# Children learning of programming: Learn-Play-Do approach

Julián-Andrés Galindo, and Monserrate Intriago-Pazmiño

***Abstract*—Writing computer programs is a skill that can be introduced to children and adolescents since early ages. Although children can gain skills in coding, there is a lack of motivation and easiness at the time to write logic structures. It raises the question, how can children be encouraged to code in a successful environment of learning and fun?. To address this question, this paper shows an experimental approach called "Learn-Play-Do" for introducing children in the programming. It shows that (1) it is feasible for children to learn about programming by following the proposed approach with (2) encouraging levels of learning, usefulness content and self-learning programming in (3) a developing country context. The results of an empirical experimentation with forty-one children are reported. This work was implemented as a social project linking the university with the community.**

***Index Terms*—Computers & programming, children learning, Scratch.**

## I. Introduction

Writing computer programs is a skill who can be introduced since early ages [1], [2]. Papert argued the main learning benefit is called the "Piagetian learning," or commonly called "learning without being taught" [3]. It has been reported as an effective device for a cognitive process instruction focus on teaching *how* rather than *what*. Through that, children can model abstract concepts to help them to develop skills such as classification, meta-cognition, left and right orientation, verbal memory and creative thinking.

Many issues have been reported about children learning of programming, documented by the UK's Computing Research Committee [4]. These issues involve a lack of motivation, easiness, and formal teachers training to guide young learners with enthusiasm and pro-activity. As a result, programming is seen as a boring, difficult and frustrating activity. Therefore, it seems like children and teachers need an approach more interactive to overlap engagement, fun, and learning.

Furthermore, some interactive environments has been released such as LEGO WeDo [5], Raptor [6], Scratch [7], Tinker [8] and Turtle Math [9]. These tools allow children to access to a complete set of features to build programs in online and offline settings. These features mainly include audio and video, events, logic sequences, conditions, loops and images control. However, technology itself is still not enough. First, children have been reported with cognitive issues to learn programming such as divergent thinking, awareness of comprehension failure, reflectivity and impulsivity, operational competence and receptive vocabulary [1]. Second, science learning emerges other children issues which include (1) children conception of the world and their influence at science learning, (2) language disabilities, (3) the role of the science teacher, (4) analysis of a teaching model and (5) the implications in the curriculum and teacher education [10]. Hence, technological solutions promote the art of programming. It requires a global and transversal approach.

This complex vision may be addressed by the implementation of an experimental project [11]. We introduce its core component Learn-Play-Do and We shall show the results of the first round with University students (fulfilling the instructor role) and children attending primary school. The key findings of using Learn-Play-Do reveals that (1) children learn by following its two stages (play and do) with (2) a valuable degree in learning, content usefulness and self-learning in programming with (3) children in an Ecuadorian context.

The rest of this article is organized as follow. The second section presents the related work about programming interactive environments for children. The third section describes the Learn-Play-Do approach. In the fourth section, we describe the design of our experimental study. In the fifth section, we report our experiment's results that compile the first experiences with this approach. The sixth section contains some discussions and limitations of this study. Finally, some conclusions and future works are presented.

## II. Related work

There are many approaches to teach children about programming. To begin, the book "Teach your kids to code" shows a traditional manner to learn where children are exposed to a console to write commands in Python and then check its output in a Graphical User Interface (GUI) (see Fig. 1). Although, this project restates the need of exploring coding in a fun environment with valuable principles such as "do it together", "Coding = Solving problems" and

Article history:
Received 15 September 2017
Accepted 28 November 2017

J.A. Galindo is a professor at the Departamento de Informática y Ciencias de la Computación, Escuela Politécnica Nacional, Quito, Ecuador. He is also a PhD candidate at the Laboratoire d'informatique de Grenoble, Grenoble Alps University (UGA), Grenoble, France (e-mail: julian.galindo@epn.edu.ec)

M. Intriago-Pazmiño is a professor at the Departamento de Informática y Ciencias de la Computación, Escuela Politécnica Nacional, Quito, Ecuador (e-mail: monserrate.intriago@epn.edu.ec)

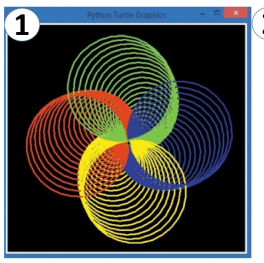


Fig. 1. Python project [2]. 1) program output, 2) source code.


Fig. 2. Turtle math project. a) turtle turner tool and b) the label lines tool.

”Explore!”, the method remains in text commands which is useless for children in level 0 (2-7 years) and level 1 (5-10 years) as reported in the project ”Should Your 8-year-old Learn Coding?” [2]. Thus, this traditional approach should be taught to children at level 3 (12 years and up) which rises a learning curve wall for others.

In the Turtle math project (see Fig. 2) building text and graphical relationships are exposed. It can be considered as a version of Logo for learning mainly mathematics [9]. This approach is based on six research principles which expose children (in upper elementary grades) and teachers in an interactive environment. It includes visual elements and text commands to control paths, shapes, scaling, coordinates, motions, drawing and number activities. The main principle related to programming is ”maintain close ties between representations”. This argues that explicit relationships between programming codes and drawings are essential. Children often lose these connections so that they need to write, save commands and see them run immediately. Although this approach underlines programming, fun seems like a fuzzy element in the interaction. The User Interface (UI) does not encourage children to play as coding because it has a lack of aesthetics and usability expressed mainly in the activity windows. For instance, children need to code by hand commands which may cause compiling errors (low usability) as well as a growth of negative user learning. A feature which can be faced by the introduction of interactive UI elements such as drag and drop widgets.

Another interesting approach is writing computer programs by using digital storytelling. It allows children to draw stories with a computer program where their imagination and composing skills are mixed with digital elements. In the 1990s, it became more popular including visual images and written text that expanded the student comprehension [12].

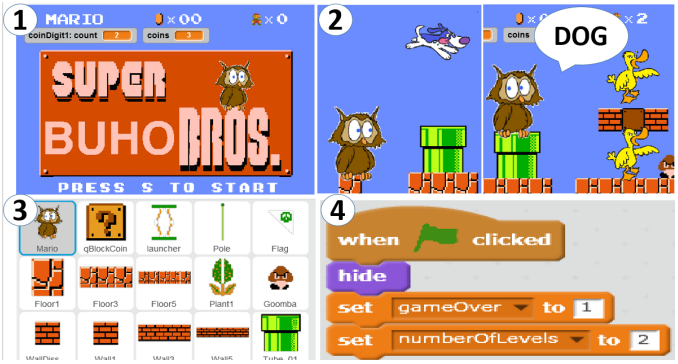


Fig. 3. Scratch project interface. 1) starting UI, 2) interactive UI, 3) visual elements and 4) script UI.

Then, Mitch Resnick in his TED conference emphasized the production of digital content by the expression “Learn to code and code to learn” where children develop writing and community learning skills by coding [13], [14].

Consequently, mixing writing skills and technology to combine audio, text, and video emerges a growing strategy to engage youth into computer programming. It was shown by Kelleher at designing a Storytelling program called Alice [15]. She argues that many girls begin to turn away from math and science-related disciplines (computer science) at the middle school. This programming environment provides to middle school girls a positive initial experience with computer programming as a means to the end of storytelling rather than an end in itself. A motivating activity for middle school girls at Pittsburgh.

Following this last approach, another robust research project was found named Scratch [7]. Scratch as a method to collect and test kids’ imagination allows them to create stories by a drag-and-drop block process. Kids stick the blocks together, forming code scripts as similar as developers create code lines in a web language such as python, C#, Java, and others. Then, when the code script is finished, kids can run it to bring to life the scratch characters of the screen. Using this tool, kids can create robust digital stories because every scratch block can represent text, audio or a video element to create an interactive story. For instance, Fig. 3 shows a Scratch project called “Super Buho Bros”. It is part of the ”Red Juega y Aprende” social project [16]. “Super Buho Bros” project aims to learn English vocabulary of animals as playing in a super Mario and fun environment. When a kid run the project, he will interact with the animations, read the English words and listen to the audios. Thus, Scratch projects represent a child ability to coordinate a different set of blocks to create projects as complex as they want to.

In spite of this, digital storytelling demands further examination in (1) the efficiency and clarity of the scripts produced by children, (2) the potential relation of programming and content, (3) analysis of imaginative and aesthetics features and (4) more broad studies to validate its

impact in children learning [17].

Overall, all approaches clarify the challenge to balance UI interaction, learnability, and playfulness. First, the traditional method (text-based) may be found difficult for children at early ages. Second, although there are advances in GUI, there is still a lack of aesthetics, usability in conjunction with enjoyability. Third, the storytelling approach by using visual elements helps children to learn about coding by mixing elements such as narration, creativity, and communication. Despite this, further examinations are mainly needed to validate the relation between digital content production and coding in children learning by these highlights.

## III. Learn-Play-Do approach

From the related work, it is highlighted that children may learn to code by harmonizing a rich UI in a fun way. Since this lesson learned, our experimental approach relies on two simple stages: Play and Do. The first stage exposes children to play interactive games (made for tutors) which aims to introduce children to the environment by playing instead of coding. Then, with this gained knowledge, children have the opportunity to create their own programs with a tutor assistance at the Do stage which attempts to promote a cognitive [18], social [19] and emotional [20] child development. It is expected that as children gain knowledge - in a Play and Do spiral - the tutoring will be less required.

Now, to ensure the children learning process during these stages, the approach is also underlined by the Suzuki methodology [21] [22] [23]. This learning method can be summarized by: *results* = *desire* + *repeat* [24]. Suzuki argues that one learns only by continual practice of basic or main concepts. For instance, when children are taught mathematics in an exiting(fun) and interesting manner they develop a desire to repeat the learned activities [24]. Consequently, in our context, *results(Learn)* = *desire(Play)* + *repeat(Do)*. *Desire* will be attached to our Play stage which should encourage children to do or perform activities again (*repeat*). Therefore, desire (Play) + repeat (Do) should evoke results to keep children in a continuous learning growth.

To be consistent, the approach should cover also the following factors or principles of Suzuki's methodology:

- Listening
- Memory
- Motivation
- Vocabulary
- Repetition
- Parental Involvement
- Step by Step Mastery
- Love

*Listening*: By listening and watching to video recordings of the interactive lessons given by tutors, children should learn the coding language and UI interaction just as they absorb the sounds of their mother tongue to interact with others.

*Memory*: Through repetition and listening to the recordings, the child will code from memory which is a skill they can use with other educational aspects (reading and maths). Coding by text commands is postponed until the child is able to code by drag and drop elements, just as we only teach children to read after they can speak fluently. In this way, the tutors can concentrate of the child's coding development of main factors such as start and stop programs (events and sensing), adding pictures, videos, widgets (look and sound), repetitive and conditional actions (control) and widget actions (motion and operators).

*Motivation*: Daily practice of games is encouraged to build the child's abilities and confidence. As the child masters a particular game (program) the motivation and sense of achievement will move them on to the next game in a desire to learn more. All students follow the same games sequence so that an standard repertoire provides strong motivation as younger children want to code games they hear older students code. Parent involvement will motivate children and give them a sense of achievement and makes playing an enjoyable experience.

*Vocabulary*: At the beginning, children should regularly repeat all previously learned games and code exercises to expand his instructions knowledge(vocabulary) and o reuse it in future programs. Just as in learning to speak, the entire vocabulary and grammar are used, not just the most recently learned words. In this way, the child gradually expands his cognitive abilities to solve problems by reusing code.

*Repetition*: Through constant repetition of games, children strengthen old skills and gain new ones. The technique is developed through the study and repetition of these games. Students can interact with their games to see the progress they have made.

*Parental Involvement*: It is required a three-way partnership between the child, the tutor, and the parent by working together. Parents need to go to tutor lessons to serve as home teachers. With this, a more enjoyable environment is made where children can consolidate the teaching given by the tutor.

*Step by Step Mastery*: Every child learn by building small coding steps so that they need to start with easy games or programs to master coding gradually.

*Love*: Tutors and parents should have a strong level of empathy, patient, tolerance, and creativity to guide and reinforce children learning.

## IV. Experimental study

The experimental study aims at exploring how feasible is children learning by using Learn-Play-Do approach. We will show that children learn programming following two stages (Play and Do) with an interactive tool "Scratch". The following section will underline the experimental protocol.

### A. Goal and hypothesis

The objective of this experiment is to examine the degree of learning and likeness during the exposition of children with Learn-Play-Do as a first attempt to understand how to teach children about programming. So the experiment's hypothesis are:

- H1: children learn by their exposition with Learn-Play-Do approach.
- H2: children like the games' content.
- H3: children are able to code by themselves basic programs.

### B. Experimental method

A quantitative research method was used [25], where children could interact with all games by following their preferences in a face-to-face tutor support. First, children interacted with the games only by playing. In Play stage, every two-children had a tutor who introduced them how to play by almost 20 minutes. Second, one tutor taught all the group how to create a single game by interacting with Scratch in a 20 minutes session. During Do stage, children are supported by their tutor to understand clearly the basic coding instructions by following the drag-and-drop interface from Scratch. Then, every child was challenged to make their own game by reusing the code of the previous stage. In this last part of the Do-stage, children had the opportunity to use their creativity and learned skills to develop a basic but fun game in a 20-minutes-period. Finally, a survey was provided by tutors to children to gather all their impressions about the programming experience.

### C. Participants

For all sessions, 41 children participated in the experiment from the Dominicas de la Inmaculada Concepción primary school with 20 tutors (University students). This children group is distributed in 19 (under 6 years old), 1 ( 7 to 9), 18(10 to 11) and 3 from 12 to 14 years old. Furthermore, and 2 professors at Computer Science Faculty who guided whole experiment sessions. They supported and reinforced the learning process.

### D. Materials

Five games were made for University students by using Scratch to expose children to interact with. Those ones include the body, the instruments, the numbers, the animals and the transport game that we will explain below.

*1) Body Parts:* The game has two options for body parts (see Fig. 4). It has its own audio and image. Once both are present, a question concerning launches into a body part which must be correctly selected. They must complete a total of five hits to win, otherwise, they lose. It reinforces the body parts in English.


Fig. 4. Body Part Game


Fig. 5. Instruments Game

*2) Instruments:* The game features two choices of musical instruments (see Fig. 5), which have their own audio and image, once both presented a question regarding a musical instrument which shall be selected appropriately launches. They must complete a total of five hits to win, otherwise, they lose. It aims to help children to recognize musical instruments in English.

*3) Numbers:* It allows children to do a review of the numbers 1 to 9 in English while showing us the correspondent quantity of different elements per number (see Fig. 6). For instance, at showing the number "3", three soccer balls are


Fig. 6. Numbers Review


Fig. 7. Animals Game


Fig. 8. Transport Game

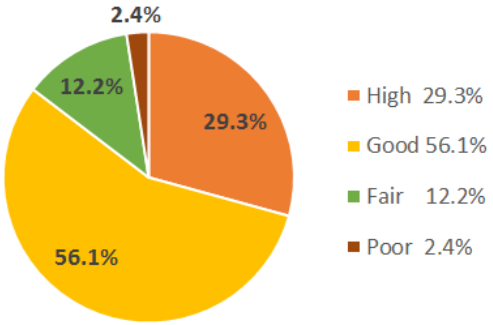


Fig. 9. About learning degree. Answers to the question: What grade of learning did you achieve through the game used?

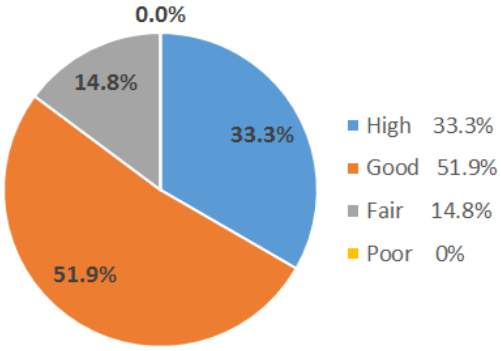


Fig. 10. About content usefulness. Answers to the question: How useful is the exposed content?

shown and the English word 'three' is playing. Hence, children can learn about the quantity and also the relation with sound and visual interaction.

*4) The animals game:* The game is designed to learn to differentiate vertebrates invertebrates, supporting English (see Fig. 7). We also have three ways to play. Vertebrate or invertebrate, here children need to locate the correct animal in the respective box classification, vertebrate or invertebrate. For the Roulette questions, it is necessary to spin the wheel with the space-bar, then we generated a question which has three options A, B and C, where one of these is the correct answer.To catch animals, an augmented reality game is shown for which it is necessary to use a webcam.

*5) Transport:* The game features two transportation options, each has their own audio and image (see Fig. 8). The main goal is a reinforcement of the means of transport in English. During the game, once both are present audio and image a question appears concerning a means of transport which shall be selected appropriately. Children must complete a total of five hits to win, otherwise, they lose.

*6) Survey:* A paper-based questionnaire was used to gather children impressions. It involved six questions regarding with the age, gender, learning degree, content usefulness, training likeness and coding independence.

## V. Results

Two resulting products are reported. Fully developed games for children and their responses to the Learn-Play-Do approach. First, children games were published in [26]. Almost all games had a basic interaction as expected; however, a few children were able to modify the game "Super Mario Bros" by including new characters. Second, the following section will explore the key points regarding the gathered responses.

*1) Learning degree:* The results confirm that children learned coding with almost 56% and 29% for an acceptable and sufficient level (see Fig. 9). Reaching approximately 15% of low gained knowledge.

*2) Content usefulness:* Children liked the content of games (materials section) revealing almost 85% of a positive content reception and 15% as a negative one (see Fig. 10).

*3) Coding independence:* Mostly all children remained with confidence at programming with approximately 78% (see Fig. 11). In consequence, an group of 8 children did not learn enough to develop their own program.

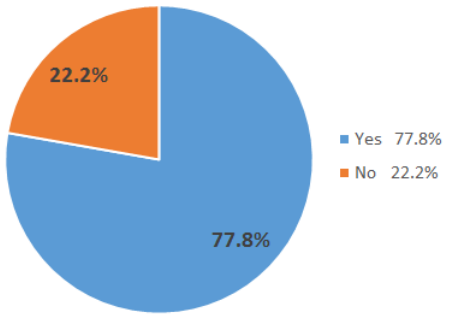


Fig. 11. About writing new code. Answers to the question: After this training, could you create your own code?

## VI. Discussion

Programming can be introduced to children through many interactive platforms since plaint-text tools until rich user interfaces. However, this activity becomes harder at facing discouragement and good support at coding logic structures. Hence, it is valuable enough to explore an approach to encourage children at coding as keeping a learning and fun environment.

In this study, we have identified three main findings. First, most children reflected a positive learning degree with almost 85% after their exposition with "Learn-Play-Do" (H1). This fact is related to previous studies where children's performance reached a mean score of 64% programming by using Scratch [2]. Here, there was also a valuable factor to encourage learning beyond technology identified as affect. In fact, it was seen not only as an accompaniment but also as a source of motivation. As stated by Duncan "Learning will hardly progress without motivation, and that is stirred and maintained by positive affect". It supports the Learn-Play-Do approach where learning should be driven by a combination of fun (play) and repetitive activities (do). Thus, it suggests that learning may gain important levels in children when affect elements are shown such as motivation and fun aligned with technology.

Second, a low level of 15 percent was revealed in content likeness by children at interacting with the pack of 11 developed Scratch games (H2). Although it could be seen also a matter of novelty in children, there is positive evidence of longer children exposition with contents developed in Scratch shown in [2]. These contents included interaction with geometric shapes, sprites, scratch cards and audio files trough tasks such as order, selection, sequence, movement, coordination, and synchronization. Furthermore, it was argued that Scratch was beneficial and fun in an 8-weeks-period for children overpassing mere novelty. Therefore, a cyclical and longer exposition may be needed to confirm or validate the positive attraction of the content games.

Third, the majority of children showed a high level of coding independence with 78% (H3). Similarly, Burke and Kafai found that 9 out of 10 children knew more programming after their exposition with Scratch into a Storytelling process [17]. These technical skills included programming concepts such as object-oriented and sequential, sprite to sprite conditionals, looping, boolean variables, sampling scripts and if-then statements. Hence, children are able to increase their development skills trough Scratch; however, our study does not measure the performance of individual logic structures(e.g. if-then or do-while). Hence, it suggests a more deep analysis in the manner of children code which may confirm the gained levels in coding by logic structure toward a bottom-up approach.

*1) Experiment limitations:* Although these initial experimental results are encouraging, some restrictions should be noted. First, children should have more exposition to the approach which can clarify the impact of learning. Second, Learn-Play-Do approach used Scratch as an interactive tool for programming. However, there are other tools which could release other insights such as Lego Boost and Lego Mindstorms. Lastly, even when the initial games were quite easy to understand and explore their programming; it should be advisable to increase games difficulty according to children age and also the definition of a formal method of games assessment.

Overall, this experimental results reveals mainly that it is possible to use the Learn-Play-Do approach to achieve initial levels of (1) learning, (2) content usefulness, and (3) coding independence which need a deeper and longer exposition to validate and/or extend its degree of learning by children.

## VII. Conclusions and Perspectives

This paper presents an approach denominated Learn-Play-Do for learning about code programming for children and an experiment to show its initial results. It validates that it is feasible for children to introduce them in learning programming skills by following the stages Play (fun) and Do (repetitive) with the interactive tool "Scratch" in the first round of the social project. Children revealed a valuable level of learning, content usefulness and coding independence. More experiments will be necessary to prove the knowledge reception in order to validate and/or extend the approach(stages and principles) with more children groups and experiment conditions. From the created games for children, more analysis in the script(code) could reveal the level of learning gained per child with more deep detail in logic sequences, variables and UI actions definition.

## Acknowledgment

The authors wish to thank God, reviewers and our priceless family as well as University students for their support, highlights and quality time.

## References

[1] D. H. Clements and D. F. Gullo, "Effects of computer programming on young children's cognition," *Journal of educational psychology*, vol. 76, no. 6, p. 1051, 1984.

[2] C. Duncan, T. Bell, and S. Tanimoto, "Should Your 8-year-old Learn Coding?" in *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, ser. WiPSCE'14. New York, NY, USA: ACM, 2014, pp. 60–69.
[3] J. Lochhead and J. Clement, *Cognitive Process Instruction. Research on Teaching Thinking Skills.* ERIC, 1979. [Online]. Available: https://eric.ed.gov/?id=ED234997
[4] UKCRC, "UK Computing Research Committee," 2010. [Online]. Available: http://www.ukcrc.org.uk/
[5] K. Mayerov?, "Pilot activities: LEGO WeDo at primary school," in *Proceedings of 3rd International Workshop Teaching Robotics, Teaching with Robotics: Integrating Robotics in School Curriculum*, 2012, pp. 32–39.
[6] M. C. Carlisle, T. A. Wilson, J. W. Humphries, and S. M. Hadfield, "RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving," in *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '05. New York, NY, USA: ACM, 2005, pp. 176–180.
[7] M. Resnick, J. Maloney, A. Monroy-Hern?ndez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and others, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.
[8] H. Lieberman, "Tinker: A programming by demonstration system for beginning programmers," *Watch what I do: programming by demonstration*, vol. 1, pp. 49–64, 1993.
[9] D. H. Clements and J. S. Meredith, "Turtle math," *Montreal: Logo Computer Systems (LCSI)*, 1994.
[10] R. Osborne and P. Freyberg, *The Implications of Children's Science*. Heinemann Educational Books, Inc, Jan. 1985.
[11] G. Julian and I. Monserrate, "Proyecto EPN-FIS de Vinculación Social, Programación para niños Red Juega y Aprende." 2015.
[12] G. Kress, "Visual and verbal modes of representation in electronically mediated," *Page to screen: Taking literacy into the electronic era*, p. 53, 1998.
[13] M. Resnick, "Learn to Code, Code to Learn," 2013. [Online]. Available: https://scratch.mit.edu/
[14] ——, "Scratch day," *EdSurge, May*, 2013.
[15] C. Kelleher, J. Hodgins, and S. Kiesler, "Motivating Programming: using storytelling to make computer programming attractive to more middle school girls," 2006.
[16] Red juega y aprende, "Interactive game, Super Buho Bros at the Scratch MIT network," 2015. [Online]. Available: https://scratch.mit.edu/projects/45950952/#editor
[17] Q. Burke and Y. B. Kafai, "Programming & storytelling: opportunities for learning about coding & composition," in *Proceedings of the 9th international conference on interaction design and children*. ACM, 2010, pp. 348–351. [Online]. Available: http://dl.acm.org/citation.cfm?id=1810611
[18] J. Piaget, "Part I: Cognitive development in children: Piaget development and learning," *Journal of research in science teaching*, vol. 2, no. 3, pp. 176–186, 1964.
[19] A. Kozulin, *Vygotsky's educational theory in cultural context*. Cambridge University Press, 2003.
[20] N. Chomsky, "A review of BF Skinner's Verbal Behavior," *Language*, vol. 35, no. 1, pp. 26–58, 1959.
[21] S. Suzuki and W. Suzuki, *Nurtured by love: The classic approach to talent education*. Alfred Music, 1983.
[22] E. Hermann, *Shinichi Suzuki: The Man and His Philosophy (Revised)*. Alfred Music, 1999.
[23] S. Suzuki and M. L. Nagata, *Ability development from age zero*. Alfred Music, 2014.
[24] D. G. Hazlewood, S. Stouffer, and M. Warshauer, "Suzuki meets P?lya: teaching mathematics to young pupils," *The Arithmetic Teacher*, vol. 37, no. 3, p. 8, 1989.
[25] N. Mandran, "Méthode de conduite de la recherche en informatique centrée humain : processus et inclusion d'une démarche centrée utilisateur," Ph.D. dissertation, Nov. 2015, working paper or preprint.
[26] Red juega y aprende, "Scratch - Imagine, Program, Share," 2015. [Online]. Available: https://scratch.mit.edu/studios/932042/

**Julian-A. Galindo** was born in Quito, Ecuador, in 1982. He received a bachelor degree in informatics engineering from Central University, UCE, Quito, Ecuador, in 2007. The Master in Information Technology from James Cook University, JCU, Townsville, Australia (2012). In 2014, he joined the Faculty of Engineering in Systems, National Polytechnic School, EPN, as a Professor. Since October 2016, he has been working with the "Laboratoire d'informatique de Grenoble", LIG, Grenoble Alps University, UGA, Grenoble, France as a PHD student in the domain of Human Computer Interfaces. His research interests include children's learning, user interfaces, programming, adaptation, user modeling and music analysis.

**Monserrate Intriago-Pazmiño** was born in Chone, Ecuador, in 1984. She received the B.S. degree in Computer Science Engineering from National Polytechnic School, Quito, Ecuador, in 2007 and the M.S degree in Computer Science from the Technical University of Madrid, Spain, in 2011. She is currently a Ph.D. candidate in Computer Science at Technical University of Madrid. From 2008 to 2009, she was Assistant Professor with the Department of Informatics and Computer Science, National Polytechnic School. Since 2011, she has been a member of the Biomedical Informatics Group, Technical University of Madrid. Since 2014, she has been Professor at the Department of Informatics and Computer Science, National Polytechnic School. Her research interests include programming, software development, biomedical informatics, machine learning.