Chapter 8
Code-Venture: A Mobile Serious Game for Introductory Programming

Leckraj Nagowah and Diksha Cuniah

Abstract In the last decade, there have been tremendous improvements in the IT field, and the demand for skilled professionals has ever since grown rapidly. For a better economic development, it is thus of primary importance for schools and universities to uncover and train new talents who will help propel our society's upward trend in IT and meet the increasing demand. On the other hand, there is a misconception among youngsters that programming is complex and not designed for everyone. Using the fact that nowadays games are becoming increasingly popular especially among the younger generation, a mobile serious programming game, *Code-Venture*, is being proposed in this chapter. Other than being fun and entertaining, the aim of *Code-Venture* is to help the players understand the basics of coding and sharpen their skills in programming. *Code-Venture* is based on the fundamental programming principles as recommended in the ACM/IEEE Computer Science Curricula 2013. Moreover, through the implementation of the teacher's application, which stores scoring information about the players of the game, a constant monitoring, assessment, and evaluation of the player's performance can be performed. Pre-game and post-game surveys have been conducted to evaluate the mobile serious game *Code-Venture*. Most of the 35 respondents found the game useful and engaging with a considerable increase in the number of students who are willing to join a career in programming and another increase in the number of respondents who now found programming easy.

Keywords Mobile serious game · Game-based learning · Educational games · Introductory programming · Serious programming game

L. Nagowah (✉) · D. Cuniah
Faculty of Information, Communication and Digital Technologies, Department of Software and Information Systems, University of Mauritius, Réduit, Mauritius
e-mail: l.nagowah@uom.ac.mu; diksha.cuniah2@umail.uom.ac.mu

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_8

8.1 Introduction

Technological innovations hold a key place in most sectors worldwide. It has become a compulsory tool to most people in the workplace and life in general. Nowadays, almost every service, such as online shopping, reservations, or even learning, can be accessed online or through mobile applications. What lies behind those applications is programming. While coding can be fun and exciting, to some it comes out as boring and tough or even stressful when traditional learning methods are used [1, 2]. Additionally, a high rate of failure coupled with students' lack of interest in programing have been reported in a number of studies [3].

It is well known in the computer science education (CSE) community that students struggle in programming classes, which can lead to high dropout and failure rates [4]. In 2007, Bennedsen and Caspersen [5] estimated through a study that out of 2,000,000 IT students worldwide, about 650,000 failed every year. They then replicated their study in 2019 and observed that the failure rate was still high at 28% [6]. Fundamental programming and computational thinking skills are crucial in the field of computer science whereby one might find it challenging to enter the world of IT without these core skills [7].

On the other hand, Halbrook et al. found that 95% of homes with children of under 18 years of age own some form of video-game platform [8]. As a matter of fact, researchers are investigating game-based computer science learning [9, 10] where numerous games are being developed that mainly focus on computer programming and aim at helping students grasp the fundamental programming concepts. Mathrani et al. [11] confirmed the effectiveness of serious games as a means of teaching and learning. Students who participated in the survey proved that game-based learning made them more engaged in the use of programming principles through gaming steps, as they were able to visualize the programming constructs and thus get a better understanding of how it works. Boeker et al. [12] noted that a conventional approach was not very effective when learning programming. Thus, using serious games as a means of teaching made the normally boring classroom environment more fun and enhanced the player's ability to grasp difficult concepts. Ding et al. also confirmed through their study that game-based learning is more effective, easier to grasp, and more preferred by students than traditional learning methods [13].

The primary aim of this work is to develop a mobile serious game that aims at enhancing the programming skills of beginners, together with a mobile application for teachers who will assess, monitor, and evaluate the performance of players. The chapter also discusses the results obtained from a pilot study with 35 students aged between 16 and 21. The authors also plan to conduct a more structured experimentation in the future to assess the perceived improvements in the programming skills of the users of the mobile application. A structured approach was used to implement our mobile serious game and is described in the subsequent sections. The remainder of this chapter is organized as follows. In Sect. 8.2, a background on the related works has been given, and an analysis of the existing works has been provided in Sect. 8.3. Section 8.4 highlights the architecture and the main components of *Code-Venture*. The implementation and testing of the application are presented in Sect. 8.5. A discussion is presented in Sect. 8.6, and finally, Sect. 8.7 concludes the chapter.

8.2 Background Study

A game refers to an organized play with rules, goals, and challenges for the purpose of entertainment [14]. The term gamification was first coined in 2008, and in contrast to normal games, it is characterized by its serious purpose. Researchers agree that gamification usually focuses on game elements and mechanics in serious contexts. Gamification therefore includes the use of game elements in non-game contexts. Game elements include levels, points, badges, leader boards, avatars, quests, social graphs, or certificates [15, 16].

Gamification is closely linked to two related concepts, namely, game-based learning and serious games. Game-based learning is the achievement of defined learning outcomes through game content and play, as well as boosting learning by involving problem-solving areas and challenges that offer learners, who are also the players, with a sense of accomplishment [17]. As the name says, game-based learning aims at educating the players. It however relies on a fully fledged game, commonly known as a serious game. A serious game refers to playful interactive applications whose primary purpose other than being fun and entertaining includes education, training, analysis, visualization, simulation, health, and therapy [18]. Serious games and game-based learning therefore differ from gamification due to the fact that they are fully functional games. They all share the idea of employing pleasant gameful experiences for the benefit of a serious goal, such as education or behavior modification, rather than focusing on enjoyment. Gamification as a broader concept merely takes components of games and applies them to the real environment [15].

8.2.1 Related Works

To review existing serious games on programming, two searches have been carried out: one on Google Scholar to find some related works and another one on common app stores to look for existing commercial applications. This section presents some of the findings.

Jemmali and Yang [19] developed a serious game called “May’s Journey,” which targets middle school and high school students. This game was primarily designed to encourage girls to choose programming as a field of study. It was a 3D puzzle game where the player had to interact with the environment to solve mazes involving basics of programming. The puzzles focused on concepts and logics but still allowed the player to type programming instructions to bridge the gap between real programming and coding in a game. The game was tested with ten teenagers aged from 14 to 17 years for educational content, and the authors reported that the teenagers were engaged with the game.

Du et al. [20] made a study about the Hour of Code by Code.org, which is a 1-h introductory tutorial to programming, making use of visual programming. The game uses blocks to program a solution for different puzzles. After solving each maze, the player received a positive feedback and advanced to mazes that are more complex. One hundred and sixteen students from two universities participated in the study, and the game proved to have a positive impact on the students' attitude toward programming. However, the game also turned out not to have a significant effect on the player's actual coding skills.

Miljanovic and Bradbury [21] developed a serious game with a systematic approach to focus on programming comprehension rather than writing codes. Thus, novices without knowledge of programming could play this game and gain some basic understanding. The player had to accomplish several comprehension tasks in order to activate a Mech Suit system. While advancing in the game, the player was able to develop a concise understanding of variable values, data types, program statements, and control flow, all of which were repeatedly tested throughout the gameplay.

Law [22] investigated on how to enhance the iteration, selection, and building of command blocks in programming through the use of video games. The skills to be developed were problem-solving skills and computational thinking skills in order to get the required result. The author used the freely available "Program your robot" game that targeted these skills and allowed the player to visualize the abstract concepts in programming. Despite the fact that the pilot study was carried out with only 42 students, the findings and the student comments showed positive results and indicated that it would be worthwhile to expand the study to a broader cohort pursuing a wider range of computing programs.

Junaeti et al. [23] conducted research about teaching basic programming concepts using the genius learning strategy. "Array Adventure" was a serious adventure game designed to target the principles behind arrays. The first level involved one-dimensional arrays and the second level catered to multi-dimensional arrays. The game was set in a 2D environment whereby the player had to complete missions in an adventure style gameplay. The game was evaluated by 30 students and 2 experts and positive results were obtained.

Jordaan explored the likelihood of making use of board games to improve the learning experience of computer science students. The findings of this study demonstrated that students appreciated the dynamic learning atmosphere provided by board games and that they accepted them as a fun and enjoyable method of instruction [24].

Lotfi and Mohammed [25] presented a mobile serious game that taught object-oriented programming concepts for beginners. "OOP Serious Game" was set in a zoo environment where the player had to create animal classes and understand the methods that were behaviors, actions, and voices. The elements of class, object, and complex paradigms like inheritance and polymorphism were taught through gamification techniques in order to facilitate the learning of these concepts. The game used an in-game assessment mechanism to gauge the player's knowledge of four OOP concepts, namely, class, object, inherence, and polymorphism. However, the results were inconclusive as to the game's efficacy in teaching the programming concepts to the players.

Yallihep and Kutlu [26] analyzed and evaluated the effect of a mobile serious game called "LightBot" for learning programming. A 5-week study was carried out in a primary school in Turkey with 36 fifth-grade students. According to the research, the game positively influenced the students' achievements. Complex concepts like recursions and procedure were taught at an early stage to students in a gamified learning approach.

Zhao et al. [27] proposed a serious game that focuses on the structure of the C programming language. "Restaurant game" was a 2D game that incorporated programming concepts like data types, variables, and structures. The player was required to engage with various game objects, which were data types' representations in a restaurant, and, as a result, gain a deeper knowledge about application of these programming concepts. Ninety first-year students tested the serious game, and the results showed that the improvements in learning outcomes were statistically significant. More than half of the participants were of the opinion that the game may help them get better grades in the programming course, and more than 60% of the participants said it improved their understanding of programming topics.

Karram analyzed "Code Combat," which is a popular game that targets object-oriented programming concepts in a game-based format [28]. The game offers an engaging and fun environment whereby the player has to complete tasks and challenges to earn points and level up. Rewards such as badges and rankings also make the player more motivated to complete the levels and thus learn the concepts along the way. The game guides the player to type the appropriate code lines in order to assign tasks to the virtual characters and thus complete the puzzles. Code Combat makes difficult concepts like inheritance, nested loops, and recursion simpler through a gaming approach.

Toukiloglou and Xinogalos [29] developed "NanoDoc" that taught programming concepts through a first-person shooter game. The players had to acquire a key by solving programming puzzles to navigate through the different rooms. The programming environment featured a hybrid 2D/3D mode where the player created a program to control the avatar's movement in a 3D grid. The solution algorithm was constructed with a 2D block-based programming environment using colored blocks that could be connected and manipulated through drag and drop actions. The game proved to be effective in improving students' motivation, engagement, and learning outcomes, as well as in reducing their anxiety toward programming.

Akkaya and Akpinar [30] designed a game aimed at teaching the fundamental concepts of object-oriented programming and computational thinking skills to students. The game adopted a constructivist learning approach set in a fantasy environment with metaphorical machines to make the abstract concepts concrete. It included interactive tools such as class and method definer machines for students to program robots. A pedagogical agent provided instructions, support, and feedback, and the game had a visual and textual feedback mechanism to understand the code execution. The game aimed at teaching students the importance and applications of object-oriented programming and computational thinking and also eased their introduction to algorithmic thinking. The game was tested by 61 students with and without prior programming knowledge, and the results showed that the students improved their understanding of the fundamental concepts of OOP.

8.2.2 Commercial Apps

Popular games about coding were also searched for on common app stores with search terms like “serious game programming,” “programming game,” and “coding game” to search for existing serious games about programming. Some of the main findings are listed below.

- Hacked

In Hacked, the player impersonated a hacker who needed to solve some problems with codes and save the world [31]. It had a progressive difficulty and offered a wide variety of options ultimately guiding the player to develop his personalized game. The player was required to have some prior knowledge on programming before attempting the game. The features included were performance tracker, assistance in writing of codes, level system, problem-solving skills, reward system, and competition with other players.

- Coding Planets

Coding planets required the user to solve puzzles through commands issued to a robot [32]. All age groups are targeted allowing them to sharpen their programming skills and gain fair knowledge of coding. The players needed to use their logic to advance through the different levels while developing their problem-solving skills. The main features included were as follows: improvement of problem-solving skill, development of logical thinking skill, sequencing, looping, functions, use of command icons to issue instructions, beginner and advanced difficulty, and reward system.

- LightBot: Code Hour

This game introduced programming concepts for beginners [33]. It consisted of commanding a robot to light up tiles by giving it instructions. The skills targeted were basic concepts like sequencing, loops, and procedures. The game had good reviews whereby players affirmed that they were able to learn about programming concepts in a fun and interactive way. The features included were learning of programming practices like planning, programming, testing, and debugging; development of problem-solving skills; learning about control flow concepts like functions, sequencing, and loops; and programming through commands.

- SpriteBox: Code Hour

This was a puzzle-platformer and adventure game allowing the player to venture through different worlds and using code to complete the objectives [34]. It targeted all players regardless of their programming knowledge and consisted of 20 levels of challenging puzzles. The features included were icon-based programming, sequencing, parameters, debugging, loops and problem-solving, exploration while learning, and beginner and advanced levels.

- Meoweb

Meoweb used fancy displays to make the process of learning programming more fun and approachable [35]. The puzzle games consisted of the manipulation of codes in order to solve the set problems. Logical thinking was also required from the players to advance through different obstacles and levels to complete goals and ultimately reach the final destination. The basic concepts of CSS programming could be grasped by completing the levels. The main features included were as follows: development of problem-solving skill, reward system, leveling system, and logical thinking.

- BeBlocky: Kids Code Easy

BeBlocky was an engaging game that taught basic programming [36]. Target players were mainly children and aspiring novices. The player encountered several robots that needed to be programmed by dragging and dropping programming blocks in a sequential way. The features included were memory boosting; developing aptitude in sequence, loops, and commands; improvement of problem-solving skills; development of logical reasoning; and leveling system.

- Coding Galaxy

The game provided an interactive and user-friendly interface for learning about basic programming concepts [37]. It was designed and reviewed by skilled teachers and specialists who incorporated core methods traditionally used for teaching programming in the system. The targeted players were students aged 5 and above. The game consisted of more than 200 levels whereby the player was expected to complete missions and objectives and solve programming puzzles. The features included were development of computational thinking, problem-solving, critical thinking, communication and leadership skills, development of creativity and teamwork, learning through adventure and quest system, monitoring of user performance, learning report, sequence, looping, conditional logic, function, and parallelism.

- Grasshopper: Learn to Code for Free

Grasshopper consisted of several mini-games guiding players toward all the basic concepts needed in the JavaScript programming language [38]. The player needed to use codes to solve puzzles. Upon completion of all the game levels, the player should be able to write basic JavaScript codes. The features included were calling functions, variables, strings, for loops, arrays, conditionals, operators, objects, arrays, recursions, and HTML.

8.2.3 Skills Required for Introductory Programming

After thorough analysis of the selected articles, the important skills deduced to be imperative when it comes to learning introductory programming have been highlighted in Table 8.1. The logical skills have been devised from the literature, while the technical skills were based on the ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013 [45].

8.3 Analysis

The previous section gave an overview of some of the related works on serious programming games. This section provides an analysis of those works with respect to the ACM/IEEE guidelines for the programming module. The comparison table visually highlights the main features of the serious games surveyed from the literature and those commercially available.

8.3.1 Comparative Analysis of the Related Works

The related works have been analyzed with respect to the ACM/IEEE-CS Joint Task Force on Computing Curricula 2013 [45]. More specifically, Table 8.2 shows how the related works try to address the skills highlighted by ACM/IEEE in terms of algorithms and design, fundamental programming concepts, and development methods. As it can be observed from the table, *Code Combat* is by far the game that provides for most of the features as recommended by ACM/IEEE followed by *Hour of Code* and *May's Journey*.

8.3.2 Comparative Analysis of the Commercial Games

Table 8.3 shows an analysis of the existing commercial programming games with respect to their gaming features and the logical and technical skills targeted for the commercial games. As it can be observed from Table 8.3, *Hacked* is the game that has the highest number of features followed by *SpriteBox* and *Coding Galaxy*. These three games attempt to cater to a high number of technical skills and at the

Table 8.1 Programming skills table

Logical skills	
Problem-solving	One of the factors that affect students' academic performance when it comes to learning introductory programming is low problem-solving skills. According to a study, a high number of novice students failing programming courses was due to a lack of problem-solving skills. Training through activities that help improve this skill can be highly beneficial to students' performance in computing studies [39]
Debugging	Debugging can be considered as a form of "learning from mistakes" strategy. It is well known that students can effectively learn through their mistakes, and in this case, debugging skills are developed when the students have to browse their own code to identify the source of the problem. This is yet another essential skill required to become a good programmer [40]
Testing	Improving testing skill is crucial to increase productivity. Without this skill, a beginner might have difficulty in making a correct working program which has all the required functionality. This skill should be developed from the start and worked on to have a positive impact on the aspiring programmer [41]
Algorithmic thinking	A study proved most students learning introductory programming had underdeveloped algorithmic and computational skills [42]. This skill is crucial for beginners in programming as this is what will enable them to define clear, concise steps to solve any problem, which is basically what the basis of programming is about. Basically, algorithmic thinking consists of the following: • The ability to analyze given problems • The ability to specify a problem precisely • The ability to find the basic actions that are adequate to the given problem • The ability to construct a correct algorithm to a given problem using the basic actions • The ability to think about all possible special and normal cases of a problem • The ability to improve the efficiency of an algorithm Improving this essential skill which has a strong creative aspect includes solving a maximum of problems. By providing the student with simple problems with gradual increasing difficulty, the latter can effectively work on this skill [43]
Sequencing	This is a common process for writing codes. Instructions are given in a specific order, and the computer processes and executes them accordingly. The development of this skill allows the programmer to think like the computer and hence solve programming problems more efficiently. This is yet another core skill required for novice programmers [44]

(continued)

Table 8.1 (continued)

Technical skills	
Variables and datatypes	These are the basics of computer programming. Every code makes use of variables and data types to solve problems. This knowledge is vital to be able to write codes effectively
Functions	Functions also form part of the basics of programming language. Use of functions allows code to be modular, clear, and concise to be able to write good-quality codes
Loops	The ACM Computer Science curricula 2013 also includes the concept of loops as the programming fundamentals. Beginners in programming must understand how loops work and how to include them in their code in order to effectively solve problems
Arrays	Fundamental data structures also include arrays that are another important factor in writing codes. The use of arrays is very common, and many problems require this concept to be able to tackle problems accordingly
Decisions/conditions	Fundamental programming concepts include decisions/conditions that is yet another core element in programming. ELSE and SWITCH statements are very common in programming and crucial for solving problems

same time aim at developing the logical skills of the player. They also have several gaming features, as one would expect from a normal game.

8.3.3 Summary of Findings

Most related works concluded that a serious game for learning programming positively influenced students' academic performance. For instance, Yallihep and Kutlu [26] conducted a research using a popular mobile serious game "LightBot," and Du et al. [20] also conducted an evaluation of a programming game, which showed a great increase in motivation as well as performance in students. Several games were developed that aimed at programming comprehension by combining programming concepts with gaming mechanisms [21, 22, 25, 27]. This method proved highly beneficial as it made learning process more fun compared to traditional ways. Learning programming through gaming mechanisms proved to reduce anxiety and significantly improve the learning curve in students [29, 30]. Karram [28] investigated *Code Combat*, which proved to accelerate and improve the learning process of students. The game targeted several important technical skills as per the guidelines of the ACM/IEEE. However, the gameplay of *Code Combat* seems to follow the same format for all the different levels. It can eventually be deduced that gamified learning has proved to be a great way to learn the basics of programming. It was also observed that a series of logical skills were also imperative when it came to learning programming. Computational thinking skills like problem-solving, sequencing, debugging, and algorithmic thinking to facilitate

Table 8.2 Comparative analysis of games from the literature

	Algorithms and design			Fundamental programming concepts									Development methods	
Serious game	Problem solving	Algorithmic thinking	Sequencing	Non-specific programming concepts	Syntax and semantics	Variables and data types	Expressions and assignments	Input and output	Conditionals and iteratives	Functions and parame-ters	Recursion	Arrays	Debugging	Testing
May's Journey [19]	✓		✓	✓	✓	✓	✓					✓	✓	
Hour of Code [20]	✓	✓	✓	✓				✓	✓		✓		✓	
Robot ON! [21]	✓		✓	✓		✓	✓							✓
Program your robot [22]	✓	✓	✓						✓		✓			
Array Adventure [23]	✓			✓		✓			✓			✓		
Board Game [24]	✓	✓												
OOP Serious Game [25]	✓			✓						✓				
LightBot [26]	✓	✓	✓						✓		✓		✓	
Restaurant game [27]			✓			✓								
Code Combat [28]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NanoDoc [29]	✓		✓	✓										✓
Curious Robots: Operation Asgard [30]	✓		✓	✓										✓

Table 8.3 Comparative analysis of existing features of commercial games

Category	Features	Hacked [31]	Coding Planet [32]	Light-Bot [33]	Sprite-Box [34]	Meo-web [35]	Be-Blocky [36]	Coding Galaxy [37]	Grasshopper [38]
Gaming features	Adventure/exploration type				✓	✓		✓	
	Multiple difficulty level		✓		✓				
	Tracking of player performance	✓						✓	✓
	Report on performance						✓	✓	
	Reward system	✓			✓	✓	✓		✓
	Tutorial/in-game assistance	✓			✓				
	Leveling system	✓	✓		✓	✓	✓	✓	
	Challenge other players/interaction	✓						✓	
Skills developed	Develop planning skill	✓		✓	✓	✓			
	Improve testing skill			✓					
	Debugging skill			✓	✓				
	Problem-solving skill	✓	✓	✓	✓	✓	✓	✓	✓
	Memory boost		✓				✓		
	Logical thinking	✓	✓	✓	✓	✓	✓	✓	
	Critical thinking							✓	
	Teamwork	✓						✓	
Technical concepts	Use of real codes	✓			✓	✓			✓
	Icon-based programming	✓	✓	✓	✓		✓	✓	
	Functions	✓	✓	✓	✓			✓	✓
	Loops	✓	✓	✓	✓		✓	✓	✓
	Sequencing	✓	✓	✓	✓		✓	✓	✓
	Variables and datatypes	✓				✓			✓
	Parameters	✓			✓				

learning process were brought to attention. Finally, a gamified learning approach to learning programming showed a very positive effect in teaching programming classes. Therefore, games targeting the skills required for programming can greatly improve the academic performance of students studying the subject or even teach beginners about how programming works and introduce them to the world of algorithms and computational thinking. The main game elements used in most of the games were levels, points, leader boards, avatars, and quests. Varying the types of games is vital in order to ensure that the players are motivated and engaged. It is also important for the players to have regular and personalized feedback. Most of the games analyzed do not provide for these two important factors, hence the aim behind our system, *Code-Venture*.

8.4 High-Level Architecture

The high-level architecture is a way of representing how the whole system works and shows the interactions between the different modules. Figure 8.1 shows the high-level architecture of *Code-Venture* consisting of the mobile device, an online database, and a mobile application for teachers.

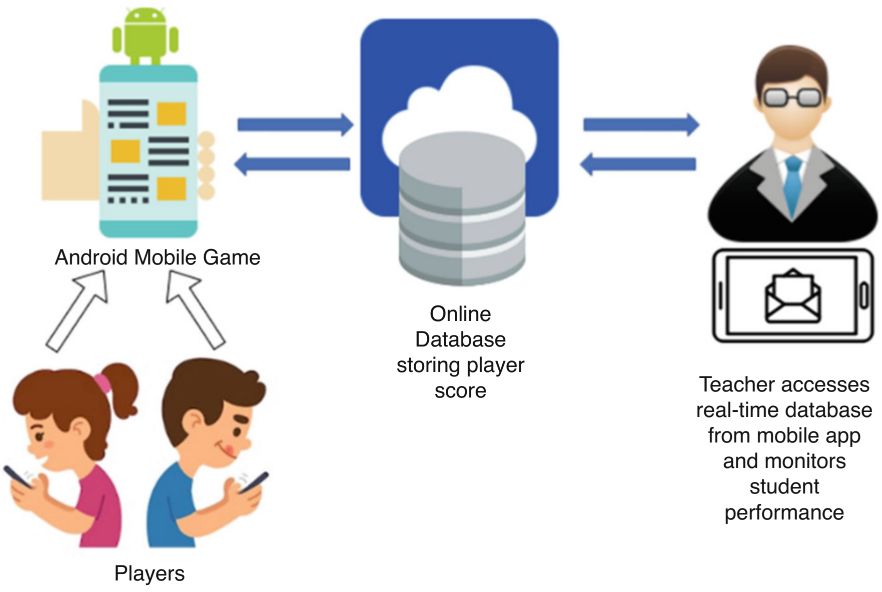

Android Mobile Game
Online Database storing player score
Teacher accesses real-time database from mobile app and monitors student performance
Players

Fig. 8.1 High-level architecture

The main components of *Code-Venture* are as follows:

Players: The players targeted are aspiring programming students, novice programmers, or teenagers with little or no experience in the field. Players will record their personal details and attempt the games through their mobile phones.

Mobile device: The mobile device is where the *Code-Venture* mobile app that consists of different gameplay will be downloaded, installed, and played. Storing of scores is made possible and pushed to the online database.

Online database: The information collected from the game, that is, the player details, scores, and areas of weaknesses, will be saved on the online database for future access by the tutor.

Teacher's mobile application: The students' information retrieved from an online database is displayed on a mobile application designed especially for the teacher. The latter can review player's score, weaknesses, and overall performance of the students on the different games through this application. This app will also allow the tutor to send personalized messages to the students.

8.4.1 The Mini-games of Code-Venture

As compared to existing serious programming games, *Code-Venture* will consist of different mini-games, namely, the *Adventure Mode*, the *Variable Runner*, the *Algorithm Puzzle*, and the *Quiz*, where each of the mini-game targets different skills. Table 8.4 shows the mini-games of *Code-Venture* and the skills that each game targets.

Table 8.4 *Code-Venture* mini-games and the skills targeted

		Code-Venture mini-games			
Category	Skills	Adventure Mode	Variable Runner	Algorithm Puzzle	Quiz
Logical skill	Problem-solving	✓		✓	✓
	Debugging	✓			✓
	Testing				✓
	Algorithmic thinking	✓		✓	
	Sequencing			✓	
Technical skill	Variables and data types	✓	✓		✓
	Functions	✓		✓	
	Decisions	✓			✓
	Loops	✓		✓	✓
	Arrays	✓			

8.4.2 Justifications for Code-Venture's Mini-games

Code-Venture includes several mini-games to differentiate it from existing serious programming games. Giving the player's the opportunity to play a variety of games ensures that the player is exposed to several gameplays and hence remains engaged and motivated. The more engaged and motivated a player is, the more the benefits that can be obtained from playing the serious game. The different mini-games of *Code-Venture* have been carefully planned and are backed by evidence from the literature whereby similar implementations have been successful in educating the players. Table 8.5 describes *Code-Venture* mini-games in more detail and gives justifications as to why these games have been chosen.

8.5 *Code-Venture*'s Implementation and Testing

Code-Venture was implemented with Unity 2019.2.15f1 personal edition and Visual Studio 2017. The C# programming language over .NET2.0 framework was used to program the different mini-games. The computer used consisted of a fourth-generation Intel core processor, a RAM of 8GB, and a 250GB SSD together with an NVidia graphic card GTX 1050ti. Two mobile phones were used to test the application, namely, a Samsung A20 (Android 9.0) and an HTC Desire 828 (Android 5.1.1) having a RAM of 3 and 2 GB, respectively. Different tests were carried out on both emulators and the actual mobile devices including a user acceptance test with 35 students most of whom were to embark on undergraduate studies.

8.5.1 Code-Venture's Main Functionalities

The main functionalities of *Code-Venture* are described and illustrated below.

- Register or Login and Main Menu

The player should enter login details and select "Play Now" on opening the game if he/she is an existing user, as shown in Fig. 8.2. In case of a new user, the "Register" button registers the necessary user details. After signing in, the user is then presented with the menu screen where the different game modes can be accessed as shown in Fig. 8.3.

Table 8.5 Justifications for *Code-Venture* mini-games

Proposed game	Game description and targeted skills	Evidence from literature
Adventure Mode (Coding Adventure)	This is the main game that consists of a 3D environment whereby the player is able to explore freely and solve programming puzzles at the same time. The player is able to trigger dialogues with non-playable characters in the game and accept quests from them. Each of the quests targets different types of mini-games that address different programming problems • Code-rearrange—rearrange bits and pieces of a code in the correct order • Find-error—choose the line of code which is causing an error in the code • Find-output—determine the output of the code when run • Fill-blanks—write the correct code in the empty boxes provided in a piece of code The game also has a "hard mode," which is the same story and environment but with more complex and challenging programming puzzles	Junaeti et al. [23] concluded through research that learning in the form of adventure games had a positive effect on the players' learning process. Players proved to be more motivated and engaged while playing this type of game Jemmali and Yang [19] made a 3D fully immersive adventure game with an interesting storyline to draw the players' attention. The implementation of programming concepts in a visual and interactive way proved to enhance the players' willingness to keep playing. Implementing programming concepts in this environment made the player have a fun gaming experience instead of the feeling of being forced to learn something that may demotivate the player quickly
Variable Runner (game for learning concept of variables)	The game is set in a platform where the player can move the hero left or right as the latter keeps running forward. While running, the hero encounters balls with different values, which are the possible answers to the question being displayed. The player is first presented with a question text and is expected to select the appropriate answer by colliding with it. A scoring system is also included whereby each correct answer adds to the score and vice versa	Miljanovic and Bradbury [21] made use of a gaming mechanism to tackle concepts like variables and data types. A puzzle-type game proved to be effective in helping to learn as well as improving skills like critical thinking and problem-solving Zhao et al. [27] investigated and came up with a game that teaches concepts of variables through foods set in a restaurant. The player is expected to engage with the game objects that represent data types in order to gain a deeper knowledge about this programming concept

Algorithm Puzzle (game teaching basic programming concepts through commands)	This game aims at teaching the player the basic programming concepts. The player has to issue commands to the hero, and the latter will execute the instructions accordingly. Icons are used to give instructions that are then applied to the hero upon execution. The skills targeted are as follows: • Computational thinking skill • Problem-solving • Planning • Testing • Debugging • Logical and critical thinking skill Furthermore, the following basic high-level programming concepts are also involved: functions, loops, and sequencing	Du et al. [20] concluded that visual programming with the use of blocks was effective in making players better understand programming concepts. Base skills like computational thinking skills, problem-solving, and debugging were developed, which further helps master programming Karram [28] deduced from the popular game *Code Combat* that learning programming through the use of commands and instructions proved to be beneficial in teaching players the basics of programming
Quiz (game consists of a series of questions to test the players' current knowledge)	This game tests the players' knowledge about programming. It also acts as a learning game since it consists of several multiple-choice questions that have to be attempted by the player. The progress and correct or wrong answers are recorded for future evaluation of the performance of the player. The incentive provided to play this quiz game and excel in it is that there are rewards that are obtained if the questions are correctly answered. This motivates the user to attempt the questions seriously and as a result gain further knowledge on programming. The game is adaptive, that is, the complexity of the questions increases as the player progresses in the game	Junaeti et al. [23] made use of incentive and competition mechanisms in order to assess the performance of the player through various quizzes after each game level. The results proved to be successful since players revealed that they gained knowledge on programming by answering the questions Lotfi and Mohammed [25] made use of in-game assessments for each game level in order to evaluate the player about four OOP concepts. The learner's performance was monitored through a scoring and timing system

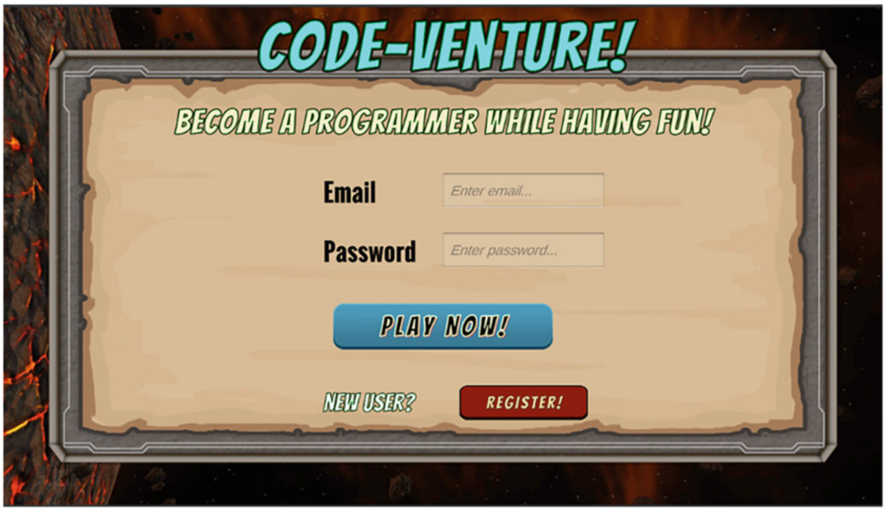


Fig. 8.2 Main menu—login or register

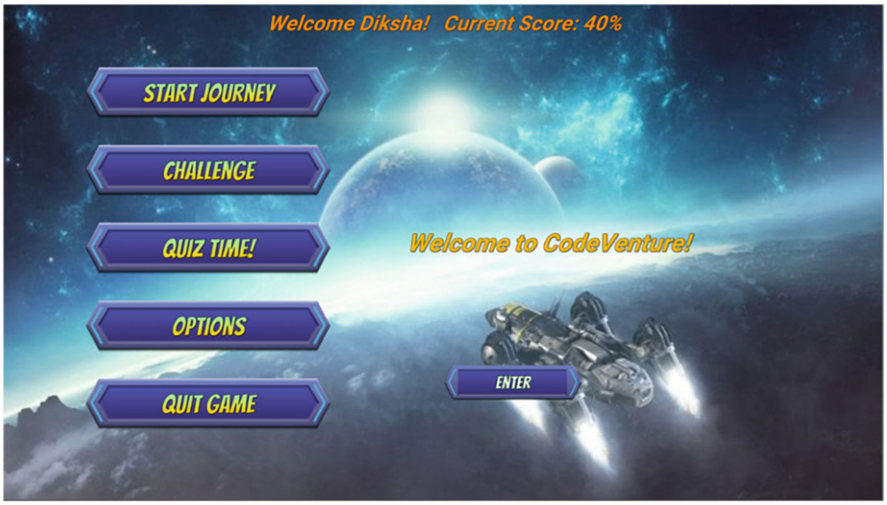
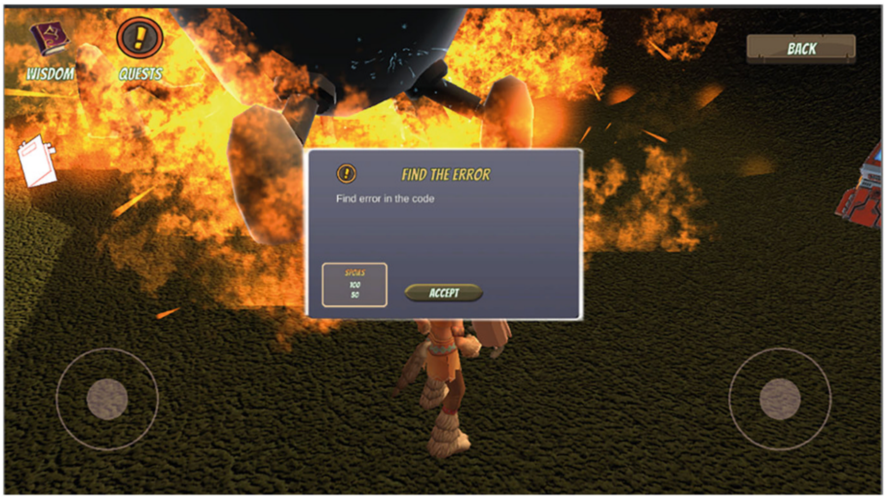


Fig. 8.3 Access main menu with player details

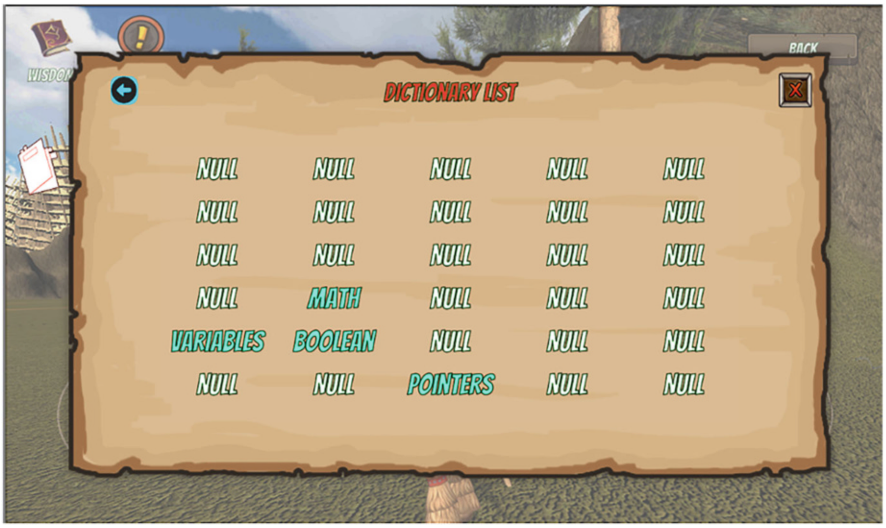
- Adventure Mode and Different Mini-Games

Upon choosing the Adventure mode, the player is able to move the character around the 3D environment and rotate the camera (Fig. 8.4). The player can gather collectibles, trigger dialogues, accept quests, and view the current progress as illustrated in Fig. 8.5.



Find error in the code

Fig. 8.4 Adventure mode—accept quest



NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL
NULL	MATH	NULL	NULL	NULL
VARIABLES	BOOLEAN	NULL	NULL	NULL
NULL	NULL	POINTERS	NULL	NULL

Fig. 8.5 Adventure mode—view progress

- Challenge Mode, Variable Runner, and View Score

The player can select the Challenge mode game option and choose to play "Variable Runner" whereby he is able to move the character left or right to choose the ball which corresponds to the correct answer to the question provided as shown in Fig. 8.6. The player is presented with final score (Fig. 8.7) when the timer is over.

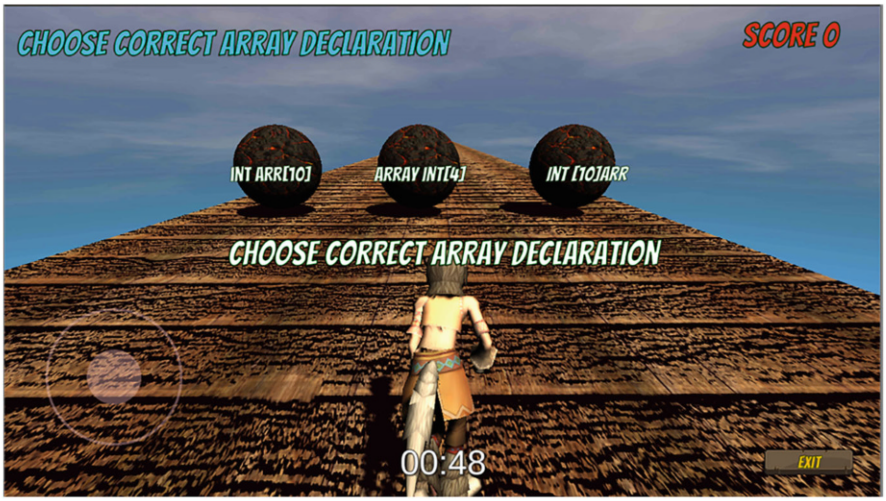
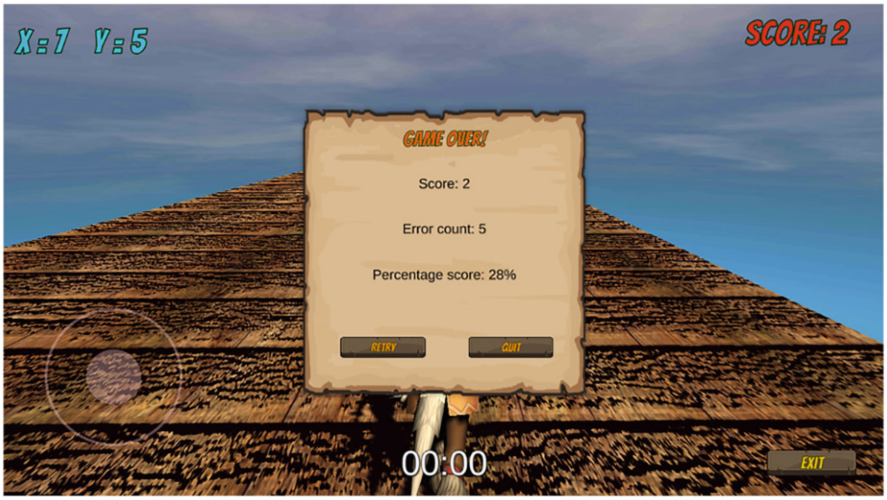

CHOOSE CORRECT ARRAY DECLARATION
SCORE 0
INT ARR[10]
ARRAY INT[4]
INT [10]ARR
CHOOSE CORRECT ARRAY DECLARATION
00:48
EXIT

Fig. 8.6 Variable runner game



Score: 2

Error count: 5

Percentage score: 28%

Fig. 8.7 Variable runner score display

• Challenge Mode, Algorithm Puzzle, and View Score

Upon choosing the "Challenge Mode" game option and entering the "Algorithm Puzzle" game, the player is presented with a scene with interface commands as shown in Fig. 8.8. The player can make use of different functions and loops in order to complete a level. The character can be moved by choosing different commands, and the score is displayed after each level is completed (Fig. 8.9).

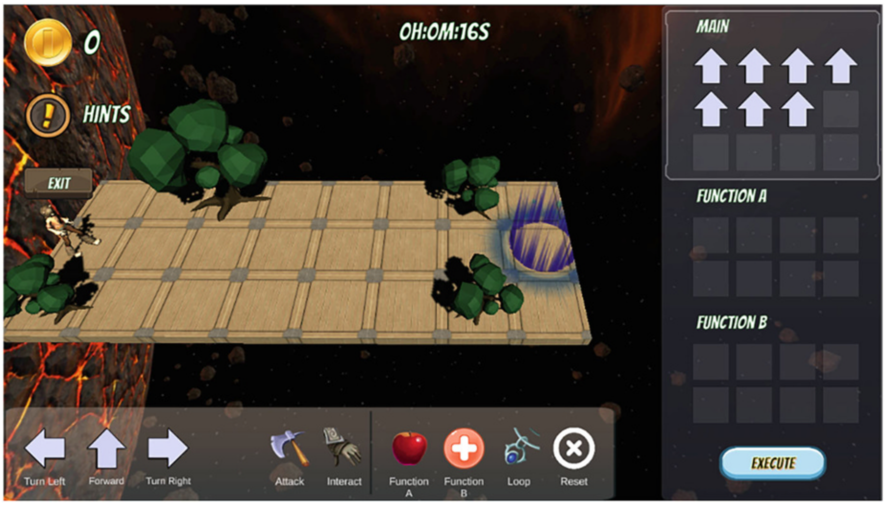
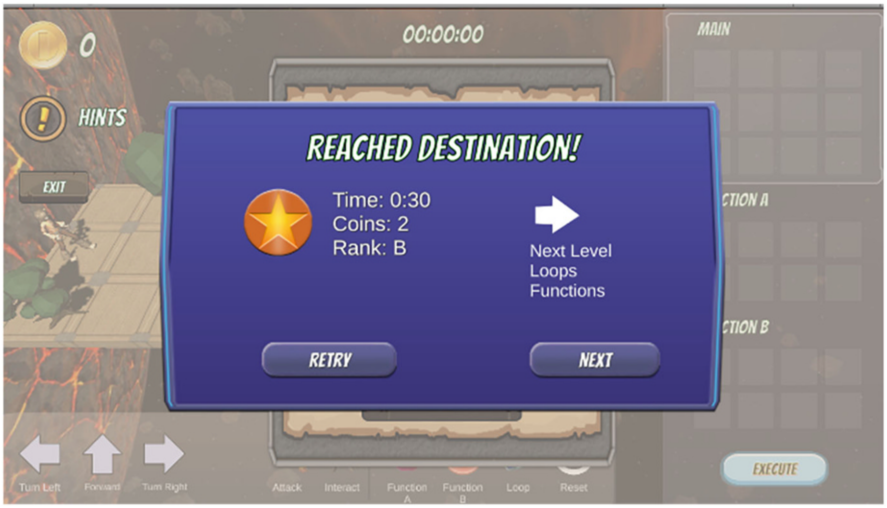


Fig. 8.8 Algorithm puzzle game

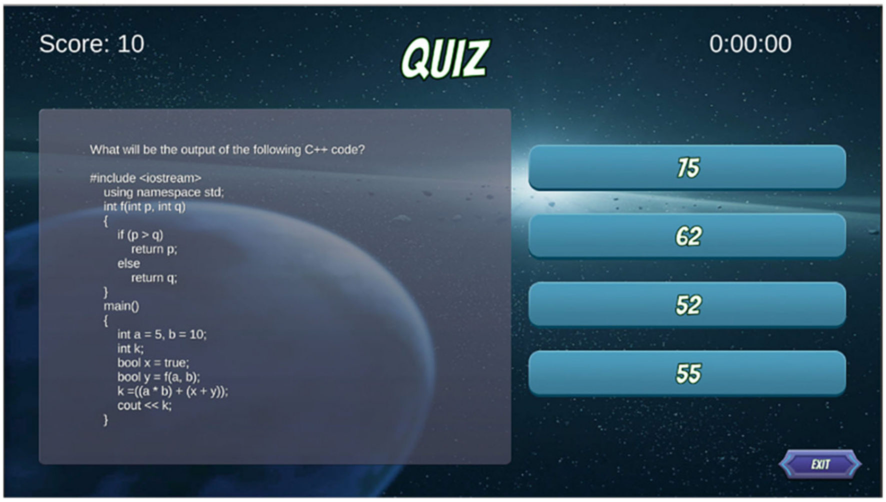


Time: 0:30
Coins: 2
Rank: B

Fig. 8.9 Algorithm puzzle score

- Quiz Game and Score

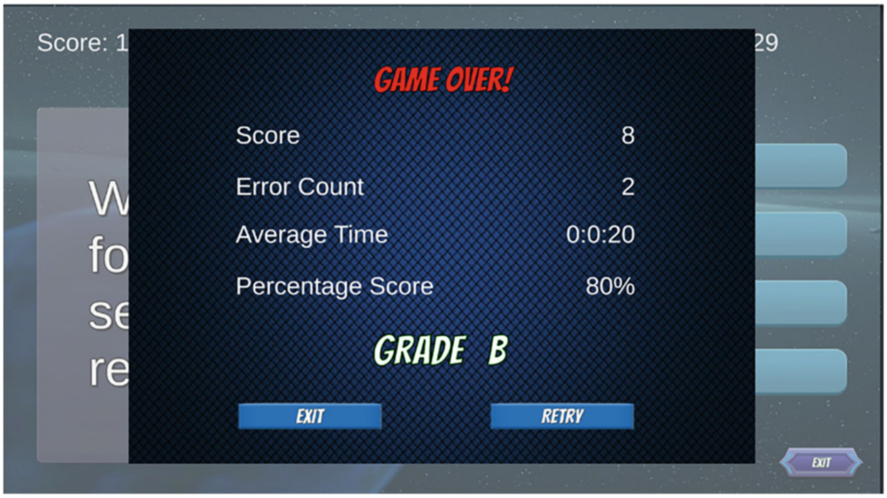
After selecting the quiz game, the player is presented with a question together with four possible answers, as shown in Fig. 8.10. The player must choose an answer whereby the correct answer is highlighted in green, while a red color is used for selection of an incorrect answer. After attempting ten questions, the player is given his final score together with the grade, error count, and time taken to complete the questions (Fig. 8.11). The quiz consists of several levels where the difficulty increases as the player progresses in the game.



What will be the output of the following C++ code?

```
#include <iostream>
  using namespace std;
  int f(int p, int q)
  {
    if (p > q)
      return p;
    else
      return q;
  }
  main()
  {
    int a = 5, b = 10;
    int k;
    bool x = true;
    bool y = f(a, b);
    k =((a * b) + (x + y));
    cout << k;
  }
```

Fig. 8.10 Quiz game

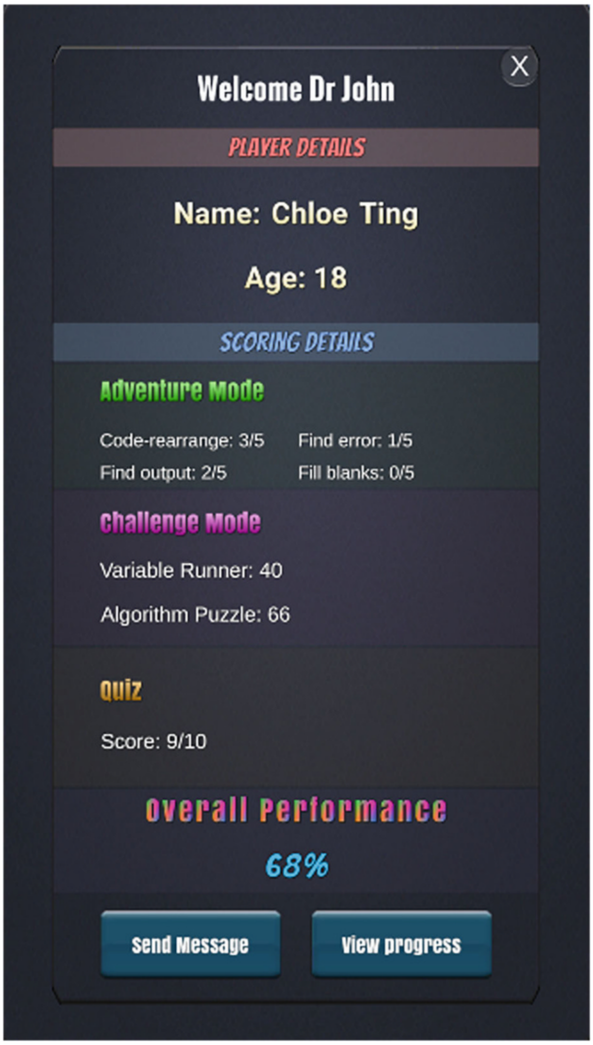


```
Score                8
Error Count          2
Average Time         0:0:20
Percentage Score     80%
```

Fig. 8.11 Quiz game score

- Teacher Application

The teacher can sign in by entering his/her login credentials and then get access to the list of players of *Code-Venture* for a specific class. After logging in, the teacher can select any player to view his/her performance details on the different games (Fig. 8.12) as well as send a private message to the player as shown in Fig. 8.13.


Welcome Dr John

PLAYER DETAILS

Name: Chloe Ting

Age: 18

SCORING DETAILS

Adventure Mode

Code-rearrange: 3/5 Find error: 1/5
Find output: 2/5 Fill blanks: 0/5

Challenge Mode

Variable Runner: 40

Algorithm Puzzle: 66

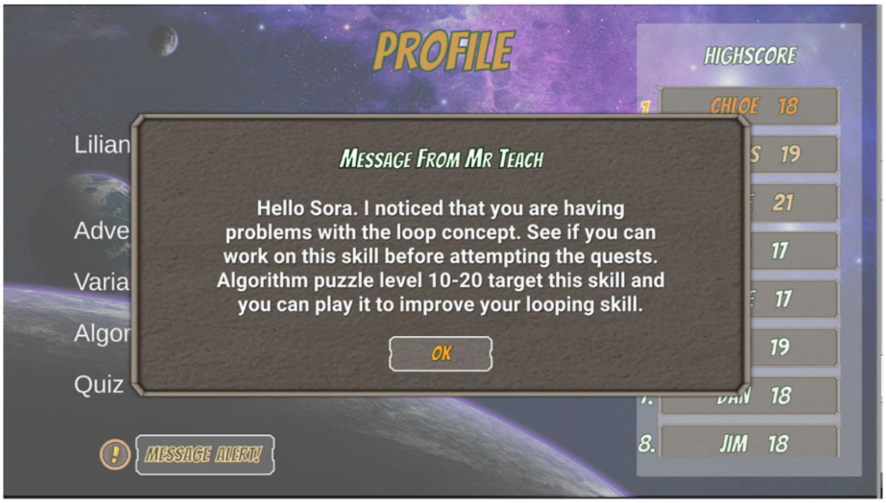
Quiz

Score: 9/10

Overall Performance

68%

Fig. 8.12 Player performance details



MESSAGE FROM MR TEACH

Hello Sora. I noticed that you are having problems with the loop concept. See if you can work on this skill before attempting the quests. Algorithm puzzle level 10-20 target this skill and you can play it to improve your looping skill.

Fig. 8.13 Personalized message sent to student

8.5.2 User Acceptance Testing

A User Acceptance Testing (UAT) was the last phase of the testing process that was performed by end users. This was to validate the software among the targeted audience, to get their precious feedback, and ensure the application met its intended purpose. Pre-game and post-game surveys have been conducted with 35 students who have very little to no perception about programming and were about to embark on undergraduate studies. After inquiring about certain basic information through an online form, they were made to play all games in *Code-Venture* for a period of 1 week. A feedback of their gaming experience was then taken by the means of a second survey.

8.5.2.1 Pre-game Results

The 35 students who tested the application were aged between 16 and 21 years inclusive, 57.1% were female, and the remaining 42.9% were male. Among others, the participants were asked on their background knowledge about programming, and the results, illustrated in Fig. 8.14, showed that 42.9% of them had only a vague notion of programming and 45.7% had none.

They were also asked about their feelings on programming, and the results are shown in Fig. 8.15. 68.6% of the participants thought that programming was hard, 22.9% were of the opinion that programming is medium difficult, and the remaining 8.6% thought programming was easy.

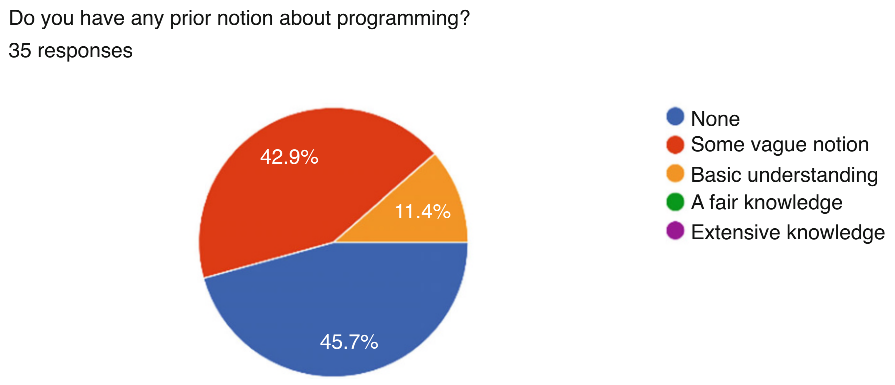


Fig. 8.14 Prior knowledge of programming

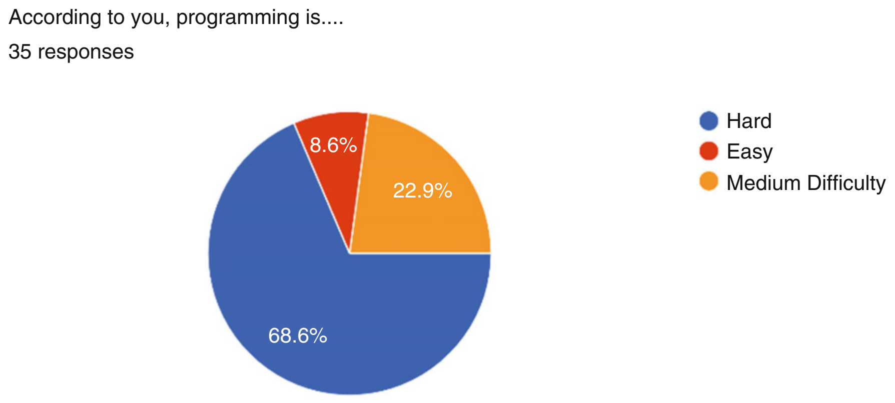


Fig. 8.15 Programming difficulty

The respondents were also asked about intention of pursuing further studies in programming, and the results are shown in Fig. 8.16. 65.7% of the participants mentioned that they did not intend to pursue their studies in programming, 11.4% intended to embrace the field, and the remaining 22.9% were unsure.

8.5.2.2 Post-game Results

The 35 students were given *Code-Venture* to play for 1 week and were required to answer a post-game survey. As illustrated in Fig. 8.17, 42.9% of the participants liked the game very much, while 31.4% liked the game. Twenty percent were neutral about the game, while 5.7% did not like *Code-Venture*. When asked whether they wanted to play *Code-Venture* more in the future, 91.4% of the respondents answered favorably.

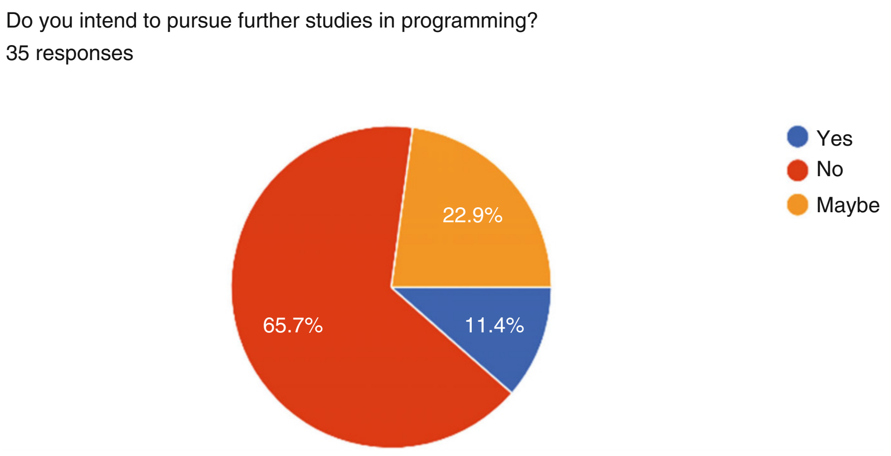


Fig. 8.16 Further studies in programming

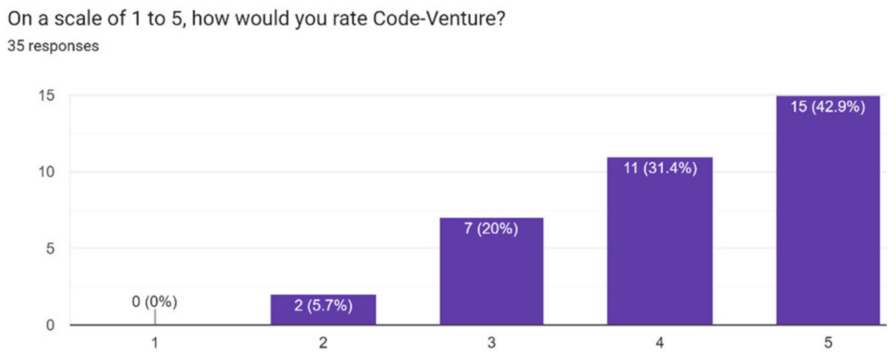


Fig. 8.17 *Code-Venture* ratings

The participants were also asked to rate *Code-Venture* as a means of teaching programming. As shown in Fig. 8.18, 51.4% found the game to be useful, and 3.14% enjoyed the game and received some information on programming. 11.4% were not able to decide whether the game was useful in helping them better understand programming, while 5.7% found the game to be minimally useful. It is also worth noting that none of the respondents indicated that the game was not useful at all in helping them in programming. When prompted about game-based learning, 77.1% of the respondents were of the opinion that game-based learning is a good approach to learn programing, 20% were unsure, and one participant mentioned that it is not suitable.

After playing *Code-Venture*, the participants were again asked about their feelings on programming. As shown in Fig. 8.19, this time, 40% of the respondents found it to be easy, compared to the 8.6% obtained prior to playing *Code-Venture*. Only 5.7% found programming to be hard post *Code-Venture*, while previously a

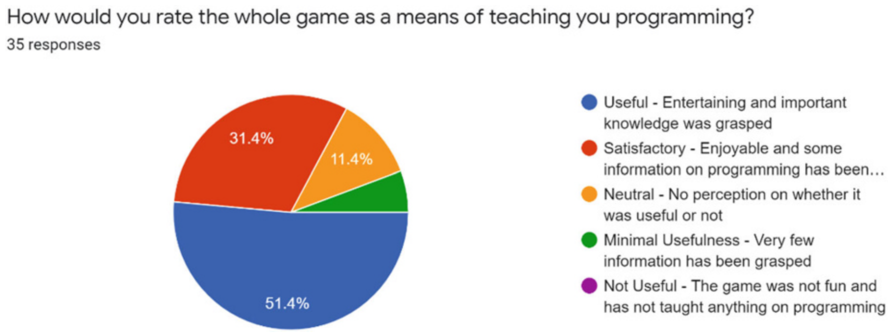


Fig. 8.18 Usefulness of *Code-Venture*

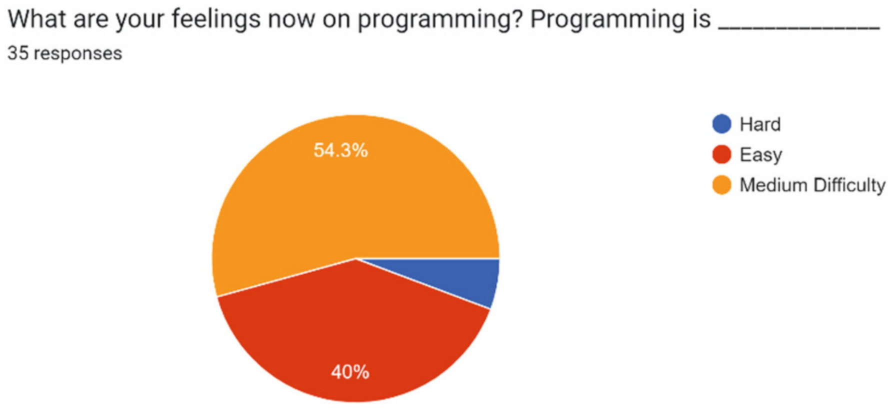


Fig. 8.19 Feelings on programming after playing *Code-Venture*

huge proportion of 68.6% found it to be difficult. The percentage of students who found programming to be slightly difficult rose from 22.9% to 54.3% with many respondents moving their feelings from *Hard* to *Medium difficult*.

When the participants were again asked about their intention to pursue further studies in programming, 32.4% responded positively, compared to the 11.4% previously. Only 5.8% (two participants) mentioned that they did not want to have their further studies in programming compared to a whopping 65.7% previously. The percentage of students who were unsure about their further studies in programming rose from 22.9% to 61.8% with many respondents moving their opinions from *No* to *May be* (Fig. 8.20).

8.5.2.3 Overall Feedback from Students

The following feedback were compiled from the post-game surveys.

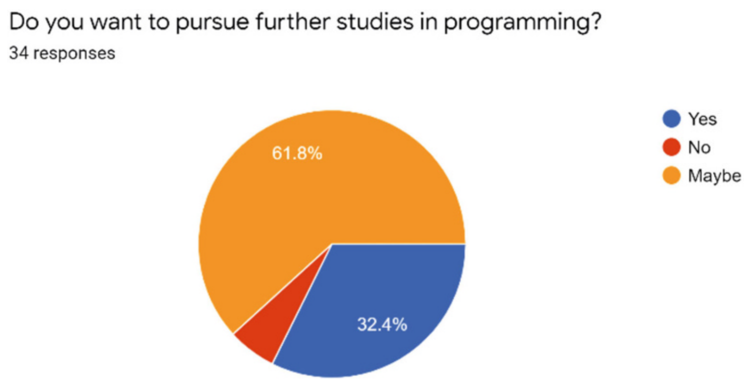


Fig. 8.20 Further studies on programming after playing *Code-Venture*

- Adventure Game—Positive

The game proved to bring a sense of motivation and eagerness to learn. The element of game exploration and quests further enhanced the player's determination to progress through the game and complete all the achievements. By doing so, the students had to go through programming concepts and rules that made them more familiar with the world of programming.

- Variable Runner—Satisfactory

This game proved to be informative and did have a good response from players. However, the players expressed that it was too basic and systematic. The concepts were only being introduced without proper explanation on how these concepts actually work. Hence, comprehension of variables and data types was only partially achieved.

- Algorithm Puzzle—Positive

This game proved to be useful to most of the participants, as they understood the game easily and managed to complete several levels with a good score. Upon questioning, they said they could understand the concept of commands and sequencing and were able to tackle these puzzles easily. The skills targeting algorithmic thinking, problem-solving, debugging and functions, and loops and sequencing proved to have been successfully inculcated in the students.

- Quiz Game—Neutral

This game ended up having a neutral effect on the students since they did not really have much knowledge about programming initially. Hence, answering a set of questions regarding this topic proved to be tough for them. As a result, a majority of the students scored low marks for this game. Thus, it has been deduced that this game mode would be much more efficient if played after attempting the other games or after being exposed to some more programming principles.

8.6 Discussion

Code-Venture has several strengths as it has been designed based on the ACM/IEEE guidelines for introductory programming. The game consists of several mini-games that target different skills required for learning programming. Each mini-game differs from the other, making the gameplay experience fun, engaging, and appealing. The game has an adventure mode where the player is able to venture out and attempt programming puzzles while exploring a beautiful interactive 3D world and making the gameplay experience interesting with in-game characters to interact with. The player gets the feeling of actually playing a real game while indirectly learning about programming. The player gets even more motivated to excel in the game due to the high score/leaderboard system that triggers the element of competition among the players. The system has a mobile application, which is used by the teacher to monitor the scores and overall performance of the players while they attempt the different games. He can hence easily determine the areas of weaknesses of the students, and thus, targeted assistance can be provided to all the players based on their gaming and learning experience. The game elements used in *Code-Venture* include levels, points, leader boards, avatars, and quests. It also uses the following game mechanics: health, energy, coins, time, position, attack, interact, movement, and opening chests.

The pilot testing carried out with 35 participants resulted in some promising outcomes. After playing *Code-Venture*, 40% of the respondents found programming to be easy, compared to the 8.6% obtained prior to playing the game. Only 5.7% found programming to be hard after playing *Code-Venture*, while initially a remarkable 68.6% found it to be difficult. When asked about their intention to pursue further studies in programming, 32.4% of the respondents replied positively after playing *Code-Venture* compared to the 11.4% previously. A mere 5.8% mentioned that they did not want to have their further studies in programming compared to a substantial 65.7% previously. It can therefore be deduced after this pilot study that the mobile serious programming game *Code-Venture* does have a positive impact on the players. These findings may also have practical implications for the tutor delivering the programming module. Game-based learning strategies could be adopted, and the use of serious programming games can be included in the teaching and delivery of the introductory programming modules. Assessments or non-curricular activities may also be designed on the use of the serious games in normal classes or during practical sessions. Obviously, a more thorough testing of the application on a larger scale and for a longer duration is required to have better statistical claims about the effectiveness of the application.

While *Code-Venture* has several strengths, it does have some weaknesses. Internet connection is required for connecting to an online database to store the score of the player, which will be viewed by a teacher. *Code-Venture* has currently been developed for Android platforms only. Moreover, mobile devices older than Android 4.1 are not able to run *Code-Venture* due to incompatibility issues with the new components included in the game. The game takes some storage space, about 130 MB, on the mobile device it has been installed due to it being a 3D game consisting of heavy game objects. As for any other 3D game, *Code-Venture* can also consume a high amount of battery life when played over a prolonged period.

Some avenues for further improvement can be considered in the future. Artificial intelligence can be integrated in the mini-games to have more responsive and adaptive gaming experiences. Customized progress details and graphs concerning the student's strengths and weaknesses in the teacher's application may also be enhanced. Multiplayer game modes, whereby the players are able to challenge and compete with others for rewards and points, will definitely be a big advantage. Additionally, more quests in the game's adventure mode, an improved storyline, and expanding the game environment to increase the areas of exploration can be envisaged. Finally, the development of more mini-games in the challenge mode to tackle more programming skills may also be useful.

Based on our analysis, *Code Combat*, *Hour of Code*, *LightBot*, and *May's Journey* are the most featured currently available serious programming games. Together with *Code-Venture*, they have all been designed to encourage computational thinking among individuals of all ages and assist them in learning fundamental computer science principles. All these games have a good combination of text, audio, and graphics. What really differentiates *Code-Venture* from the other serious games is that *Code-Venture* makes use of a varying gameplay for the different integrated mini-games while the other games have a similar gameplay for all the different levels. *Code-Venture* also includes a mobile application for the tutor who can visualize the progress of the students on the different games. What makes *Code-Venture* unique is the possibility of sending personalized messages to students to advise them on their progress.

There are several threats to validity that could have an impact on the results obtained. Firstly, the sample size was very small with only 35 participants and limited to the students available from the researchers' contacts. The use of a convenience sampling poses a threat to internal validity. Moreover, some students have parental or siblings support at home, which may have affected their interaction with *Code-Venture*. Additionally, since the ages of the participants were between 16 and 21, the maturity of the respondents may also affect the results. Finally, the students were given the mobile application for a period of only 1 week. The time that the students interacted with the application may therefore not be the same. A greater sample size would have helped minimize these issues.

8.7 Conclusion

This work investigated the possibility of applying a game-based learning strategy to teach novice students about programming through a serious game named *Code-Venture*. *Code-Venture* was based on the ACM/IEEE Computing Curriculum for programming. The main objective of *Code-Venture* was to enlighten the students about this seemingly complex programming subject and show them that it could be fun and enjoyable to learn the programming principles. *Code-Venture* makes use of varying gameplays to ensure the player is engaged and motivated. While the mini-games serve as a challenge for the player, the adventure-mode creates an immersive experience for learning. This game is associated with an application that helps teachers monitor their students' performance. With the elaborate scoring system, the student's skills, strengths, and weaknesses can be evaluated.

A pilot study has been carried out with 35 students together with pre-game and post-game surveys. The survey results were analyzed, and the outcomes were very promising. The participants who tested *Code-Venture* were very entertained and engaged and showed keen interest in playing more. They got a better perception about programming and expressed their will to learn more about it. A positive change was noted in the opinion of the students regarding programming. After playing *Code-Venture*, 40% of the respondents found programming to be easy compared to the 8.6% obtained prior to playing the game. Only 5.7% found programming to be hard post-*Code-Venture*, while previously an impressive 68.6% found it to be difficult. When the participants were asked about their intention to pursue further studies in programming, 32.4% responded positively post playing *Code-Venture* compared to the 11.4% previously. Only 5.8% mentioned that they did not want to have their further studies in programming compared to a massive 65.7% previously.

Code-Venture is a promising game providing ease of access, viability, and the opportunity to sharpen one's knowledge and expand one's understanding of programming in a fun and entertaining way. While *Code-Venture* has several benefits, the game is still lacking in some areas. A more elaborated and captivating storyline has yet to be implemented in the *Adventure-Mode* to better grasp the player's attention with more interesting quests and dialogues. The mini-games could be improved by including a better tutorial system to guide the player. The teacher's mobile application has few options to view progress of the students that can also be further enhanced. Moreover, budget limitations resulted in only free assets and resources being considered for the development *Code-Venture*. The lockdown due to the COVID-19 pandemic resulted in a pilot study where the testing was carried out by only 35 participants. Hence, there is a need to carry out a thorough testing of *Code-Venture* over a long period to evaluate the impact of the application on the skills of the players and confirm its effectiveness in helping the players better grasp the main programming principles.

Disclosures No funding was received for this article, and the authors declare no conflicts of interest.

References

1. Papadakis, S., Kalogiannakis, M., Orfanakis, V., Zaranis, N.: Novice programming environments. Scratch & app inventor: a first comparison. In: Proceedings of the 2014 Workshop on Interaction Design in Educational Environments, pp. 1–7 (2014)

2. Miskon, M.T., Hilmi, F.D., Khusairi, W.A., Rustam, I.: Development of constructionist robotics to facilitate learning in C programming course. J. Phys. Conf. Ser. **1529**(2), 022039 (2020)
3. Mathew, R., Malik, S.I., Tawafak, R.M.: Teaching problem solving skills using an educational game in a computer programming course. Inf. Educ. **18**(2), 359–373 (2019)
4. Lahtinen, E., Ala-Mutka, K., Järvinen, H.M.: A study of the difficulties of novice programmers. ACM SIGCSE Bull. **37**(3), 14–18 (2005)
5. Bennedsen, J., Caspersen, M.E.: Failure rates in introductory programming. ACM SIGCSE Bull. **39**(2), 32–36 (2007)
6. Bennedsen, J., Caspersen, M.E.: Failure rates in introductory programming: 12 years later. ACM Inroads. **10**(2), 30–36 (2019)
7. Lye, S.Y., Koh, J.H.L.: Review on teaching and learning of computational thinking through programming: what is next for K-12? Comput. Hum. Behav. **41**, 51–61 (2014)
8. Halbrook, Y.J., O'Donnell, A.T., Msetfi, R.M.: When and how video games can be good: a review of the positive effects of video games on well-being. Perspect. Psychol. Sci. **14**(6), 1096–1104 (2019)
9. Schez-Sobrino, S., Vallejo, D., Glez-Morcillo, C., Redondo, M.Á., Castro-Schez, J.J.: RoboTIC: a serious game based on augmented reality for learning programming. Multimed. Tools Appl. **79**, 34079–34099 (2020)
10. Shahid, M., Wajid, A., Haq, K.U., Saleem, I., Shujja, A.H.: A review of gamification for learning programming fundamental. In: 2019 International Conference on Innovative Computing (ICIC), pp. 1–8. IEEE (2019)
11. Mathrani, A., Christian, S., Ponder-Sutton, A.: PlayIT: game based learning approach for teaching programming concepts. J. Educ. Technol. Soc. **19**(2), 5–17 (2016)
12. Boeker, M., Andel, P., Vach, W., Frankenschmidt, A.: Game-based e-learning is more effective than a conventional instructional method: a randomized controlled trial with third-year medical students. PLoS One. **8**(12) (2013)
13. Ding, D., Guan, C., Yu, Y.: Game-based learning in tertiary education: a new learning experience for the generation Z. Int. J. Inf. Educ. Technol. **7**(2), 148 (2017)
14. Cheng, M.T., Chen, J.H., Chu, S.J., Chen, S.Y.: The use of serious games in science education: a review of selected empirical research from 2002 to 2013. J. Comput. Educ. **2**, 353–375 (2015)
15. Krath, J., Schürmann, L., Von Korflesch, H.F.: Revealing the theoretical basis of gamification: a systematic review and analysis of theory in research on gamification, serious games and game-based learning. Comput. Hum. Behav. **125**, 106963 (2021)
16. Zainuddin, Z., Chu, S.K.W., Shujahat, M., Perera, C.J.: The impact of gamification on learning and instruction: a systematic review of empirical evidence. Educ. Res. Rev. **30**, 100326 (2020)
17. Qian, M., Clark, K.R.: Game-based learning and 21st century skills: a review of recent research. Comput. Hum. Behav. **63**, 50–58 (2016)
18. Tori, A.A., Tori, R., Nunes, F.L.: Serious Game Design in Health Education: A Systematic Review. IEEE Transactions on Learning Technologies (2022)
19. Jemmali, C., Yang, Z.: May's journey: a serious game to teach middle and high school girls programming. Master's thesis, Worcester Polytechnic Institute (2016)
20. Du, J., Wimmer, H., Rada, R.: "Hour of Code": can it change students' attitudes toward programming? J. Inf. Technol. Educ. Innov. Pract. **15**, 53 (2016)
21. Miljanovic, M.A., Bradbury, J.S.: Robot on! A serious game for improving programming comprehension. In: Proceedings of the 5th International Workshop on Games and Software Engineering, pp. 33–36 (2016)
22. Law, R.: Teaching programming using computer games: a program language agnostic approach. In: European Conference on Games Based Learning, pp. 368–376. Academic Conferences International Limited (2017)
23. Junaeti, E., Sutarno, H., Nurmalasari, R.R.: Genius learning strategy of basic programming in an adventure game. In: IOP Conference Series: Materials Science and Engineering, vol. 288, No. 1, p. 012057. IOP Publishing (2018)

24. Jordaan, D.B.: Board games in the computer science class to improve students' knowledge of the python programming language. In: 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC), pp. 1–5. IEEE (2018)
25. Lotfi, E., Mohammed, B.: Teaching object oriented programming concepts through a mobile serious game. In: Proceedings of the 3rd International Conference on Smart City Applications, p. 74. ACM (2018)
26. Yallihep, M., Kutlu, B.: Mobile serious games: effects on students' understanding of programming concepts and attitudes towards information technology. Educ. Inf. Technol., 1–18 (2019)
27. Zhao, D., Muntean, C., Muntean, G.: The Restaurant Game: a NEWTON PROJECT serious game for C programming courses. In: Society for Information Technology & Teacher Education International Conference, pp. 1867–1874. Association for the Advancement of Computing in Education (AACE) (2019)
28. Karram, O.: The role of computer games in teaching object-oriented programming in high schools-code combat as a game approach. WSEAS Trans. Adv. Eng. Educ. **18**, 37–46 (2021)
29. Toukiloglou, P., Xinogalos, S.: NanoDoc: designing an adaptive serious game for programming with working examples support. Eur. Conf. Games Based Learn. **16**(1), 628–636 (2022)
30. Akkaya, A., Akpinar, Y.: Experiential serious-game design for development of knowledge of object-oriented programming and computational thinking skills. Comput. Sci. Educ. **32**(4), 476–501 (2022)
31. Hacked. Google Play. [online]. https://play.google.com/store/apps/details?id=com.hackedapp&hl=en (2015). Accessed 2 Oct 2019
32. Coding Galaxy. App Store. [online]. https://apps.apple.com/us/app/coding-galaxy/id1240651393 (2022). Accessed 20 Feb 2023
33. Lightbot: Code Hour. App Store. [online]. https://apps.apple.com/us/app/lightbot-code-hour/id873943739 (2018). Accessed 20 Feb 2023
34. SpriteBox: Code Hour. App Store. [online]. https://apps.apple.com/us/app/spritebox-code-hour/id1161515477 (2018). Accessed 20 Feb 2023
35. Meoweb: The Puzzle Coding Game. Google Play. [online]. https://play.google.com/store/apps/details?id=br.com.tapps.meoweb&hl=en&gl=US (2020). Accessed 20 Feb 2023
36. BeBlocky: Kids Code Easy. Google Play. [online]. https://play.google.com/store/apps/details?id=com.beblocky.beblocky&hl=en&gl=US (2022). Accessed 20 Feb 2023
37. Coding Planets. Google Play. [online]. https://play.google.com/store/apps/details?id=com.material.design.codingplanet&hl=en&gl=US (2017). Accessed 10 Dec 2022
38. Grasshopper: Learn to Code. Google Play. [online]. https://play.google.com/store/apps/details?id=com.area120.grasshopper&hl=en&gl=US (2023). Accessed 20 Feb 2023
39. Gomes, A., Mendes, A.J.: Problem solving in programming. In: PPIG, p. 18. (2007)
40. Ahmadzadeh, M., Elliman, D., Higgins, C.: The impact of improving debugging skill on programming ability. Innov. Teach. Learn. Inf. Comput. Sci. **6**(4), 72–87 (2007)
41. Fucci, D., Turhan, B., Oivo, M.: On the effects of programming and testing skills on external quality and productivity in a test-driven development context. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–6 (2015)
42. Csernoch, M., Biró, P., Máth, J., Abari, K.: Testing algorithmic skills in traditional and non-traditional programming environments. Inf. Educ. **14**(2), 175–197 (2015)
43. Futschek, G.: Algorithmic thinking: the key for understanding computer science. In: International Conference on Informatics in Secondary Schools-Evolution and Perspectives, pp. 159–168. Springer, Berlin (2006)
44. Farrell, J.: Programming Logic and Design, Comprehensive. Cengage Learning (2014)
45. ACM/IEEE-CS Joint Task Force on Computing Curricula: Computer Science Curricula 2013. Technical Report. ACM Press and IEEE Computer Society Press (2013)