Melodic – Teaching Computational Thinking to Visually Impaired Kids

Rui Costa
Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Cristiana Araújo
Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Pedro Rangel Henriques
Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Abstract

This paper presents a proposal, called Melodic, to develop Computational Thinking skills in kids with special educational needs, in this case blindness. The aim of this research is to characterize the subject and identify what are the current most used and best practises to teach this different way of Thinking to kids under those circumstances. In this paper more technical aspects are discussed, such as the architecture for the proposed application in order to accomplish the project goals. Melodic was carefully designed to have an intuitive interface for blind people and a seamless workflow, combining tactile hardware, and QR Code technology with a sound based output.

2012 ACM Subject Classification Social and professional topics → Computational thinking

Keywords and phrases Computational Thinking, Visual Impaired Students Education, Teaching through Music, Learning Resource

Digital Object Identifier 10.4230/OASIcs.ICPEC.2021.4

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

Acknowledgements We would like to thank Isabel Barciela and Marta Paço both from Íris Inclusiva for their valuable comments and suggestions; Without their collaboration this work would not be possible.

1 Introduction

In today's world technology is everywhere we look. As a consequence employers look for people with strong skills in computational field. This fact leads many young people to choose programming courses. Programming students normally face a great difficulty due to the traditional way of thinking they are taught in conventional schools, while they should be trained in Computational Thinking that is the key to solve general or programming problems. Computational Thinking is based on some key concepts such as *Logical Reasoning*, *Algorithm Design*, *Decomposition*, *Pattern Recognition*, *Abstraction* and *Evaluation* and changes the whole way of thinking about the resolution of a given problem [10, 3, 12]. This change of the way of thinking is not an easy task so the sooner it is introduced to people the better. That said, it is of uttermost importance to research for a strategy to teach children in a captivating way to keep them motivated. Currently one technique that is very used is "Game Based Learning" [8].

This technique is characterized by being a type of game play with defined learning outcomes. Usually the game used is a digital one, but this is not always the case. There are many arguments in favour of game based learning, such as:

- **Motivation**: Normally games for entertainment are able to motivate learners to stay engaged over longer periods of time through a series of features of motivational nature. These include incentive artifacts, such as stars, points, leader boards, badges, and trophies, as well as game mechanics and challenges that learners find interesting;
- **Player Engagement**: Is related to motivation and relies on design decisions that reflect the specific learning goal, learner characteristics, and setting;
- **Adaptability**: Learner engagement is facilitated by the adaptive part of the game, being this customizable by the player or personalized. This characteristic is defined by the capability of engage each learner in a way that reflects his or her specific profile;
- **Graceful Failure**: Rather than putting failure as an undesirable outcome, it is an expected and sometimes necessary step in the learning process. This is achieved by lowering the consequences on a failure situation and feedback on the wrong answer so that the player can learn from his or her mistakes.

In order to teach Computational Thinking properly it is mandatory to adapt the Learning Resources to the target learners. Resources must be adequate to the training of Computational Thinking so that the students develop new knowledge and the different abilities and acquisition of values that define Computational Thinking [1, 11].

Reviewing the literature in the area, it is clear that there are some platforms to support the teaching/learning process aimed at a general audience; however, when it concerns special need kids, the available resources are very few. It is very important to realize that mos of the referred resources rely upon a visual interface based program paradigm. In order to have a system that successfully helps blind kids to learn the concepts of Computational Thinking, it is mandatory that the system does not rely on a visual interface but on an audio or tactile one [9].

With this in mind, the audio based interface combined with a tactile input device seems the best approach to create resources to teach some Computational Thinking concepts through music. That musical approach has clear advantages, as listed below [6]:

- Debugging a song to make it sound like desired, is similar to debugging a program in order to make it to have the desired behavior;
- Musical phrases correspond to program commands;
- A song is a combination of musical phrases just as a program is a combination of statements.

In this way, we intend to create a Learning Resource capable of, through musical based games, teach the fundamental concepts of Computational Thinking to blind kids to help them to solve programming or general problems using this approach.

The paper is organized as follows. Section 2 explains the concept of Learning Resource, characterizes blind and visually impaired people way of learning, ways to teach visual impaired students and the special characteristics that we have to take in consideration when choosing or creating learning resources for this context. Section 3 presents the principle of the Melodic. Here we can see understand the way of interacting with this application and what kind of output we are expecting to get. Section 4 talks about Melodic's architecture and used technologies. Also in this chapter are described the implementation approaches and prototype testing. Section 5 are given some examples of teaching approaches and ways to use this system to reach its goal, train Computational Thinking. Section 6 summarizes all the work done until this point and the next steps that are predicted to be done for this project.

2 Adaptability of Learning Resources to Blind Students

Learning Resource is a device used for educational purposes in any format, real or virtual, that illustrates or supports one or more elements of a course of study; and may enrich the experience of the pupil or teacher. Learn resources will serve to train Computational Thinking, so they play a very important part in this process. For this reason the resources must be adequate to the training of Computational Thinking so that students can develop new knowledge and the different abilities and acquisition of values that define Computational Thinking [1].

As far back as 1967, there were reported various issues related to the education of visually impaired learners. The focus was on the importance of vision and the mode of reading, and attempted to illustrate how intelligence manifests itself in blind learners as compared to deaf. Their work showed the relevance of blindness and information processing and also illustrated the maximum utilization of available sensory data during learning activities and the translation of visual stimuli [7].

In visual impaired learners conceptual development and abstract thinking seem to be delayed by the absence of visual stimulation or images. With this cognitive development occurs more slowly independently of the age group [5].

A technique to teach conceptual subjects is using tactile stimuli to overcome the difficulty imposed by the lack of sight. Figure 1 illustrates how multiple tactile stimuli can compensate for the loss of sight, allowing learners to perceive size and shape of objects [4]. However, blind people are easily overwhelmed by the complexity of very “full” or “busy” diagrams, sketcher or models. A way to counter this feeling is to be spatially oriented when walking or reading documents.

In the learning point of view, blind and visually impaired learners require specific strategies, addressing their unique learning mediation needs. With loss or absence of vision, the amount of sensory data available to the learner is reduced and for this reason is very important the use of multi-sensory approaches, like the one in Figure 1 [2].

To facilitate this multi-sensory interface, Braille text, talking calculators, computer voice over and many more became standard equipment to teach blind and visually impaired people.

In order to best adapt visually impaired students it is mandatory to adapt the curriculum with strategies such as [4]:

- setting a substitute task of similar scope and demand;
- replacing one impossible or unfriendly task with one of a different kind;
- allowing the learner to undertake the task at a later date;
- using another planned task to assess more outcomes than originally intended;
- giving the learner concessions to complete a task;
- using technology, aides or other special arrangements to undertake assessment tasks;
- using an estimate based on other assessments or work completed by the learner;
- considering the format in which the task is presented.

With this techniques we can adapt conventional teaching to a visually impaired students so that they can learn most of the subjects regarding programming skills.

3 Melodic, the principle and workflow

In this section the fundamental principle of Melodic will be discussed: helping visual impaired people train Computational Thinking. For this we use a set of tactile blocks with different

Figure 1 Combination of three-dimensional models with "Zytec" sketches labelled in Braille as a learning strategy.

shapes, textures and meanings (Figure 2). Each one of this blocks has an assigned value and are part of a class of blocks:

- **Special Block**: This block class is characterized by a smooth surface. Special blocks can be of three different types:
  - **Control**: These blocks control the beginning and the end of an algorithm or instruction. The two blocks shown at the top of Figure 3 are used for beginning and ending sequences. The two blocks shown at the bottom of Figure 3 are used to start and stop loops.
  - **Speed**: These blocks control the time interval between musical notes. As we can see in Figure 4, there are three levels of speed: *slow* (represented by the symbol "`>`"), *medium* (represented by the symbol "`> >`") and *fast* (represented by the symbol "`> > >`"). The usage of this block is not mandatory; default value is "slow speed";
  - **Instrument**: These blocks control the instrument that shall be played. The letter `G` means *Guitar*, `P` means *Piano*, and `F` means *Flute* (Figure 5). This block can be used multiple times throughout the sequence creating a set of notes played by different instruments. This is also a non-mandatory block being Piano the default instrument.
- **Number Block**: This block class is characterized by a texture with lines and represent numbers from one to ten. These assign value to the number of iterations of a loop or the number of repetitions of a single note (Figure 6).
- **Musical Note Block**: This block class is characterized by a rough texture and represent musical notes. The output will be a sound representation of these musical notes (Figure 7); The musical notes played are: *Dó*, *Ré*, *Mi*, *Fá*, *Sol*, *Lá*, *Sí*, where *Dó* is represented by the block that contains only *one point*, the *Ré* is represented by the block that contains *two points*, and so on.

As previously mentioned, each block class is characterized by its texture. It is very important that each class of blocks has a different texture so that the visually impaired user can easily distinguish each class. The blind user must first learn the texture that identifies each class of blocks.

Figure 2 Tactile Blocks used to interact with Melodic.

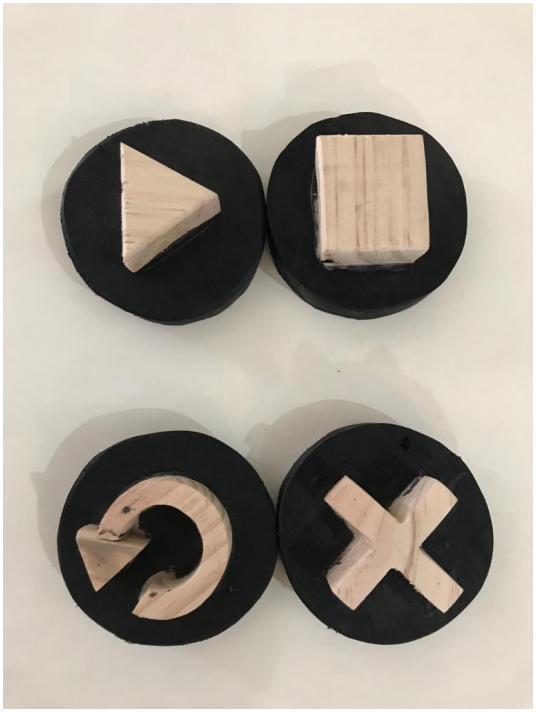

Figure 3 Control Blocks.

Figure 4 Speed Blocks.

Figure 5 Instrument Blocks.

Figure 6 Number Blocks.

Figure 7 Musical Note Blocks.

Figure 8 Top and Bottom faces of one Block.

In addition to the texture, the blocks always have the base painted in black (a dark color) and the symbol of the block in wooden color (light color). This color contrast is very important for low vision users, as it allows them to more easily identify the symbol of each block.

With this tactile blocks the user can create algorithms to represent some melodies, being able to increase in complexity as he improves his/her Computational Thinking skills. Each block contains a QR code at the bottom that identifies its meaning (Figure 8). To read the meaning of each block, the user must install the Melodic application on his smartphone. After that, with using the mobile app the user must read the QR code located at the bottom of the block.

In Section 4 will be presented the architecture of Melodic.

4 Melodic, system architecture

Melodic is a system aimed to be used with a mobile device, being this either a smartphone or tablet. To mitigate incompatibilities between *Apple* and *Android* devices, the *React Native*[1] framework was used, which provides parity between the two platforms with the same JavaScript code.

In Figure 9 we can see Melodic's Architecture Diagram that depicts the system components:

- **User**: The user controls the composition of the blocks to form a musical sequence that produces the desired sound after being scanned by the mobile device;
- **Melodic Blocks**: Set of ordered blocks (algorithm) composed by the user to form the desired musical sequence. Each block has a QR code, at the bottom, so that the mobile device can read its meaning;
- **Mobile Device**: Component in charge of all the logical operations behind this app containing multiple sub-domains:
  - **Camera**: The camera is in charge of reading the QR codes of the blocks;
  - **Melodic App**: It is the core of the system. This module receives, through the camera, the QR code of each block and processes it, behaving like a compiler. After processing the QR codes, the Melodic App sends the information (musical melody) to the Speaker module;
  - **Speaker**: The speaker receives the information, sent by the Melodic App, with the musical melody to play and converts it into a real sound to output to the user.

It is important to notice that the mobile device also provides tactile feedback to the user, in the form of vibrations, to alert the user that the QR code of a given block has been scanned.

In order to facilitate the testing of this system in a easy way, it was developed with the help of Expo[2]. Expo creates a project capable of being accessed by anyone without the need of publishing it on the App Store or Play Store. To access Melodic it is just necessary to install the app Expo.Go and then scan the QR code exhibited Figure 10. Of course this means that the mobile device must be able to read QR codes. To suit this need it is used a library called *expo-barcode-scanner* that provides the necessary functions.

[1] For more information about React Native consult: `https://reactnative.dev/`
[2] To download Expo consult: `https://expo.io/client`

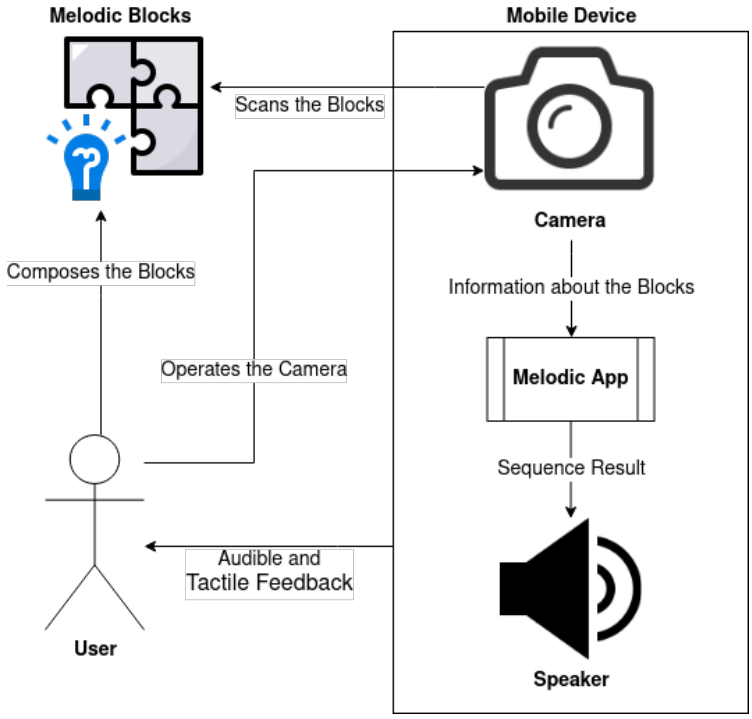

Figure 9 Melodic's System Architecture Diagram.

After the first tests set succeedsswd ed, a new set of tests is being prepared to be experimented with members of **Íris Inclusiva**[3].

Figure 10 QR code to access Melodic.

5 Melodic, using the system

After describing Melodic suite, components and interconnection, this section will provide some examples to illustrate how to utilize Melodic as a Learning Resource to train Computational

[3] For more information about **Íris Inclusiva** consult: `https://irisinclusiva.pt/`

Thinking. An incremental approach to the complexity of problems presented is advised. For this a collection of different exercises can be designed, each one increasing the complexity level:

- **Simple Algorithm**: This type of algorithm consists of only basic instructions. Figure 11 shows an example of a very simple algorithm composed of four instructions: starting block, two musical note blocks and the ending block.
- **Repetition Algorithm**: This type of algorithm is composed of blocks of basic instructions and blocks of repetition. Figure 12 shows an example of this type of algorithm. In this case, in addition to the starting and ending blocks, we have one musical note block in second position, the block with the number three, and then, another musical note block. The block with the number three is intended to play three times the musical note block preceding the repetition number.
- **Changing the Instrument and Speed Instruction**: In this case, the user can create an algorithm with attributes:
  - **Instrument Played**: With the Instrument Blocks the user can change one or more times the instrument played in the sequence (Figures 13 and 14 respectively). When the instrument block is used, the musical notes after the instrument block are played using the sound of the defined instrument;
  - **Speed between notes**: With the Speed Block the user can change the speed between notes at the time of reproduction. This changes the speed for the whole sequence (Figure 15).
- **Iterative Algorithm (Loop)**: In this case the user can program sequences that must be executed one or more times implementing the loop concept. Besides the start and end blocks for the whole algorithm, when the user wants to insert a loop, he must insert also special blocks as follows: start, end, number of iterations and the loop body. In this case, the order of the blocks is very important. Figure 16 shows an example of how to implement a loop sequence. Like all algorithms, the program starts with the starting block. To build the loop, it is necessary to place the Start Loop block, then the number of the iteration, then the body of the loop (composed of several other blocks) and, finally, the End Loop block. The end block determines the end of the algorithm. This type of algorithms is the highest degree of complexity in this system.

The user must feel free to experiment with other block sequences at his will.

To use the system the user must firstly create the required block sequence. With the sequence prepared the user must open the Melodic App and start scanning the blocks. For each one the user must turn it around, scan it, and turn it back over to avoid QR Code scanning conflicts. At the end of the sequence, when the end sequence block is read the system plays the programmed notes.

To listen the melodies resulting from the presented algorithms please access the project Web site at: `https://epl.di.uminho.pt/~gepl/Melodic`.

In order to obtain a melody, blocks must be organized in a correct sequence. Otherwise Melodic will detect and signalize an error. Regarding wrong block sequences, Melodic copes with 2 different situations, namely:

- **Start Error**: This error occurs when the user forget to initialize the sequence of blocks with the *Start Block*. Figure 17 shows a sequence that misses a *Start Block* causing an error message to be thrown.
- **Loop Error**: In this case, the user composes the blocks to create the loop sequence in a wrong way leading to an incorrect syntax. Figure 18 shows a loop error caused by missing the number of iterations.

Figure 11 Example of a Simple Algorithm.

Figure 12 Example of a Repetition Algorithm.

Figure 13 Example of an Algorithm with a Single Instrument Change.

Figure 14 Example of an Algorithm with Multiple Instrument Changes.

To listen to examples of error messages the reader can consult the project homepage at: `https://epl.di.uminho.pt/~gepl/Melodic`.

Figure 15 Example of an Algorithm with Speed Change.

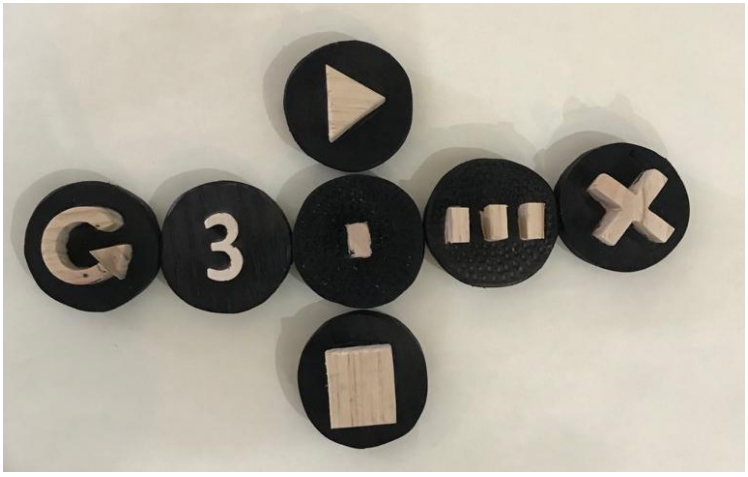

Figure 16 Example of an Iterative Algorithm (Loop).

Figure 17 Example of a Start Block Error.

6 Conclusion

This paper describes Melodic, the motivations behind it and all its features. Melodic is a mixed[4] serious game intended to be used as an Learning Resource to train Computational Thinking adapted to visual impaired people.

[4] In the sense that it is partially unplugged but also it uses a software application to complement its functionality.

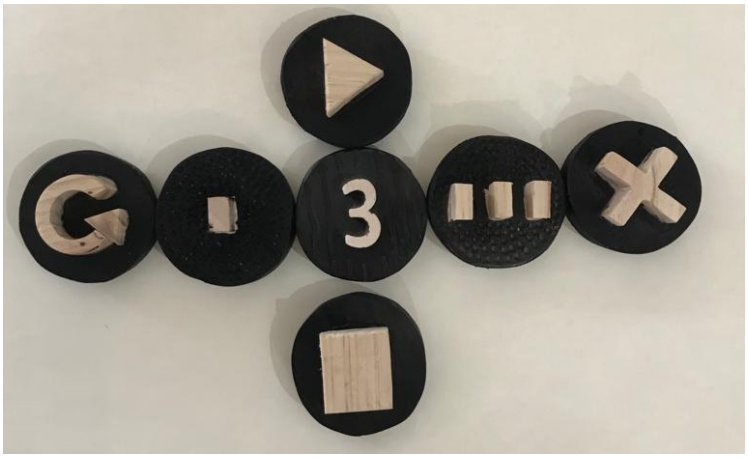

Figure 18 Example of a Loop Creation Error.

The paper starts by an introduction to the Computational Thinking subject and how it is important to the 21.st Century youth. Following this is an explanation of some methods on how to adapt learning resources to the visual impaired community, focusing on young learners and multiple sensory platforms. Next is the introduction to Melodic, it's principle and workflow, explaining how, with the help of tactile blocks specifically designed for this system, we can train Computational Thinking concepts to visual impaired students. Then Melodic's Architecture is presented. In this section the design and all technological details of the system implementation are described. Finally a description of how to use the system illustrated by simple examples is made.

A core component of Melodic is the set of the tactile blocks. This was one of the biggest difficulties of this project due to our lack of knowledge in the area of visually impaired people. It was here that Íris Inclusiva came to help. They gave us a lot of valuable opinions and advises on what should the block's shapes and textures be. This guided us into the final system design described previously in this document.

Melodic is also thought to be available to everybody, being an institution or an individual person, who wants to have a set of blocks for their own. For this, the person can use his creativity and build each block using materials such as: wood, styrofoam, clay, among others. After having all the blocks ready, it is necessary to paste the respective QR codes at the bottom of each block. These QR codes are all labeled in one document accessible online at `https://epl.di.uminho.pt/~gepl/Melodic`.

Melodic is an inclusive Learning Resource because although it is thought to be used by blind or low vision students, it can also be used by students who do not have visual difficulties. This is due to the fact that blocks use a general and intuitive alphabet and do not resort to a blind people specialized one.

In conclusion Melodic aims to help visually impaired people to get in touch with a reality that normally relies on a visual interface, that is programming, enabling them to train Computational Thinking.

In order to prove that Melodic works, a preliminary test was performed. For this Íris Inclusiva was a major help providing the space and the user to be the first visual impaired to test the system.

In this test we could prove that the blocks are easily identifiable (Figure 19) and that the user could in few minutes get very proficient in reading the block's QR codes (Figure 20).

To proceed with Melodic testing, a list of people to test is being developed in partnership with Íris Inclusiva. This tests is a proof of concept of the system proposed.

To access the effectiveness of Melodic, two main issue have to be proven:

- Visual Impaired people can interact with the system;
- The system develops Computational Thinking.

The first one has already been tested and for the second one, diagnostic and final tests are being designed. These tests have the objective to evaluate the student's capability to solve problems before and after using Melodic, to understand if their Computational Thinking skill were improved.

Figure 19 User Finding the Tactile Blocks.

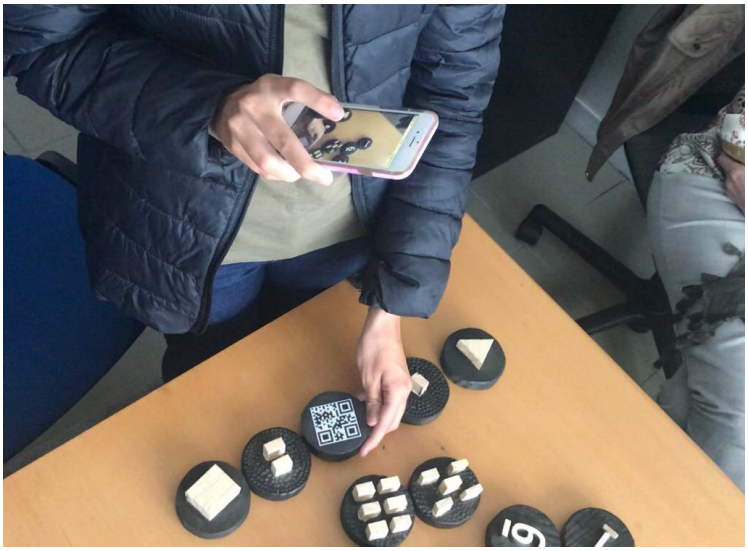

Figure 20 User Reading the Sequence Tactile Blocks QR Codes.

As future work, it is necessary to test the application with more blind and low vision students. For that, we will design and conduct some experiments with impaired members of Íris Inclusiva association. Then, we will make possible changes identified in the testing phase. Finally other more sophisticated experiments will be conducted with other groups.

References

1 Cristiana Araújo, Lázaro Lima, and Pedro Rangel Henriques. An Ontology based approach to teach Computational Thinking. In Célio Gonçalo Marques, Isabel Pereira, and Diana Pérez, editors, *21st International Symposium on Computers in Education (SIIE)*, pages 1–6. IEEE Xplore, November 2019. doi:10.1109/SIIE48397.2019.8970131.

2 Dr. Elizabeth J. Erwin, Tiffany S. Perkins, Jennifer Ayala, Michelle Fine, and Ellen Rubin. "you don't have to be sighted to be a scientist, do you?" issues and outcomes in science education. *Journal of Visual Impairment & Blindness*, 95(6):338–352, 2001. doi:10.1177/0145482X0109500603.

3 Council for The Curriculum Examinations and Assessment. Computing at school: Northern ireland curriculum guide for post primary schools. Technical report, CCEA, 2018.

4 William Fraser and Mbulaheni Maguvhe. Teaching life sciences to blind and visually impaired learners. *Journal of Biological Education - J BIOL EDUC*, 42:84–89, March 2008. doi:10.1080/00219266.2008.9656116.

5 J. Freeman. *The Psychology of Gifted Children*. Wiley Series in Developmental Psychology and Its Applications. Wiley, 1985. URL: https://books.google.pt/books?id=dtB9AAAAMAAJ.

6 Mark Guzdial. Teaching programming with music: An approach to teaching young students about logo. *Logo Foundation*, 1991.

7 N.G. Haring and R.L. Schiefelbusch. *Methods in Special Education*. McGraw-Hill series in education: Psychology and human development in education. McGraw-Hill, 1967. URL: https://books.google.pt/books?id=ADQ1AAAAMAAJ.

8 Jan L. Plass, Bruce D. Homer, and Charles K. Kinzer. Foundations of game-based learning. *Educational Psychologist*, 50(4):258–283, 2015. doi:10.1080/00461520.2015.1122533.

9 Alpay Sabuncuoglu. Tangible music programming blocks for visually impaired children. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '20, page 423–429, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3374920.3374939.

10 Victor Silva, Heleniara Moura, Suelen Paula, and Ângelo Jesus. Algo+ritmo: Uma proposta desplugada com a música para auxiliar no desenvolvimento do pensamento computacional. In *Anais do XXV Workshop de Informática na Escola*, pages 404–413, Porto Alegre, RS, Brasil, 2019. SBC. doi:10.5753/cbie.wie.2019.404.

11 Salete Teixeira, Diana Barbosa, Cristiana Araújo, and Pedro Rangel Henriques. Improving Game-Based Learning Experience Through Game Appropriation. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASIcs)*, pages 27:1–27:10, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ICPEC.2020.27.

12 Stephen Wolfram. How to teach computational thinking, 2016. URL: https://writings.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/.