

## PAPER

# BlockCode: A Web Application to Create Games that Support the Learning of Computer Programming Logic

Carlos R. Jaimez-González(✉), Javier Erazo-Palacios, Betzabet García-Mendoza

Universidad Autónoma Metropolitana, Ciudad de México, México

[cjaimez@cua.uam.mx](mailto:cjaimez@cua.uam.mx)

## ABSTRACT

This paper presents *BlockCode*, a web application developed to create board games that support the learning of computer programming logic. The goal of the games created with *BlockCode* is to move a bunny around a board using block-based programming in order to collect all the carrots that are placed on the board. In every game, the bunny has to deal with obstacles, represented by rocks and holes, which have to be avoided using the commands available. *BlockCode* was designed for teachers and students: teachers can create games, specifying the arrangement of the boards; and students play the games, specifying the sequence of commands with the aim of collecting all the carrots.

## KEYWORDS

educational technology, programming logic, board games, block-based programming, educational games

## 1 INTRODUCTION

Programming logic has been included in the curriculum of many schools at various levels. Although some may question its usefulness, learning programming enriches the student's perspective, even though the knowledge acquired may not have immediate practical application [1]. When observing student performance results, the way programming is taught also plays a crucial role. Teaching programming using a visual programming language versus a text-based programming language has an impact on how students comprehend the subject [2].

The work presented in this paper is a web application called *BlockCode*, which was designed and developed to create personalized board games, with the aim of supporting the learning of computer programming logic. The rest of the paper is organized as follows. Section 2 provides the justification for this work. An overview of the context and theoretical foundation is provided in Section 3, as well as different methods employed in similar systems and the outcomes observed in global research

Jaimez-González, C.R., Erazo-Palacios, J., García-Mendoza, B. (2023). *BlockCode: A Web Application to Create Games that Support the Learning of Computer Programming Logic*. *International Journal of Emerging Technologies in Learning (ijET)*, 18(15), pp. 240–257. <https://doi.org/10.3991/ijet.v18i15.40901>

Article submitted 2023-04-28. Resubmitted 2023-06-02. Final acceptance 2023-06-02. Final version published as submitted by the authors.

© 2023 by the authors of this article. Published under CC-BY.

investigations. Section 4 carries out an analysis of some systems similar to the web application developed. The development of *BlockCode* with the modules that compose it, including the technologies and tools used for its construction, is described in Section 5. Section 6 explains the functionality of *BlockCode*, including the objective of the existing board objects, the command blocks available, the execution panel and the animation controls. Finally, conclusions and future work are presented in Section 7.

## 2 JUSTIFICATION

It is indisputable that technology has a significant impact on our lives and education. According to a national survey on availability and use of information technologies in households [3] in 2018, more than 50 million Mexicans had internet access, which constituted over 50% of the population aged six or older. Among the most common activities carried out by these individuals was searching and acquiring information from the internet, and more than 50 million Mexicans also used a computer. Therefore, it is crucial to learn how to utilize these technological tools from an early age and take advantage of their capabilities.

Learning programming can be daunting, particularly if the teaching method is not effective. It is not uncommon for university students taking programming courses to struggle and ultimately withdraw from the course, perceiving the subject as too challenging [4]. Nevertheless, learning programming offers numerous advantages. An analysis of 65 studies on the cognitive effects of programming revealed that students with programming experience scored better in tests of cognitive skills than those without such experience [5].

Incorporating computer tools into classroom instruction produces diverse effects on students. Compared to traditional methods, computer tools encourage students to participate, increasing engagement and making the learning process more appealing and dynamic. Additionally, they offer immediate feedback, enabling students to track their progress in real time [6]. The use of visual programming languages, such as block-based programming, has previously been highlighted for its benefits over text-based languages in teaching. The simplicity of concepts and reliance on visual recognition in block-based programming make it easier to learn than other text-based languages, which can be more challenging due to additional cognitive demands [7].

Based on the information provided earlier, the web application developed aims to provide a platform for supporting the learning of programming logic using board games, which are solved using command blocks. *BlockCode* is suitable for young students and adults who are interested in learning programming.

## 3 BACKGROUND

In earlier sections, it was explained why there is a need to develop a system that employs a block-based programming language for teaching programming. The use of computers in education is now common across all levels of schooling. Its integration into the education system provides teachers with new teaching methods and enables students to learn in ways that were not available in the past.

However, the mere use of a computer in learning does not guarantee effective and meaningful learning. Often, students of all levels face frustration and

discouragement due to the complex nature of the learning process, resulting in poor learning outcomes and sometimes leading to dropping out. This is particularly evident in programming learning. In a preliminary study [8] of university students in Scotland, where they were asked to name a difficult subject and explain their strategies for tackling it, it was revealed that half of the students relied solely on a search engine to solve their problems. Moreover, many of them failed to grasp the connection between the difficult topic and other related topics. This indicates that their problem-solving skills were inadequate, as they were focused only on finding a quick solution. Additionally, they lacked perspective due to their inability to link concepts across different topics. To prevent such learning difficulties in programming, one possible solution could be to introduce it to students at an early age, using a variety of strategies.

Research has been conducted over time to explore the influence and consequences of teaching programming to children and youth. A study [9] was conducted on individuals between eight and eighteen years of age who were members of a Computer Clubhouse, where they used the block-based visual programming language Scratch. The study reports that Scratch became popular among young people as they learned programming basics in an entertaining manner by creating games and videos, with mentor support provided on an as-needed basis. The study also analyzed more than 400 projects to determine the concepts that the students were learning, with cycles and user interactions emerging as noteworthy.

The aforementioned study demonstrates that implementing a new and distinctive approach to teaching programming can have positive and enjoyable outcomes. Nonetheless, it should also be recognized that not all students possess the same problem-solving skills. While some individuals may find conventional methods easier to comprehend, others may require unique strategies for effective learning. It becomes crucial to understand which tool should be used to teach which type of student, as well as how the approach is linked to the student's emotions. Therefore, it is imperative to grasp the distinctions and consequences between employing block-based and text-based programming languages as introductory languages to programming, as these differences can be quite significant.

Another study [10] focused on 50 fifth-grade students from two primary schools who had no prior programming experience and had varying levels of problem-solving skills. The research revealed some interesting findings on the understanding of programming concepts using the Python programming language and Scratch. Students who had better problem-solving skills performed better on Python tests compared with those with intermediate ability. However, when it came to Scratch, both groups of students performed similarly. The study also compared intermediate-level students with low-level students and found that intermediate-level students performed better in both Python and Scratch. This suggests that low-level students had programming difficulties with both approaches. Furthermore, over half of the students preferred using Scratch, and their overall opinion on programming even improved.

The way in which programming is taught cannot only improve learning but also address common learning and problem-solving challenges. Hybrid approaches that combine digital technologies with traditional materials like paper and colors can be used. Research conducted in Croatia [11] proposed an approach to teach programming concepts to primary school children with concentration difficulties and a lack of problem-solving strategies. The approach used a game that involved navigating a robot through a maze to find a prize. Colored cards were used to simulate

the directions in which the robot could move. NetLogo, an integrated development environment for modeling, was then used to simulate an environment similar to the game. The study concluded that the children enjoyed the activity, and only one solution created in NetLogo was incorrect, indicating the success of this approach. This is a teaching method that incorporates games, specifically computer games, as a crucial element. It has been evaluated comparatively with traditional teaching methods and has been found to improve students' motivation, although it does not necessarily result in better learning outcomes [12]. Motivation is considered an essential element in learning as it gives students the drive to learn and expand their knowledge.

A study [13] was conducted on the IBM Robocode game, a programming game that simulates battles between robots and is played online. The study found that over 50% of the participants, including youths and adults, reported an improvement in their programming skills while playing the game. One of the reasons for this improvement was because they enjoyed playing it. Another study [14] focused on an educational game called PiktoMir, designed to introduce basic programming concepts to preschool children between five and seven years old. Although five out of seven children under six years old did not pass the test, the majority of children did. The study concludes that while some children did not enjoy the tasks in PiktoMir, half of the children wanted to continue playing because they found it fun.

Several other studies and initiatives have explored similar approaches. One initiative was a workshop [15] aimed at increasing the number of women in computer science at the middle and high school level. The workshop introduced programming concepts for robotic applications using a presentation with Python. The initial evaluation of the workshop showed that it motivated students to learn programming concepts. Another study [16] was conducted using sCool, a game that teaches children computational thinking and coding in Python. In the experiment, students learned a concept and had to apply it to a practical task. Two groups of students participated in the activity, and the study showed that students were interested in learning to code but had difficulty transferring the learned content to similar areas. Another study [17] aimed to explore the potential benefits of introducing concurrent programming concepts early in the learning process. The study indicated that uninitiated programming students at the age of ten could understand basic concurrency concepts, while students at the age of eleven with programming familiarity could understand advanced topics.

There are numerous approaches and factors that can be used for learning programming, but one of great importance is the visual factor. Although it is common to find books with excessive text and little visual material, it is crucial to recognize that there are alternative ways to learn, and more importantly, that they are effective. It is also important to realize that the problem of learning programming does not solely lie in the type of material being used, but also in the students' problem-solving abilities.

## 4 RELATED WORK

This section examines five distinct tools that aim to teach programming concepts to children and young adults. These tools are accessible on mobile devices, desktop computers, and laptops, and use either block-based or text-based programming.

**Code Combat** [18] is an online game that can be accessed through a web browser and is designed for students over the age of nine to learn programming concepts in Python and JavaScript. When starting the game, students are asked to create a character and choose the programming language to use. They can change these settings

later, as needed. The game features different levels organized by content type, and each level has corresponding missions. Clicking on a mission reveals its name, objective, and the programming concepts to be learned. During gameplay, students can assign abilities (programming commands) to their character to complete the mission. Code Combat has had a significant impact on schools, as demonstrated by the experience [19] of a technology teacher, who created a coding class with 180 seventh and eighth-grade students. By using Code Combat, he covers four computer science courses where students learn concepts such as decision statements, functions, and boolean logic. Since they started using this tool, the students have written 54,777 computer programs and 839,326 lines of code, and completed 613 network and game projects.

**Coding Adventure** [20] is a program that teaches programming using the CoffeeScript language, aimed at children aged eight and up. It is part of the Code Monkey teaching platform. After selecting an activity from the start screen, the game begins, and the student can use the editor on the right side of the interface to write commands for the character to perform. The graphic portion of the interface on the left reflects the executed code, and the results are displayed on screen after the objective has been achieved. A map is available to select the level to play, which indicates the number of levels and the objectives to be achieved. There is a case study [21] about the impact of Coding Adventure, which was carried out by an academic integration specialist who was searching for a way to teach elementary school students to code and found the solution on this platform. Her students enjoyed solving the codes and working in pairs, while developing problem-solving skills, critical thinking, and more, all through what they perceived as a simple game.

**Lightbot** [22] is an app that is a puzzle game aimed at children aged four and above. The game uses a block-based programming language and has a menu that focuses on teaching programming concepts. When selecting a level, the student can choose a mission to complete. The game screen is divided into two parts: on the right side, there is a box where blocks are placed as instructions, such as advancing, rotating, and turning on lights. On the left side, the graphic part shows the executed set of blocks. After executing a set of blocks and achieving the goal, the student can progress to the next mission. Lightbot has been highly recommended by both teachers and students on the Common Sense website and is widely used in schools. An elementary school teacher shared her experience of introducing Lightbot to her fifth-grade students, who were able to grasp basic programming concepts such as algorithms and loops. The same students then helped guide their fourth-grade peers during an international event about teaching Computer Science to students worldwide [23].

**Code Karts** [24] is a puzzle game aimed at children aged four and above, where the objective is to guide a small car to the finish line using block instructions. The game interface consists of a box on the left side containing the set of instruction blocks that can be used to achieve the goal, and a box on the upper side where the blocks are placed for the instruction to be executed. The interface also displays the start line, path, and finish line for the car. If the car crashes due to an incorrect instruction block, it loses a life, which decreases the final performance rating. Once the level objective is reached, the game displays a performance result screen.

**Spritebox** [25] is an adventure game suitable for all ages, where players overcome obstacles by coding using either block-based programming or by writing commands using the Swift or Java programming languages. The game aims to teach programming concepts such as sequencing, parameters, and cycles. Players can choose to play the game in either English or Portuguese and can also customize the character's features and the type of programming language used to solve obstacles.



The game interface shows a student solving obstacles using blocks, with the right side displaying the available blocks (up, left, right, down) and the left side showing the game with empty boxes where the student must place the blocks.

A comparative analysis of the tools examined in this section is carried out in [26]. Additionally, there is a broader examination and comparison of educational games intended for teaching computer programming and computational logic in [27]. There is also a review of emerging technologies in the field of education presented in [28].

## 5 DEVELOPMENT OF THE WEB APPLICATION

This section presents the development of *BlockCode*. The modules that compose it are explained, and the technologies and tools used to develop it are described.

### 5.1 Modules

This section provides an explanation of the modules that compose *BlockCode*. These modules are in charge of validating users, displaying maps and levels, executing actions to play games that correspond to specific levels, validating solutions, generating animations, presenting results, and creating maps and levels, among others.

**Access and validation module.** This module allows teachers and students to access the web application through a login page. A form is used to enter the access data, which is then validated through a server object; the user is redirected to the corresponding web page. The data entered is checked in the database through a validation method to determine whether the user exists and is valid.

**Student menu module.** This module corresponds to the main-menu web page for students, where they can start playing games, unlock maps and levels, and even watch a video tutorial. The visualization of maps, levels, and their corresponding states (locked and unlocked) is dynamic and is achieved by getting the corresponding data using the student's ID, which is obtained through an attribute of the current session. Figure 1 shows the interface of the student main menu: the black rows represent the maps; each map has a difficulty (easy, medium, and hard) and a number of levels; the first map in Figure 1 has three levels (green and blue buttons); the first map allows the student to play the first level (second and third levels are locked).



Fig. 1. Interface of the student main menu in *BlockCode*

**Levels module.** This module has an interface that allows students to play a specific game level. Figure 2 shows the interface of a level with its corresponding panels: the *command panel* with the commands available (walk, jump, turn right, turn left, loop and function); the *execution panel*, where the commands to be executed are placed; the *function panel*, where the commands stored will be replicated when using a function command; the *main panel*, which contains the *board of the game* with various objects (carrots, obstacles and the main character of the game), the *textual commands box* of the commands used, some *animation controls* to manipulate the animation on the board, one button to reset the board, and another to exit the level.



Fig. 2. Interface of a game level with the different panels in *BlockCode*

**Solution verification module.** This module verifies a given solution. Once a student determines his sequence of blocks and executes it, it is time to verify if that is the solution to the level. This is accomplished by creating a string that represents the commands used in the execution panel, and another that represents the commands used in the function panel. These representations are used later to determine whether or not it is the solution. The current time is also obtained to determine how long it took the student to reach the solution.

**Game animation module.** This module generates the animation of the game. *Sprites* were used in order to implement the graphics of the level boards, which are two-dimensional images that are integrated into larger scenarios. They can represent a character or an object; in the case of *BlockCode*, they represent the bunny, the carrots, the grass, and the obstacles (rocks and holes). In order to achieve the different animations of the bunny, it was necessary to use a *Sprite Sheet*, which is an image that contains many *Sprites* of a certain size. In this case the *Sprites* represent the bunny in four different orientations, simulating the action of moving forward (walking). This sheet, or document, is placed with a certain visible size inside an HTML *div* element, and then, using JavaScript and GreenSock, it is manipulated at a certain speed and number of frames (*Sprites*) in order to generate the animation. *BlockCode* uses four *Sprite Sheets* to achieve the animations of the bunny in its four actions (walk, jump, turn right and turn left). Figure 3 shows the *Sprite Sheet* used for the action of walking.



Fig. 3. Sprite sheet used for the action of walking in *BlockCode*

**Student results module.** This module allows students to review their performance obtained by completing levels. The interface of this module provides students the following information through a results table: the level played, the size of the board game, the total resources on the board, whether the level is locked or not, whether the level has been passed or not, the number of command blocks used in the solution, and the time employed to complete the level. Figure 4 shows the interface of this module.

**BLOCKCODE** INICIO RESULTADOS ? alumno1

Mapa Uno				Niveles superados: 3/3			
NIVEL	ALTO	ANCHO	TOTAL RECURSOS	BLOQUEADO	SUPERADO	BLOQUES USADOS	TIEMPO
1	5	3	2	NO	SI	9	00:00:10
2	3	5	10	NO	SI	20	00:00:16
3	3	3	4	NO	SI	20	00:00:40

Fig. 4. Interface of the student results in *BlockCode*

**Teacher menu module.** This module corresponds to the main menu web page for teachers, where they can see all the maps and their corresponding levels available, as well as the option to view the levels individually. The visualization of maps and levels is dynamic. Figure 5 shows the interface of the teacher main menu: the black rows represent the maps; each map has a difficulty and a number of levels; the first map in Figure 5 has three levels (green and blue buttons); the teacher can view all the levels.

**BLOCKCODE** VER MAPAS CREAR MAPA Profesor1

Mapa Uno	Facil	Total Niveles: 3
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="background-color: #4a4a9a; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 1</div> <div style="background-color: #00b050; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 2</div> <div style="background-color: #4a4a9a; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 3</div> </div> <div style="display: flex; justify-content: center; margin-top: 5px;"> <div style="background-color: #ff0066; color: white; padding: 5px; border: 1px solid #ccc;">VER</div> </div>		
Mapa Dos	Medio	Total Niveles: 6
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="background-color: #4a4a9a; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 1</div> <div style="background-color: #4a4a9a; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 2</div> <div style="background-color: #4a4a9a; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 3</div> <div style="background-color: #4a4a9a; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 4</div> <div style="background-color: #4a4a9a; color: white; padding: 5px; border: 1px solid #ccc;">NIVEL 5</div> </div>		

Fig. 5. Interface of the teacher main menu in *BlockCode*



**Level visualization module.** This module has an interface that allows teachers to visualize the board of the level as well as some information about it, such as the map number to which the level belongs, the level number, the width and height of the board, the total number of squares in the board, and the total number of resources (carrots, rocks, and holes) on the board. This interface is shown in Figure 6 and is only for visualization purposes; it is not possible to play the game from this interface.



Fig. 6. Interface for teachers to visualize a level in *BlockCode*

**Map creation module.** This module allows teachers to create a new map and new levels for the students to play. Teachers need to provide the name and difficulty category of the map. Once the map is created, the teacher can add any number of levels and edit them as many times as necessary. Figure 7 shows the interface for teachers to create new maps and levels.



Fig. 7. Interface for teachers to create maps and levels in *BlockCode*

**Level creation and editing module.** This module allows teachers to start creating and editing levels by generating a board based on height and width, dragging and arranging the available objects to the board and saving it. The interface to create and edit levels is shown in Figure 8, where teachers can drag the objects from the left panel (bunny, grass, carrots, rocks, and holes) and drop them on the board.

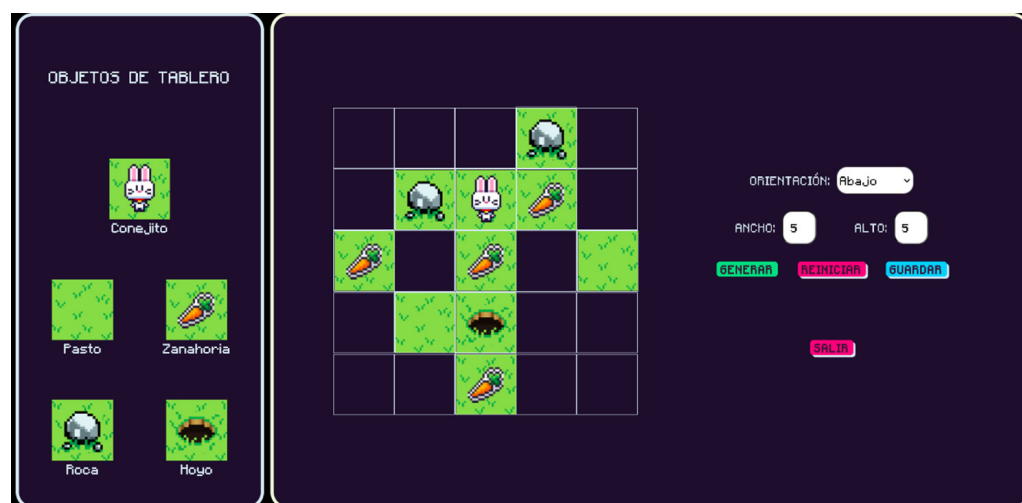


Fig. 8. Interface for teachers to create and edit a level in *BlockCode*

## 5.2 Technologies and tools used

There were several technologies and tools used for the development of *BlockCode*, which are described in the following paragraphs.

**Java Server Pages (JSP).** This is a Java technology that allows development of dynamic web pages, combining static HTML with HTML dynamically generated. JSP files were used for the creation of dynamic web pages, where information is provided to the web application, extracted from the database, and processed. Specifically, JSP was used for the modules that create the games and the modules for playing the games.

**JavaScript.** This is an interpreted programming language, designed to make interactive and dynamic web pages. This language is based on objects because it has an implementation of the document object model (DOM), which is a model that translates the structure of an HTML document to a tree of objects when the document is interpreted by a web browser. JavaScript was used for dragging and dropping elements on the boards and for the character movements, among other functionalities.

**GreenSock.** This is a set of tools based on JavaScript, which is used to create and manipulate animations in an easy way. GreenSock was specifically used for creating the animations of the character that is used on the boards. It was also used to animate the buttons and actions performed when a game is played.

**Cascading Style Sheets (CSS).** This is a language for styles used to change the presentation of a web page, such as the background color or font size. CSS were used for the presentation of the entire web application to create games, for all the layout of the menus and boards, and for the structure of the games.

**NetBeans.** This is an integrated development environment (IDE), which allows one to write, compile, debug and execute programs. NetBeans allowed the concentration and creation of all the files of the project, such as Java classes, JSP files, CSS files, and JavaScript files, among other files.

**Apache Tomcat.** This is a web server and servlet container, which allows one to compile JSP files that are converted into servlets. Apache Tomcat was used as a web server and servlet container, and for processing all the JSP files of the system.

**MySQL.** This is a relational database management system (RDBMS) based on the structured query language (SQL), multi-threaded, multi-user, and multi-platform, which is widely used for storing data in web applications for different platforms. MySQL was used for the creation of the relational database of the web application.

**Balsamiq Mockups.** This is an application for creating prototypes for web graphical interfaces, with the aim of showing them to the students and teachers in early stages of the development process. Balsamiq Mockups was used to create very quickly interface prototypes, which were used to validate and test the interface.

## 6 OPERATION OF THE WEB APPLICATION

This section presents the operation of *BlockCode*. Figure 9 shows an example of the game-level page with its different sections: command panel, board game, textual commands box, help, function panel, slide arrows, execution panel and animation controls. The following paragraphs describe each of these sections. An example of the execution of a game is presented at the end of this section.

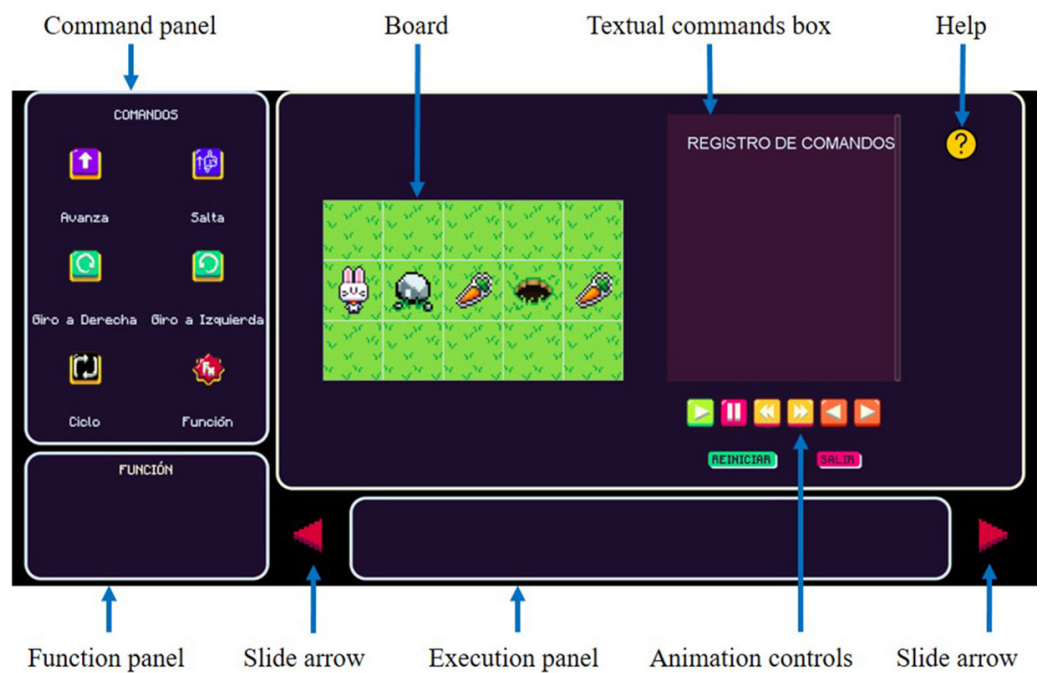


Fig. 9. Interface of the game-level page with its different sections in *BlockCode*



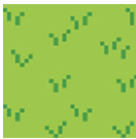


**Board.** This is a visual representation with various objects in it; its item distribution is always different, adding a factor of difficulty and variety to the game. It can have a regular shape, where each square is occupied by a board object, or it can have an irregular shape, where there are completely empty squares whose sides act as a wall. Both shapes are playable, and the teacher decides on their size, this being defined by the number of squares in width and height. Figure 10 illustrates two boards: a board with regular shape on the left and a board with irregular shape on the right.



Fig. 10. Boards with regular and irregular shapes in BlockCode

**Board objects.** These are visual representations of the game character (bunny), the carrots, the grass, and the obstacles (rocks and holes). Table 1 shows the different board objects available and provides a description of them.

Table 1. Board objects available in BlockCode

Objects	Description
	<b>Bunny.</b> It is the main character of the game. Its goal is to get all the carrots on the board. In order to do this, it moves around the board using the command blocks available. It moves on the board depending on its orientation (up, down, left, right).
	<b>Carrot.</b> It is the main objective of the bunny and also the main objective of the game. All carrots must be collected by the bunny in order to complete a level.
	<b>Grass.</b> It is a patch of grass onto which the bunny can move.
	<b>Hole.</b> It is an obstacle. The bunny must jump over the holes, landing on the first square attached to the bunny's orientation, as long as that square is grass or a carrot.
	<b>Rock.</b> It is an obstacle. The bunny must walk around the rocks because it cannot advance or jump over them.

**Execution panel.** This is a container where command blocks can be dragged and dropped or clicked to appear inside the execution panel. In order to remove them, simply drag and drop them out of the panel or click on them. The direction of the commands inside the execution panel is from left to right, so the last command block put inside will scroll to the right. In case the execution panel fills up, the command blocks will automatically slide to the left, so if it is needed to control their visibility, the user should use the red slider arrows, which allow to scroll through the visibility of the sequence of command blocks contained in the execution panel. Figure 11 illustrates an execution panel with some command blocks inside.










Fig. 11. Execution panel with command blocks in *BlockCode*

**Function panel.** This can contain command blocks, except loop and function commands. The user must drag and drop commands inside the panel to add them, and drag and drop those no longer required out of the function panel. The function panel can contain a maximum of 10 commands. When the function command is used in the execution panel, all the commands contained in the function panel will be executed.

**Command panel.** This has six available command blocks. They can be dragged directly into the function panel or into the execution panel to place them, or clicked to automatically place them in the execution panel. Each command has an effect on the bunny character on the board. Table 2 shows the different commands available.

Table 2. Command blocks available in *BlockCode*

Commands	Description
	<b>Walk.</b> The bunny walks one square from where its current orientation is (up, down, left, right). The bunny can walk only if the square contains a carrot or grass.
	<b>Jump.</b> The bunny jumps on a square and lands on the next one, wherever its orientation is. It can be used only if there is a hole in the square on which the bunny must jump.
	<b>Turn right.</b> The bunny rotates on its axis to the right, thus changing its orientation.
	<b>Turn left.</b> The bunny rotates on its axis to the left, thus changing its orientation.
	<b>Loop.</b> This allows formation of a sequence of command blocks by allowing the user to drag and drop a command block onto itself.
	When the loop is placed in the execution panel, a white editable box appears, which contains an integer that determines how many times the sequence is repeated.
	<b>Function.</b> This allows the execution of the commands that are inside the function panel.

**Textual commands box.** This has the functionality of representing in a textual way the actions of the bunny within the board game. All the actions carried out by the bunny while it is moving on the board are reflected in this box. Figure 12 shows a fragment of a *commands* box with the following actions carried out by the bunny:

bunny moves to the left (it got a carrot); error, the bunny tried to jump to the wall; bunny turns right; bunny moves up; bunny turns left; bunny moves to the left.

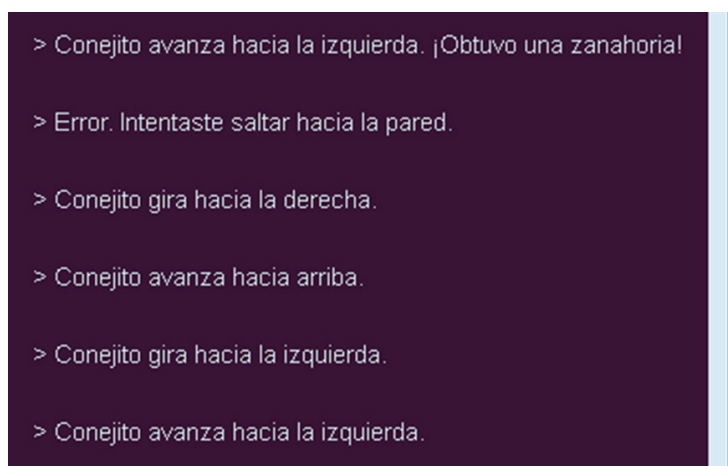








Fig. 12. Textual command box in *BlockCode*

**Animation controls.** These allow the student to control the execution of the sequence of command blocks, as well as the animation of the board and its objects. Table 3 shows the different controls available and provides a description of them.

Table 3. Animation controls available in *BlockCode*

Controls	Description
	<b>Execute.</b> It allows the execution of the sequence of command blocks contained in the execution panel to determine if it is the solution or not, as well as the corresponding animation.
	<b>Pause.</b> It allows the student to pause the animation once the execution has finished.
	<b>Rewind.</b> It allows the student to rewind the animation from where it is to the beginning, after the execution has finished.
	<b>Forward.</b> It allows the student to play the animation from the point where it is until the end, once the execution has finished.
	<b>Go back one step.</b> It allows the student to return the animation from the point where it is to one step back, once the execution has finished.
	<b>Go forward one step.</b> It allows the student to advance the animation from the point where it is to one step ahead, after the execution has finished.

**Example of a game with a solution.** Figure 13 shows an example of a game with a sequence of command blocks, which provide the solution for the board presented. The sequence of commands in the execution panel is the following: walk,



jump, turn left, walk, jump, turn left, walk, jump, turn left, walk, jump, turn left, walk. It should be noticed that a board can have more than one solution.



Fig. 13. Interface of a game with a solution in *BlockCode*

## 7 CONCLUSIONS AND FUTURE WORK

This paper presented *BlockCode*, a web application that allows the creation of personalized board games with the objective to support the learning of computer programming logic. The web application was designed for teachers and students interested in learning programming. Teachers are capable of creating games with their own board arrangements, placing in each square of the board the different available objects: bunny, grass, carrot, rock, or hole. Students play the games created by teachers, generating their own sequence of command blocks to give a solution to the game.

We carried out a comparative analysis of five existing games with similar purpose, which use either block-based or text-based programming languages. The most relevant features were highlighted. It should be noted that none of the tools analyzed allows the creation of personalized board games. *BlockCode* is a powerful web application that allows the generation of any number of games with different board sizes and with different board arrangements, using all the available board objects. Furthermore, the animation controls allow students to execute the animation step by step to understand the programming logic needed to solve a specific problem.

Further work is needed to evaluate *BlockCode* with teachers and students in four aspects: functionality, usability, design and didactic features. It will also be important to get feedback about the perception of students when they think of a solution to a given problem (board game). After the evaluation, it is also planned to put it on a web server so that teachers and students can use it permanently.

Finally, it should be noted that there have been other initiatives to support the teaching-learning process of computer programming at different levels, such as a web application that shows the execution of a program through a graphic visualization [29], a web application to create flowcharts [30], and a set of learning objects to support structured programming undergraduate courses [31].

## 8 REFERENCES

- [1] Reichert, R., Nievergelt, J., Hartmann, W. (2001). Programming in Schools – Why, and How? In C. Pellegrini, A. Jacquesson (Eds.): Enseigner l'informatique, pp. 143–152.
- [2] Weintrop, D., Wilensky, U. (2015). Using Commutative Assessments to Compare Conceptual Understanding in Blocks-Based and Text-Based Programs. Proceedings of the 11th International Conference on International Computing Education Research (ICER'15), pp. 101–110. <https://doi.org/10.1145/2787622.2787721>
- [3] INEGI. (2019). En México hay 74.3 millones de usuarios de Internet y 18.3 millones de hogares con conexión a este servicio: ENDUTIH 2018. Press release 179/19.
- [4] Yadin, A. (2011). Reducing the Dropout Rate in an Introductory Programming Course. ACM Inroads, 2 (4), pp. 71–76. <https://doi.org/10.1145/2038876.2038894>
- [5] Liao, Y.K.C., Bright, G. W. (1991). Effects of Computer Programming on Cognitive Outcomes: A Meta-Analysis. Journal of Educational Computing Research, 7 (3), pp. 251–268. <https://doi.org/10.2190/E53G-HH8K-AJRR-K69M>
- [6] Caccuri, V. (2013). Educación con TICs. 1st ed. Buenos Aires: REDusers. <https://issuu.com/redusers/docs/educacion-con-tics>
- [7] Bau, D., Gray, J., Kelleher, C., Sheldon, J., Turbak, F. (2017). Learnable Programming: Blocks and Beyond. Communications of the ACM, 6 (6). <https://doi.org/10.1145/3015455>
- [8] Bain, G., Barnes, I. (2014). Why Is Programming So Hard to Learn? Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference (ITICSE '14), Uppsala, Sweden. <https://doi.org/10.1145/2591708.2602675>
- [9] Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., Rusk, N. (2008). Programming by choice: urban youth learning programming with Scratch. Proceedings of the 39th ACM Technical Symposium on Computer Science Education (SIGCSE '08), 40 (1), pp. 367–371. <https://doi.org/10.1145/1352322.1352260>
- [10] Mladenović, M., Krpan, D., Mladenović, S. (2017). Learning programming from Scratch. Proceedings of the International Conference on New Horizons in Education 2017, Berlin, Germany.
- [11] Zaharija, G., Mladenović, S., Boljat, I. (2013). Introducing basic Programming Concepts to Elementary School Children. Procedia – Social and Behavioral Sciences, 106, pp. 1575–1584. <https://doi.org/10.1016/j.sbspro.2013.12.178>
- [12] Egenfeldt-Nielsen, S. (2005). Beyond Edutainment Exploring the Educational Potential of Computer Games. PhD Thesis, IT-University of Copenhagen, Copenhagen, Dinamarca.
- [13] Long, J. (2007). Just for Fun: Using Programming Games in Software Programming Training and Education – A Field Study of IBM Robocode Community. Journal of Information Technology Education, 6, pp. 279–290. <https://doi.org/10.28945/216>
- [14] Rogozhkina, I., Kushnirenko, A. (2011). PictoMir: teaching programming concepts to pre-schoolers with a new tutorial environment. Procedia – Social and Behavioral Sciences, 28, pp. 601–605. <https://doi.org/10.1016/j.sbspro.2011.11.114>
- [15] Keller, L., John, I. (2020). Motivating Female Students for Computer Science by Means of Robot Workshops. International Journal of Engineering Pedagogy (IJEP), 10(1), pp. 94–108. <https://doi.org/10.3991/ijep.v10i1.11661>
- [16] Steinmaurer, A., Pirker, J., Gütl, C. (2019). sCool – Game-Based Learning in Computer Science Class: A Case Study in Secondary Education. International Journal of Engineering Pedagogy (IJEP), 9(2), pp. 35–50. <https://doi.org/10.3991/ijep.v9i2.9942>
- [17] Fatourou, E., Zygouris, N. C., Loukopoulos, T., Stamoulis, G. I. (2018). Teaching Concurrent Programming Concepts Using Scratch in Primary School: Methodology and Evaluation. International Journal of Engineering Pedagogy (IJEP), 8(4), pp. 89–105. <https://online-journals.org/index.php/i-jep/article/view/8216>

- [18] Code Combat. The Most Interesting Game to Learn Programming. Available: <https://codecombat.com>
- [19] Bobby Duke Case Study. Bobby Duke Middle School. Available: [https://codecombat.com/images/pages/impact/pdf/CodeCombat\\_CaseStudy\\_BobbyDukeMS.pdf](https://codecombat.com/images/pages/impact/pdf/CodeCombat_CaseStudy_BobbyDukeMS.pdf)
- [20] Coding Adventure. Teach Text-Based Coding by Helping a Monkey to Catch Bananas. Available: <https://www.codemonkey.com/coding-adventure/>
- [21] Fudale, K. (2016). Case Studies for CodeMonkey. Ohio, USA: Edsurge. Available: <https://www.codemonkey.com/case-studies/>
- [22] Lightbot. Puzzle Game Based on Coding. Available: <https://lightbot.com>
- [23] Jana, C. (2016). Teacher Review for Lightbot: Programming Puzzles. Commonsense. Available: <https://www.commonsense.org/education/reviews/lightbot-programming-puzzles>
- [24] Code Karts. A Fun App to Develop Observation, Concentration and Logic. Available: <https://montessori.edokiacademy.com/en/our-games/discovery/car-game>
- [25] Spritebox. A Full-Blown Adventure Game that Gets you Coding. Available: <https://spritebox.com>
- [26] Erazo-Palacios, J., Jaimez-González, C. R., García-Mendoza, B. (2022). Towards a Web Generator of Programming Games for Primary School Children. *International Journal of Engineering Pedagogy (iJEP)*, 12 (4), pp. 98–114. <https://doi.org/10.3991/ijep.v12i4.17335>
- [27] Pereira, J., Frango, I. (2020). A Systematic Review on Open Educational Games for Programming Learning and Teaching. *International Journal of Emerging Technologies in Learning*, 15 (9), pp. 156–172. <https://doi.org/10.3991/ijet.v15i09.12437>
- [28] Sosa Neira, E., Salinas, J., Benito, B. (2017). Emerging Technologies (ETs) in Education: A Systematic Review of the Literature Published between 2006 and 2016. *International Journal of Emerging Technologies in Learning*, 12 (5), pp. 128–149. <https://doi.org/10.3991/ijet.v12i05.6939>
- [29] Jaimez-González, C. R., Castillo-Cortes, M. (2020). Web Application to Support the Learning of Programming Through the Graphic Visualization of Programs. *International Journal of Emerging Technologies in Learning*, 15 (6), pp. 33–49. <https://doi.org/10.3991/ijet.v15i06.12157>
- [30] Vazquez-Peñaloza, F., Jaimez-González, C. R. (2019). Towards a Web Application to Create Flowcharts for Supporting the Teaching-Learning Process of Structured Programming Courses. *American Journal of Educational Research*, 7 (12), pp. 976–982. <http://www.sciepub.com/education/abstract/11332>
- [31] Luna-Ramírez, W. A., Jaimez-González, C. R. (2014). Supporting Structured Programming Courses Through a Set of Learning Objects. *Proceedings of the IEEE International Conference on Information Society (i-Society 2014)*, pp. 124–128, London, UK. <https://doi.org/10.1109/i-Society.2014.7009024>

## 9 AUTHORS

**Carlos R. Jaimez-González** is a Professor at the Information Technology Department at the Universidad Autónoma Metropolitana Campus Cuajimalpa, in Mexico City. He received his PhD degree in Computer Science from the University of Essex, United Kingdom in 2011. His research interests include technologies for supporting education, interoperability in distributed systems, XML and related technologies, and the development of web and e-commerce applications. He has a distinction as a national researcher from the Mexican Government (email: [cjaimez@cua.uam.mx](mailto:cjaimez@cua.uam.mx)).

**Javier Erazo-Palacios** is a mobile and web application developer. He is studying for a BSc degree in Information Technologies and Systems from the Universidad Autónoma Metropolitana Campus Cuajimalpa. His research interests include technologies for supporting education, mobile and web application development (email: [2163030922@cua.uam.mx](mailto:2163030922@cua.uam.mx)).

**Betzabet García-Mendoza** is an Associate Professor at the Information Technology Department at the Universidad Autónoma Metropolitana Campus Cuajimalpa, in Mexico City. She received her MSc degree in Design, Information and Communication from the Universidad Autónoma Metropolitana. Her research interests include technologies for supporting education and web application development (email: [bgmendoza@cua.uam.mx](mailto:bgmendoza@cua.uam.mx)).