Multimedia Tools and Applications
https://doi.org/10.1007/s11042-020-09202-z

RoboTIC: A serious game based on augmented reality for learning programming

Santiago Schez-Sobrino[1] · David Vallejo[1] · Carlos Glez-Morcillo[1] · Miguel Á. Redondo[1] · José Jesús Castro-Schez[1]

Received: 3 September 2019 / Revised: 26 May 2020 / Accepted: 8 June 2020 /
Published online: 02 July 2020

Abstract

Coding skills are becoming more and more important in today's world, especially within the context of the fourth industrial revolution. They also help practice other 21 century skills such as computational thinking, problem solving and teamwork. Unfortunately, learning how to program is tough and can be also frustrating for beginner students. In this work we introduce RoboTIC, a serious game based on gamification and Augmented Reality that facilitates the learning of programming to students in lower levels of the education system by using a novel set of visual metaphors derived from a notation of roads and traffic signs. The architecture that supports RoboTIC has been designed to allow the integration of multimedia components when new programming concepts and techniques must be addressed and to add game levels that enable students to learn incrementally. Experiments have been conducted in a youth center with children who do not have coding skills at all to demonstrate the feasibility of the proposal. The results show promising conclusions in terms of children's motivation and interest in programming.

Keywords Learning programming · Serious games · Visual programming · Augmented reality

✉ Santiago Schez-Sobrino
santiago.sanchez@uclm.es

David Vallejo
david.vallejo@uclm.es

Carlos Glez-Morcillo
carlos.gonzalez@uclm.es

Miguel Á. Redondo
miguel.redondo@uclm.es

José Jesús Castro-Schez
josejesus.castro@uclm.es

[1] University of Castilla-La Mancha, Paseo de la Universidad 4, Ciudad Real, 13071, Spain

1 Introduction

The learning of programming is one of the most important competences that can be acquired today. This is also considered to be critical in the area of the so-called fourth industrial revolution [39], in which the impact of the technologies such as artificial intelligence or robotics are changing the way we live, our relationships, and how we work together. Such is the case, that there are studies that predict that a significant part of the work currently being undertaken in this field will be totally automated [7], drastically modifying the economic model and generating, inherently, challenges and new business possibilities [2].

At the educational level, there are already ambitious programs based on including skills related to learning programming in the public curricula or, at least, after school programs. Countries such as Finland or South Korea, which have a very good reputation in terms of basic education and achieve excellent scores on the PISA tests, are representative examples in this respect. On a more general level, there are also initiatives such as the *Hour of Code* [29], aimed at introducing programming to people between the ages of 4 and 10, typically through the construction of interactive applications such as video games and robot programming, among others.

In stages prior to professional development, during primary and secondary education (K-12), it has been demonstrated that programming provides certain benefits over problem solving through so-called computational thinking, which implies the use of programming concepts through the analysis and recognition of repetitive patterns, the design of algorithms that allow solving problems step by step, the use of abstractions that simplify those problems and the decomposition of those problems into other simpler problems easier to be solved [50].

Unfortunately, programming is a complex task and therefore the teaching/learning process represents a significant challenge in the context of increasing students' chances of success in the new digital world. Thus, it is essential to motivate and stimulate the student by a qualified teacher who has the necessary mechanisms to adequately guide this process through approaches that go beyond the traditional ones that materialize in face-to-face classes. In fact, these approaches tend not to take full advantage of the cognitive abilities of the individual, which have been shown to be optimized to process information in a multimodal way through the combination of experiences involving the use of voice, gestures, sounds, and images, among others [30]. In this context, the use of active learning environments as traversal platform, based on a learn anywhere, anytime approach can contribute to increase the flexibility of the students when practicing and improving their skills as a programmer.

One of these techniques is to use gamification tools that include elements of video games in less related contexts [20]. Thus, these tools can be used to enhance the learning experience of programming by increasing the motivation of students and their participation in activities. This can be done in a general way, applying analogies to the evaluation process by replacing traditional terms with more gamified ones (e.g. current course by game, subject credits by points, learning units by levels, etc.) [13, 26], and more specifically, by creating video games that integrate programming concepts into their gameplay [17, 40].

In the field of video games, these concepts can be encompassed in the so-called serious games [8]. Thus, by means of specific games, learning mechanisms integrated in the gameplay can be included where the player has to solve programming challenges in order to advance in the game (e.g. [51]), and others where the player interacts with the game world by means of commands that represent sentences in a programming language. An example

of the latter would be LOGO [1] and its derivatives (e.g. [34]), which emerge as tools for teaching logic and programming concepts to the player.

Thus, both gamification and serious games allow to reduce the level of abstraction at the time of understanding programming concepts, through its relationship with elements in the context of video games. This relationship aims to facilitate the understanding of concepts explained through other familiar to students through the use of metaphors. In other words, using the concept of metaphor to make use of techniques that reduce the level of abstraction required during learning by creating visual metaphors that serve to understand specific concepts of programming [19]. These visual metaphors can be used in software visualization environments to represent the structure of programs, visualize the execution of an algorithm, or visually program applications [9].

The visual metaphors can be transferred to a three-dimensional space to benefit from different advantages, such as being able to show a greater amount of information due to the new dimension [38], use a realistic learning environment by using representations close to the real world [44] and cause a more effective use of the student's spatial memory in order to recognize and remember certain behaviors found in programs through the human subconscious [3, 28].

This three-dimensional space can be deployed in an Augmented Reality (AR) environment that allows interaction with visual elements in a more natural way, resulting in a more enriching experience for the student, given the potential benefits of using this approach in an educational context [4, 10]. Such AR environments can be leveraged to improve the learning gains in contrast to more traditional learning models [21], reduce the level of abstraction required by a virtual environment by including real-world elements [25] and improve the collaborative problem solving [10, 27], among others.

In this context, the present work introduces RoboTIC, a serious game designed to facilitate the learning of programming concepts in students of elementary educational levels (from 8 to 15 years), in order to develop their computational thinking through the ideas of gamification, serious games, visual metaphors, and 3-D representation and interaction. The main objective of the proposal is to motivate the learning and self-learning, to improve the understanding of fundamental programming concepts and to facilitate the comprehension of problem solving techniques. RoboTIC incorporates elements of the real world through a set of metaphors based on roads and traffic signs visualized in a three-dimensional environment of AR and grouped in different levels that the student must solve. The proposed architecture has been designed to add, in a scalable way, multimedia resources that facilitate the integration of new programming elements and new levels so that the students themselves can approach them in the context of an incremental complexity.

RoboTIC has been evaluated through an analysis of the serious game with a group of children, using an experiment that consisted of several phases and made it possible to evaluate in terms of learning basic programming concepts such as instructions, conditional sentences and loops.

The main contributions of this article are as follows:

- Firstly, we introduce RoboTIC, a serious game for learning programming concepts through a block based programming environment, supported by a scalable architecture that allows to integrate new levels in an incremental way.
- Secondly, an evaluation of RoboTIC is conducted by children that are faced with the challenge of completing a level where basic programming concepts are addressed.
- Thirdly, RoboTIC has been developed considering leveraging 3-D graphics and using AR gameplay mechanisms actively in the game levels.

– Fourthly, natural interaction mechanisms are introduced to play RoboTIC, as well as other physical elements to affect the virtual world.

The rest of the paper is organized as follows. In Section 2, some similar tools and environments based on learning programming concepts are described, as well as introducing the approach of this work. Section 3 focuses on the tool proposed, its architecture and its main features. In Section 4, the research questions raised and an evaluation of the tool with actual children are presented. Finally, Section 5 draws some final conclusions, discusses the experimental results and suggests possible lines of future work.

2 Related work

Within the scope of serious games intended to train in programming competitions there is a wide variety of alternatives proposed by several authors. The proposal for this work is based on LOGO [1], although the existing literature also includes other related works. This is the case of PlayLOGO 3D [36], an implementation of LOGO on a 3D environment that allows turn-based battles between players using movement commands to try to collide with the other player's character. Other examples would be Cube Game [37] and Lightbot [18], where you have to solve block labyrinths by guiding the main character to a specific position using movement commands. The main contribution of these works lies in introducing the operation of a computer through predefined commands, making players see that a computer can only understand a limited set of instructions. The reader is invited to go to [46], which includes a systematic review of these and other similar serious games in the context of learning programming.

Other more advanced alternatives venture into the field of visual programming [6] using tools such as Scratch [31] or Blockly [15], which expand the possibilities of the previous tools by allowing to define the logic of more complex programs. These tools would be oriented to introduce concepts of programming languages, such as the use of variables, functions, loops, conditional sentences, mathematical operations, etc.

In line with the above but in relation to the metaphors used during the learning of programming, in [47] it is proposed a visual programming system that uses real-world metaphors to build programs, such as telephones to call functions, boxes for variables, and arrows next to people to represent entry and exit commands, among others. Also, one would find the set of metaphors presented in [41] to visually represent the execution of programs. These metaphors represent the possible roles that a variable can take, for example, if it is a variable that will increase in a loop, a series of footprints will be used in the ground with the past and future values that the variable will take, while if it is an array, it will be made by means of a set of tombs with the values of the array sculpted on them.

In the field of 3-D graphics there are also other works that aim to facilitate the learning of programming through different metaphors. This is the case of [32], where the authors used three-dimensional animations involving pipes and cubes to represent data assignments between variables, as well as other concepts such as planes to represent scopes of functions, among others. In a more abstract way, in [22], avatars and 3-D animations are used to represent programming concepts, such as message passing 3 between objects (avatars). Also, in a more advanced way using Virtual Reality techniques, in [33] it is proposed a system that facilitates the understanding of concurrent programming concepts through actors and 3-D boxes.

Some other works intend to improve the computational thinking and the process of learning to program by including different contents based on AR. In [35] a tool combining immersive environments and 2-D interaction is proposed to foster interest in computational thinking through dancing. In [16] an AR book is proposed to explain electronic circuits and programming using Scratch for Arduino. In [11] was presented a programming toy that consisted of physical markers used to build a virtual map on which players could program a character's movement to reach a target through an Android application. In the same line, in [23], a similar tool is proposed but the character's movement is performed through setting physical cards on the board. Finally, in [45] a setup consisting of a computer, a camera and physical cards with QR codes was used to facilitate the learning of 3-D programming concepts like translating an object through the 3-D space, change the lights in the scene, etc.

The tool proposed in this work aims to improve the current situation by using AR as a necessary gameplay mechanism to complete the levels of the game and keep the players motivated. In the list of previous works, AR was used by means of physical markers to slightly affect some gameplay mechanics (e.g. visualizing information). In the case of the proposal of this work, AR is intended to be used (i) passively so that the player uses the perspective of the level to discover the best strategy to solve it and (ii) actively to increase the motivation of the player to understand programming concepts such as conditional sentences through physical cards. These AR mechanisms are intended to be leveraged using more natural interaction ways that reduce the existing limitations of physical markers. Moreover, the visual metaphor based on roads and traffic signs that is included in RoboTIC is actively used in the game to reinforce the explained programming concepts by representing the execution flow, loops and conditional sentences. This visual metaphor, called ANGELA, has been proposed in previous works [42, 43] where it was successfully evaluated as a notation to facilitate the learning of programming.

3 RoboTIC approach to learn computational thinking competence

3.1 General overview

RoboTIC is an educational serious game oriented to the learning of elementary programming concepts such as, for example, execution flows, conditionals and loops, among others, and thus improving the overall computational thinking. The application is mainly aimed at children between the ages of 8 and 15 who want to start programming through video-games.

It includes different multimedia game elements (graphic content, audio and multimodal interaction) that facilitate the comprehension of different programming concepts. On the one hand, there is the execution flow of the program, which is defined by a sequence of graphic elements based on the metaphor of roads and traffic signs, derived from the ANGELA notation. This notation allows to compose the structure of a program and the associated logic through the use of roads and traffic signs, respectively. In the case of RoboTIC, an abstraction of the metaphor has been used to simplify and facilitate the understanding of the game to the target audience. Thus, the loops are represented by a roundabout and a traffic sign indicating the number of times the body of the loop will be repeated. In the case of conditional sentences, the use of a bifurcation is proposed where the possible conditions are predefined beforehand by means of a series of conditional cards that represent the possible events that may lead to one or the other path of the bifurcation.

The video-game represents the virtual world applying AR technologies that allow the player to interact with that world through the use of natural user interfaces (NUI). These

means of interaction allow to influence the virtual world through the use of gestural and voice interfaces, or through the alteration of elements of the physical world. In such way, AR in RoboTIC is used to keep players motivated towards the levels they have to solve. In the same way, it is used as a game mechanic through the use of AR stickers and the perspective itself, forcing the player to explore the level from different angles to get a correct solution of the level.

Each level in RoboTIC presents a challenge that the player must solve by instructing the main character (robot) how to proceed to reach the goal of the stage. This stage is built of blocks that share the same size, so that the movement of the robot through the level is done discreetly. The actions that the robot can perform for the level are indicated by the player in the form of commands that the robot will perform sequentially. Thus, the player could communicate to the robot two commands to advance, one to turn to the right and another one to advance, to achieve a movement similar to the one made by a horse in chess. In case the goal is in the final position reached by the robot, the player would complete the level; otherwise, the sequence of orders would have to be corrected with a new alternative. These commands are executed by a small replica of the robot (mini-robot), which moves over the sequence of instructions at the same time as the robot executes them.

The video-game is compiled for the Microsoft HoloLens AR device and is available for download at: www.esi.uclm.es/www/santiago.sanchez/robotic/RoboTic_1.2.4.2.appxbundle. Fig. 1 shows a snapshot of RoboTIC running one of the available levels.

To support the functionality provided by RoboTIC, the system architecture is defined by different layers that communicate with each other, maintaining a division of responsibilities based on what each layer brings to the user. This division of responsibilities facilitates its expansion and improvement depending on the domain to be affected:

– **Perception layer**: is in charge of obtaining information from the user and the surrounding environment for further processing. Specifically, the AR device is responsible for analyzing the real world, the movement of the user, the actions it performs, etc. to allow interaction with the system. This layer provides the user with the necessary interaction mechanisms to influence the game world.

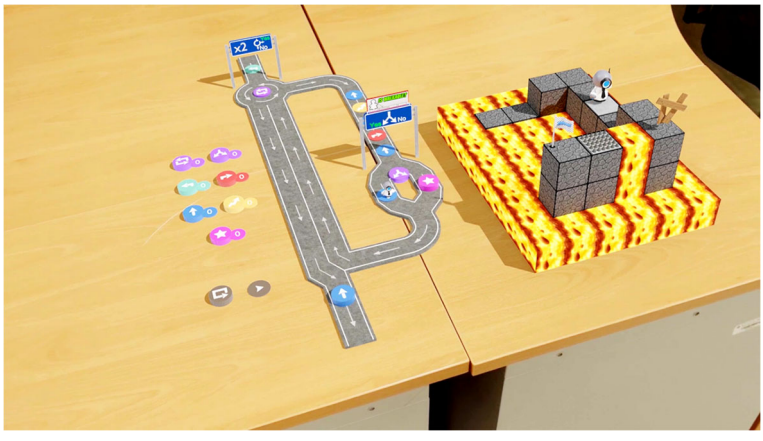


Fig. 1 A snapshot of the RoboTIC game running the level used during the evaluation. A video playing this full level can be found at https://youtu.be/T1I2mrX7BUw

– **Cognitive layer**: this layer processes the behavior defined by the user and assigned to the robot of the virtual world with which it interacts, as well as the influence of other elements of the physical world on the virtual one and the load of levels. This layer provides the user with the challenges that must be solved to overcome the levels of play, as well as the validation of the rules defined.
– **Representation layer**: this layer is dedicated to the construction of the virtual world from metaphors of the real world that facilitate the assimilation of programming concepts. This layer provides the user with the visual feedback and awareness mechanisms necessary to maintain their motivation and interest during the game experience.

Fig. 2 shows a diagram of those layers with the main components that define them, as well as the elements with which the user interacts in the game.

3.2 Perception layer

The domain of the perception layer supposes the users' entry point to RoboTIC, so it is necessary that it offers simple interaction mechanisms that the user knows how to take advantage of to maintain their interest at all times. For this reason, and given the capabilities it offers over other alternatives, the Microsoft HoloLens AR device has been selected as the execution medium for the video-game and the Unity video-game engine for its development.

The device makes it possible to analyze the environment and detect real-world elements such as the floor, ceiling, walls and tables, among others. In this way, the perception layer is able to identify any horizontal surface and use it to load the level of the game on it. These horizontal surfaces are typically the floor or the tables, although the seats of the chairs are also identified as such. The perception layer will only load the game levels onto the real-world elements identified as tables or on the floor.

The game levels are adapted to the size of the surface, never exceeding one square meter in size so that the level can be viewed completely through the field of view of the AR device at an appropriate distance. These surfaces are divided in half into two distinct parts, being the first half (i) the part of the game area that contains the sequence of instructions along with the mini-robot and the second half (ii) the part that shows the level of the game to be solved and the robot.

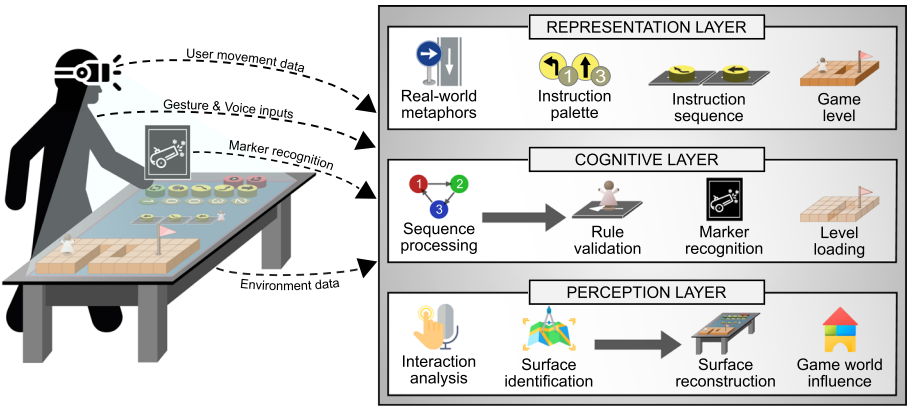


Fig. 2 General overview of the layers and elements that comprise the system

At the same time, this layer also provides the player with different interaction mechanisms to influence the game. The main method would be based on gestures recognized by the AR device and that allow the user to select the instructions that the robot will execute on the level, besides allowing the interaction with the different elements of the interface. Another method of interaction is provided implicitly by the device and the sensors it includes for positioning and orientation, allowing the user to move through the physical world to observe the level of the game from all its angles in order to identify the appropriate sequence of instructions that solves the level. Finally, interaction mechanisms are also integrated through recognition of physical marks that allow special actions to be performed on the level, such as freezing a lava path or destroying an obstacle for the robot to advance.

The perception layer shares with the cognitive layer the information related to the surface where the game level is executed, as well as the information and actions triggered by the interaction that the player makes with it.

3.3 Cognitive layer

The cognitive layer is where the processing of the information obtained from the perception layer is performed, in such a way that the robot executes the sequence of instructions defined by the user.

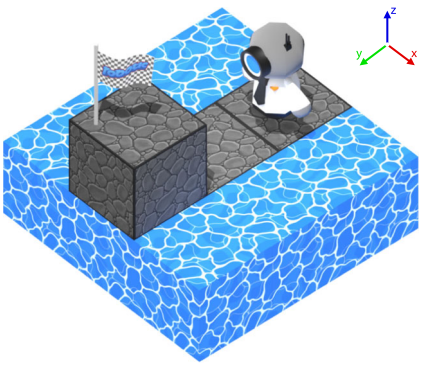
The instructions that can be executed by the robot are divided into control instructions, condition instructions and execution instructions. The former and the second correspond to the definition of the logical structure of execution by means of loops, conditional sentences and the conditions themselves, while the latter include those instructions that indicate to the robot the actions to be performed. The simile used is that the execution instructions tell the robot what to do, while the control and condition instructions define how to do it.

This layer is also in charge of the logic necessary to load the game levels. The game initially has ten levels, which present a progressive difficulty to the player. During the first levels the player is introduced with the execution instructions, and then give way to the introduction of the control instructions and conditions. Finally, the player must use the physical marks to overcome the last levels.

The levels are stored as files in JSON format that contain all the necessary information to be loaded during execution and replicate the full level. This information includes: (i) the size of the level indicated in maximum number of blocks along each axis of coordinates, (ii) the location where the robot will appear, (iii) the location where the target or goal to be reached by the player will be found, (iv) the number of instructions of each type that can be used in the level, (v) the items available on the level and (vi) the location and type of each of the blocks that compose the level represented by numerical identifiers. An example of a level and its internal representation in the specified data file is shown in Fig. 3. Specifically, the key `map` of the level data file defines the blocks that form the level through a two-dimensional list that indicates the type of block that exists in each position, according to the Equation 1

$$B_{(X,Y,Z)} = B_{i,j} \mid i = X + w \times Y,\ j = Z \tag{1}$$

where B is the type of block that occupies the position (X, Y, Z) of the level, i is the index of the list corresponding to the coordinates (X, Y), j is the index of the list that indicates the height of the level on the Z axis and w is the width of the level along the X axis. Thus, the block $B_{(1,2,1)} = B_{7,1}$ would correspond to block type 0, i.e. the stone block on which the level goal is located in the figure.




```
{
  "level-size": [3, 3, 2],
  "player-start": [1, 0, 0],
  "goal": [1, 2, 1],
  "available-instructions": {
    "condition": 0,
    "loop": 0,
    "turn-right": 0,
    "turn-left": 0,
    "jump": 1,
    "move": 1,
    "action": 0
  },
  "items": [],
  "map": [
    [1], [0], [1],
    [1], [0], [1],
    [1], [1, 0], [1]
  ]
}
```


Fig. 3 A simple 3x3 level with two layers (left) along with its defining JSON data (right)

This loading of levels facilitates the addition of new levels, requested at the beginning to a web service that is in charge of serving the available list of levels. Thus, users can load their own levels in this web service so that they are available to the rest of the players, in such a way that a selection with the desired levels can be configured from the game.

3.4 Representation layer

The representation layer is oriented to the rendering of the different virtual elements that shape the game interface and the levels, both the part where the sequence of instructions is defined and the part where the level is built.

On the one hand, the game interface is defined by various menus and feedback messages, which provide the user with information about each state of the game.

During the resolution of the levels, the player is introduced mainly with the graphic elements that conform the level, the robot and the mini-robot. In the part of the level where the sequence of instructions is defined, the mini-robot would be located on a fragment of road. This road will increase in size as new instructions are added to the sequence and finally the mini-robot will move through it, executing the instructions that it finds in its path for the robot to move.

To the left of the sequence of instructions is the palette of instructions, each of which is represented by means of circular buttons. Here, the player can select those actions that he/she wants to add to the sequence, and once he/she has finished, execute the sequence to try to make the robot reach the target. The user has total freedom to remove from the sequence those instructions that he/she does not consider useful or replace them with others, as long as he/she respects the limitation on the number of instructions of each type available for the level. In special cases where a control instruction is added from the palette, the player's focus will shift to the sequence of instructions to complete the parts required by that control instruction.

The control instructions regarding conditional sentences are represented by a bifurcation on the road, assuming the left lane as the sequence to be executed if the condition is met, while the right lane would contain those instructions that will only be executed if the condition is not met. The condition to be evaluated in this type of control instructions can be selected by the player from a list of cards with predefined conditions.

In the case of control instructions concerning loops, their visual representation would be made by means of several sections of road arranged in the form of a roundabout, thus

reflecting the concept of repetition. These instructions are limited only to the repetition of a number of times of the body of the loop (right part). The number of repetitions can be increased by interacting directly on the signal.

4 Experimental results

In order to evaluate the proposal of this work, an experiment has been conducted with children where they tested the game in a real scenario of AR with the Microsoft HoloLens visualization device and in the context of a level that addressed programming concepts related to instructions, conditional sentences and loops. This section describes the research questions raised, the performance of the experiment, the usage scenario, the participants, the results obtained and the possible limitations faced. Discussion of the results and their relation to the research questions formulated are left to the reader in the Section 5.

4.1 Participants

This experiment involved 12 children (9 boys and 3 girls) from the youth center of Torralba in Calatrava (Ciudad Real, Spain). The average age of the participants was 12 years old. They were recruited from a campaign based on the city council's social networks and promotional posters on the walls of the youth center. They did not receive any incentive to participate in the experiment. None of them knew the game or had tried it before. None of the organizers of the experiment knew the participants beforehand.

Due to the above, the authors consider that the results obtained from the experiment are not biased by the profile of the participants or the way in which they were recruited.

4.2 Method

To validate the usefulness of the proposal, the following research question arises, which we can be divided into more specific ones:

- **RQ**: Does the use of programming learning tools based on AR improve the children's motivation and interest in programming?
    - **RQ1**: Does understanding and knowing the game and how it works intrinsically increase children's interest in programming?
    - **RQ2**: Does using AR interfaces improve children's motivation for programming?
    - **RQ3**: What is children's subjective perception of the ease of use, usefulness and intention of use of the game?

In order to answer these questions, an experiment has been proposed following the experimental process of Wohlin et al. [48] and formulating the experimental objective using the template provided in the Goal-Question-Metric (GQM) [5] as shown below:

*To **analyze** the RoboTIC serious game for learning programming **with the purpose of** evaluating the user experience and to know the subjective opinion **with regard to** ease of use, usefulness, intention, interest and motivation of this game, **from the point of view** of children attending the school **within the context of** an scenario where there is no background in programming knowledge nor AR.*

Therefore, the experiment aims to demonstrate the usefulness of the tool for acquiring knowledge to solve problems using computational thinking, the ease of use, and its motivational component. For this purpose, a learning activity has been designed for children with the profile mentioned above, and with reduced groups of participants to avoid distractions.

Fig. 4 graphically illustrates the experimental design followed in this learning activity, describing the elements involved in each phase of the evaluation.

The experiment consisted of 2 phases; a first phase of preparation and another phase of intervention that included pre-test, development and post-test.

During the preparation phase the participants were divided into 3 groups of four children each given the reduced age of the participants, in order to isolate them in a room during the experiment to reduce possible distractions and thus achieve a working group more focused in the experiment. In addition, a tutorial was given on the use of the game and the AR device, explaining the possibilities of interaction, the functioning of the game and the objective they had to reach to solve the level. Fig. 5 shows a photo taken during this phase with the 4 members of the first group.

In the intervention phase, each working group performed three tasks of the experiment: i) pre-test, ii) development and iii) post-test. The total duration of the experiment for each working group was between 20 and 30 minutes, considering that each participant had to perform the development task individually. For both the pre-test and the post-test, the participants completed surveys in which several questions were asked in order to evaluate certain variables. In both tests, questions that were not open-ended had to be answered in a 5-point Likert scale. The answer options were replaced by *emojis* to make them easier to be understood by the participants [24] (1: *strongly disagree* being the angriest emoji to 5: *strongly agree* being the happiest emoji):

- *pre-test*: the participant's demographic information was requested, i.e. sex, age, current education level and desired future education level. In addition, some general questions were asked in the context of programming, in order to know their personal experience and opinion about video games, computers, and more specifically, about programming. In the Table 1 are included the questions of the questionnaire for this pre-test.
- *post-test*: some general programming questions were repeated again (G5A, G8A-G11A) in order to determine whether they had changed their minds in any of their answers after the development phase. This test mainly included questions aimed at evaluating the participants' subjective perception of the game, inspired by the Technology Acceptance Model (TAM) [12] and analyzing the variables Perceived usefulness (PU), Perceived ease-of-use (PEOU) and Intention of use (ITU) of the model, as well as others aimed at learning about the participants' intrinsic motivation. Finally, some open-ended questions were included where participants could indicate what they liked most about the experience and what they liked least. In the Table 2 are included the questions of the questionnaire for this post-test.

During the development phase, participants had to solve a RoboTIC level where various elements of the game were used, such as AR stickers, instructions, use of loops and conditions (listed in Fig. 4). Thus, the level introduced a programming problem that had to be solved by a combination of the above elements (see video referenced in Section 3). The instructions allowed to define the sequence of commands to be executed at the level, while the loops and conditions allowed to define the execution flow.

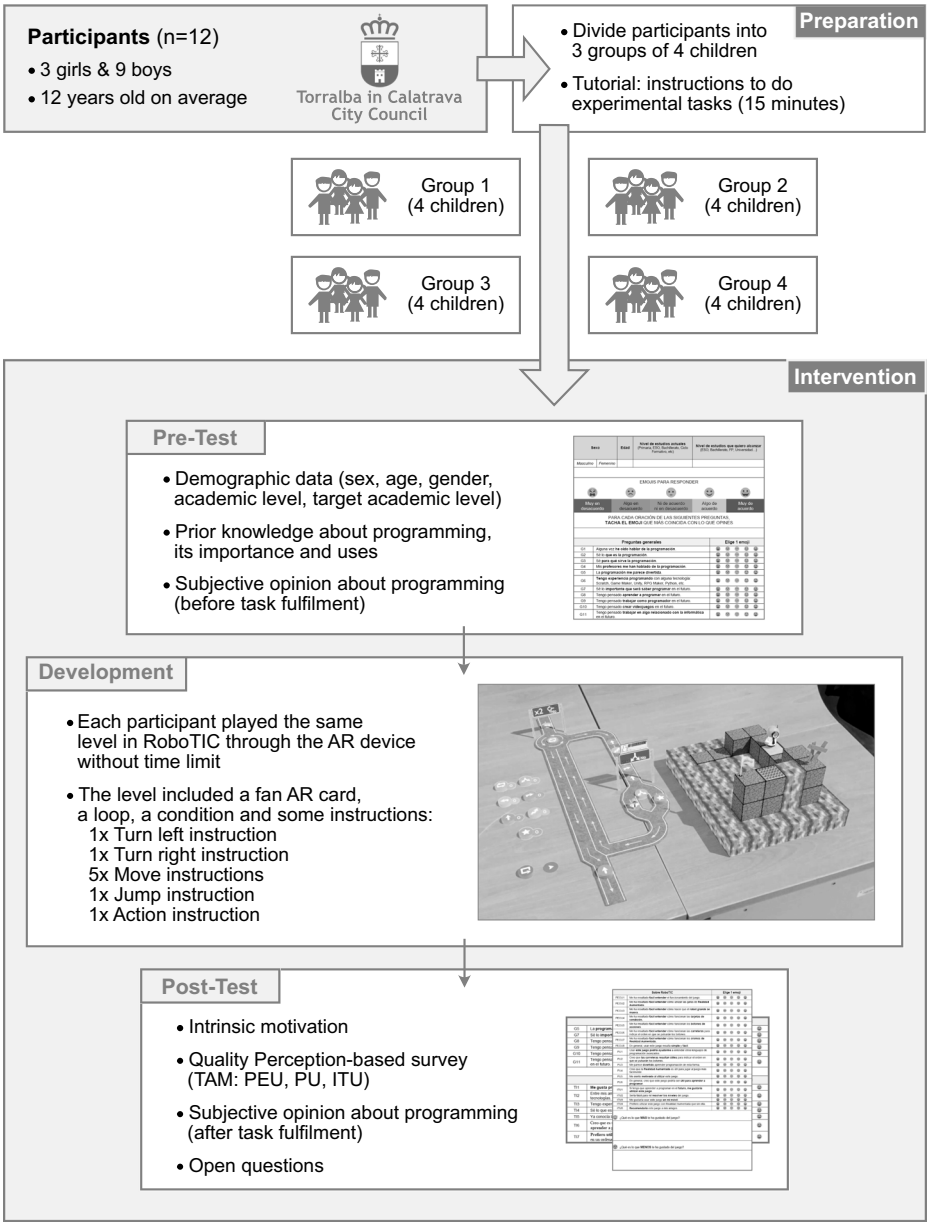


Fig. 4 Experimental design

Each participant had to put on the AR device and start solving the level individually. The Microsoft HoloLens device allows streaming video via the web in order to see what the person wearing the device is seeing at that moment. This was used to see what the participants were doing to solve the level and how they interacted with the game.

Fig. 5 Photo taken during the preparation phase of the experiment

4.3 Results

First, the data obtained in the pre-test were analyzed considering the answers whose value on the Likert scale was greater or equal to 3, obtaining percentages of participants from the grouping of similar questions. This analysis concluded that 92% of the participants wanted to pursue higher education, only 12.5% had heard about programming, knew what it is or what it is used for (G1-G4 items) and none of them had previously programmed (G6 item), didn't want to dedicate to programming in the future (G8-G9 items), didn't know its importance (G7 item) or didn't think it was fun (G5 item), although 33.33% wanted to dedicate to creating video games or to computing in the future (G10-G11 items).

After analyzing again the same questions in the post-test part after the development phase and having been able to test a level of RoboTIC, it was found that now 66.67% of the

Table 1 Items in the survey for the pre-test

Item	Item statement
G1	I've ever heard of programming.
G2	I know what is programming.
G3	I know what programming is used for.
G4	My teachers have told me about programming.
G5	I find programming fun.
G6	I have experience programming with some technology: Scratch, Game Maker, Unity, RPG Maker, Python, etc.
G7	I know how important it will be to know how to program in the future.
G8	I intend to learn to program in the future.
G9	I intend to work as a programmer in the future.
G10	I intend to make video games in the future.
G11	I intend to work on something related to computer science in the future.

Table 2 Items in the survey for the *post-test*

Item	Item statement
G5A	I find programming fun.
G8A	I intend to learn to program in the future.
G9A	I intend to work as a programmer in the future.
G10A	I intend to make video games in the future.
G11A	I intend to work on something related to computer science in the future.
AR1	I have understood what is Augmented Reality.
AR2	I already knew the Microsoft HoloLens or other Augmented Reality device.
AR3	I think it's useful and fun to use Augmented Reality to learn how to program.
AR4	I prefer to use tools that use gestures and Augmented Reality rather than keyboard and mouse to learn how to program.
PEOU1	I found it easy to understand how the game works.
PEOU2	I found it easy to understand how to use the Augmented Reality device.
PEOU3	I found it easy to understand how to make the robot to move.
PEOU4	I found it easy to understand how the condition cards work.
PEOU5	I found it easy to understand how the action buttons work..
PEOU6	I have found it easy to understand how roads work to indicate the order in which buttons will be pressed.
PEOU7	I found it easy to understand how the Augmented Reality stickers work.
PEOU8	In general, using this game is simple and easy.
PU1	Using this game could help me understand other advanced programming languages.
PU2	I think roads are useful to indicate the order in which the buttons will be pressed.
PU3	I think it's fun to learn programming this way.
PU4	I think augmented aeality is useful for playing the game more easily.
PU5	I feel motivated to use this game.
PU6	Overall, I think this game could be useful for learning how to program.
ITU1	If I have to learn to program in the future, I would like to use this game.
ITU2	It would be easy for me to solve the levels of the game.
ITU3	I would like to use this game on my smartphone.
ITU4	I'd rather use this game with Augmented Reality than without it.
ITU5	I would recommend this game to my friends.

participants thought programming was fun (item G5) and 62.5% plan to learn to program in the future, dedicate to computing or work creating video games (item G8-G11).

The conclusions drawn from the above variables are justified by applying the McNemar's test for two related samples [14], with the aim of demonstrating the difference between these

variables before (G5, G8-G11) and after (G5A, G8A-G11A) the development part during the intervention phase. These variables have been transformed into dichotomous variables where the values of the Likert scale between 1 and 2 would represent a disagree response from the participants, and the values between 3 and 5 an agree response. Thus, the test would give the results shown in the Table 3.

As can be observed, there are significant differences for the variables G5, G8, G9 and G11 ($p < 0.05$), thus confirming the impact that the development part has had on the participants. In the case of variable G10 there would be no significant differences ($p = 0.508$; $p > 0.05$), from which it can be assumed that the participants will not necessarily want to commit themselves to the creation of video games in the future.

Regarding the intrinsic motivation of the participants when using Augmented Reality tools, in the Table 4 the values for averages, standard deviations and medians of the related post-test items (ARx) are shown. The results obtained for items AR3 and AR4 show a mean score on the Likert scale above 4 indicating that participants were more predisposed to use augmented reality technologies to learn programming, even though they had never used an AR device before (item AR2). The AR1 item is directly related to the explanation given to the participants of what AR is previously in the preparation phase, demonstrating with a score on the Likert scale higher than 4 that they had understood the explanation.

In relation to the participants' subjective perception of ease of use (PEOU), utility (PU) and intention of use (ITU), the results obtained for the means, standard deviations and medians of the items of the post-test involved are shown in the Table 5, as well as the overall mean of each variable category. The results obtained show scores higher than 4 on the Likert scale for all cases except for the item PEOU4, whose standard deviation (0.79) is also far from that of the rest of the items. This may indicate that RoboTIC condition cards are one of the most difficult elements of the game to understand. On the contrary, the item PEOU3 presents a score of 5 with a standard deviation of 0.0, indicating a consensus among the participants on the understanding of the functioning of the movement of the main character from the instruction buttons. In particular, we would like to highlight the item PEOU6 related to the metaphor of roads and traffic signs, which allows us to validate the simplification of the notation ANGELA as suitable for the context of serious games.

The validity of such results can be ensured by a Wald-Wolfowitz runs test [49] that confirms that the responses obtained from the participants are randomly enough, i.e. the sample values are mutually independent. The results after running such test is shown in the last columns of the same table, where we can conclude that the majority of the values of the variables are randomly distributed ($p < 0.05$), with the exception of the PU5 and ITU3 variables and the PEOU1, PEOU3 and PEOU4 variables, not being possible to perform the

Table 3 McNemar test results for variables G5, G8, G9, G10 and G11 before and after the development phase

Items related	Sig.	Interpretation
G5 & G5A	0.008	Significant differences
G8 & G8A	0.016	Significant differences
G9 & G9A	0.008	Significant differences
G10 & G10A	0.508	Non-significant differences
G11 & G11A	0.008	Significant differences

Table 4 Mean and median values for the intrinsic motivation regarding the AR items. Standard deviations are shown in parentheses

Item	Mean	Median
AR1	4.33 (0.49)	4
AR2	1.08 (0.29)	1
AR3	4.58 (0.67)	5
AR4	4.42 (0.51)	4

runs test for these last three variables due to the reduced diversity of values obtained from the participants.

Finally, the open-ended questions were answered in a variety of ways, focusing mainly on the fact that the participants liked the graphic aspect of the game, namely the instruction buttons, the roads and the movement of the mini-robot through them.

4.4 Findings and suggestions

After conducting the learning activity, we can extract some valuable experience that could be transferred to a real class environment. Using RoboTIC in such environment would imply the design of a series of game levels of incremental complexity aimed to explain some of the programming concepts in the way they are usually structured:

Table 5 Mean and median values for the TAM framework variables; perceived ease-of-use (PEOU), perceived usefulness (PU) and intention of use (ITU), along with the runs test result and its interpretation. Standard deviations are shown in parentheses

Item	Mean	Median	Mean (global)	Sig.	Interpretation
PEOU1	4.17 (0.58)	4.0	4.35 (0.40)	-	Unable to compute
PEOU2	4.50 (0.52)	4.5		0.762	Randomly distributed values
PEOU3	5.00 (0.00)	5.0		-	Unable to compute
PEOU4	3.58 (0.79)	3.0		-	Unable to compute
PEOU5	4.50 (0.52)	4.5		0.364	Randomly distributed values
PEOU6	4.33 (0.49)	4.0		0.908	Randomly distributed values
PEOU7	4.50 (0.52)	4.5		0.130	Randomly distributed values
PEOU8	4.25 (0.45)	4.0		1.000	Randomly distributed values
PU1	4.50 (0.52)	4.5	4.49 (0.19)	0.762	Randomly distributed values
PU2	4.25 (0.45)	4.0		1.000	Randomly distributed values
PU3	4.67 (0.49)	5.0		0.051	Randomly distributed values
PU4	4.33 (0.49)	4.0		1.000	Randomly distributed values
PU5	4.75 (0.45)	5.0		0.012	Non-randomly distributed values
PU6	4.42 (0.51)	4.0		0.145	Randomly distributed values
ITU1	4.58 (0.51)	5.0	4.67 (0.13)	0.405	Randomly distributed values
ITU2	4.67 (0.49)	5.0		0.206	Randomly distributed values
ITU3	4.75 (0.45)	5.0		0.012	Non-randomly distributed values
ITU4	4.50 (0.52)	4.5		0.130	Randomly distributed values
ITU5	4.83 (0.39)	5.0		0.843	Randomly distributed values

1. Sentences: players would define the set of instruction buttons that would serve to complete a simple game level, for example, by moving the robot just two tiles forward. These instructions are executed sequentially, resembling how programs are executed line by line. This is the same as programmers would write their programs using code.
2. Combination of sentences: in more complex levels, the players would need to use a combination of all the available instruction buttons to complete the level. This would imply that the players plan and evaluate the level beforehand to achieve the best combination of instructions to solve the game level.
3. Conditional sentences: next concept to understand would be conditional sentences. These ones would allow students to complete the game levels depending on random events happening in the game and that require specific flow control.
4. Conditional sentences (nested): as before, more complex in-game events would require more complex conditional sentences. In these levels, the user is introduced in how to nest conditional sentences to solve more advanced problems (game levels).
5. Conditional sentences (AR cards): another way to explain flow control through conditional sentences is by using AR cards that can be used to trigger some behavior in the game levels. Therefore, this mechanism can be used to motivate the students learning this concept.
6. Loops: after introducing conditions, students can use the concept of loops to accomplish repetitive tasks. This concept is the same one used in programs made by programmers.
7. Loops (nested): as before, more complex tasks would require nesting a loop within another one.
8. Functions: finally, the last game levels would involve using the function concept to execute instruction sequences made for other levels but which can be reused for the new level. In this way, students can easily understand the reuse and modularization of programs.

Then, the teacher would present the concept in class and then would provide the game to the students so that they can play the related game level to better understand the explained concept. If the students complete the level of play, it would mean that they have understood the related concept successfully.

4.5 Study limitations

Despite the results obtained, there are some limitations and risks for the external validity of the study. In terms of the nature of the participants, given the small size of the sample treated, it has not been possible to conduct an exhaustive hypothesis validation, leading to a descriptive statistical analysis. Regarding the experimental task performed, although the game level that the participants have tested is complete enough to include all the elements of the game and to deal with all the available programming concepts, experimental tasks that respect the expected learning flow are considered, so that the participants learn programming concepts incrementally.

5 Conclusions and future work

In this work we have presented RoboTIC, a serious game oriented to learning programming and improve the computational thinking for children who have no coding skills. The different components that are integrated into the RoboTIC architecture have been introduced

within the context of the learning possibilities it provides thanks to the use of AR and the game mechanics designed to motivate students. This layered architecture is scalable enough to add new levels and multimedia resources that make possible to address new programming concepts and techniques. Such scalability is achieved through the usage of different data files that allow to create new game levels and load them without having to rebuild the tool. In case of requiring more advanced resources, the architecture facilitates adding new game elements, such as new AR cards or game level blocks.

The analysis of the results obtained after the evaluation of RoboTIC concludes answering the research questions posed at the beginning of this research work. Thus, with regard to the children's motivation and interest in programming (RQ1) it is concluded that the students have felt more motivated after playing RoboTIC and have been more receptive to learning programming. Regarding AR, the study reveals that its use as an immersive technology improves the children's motivation towards programming, making learning more attractive (RQ2). Finally, the analysis of the subjective perception towards ease of use, usefulness and intention to use RoboTIC shows that the serious game proposed in this work is easy to understand and use (RQ3). The affirmative answers to the raised research questions imply, therefore, returning to the initial research question: *Does the use of programming learning tools based on augmented reality improve the children's motivation and interest in programming?*, which can be answered affirmatively considering, however, the limitations previously stated in Section 4.5.

The evaluation of RoboTIC and the execution of the experiment have allowed us to extract some valuable experience that could be reused in real class environment. Thus, we have proposed a possible learning path on how to use RoboTIC in the classroom by firstly explaining the programming concept and then leaving the students play the related game level to check if they have really understood such concept.

As lines of future work, the inclusion of new programming concepts are considered to be included in RoboTIC, such as the use of functions to introduce structured programming or recursion as a more clean way to reduce program complexity when coding. Migrating RoboTIC to another hardware platform with a lower cost than the Microsoft HoloLens will be also addressed so that the tool becomes more accessible. Thus, it is considered a simplified version that works on smartphones using some technology such as ARCore (Android) or ARKit (iOS), or using physical marks that are employed for placing the game levels using AR; in this case, we would have to assess new NUIs that allow for easy use of the system. In the same way, we also intend to introduce RoboTIC in education centers that are specialized in teaching coding skills and robotics. This will allow us to conduct a more exhaustive study of the current version of RoboTIC, involving a greater number of students. Finally, the integration of another gamification techniques is proposed to keep players motivated, such as, for example, medals and achievements depending on the student's performance when solving a level.

Acknowledgements This work has been funded by the Ministry of Economy, Industry and Competitiveness, and the European Regional Development Fund through the project TIN2015-66731-C2-2-R. The authors would like to thank Pablo Gutiérrez Caravantes for coordinating the experiments carried out in the youth center of Torralba in Calatrava (Ciudad Real, Spain) and the undergraduate students that participated in the project *Telefónica Talentum* for the development of the first software prototype.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

References

1. Abelson H, Goodman N, Rudolph L (1974) LOGO manual
2. Bloem J, Van Doorn M, Duivestein S, Excoffier D, Maas R, Van Ommeren E (2014) The fourth industrial revolution. Things Tighten 8:1–40
3. Burgess N, Maguire EA, O'Keefe J (2002) The human hippocampus and spatial and episodic memory. Neuron 35(4):625–641
4. Bacca J, Baldiris S, Fabregat R, Graf S (2014) Augmented reality trends in education:, A systematic review of research and applications. Educational Technology and Society 17(4):133–149
5. Basili VR, Caldiera G, Rombach HD (1994) The goal question metric approach. Encyclopedia of Software Engineering 2:528–532
6. BENTRAD S, Meslati D (2011) Visual programming and program visualization – towards an ideal visual software engineering system –. ACEEE International Journal on Information Technology 1:56–62
7. Colombo AW, Karnouskos S, Kaynak O, Shi Y, Yin S (2017) Industrial cyberphysical systems: a backbone of the fourth industrial revolution. IEEE Ind Electron Mag 11(1):6–16
8. Connolly TM, Boyle EA, MacArthur E, Hainey T, Boyle JM (2012) A systematic literature review of empirical evidence on computer games and serious games. Computers & education 59(2):661–686
9. Diehl S (2007) Software visualization: visualizing the structure, behaviour, and evolution of software. Springer Science & Business Media
10. Dunleavy M, Dede C (2014). In: Spector JM, Merrill MD, Elen J, Bishop MJ (eds) Augmented reality teaching and learning, 4th eds. New York, Springer, pp 735–745
11. da Silva Esteves AM, Santana ALM, Lyra R (2019) Use of augmented reality for computational thinking stimulation through virtual
12. Davis FD (1993) User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. International journal of man-machine studies 38(3):475–487
13. Elshiekh R, Butgerit L (2017) Using gamification to teach students programming concepts. Open Access Library Journal 4(08):1
14. Eliasziw M, Donner A (1991) Application of the mcnemar test to non-independent matched pair data. Statistics in medicine 10(12):1981–1991
15. Fraser N (2015) Ten things we've learned from Blockly. In: 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond), IEEE, pp 49–50
16. Figueiredo M, Cifredo-Chacón MÁ, Gonçalves V (2016) Learning programming and electronics with augmented reality. In: International Conference on Universal Access in Human-Computer Interaction, Springer, pp 57–64
17. Gallego-Durán FJ, Villagrá-Arnedo CJ, Llorens Largo F, Molina-Carmona R (2017) Plman: A game-based learning activity for teaching logic thinking and programming. International Journal of Engineering Education
18. Gouws LA, Bradshaw K, Wentworth P (2013) Computational thinking in educational activities. In: Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13, ACM Press, New York, New York, USA, pp 10
19. Hidalgo-Céspedes J, Marín-Raventós G, Lara-villagrán V (2016) Learning principles in program visualizations: a systematic literature review. In: Proceedings of the 46th Annual Frontiers in Education (FIE) Conference. IEEE, pp 1–9
20. Ibanez MB, Di-Serio A, Delgado-Kloos C (2014) Gamification for engaging computer science students in learning activities: a case study. IEEE Transactions on learning technologies 7(3):291–301
21. Jee HK, Lim S, Youn J, Lee J (2014) An augmented reality-based authoring tool for e-learning applications. Multimedia Tools and Applications 68(2):225–235
22. Jimenez-Diaz G, Gonzalez-Calero PA, Gomez-Albarran M (2012) Role-play virtual worlds for teaching object-oriented design: the viRPlay development experience. Software:, Practice and Experience 42(2):235–253
23. Krpan D, Mladenović S, Ujević B (2018) Tangible programming with augmented reality. In: 12th International Technology, Education and Development Conference
24. Kaye LK, Malone SA, Wall HJ (2017) Emojis: insights, affordances, and possibilities for psychological science. Trends Cogn Sci 21(2):66–68
25. Kim TJ, Huh JH, Kim JM (2018) Bi-directional education contents using vr equipments and augmented reality. Multimedia Tools and Applications 77(22):30089–30104
26. Kumar B, Khurana P (2012) Gamification in education-learn computer programming with fun. International Journal of Computers and Distributed Systems 2(1):46–53

27. Kaufmann H, Schmalstieg D (2002) Mathematics and geometry education with collaborative augmented reality. In: ACM SIGGRAPH 2002 conference abstracts and applications, pp 37–41
28. Knight C, Munro M (2000) Virtual but visible software. In: Proceedings of the IEEE International Conference on Information Visualisation, pp 198–205 https://doi.org/10.1109/IV.2000.859756
29. Majumdar A (2018) The hour of code: an initiative to break the barriers of coding. XRDS 24(3):12–13. https://doi.org/10.1145/3186711
30. Myers BA (1990) Taxonomies of visual programming and program visualization. Journal of Visual Languages & Computing 1(1):97–123
31. Maloney J, Resnick M, Rusk N, Silverman B, Eastmond E (2010) The scratch programming language and environment. ACM Trans Comput Educ 10(4):1–15
32. Milne I, Rowe G (2004) Ogre: Three-dimensional program visualization for novice programmers. Educ Inf Technol 9(3):219–237
33. Mathur AS, Ozkan BK, Majumdar R (2018) Idea: An Immersive Debugger for Actors. In: Proceedings of the 17th ACM SIGPLAN International Workshop on Erlang, St. Louis, MO USA, ACM, pp 1–12
34. Paliokas I, Arapidis C, Mpimpitsos M (2011) PlayLOGO 3D: A 3D Interactive Video Game for Early Programming Education: Let LOGO Be a Game. In: 2011 Third International Conference on Games and Virtual Worlds for Serious Applications, Athens, 2011, pp. 24–31. https://doi.org/10.1109/VS-GAMES.2011.10
35. Parmar D, Isaac J, Babu SV, D'Souza N, Leonard AE, Jörg S, Gundersen K, Daily SB (2016) Programming moves: Design and evaluation of applying embodied interaction in virtual environments to enhance computational thinking in middle school students. In: 2016 IEEE Virtual Reality (VR). IEEE, pp 131–140
36. Paliokas I, Arapidis C, Mpimpitsos M (2011) PlayLOGO 3D: A 3D Interactive Video Game for Early Programming Education: Let LOGO Be a Game. In: 2011 Third International Conference on Games and Virtual Worlds for Serious Applications,Athens, 2011, pp 24–31 https://doi.org/doi:10.1109/VS-GAMES.2011.10
37. Piteira M, Haddad SR (2011) Innovate in your program computer class. In: Proceedings of the 2011 Workshop on Open Source and Design of Communication - OSDOC '11, ACM Press, New York. New York, USA, pp 49
38. Robertson GG, Card SK, Mackinlay JD (1993) Information visualization using 3d interactive animation. Commun ACM 36(4):57–71
39. Schwab K (2017) The fourth industrial revolution. Currency
40. Sarkar SP, Sarker B, Hossain SA (2016) Cross platform interactive programming learning environment for kids with edutainment and gamification. In: 19Th international conference on computer and information technology, ICCIT, IEEE, pp 218–222
41. Sajaniemi J, Kuittinen M (2003) Program animation based on the roles of variables. In: Proceedings of the 2003 ACM symposium on Software visualization, San Diego, California, USA, ACM, pp 7–ff
42. Schez-Sobrino S, García MÁ, Gómez C, Vallejo D, Lacave C, Glez-Morcillo C, Molina AI, Albusac JA, Redondo MÁ (2019) ANGELA: A novel approach of graphic notation based on the metaphor of road signs to facilitate the learning of programming. In: Proceedings of the 7th International Conference on Technological Ecosystems for Enhancing Multiculturality
43. Schez-Sobrino S, Gmez-Portes C, Vallejo D, Glez-Morcillo C, Redondo MÁ (2020) An intelligent tutoring system to facilitate the learning of programming through the usage of dynamic graphic visualizations. Appl Sci 10(4):1518
44. Teyseyre AR, Campo MR (2009) An overview of 3d software visualization. IEEE transactions on visualization and computer graphics 15(1):87–105
45. Teng CH, Chen JY, Chen ZH (2018) Impact of augmented reality on programming language learning: Efficiency and perception. J Educ Comput Res 56(2):254–271
46. Vahldick A, Mendes AJ, Marcelino MJ (2014) A review of games designed to improve introductory computer programming competencies. In: 2014 IEEE Frontiers in education conference (FIE) Proceedings. IEEE, pp 1–7
47. Vasilopoulos IV, van Schaik P (2018) Koios: design, development, and evaluation of an educational visual tool for greek novice programmers. J Educ Comput Res, 0(0)
48. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experiment Process. pp 31–39
49. Wald A, Wolfowitz J (1940) On a test whether two samples are from the same population. The Annals of Mathematical Statistics 11(2):147–162

50. Wing JM (2006) Computational thinking. Commun ACM 49(3):33–35
51. White R, Tian F, Smith P (2016). In: Code lab: a game that teaches high level programming languages. In: Proceedings of the 30th International BCS Human Computer Interaction Conference: Fusion!, Poole. BCS Learning & Development Ltd., United Kingdom, pp 1–8, https://doi.org/10.14236/ewic/HCI2016.76