Instructional Application for Programming and Algorithmic Self-Learning

A Didactic Approach with Mobile Robotics as Pedagogical Context

Pedro G. Feijóo-García[1] and Fernando De la Rosa[2]
[1]*Systems Engineering Program, Universidad El Bosque, Bogotá D.C., Colombia*
[2]*Systems and Computing Engineering Department, Universidad de los Andes, Bogotá D.C., Colombia*

Keywords: Programming Learning, E-Learning, Programming Teaching, E-Robotics, Instructional Technology, Visual Programming Language.

Abstract: One of the main challenges related to algorithmic and programming teaching with novice students, is to focus their process on acquiring concepts and developing problem solving skills in programming, without spending time overcoming syntax-oriented learning curves of specific languages. The application here explained is proposed as an instructional technology that, using the advantages of Visual Blocks Programming, through virtual and remote mobile robotics' scenarios, seeks to give playful and friendly mechanisms for programming and algorithmic self-learning. This paper presents the pedagogical design and approach of the tool, evaluated through a User Experience approach with high school students in the Colombian educational context.

1 INTRODUCTION

In the last decade, there has been of projects focused on designing educational tools to teach and enhance programming and algorithmic skills. Several solutions propose games' design, narratives' development and other types of approaches, offering the student the possibility to learn to program using powerful visual and interactive mechanisms. However, even if these solutions offer novel pedagogical approaches, they are usually limited because they need the presence of a trained tutor to articulate them in the student's process, most of which are applications that even depend on special technical configurations. Many of these solutions are complements to a programming course, which means that the student lacks an environment that favors self-learning scenarios.

Although there has been global advance in the design and development of new instructional technologies emphasized in programming and algorithmic teaching, within the Colombian educational context, these technologies' promotion and elaboration have not been encouraged as they should. Programming and algorithmic skills are unjustifiably delegated to higher education in the national educational scheme. This situation jeopardizes students' motivation towards studying professional careers related to Information Technology; careers that nowadays are highly demanded within the local and international industry.

Through this article, we present the conception, design, implementation, and validation of *RoBlock;* a Web application proposed to offer self-learning environments for algorithmic and programming concepts for high school students. This, to answer the following research question:

Is it possible to design an instructional application, contextualized in mobile robotics and in a self-learning visual programming approach, to motivate high school students to learn to program autonomously?

This paper has the following organization: section 2 presents the theoretical framework that supports this study; followed by section 3 which presents related works and technologies. Later, *RoBlock's* modules' design is presented in section 4, justifying it with the pedagogical strategy described in section 5. Section 6 describes the main results obtained; and finally, the conclusions and future work are exposed in section 7.

Feijóo-García, P. and De la Rosa, F.
Instructional Application for Programming and Algorithmic Self-Learning.
DOI: 10.5220/0006690002300237
In *Proceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018)*, pages 230-237
ISBN: 978-989-758-291-2

2 THEORETICAL FRAMEWORK

2.1 Behaviorism and Constructivism

Behaviorism is a trend that was quite popular in the early twentieth century, with a peak of popularity in 1958 when Skinner and Holland included the principles of the reinforcement theory in their first programmed learning course (Galvis, 1992).

This theory proposes that teachers may try to bring students from a first state (a starting knowledge base), to a higher one, where there is a knowhow or skill they should acquire. Different paths can achieve this, but one of these, is through the programming of instructions along the teaching process. By mentioning "programming of instructions", we refer on how the student's learning process is carried out through a series of pedagogical stages, designed to guide the student from a reference knowledge base, towards a level in which there is a new knowledge she/he does not have at that specific moment.

What makes Behaviorism a programming approach, is the treatment given to the student to guide her/him from the starting point where she/he is, to the higher knowledge stage desired. The basic behavioral principles are listed below:

- An individual manages to learn by observing the consequences of his/her actions.
- Reinforcements are those consequences that strengthen the possibility of repeating an action.
- The more reinforcement the student receives, the more likely she/he is to perform the desired action successfully.
- The absence or delay of the reinforcement after an action, limits the chances to repeat it.
- Differential reinforcement can gradually shape a student's learning behavior.

Unlike Behaviorism, it is fundamental in Constructivism to understand the subject that learns. This involves the understanding of his/her vital field, as well as the interaction between his/her environment (the influence context) and what the subject has been, is, and desires to be. When referring to the vital field of the student, the theory refers to the student's understanding of his/her environment, formed by his/her past, present and future (Galvis, 1992).

Within this theory's approach, the learning process is considered born with the creation of knowledge through interaction with environments that give or allow the exploitation of curiosity, experiential experiences and the enhancement of the student's imagination. In other words, it focuses on providing the learner with the ability to learn while, at the same time, he/she constructs his or her own mental models.

Based on this approach, Constructivism declares that the learning obtained by a student, is not the result of a predefined structure of operations (operation is an internalized action that changes the object of knowledge). According to this, the structures construct based on the learning process that the student carries, considering that the student does not evolve in his knowledge through the association of knowledge, but through the assimilation of new mental models.

2.2 Mobile Robotics as Pedagogical Context

Mobile Robotics is a branch of knowledge that is oriented in the study associated with machines that can move on land, through the air or in the water; spatially in two or three dimensions (Matarić, 2007).

For this study, it is interesting to use this pedagogical context for programming and algorithmic teaching, because it incorporates concepts that involve the automation of activities and decision making, whilst considering the interaction of a Robot (physical device or machine) with its environment using sensors of different types (Feijóo & De la Rosa, 2016).

An interesting characteristic of Mobile Robotics, is the possibility to build a virtual programming environment, consistent with a respective physical environment. This, considering that both scenarios are, as far as possible, equivalent, if and only if, both respect the behavior of elements considered and keep the similarity of the Robot's positioning in a two or three dimensions' environment.

A virtual programming tool allows multiple students to interact and learn simultaneously. Furthermore, the proposal of a virtual environment, developed considering the operation of a real robot, offers the possibility to consider problems and solutions that could be raised equally in an experimental environment with a physical robot.

3 STATE OF THE ART

3.1 Academic Background

Aggarwal et al. (2017) conducted a research to determine the effectiveness and usefulness of the

Microsoft's Kodu Game Lab to support students' learning for programming. For their study, they worked for two 90-minutes sessions, with two groups of elementary students, obtaining and comparing results from the usage of Kodu's tiles and flashcards (first group), and from the usage of paper sheets with color prints of the design patterns considered for the learning process (second group). Both groups were composed by students without previous programming knowledge, assessing them before and after the study. The results of this research present the benefits and the drawbacks of using physical manipulatives in different scenarios. On one hand, the students who did not interact with the tiles showed to had acquired a better understanding of the rules' execution from their interaction with the programming environment. On the other hand, students who interacted with the tiles, demonstrated a better understanding about rules' syntax and construction.

Grover and Basu (2017) present the design and development of assessments items to measure students understanding in introductory CS courses, by answering the following research question: "How can learning outcomes for computing constructs such as variables, expressions (arithmetic and logical), and loops, be organized into a structured assessment framework and measured with technical quality?". They worked with 100 students from different middle school courses, applying an assessment framework that was developed following the Evidence-Centered Design (ECD). From their study, they conclude that, even though Visual Blocks Programming (VBP) simplifies programming syntax, there are still conceptual difficulties towards using and understanding key structures in programs, such as variables, conditionals and loops. Finally, they invite to put additional effort on pedagogical strategies, especially on formative and summative assessments, to measure students' understanding and misconceptions, looking forward to refining pedagogy and curricula.

Weintrop and Holbert (2017) present findings from a study in which they test how learners use a dual-modality environment, having the possibility to choose to work either a Visual Blocks Programming (VBP) approach, or a text-based approach. *Pencil Code,* the proposed environment, is used to understand which is the modality preferred by learners, and why they move from one modality to another in a same project. From this study, the authors conclude that the dual-modality approach is effective for programming learning, considering that all the participant students completed successfully the programming assessments. Furthermore, they indicate that blocks are useful to introduce a new programming environment, as well as support items for conceptual comprehension.

Paramasivam et al. (2017) performed a research applying end-user programming tools for functional robots in Computer Science education, presenting results according to a week-long introductory workshop, in which eleven students with different disabilities programed a Clearpath Turtlebot. For their workshop, they used an end-user programming tool (EUP) named *CodeIt*, using a text-based interface rather than a visual blocks programming approach, to increase accessibility for students with less motor skills and visual disabilities. Their findings report that EUP tools can be used to create advanced robotics platforms, accessible and useful for novice programming students. Furthermore, they indicate that the pairing of robotics with EUP tools, enhance students' confidence and interest towards programming and Computer Science topics.

Gucwa and Cheng (2017) present a methodology to create challenge problems, using simulation environments for hardware robot-based programming competitions. For their proposal, they center on the *RoboPlay Challenge Competition*, which involves *Linkbots* and *RoboSim* as hardware and simulation technologies. The authors argue that this competitive context offers a unique opportunity for students to apply learned skills. Furthermore, they conclude that tools like *RoboSim*, are useful to let students and teachers to work with robots, without the need of setup of physical hardware. They finally find that the students' response to the competition context with *RoboSim* is positive, mainly because of the opportunity this tool gives towards rapid code improvement and validation, arguing that virtual scenarios let students gain effective and useful feedback.

3.2 Similar Technologies and Languages

The technologies and languages presented here are previous solutions that have characteristics like those of *RoBlock*:

- Scratch: Tool designed for people with no notions of programming, for the design and elaboration of 2D video games and animations (MIT Media Lab & Lifelong Kindergarten Group, 2006). This, using Visual Blocks Programming as a playful mechanism of interaction for learning. Scratch is Web, free

and offers users' profiling through authentication mechanisms.

- RoboBlockly: This is a web based tool that uses Visual Blocks Programming in a robot-approach context. With this tool, students program a robot in a simulated environment, enhancing skills towards computing, science, technology, engineering and mathematics (C-STEM). The visual blocks can be downloaded as C++\C code with the Ch interpreter, letting students use their code with Lego Mindstorm and Linkbot robots (Frankie et al., 2017).
- RoboSim: This is a standalone robot simulation environment that allows students to program virtual Linkbot and Lego Mindstorm robots, working with the C++\C interpreter Ch. The programs that students create with this environment can be used to control physical robots (Gucwa and Cheng, 2017).
- Pencil Code: This is a tool that uses a Visual Blocks Programming approach to teach programming through art, music and stories elaboration. Its main characteristic is that, differently from other approaches, this tool offers a collaborative scenario for learning. Additionally, the tool offers the possibility to switch from a visual approach to a textual one, which enhances programming skills from algorithms design to syntax understanding (Weintrop and Holbert, 2017).

4 MODULES' DESIGN

Following the application's final design reported by Feijóo and De la Rosa, 2016, *RoBlock* articulates a methodology encompassed between Behaviorism and Constructivism, by guiding the student through modules dedicated to basic concepts of programming and algorithmics, that are applied in tasks that must be solved by a robot in environments with free cells for displacement, marks to be discovered, obstacles to avoid, colors to identify, among other elements included. In total, the tool has five virtual modules and a remote module that allows the student to interact with real scenarios available in a remote laboratory (module that is still in a prototype state). The modules appear in an incremental way, requiring from the student to achieve each module before moving on to the following one.

To carry out the incorporation of the pedagogical strategies previously described through this article, each of the five virtual modules corresponding to *RoBlock*, offers a different interactive approach.

Considering that the pedagogical context that is intervened with this project is the Colombian one, the designed application is in Spanish, being it the native language of the Colombian population.

4.1 Virtual Modules

RoBlock offers a total of five virtual modules, exposing tasks and scenarios for the student's learning process towards programming and algorithmic concepts and skills. The modules included respond to the following big topics: variables, sensors, conditionals, loops, and functions.

Mobile robotics offers a pedagogical context apt to teaching every one of the topics of interest because of the successful convergence with the needs and operations of a mobile robot.

4.1.1 Module for Variables

For this proposed module, the student is asked to solve exercises in which the robot must find a series of marks that are distributed along a scene, within a determined time frame (Figure 1).

Figure 1: Module for Variables (Content in Spanish).

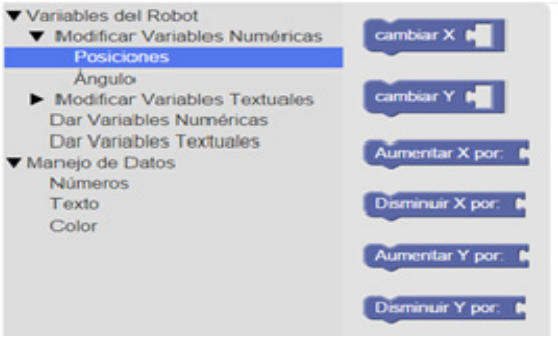

Figure 2: Module for Variables – Blocks' Menu (Content in Spanish).

For this purpose, the tool gives the student a series of Visual Blocks designed to manage the robot's position variables (translation and rotation), in addition to an editor that helps with the integration of these and the control of their respective execution (Figure 2).

In this module, the student does not require yet to reach all the marks in the scene in a single code execution. Each achieved mark is saved as "reached", letting the student to experience, use and play with several of the blocks arranged for this module.

4.1.2 Module for Sensors and Module for Conditionals

Throughout these modules, the student is asked to solve exercises in which marks are hidden from the scene, so the student requires to use sensors and conditionals to find them in the time determined by the problem (Figure 3).

Both modules differentiate in the type of visual blocks available to the student. In the case of the module for sensors, self-contained blocks for the sensors management are provided, by which the student does not need to declare or define any condition. On the other hand, for the module for conditionals, each sensor is summarized in a boolean block, asking the student to define the evaluation of conditions with additional conditions' blocks.

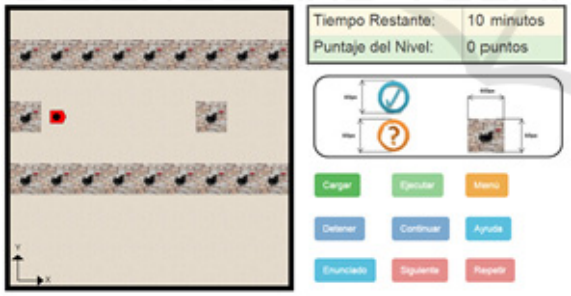

Figure 3: Module for Conditionals (Content in Spanish).

The learning process of the student is incremental, so that, having already passed the module of variables, the tool gives the student a series of blocks to handle position variables of the robot (Figure 4), in complement with a series of new blocks that allow the student to manage sensors (module for sensors) and conditions (module for conditionals).

As in the module for variables, in this module the student does not require yet to find all the marks in the scene with a single code execution. Each mark that found is visible in the scene and saved as "reached", making it easier for the student to experiment with the use of several of the blocks arranged for this level.

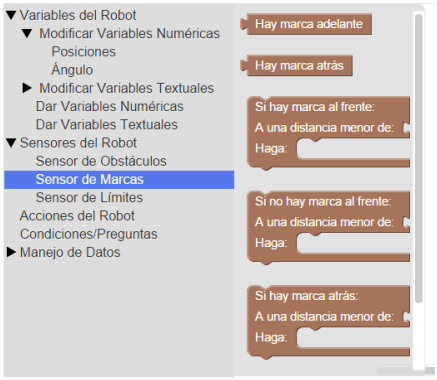

Figure 4: Module for Conditionals – Blocks' Menu (Content in Spanish).

4.1.3 Module for Loops

Through this module, the student works with exercises in which the robot must move through routes and proposed alleys (Figure 5).

Figure 5: Module for Loops (Content in Spanish).

As in the modules already described, the tool grants the student a series of visual blocks proposed for the management of position variables of the robot, in complementing with a series of elements that allow the student to handle sensors and conditions. Furthermore, and from this module, the tool provides blocks that let the student model and use control instructions for loops and paths' following (Figure 6).

Unlike the previous modules, in this module the student requires solving the task through the scene (or maze) in a single code execution. This is achieved because each time the position of the robot restarts, those marks already reached remain unmarked. The premise is that the exercise must be solved, emphasizing the need to declare and use loops.

Figure 6: Module for Loops – Blocks' Menu (Content in Spanish).

4.1.4 Module for Functions

Throughout this last virtual module, the student works with exercises in which all the types of scenarios provided in previous modules appear. The only difference is that, for this module, the student must use functions (blocks enabled from this level) that the tool validates from its declaration to its invocation.

This module provides blocks for the declaration and use of functions (Figure 7), in addition to those blocks already grouped and offered to the student in the previous modules (Figure 8).

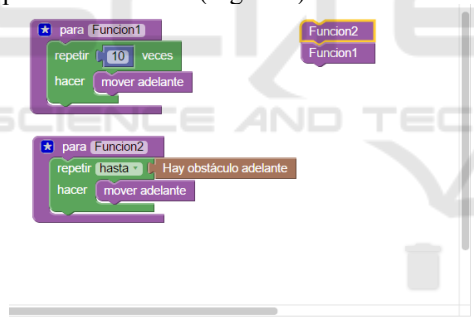

Figure 7: Module for Functions – Declaration and Blocks' Usage (Content in Spanish).

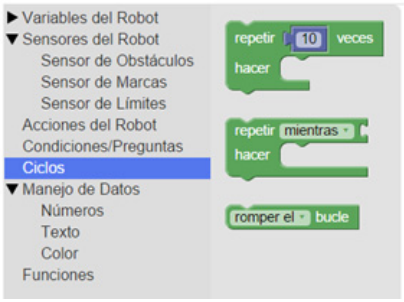

Figure 8: Module for Functions – Blocks' Menu.

In the same way, as in the previous module, in this module the student requires navigating the raised scene with a single code execution. This occurs because each time the robot's position restarts, the marks already reached remain unmarked. The latter, with the premise that the exercise is solved, emphasizing the need to declare and use functions.

5 PEDAGOGICAL APPROACH

The designed tool proposes to offer the student self-learning environments for programming and algorithmic concepts and skills. This is because, according to the directors of the participant educational institutions in this project, and considering the national educational scheme, there is not enough curricular time to cover programming and algorithmics topics, without sacrificing time for the strengthening of core areas required in Colombia: mathematics, natural sciences, history, and Spanish.

Pedagogically, *RoBlock* makes use of both Behaviorism and Constructivism; inviting the student to design and construct her/his own answers to tasks proposed by the tool, while raising levels of knowledge that lead the student to the appropriation of concepts of different complexities. These problems are based on the principle that their solution is unique. In this way, the students obtain stimuli against their answers, based on the personal development of its algorithmic solutions.

Considering that different students may not learn programming equally, it is fundamental that the first levels allow free will, but at the same time, focus on the functionality of the algorithm or solution towards the understanding and management of basic concepts. This implies that the student is implicitly guided to find the solution of the tasks proposed, by offering only, and in a strategic way, those elements needed to solve the proposed tasks.

6 ROBLOCK'S EVALUATION

To test *RoBlock* as a learning technology, 46 school students from three institutions (one public and two private) actively participated in a comparative pedagogical evaluation environment using Scratch as a reference technology, generating interesting and significant results in terms of the programming and algorithmics learning process. Among the target

population, 27 students evaluated *RoBlock*, providing feedback, given the functionalities offered by the software. The results here presented, which correspond to the qualitative assessment conducted within the study, complement the quantitative results previously reported by Feijóo and De la Rosa (2016).

6.1 UX: Students' Perception

The study group interacted with *RoBlock* in a period equivalent to five hours of experimentation. Throughout the established period, they solved exercises for each of the five virtual modules offered, working approximately one hour per module. At the end of their experience, the students made a qualitative evaluation, providing their opinions towards the tool and its purpose.

The results here presented show that the tool was well received by the target population and that, for five hours of interaction with it, a large minority (close to 50%) was partially or totally in agreement that it is possible to learn programming and algorithmics only with *RoBlock*. This is of interest, especially given the characteristics of the study group, previously indicated in this document.

The first question was raised to know if the group considered *RoBlock* an interesting tool. Most students (96.3%) who worked with *RoBlock* considered the tool interesting or very interesting (Figure 9). This is favorable for the study, considering that these students are the target population of the project, and that those who considered the tool not interesting, do not exceed 4% of the population surveyed.

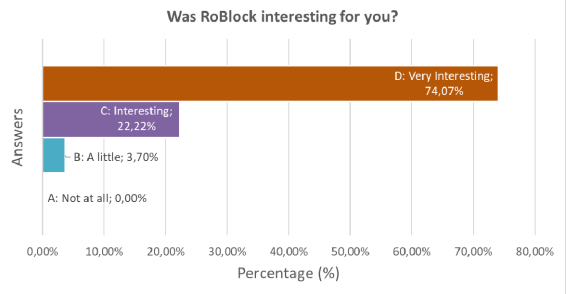

Figure 9: Answer: Do you consider *RoBlock* an interesting tool?

Complementarily, most students (96%) considered *RoBlock* as a good tool for teaching programming and algorithmics in a playful way (Figure 10).

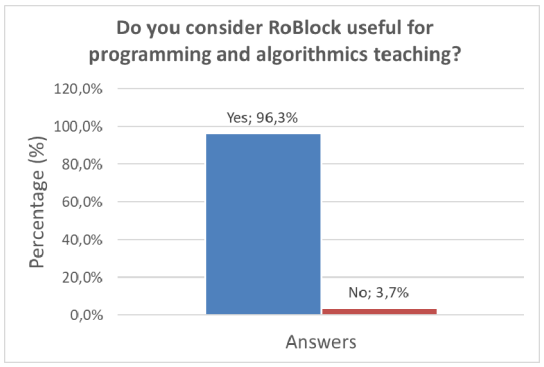

Figure 10: Answer: Do you consider *RoBlock* useful for Programming and Algorithmics teaching?

Finally, a higher proportion of students (52%) indicated that they did not consider *RoBlock* enough to learn to program in a self-taught way (Figure 11). Yet, 48% of the students who worked with the tool are partially or totally in agreement that *RoBlock* serves to learn autonomously, which is considered a favorable response for *RoBlock* in this study.

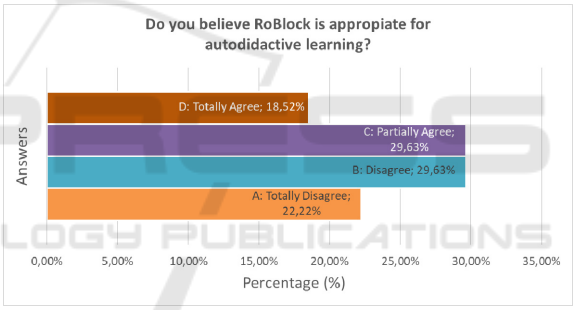

Figure 11: Answer: Do you believe *RoBlock* is appropriate for self-learning?

7 CONCLUSIONS AND FUTURE WORK

From this study, we conclude that the first version of *RoBlock* was a success, and that this is a friendly, interesting, and pleasant tool for the target population to which it is directed. This, based on the results obtained from the tests of User Experience and the acceptance presented by the students towards the tool.

In addition, we conclude that the research question of interest to this project is satisfactorily answered. With *RoBlock*, it is evident that it is possible to design a technological tool to motivate high school students to learn independently

programming and algorithmics. This, considering the results based on the second and third question (Figures 10 and 11).

From this pilot study, there is a broad horizon for future work involving *RoBlock*. In this study we evaluated only the virtual scheme that *RoBlock* offers through its first five modules, so it is worth evaluating the use of physical and remote scenarios with students, through the sixth module of the tool. A future study will include the latest module offered by *RoBlock*, and will evaluate the remote interaction of students with physical robots in preconfigured scenarios.

Also, based on the evolution of mobile technologies, *RoBlock* could evolve towards to operate with mobile devices, such as tablets and smartphones. Likewise, it would be interesting to carry out an impact study that indicates which technological approach is most striking to the target population, evaluating not only the appreciation for the tool, but also the level of learning obtained by the users.

ACKNOWLEDGEMENTS

We sincerely thank the educational institutions that helped us in this first version of the project: Colegio Provinma–Providencia Inmaculada (Bogotá, Colombia), Centro Educativo y Cultural Reyes Católicos (Bogotá, Colombia), and Programa Progresa Fenicia (Universidad de los Andes, Bogotá, Colombia). Their collaboration was essential for this pilot study to succeed, as they efficiently collaborated with their students and with all the necessary computational resources.

REFERENCES

Aggarwal, A., C. Gardner-McCune, and D. S. Touretzky. 2017. Evaluating the Effect of Using Physical Manipulatives to Foster Computational Thinking in Elementary School. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). doi: https://doi.org/10.1145/3017680.3017791

Feijóo, P. G., and F. De la Rosa. 2016. RoBlock – Web App for Programming Learning. *International Journal of Emerging Technologies in Learning (iJET),* Vol. *11, No.* 12, 45-53. doi: https://doi.org/10.3991/ijet.v11i12.6004

Frankie, T., D. Wesley, J. Gappy, and H. Cheng. 2017. C-STEM: Engaging Students in Computing with Robotics (Abstract Only). In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). doi: https://doi.org/10.1145/3017680.3017832

Galvis Panqueva, A. 1992. Ingeniería de software educativo. Bogotá: Uniandes.

Grover, S., and S. Basu. 2017. Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). doi: https://doi.org/10.1145/3017680.3017723

Gucwa, K.J, and H.H. Cheng. 2017. Making Robot Challenges with Virtual Robots. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). doi: https://doi.org/10.1145/3017680.3017700

Matarić, M. J. 2007. The Robotics Primer. Cambridge, MA: MIT Press.

MIT Media Lab, and Lifelong Kindergarten Group. 2006. Scratch. MIT. Retrieved from http://scratch.mit.edu

Paramasivam, V., J. Huang, S. Elliott, and M. Cakmak. 2017. Computer Science Outreach with End-User Robot-Programming Tools. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). doi: https://doi.org/10.1145/3017680.3017796

Weintrop, D., and N. Holbert. 2017. From Blocks to Text and Back: Programming Patterns in a Dual-Modality Environment. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). doi: https://doi.org/10.1145/3017680.3017707