

A Framework to Assess Melodic Effectiveness in Training Computational Thinking to Visually Impaired People

Rui Costa

ALGORITMI Research Centre/LASI, Univ. Minho
Braga, Portugal
rui.diogo.costa@hotmail.com

Alvaro Costa Neto

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Barretos, Brazil
alvaro@ifsp.edu.br

Cristiana Araújo

ALGORITMI Research Centre/LASI, Univ. Minho
Braga, Portugal
decristianaaraujo@hotmail.com

Pedro Rangel Henriques

ALGORITMI Research Centre/LASI, Univ. Minho
Braga, Portugal
prh@di.uminho.pt

Abstract—Computational Thinking education and training are considered crucial activities to achieve success in later challenges like computer programming. To train the brain to think in logical terms it is necessary to use adequate resources. In this paper we address the difficult problem that is assessing efficacy of a given resource to the Computation Thinking development. To discuss that topic we will use the Melodic System which is a physical learning resource capable of constructing melodic sequences in a logical manner to be used by visually impaired people. Melodic was created aiming at training Computational Thinking through music and sounds and its impact in the learning process must be evaluated. This paper presents the tests and results applied to this intent, while also providing a general and summarized presentation of the Melodic System itself.

Index Terms—Computational Thinking, Visually Impaired Students Education, Inclusive learning, Special Education, Music based Teaching, Assessment

I. INTRODUCTION

Computational Thinking is presented by several authors as a problem solving method [1], [2]. Computational Thinking includes several features, the main ones being: logical reasoning, abstraction, pattern recognition, algorithmic thinking. These skills are supported and improved by a set of attitudes or dispositions, namely: perseverance; collaboration; experimenting with changes and analyzing the result; identifying and correcting errors [1], [2].

But for students to acquire Computational Thinking skills, they must train them using appropriate learning resources. According to Wing [1], for this training to have a great impact, it should be carried out from an early age. There are already many initiatives to promote Computational Thinking and therefore there are also several learning resources adapted to train it. However, the vast majority of these resources are not adapted to teaching blind or low vision children, thus creating barriers to their learning. Actually the most used resources of this kind are: visual and digital (for instance,

Lightbot¹, code.org²); supported on paper, without braille (like *FlorestaBook*, or *ALGO+RITMO*); or robot-based (for instance, *Robot KIBO*, *Robot DOC*. This makes it impossible for blind people to use them. Throughout the review of the literature on learning resources to train Computational Thinking suitable for blind or low vision children, we found very few, but inspiring, resources: *Story Blocks*, *TACTOPI*, *ACCembly*, *Project Torino* [3], and the work of *Sabuncuoglu* [4].

The fact that there is a high deficit of resources for people who are blind or low vision led us to develop a new resource in order to contribute for the education of those students. In order to build a resource suitable and easy to use for blind or low vision people, we decided to look for institutions that work with these special needs kids, so that we could gather requirements and validate the tool along its construction. The only association that responded to our contact and agreed to collaborate was *Iris Inclusiva*. *Iris Inclusiva* was crucial for the development of the platform; in particular the blind student who tested it has provided valuable contributions to the system as it was being built. Thus, Melodic was born.

Melodic [5], is a system composed of software and hardware where the user creates sequences with tactile blocks (the hardware) and then reads them with the mobile application (the software), which converts the created sequence into sound. As it does not use braille, it is a system that can also be used by children who do not have visual difficulties. This system is free and the blocks can be easily built at home.

In this article we intend to present a method of evaluating the effectiveness of using Melodic to train Computational Thinking.

This article is organized into 5 sections. Section II presents the Melodic system with concrete examples of its use. Sec-

¹ Accessible at: <https://lightbot.com/>

² Accessible at: <https://studio.code.org/courses>



Fig. 1. Melodic's blocks used to compose musical sequences.

tion III presents the method for evaluating the effectiveness of Melodic. The analysis of the results achieved in the evaluation of Melodic is presented in Section IV. Finally, in Section V the conclusions and future work are presented.

II. THE MELODIC SYSTEM

Melodic is a multi-platform system³ that aims to support the complex tasks of teaching and learning Computational Thinking, specially targeted at aiding visually impaired and musically inclined people [5]. It is based on a predefined set of physical and tactile blocks that represent musical operations (Figure 1) and a mobile App that identifies the spatial distribution of these blocks to compose equivalent melodies.

A. The Blocks

The blocks that represent musical operations are integral to the Melodic System. Figure 1 shows all currently implemented blocks, ranging from playback and loop operations (1), instrument selection (2), numbering and counting (3), musical notes (4) (beginning with *Dó* and ending with *Si*), and velocity (5). These blocks incorporate two different roles: instructions used by users to program the melody—akin to mnemonics in assembly or keywords and tokens in programming languages—and a translation mechanism, converting from a pure physical representation into an actual melody, communicating to the Melodic App.

Being a system focused on training visually impaired students, Melodic's blocks are constructed in a specific way to facilitate their handling. Several characteristics are important and help users to identify each kind of block:

- **Material:** wood was chosen as standard material for its natural texture, ease of shaping, low cost, and high durability and availability. Although wood is the standard choice, any other kind of easy-to-shape and affordable material could also be used, such as clay as an example. Another possibility would be to 3D print the blocks using

³ Accessible at: <https://epl.di.uminho.pt/~gepl/Melodic/>

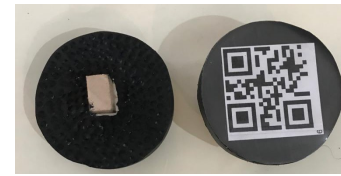


Fig. 2. Top and bottom of a block, showing the QR Code used to identify each block.

different filaments, which would cost more, but generate higher quality blocks;

- **Texture:** block types (numbers, notes and special operations) are carved with different textures to be promptly identifiable by touch;
- **Color:** two contrasting tones of wood (natural or dyed) are applied in the construction to aid identification by low-vision users;
- **Shape:** blocks are formed of 3D wooden icons (shapes) on top of a standard circular base, providing instant feedback as to which specific operation a certain block represented.

The translation from a block's physical entity to a logical representation in the Melodic App is achieved through QR Codes glued to the bottom of each piece, such as shown in Figure 2. The Melodic App asks the user to scan these codes in order to identify each block, who in turn, must take care to scan only one QR Code at a time to avoid misidentifying blocks.

Inclusion was a centerpiece to the design of the Melodic System, both in accessibility and affordability. The design of the physical blocks and the use of low-cost, highly available materials and simple building techniques support this goal in such a way that even without modern fabrication techniques, such as 3D printing, people should be able to construct the blocks by hand and use the system to support Computational Thinking education.

B. The App

The mobile App is the digital counterpart to the implementation of the Melodic System. Its intended use is to scan a distribution of blocks, in an order dictated by the user, and playback either the resulting melody or a message explaining that some error has occurred.

To minimize incompatibilities between different mobile devices' brands and models, Melodic's App was constructed using the React Native framework⁴, which uses JavaScript code to build apps for different platforms.

C. Use and Examples

The use of the Melodic System may be described in three main steps:

- 1) A sequence of blocks is distributed on a surface in a logical manner to form a melody. There is no obligation as to which kind of relative positioning is used (vertical,

⁴ Accessible at: <https://reactnative.dev>

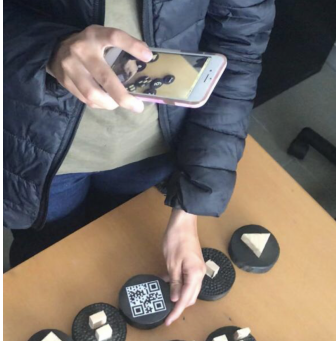


Fig. 3. Blocks scanning process that allows the App to identify the sequence.

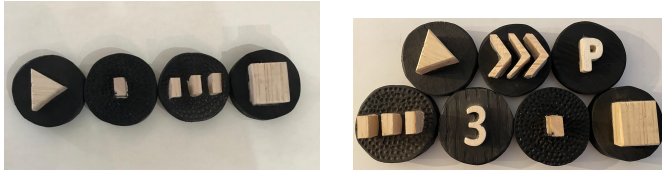


Fig. 4. Melody sequences ranging from simple to complex.

horizontal, tree-like *etc.*), but to avoid errors in the next step, a common rule for distribution may be established by the user;

- 2) Each block's QR Code in the sequence is scanned through the mobile camera (Figure 3), taking care to avoid showing the App more than one QR Code at a time. To this intent, it is useful to attach the QR Codes to the blocks' bottoms and turn them over one at a time;
- 3) The App recognizes the sequence and plays back the resulting melody (or the error message).

In a teacher-guided situation, exercises might be conducted using either pre-recorded melodies or by constructing the required melodies live using the blocks and the App. In the later case, the teacher should require students to only listen to the melody and not show them the sequences.

A simple melody that could be used as an introductory example or exercise is shown at the top of Figure 4. This sequence, scanned from left to right, indicates that a new melody is starting (the *Play* block), then a *Dó* note is played, followed by the *Mi* note. The sequence finishes the melody with the *Stop* block. As long as the user (student or teacher) scans the blocks in the right order (*Play–Dó–Mi–Stop*), the melody should be correct.

Complex melodies may also be proposed as exercises, in order to train advanced Computational Thinking concepts such as change of state (by changing the instruments or the speed) and repetition (through the *Loop* or *X times* blocks). The bottom image in Figure 4 shows a complex sequence that changes the state of the melody via the *Fast* block, defines the instrument to be played with the *Piano* block and applies repetition using the *Three times* block before the *Mi* note.

D. Formal Melodic Language Definition

The fact that a melody is logically created using concepts such as changes of state and repetitions indicates that the Melodic sequences obey an intrinsic language that can be formally defined as such:

Terminals: START, STOP, REPEAT, EREP, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, SPEED1, SPEED2, SPEED3, DO, RE, MI, FA, SOL, LA, SI, GUIT, PIANO, FLUTE;

Productions:

- **Prog:** START Inst+ STOP;
- **Inst:** Note|NoteRep|Speed|Instrument|Cycle;
- **NoteRep:** Note Num;
- **Cycle:** REPEAT Num CycleInst+ EREP;
- **CycleInst:** Note|NoteRep|Speed|Instrument;
- **Note:** DO|RE|MI|FA|SOL|LA|SI;
- **Num:** 1|2|3|4|5|6|7|8|9|10;
- **Speed:** SPEED1|SPEED2|SPEED3;
- **Instrument:** GUIT|PIANO|FLUTE;

This formal definition supports the main objective of teaching and learning Computational Thinking. It proves that a melody correctly constructed through Melodic's blocks is an actual algorithmic implementation, akin to the source code for an application written in a computer programming language⁵. For more detailed information on the Melodic System, see [5], [6].

III. MELODIC'S ASSESSMENT

Melodic's contribution to teaching and learning Computational Thinking was assessed through a direct, progressive and extensive approach:

- 1) Users participated in a diagnostic session in order to assess their Computational Thinking skills before Melodic was introduced (Subsection III-E);
- 2) All of Melodic's components were explained and a series of exercises applied aiming to develop and train their Computational Thinking skills. These exercises ranged from the construction of a very simple, one note sequence, to more complex, iterative and sophisticated melodies;
- 3) Finally, a sequence of Computational Thinking tests was applied in an evaluation session to observe the users' progress.

The tests mentioned in item 3 were carefully selected in order to obtain an overview of how Computational Thinking development occurred as a result of Melodic's use.

A. Computational Thinking Assessment

A structured strategy was defined in order to identify Melodic's contribution to Computational Thinking development. Instead of using Melodic itself as an evaluation tool, which could bias the success rate by repetition and familiarity with the system, the assessment was based on simple logical exercises, designed specifically for this purpose [7]. These exercises were proposed to understand users' reasoning and algorithmic design abilities.

⁵That is not to say that Melodic's formal definition generates a language that is computationally equivalent to a lambda-complete programming language.

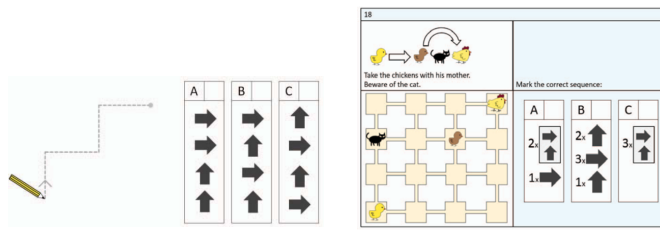


Fig. 5. A simple exercise (left) to identify which of the proposed sequences (A, B or C) would draw the line on the left and a complex exercise (right) to identify a valid path between a source and a destination, while avoiding obstacles (the cat) and attending to required points (the brown chick) (from [7]).

Two different kind of exercises were applied:

- 1) Simple and direct exercises to test pattern matching and algorithmic structures recognition, such as the one shown on the left in Figure 5. This kind of exercises was proposed to evaluate a user's capacity to correlate the reasoning behind the drawing sequence—a line drawn on the left side of the image—and the directions that should be taken in order to draw it—the arrows in columns A, B and C;
- 2) Complex algorithmic reasoning exercises, where a valid path had to be constructed in order to reach a destination. This kind of exercise, as shown on the right in Figure 5 demands users to construct the path from beginning to end positions, without a pre-established guide. The only requirements are that the path should start at the source—the chick—follow only connected cells, avoid obstacles—the cat—go through required cells—the brown chick—and stop at the destination—the hen.

The complexity of each exercise was defined by the presence of elements in the grid, such as obstacles and required cells, and the available options of answers, that could demand recognition of algorithmic concepts such as repetitions and blocks of operations. An example of the latter is shown in option A on the right of Figure 5, where the first two arrows are grouped inside a rectangle—indicating that they form a block of operations—that should be repeated—represented by the preceding “2x” label.

While these exercises were used as basis for testing users' Computational Thinking development after practicing with Melodic, they were not applied in their original form. Since Melodic is a system aimed at training Computational Thinking to visually impaired people, a series of adaptations had to be made in order to reach its intended audience.

B. Adapting the Exercises to Visually Impaired Users

Just as it was done with the blocks, the grid and the answers had to be physically represented through the use of tactile materials, such as wood or clay. For the same reasons stated in Subsection II-A, wood was chosen as the default material for the construction. Its texture, versatility and availability made it the optimal choice to implement the grid, the paths and the elements in the cells. Figure 6 shows the engraved wooden grid

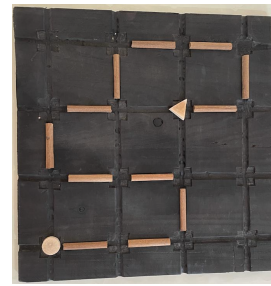


Fig. 6. Example of a simple exercise implementation using the physical grid and its parts.

(board) and the pieces used to represent both the problems and the answers of the exercises. As can be seen in the image, there are chiseled channels that represent the connections between cells, which in turn are indicated by square recesses. There are also wooden pegs (dowels) to construct the path and four wooden pieces (shown in Figure 7): circles for the source, triangles for the destination, squares to define obstacles, and crosses to establish required cells.

All parts were stained (or had their wood species chosen by their natural colors) to present visual contrast, aiming to aid users with low vision, while the shapes and engravings (channels and recesses) were created to support people with total vision loss. Using these physical implementations—the board, the pegs and the pieces—both kinds of exercises from Subsection III-A were adapted and applied.

C. Implementation of Simple Exercises

Exercises of the simple kind were implemented with some adaptations. Presenting multiple choices of answers as shown in columns A, B and C to the left in Figure 5 would be inaccessible to blind users, so the final path was constructed and promptly presented such as shown in Figure 6. In order to complete this exercise, users had to completely describe the path in an algorithmic manner, such as: two times right, up, two times left *etc.*

Users were evaluated by both time to completion and conciseness of the answer, including the use of algorithmic structures and operation blocks (groupings). Three levels of answers were defined:

- *Basic*, for answers that missed all possible groupings of steps and algorithmic structures, being “Left, left, up, right, up, right” an example;
- *Proficient*, where single steps were grouped and repetitions were used with single inputs, such as “Two times left”;
- *Expert*, for answers that grouped multiple steps and used them as inputs to other structures, effectively combining both concepts, as in “Two times (up, right)”.

D. Implementation of Complex Exercises

For complex exercises, users were required to construct the path from a set source, to a defined destination, while also taking into consideration obstacles and required cells. Only

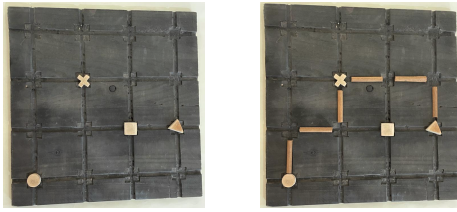


Fig. 7. Complex exercise implementation using all possible pieces. This represents the most complex level of exercise. Initial state on the left and a valid answer to the right.

pieces were initially placed on the board for identification. After examination, the users constructed their answers using the pegs to represent the path.

Four levels of complexity were defined for this kind of exercise:

- *Simple*, in which only the source (circle) and destination (triangle) pieces were used and users were expected to just create the shortest path between both;
- *Avoidance*, where besides the source and destination, some cells contained obstacles (square). Users were expected to create the shortest path that avoided the obstacles;
- *Requirement*, in which instead of obstacles to avoid, required cells (cross) were added. For this level of complexity, users were expected to create the shortest path between source and destination that passed through the required cells;
- *Complete*, in which all possible pieces were used to construct the most complex kind of exercise. An example of this level of exercise is shown in Figure 7.

Similarly to the simple exercise kind, the expected answers should use the lowest possible number of operations (pegs) and apply algorithmic structures, while also taking the shortest time possible to be constructed, counted from the beginning of the users' analysis to the completion of the answer.

E. Assessment Methodology

The complete methodology was composed of a diagnostic session using the wooden grid, followed by Computational Thinking training with Melodic, and ending with evaluation exercises, once again using the wooden grid.

An incremental approach to complexity was adopted and five exercises of each kind were created, from simpler to more complex, both in diagnostic and evaluation sessions. Care had to be taken to keep complexity levels equivalent between diagnostic exercises and their evaluation equivalents, so that Melodic's contribution to Computational Thinking development could be effectively measured. In order to achieve this, evaluation exercises represented rotations or mirroring of their diagnostic equivalents, such as shown in Figure 8 in which the grid was rotated twice counter-clockwise.

Users' evolution was carefully identified between the diagnostic and evaluation sessions in order to compile meaningful results for Melodic's contribution to Computational Thinking development.

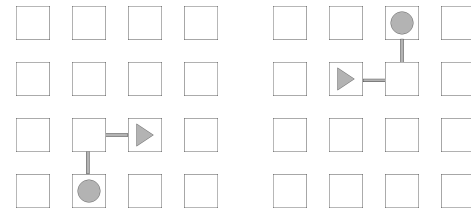


Fig. 8. Rotation of the diagnostic exercise (left) to create the evaluation equivalent (right).

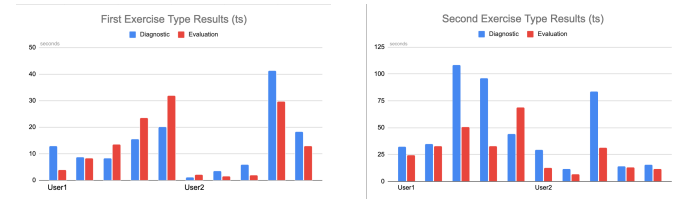


Fig. 9. Execution times for the all ten exercises.

IV. RESULTS

Two users participated in Melodic's assessment, a 15 years old student and an university staff member around 40 years old, both with complete vision loss. Melodic's contribution to Computational Thinking development was evaluated under three perspectives: execution time, answer complexity and number of instructions.

A. Execution Time

Time was measured from the beginning to the end of the exercises resolution for each user. The execution times taken to solve the simple and complex exercises are shown in Figure 9. In general, it is possible to observe that time taken to finish an exercise decreased after Melodic's use (difference from the diagnostic times in blue and evaluation times in red), with just a few inversions. The first user presented worse times in the evaluation session in three simple and one complex exercise, although in average there was a 7% improvement (time reduction) for the simple exercises and 42.6% for the complex.

B. Answer Complexity

Answer complexity was evaluated by asking both users to describe their reasoning to construct the exercise solutions on the board. The goal was to qualify and quantify the logic reasoning followed to build the answer. The explanation was given for the ten exercises—five simple and five complex—and the application of more sophisticated algorithmic concepts was identified. The self-described resolutions were ranked in levels of complexity, as stated in Subsection III-C. Improvements were detected in 15% of the answers for the complex exercises. Simple exercises, given their lower challenge level, ranked equally in both diagnostic and evaluation sessions.

C. Number of Instructions

The number of instructions involved in each exercise solution was counted when users self-reported their reasoning

for the answers, using the same strategy applied to the measurement of answer complexity. Overall, a decrease of 36% in the number of instructions was observed between the diagnostic and the evaluations sessions. This result was only possible through the correct application of operation blocks and repetition. Both facts evidence an increase in Computational Thinking skills.

V. CONCLUSION

It is important to ensure inclusive education and that blind or low vision children have the same educational opportunities as other students. In this way, we must build inclusive resources that allow everyone (blind or not) to be able to use them to learn. Melodic is a free learning resource that anyone can play with or even build. Its purpose is to allow blind people to train their Computational Thinking skills. Melodic is composed of blocks with symbols that have meaning (musical notes, instruments, beginning, end, ...) and a mobile app that allows you to read the sequence of blocks created. After reading the sequence with the app, the sequence is converted into a sound melody.

To assess the efficiency of Melodic, a three-step evaluation approach was created: diagnostic assessment of users' Computational Thinking skills; training of Computational Thinking using the Melodic system; and, finally, a final evaluation to measure the participants' progress. We only got two participants (one young and one adult) to test the Melodic system following the approach summarized above. Melodic's evaluation for Computational Thinking training was appraised through three parameters: execution time spent to finish an exercise; response complexity, considering the answer proposed to the exercise; and number of instructions of the resolution.

Regarding the execution time, we concluded that in general the exercise completion time decreased after Melodic training. The complexity of the responses was assessed by asking users to describe their reasoning for constructing the responses on the board. In the complex exercises, improvements were detected for 15% of the answers. Simple exercises were not detected changes. The number of instructions was calculated when users self-reported their reasoning for the responses. It was concluded that in general there was a 36% decrease in the number of instructions between the diagnostic and evaluation sessions.

However the main contribution of this paper is not the assessment of Melodic, that we recognize to be in a preliminary stage, but is the definition of a validation strategy for learning resources aimed at training Computational Thinking. This is, in our opinion, the focus of the work here reported, to enable us to assessment properly the effectiveness of the tools we are developing to prepare students to solve problems using a computer. That is not to say that the strategy is without flaws. There is some possibility that the evaluation techniques and tools applied to assess Melodic's effectiveness might have biased the results. One feasible way to minimize this effect would be to apply a controlled study, measuring the relative evolution of a group that uses Melodic when compared to a

group that does not. While not a trivial endeavour, this kind of assessment would further strengthen our current strategy.

Throughout this work, we have experienced serious difficulties in finding schools or institutions for the blind to collaborate with us in the development and testing of the platform. In addition, it was also difficult to find participants (blind or low vision) available to collaborate. We are aware that, as development and testing occurred during the pandemic at the end of confinement, this situation influenced the difficulties felt. The feedback from *Iris Inclusiva* and the people who participated was positive, as they considered that Melodic has potential, it is easy to use, and that it helps to train various Computational Thinking skills as well as music.

As future work, we consider that it is necessary and urgent to design and conduct more experiments (with training plans and evaluation tests) to complete Melodic's assessment. It is also important to build usability and feedback surveys. Allied to this, it is necessary to evaluate the effectiveness of Melodic with more users. In order to extend Melodic's potential to teach Computational Thinking, we intend to create conditional and concurrency blocks to play different pieces of music in parallel and train traditional conditional algorithmic structures. Finally, we intend to implement a computer vision module to the App to identify each block by its top, improving usability. Since the blocks have different shapes and contrasting colors, the user would not have to turn the block over to perform the scanning, avoiding the use of QR Codes entirely while facilitating Melodic' use.

ACKNOWLEDGMENT

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

REFERENCES

- [1] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, mar 2006.
- [2] CSTA & ISTE, "Operational Definition of Computational Thinking for K–12 Education," https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf, 2011, accessed: 2021-11-07.
- [3] C. Morrison, N. Villar, A. Thieme, Z. Ashktorab, E. Taysom, O. Salandin, D. Cletheroe, G. Saul, A. F. Blackwell, D. Edge, M. Grayson, and H. Zhang, "Torino: A tangible programming language inclusive of children with visual disabilities," *Human-Computer Interaction*, vol. 35, no. 3, pp. 191–239, 2020.
- [4] A. Sabuncuoglu, "Tangible music programming blocks for visually impaired children," in *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, ser. TEI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 423–429.
- [5] R. Costa, C. Araújo, and P. R. Henriques, "Melodic - Teaching Computational Thinking to Visually Impaired Kids," in *Second International Computer Programming Education Conference (ICPEC 2021)*, ser. Open Access Series in Informatics (OASIs), vol. 91. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 4:1–4:14.
- [6] R. Costa, "Melodic, Using Music to train Visually Impaired Kids in Computational Thinking," Master's thesis, Universidade do Minho, Braga, Portugal, February 2022, MSc dissertation.
- [7] M. Zapata-Cáceres, E. Martín-Barroso, and M. Román-González, "Computational thinking test for beginners: Design and content validation," in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020, pp. 1905–1914.