GoDonnie: A robot programming language for teaching people who are visually impaired

Juliana Damasio Oliveira[1] , **Márcia de Borba Campos**[2] , **Alexandre M. Amory**[1] , **Rafael H. Bordini**[1]

[1]School of technology– Pontifical Catholic University of Rio Grande do Sul (PUCRS)
90.619-900 – Porto Alegre – RS – Brazil

[2]Inedi College – CESUCA –94.935-630 – Cachoeirinha – RS – Brazil

juliana.damasio@acad.pucrs.br, marciabcampos@hotmail.com

{alexandre.amory, rafael.bordini}@pucrs.br

Abstract. *Recent work indicates that many programming environments and languages are difficult for people who are visually impaired to use, which may contribute to excluding them from a professional career in computing. This work aims to create and evaluate the use of a programming language called GoDonnie for students who are visually impaired. We developed a programming language that students use to simulate a robot's behavior. We also introduce two usability studies that were conducted with end-users. The initial results indicate that our programming language is easy to learn by beginners and experienced programmers alike. In addition, users report high levels of enjoyment when programming in GoDonnie.*

1. Introduction

Choosing the initial programming language for students who are visually impaired is an open problem. It has been reported that some languages are difficult to use for this user group [Kane and Bigham 2014, Mealin and Murphy-Hill 2012]. Among the programming languages that present such difficulties, we mention Python, which uses white spaces to delimit code blocks, which is hard to navigate with screen readers [Kane and Bigham 2014, Mealin and Murphy-Hill 2012]. Consider also languages such as C and Java, which use braces to delimit code blocks. In this case, the obstacle is when there is a block with sub-blocks because students who have visual impairment tend to have difficulty in figuring out the matching braces.

There are several programming languages for educational purposes. For example, Logo and its variants, such as NetLogo[1], Kturtle[2], StarLogo[3], and Etoys[4]. However, these initiatives do not address all students, especially those who have visual impairment [Barros et al. 2014]. This is because many programming environments are based on graphical interfaces, which makes them inaccessible to this group of users [Ludi and Reichlmayr 2011, Barros et al. 2014]. Another important aspect is that

[1]http://ccl.northwestern.edu/papers/agent2004.pdf
[2]https://edu.kde.org/kturtle/
[3]http://education.mit.edu/starlogo-tng/
[4]http://www.ufrgs.br/soft-livre-edu/software-educacional-livre-na-wikipedia/etoys-linguagem-de-programacao/

DOI: 10.5753/cbie.sbie.2019.1181

traditional teaching techniques are based primarily on visual models such as diagrams, flowcharts, tables, and images. Unfortunately, this type of resource is not useful for students who are visually impaired unless enabled by specially crafted assistive technology and concrete resources [Al-Ratta and Al-Khalifa 2013].

A person who is visual impairment needs complementary support from non-visual stimuli to perceive the programming environment. This work shows our efforts to create an accessible programming environment for students who are visually impaired. The programming language is called GoDonnie and through language, the student who is visually impaired commands a robot named Donnie. In this paper, we present the GoDonnie language and some concrete teaching strategies explored so far. We conducted two studies to evaluate GoDonnie's usability with 3 participants who are visually impaired. The results are presented in this work.

This paper is organised as follows. Section 2 reviews prior papers about the use of robotics for people with visual disabilities with focus on initiatives to promote Computer Science as a future profession for those students. Section 3 presents the proposed programming language. Section 4 describes the evaluation of GoDonnie. Section 5 presents final remarks and future work.

2. Related Work

We conducted a systematic literature review which was published on [Oliveira et al. 2017]. This research showed that accessible programming environments for students who are visually impaired have already been subject of research, particularly using robots. We introduce the most relevant ones to our study. Ludi et al. [Ludi et al. 2010, Ludi et al. 2014, Ludi and Reichlmayr 2008] developed a software called JBrick, for students who are visually impaired. It provides accessibility features for Lego Mindstorms NXT and was tested with ten participants who were satisfied in using it.

In [Ludi et al. 2014], the authors also identified that blind users have difficulty in using the keyboard to write curly brackets (or braces). Moreover, these keys are not suitable for identifying the end of a programming block (including selection and loop blocks). Howard et al. [Howard et al. 2012, Park and Howard 2013] also use a Lego Mindstorms NXT robot kit that was integrated with JAWS screen reader and MAGIC magnifier. Multiple haptic and auditory signals were used to transmit messages during the robot interaction in the environment. Their results showed that people who are visually impaired can program a robot if extra sensory feedback is provided.

Another approach for teaching programming mobile robots to people who are visually impaired is to use cubic blocks and RFID tags, allowing operation via tactile information [Kakehashi et al. 2013, Kakehashi et al. 2014, Motoyoshi et al. 2015]. In this approach, students can control a mobile robot by positioning wooden blocks on a mat. This system is called P-CUBE and it was used to operate a robot composed by an Arduino UNO microcontroller board, a wireless microSD shield, a buzzer, two motors, a gearbox, and batteries. It was tested with four people who are visually impaired and was improved with their feedbacks. Recently, P-CUBE was also experimented with Arduino sketch, which is a programming tool. It was demonstrated to be useful programming education tool for beginners [Kakehashi et al. 2013, Motoyoshi et al. 2015].P-CUBE tactile

approach is promising, but the programming commands are very limited. The systematic review was replicated in 2019. We highlight the publication [Tsuda et al. 2018].

3. GoDonnie

The GoDonnie[5] programming language was developed as an educational tool, for the teaching of programming to students who are visually impaired. For this, Papert's fundamentals were used, like the Logo approach [Solomon and Papert 1976] as guidelines for the creation of commands, and the way of executing them. Thus, we developed a programming language that students use to simulate a robot's behavior in a virtual environment; the behavior is described to the user through audible messages. Among GoDonnie's creation, Logo's main contribution was the use of natural language to move an object in a scenario. Thus, commands have been inserted to move and to rotate the robot. GoDonnie also has the selection, repeat, procedure, and assignment commands that are common to general-purpose programming languages.

However, it lacked a better integration with screen readers, and commands that avoided the use of simultaneous keys, as well as commands that would allow the user to better understand the scenario, how the robot relates to that environment and moves in it. From these limitations, specific commands were defined for the end-users such as SCAN, COLOR, POSITION, and STATE. Also, changes were made to the syntax of existing commands in Logo. For example, in LOGOWritter, SuperLogo, NetLogo, and Lego-LOGO, the IF conditional command and REPEAT loops have brackets to delimit start and end blocks. This choice makes the command less intuitive when using screen readers. GoDonnie avoid the use of symbols that require the use of more than one key (e.g., the *SHIFT* key). In this version of GoDonnie, due to implementation constraints, the *SHIFT* key is kept for the multiplication and addition mathematical operations, and double quotation marks for transmitting text to Text-To-Speech (TTS) using the SPEAK command. In GoDonnie, it is not necessary to indent the code. It is also possible to write the code on the same line, without using the enter key. Finally, GoDonnie language is not case sensitive. The commands used in this work are described below:

- Movement, Rotate and Wait Commands:
  FW n: moves the robot forward for *n* steps.
  BW n: moves the robot backwards for *n* steps.
  TR d: rotates the robot *d* degrees to the right. There is no robot displacement.
  TL d: rotates the robot *d* degrees to the left. There is no robot displacement.
  WAIT n: waits *n* seconds.
- Selection Commands:
  IF...THEN...END IF: selects the code block to be executed based on the boolean value of an expression.
- Loop Commands:
  FOR...DO...END FOR: this command has three parts, namely variable initialization, loop condition, and incrementing or decrement the initialized variable; while the condition is true the robot executes the code block.

[5]The language and system are available at `https://donnie-user-manual.readthedocs.io/en/latest/docs/godonnie/index.html`. The language commands are in Portuguese and English.

REPEAT t TIMES...END REPEAT: repeats the code block *t* times.
WHILE...DO...END WHILE: repeats the code while the condition is true.

- Position and Perception Commands:
  COLOR c: returns the number of objects of a given color *c* (blue, red, green).
  SCAN: scans objects within 180^o in front of the robot, then it returns the color, the distance, and the angle to the detected objects.
  DISTANCE d: the robot has six range sensors to measure the distance from obstacles; this command returns the distance of the sensor *d* (front; frontal right; frontal right; back; back left; back right) to an object in steps (one robot step equals 5 cm).
  STATE: returns a series of current information about the robot such as its current pose and the last instruction.
  POS or POS k: returns all coordinates (X,Y, and angle A) or only the indicated *k* (*X*, *Y*, or *A*).
  HISTORY: returns the number of lines of the program; the user can navigate in the code by typing the line number or using the keys *P* to move to the next line, *A* for the previous line, and ESC to exit the history.
- Audio Commands:
  SOUND on or off: turns on or off the robot's audible messages.
  SPEAK v or "w": speaks a content of a variable *v*, phrase or word *w* using a Text-to-speech (TTS).
- Procedure and Variable Commands:
  PROCEDURE...DO...END PROCEDURE: it implements procedures with an arbitrary number of arguments.
  VAR v: creates a variable *v* to store a value or the result of an expression.

The environment has two types of feedbacks: spoken messages that run with Google's TTS feature and iconic sounds (sounds of steps, turns, and collisions) that are mp3 files loaded automatically into the system. Some commands combine feedbacks of spoken messages and iconic sounds, and others only spoke messages. In addition, the Orca screen reader that is native to the Linux operating system is also used. Orca reproduces what the user types in the terminal.

4. Evaluation Details

The Donnie Programming Environment (DPE) includes an interpreter (Client), an editor where GoDonnie is typed, and an 2D robotics simulator. This section describes the evaluation of the GoDonnie programming language without DPE and some strategies to teach it. The main objective was to assess whether the language would be easy to learn by beginners (Study A) and experienced programmers (Study B), as well as whether the concrete materials used for teaching were easy to understand. These studies occurred only using GoDonnie programming language; it is necessary to isolate problems related to the programming language from its environment (DPE).

4.1. Study A

The experiment was performed with a participant referred to as P1. P1 is male, 23 years old, and a student at a Psychology graduate school. He has no previous programming or robotics experience. Activities with different levels of difficulty were developed to evaluate the GoDonnie, so that the commands of the language could be validated. The list contained exercises divided by levels of difficulty, for instance:

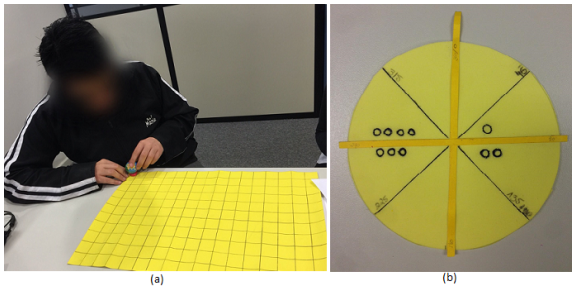

Figure 1. (a) Participant who is blind using the tactile map and the object representing the robot, (b) Angle concept experiment

- Easy: write a program for a robot to walk forward 20 steps and to return to the initial point. Expected result: FW 20 BW 20
- Intermediate: write a program to say whether a number is positive or negative. Expected result: CREATE n = 3 IF n > 0 THEN SPEAK "positive" ELSE "negative" END IF
- Hard: make the robot walk forward 20 steps; every 2 steps, make it say something. Expected result: REPEAT 10 TIMES FW 2 SPEAK "hello" END REPEAT

Initially, a small object was used to represent the robot, as well as a tactile map, produced in EVA (Ethylene Vinyl Acetate) material (Fig. 1a). From these resources, programming and robotics concepts were introduced. This option has been adopted so that the user could validate these concepts while touching them. Thus, it was also evaluated if the user who is blind could understand initial programming concepts using GoDonnie.

During the evaluation, we recorded (i) whether the commands were easy to learn and to remember their uses; (ii) the comments made by the users; and (iii) the notes about the teaching material. To assess whether the P1 remembered the use of the commands, it was asked whether each command was easy to learn and, in the next session, if the commands were remembered. Another questionnaire was used to check if the previous answers by the user about commands were related to problems of syntax and/or semantics. Commands not recalled by the P1 were reworked based on the questionnaire responses. In the sessions, we recorded every movement the P1 made, including the movement in the virtual environment as well as the robot on the tactile map.

We also design another questionnaire to check concepts related to angles. For this assessment we use EVA material and paperboard as shown in Fig. 1b. That was necessary to explain the concepts of angle degrees to the P1. The GoDonnie language evaluation occurred in meetings and through assignments.

- meetings: with the presence of 3 examiners in a controlled environment. One examiner conducted the assessment while the other two helped to control time, record videos, and take photos. This type of meeting occurred on different dates. We had to explain the commands and simulate them on the tactile map, performing tasks of increasing levels of difficulty.
- assignments: this part of the assessment took place without the presence of examiners. The P1 was asked to do the assignments where they had to review the commands in controlled environment sessions; without the use of a tactile map. The P1 send the completed assignment by email for us.

Table 1. Commands seen in each session

Session	Type	Taught/reviewed commands	Commands not remembered
1	Meeting	FW, BW, TR, TL, SOUND, COLOR, DISTANCE, SCAN, SPEAK, WAIT, POS	–
2	Meeting	STATE, HISTORY, REPEAT, IF	TR, TL, WAIT
3	Meeting	CREATE, FOR	REPEAT, IF
4	Assignment	FW, TR, TL, BW, SPEAK, WAIT, REPEAT, IF	–
5	Assignment	FW, BW, SOUND, DISTANCE, POS, SPEAK, IF	–
6	Assignment	CREATE, OPERATORS, SPEAK, IF	–
7	Meeting	WHILE, PROCEDURE	FOR

4.1.1. Results

We conducted 7 assessment sessions in a period of one month, 4 of them were meetings and 3 of them were assignments (Table 1), and each meeting session lasted an average of 1h20min. In the first session was conducted as a meeting that aims to explain the robot, the tactile map, and basic forms to locate and move the robot on the scenario. It includes explanations of the Cartesian plane and the validation of the concept of angles; this is important to check the rotation of the robot. The explanation of degree concepts was necessary because the P1 used to express directions as hours in a clock instead of degrees. Examiners gave feedback during the exercise, sending sound signals, successful messages, and alerts; the user chose to perform the activities without writing down what he was doing. After every command given by the P1, the location of the robot on the tactile map was changed so that the user could perceive the robot's movement. We will show below the most relevant results in each session:

- Session 1: all commands were considered easy to learn. P1 suggested to change the angle conferencing instrument (Fig. 1b), requested a marking to identify each quadrant of the circle. He also suggested improvements to the tactile map to have emphasis between the rows and columns delimiters. P1 suggested that the COLORS command could be replaced by COLOR since it only indicates the existence of objects of a given colour. Fig. 1a depicts the P1 carrying out an activity.
- Session 2: P1 did not initially remember the syntax of two commands that were related to the rotation, and WAIT. These commands were reviewed. P1 considered the commands STATE, HISTORY, and REPEAT easy to learn and the IF command intermediate of difficulty to learn. P1 had requested for examples of real applications in which these commands could be applied; being interested and motivated to solve more and more challenges. P1 continued to suggest changes in the angles conferencing instrument so that the markings were improved. The tactile map was approved.
- Session 3: we taught to create variables using the command CREATE and to repeat using the command FOR, which were considered easy to learn by the participant. P1 started to write the commands in the text editor while using the *Think-aloud* technique. P1 made use of the tactile map during the activities execution. We noted that the participant was amused by simulating the robot with his own body, indicating that he understood the use of this command.
- Session 4: 13 activities were sent to review the commands worked on the previous

sessions. The assignment was composed of 5 easy, 6 intermediate, and 2 hard level activities. Only one hard level activity was not completed successfully. P1 was motivated to perform the activities and was satisfied with his learning in the programming area.

- Session 5: 19 activities were successfully performed by P1:
  - 3 program development activities (2 easy and 1 hard level).
  - 3 activities to verify correction of programs (2 easy level, carried out partly correct, and an intermediate level).
  - 13 activities to correct command syntax (intermediate level). He proved to be motivated and requested a new list of exercises.
- Session 6: P1 carried out 10 program development activities, with no difficulty level indicators (the P1 was not aware of the difficulty level of each question). P1 finished 6 activities successfully; 2 activities were resolved partially; and 2 have not been carried out successfully. The errors were related to the variable assignment in which the P1 tried to assign characters to integer variables.
- Session 7: P1 remembered the CREATE command, but did not remember the FOR repeat command. The primary goal was to teach another repetition command WHILE; after that, the creation of PROCEDURE. The P1 considered the WHILE command intermediate level of difficulty to learn and the PROCEDURE was easy. P1 solved the activities using a text editor. He was motivated by the possibility of teaching new commands to the robot and he was aware of the increasing complexity of each activity.

4.2. Study B

We have intentionally invited two people who are visually impaired to participate in this experiment; one who is low vision (P2) and the other who has acquired blindness (P3). P2 was 44 years old and P3 was 39 years old during the period of the experiments. Both participants were male. Considering the programming experience, P2 has an intermediate level of knowledge in C, C++, Python, Java, Pascal, and Delphi programming languages; P3 has a basic level of knowledge in Java and C programming languages. Although both participants have programming skills, they did not develop code frequently. For this study we also use meetings and assignments. We created five guided activities for the meeting session. These activities are direct, application-oriented, and problem-oriented with different levels of difficulty. After each activity, there was a discussion about it. In the following we introduce an example of easy guided activity.

- Make the robot move around and return to the original position. The robot is in the position of axes x = 0 and y = 0. Command suggestion: move the robot forward, during 3 steps. After that, rotate the robot to the right 180 degrees and move the robot forward during another 3 steps. Expected result: FW 3 TR 180 FW 3
- Discussion: try to complete the challenge using the left-turn command. Expected result: FW 3 TL 180 FW 3. After that, try to complete the challenge without using turn commands. Expected result: FW 3 BW 3

For the assignment, sent by email, we created a list of 19 programming questions, which were organised into blocks of commands. After each block, we asked questions related to the usability of the programming language and the user's satisfaction. The meeting session process was the same for P2 and P3; each one done individually. We

conducted two evaluations with P2, a meeting, and an assignment sent by email. Related to P3, we conducted only one meeting, since no response for the assessment was given by email. During the meeting session, we presented GoDonnie commands using the tactile map and the robot simulator object.

The participants performed 5 guided programming activities (step-by-step). They had the tactile map available to use during the programming activities. After that, we conducted a structured interview about the programming language and the performed activities with participants. Finally, they received another list of programming activities to be performed, along with a GoDonnie guide; the answers were sent by email.

4.2.1. Results

The activities related to meeting session were successfully performed by P2 and P3. We noted that P2 easily remembered the syntax of GoDonnie commands, but had a bit of difficulty to formulate logic expressions (e.g., for comparing values). On the other hand, P3 had no difficulty to formulate logic expressions; however, he sometimes forgot the syntax of GoDonnie commands. In the latter case, we provided tips about the commands to P3; then he decided which command to use and how to make it work. After performing the guided activities, we conducted a structured interview to evaluate the comprehension of the commands and activities. Both participants agreed that GoDonnie was easy to understand. They think the programming language is robust and it does not miss any other commands. They also considered the proposed activities appropriate. At this stage, they had no suggestions about the activities and the language guide.

The assignment goal was to review and reinforce the commands so the participants could solve problems without the presence of the examiners. We delivered to each participant a copy of the language guide, and we explained for each task of the activities where, in the guide, the users could find information about the required command. The questionnaire was organised into exercises and evaluation questions about usability criteria, such as: ease of learning, ease of use, and appropriateness of the command to its syntax and action. P2 considered most of the commands easy to learn, to use, and appropriate to the action performed. The COLOR command was considered partially easy to learn because initially P2 did not use the parameters of this command. Then consider this command easy to use and appropriate. arithmetical operators were considered unsuitable, although they were considered easy to learn and use. The suggestion of P2 was that one could include precedence among these operators.

Another issue highlighted by P2 is that of procedure blocks could be signaled with the terms DO and END, as in the WHILE and FOR commands. At this stage of the evaluation, the procedure creation command PROCEDURE was defined with BEGIN and END, and the other commands (FOR, WHILE, REPEAT) already used the terms DO and END. At the moment PROCEDURE follows the pattern of these commands.

5. Discussion and future work

Among usability criteria, from the point of view of students who used GoDonnie, results showed that GoDonnie's commands are intuitive, easy to use, and easy to remember. From our point of view, evaluations showed that GoDonnie is a suitable language to teach

programming to those who do not have prior programming knowledge. Also, meeting the needs of problem-solving by those who already had prior knowledge in programming.

Regarding GoDonnie's accessibility and teaching strategies, the study showed that it was important to use the language before to use the programming environment. Thus, the emphasis can be given to the commands, regardless of the environment in which they were used. In the experiments, text editor and screen reader was used, according to the preference of each participant, resulting in a better fixation of the GoDonnie commands. By isolating the environment-related variable, and we provide the feedbacks simulating the system, P1 can focus exclusively on how to learn to program, how to use the commands and when to combine them. Still, it allowed him to imitate the robot, using his body and moving in space, as the robot would do in the virtual environment.

Also, it is important to highlight that there was an improvement in the identification of the university spaces and how to move through different spaces by P1. According to P1, before using the GoDonnie he moved by the university with the help of a sighted monitor, without worrying about issues of orientation and mobility. After using GoDonnie, P1 started to position itself in the position of the robot thinking about the steps and changes of direction that needed to be made to move from a starting point to the final point. He also learned about the distance between objects and between the buildings of the University, and that there were different possibilities for reaching the same destination. This autonomy and maturity in factors of spatial orientation were perceived not only by P1, but by those who conveyed it (family, teachers, and colleagues).

Although further evaluation is still required to demonstrate the effectiveness of the language. We suggest that an evaluation process of a programming language that manipulates a robot, whether virtual or physical, includes: concrete materials to explain programming and robot concepts; guided programming tasks and also with difficulty levels such as easy, intermediate, and hard. It is interesting to apply questionnaires on the use at the end of activities, such as those proposed in this study. The next steps include the usage of GoDonnie and the assessment of the virtual environment, with a robotic simulator, and a haptic belt. We also plan to evaluate if the environment helps the students to improve the orientation and mobility (O&M) and programming skills.

Acknowledgments

The work was financed by CNPq - 20/2016-6 - 442126 and CAPES - Finance Code 001.

References

Al-Ratta, N. M. and Al-Khalifa, H. S. (2013). Teaching programming for blinds: A review. In *International Conference on Information and Communication Technology and Accessibility*, pages 1–5.

Barros, R. P., Torres, V. P., Burlamaqui, A. M. F., and Natal, R. (2014). Cardbot: Tecnologias assistivas para imersao de deficientes visuais na robótica educacional. In *Workshop de Robótica Educacional*, pages 11–16.

Howard, A. M., Park, C. H., and Remy, S. (2012). Using haptic and auditory interaction tools to engage students with visual impairments in robot programming activities. *IEEE Transactions on Learning Technologies*, 5(1):87–95.

Kakehashi, S., Motoyoshi, T., Koyanagi, K., Ohshima, T., and Kawakami, H. (2013). P-cube: Block type programming tool for visual impairments. In *Conference on Technologies and Applications of Artificial Intelligence*, pages 294–299. IEEE Computer Society.

Kakehashi, S., Motoyoshi, T., Koyanagi, K., Oshima, T., Masuta, H., and Kawakami, H. (2014). Improvement of p-cube: Algorithm education tool for visually impaired persons. In *IEEE Symposium on Robotic Intelligence In Informationally Structured Space*, pages 1–6.

Kane, S. K. and Bigham, J. P. (2014). Tracking stemxcomet: teaching programming to blind students via 3d printing, crisis management, and twitter. In *ACM Technical Symposium on Computer Science Education*, pages 247–252.

Ludi, S., Abadi, M., Fujiki, Y., Sankaran, P., and Herzberg, S. (2010). Jbrick: Accessible lego mindstorm programming tool for users who are visually impaired. In *ACM Conference on Computers and Accessibility*, pages 271–272.

Ludi, S. and Reichlmayr, T. (2011). The use of robotics to promote computing to pre-college students with visual impairments. *Trans. Comput. Educ.*, 11(3):20:1–20:20.

Ludi, S. A. and Reichlmayr, T. (2008). Developing inclusive outreach activities for students with visual impairments. In *ACM Technical Symposium on Computer Science Education*, pages 439–443.

Ludi, S. L., Ellis, L., and Jordan, S. (2014). An accessible robotics programming environment for visually impaired users. In *ACM Conference on Computers and Accessibility*, pages 237–238.

Mealin, S. and Murphy-Hill, E. (2012). An exploratory study of blind software developers. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 71–74.

Motoyoshi, T., Kakehashi, S., Masuta, H., Koyanagi, K., Oshima, T., and Kawakami, H. (2015). The usefulness of p-cube as a programming education tool for programming beginners. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 297–300.

Oliveira, J. D., de Borba C., M., de Morais A., A., and Harb M., I. (2017). Teaching robot programming activities for visually impaired students: A systematic review. In *International Conference on Universal Access in Human-Computer Interaction*, pages 155–167.

Park, C. H. and Howard, A. (2013). Engaging students with visual impairments in engineering and computer science through robotic game programming (research-to-practice). In *Annual Conference and Exposition*, pages 1–14.

Solomon, C. J. and Papert, S. (1976). A case study of a young child doing turtle graphics in logo. In *National Computer Conference and Exposition*, pages 1049–1056.

Tsuda, M., Motoyoshi, T., Sawai, K., Tamamoto, T., Masuta, H., Koyanagi, K., and Oshima, T. (2018). Improvement of a tangible programming tool for the study of the subroutine concept. In *International Conference on Computers Helping People with Special Needs*, pages 611–618.