Received: 19 October 2023 | Revised: 30 January 2024 | Accepted: 20 March 2024

DOI: 10.1002/cae.22740

RESEARCH ARTICLE

WILEY

Fostering computational thinking with simulated 3D robots in secondary education

Luis Castro-San Martín[1] | **Raquel Hijón-Neira**[1] | **Celeste Pizarro**[2] | **José M. Cañas**[3]

[1]Dpto. Informática y Estadística, Universidad Rey Juan Carlos, Móstoles, Spain

[2]Dpto. Matemática Aplicada, Ciencia e Ingeniería de los Materiales, Universidad Rey Juan Carlos, Móstoles, Spain

[3]Dpto. Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación, Universidad Rey Juan Carlos, Fuenlabrada, Spain

Correspondence
Celeste Pizarro, Dpto. de Matemática Aplicada, Ciencia e Ingeniería de Materiales y Tecnología Electrónica, Universidad Rey Juan Carlos, Office 018-Dept-II, C/Tulipán, s/n 28933 Móstoles, Spain.
Email: celeste.pizarro@urjc.es

Funding information
ERDF, EU, Grant/Award Numbers: PID2022-137849OB-I00, MICIU/AEI/10.13039/501100011033

Abstract
Computational Thinking (CT) can be defined as the thought processes involved in formulating problems so that their solutions can be represented as sequential steps and algorithms. It is a key skill for children in the 21st century. However, it is unclear how CT can be developed most effectively in children. Several pedagogical methodologies have been proposed and are being investigated. The main aim of this paper is to test the hypothesis which states that using three-dimensional (3D) simulated robots helps in the learning of programming and CT concepts, such as directions, loops, conditionals, and functions. The research questions are: Does this hypothesis hold true? Are some concepts easier or better learned than others and to what extent? The goal is to measure and evaluate the effect of using as a learning tool a platform with 3D simulated robots and realistic physics, and compare it with the standard Scratch learning tool which does not use robotics but a two-dimensional (2D) cartoon avatar they are already familiar with. For practical reasons, a quasiexperimental design with nonequivalent groups and 85 second-year Secondary Education students (ages 12–13) was performed. They were separated into control and experimental groups and followed a seven-session intervention with the baseline 2D Scratch and the 3D simulated robots platform, respectively. Both used a visual block programming language and the same activities. To have quantitative and reliable results, a widely accepted CT test has been used, pre- and postintervention. Also qualitative feedback is presented. The obtained results show that using the platform with simulated 3D robots significantly helps when developing students' CT. With it, the students do learn basic programming concepts and reach higher scores in the CT test. This improvement applies to all CT-analyzed concepts except in functions where the grades are maintained. Furthermore, students manage to master the activities on the 3D simulated robots platform, which reflects on the empowerment the platform has got in them.

KEYWORDS
3D simulated robots, computational thinking, e-learning platforms, programming, robotics, Scratch, secondary education

1 | INTRODUCTION

Computational Thinking (CT) is a human ability that has been increasing in relevance over the last 15 years for educators, researchers, and politicians in the field of education [10, 14]. There are several definitions for it [55]. CT can be defined as the ability to solve problems, design systems, and understand computational concepts based on human behavior [63]. It can also be defined as the thought processes involved in formulating problems so that their solutions can be represented as computational steps and algorithms [2]. CT is a key skill for children in the 21st century [62, 64]. It is related to the capability of understanding (and creating) technology, not just using it.

Given the relevance of CT learning and its increasing presence in the secondary education curriculum, education professionals need effective tools for teaching this skill in the classroom. The goal of the study is to evaluate the influence of the use of three-dimensional (3D) simulated robots on the learning of programming concepts and CT, such as directions, loops, conditionals, and functions.

The study uses the experimental research method with a quantitative approach where the dependent variable is the students' learning. The experimental study (quasiexperimental pretest–posttest design with the control group) is conducted in a secondary school with students in the second-year secondary education in the subject "Technology, Programming, and Robotics," using a nonprobabilistic casual sampling.

This research paper asks the following question: Can CT be improved by using simulated 3D robots to teach computer programming and CT, such as directions, loops, conditionals, and functions to Secondary Students? Our hypothesis (H) proposes that the answer is yes. For this study, we asked 85 second-year Secondary Education students (ages 12–13) to follow a seven-session intervention. There were two research questions: (RQ1) Is it possible to foster students CT (directions, loops, conditional, and functions) by practicing concepts of computer science programming with simulated 3D robots? and (RQ2) Are there concepts easier or better learned than others and to what extent? The results derived from this study show that using 3D simulated robots, which include such robots, can significantly develop students' CT, but also that students are able to learn basic programming concepts, and that there is improvement in all concepts, except in functions, where the grade is maintained.

The paper is organized as follows: Section 2 presents related work; Section 3 describes the experimental design, materials, and methodology; Section 4 presents the results; Section 5 presents the discussion and threats to validity; and Section 6 ends the paper with the main conclusions and lines of future work.

2 | RELATED WORK

Despite the general consensus in the education community on the importance of CT skills, it is still unclear how CT can be developed most effectively in children. Different pedagogical methodologies that can be used to develop it are being investigated.

In recent years, some authors have claimed that CT can be acquired and developed by teaching programming, and coding, to children. Furthermore, it has been claimed that this should be done as early as possible [29, 34, 39, 46, 58]. Learning to program can induce changes in the way people think [1, 47]. This is probably due to the analytical component of CT, which is quite similar to mathematical thinking (i.e., problem-solving), engineering thinking (process design and evaluation), and scientific thinking (systematic analysis). A common worldwide tool for teaching children programming is Scratch [40], developed by the Lifelong Kindergarten research group at the MIT Media Lab. It allows one to design interactive programs by linking programming instructions with visual blocks [33, 45, 52].

CT can also be learned through other approaches, such as generating computer games [24], storytelling [37], disconnected computing activities [12, 41], ScratchJr [46], or even in ethics lessons [48].

In addition, CT may be taught and learned using robots. In fact, educational robotics (ER) [4, 5, 8, 31] is widely extended as a way to teach STEM skills, including CT, and a nice and efficient way to introduce children to technology basic concepts. Not only in robotic competitions, such as RoboCup Junior, First LEGO League, and so forth, but also in regular courses. A comprehensive literature review on the combination of Robotics and CT can be found in [66, 6].

Physical robots are naturally compelling and awaken the students' curiosity and motivation. And so, robotics is a common way to introduce CT concepts to children. Many successful robots are used in the ER, such as WeDo, Mindstorms EV3 [57], or Spike from LEGO; Mbot from Makeblock; Thymio and Edison programmable robots.

The combination of ER and CT has been widely explored with students of different ages, from elementary school [42] to primary education [18] and secondary education. García-Valcárcel-Muñoz-Repiso and Caballero-González [26] conclude in their study with early childhood education students, aged 3–6 years, that it is possible to develop CT skills at such early ages and that the use of robotics activities has a significant positive impact on learning.

Beyond physical robots, virtual robots have also been explored [43] in the last few years. Simulated robots keep the potential for learning while providing some practical advantages such as cheaper cost, no need for maintenance, and anywhere-anytime availability. An interesting comparison between physical and virtual robots can be found in [7].

After their two studies with US elementary and high school students, Witherspoon et al. [65] stated that participating in a directed programming curriculum, in which virtual robots are programmed, develops CT knowledge and skills.

There are several tools for teaching robotics with simulated robots: OpenRoberta [25, 32], GearsBot [27], and CoderZ [21] to mention a few.

It is interesting to have a way to assess CT in students because it is a valuable and productive skill. [3]. This way, several teaching approaches may be explored in a quantitative and scientific way, and the performance of several courses may be somehow assessed.

Assessing learning processes is a tricky topic, but there are several resources. First, the Bebras international informatics contest [23] with its scoring system. Second, several tests have been published and validated in the research community, suitable for different student ages. For elementary students [19] is a nice study. For middle school students [61] is an interesting work. For secondary students, the CT test [53] is widely accepted. Recently, a new test specific for adults is also available [36].

3 | METHODS

3.1 | Participants and context

The experience was carried out in the 2021/2022 academic year with 85 students from the second-year Secondary Education (12–13-years olds), who took the subject "Technology, Programming, and Robotics" at the IES Velázquez (Móstoles, Spain) high school. This subject includes teaching units on programming, CT and robotics in its contents. Traditionally, instructors at the school had used the Scratch platform to teach basic concepts first, and after that they used real robots in subsequent courses. When they had the opportunity to use a virtual robot programming platform, they decided to teach programming, CT, and robotics together using the educational Kibotics web platform.

A nonprobabilistic random sampling was carried out with a sample of 85 students. The sample is divided into two groups of students: a control group of 37 students and an experimental group of 51 students, see Table 1. The groups are formed by randomly assigned groups of complete classes: two classes to the control group and two to the experimental group. The control group uses the Scratch web platform to complete the programming activities, and the experimental group uses the ER web platform Kibotics [35].

3.2 | Experimental design

Both Scratch and Kibotics are online platforms, all students have user access to the assigned platform of the group (Control or Test), and they must complete the activities individually. Each student has his/her own computer during test sessions (Test group) and during programming lab sessions (Control group).

No type of selection is made among the students for participation in the study, this being totally voluntary and offering the possibility of participation to all students in the second-year high school.

Seven 50-min sessions are held in which students learn robotics and programming concepts on the Kibotics and Scratch platforms (see Figure 1).

The first two sessions are preparatory for the programming activities (workshops). In the first, students are introduced to ER, and in the second, it is explained how the robots are programmed and the platforms that they will use during the programming workshops are presented

TABLE 1 Gender of participants divided into groups.

	Experimental (n = 51)		Control (n = 37)	
Gender	# Participants	%	# Participants	%
Male	25	49.01	23	62.16
Female	21	41.17	12	32.43
Prefer not to say	2	3.92	2	5.40

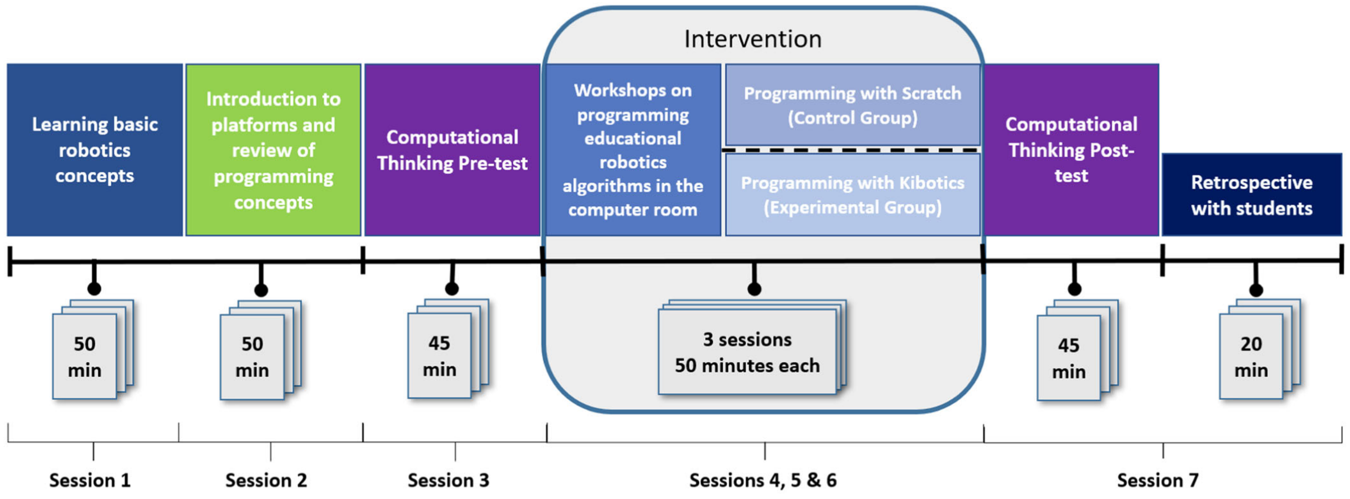

Intervention
Learning basic robotics concepts
Introduction to platforms and review of programming concepts
Computational Thinking Pre-test
Workshops on programming educational robotics algorithms in the computer room
Programming with Scratch (Control Group)
Programming with Kibotics (Experimental Group)
Computational Thinking Post-test
Retrospective with students
50 min
50 min
45 min
3 sessions 50 minutes each
45 min
20 min
Session 1
Session 2
Session 3
Sessions 4, 5 & 6
Session 7

FIGURE 1 Outline of the educational intervention.

In the third session, the first CT test (Pretest) is performed. In the fourth, fifth, and sixth sessions, the programming activities are carried out, each group with the corresponding platform according to their belonging to the Control (Scratch) or Test (Kibotics) group.

In the seventh session, the last one, the CT test is repeated (posttest). After the completion of the test, students are invited to transfer their comments (retrospective) on the different platforms.

From a methodological point of view, the sessions consist of the presentation of theoretical and practical concepts by the teacher, while receiving feedback on the previous knowledge of the students. Once the necessary information has been presented, the students take the initiative to solve the practical programming activities proposed on the Kibotics and Scratch platforms, with support at all times upon request of the students. In addition, during the development of the sessions, attention is paid to the relevant information contained in the material delivered to the students if excessive difficulty is perceived in the execution of the activities.

The educational intervention has the favorable assessment of the Ethics Committee of the Rey Juan Carlos University with the internal registration number 1901202203322.

3.3 | Materials

For the programming workshops, the experimental group used Kibotics platform [35] and the control group used Scratch platform, which is well known and used by computer science teachers to teach programming, but does not support ER directly. In the following subsections, the Scratch and Kibotics platforms are briefly presented. The experimental materials used and the tasks carried out by the students in each of them are also described below.

3.3.1 | Scratch web platform

Scratch is a simple web Integrated Development Environment (webIDE) that was developed and published in 2003 by MIT Media Lab and the Playful Invention Company. Its purpose is to help young students and kids to learn computer programming and CT. Its first release was available only as a desktop application. From 2013, with the version 2.0, it is also available as a web service. In 2018 version 3.0 was released, both as a desktop application and as a web platform. This version integrates interactive elements that help the user to learn the platform usage and the language itself. In addition, there is ScratchJr for tablets, which has been specifically designed for kids ages between 5 and 7 years [60].

Scratch is also a visual programming language, block-based, used on that platform. It allows the development of CT [67] and the introduction of users to programming abilities without any prior knowledge from Scratch. Its concepts may also be used in high-level text-based languages, such as Java, Python, or C#. There are blocks for loops, for conditional instructions, for arithmetic operations, variables, and so forth. The program is created by assembling blocks, and its execution flow runs in a sequential fashion. In 2020, Scratch was the first programming language specific for children entering into the top-20 TIOBE ranking [59].

The user community of Scratch platform has grown continuously worldwide since its creation. Users may contribute by publishing their projects and allowing other users to reuse them as a base for their own new developments.

3.3.2 | Kibotics web platform

It is a web platform for learning CT and ER. It is mainly designed for primary and secondary school students. It supports robot programming in Scratch language (Blockly variant) and in Python language. It includes an online 3D robot simulator and it also supports common physical robots, such as LEGO EV3, Makeblock Mbot, or Tello Drone. Kibotics provides several courses which include several robot programming challenges to the students, and presents CT concepts in the context of robots that have to perform a task (following a line, avoiding obstacles, cleaning a room, etc.), with their sensors and actuators. It follows the pedagogical learn-by-doing approach.

The Kibotics platform shows a block-based code editor and a 3D robot scenario, as shown in Figure 2. The robot runs the Scratch program. In the editor, the child may build her program by dragging and dropping blocks from the palette. The 3D scene can be seen from a fixed observation position, from the bird-eye view, or from the onboard robot camera. Kibotics provides several debugging tools such as the robot teleoperator, the sensor viewer, the map, and a console.

3.3.3 | Experimental materials and tasks

Scratch webIDE has differences and similarities with Kibotics. First, one of the main differences is the programming environment: in Scratch, the programmed characters (e.g., a cat avatar) move in a 2D environment; Kibotics includes simulated robots in a 3D scenario. Second, Scratch scenarios do not simulate physical laws that affect the movement of your characters, such as friction, nor do they include preprogrammed obstacles and moving objects to interact with. Third, Scratch characters have motion instructions and sensors, but these are not designed to simulate the real actuators and sensors of educational robots such as those in Kibotics. Fourth, the Kibotics web IDE includes debugging tools so the student may see the measurements of the sensors of the virtual robot to debug the algorithm in execution time. On the other hand, they also have similarities, the main one is that both use the Scratch block programming language. Furthermore, in spite of the differences, both platforms allow working on the concepts of programming and CT.

The programming activities of Kibotics platform were selected with the objective of initiating the students in robot programming, because they did not have any previous experience in this type of programming. All of them are programmed in Scratch language. They were the following (from the "Learn robotics with Scratch" course):

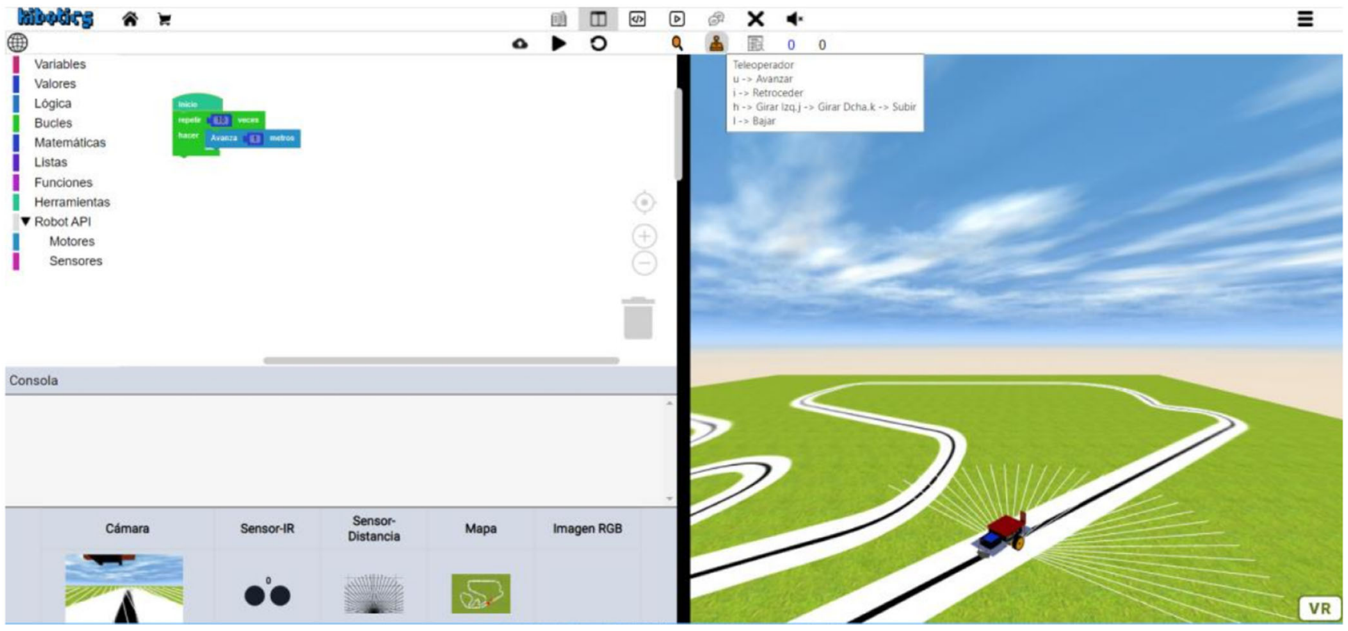


FIGURE 2 The robot programming and execution panel at Kibotics: code editor (left) and three-dimensional robot scenario (right).

1. *Square movement with GoPiGo robot*: It is an initiation activity where the user uses only the robot's actuators, learns, and works on the concept of open loop or "blind" robot programming while familiarizing the user with the Scratch blocks that control the robot's actuators. The user is asked to program an algorithm for the robot to move, describing a whole square, four straight segments returning to the starting point. The computational concepts worked on in this activity are Direction and repetitive loops.
2. *Bump and go behavior with GoPiGo robot using ultrasonic sensor*: It is an activity about the concept of closed loop using the sonar sensor. The student is asked to use the ultrasonic sensor to avoid collisions with obstacles on the floor (walls, furniture, etc.) while the robot is moving through a house with rooms. The computational concepts worked on in this activity are Directions, Repeat-Until loops, Simple conditionals (if), and Functions.
3. *Follow-Road behavior with GoPiGo*: In this activity, the infrared sensor is introduced. This sensor requires a more sophisticated algorithm to control the robot without going off the road or line. It consists of programming the robot to run through a circuit using a sensor that detects the path. The computational concepts worked on in this activity are Directions, Repeat-through loops, Compound conditionals (if-else), and Functions.
4. *Collect confetti with robotic vacuum cleaner*: This activity consists of improving the algorithm developed in the "Bump and go behavior with GoPiGo robot using ultrasonic sensor" activity but using a vacuum cleaner robot instead of the GoPiGo and beyond dodging the obstacles in the room, the robot should travel over the floor to collect as many confetti pieces as possible. The computational concepts worked on in this activity are Directions, Repeat-through loops, Simple conditionals (if), and Functions.

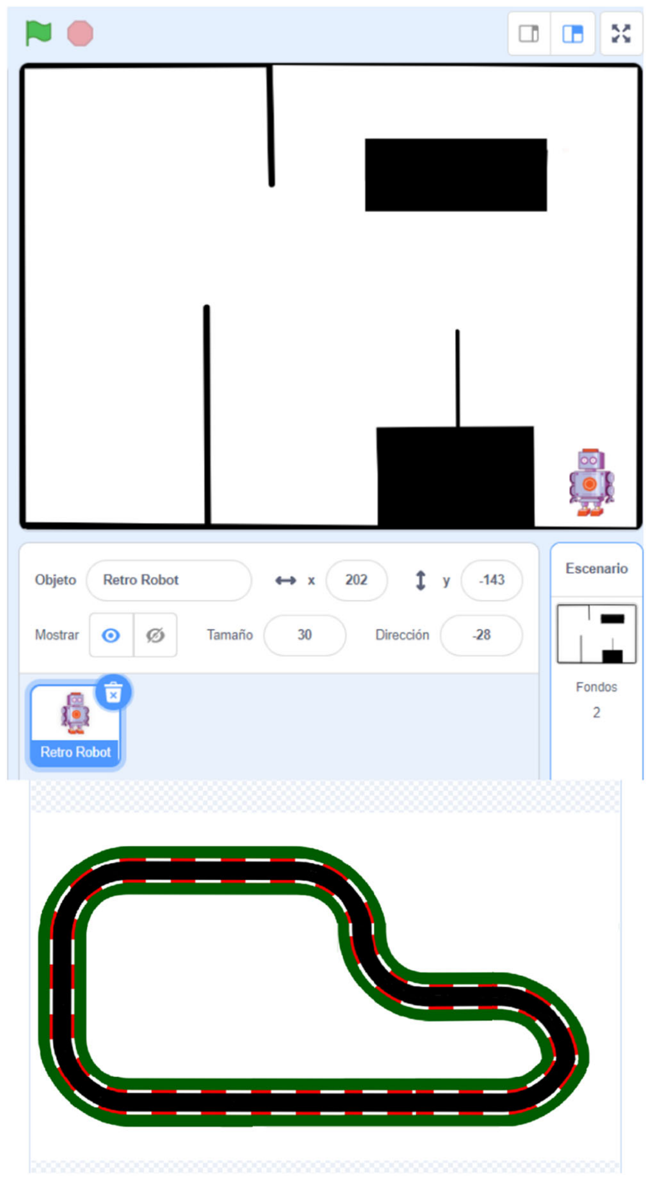
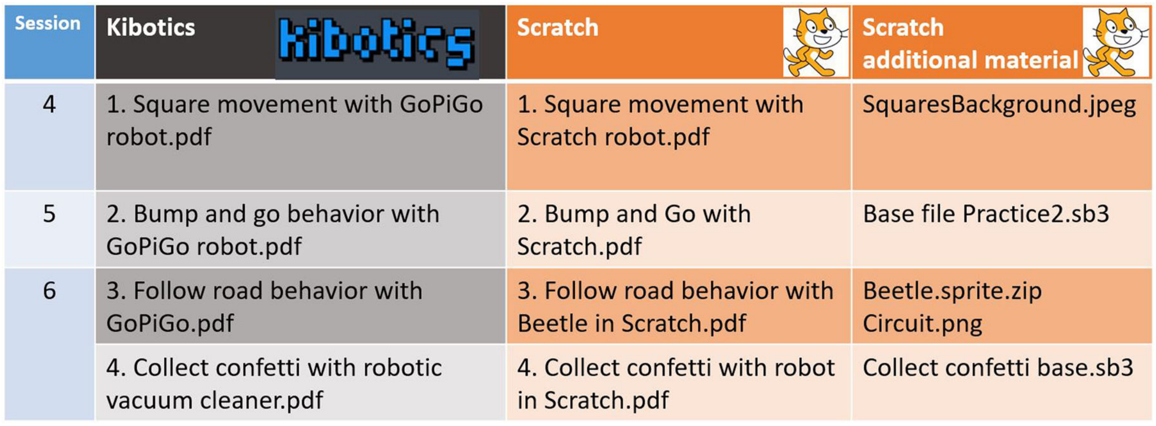

Objeto Retro Robot x 202 y -143
Mostrar Tamaño 30 Dirección -28
Retro Robot
Escenario
Fondos 2

FIGURE 3 Scenarios for activities 2 and 3 in Scratch platform: 2D avatar, a house with rooms to move along with the "Distance to ()" sensing block; and Circuit to follow with the "Touching Color ()?" sensing block. 2D, two-dimensional.

Once the activities in Kibotics were selected, the equivalent activities in Scratch were designed to allow the students in both groups, experimental and control, to work on the same concepts and algorithms. The same structure and context in the Kibotics platform were replicated in the Scratch activities because the students should have the same theoretical information to face the activities and to work with the same concepts. Not only the CT concepts, but also the used motion and sensing blocks in both platforms were very similar. For instance, the main used motor command in Scratch is the "GoTo XY" motion block, equivalent to the "Advance X m" or "Steer X degrees" blocks available at Kibotics for the square movement activity. They both cause the two-dimensional (2D) motion of the avatar or the robot. Regarding sensors, the "Distance to ()" sensing block at Scratch is equivalent to the "Get Sonar distance" block available at Kibotics. The "Touching Color ()?" sensing block in Scratch is equivalent to the "Get-IR-sensor" available at Kibotics for the follow-road activity. The Kibotics scenarios were also carefully replicated in Scratch (e.g., Figure 3). The main difference between both platforms was the 2D avatar in Scratch (the "retro robot" sprite and the "beetle" were used) versus the 3D robot in Kibotics. Figure 4 shows the guided assignments created for Kibotics and Scratch, as well as the Scratch additional materials handed to students to easily



Session	Kibotics	Scratch	Scratch additional material
4	1. Square movement with GoPiGo robot.pdf	1. Square movement with Scratch robot.pdf	SquaresBackground.jpeg
5	2. Bump and go behavior with GoPiGo robot.pdf	2. Bump and Go with Scratch.pdf	Base file Practice2.sb3
6	3. Follow road behavior with GoPiGo.pdf	3. Follow road behavior with Beetle in Scratch.pdf	Beetle.sprite.zip Circuit.png
	4. Collect confetti with robotic vacuum cleaner.pdf	4. Collect confetti with robot in Scratch.pdf	Collect confetti base.sb3

FIGURE 4 Scratch activities and additional material for each session 4–6 on the educational intervention (Figure 1).

replicate the robot-like environment of Kibotics for sessions 4–6 of the educational intervention (see Figure 1).

The activities and materials used during the workshops of both groups can be downloaded through the following link (https://bit.ly/3LvX1Nd) and are available in both English and Spanish.

3.4 | Variables

The dependent variables of the pre- and posttest are, first, related to learning results as measured with a validated CT questionnaire, see [53]. This questionnaire contains 28 items. For simplification purposes, pre- and posttest variables are rescaled from 0 to 10. Second, four dependent variables are related to the different computational concepts involved in the experiment: Directions, Loops, Conditionals, and Functions. In a more detailed study, new variables appear, with the aim of giving more information: Loops-Repeat-times, Loops-Repeat-until, Conditionals-Simple, Conditionals-Composite, Conditionals-While, and Functions-Simple. All of them scored from 0 to 10. One factor, Group, is considered to be an independent variable, related to the use or not of Kibotics. Table 2 contains a list of these variables, in the same order.

TABLE 2 Summary of variables, type (DV, dependent variable; IV, independent variable), name, and description.

Aspect	Type	Variable	Name
Learning programming and	DV	Pretest scores	Pretest
computational thinking	DV	Posttest scores	Posttest
Learning programming	DV	Direction	Direct
concepts	DV	Loops	Loops
	DV	Conditionals	Condit
	DV	Functions	Function
	DV	Loops-Repeat-times	Loops_t
	DV	Loops-Repeat-until	Loops_u
	DV	Conditional-simple	Condit_s
	DV	Conditional-composite	Condit_c
	DV	Conditional-while	Condit_w
Methodological factor	IV	Use of Kibotics	Group

3.5 | Procedure and tools

As a tool to measure CT and programming skills, the validated questionnaire "Computational Thinking Test" [53] is used, which is carried out before the programming activities (pretest) and after (posttest). The test was answered online, in Google Forms (https://bit.ly/4630bk1), which facilitates the distribution of the questionnaire, its automatic evaluation and the collection and processing of information. The questionnaire consists of a section at the beginning in which the student has to enter their anonymous unique participation code. Then, the instructions are presented, and examples of the types of questions and answers are shown, with the aim that the participants are clear about the dynamics of the questions. After the instructions, the CT test begins, consisting of 28 questions with four possible answers and only one of them correct. Each correct question adds 1 point to the total score of the test. Incorrect or unanswered questions do not subtract anything from the total score. After the 28 questions, two self-assessment questions are included, which do not count in the total result of the PC test. In them, the respondent is asked how well does he/she consider his/her performance in the test, and to what extent he/she is familiar with computers and information technology, which are answered on an 11-point Likert scale (from 0 to 10) (which allows greater sensitivity than the 5-point scale commonly used [9]). The maximum completion time is 45 min.

3.6 | Validity and reliability

The entire statistical study was carried out with IBM SPSS Statistics Version 25. This paper used questionnaires designed by [53] to evaluate CT. To measure the internal consistency of the results in the CT questionnaires both in pre- and posttest, as well as the questions asked to evaluate the programming concepts, Cronbach's alpha [22] is 0.765, an acceptable value. Deleting items does not increase this value.

4 | RESULTS

This study focuses, first, on the general results, where it is studied whether the use of the Kibotics platform influences the learning of programming concepts and CT. Second, on how these effects are distributed on each of the learned programming concepts. All scores have been rescaled to 10.

4.1 | Overall results

To generally measure this effect, the differences between the pre- and posttest scores are studied. If there are differences in both, they are quantified.

Table 3 shows mean, median, variance, minimum and maximum, main statistics of centralization, position, and dispersion for each of them.

As can be seen in Table 3, both groups, control and experimental, show a higher (or equal) value of both mean and median in the posttest compared with the pretest. In the control group, mean and median have close values, 5.39 and 5.35, respectively, for the pretest, and 5.46 and 5.35 for the posttest. In the experimental group, the pretest mean and median are 6.22 and 6.60, respectively. In posttest, similar values are shown, 6.76 and 6.78. Also, in both groups, control and test, the dispersion increases from 1.11 to 2.12 in the control group and 2.20 to 3.21 in the test group.

Figure 5 shows box plots with the pre- and posttest scores for both the control and experimental groups. It confirms visually what was explained before about centralization measures. Also, it shows a lack of homogeneity between the control and experimental groups in pretest scores. This difference between them is statistically checked with a t test for independent mean samples obtaining a $p = .005$ (previously, normality distribution of both groups is verified using a Shapiro–Wilk test, with $p > .05$), see Table 4. This lack of homogeneity in the pretest scores makes it impossible to directly compare the posttest scores, since the groups started from different prior knowledge. So, the study is focused on comparing the possible *increment* in the score between pre- and posttests for both groups.

Analyzing separately the evolution of each group, Table 5 shows the results of a test for paired samples based on Student's t test. In the case of the control group, the increment in the score between the pre- and posttests is not statistically significant ($p = .763$). However, in the experimental group, this increase is statistically significant ($p = .001$).

To take into account all these particularities that have been presented separately, a more advanced mathematical model is presented, analysis of variance (ANOVA) for repeated measures for two factors. In this model, several

TABLE 3 Descriptive analysis of the sample.

	Pretest		Posttest	
	Control	Experimental	Control	Experimental
Mean	5.39	6.22	5.46	6.76
Median	5.35	6.60	5.35	6.78
Variance	1.11	2.20	2.12	2.64
Minimum	3.21	3.21	2.86	3.21
Maximum	7.50	8.57	8.21	9.64

related dependent variables are analyzed, which appear two factors that interact in the model.

First of all, it has been verified that the required conditions to apply this model are verified, namely, normality of the data, sphericity (Mauchly, $p > .05$), and equality test in the covariance matrix (Box M tests, $p > .05$). Sex and class factors are discarded in the model for not being significant. Thus, the final model is presented in Table 6.

Table 6 reflects that, globally, there is a statistically significant improvement between the pre- and the posttests ($p = .023$) with a partial η^2 value of 0.60, which shows a high effect between both variables. In addition to this general increase, there is a significant statistical difference between the improvement of the control group and the experimental group ($p = .017$ for the Pre–post * Group interaction), being greater in the experimental group. This difference in increase can also be seen visually in Figure 6. Namely, the Group factor influences the model, that is, the scores do not evolve in the same way in the control group and in the experimental group. The influence, in this case, is moderate, as can be seen from the value of partial η^2, 0.37.

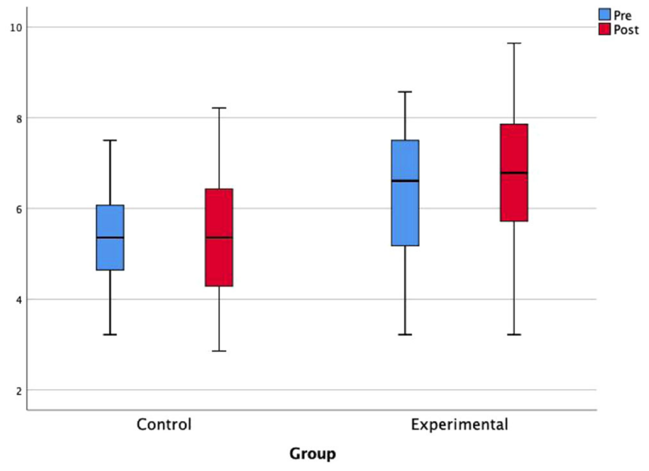


FIGURE 5 Box-plot for control and experimental groups in pre- and posttest.

TABLE 4 Results of Student's *t* test for independent samples in pretest scores.

	t	df	Significance
Pre	−2.89	83	.005

4.2 | Results by concepts

The analysis now focuses on seeing what happens when we consider each concept separately. These are Directions, Loops, Conditionals, and Functions. Table 7 shows the descriptive values for these variables in the pre- and posttest for the control group and the experimental group, which are also visualized in Figure 7.

As can be seen in the descriptive analysis, in the control group the mean values have decreased in Directions (Direct) and Loops. Conditionals (Condit) value has increased slightly (0.1 points) and Function is the one whose value has increased the most, albeit modestly (0.3 points). In the experimental group, all concepts have increased their average value, the improvement being between 0.4 and 1 points, except in the case of Function, where the average value is maintained.

Even so, all these previously mentioned improvements are only statistically significant in the experimental group, in the case of the Conditionals (Condit) variable, see Table 8 (*p* value marked in bold).

The descriptive analysis of the data related to the computational concept reflected in Figure 8 shows average differences in the scores obtained for each of them, according to the legend, respectively: Directions, Loops-Repeat-times, Loops-Repeat-until, Conditionals-Simple, Conditionals-Composite, Conditionals-While, and Functions-Simple. It is important to remark that scores for the concepts are evaluated between 0 and 10.

As observed, several concepts in the control group (Directions, Loops-Repeat-times, and Loops-Repeat-until) have decreased. In the Simple Conditionals concept, the average score is maintained, and in all the others, Conditionals-composite, Conditionals-while, and Functions, there has been a light improvement, except for Conditionals-composite concept, where it has been greater.

In the case of the test group, there have been improvements in all concepts, except in Functions, where the score has just been maintained. Table 9 shows

TABLE 5 Results of Student's *t* test for paired samples (differences between pre- and posttests) in control and experimental groups.

	Mean	Deviation	*t*	df	Significance
Control	−0.067	1.351	−0.304	36	0.763
Experimental	−0.535	1.057	−3.510	47	0.01

TABLE 6 ANOVA for repeated measured pre- and posttest.

	df	Quadratic mean	*F*	Significance	Partial η^2
Pre–post	1	3.802	5.334	0.023	0.60
Pre–post * Group	1	2.290	3.212	0.017	0.37
Error	83	0.0713			

Abbreviation: ANOVA, analysis of variance.

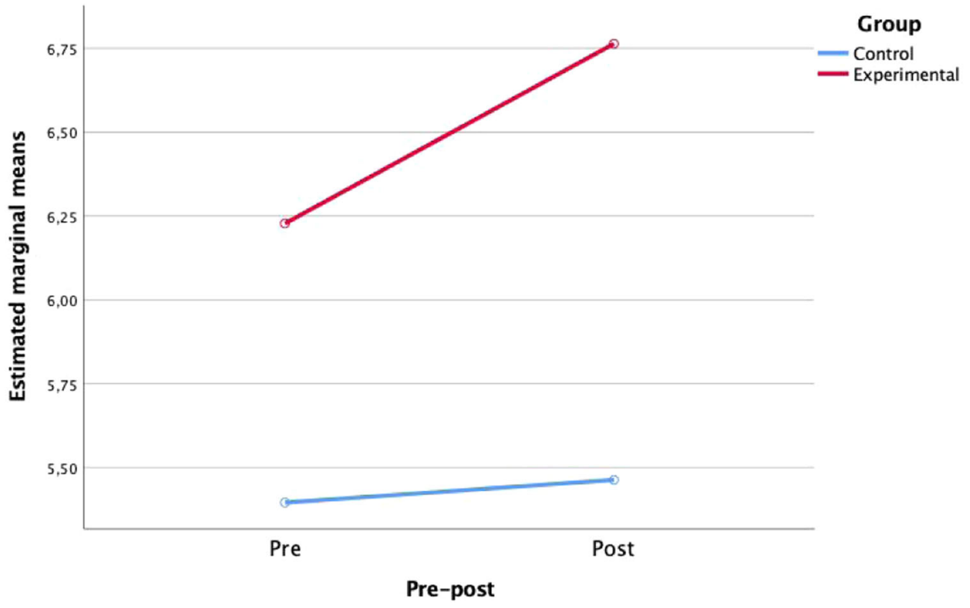


FIGURE 6 Estimated marginal means for pre- and posttest in control and experimental group.

which of these differences between pre- and posttest by concepts are statistically significant (*p* values marked in bold). For computing this, a test based on the Student's *t* test for paired samples is used. In addition, the differences in sample means are observed, as well as their standard deviation.

4.3 | Qualitative results

In the first session of practical exercises with the platforms (session 4 in Figure 1), both groups must program the movement of a robot so that it describes four corners that form a square. The experimental group, which is using the Kibotics platform for the first time, has greater difficulties in starting the activity than the group that uses the Scratch platform, which already knows the platform. However, the greatest difficulty that is seen is not the platform itself, but the way in which the activity is approached, since the students state that they have already programmed algorithms that make this movement but not using the actuators or movements of a robot, which means they have to rethink the algorithm and define the steps one by one without a prior guideline to follow. Another issue quite disconcerting for students is the unexpected behavior of robots on the Kibotics platform, which, unlike the Scratch platform, incorporates variables in the movement such as friction, which require adding additional conditions to the algorithm, using sensors to maintain control of the robot's movement. In the last session (session 7 in Figure 1), a retrospective was held with the students to gather additional information about their perception from the groups that used Kibotics. The students were invited to express their opinion and those who participated indicated that although Kibotics was based on the Scratch language, they found it less intuitive, which they related to the arrangement of the programming blocks and 'unexpected' movements, using as an example the square exercise, in which the robot moves differently than expected due to friction with the surface, which posed an added difficulty in the first steps on the platform. In addition, they suggested improvements to the platform such as the arrangement of the programming blocks in a single list, not within sections, and incorporating a block search engine. Another improvement they proposed was to incorporate the possibility of having more than one

TABLE 7 Descriptive analysis of the sample by concepts.

	Pretest		Posttest	
	Control	Experimental	Control	Experimental
Direct				
Mean	7.97	8.43	7.58	8.92
Median	7.50	10	7.50	10
Variance	3.41	3.62	3.95	3.02
Minimum	5	5	5	5
Maximum	10	10	10	10
Loops				
Mean	6.62	7.18	6.16	7.64
Median	6.25	7.5	6.25	7.5
Variance	2.94	2.82	4.57	2.41
Minimum	3.8	3.8	2.5	3.8
Maximum	8.8	10	10	10
Condit				
Mean	3.94	4.93	4.40	5.94
Median	4.16	5	4.16	6.25
Variance	2.26	3.52	3.20	3.63
Minimum	1.7	1.7	1.7	1.7
Maximum	6.7	8.3	8.3	9.2
Function				
Mean	4.73	6.04	5	6.03
Median	5	5	5	6.25
Variance	8.25	9.53	7.08	7.65
Minimum	0	0	0	0
Maximum	10	10	10	10

version of an activity, so that they could work on different solution approaches at the same time and do multiple tests without losing changes or be able to export the code to a file for versioning or sharing the code with third parties. All in all, their accurate suggestions for improvement of the platform reflect their mastering on the activities developed and the empowerment which Kibotics platform has got in the students that used it.

5 | DISCUSSION

5.1 | Answers to research questions

This paper explored whether Secondary Education students' CT can be improved by using simulated 3D robots and, to what extent students are able to learn programming concepts since programming is the most popular way of implementing CT skills as well as being a great tool to show students how CT can be applied to real-world problems [38]. We compared the differences between the impact of Kibotics ER platform, which includes simulated 3D robots, and Scratch platform on students' CT improvement. It included second-year Secondary Education (12–13-years olds) students and used a validated test to measure students' CT.

The hypothesis (H) was validated: there is a statistically significant increase in the posttest scores in CT when using the Kibotics platform on secondary education students, whereas using Scratch platform in this age group does not reach the same results. This contrasts with previous studies, as in [30] that used a TPACK Scratch Visual Execution Environment. This finding may be linked with what was found in [51], in

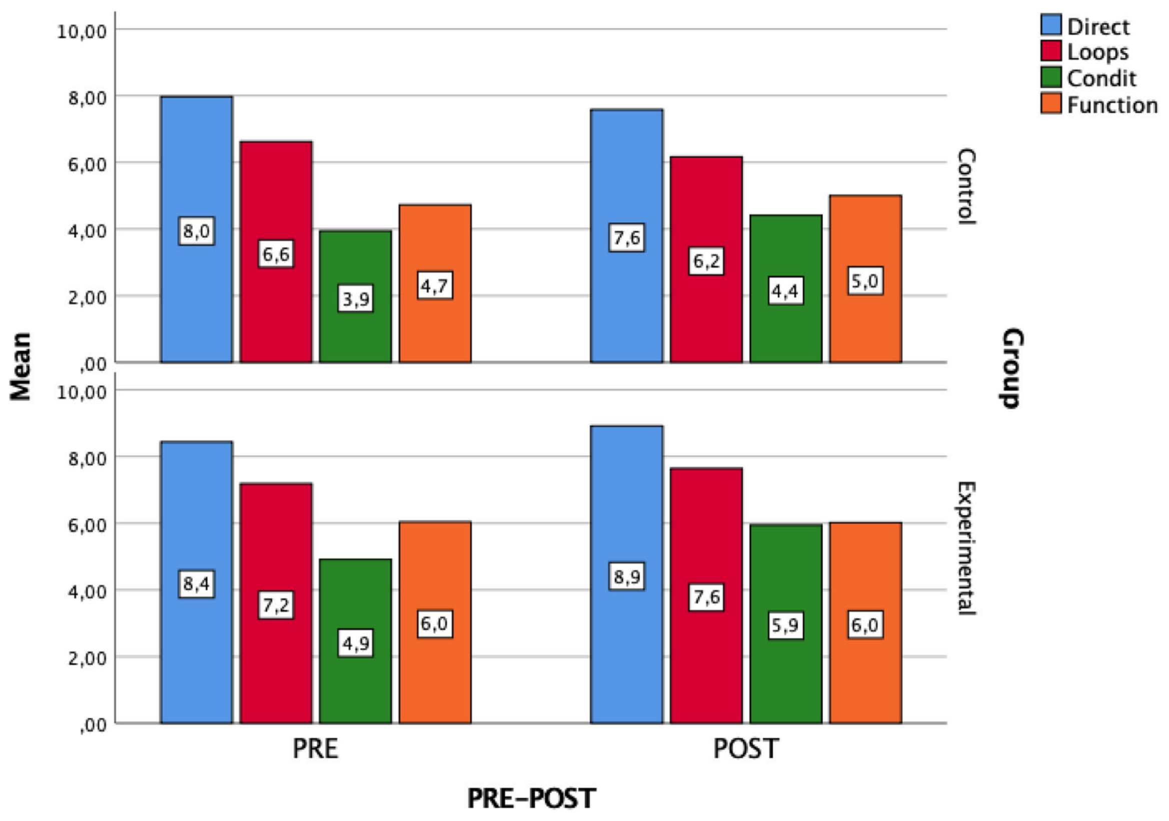


FIGURE 7 Graph of means for concepts variables in pre- and posttest.

TABLE 8 Results of Student's *t* test for paired samples (differences between pre- and posttest) in control and experimental groups for each concept.

	Mean	Deviation	*t*	df	Significance
Direct					
Control	0.347	2.167	0.961	35	0.343
Experimental	−0.372	2.017	−1.265	46	0.212
Loops					
Control	0.416	1.690	1.479	35	0.148
Experimental	−0.442	1.776	−1.726	47	0.091
Condit					
Control	−0.0661	2.119	−1.820	33	0.078
Experimental	−1.037	1.270	−5.475	44	**0.000**
Function					
Control	−0.147	4.033	−0.201	33	0.833
Experimental	0.156	2.935	0.369	47	0.714

which primary education students significantly increased their CT and learning of programming concepts by using a methodology based on metaphors and Scratch to learn them and with [50], where prescholers (nursery) used robots, such as Cubetto and KIBO (as well as other unplugged approaches) that resulted on the improvement of their skills of sequencing and plotting a route. Furthermore, similar findings were observed in first-year university students who used augmented reality-based virtual ER and also improved some of their CT Skills,

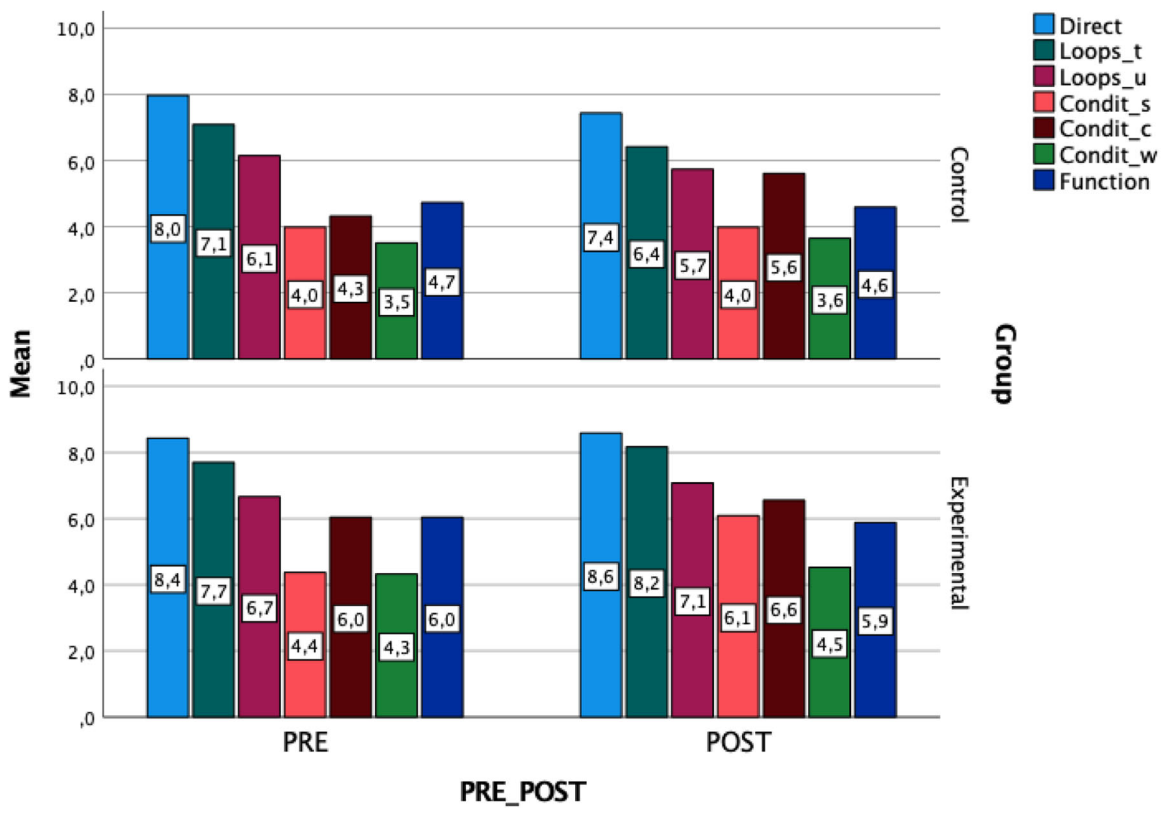


FIGURE 8 Graph of means for concepts variables in pre- and posttest (more complete study).

compared with those who used Scratch [44]. Therefore, it suggests that the context and the platforms used to teach CT may affect in several ways the different age groups of students.

This study presents an ER approach to foster CT perfectly guided and in two dimensions, instead of three, and whose characters do not require additional programming to adapt to an environment with their own simulated physical laws, such as friction, or unexpected obstacles and interaction with other moving objects. A guided intervention was performed, following [20] where they found that a noninstructional approach for ER promoted a trial-and-error behavior, and also that a guided path on the programming interface fostered the cognitive process of students and provided better scaffolding.

The RQ1, if it is possible to foster students' CT (directions, loops, conditional, and functions) by practicing concepts of computer science programming with simulated 3D robots, has been answered. We have shown in this paper that by practicing computer science concepts with the Kibotics platform in seven sessions increase significantly the CT of the students, whereas on the Scratch platform the improvement is not significant. Something similar was also found in [49] where Secondary Education students participated in a second year of using the Scratch platform within a project-based learning environment and improved their skills, gained and acquired CT knowledge but not significantly. This may suggest that using an ER platform with simulated robots and age-limited tasks in this age group may be more effective than a platform without simulated robots.

The RQ2, if are there concepts easier or better learned than others and to what extent?, when we consider each CT concept (directions, loops, conditionals, and functions) separately, has also been answered. In the control group the mean values have decreased in Directions and Loops, whereas Conditionals and Functions have increased slightly. In the experimental group, all concepts have increased their average value, with a higher improvement than in the control group, except in the case of Functions, where the average score is maintained. All these previously mentioned improvements are only statistically significant in the experimental group.

The descriptive analysis of the data related to the computational concept reflected in Figure 8 shows the differences, on average, of the scores obtained for each of them, respectively: Directions, Loops-Repeat-times, Loops-Repeat-until, Conditionals-Simple, Conditionals-Composite, Conditionals-While, and Functions-Simple. In the control group, in various concepts not only there

TABLE 9 Results of Student's *t* test for paired samples (differences between pre- and posttest) in control and experimental groups for each concept.

	Mean	Deviation	*t*	df	Significance
Direct					
Control	0.540	2.439	1.348	36	0.186
Experimental	−0.156	2.495		−0.434	0.666
Loops_t					
Control	0.675	2.177	1.888	36	0.186
Experimental	−0.4688	2.2864	−1.420	47	0.162
Loops_u					
Control	0.4054	2.6688	0.924	36	0.362
Experimental	−0.4167	2.4372	−1.184	47	0.242
Condit_s					
Control	0.0000	3.1732	0.000	36	1.000
Experimental	−1.7187	2.8320	−4.205	47	**0.000**
Condit_c					
Control	−1.2838	3.3136	−2.357	36	0.054
Experimental	−0.5208	3.4961	−1.032	47	0.307
Condit_w					
Control	−0.1351	3.1703	−0.259	36	0.797
Experimental	−0.2083	2.6714	−0.540	47	0.592
Function					
Control	0.1351	4.0802	0.201	36	0.841
Experimental	0.1563	2.9358	2.9358	47	0.714

has been no improvement, but also a slight worsening, such as in Directions, Loops-Repeat-times, and Loops-Repeat-until.

The shown results are for Secondary Education. Regarding their extensibility, the Scratch language is usually employed in the last courses of Primary Education, with previous good results too [30] and maybe similar effects would be expected when using 3D robots to foster CT. Regarding Nursery, good results have been observed for teaching CT by using robots without screens [50], such as Cubetto and KIBO, but still they are very different from the robots used in this research. Thus, these results may not be directly applied to that level.

5.2 | Threats to validity

This study presented some threats to validity [54] that might influence the results.

- *Construct validity*. The use of a computer can present some novel effects to students, since many of them abandon the routine and find something interesting and attractive for them. This innovation can create enthusiasm that makes it easier for students to achieve their goals [11]. The possible threat derived from the CT measurement instrument is solved by using a validated test [53].
- *External validity*. As it is not possible to randomize the individuals within each class to form the different control and test groups, the study cannot be certain that the sample is representative of the general population. Therefore, care must be taken when generalizing this result to all students of their age. However, it has been possible to randomize which class is assigned to the test group and to the control group, so that, in some way, it can be similar to a cluster sample [28], where the objective to be randomized is each group.

- *Internal validity*. Quasiexperiments avoid most of the threats to internal validity that arise in other kinds of experiments and randomizing the classes to assign them to be a part of either the control or the experimental groups eliminates selection bias, see [15, 17]. Even so, some maturation effects may be presented, such as getting older or more experienced. That all students had the same teacher could also have an impact, but in learning how to program with Kibotics and Scratch, the teacher is only for questions regarding problems with the computer, so this factor is not so relevant here.
- *Conclusion validity*. Campbell and Cook [16] called it statistical conclusion validity (SCV). They stressed that SCV "is concerned with sources of random error and with the appropriate use of statistics and statistical tests," so there is no perfect validity. Although a descriptive analysis is provided together with all assumptions needed in the inferential studies, type 1 error (incorrect rejection of null hypothesis) and type 2 errors (the failure to reject a false null hypothesis) are a part of statistical tests, a perfect result cannot be ensured, although it can be minimized. Assumptions about normality (required for all parametric tests used) and homoscedasticity (also required in the ANOVA procedure) in the data have also been tested. It is important to mention, as a general limitation when drawing conclusions, that there is a disparity in prior knowledge between the control and experimental groups. It is possible that a group, simply by having greater a priori knowledge, will have a greater improvement in their learning regardless of the method used. However, prior knowledge is investigated in many strands of the neurocognitive, psychological, and educational literature. Some currents in pedagogy affirm that prior knowledge does not necessarily imply greater learning, since other factors influence this process. For example, Simonsmeier et al. [56] performed a meta-analysis in which they found that prior knowledge indeed explained large portions of variance in learning outcomes, but it did not—on average—explain variance in learning gains. Brod [13] comments, in his work that unraveling the systematics of whether and how this prior knowledge then steers the learning process is not clear and is an issue for future research.

6 | CONCLUSION

This research paper has meant to prove the hypothesis (H) that it is possible to improve the CT skills of second-year students of secondary schools by using simulated 3D robots to teach computer programming and CT concepts, such as directions, loops, conditionals, and functions.

For this study, in the 2021/2022 academic year, we asked 85 second-year Secondary Education students (ages 12–13) to follow an intervention with seven 50-min sessions. A nonprobabilistic random sampling was carried out to divide them into two groups of students, a control group of 37 students, and an experimental group of 48 students. The groups were formed by randomly assigned groups of complete classes, two classes to the control group and two to the experimental group. The control group performed the programming activities at the Scratch web platform, which includes 2D avatars, and the experimental group performed them at the Kibotics ER web platform, which includes simulated 3D robots. The teachers (two) were the same in all classes. Both Scratch and Kibotics are online platforms; all students had user access to the platform of the corresponding group and completed the same activities individually. Both groups had exactly the same number of sessions and were assisted by the same two teachers.

Derived from the initial hypothesis, there were two research questions: (RQ1) Is it possible to foster the students' CT (directions, loops, conditional, and functions) by practicing concepts of computer science programming with simulated 3D robots? and (RQ2) are there some concepts easier or better learned than others and to what extent?

Regarding RQ1, in the analysis of the results from the validated CT test, it was found a lack of homogeneity in pretest scores between the control and experimental groups. This made it not possible to directly compare the posttest scores since they started with different prior knowledge. When analyzing the possible improvement of each group separately, in the case of the control group (Scratch), the increase in the score between the pre- and posttests is not statistically significant. However, in the experimental group (Kibotics) this increase is statistically significant. It can be concluded that the Group factor influences the model, that is, the scores do not evolve in the same way in the control group and in the experimental group. Therefore, the results derived from this study show that using the Kibotics platform with 3D simulated robots can significantly develop the students' CT, therefore, the hypothesis (H) is proven. So the experimental results confirm the positive response to RQ1.

To answer the second research question, RQ2) the study also focuses on analyzing what happens when we consider each concept separately. These are: Directions, Loops, Conditionals, and Functions. In the control group, the mean values have decreased in Directions and Loops. Conditionals have increased slightly and Function is the one that has increased its value the most, although this increase is really small. In the case of the experimental group, all concepts increased their average score except in the case of Function, where the average value did not change.

In addition, a more complete analysis was also performed with fine grain concepts, such as Loops-Repeat-times, Loops-Repeat-until, Conditionals-Simple, Conditionals-Composite, and Conditionals-While. In the control group, the Conditionals-Simple average score has been maintained, the improvement in Conditionals-composite has been significant and for the other concepts there has been a minimal improvement. In the case of the test group, there has been improvement in all concepts, except in Functions, where the calification has been maintained.

Regarding the results of the self-assessment of the students, no differences are obtained from their analysis, since they hardly change between the pre- and posttests. This allows one to infer that the use of one tool or another has not influenced the self-assessment of the result of the CT test or knowledge about computers in general.

Regarding the development of the learning sessions, it should be noted that the greatest difficulty was addressing the technical issues and incidents that arose during their course due to the high number of students involved in the activities and the short duration of each session, 50 min. However, despite the difficulties, it was motivating for the students to attend the computer workshops and put into practice the robotics concepts seen in class. On the other hand, despite the fact that it was easier for both groups a priori to use the Scratch platform because they already knew it from previous experiences, the feeling was that the students who used Kibotics maintained the highest motivation towards the last session.

For future lines of work, first, it is planned to do another intervention with control and experimental groups having similar knowledge and abilities on CT, expanding the number of schools and participant students, even of different ages. This will help to reduce the effect of confounding variables, such as prior knowledge. Second, we want to perform a gender analysis to explore differences, if any. And finally, the influence of using real robots on learning CT could be compared to using virtual robots.

ACKNOWLEDGMENTS

This work was supported by research Grants PID2022-137849OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by ERDF, EU. We thank the participants of this study and the anonymous reviewers for their useful suggestions. The educational intervention has the favorable assessment of the Ethics Committee of the Rey Juan Carlos University with the internal registration number 1901202203322.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ORCID

Luis Castro-San Martín https://orcid.org/0009-0001-8562-7085
Raquel Hijón-Neira https://orcid.org/0000-0003-3833-4228
Celeste Pizarro http://orcid.org/0000-0003-2447-8239
José M. Cañas https://orcid.org/0000-0003-4179-2211

REFERENCES

1. E. Ackermann, *Perspective-taking and object construction: Two keys to learning*. In Y. B. Kafai and M. Resnick (eds.), Constructionism in practice, designing, thinking, and learning in a digital world, Routledge, Abingdon, UK, 1996.
2. A. V. Aho, *Computation and computational thinking*, Comput. J. **55** (2012), no. 7, 832–835.
3. A. L. S. O. Araujo, J. S. Santos, W. L. Andrade, D. D. S. Guerrero, and V. Dagienė, *Exploring computational thinking assessment in introductory programming courses*, 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, USA, 2017, pp. 1–9.
4. S. Atmatzidou and S. Demetriadis, *How to support students' computational thinking skills in educational robotics activities*. In D. Alimisis, G. Granosik, and M. Moro (eds.), Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education, Padova, Italy, vol. **18**, 2014, 43–50.
5. S. Atmatzidou and S. Demetriadis, *Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences*, Robot. Auton. Syst. **75** (2016), 661–670.
6. E. Bakala, A. Gerosa, J. P. Hourcade, and G. Tejera, *Preschool children, robots, and computational thinking: A systematic review*, Int. J. Child–Comput. Interact. **29** (2021), 100337.
7. M. Berland and U. Wilensky, *Comparing virtual and physical robotics environments for supporting complex systems and computational thinking*, J. Sci. Educ. Technol. **24** (2015), 628–647.
8. M. U. Bers, *The tangiblek robotics program: Applied computational thinking for young children*, Early Child. Res. Pract. **12** (2010), no. 2, n2.
9. R. Bisterra and N. Pérez-Escoda, *Â£pueden las escalas Likert aumentar en sensibilidad?* REIRE, Rev. d'Innov. Rec. Educ. **8** (2015), no. 2, 129–147.

10. S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari, and K. Engelhardt, *Developing computational thinking in compulsory education—Implications for policy and practice*, European Comission, Joint Research Centr, 2016.
11. G. H. Bracht and G. V. Glass, *The external validity of experiments*, Amer. Educ. Res. J. **5** (1968), no. 4, 437–474.
12. C. Brackmann, D. Barone, A. Casali, R. Boucinha, and S. Muñoz-Hernandez, *Computational thinking: Panorama of the Americas*, 2016 International Symposium on Computers in Education (SIIE), Ediciones Universidad de Salamanca (España), Salamanca (España), 2016, pp. 1–6.
13. G. Brod, *Toward an understanding of when prior knowledge helps or hinders learning*, npj Sci. Learn. **3** (2021), no. 24, 1–3.
14. Y. A. Caballero-González and A. García-Valcárcel, Â£aprender con robótica en educación primaria? Un medio de estimular el pensamiento computacional. **21** (2020), 15.
15. D. T. Campbell, Factors relevant to the validity of experiments in social settings, Psycholo. Bull. **54** (1957), no. 4, 297–312. https://doi.org/10.1037/h0040950
16. D. T. Campbell and T. D. Cook, *Quasi-experimentation*, Rand Mc-Nally, Chicago, IL, 1979.
17. D. T. Campbell, and J. C. Stanley, *Experimental and quasi-experimental designs for research*, Rand McNally & Company, Chicago (USA), 1963.
18. C. Chalmers, *Robotics and computational thinking in primary school*, Int. J. Child–Comput. Interact. **17** (2018), 93–100.
19. G. Chen, J. Shen, L. Barth-Cohen, S. Jiang, X. Huang, and M. Eltoukhy, *Assessing elementary students' computational thinking in everyday reasoning and robotics programming*, Comput. Educ. **109** (2017), 162–175.
20. M. Chevalier, C. Giang, A. Piatti, and F. Mondada, *Fostering computational thinking through educational robotics: A model for creative computational problem solving*, Int. J. STEM Educ. **7** (2020), no. 1, 1–18. https://doi.org/10.1186/s40594-020-00238-z
21. CoderZ, *CoderZ web platform*. https://gocoderz.com
22. L. J. Cronbach, *Coefficient alpha and the internal structure of tests*, Psychometrika. **16** (1951), no. 3, 297–334.
23. V. Dagiene and G. Stupuriene, *Bebras—A sustainable community building model for the concept based learning of informatics and computational thinking*, Inform. Educ. **15** (2016), no. 1, 25–44.
24. J. Denner, S. Campe, and L. Werner, *Does computer game design and programming benefit children? A meta-synthesis of research*, ACM Trans. Comput. Educ. (TOCE). **19** (2019), no. 3, 1–35.
25. Fraunhofer-IAIS, *OpenRoberta web platform*. https://lab.open-roberta.org, 2002.
26. A. García-Valcárcel-Muñoz-Repiso and Y.-A. Caballero-González, *Robotics to develop computational thinking in early childhood education*, Comunicar. Media Educ. Res. J. **27** (2019), no. 1. https://doi.org/10.3916/c59-2019-06
27. GearsBot, *GearsBot web platform*. https://gears.aposteriori.com.sg/
28. M. H. Hansen, W. N. Hurwitz, and W. G. Madow, *Sample survey methods and theory. vol. i. Methods and applications*, Wiley, New York, 1953.
29. F. Heintz, L. Mannila, and T. Färnqvist, *A review of models for introducing computational thinking, computer science and computing in k-12 education*, 2016 IEEE Frontiers in Education Conference (FIE), IEEE Press, Eire, PA, USA, 2016, pp. 1–9.
30. R. Hijon-Neira, M. Garcia-Iruela, and C. Connolly, *Developing and assessing computational thinking in secondary education using a TPACK guided Scratch visual execution environment*, Int. J. Comput. Sci. Educ. Sch. **4** (2021), no. 4, 3–23.
31. A. Ioannou and E. Makridou, *Exploring the potentials of educational robotics in the development of computational thinking: A summary of current research and practical proposal for future work*, Educ. Inf. Technol. **23** (2018), no. 6, 2531–2544.
32. B. Jost, M. Ketterl, R. Budde, and T. Leimbach, *Graphical programming environments for educational robots: Open roberta-yet another one?* 2014 IEEE International Symposium on Multimedia, IEEE Press, Washington, DC, USA, 2014, pp. 381–386.
33. F. Kalelioglu and Y. Gülbahar, *The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective*. Inform. Educ. **13** (2014), no. 1, 33–50.
34. E. R. Kazakoff, A. Sullivan, and M. U. Bers, *The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood*, Early Child. Educ. J. **41** (2013), no. 4, 245–255.
35. Kibotics, *Kibotics web platform*. https://kibotics.org, 2018.
36. M. LafuenteMartínez, O. Lévêque, I. Benítez, C. Hardebolle, and J. D. Zufferey, *Assessing computational thinking: Development and validation of the algorithmic thinking test for adults*, J. Educ. Comput. Res. **60** (2022), no. 6, 1436–1463.
37. I. Lee, M. Fred, D. Jill, C. Bob, A. Walter, E. Jeri, M.-S. Joyce, and W. Linda, *Computational thinking for youth in practice*, ACM Inroads. **2** (2011), no. 1, 32–37.
38. J. Lockwood and A. Mooney, *Computational thinking in secondary education: Where does it fit? A systematic literary review*, Int. J. Comput. Sci. Educ. Sch. **2** (2018), no. 1, 41–60. https://www.ijcses.org/index.php/ijcses/article/view/26
39. S. Y. Lye and J. H. L. Koh, *Review on teaching and learning of computational thinking through programming: What is next for k-12?* Comput. Hum. Behav. **41** (2014), 51–61.
40. MIT, *Scratch web platform*. https://scratch.mit.edu, 2007.
41. R. Montes-León, H. Montes-León, D. Pérez-Marín, and R. Hijón-Neira, *Mejora del pensamiento computacional en estudiantes de secundaria con tareas unplugged*, 2020.
42. J. Noh and J. Lee, *Effects of robotics programming on the computational thinking and creativity of elementary school students*, Educ. Technol. Res. Dev. **68** (2020), 463–484.
43. F.-C. Ou-Yang, H.-M. Lai, and Y.-W. Wang, *Effect of augmented reality-based virtual educational robotics on programming students' enjoyment of learning, computational thinking skills, and academic achievement*, Comput. Educ. **195** (2023), 104721.
44. F.-C. OuYang, H.-M. Lai, and Y.-W. Wang, *Effect of augmented reality-based virtual educational robotics on programming students' enjoyment of learning, computational thinking skills, and academic achievement*, Comput. Educ. **195** (2023), 104721. https://www.sciencedirect.com/science/article/pii/S0360131522002925

45. I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar, and S. Lahmine, *Learning basic programming concepts by creating games with Scratch programming environment*, Procedia—Soc. Behav. Sci. **191** (2015), 1479–1482.
46. S. Papadakis, M. Kalogiannakis, and N. Zaranis, *Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: A case study*, Int. J. Mobile Learn. Organ. **10** (2016), no. 3, 187–202.
47. S. A. Papert, *Mindstorms: Children, computers, and powerful ideas*, Basic Books, New York City (USA), 1993.
48. A. M. S. Pardo, *Computational thinking between philosophy and stem–programming decision making applied to the behavior of "moral machines" in ethical values classroom*, IEEE Rev Iberoam. Tecnol. Aprendizaje. **13** (2018), no. 1, 20–29.
49. A. V. Pou, X. Canaleta, and D. Fonseca, *Computational thinking and educational robotics integrated into project-based learning*, Sensors. **22** (2022), no. 10, 3746. https://www.mdpi.com/1424-8220/22/10/3746
50. D. Pérez-Marín, R. Hijón-Neira, and C. Pizarro, *Coding in early years education: Which factors influence the skills of sequencing and plotting a route, and to what extent?* Int. J. Early Years Educ. **30** (2022), no. 4, 969–985.
51. D. Pérez-Marín, R. Hijón-Neira, A. Bacelo, and C. Pizarro, *Can computational thinking be improved by using a methodology based on metaphors and Scratch to teach computer programming to children?* Comput. Hum. Behav. **105** (2020), 105849. https://www.sciencedirect.com/science/article/pii/S0747563218306137
52. M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, *Scratch: Programming for all*, Commun. ACM. **52** (2009), no. 11, 60–67.
53. M. Román-González, J.-C. Pérez-González, and C. Jiménez-Fernández, *Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test*, Comput. Hum. Behav. **72** (2017), 678–691.
54. W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and quasi-experimental designs for generalized causal inference*, vol. **6**, Houghton Mifflin Company, Boston, NeW York, 2002.
55. V. J. Shute, C. Sun, and J. Asbell-Clarke, *Demystifying computational thinking*, Educ. Res. Rev. **22** (2017), 142–158.
56. B. Simonsmeier, M. Flaig, A. Deiglmayr, L. Schalk, and M. Schneider, *Domain-specific prior knowledge and learning: A meta-analysis*, Educ. Psychol. **57** (2021), no. 1, 35–54.
57. A. Sović, T. Jagušt, and D. Seršić, *How to teach basic university-level programming concepts to first graders?* 2014 IEEE Integrated STEM Education Conference, Princeton, NJ, USA, IEEE, 2014, pp. 1–6. https://doi.org/10.1109/ISEC32923.2014
58. A. Strawhacker, D. Portelance, M. Lee, and M. Bers, *Designing tools for developing minds: the role of child development in educational technology*. In M. U. Bers and G. Revelle (eds.), Proceedings of the 14th International Conference on Interaction Design and Children, Medford, MA, USA, ACM, 2015.
59. TIOBE-Organization, *Tiobe index*. https://www.tiobe.com/tiobe-index, 2020.
60. A. Unahalekhaka and M. U. Bers, *Taking coding home: Analysis of ScratchJr usage in home and school settings*, Educ. Technol. Res. Dev. **69** (2021), no. 3, 1579–1598.
61. L. Werner, J. Denner, S. Campe, and D. C. Kawamoto, *The fairy performance assessment: Measuring computational thinking in middle school*, Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, Raleigh North Carolina USA, ACM, 2012, pp. 215–220.
62. J. Wing, *Research notebook: Computational thinking–what and why*, Link Mag. **6** (2011), 20–23.
63. J. M. Wing, *Computational thinking*, Commun. ACM. **49** (2006), no. 3, 33–35.
64. J. M. Wing, *Computational thinking, 10 years later*, Microsoft Res. Blog. **23** (2016). https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/
65. E. B. Witherspoon, R. M. Higashi, C. D. Schunn, E. C. Baehr, and R. Shoop, *Developing computational thinking through a virtual robotics programming curriculum*, ACM Trans. Comput. Educ. (TOCE). **18** (2017), no. 1, 1–20.
66. K. Yang, X. Liu, and G. Chen, *The influence of robots on students' computational thinking: A literature review*, Int. J. Inf. Educ. Technol. **10** (2020), no. 8, 627–631.
67. L. Zhang and J. Nouri, *A systematic review of learning computational thinking through Scratch in k-9*, Comput. Educ. **141** (2019), 103607.

AUTHOR BIOGRAPHIES

Luis Castro-San Martín is a senior IT consultant, a graduate in computer science, and a master in secondary school education. He loves learning new things and collaborate in new projects. This is his first education research collaboration.

Raquel Hijón-Neira received the European PhD degree in Computer Science, and the Best Thesis Award from the IEEE Education Society. Her research interests include software for and innovation in programming education, educative technologies, computational thinking, and serious games. Member of the Laboratory of Information Technologies in Education (LITE) since 2007.

Celeste Pizarro is an assistant professor at the Department of Applied Mathematics at Universidad Rey Juan Carlos. Her research interests include Computer Science Education and different mathematical programming fields (stochastic, integer, and linear) and their applications. She has a PhD in Mathematical Modeling, a degree in Mathematics, and also a degree in Statistics.

José M. Cañas cofounded the RoboticsLabURJC in 2000. He leads the JdeRobot open-source organization. He is interested in robotics education at all levels, both for children and in higher education, and in robot programming with Artificial Intelligence technologies.

How to cite this article: L.-S. Martín, R. Hijón-Neira, C. Pizarro, and J. M. Cañas, *Fostering computational thinking with simulated 3D robots in secondary education*, Comput. Appl. Eng. Educ. **32**, (2024), e22740. https://doi.org/10.1002/cae.22740