a sound, allowing them to identify what was its position and state.

Castro-Gonzalez et al. [11] proposed a physical haptic interaction study that focuses on the communication between anthropomorphic robots and children with visual-auditory impairments in a game mode.

Researchers in Australia presented a project[3] that uses a SONY Aibo platform targeting blind toddlers and focusing on how robots can assist in developing and learning pre-orientation. The children enjoyed interacting with it and adopted different postures before and after the session.

Metatla et al.[27], used a co-design approach to design and evaluate a robot-based educational game that could be inclusive of both VI and sighted children in the context of mainstream education. They ran focus a focus group and co-design workshops to evaluate the existing barriers and opportunities in inclusive play, and design an inclusive educational game that could address some of the barriers outlined. From the sessions resulted a set of design guidelines that emphasise multisensory feedback, hands-on creation, and narration as a means for modulating pace and stimulation; all of which were highlighted as potential barriers to inclusive interactions.

RELATED WORK
Programming for Children
Scratch[26] is a visual programming environment to learn computer programming with the goal of self-directed learning and tinkering. Children code through snapping together blocks to control sprites on screen. The platform is visually intuitive to navigate with key components always visible and with commands divided into color-coded categories. Scratch is always live with no need for a compilation step or an edit/run mode distinction. It allows the user to tinker, encouraging the learning process. This strategy eliminates syntax errors since the blocks only fit together in specific ways or with certain block types. All these features allow users to learn and evolve in programming thinking quickly.

ScratchJr [14] is the redesign of the Scratch platform for the developmental and learning needs of children from kindergarten to second grade. ScratchJr was developed, focusing on making the interface easier to understand, allowing children to explore and develop while fostering their creativity. Illiteracy was an identified barrier for the kindergartens who could not identify blocks by the category name. As children tend to be imaginative and play freely, they would explore the tool alone during the pilot tests.

Strawbies[16] is a real-time tangible block-based programming game. Children use an iPad app that can identify the tangible blocks that construct a sequence of instructions. Tangible technology is an excellent way to introduce computational thinking since it evokes familiar teaching and learning. The tiles have a design to make them easy to assemble, and once connected, prevent the set from falling apart if dragged around. The game was evaluated in six play sessions where the authors could observe the users. Results revealed that the system was engaging, as children wanted to play more and new children joined the open sessions.

TuTan [15] is another tangible programming system. It is operational in a tabletop interface designed to overcome the main difficulties in learning computer programming. The authors identified that the main difficulties were related to syntax and the used tools than the programming concepts themselves. This resulted in using tangibles to promote the complete exploration and expressiveness of the users.

E-Block [40] and T-Maze [39] are block-based tangible programming tools for children. The goals in the design and implementation of this system were for the children to program with no space limitation, the system being easy to learn, and having real-time feedback for the debugging process. The system has programming blocks that guide a character through a maze. The blocks are big with drawings representing the actions. The computer reads the blocks. There are four types of blocks: start and end, direction, loop, and sensor. The system can give the user feedback if the semantics are not correct, allowing for a more straightforward debugging process.

The Cubetto [1] robot was designed for children. The tool does not use a screen, and it consists of a robot, a programming tray, and blocks. The system promotes learning programming concepts such as algorithms, sequencing, and debugging with a code-by-code methodology and a robot to perform spatial activities. The activities can progress into more complex programming as the skills of the children progress.

KIBO [35] is also a robotic kit that was developed considering three learning goals for the kit: foundational engineering concepts of sturdy building and construction, foundational programming skills of sequencing, repeat loops, and conditional branching, and open-ended creativity and artistic design. KIBO involves hardware parts to assemble and tangible blocks to program the robot. After constructing the code sequence, the children can scan the barcode on the block with the robot and then watch it follow the instructions. By observing and interviewing the children testing the kit, it was possible to infer that it suited the proposed learning goals.

Object-oriented programming is easily accessible to beginners since it allows modeling of real objects as software objects. Research has shown that programming can have a positive impact on the educational achievements and social-emotional interaction. Tangible Interfaces are a popular method that connects the interaction with the physical world with computer
programming. With the previous ideas in mind, TanProStory [32] is a tangible programming tool for a storytelling environment to transmit object-oriented programming concepts to children. The tool consists of different cubes that connect between them and the computer so that the children can create their characters building the story and observing the result on the computer screen.

**Non-Visual Programming**

Blocks4All [28] is a prototype environment developed to overcome the current accessible barriers for VI children in digital block-based programming environments. The application was developed for a touchscreen Apple iPad since these have a built-in screen reader and zoom capabilities. To surpass the identified barriers, Milne et al. used a Dash Robot as an accessible and tangible output. Blocks are accessible by VoiceOver to announce their location and type of block.

Bonk [20] is an accessible programming toolkit that enables the creation of interactive audio games based on the JavaScript programming language. The code is JavaScript, with many convenient functions that reduce programming overload in listening to the code. The toolkit has a text-to-speech feature adjustable to the needs of the users that produce words or sound effects.

Pseudospatial Blocks [23] is another block-based language, based on arrow key navigation. In the case of (PB), the toolbox located is to the left of the working space where the blocks are stacked vertically, so regardless of the position of the block, a left movement will lead to the same location in the toolbox. In (PB), the code is constructed by first selecting an insertion point to the new block, and the system will filter so that only the eligible candidates can be selected. The system produces audio as output.

Audio Programming Language(APL) [33] is based on audio interfaces to enhance problem-solving and thinking skills in teenage blind learners with computer programming. (APL) has two main layers: Audio Interface and Programming Logic. The first includes a circular command list and query; the second consists of four states that define the logic: run program, loop or condition, delete the last command, save command and verify the next step. The program can request answers to a set of questions to complete the command line at the audio interface layer. The output of the code is always audio-based. The authors implemented a qualitative case study methodology where they observed and recorded the interactions of the users with the program. The main goal was to familiarize the users with elements and concepts used to solve problems with the (APL) program while learning computer programming basics. In the beginning, users had difficulties understanding the more complex concepts, such as loops and conditionals, but they could apply them and build their projects with practice.

Torino [37] is a physical programming language to teach the basic programming concepts and computational thinking regardless of their level of vision. The primary purpose is to explore collaborative learning scenarios between children with mixed visual abilities and observe how technology can participate. The system comprises instruction beads that can be physically connected and manipulated to generate sound as output. Each bead has buttons to control the repetition of the sound to be played, introducing the concept of variable, different size and shape to be better identified, and a custom circuit that allows them to connect and transmit data to its neighbors and the hub to form the sequence. In addition to the physical, spatial, and auditory configuration of the system, the authors designed a set of activities to assist the children in exploring programming concepts as commands, sequences, variables, abstraction, threading and iteration, problem-solving skills, and domain-specific vocabulary.

StoryBlocks [21, 22] is another tangible programming toolkit. The blocks represent story components, and the code produces an audio story or game as an output. Users assemble physical blocks that represent characters and actions to create an interactive story. The authors conducted a formative study with 16 participants. In general, all the participants were able to produce their own stories. During the evaluation, the authors identified some issues in the interaction between the users and StoryBlocks. Overall, the authors believe the combination of tangible programming blocks and audio output can be a compelling way to teach basic computational concepts.

Kane et al. [19] describe an introductory environment implemented in a workshop where students from different areas were working towards a common goal, determine the fictional impact of Comet ISON. Students had the task to monitor comet sightings using the Twitter social media and write programs to produce visualizations of the predicted impact zone. The students could send the data to a web service that would plot the coordinates on a USA map and explore using VoiceOver, or generate a credit-card-sized 3D tactile graphic based on the map. The authors found the workshop successful because even though not all students finished the activities, they spent a week developing their programming skills.

Ludi et al. [25] performed pre-college workshops targeting VI students between grades 7 and 12 to engage them in robotics programming and address the existing gap in introducing students to a future in the computing area. The goal was to present computer hardware concepts, general computing skills, and problem-solving activities in an accessible environment. The students performed different planned activities, starting with building the robot kit and then programming it, writing code with the help of screen readers and magnifying tools, and careful placement of comments in the code. While observing the workshops, researchers noticed the students were engaged in the proposed activities to interact with the robot and collaborated to achieve their goals.

Ludi et al. [24] also proposed JBrick as a part of a pre-college workshop for the VI since the LEGO software to control and program LEGO Mindstorms is not accessible because it is not screen reader compatible. The user will be able to hear the source code and errors from the compiler, and the system will recognize the NXT brick, compile the code, download it to the brick while indicating the process status with audio. The interface was designed to merely having a streamlined menu with fewer and larger icons and adjustable to accommodate needs of the user, such as modifying font size and color, key-
words, and background. The keyboard is the only input device to navigate the platform, and the output is the robot behaviors and sound.

A group of informatics teachers also decided to introduce robotics and computational thinking to VI pupils [17]. They planned separate lessons for every secondary school grade. To the younger classes, they introduced Bee-Bot, a programmable toy in the shape of a bee with seven color-coded buttons at the top embossed with different shape symbols. The students had a gridded map to perform the proposed spatial programming tasks. For the older classes, the activities involved a LEGO WeDo kit. The kit needed assembly and programming using a text-based language. This activity proved to be difficult for VI students, that had problems assembling the bricks.

There is progress with a block-based tangible programming tool with a robot for spatial output. CardBot 2.0 [9] is a low-cost solution that uses geometric cards with QR Codes to program the robot. The user has to layout the cards building the program, and then use a mobile phone to read the codes in the correct order. P-Cube [18] is another solution that makes use of a mat with RFID readers where the children can place the tangible blocks to construct the desired sequence. The blocks have RFID tags to be identifiable and have embossed icons representing their action. The mat connects to a computer sending the instructions to the robot, although users do not have to interact with it. Younger children can program the robot by drawing lines for it to follow.

Discussion
The described works regarding programming languages and toolkits for children to learn programming present useful strategies. Block-based have advantages, such as avoiding syntax errors, interacting with a simple interface, and rapid code creation progress. The existing tangible toolkits are attractive to young children since they allow them to tinker with blocks. If the system provides feedback to the user, it can help the children understand what they are coding, debug errors, and distinguish similar concepts and blocks.

Hands-on experience is also essential for blind learners and young children to facilitate what they are learning and build abstract thinking. An example of an output that has proven to be very popular with children of all ages is a programmable robot. Robots can move in space, produce audio and light, and allow the children to follow their behavior. Even though there is little research with an entirely tangible system with blocks and a robot to execute spatial tasks, we acknowledge that it is a compelling way to teach VI children basic computational concepts.

EXPLORING ACCESSIBLE PROGRAMMING
Before designing our prototype, we sought to understand the qualities and flaws present in current programming environments for children regarding accessibility.

Study 1: Exploring Current Programming Tools
We conducted a focus group with IT and SNEs, where we presented four environments with at least one tangible component. The educators interacted with each one and gave feedback. We recruited 4 SNEs and 2 IT instructors from two primary and secondary inclusive schools from our city.

We started the focus group by presenting four different environments (2). For each of the environments, we made a demonstration of the features and highlighted the most relevant. We commanded the virtual character or robot to perform one action, a sequence, and walk in a square. For the character to move in a square, we made a repeating sequence - a loop - except with Puzzlets and DOC that do not make it possible. The educators were encouraged to interact with the environments and brainstorm to identify qualities and opportunities in each one.

Tangible User Interfaces were one of the most commented during the session. Educators were very favorable of this type of system that combines a hands-on approach with digital feedback. Educators highlighted the advantage of children developing their fine motor skills. None of the educators considered screen-based technology as a solution for blind children.

Educators highlighted the importance of using a robot in this environment and the inclusion of physical and socio-emotional affordances. The educators preferred the robot Dash for observed aesthetic qualities. On the contrary, observed limitations such as the presented programming methods not being accessible to VI children and the robot not giving any audio feedback on performed actions or its position. The educators admitted the possibility of augmenting Dash to surpass these limitations.
We used augmented Osmo Awbie blocks as the programming blocks, where the robot had to move in a map placed on the table. The activities were goal-directed, with the children to work with both hands simultaneously, and the robot used it to code the videos. With each iteration, we enriched the codebook with new observed codes.

The activities started with a discussion about robots, with one investigator asking the children questions to motivate them. The children had to complete the challenges the investigator proposed. They were spatial activities where the robot had to move in a map. The robot used was Dash, with felt pads making eyebrows. The children manipulated adapted Osmo blocks to control an augmented Dash robot moving in a tactile map. The findings reported here emerged from our observations later validated in a follow-up focus group with their educators.

Seven VI children from the same school of Study 1 agreed to participate in this second workshop. The children were between five and eleven years old with different comorbidities. The children were divided in three groups according to age.

We used augmented Osmos Awbie blocks as the programming tools. The blocks needed strong tactile cues. We added a Flic Smart Button\(^3\) with audio feedback to the action blocks and an arrow of felt pad to the direction blocks. The robot used was Dash, with felt pads making eyebrows. The children manipulated adapted Osmo\(^3\) blocks to control an augmented Dash robot moving in a tactile map. The findings reported here emerged from our observations later validated in a follow-up focus group with their educators.

Study 2: Exploring Potential Approaches
The solutions presented in Study 1 were all inaccessible. However, they all showed to have qualities. Considering the first study results, we conducted a second one following a Wizard of Oz methodology with VI children. We adapted a solution to include a set of qualities identified in the previous study and engage VI children in a programming workshop with spatial activities. The children manipulated adapted Osmo\(^3\) blocks to control an augmented Dash robot moving in a tactile map. The findings reported here emerged from our observations later validated in a follow-up focus group with their educators.

Seven VI children from the same school of Study 1 agreed to participate in this second workshop. The children were between five and eleven years old with different comorbidities. The children were divided in three groups according to age.

We used augmented Osmos Awbie blocks as the programming tools. The blocks needed strong tactile cues. We added a Flic Smart Button\(^3\) with audio feedback to the action blocks and an arrow of felt pad to the direction blocks. The robot used was Dash, with felt pads making eyebrows. For the spatial activities, we used a map. It consisted of 6 EVA foam tiles of 33 x 33 cm with two colors: red and grey.

The activities started with a discussion about robots, with one investigator asking the children questions to motivate them. The following activities were goal-directed, were the children had to complete the challenges the investigator proposed. They were spatial activities where the robot had to move in a map placed on the table.

The videos were coded using a reflexive thematic analysis\(^7\) following a hybrid coding approach. First, we designed a codebook from our theoretical background and knowledge and used it to code the videos. With each iteration, we enriched the codebook with new observed codes.

The children explored the blocks and the robot with enthusiasm. Engaged in the proposed spatial activities and even

3\(^3\)https://www.playosmo.com/en/coding/
4\(^4\)https://flic.io/
5\(^5\)http://users.eecs.northwestern.edu/ mhorn/topcodes/
The blue block with a speech bubble to give the instruction of talking; the yellow block with a music note to give the instruction of dancing, the red with a droplet-shape arrow to give the instruction of walking and chose the direction with the arrow; the green blocks have two or three dots to represent the number of times the following sequence will be executed, or a hand representing the end of the repetition; the play block has a play symbol in the top part of the block that can be pressed to uncover the TopCode below.

**Robot** To enhance the physicality of Dash and help with the identification of the front of the robot, we added felt pads making eyebrows above the eye and a bowtie in the body.

**Workspace** The workspace represents the field of view of the camera. The Legos are in a rectangle giving the children tactile cues to help identify where they should construct the sequence of blocks. There is also a recognition area where the user can place any block for the system to announce which action it represents. The robot moves on the foam tiles map where objects or obstacles can be added.

**Block Interpreter** The mobile device app uses the back camera to scan the workspace and identify the tangible blocks. To recognize their identification number, general position, and orientation, we use the TopCode vision library, transforming them into digital markers. First, we need to determine if the TopCode associated with the play block is present. Otherwise, Dash will not execute the sequence. If the marker is present, it plays a feedback sound to inform the users. The remaining markers are then sorted from left to right, and each marker represents an action for Dash to execute. The Wonder Playground API for Android⁶ translates the code to actions then sent for Dash to execute. If the TopCodes associated with the repetition blocks are present, the app identifies how many times the following sequence will execute and where the sequence starts and ends. The mobile device and the robot communicate via Bluetooth.

**REMOTE USER STUDY**

The goal of this study was to explore BATS usage in a naturalistic setting and identify its benefits and limitations in the context of VI children. These studies aimed to address three main research questions: (1) Does BATS support computational thinking learning? (2) Is BATS engaging and what aspects are effective in engaging children and parents? (3) Can the system be used in collaboration between two users?

We created a remote testing toolkit to be autonomously used by the participating families. The toolkit included the BATS prototype with the tangible blocks, the Dash robot, an Android mobile device with the app installed, and a foam map for the robot to move.

We carried out a within-subjects design study, where each participant tested the system usability by performing the spatial tasks on the provided guidebook. For each activity, the participant had to manipulate the tangible pieces so that the Dash robot achieved the proposed goal. Interviews were held after the participants tested the system via video call. The questions focused on the personal experience and engagement of the participants with the components of the system.

The first activity is the system set up and start recording. The first position of the robot should always be the one closer to the child. For the following activities, the participant has to manipulate the tangible pieces so that the robot can achieve the proposed goal. There is no time or a limited number of trials. The second activity is composed of simple tasks of one block sequence, connecting each action block to the play and observing the robot. The first activity also instructs on how to use the recognition area to hear the system announce the action the block represents. The third activity has three tasks incorporated inside a story and the answers to each. The fourth activity presents an obstacle for Dash. In this activity, we did not give answers in the guidebook. The fifth activity makes use of the loop block.

**Conclusions**

We designed activities to guide the family to interact and learn how to command the robot. Such tasks would afford children to apply five CT competencies. Debugging, Data collection, Algorithms and Procedures, Problem Decomposition, and Pattern Recognition [34]. When coding the videos, we used these competencies as codes. We also observed if children could apply CT concepts during the interaction, such as building sequences and repetition.

We observed that children enjoyed the use of tangible blocks. Children identified blocks by their color or the embossed pictograms and sometimes used the recognition zone. Some children did not recognize the symbol itself, but they memorized which shape identified which action. The families found the robot attractive. The parents mentioned the sound and the lights as relevant characteristics for the children to perceive the behavior of the robot.

Most parents believe that the children could use the system autonomously after the first interaction with their help. We observed the families performing the activities together, and different types of collaboration between them.

In sum, we believe that BATS is an engaging way for VI children to learn CT. We could not test it with more families,
but overall, the qualitative feedback of this experience was very positive.

CONCLUSION

Everyday technology evolves and children have the opportunity to be part of this evolution. However, this is not a direct approach for VI children since they find many barriers in learning computer programming. Most programming scenarios for children include block-based programming languages with graphical output, or input.

ACHIEVEMENTS

We presented the design and implementation of a tool to help VI children learn computational thinking. We reported on two studies addressing the flaws and benefits present in current programming environments in terms of accessibility. We designed the tool taking into consideration the previous work in the field and the results from studies. Opposing other environments in the market, our tool is block-based and fully tangible, with multimodal feedback.

Then we report on a remote user study to test the tool with five families. The results show the system is engaging for children and they were able to apply CT competencies. However, users helped identify improvements in the tool.

LIMITATIONS

Our study included five families, which represents a small sample of VI children interacting with the system. Of the five children, only one was completely blind. A higher number of participants would result in a more significant analysis of quantitative data. During the user study, we were able to identify some complications with our prototype.

Due to the COVID-19 pandemic, we had to perform a remote user study. The families recorded themselves, which means that sometimes the field-of-view is not the best to analyze the interaction between the user and the system, or the recording does not show the entire interaction.

FUTURE WORK

The last user study participants gave feedback on the tool and what improvements to make to all the components. Regarding the blocks, the families suggested braille inscriptions or colored symbols embossed to identify the represented actions. Even though the blocks already have magnets and saliences to help with the coupling, this could be made more evident or stronger.

The robot could have more tactile clues to identify its orientation. It could also have more audio feedback for the user to identify its position and when it reached or failed the goal.

Finally, the activities presented should guide the children in the learning evolution.

REFERENCES


[29] Macià Mut-Puigserver, M. Magdalena Payeras-Capellá, Jordi Castellá-Roca, and Llorenç Huguet-Rotger. 2018. mCITYPASS: Privacy-preserving secure access to federated touristic services with mobile devices. *Studies in Computational Intelligence* 727 (2018), 135–160. DOI: http://dx.doi.org/10.1007/978-3-319-64051-8_9


http://dx.doi.org/10.1145/2702613.2732806

http://dx.doi.org/10.1145/1056808.1057018

http://dx.doi.org/10.1016/j.edurev.2017.09.003

http://dx.doi.org/10.1145/2771839.2771868


http://dx.doi.org/10.1145/3064663.3064689

http://dx.doi.org/10.1007/978-0-585-33485-1_11

http://dx.doi.org/10.1145/1999030.1999045

http://dx.doi.org/10.1145/2380296.2380327