Using Greenfoot as a Tool for Serious Games Programming Education and Development

Stelios Xinogalos, Margarita Maria Tryfou
Department of Applied Informatics, University of Macedonia, Greece
stelios@uom.edu.gr, mai1517@uom.edu.gr

Abstract

Greenfoot is an educational programming environment that aims to motivate students in learning object-oriented programming (OOP) through the development of simple games and simulations. Several studies have presented positive results regarding the usage of Greenfoot as a tool for introducing novices to OOP. In this article, we propose using Greenfoot as a tool for introducing novices to (serious) games programming, as well as a tool for implementing serious games (SGs). As a proof of concept, the design and implementation of an SG for learning OOP utilizing Greenfoot and a survey on the perceptions of graduates that had been introduced to SGs programming with Greenfoot in the context of a relevant Master course are presented. The results show that Greenfoot, although not developed for this purpose, can be utilized both as a tool for introducing novices to (serious) games programming and as a tool for developing SGs.

Keywords: *Greenfoot, Serious Games, Serious Games Programming Education; Serious Games Development; Serious Games for Programming*

1 Introduction

Serious Games (SGs) have attracted great interest for several years now. There is currently a big demand on educating novices on designing and programming serious games [1], as well as on devising tools and game engines that can be used by people with varying skills on pedagogical issues, programming knowledge, knowledge on game technology, and so on for developing SGs [2].

Education on SGs is mostly offered in the context of graduate programs. Although there are several programs and/or courses on various aspects of SGs, the relevant literature is limited [2][3][4]. In most cases, the literature presents the structure and the contents of relevant courses, barriers and corresponding solutions, as well as lessons learned. Information on the tools used for introducing novices to SGs programming is not usually thoroughly addressed. Taking into account the large number of tools and game engines that can be used either for educating novices or for developing SGs, we consider research on tools and game engines for both the aforementioned reasons rather important. In [2] an overview and comparative analysis of nine game engines for developing games is presented in terms of a large number of criteria based on the literature. The game engines analyzed in [2] are grouped into the following categories that cover different needs: professional game engines that offer overwhelming features but usually require prior knowledge in programming and game technology, such as Unity and Unreal; game engines that do not require prior programming knowledge and can be used by novices, such as GameMaker; game engines and frameworks that are based on web technologies, such as Turbulenz; and open source game engines that can be customized by experienced programmers, such as JMonkey. In this article, we propose as an alternative to the

aforementioned types of game engines using an educational programming environment both for teaching game programming and developing games.

Specifically, we aim to investigate the possibility of using Greenfoot as a tool for introducing novices to game programming, as well as a tool for implementing SGs. Greenfoot [5][6] is an educational programming environment that aims to support novices in learning object-oriented programming (OOP) through the development of simple games, simulations, and graphical programs. This is accomplished through a simple and straightforward graphical user interface with various visualization and interaction tools, a simple application programming interface (API) and the widely used programming language of Java. Greenfoot was considered appropriate for a short, three-lesson introduction to programming fundamental game mechanics in the context of a Master course on Serious Games Programming. The initial positive feedback from students in combination with some interesting SGs prototypes implemented by students in the context of this course [1], motivated us to carry out the research presented in this article that aims to answer the following research questions (RQs):

RQ1: Can Greenfoot be used as a tool for an introduction to (serious) game programming?

RQ2: Can Greenfoot be used for implementing SGs?

At this point, we consider that it is important to clarify why RQ1 investigates the effectiveness of Greenfoot as a tool for an introduction to game programming in general and not SGs in specific, while RQ2 investigates the appropriateness of Greenfoot as a tool for implementing SGs. What is the difference between the development of entertainment games and SGs? According to Michael and Chen [7]: entertainment games focus on the richest possible experience that is accomplished through immersive worlds and gigabytes of high-quality content, while SGs focus on the "serious" elements of the game and although entertainment is still an essential element in SGs, the available time and budget are limited; entertainment games are targeted to users that own cutting edge hardware, software and gaming peripherals, while SGs are mainly targeted to users (for example schools) that usually have outdated equipment. Although entertainment is an important feature of SGs, it is not usually approached through high-quality 3D graphics as is the case in entertainment games. Common features that are essential to SGs refer to the presentation of educational material and activities for practicing and/or evaluating students' knowledge, such as quizzes, filling in blanks and so on. Taking all these facts into account, Greenfoot seems to be a good choice for an introduction to (serious) game programming, but also for implementing SGs.

The methodology utilized for investigating the aforementioned RQs comprises of:

- *A literature review*. A literature review on utilizing Greenfoot as a tool for an introduction to game programming and a tool for implementing SGs was carried out. Although several studies have investigated the effect of Greenfoot on teaching and learning OOP, the literature on utilizing Greenfoot for teaching/learning game programming and even more for SGs development is rather limited.
- *A case study*. In order to investigate whether Greenfoot can be utilized for implementing SGs and not just SGs prototypes [1], we decided to design and implement an SG. The SG implemented, called "Game of Code: Lost in Javaland", aims to introduce primary school students to the main concepts of OOP. The game was designed and implemented by the second author of this article under the supervision of the first author. We also have to note that the decision to implement an SG for OOP was heavily based on the authors' special interest in programming and is not considered as limiting the generalizability of the conclusions drawn on the appropriateness of Greenfoot as a tool for implementing SGs.
- *A survey*. A survey with students who had attended and successfully completed a Master course on Serious Games Programming that utilized Greenfoot, among

other tools, for programming SGs was carried out. This survey aimed to assess students' perception of Greenfoot and confirm our initial hypotheses about the effectiveness of Greenfoot for learning game programming and implementing SGs.

The rest of the article is organized as follows. In Section 2 the results of the literature review on the usage of Greenfoot as a tool for introducing novices to game programming, as well as implementing SGs are presented. In Section 3 the design of "Game of Code: Lost in Javaland" is briefly presented utilizing the Serious Game Design Assessment (SGDA) Framework [8], while in Section 4 implementation issues are presented. In Section 5 the methodology and the results of the survey are presented, while in Section 6 the results and limitations of the study are briefly discussed. Finally, in Section 7 some final conclusions are drawn.

2 Related work

Teaching and learning programming is accompanied with several difficulties that are either intrinsic to programming and the teaching approach adopted [9] or are related to specific programming concepts/constructs and programming languages [10]. In order to face such difficulties many educational programming environments [11][12] and SGs for programming [13] have been developed and utilized for an introduction to programming in all levels of education. Currently, the most successful educational programming environments attempt to support students and even more engage and motivate them in learning programming concepts by creating simple games [14]. These environments are targeted to students of all ages. For example: ScratchJr can be used by 5-7 years old children; Scratch [15] is targeted to 8-16 years old students; Alice [16] is appropriate for middle school and even university students; AgentCubes [17] can be utilized by students from the 3rd grade of the primary school; and Greenfoot [5] is proposed for high school or even university students.

Research results regarding the effectiveness of all the aforementioned environments on teaching programming are rather positive [14][18][19]. However, besides the support provided for teaching programming, these environments have the potential to be used for introducing basic game programming concepts as well. Of course, we have to note that Greenfoot is based on the OOP programming language of Java, while the rest of the environments are based on block programming. Consequently, Greenfoot gives the chance to present even more game programming concepts, while it seems to have the potential to be used for implementing SGs due to the use of a conventional programming language. Having all these facts in mind, we further researched for literature on the utilization of Greenfoot.

Several studies with Greenfoot have reported positive results, starting with 9th and 10th grade students in a two week game camp [18] and including even first year higher education students attending their first computing programming course based on labs and a capstone project utilizing Greenfoot [19]. However, the literature on utilizing Greenfoot as a tool for learning game programming or a tool for implementing SGs is much more limited.

Villaverde and Murphy present their experience from using Greenfoot for teaching a senior project course on 2D game development [20]. Greenfoot was considered to be ideal for the purposes of the course for a lot of reasons: it is based on Java that is a widely used programming language and nearly every student has experience in; it has a flat learning curve; it is free and multi-platform; it has an active community and lot of resources for learning Greenfoot and implementing games. Both instructors' and students' experiences were positive. However, some negative aspects of Greenfoot were the lack of version control and the fact that it does not apply any game optimizations, such as memory and time complexities of the data structures and algorithms used [20]. The authors mentioned

as a future plan asking students to prepare a design and optimization document for their Greenfoot games.

Rick et al. propose using Greenfoot for familiarizing students with Business Informatics issues in addition to computer programming [21]. In their work, they present a one-week school project that aimed at developing a simplified airport baggage-handling system.

Finally, Greenfoot was used in the context of a Master course on “Serious Games Programming”, for a brief introduction to SGs programming, as well as for implementing SGs prototypes [1]. The initial feedback from students regarding the usage of Greenfoot for a brief introduction to SGs programming was quite positive [1], while some interesting SGs prototypes were developed in the context of the course [22][23]. Three mini-games on math and literacy, covering material on addition & subtraction, multiplication & divisibility, and filling in words with the correct letter, diphthong or consonant were utilized in two primary schools and were positively evaluated by 21 teachers and 245 students [22]. Two open-source SGs on learning to use effectively the keyboard and the categories of food and healthy nutrition were awarded in an open-source software competition [23]. These initial positive results motivated us to further investigate the usage of Greenfoot for teaching SGs programming and also for SGs development.

3 Design of “Game of Code: Lost in Javaland”

In this section the Serious Game Design Assessment (SGDA) Framework proposed by Mitgutsch and Alvarado [8] is briefly described and then the design of the “Game of Code: Lost in Javaland” is analyzed based on the components and features of the framework.

3.1 Serious Game Design Assessment Framework

The Serious Game Design Assessment (SGDA) Framework was created by Mitgutsch and Alvarado in order to provide a new assessment tool for SGs designing [8]. The SGDA framework consists of six components [8]:

- *Purpose* is the core component around which all the components of an SG revolve. Unlike entertainment games where the main focus is on the gaming experience, SGs are designed to provide more than just fun.
- *Content and information* refer to the information, facts and data offered and used in the game. For example, character names, player statistics, and generally the data that a player has access to, are some of the elements that this component can describe.
- *Framing* relates to the target group to which the educational game, the player's educational level and the relevant knowledge base are addressed.
- *Mechanics* involves establishing the rules of the game, the reward and navigation system, the obstacles, the challenges, the balance between difficulty, the success rate of completion, the activities and how they all relate to the central objective of the game. Actually, mechanics are actions translated in verbs, a widespread technique used by game designers.
- *Fiction and narrative* are both an important component of an SG. The plot focuses on the platform's environment and usually, at the start of the game, there is a brief introduction to the plot. Otherwise the player has the ability to discover it on his/her own during the game. On the other hand, the narrative of the game allows the player to perform different scenarios. That is, if the player can change the route of the game according to the decisions s/he makes.
- *Aesthetics and graphics* refer to the "audiovisual language" of a game, namely images, graphics, sound, style and generally all the artistic means designed,

selected and developed for visualization and the representation of the elements involved in the game.

Every component is purpose-based and has a great impact on the game's *coherence and cohesiveness* which lead to a complete game system. After analyzing the individual components of an SG, a holistic review of the purpose and its relation with all the components has to be conducted. There should be coherence and cohesiveness between storytelling and mechanics in relation to the original concept of design.

3.2 Designing "Game of Code: Lost in Javaland" with the SGDA framework

The open source SG "Game of Code: Lost in Javaland" (Figure 1) was created for educational purposes with the main goal of teaching OOP concepts in Java. The game was designed using the components and features of the SGDA framework. Prior designing the game, SGs for programming that can be freely downloaded, such as CodeMonkey, CodeCombat and Robocode, were studied along with relevant literature.

Miljanovic and Bradbury reviewed forty-nine SGs for programming and eleven of them are targeted to primary and high school students [13]. Thirty six of these games were evaluated mainly through surveys and less often using game play statistics and skill tests or interviews. Out of the sixteen games that were evaluated for their learning effects on students, seven SGs did not yield a statistically significant learning effect. In another recent review, twenty-two games for teaching programming to primary school students were analyzed with the majority of them using some sort of visual block-based language [24]. In contrast with these games, "Game of Code: Lost in Javaland" is an adventure puzzle game where the player actually handles the main character with the keyboard and interacts with other creatures and elements in the game through a variety of activities. The design of the game is described in the following subsections based on the components of the SGDA framework, while the elements of each component are provided using italics.

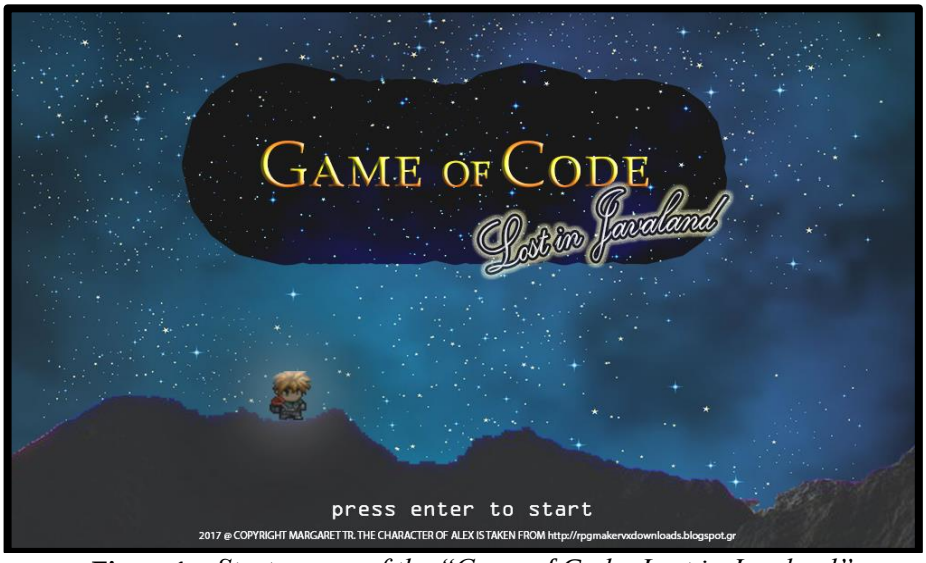


Figure 1. *Start screen of the "Game of Code: Lost in Javaland"*

3.2.1 Purpose

The purpose of "Game of Code: Lost in Javaland" is *educational*. Through the entertainment the player becomes familiar with OOP concepts. Both basic and advanced OOP concepts are presented using theoretical slides and educational activities so that the learner can practice what s/he has learned.

The character of the game has to learn how to code in Java in order to find his way through Javaland, defeat the enemies and save humanity (*the central theme of the game*).

The player is gaining knowledge on Java programming and object-oriented concepts, while experiencing a self-driven flow of education.

3.2.2 Framing

The game is aimed at primary school students (*target group*) without prior programming knowledge. In addition, ease of use of the keyboard and mouse are considered essential (*computer literacy*), as they are the basic means of control and interaction with the game.

The *difficulty of levels* increases from level to level and the *challenges* are getting more demanding. The trainee has to accomplish a variety of tasks by writing real code (*educational tools*) after studying slides of theory and coding examples.

3.2.3 Fiction and narrative

The game takes place in space starring Alex, a little boy who is looking for new adventures. Alex wanders in space exploring new places and planets. During his visit to a distant galaxy, Alex faces a major challenge: Five planets have been destroyed by the dominant enemy of space, DragonDrop, and the remaining extraterrestrial aliens are begging for help. *Will Alex be able to help the affected residents and deal with DragonDrop?*

The purpose is for Alex to try to carry out the missions assigned to him by the alien inhabitants of each planet. His only defense against the enemy is programming, as DragonDrop challenges him in several Java programming battles. The planets' leaders are training Alex in OOP skills in exchange for helping them rebuild what the enemy has destroyed on their planets (*scenario)*.

3.2.4 Content and information

The *educational content* of the game covers the following concepts/constructs: classes; object construction; calling methods with or without parameters; inheritance; polymorphism; method overriding; super keyword; substitution principle and the instanceof operator; access modifiers; and finally the collection ArrayList along with its basic methods.

The game consists of five thematic levels of full functionality, as shown in Figure 2, and a level for learning the basic programming rules (*explanation of rules / theory*). Each level represents a planet that the trainee must visit. Each planet corresponds to a specialized theory course that the player must understand in order to respond successfully to the activities of the game.

In the *first level* the concept of "method" is presented. The student has to understand what the signature of a method is, how a method is called, its usefulness and what elements it may enclose in its body. The *second level* explains the concepts of "class", "constructor", and "object creation". The *third level* contains "method calling with one or more parameters", while the *fourth level* presents the concepts of "inheritance", "polymorphism" and "overriding", as well as the "super" keyword, the substitution principle, the "instanceof" operator and printing on the console. Reference is also made to "access modifiers". In *level five*, the "ArrayList" collection with or without a data type is explained and the functions of adding, deleting and modifying elements are presented. The *sixth level* contains theory slides with basic programming concepts, such as: if/else control structure, while and for loop structures, assignment, equality, some keywords, variable type and naming. Table 1 summarizes the educational content.

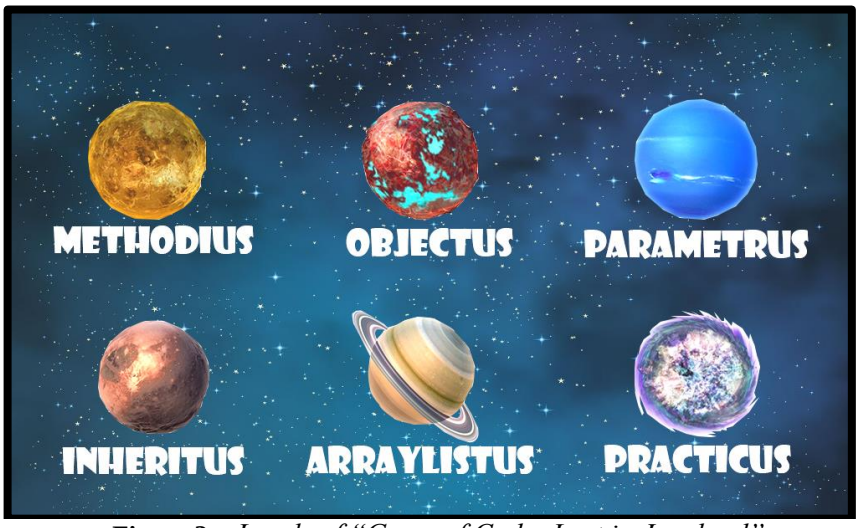


Figure 2. *Levels of "Game of Code: Lost in Javaland"*

Table 1. *The educational content of "Game of Code:Lost in Javaland"*

Level	Planet	Educational content
1	Methodius	Method call Method signature and body
2	Objectus	Class Object instantiation Constructor
3	Parametrus	Method call with parameters
4	Inheritus	Inheritance Overriding Super keyword Polymorphism Substitute principle instanceof operator Modifiers System.out.println()
5	Arraylistus	ArrayList Methods add, remove, set
6	Practicus	if/else for/while Type of Variables Keywords

Each level starts and ends with appropriate messages in Greek language (*level start/end messages*, *spoken languages*) where the player can be informed about the status of the game. There are also instant *feedback* messages in case of a game over as well as *reward* messages when the player achieves a goal. The player has to move around the main *character* named Alex and find the chief of the planet. The chief asks for Alex's help to save the planet and defeat the enemies. In order to achieve that, the chief must teach Alex a new OOP concept in Java (*knowledge transfer*). The inventory holds the collected materials and it is accessible throughout the game (*access to data*) so that the player is aware of the items s/he carries at any point of time.

3.2.5 Mechanics

At the start of the game the player can navigate to the special *rules* section. In that section the player can find the main instruction rules and the keyboard interactions (*navigation system)*. The trainee can navigate using the keyboard but also the mouse is useful as well.

In every planet-level there is a leader who assigns a mission to the player. In order for the mission to be successful the player has to learn new programming concepts and/or constructs. The chief is responsible for giving valuable information about programming and teaching the player in a step by step manner how to program in real life (*system knowledge transmission)*.

In the first level the player comes in contact with Methodis, the chief of the *planet Methodius*. The leader gives Alex his first lesson on programming. Immediately after the first lesson the leader assigns Alex his first mission, which is to repair the destroyed roof of his house. In order to achieve that, Alex has to pick up all the materials (bricks, woods) by calling the appropriate method, while avoiding the comets that are falling from the sky.

In level 2, Alex travels to *planet Objectus*. His mission this time is to fix the broken bridge so that he reaches the house on the other side of the river. All the materials he will use are depicted with the form of wooden signs which represent the class concept in OOP. What Alex has to do is to create new instances of the material classes. When Alex manages to get into the house he fights against the vicious enemy DragonDrop. DragonDrop invites Alex to a knowledge contest where he has to answer correctly to a series of programming questions.

Planet Paremetrus is the third level. In this level Alex learns about method calling with parameters. His mission is to fix the well in order to restore the water supply of the planet (Figure 3). In order to achieve this, he has to go through a number of tasks such as finding a missing key, creating his own materials, or opening secret doors.

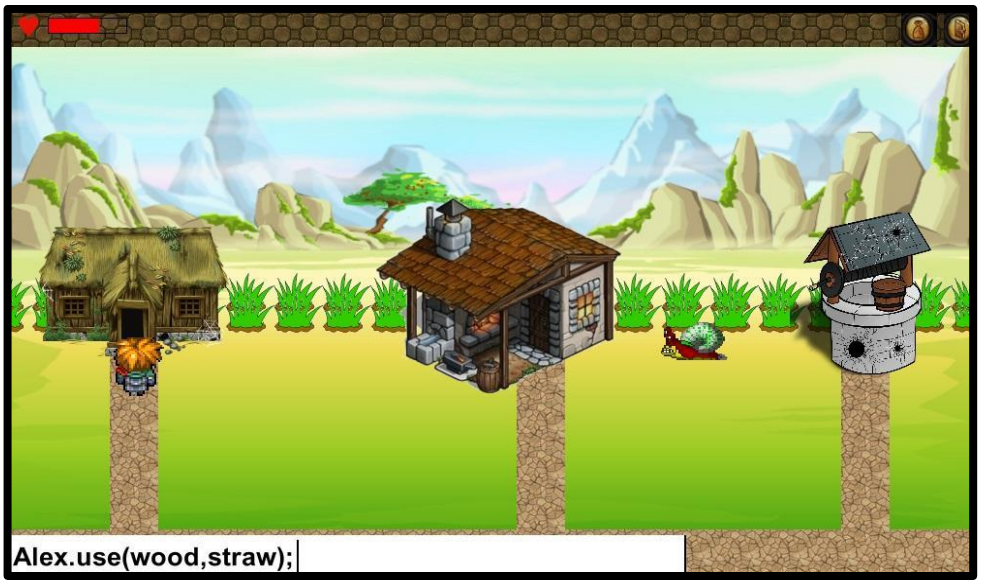


Figure 3. *Level 3, restoring the broken hut.*

Level 4 is quite different from what the player has seen so far. *Planet Inheritus* is a maze where Alex has to go through quizzes regarding the concept of inheritance, collect materials and avoid dangerous enemies such as hostile ants and venomous snakes (Figure 4). At the end of the maze Alex has to deal with DragonDrop once more.

In level 5, *planet Arraylistus*, Alex has to load the proper materials on the boat and transport them to the opposite side of the river. There, he has to unload the cargo and place it in the box. In order to accomplish this he must learn about the ArrayList object collection and answer correctly to quizzes and exercises.

Planet Practicus is a non-interactive level where the trainee goes through a bunch of slides regarding the basics of programming with Java. The theory slides present basic Java syntax through the use of easy to understand examples and blocks of code (Figure 5).

Figure 4. *Level 4, the maze.*

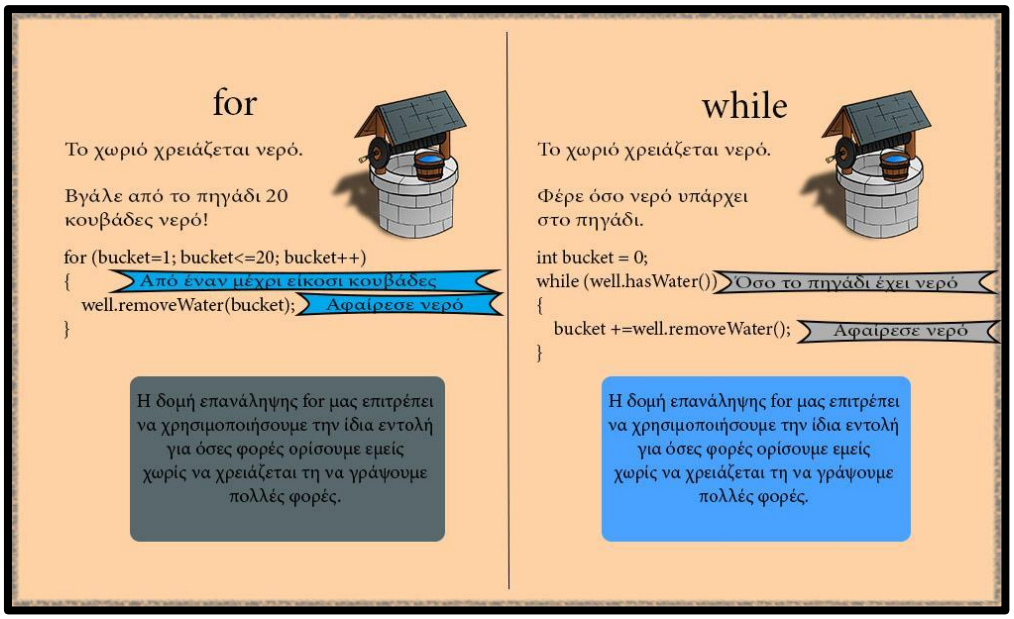


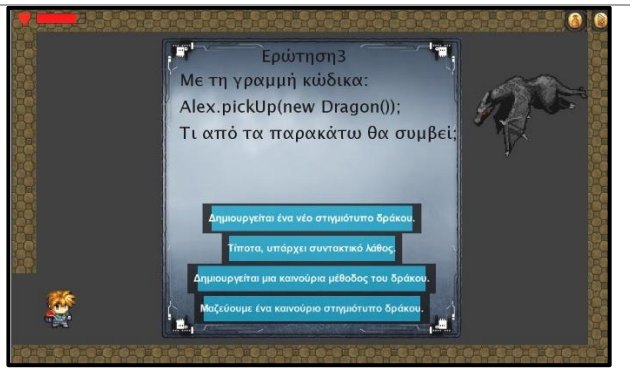
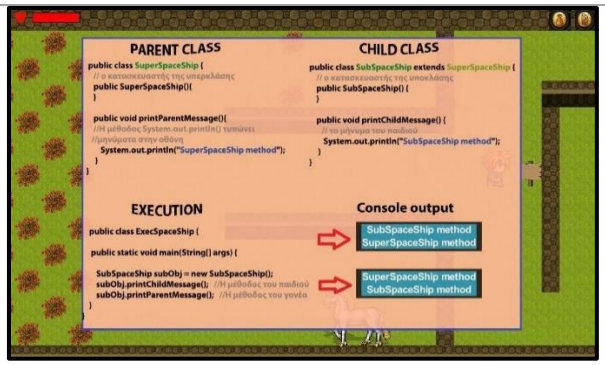
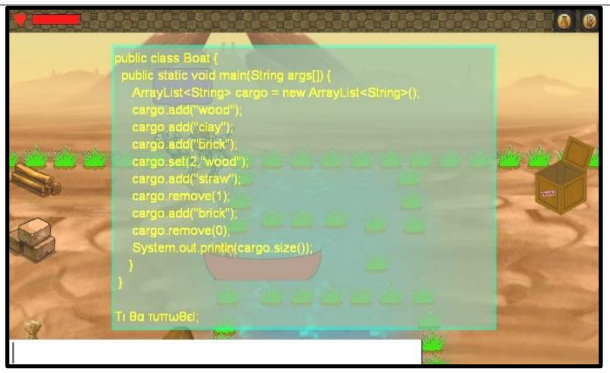
Figure 5. *Level 6, planet Practicus.*

The player has six lives. If Alex loses all his lives, the game is over (*win/lose system)*. The player's life is reduced by:

- Incorrect code commands in the single-line-editor.
- Wrong answers to quizzes.
- Collisions with the enemy.

In addition, there is the "sudden death". The player has only one chance to respond correctly or s/he faces immediate game over. In Table 2 some of the *activities & actions* the trainee should go through are presented.

Table 2. *Activities and actions*

Activity	Description	Screenshot
Write code in a single-line processor	The player writes the required command and presses enter to verify its correctness.	
Multiple choice quiz questions	The player must choose the correct answer to each question.	
"What will be printed?" ***Choosing the correct answer.***	The player must comprehend the code and choose one of the two possible answers.	
What will be printed?" ***Giving the answer to the single-line editor.***	The player must comprehend the code, write the answer to the single-line editor and press enter.	

"Fill in the blanks"	The player must fill in the blanks to complete the block of code.	

All *levels* are characterized by an increasing degree of *difficulty* with the fifth level being the most difficult. The *action verbs* that could describe the game are *learning, programming, avoiding, defeating, and helping.* A *reward system* is not included and the game flow is linear, meaning that the trainees do not have any control over the story outcome since there is always one ending (*decision making*). The *impact on the player* can be assessed in future research, namely investigating the impact on students' knowledge in a pre/post-test design.

3.2.6 Aesthetics and graphics

The story unfolds in space shrouded by mystery. The *environment* is in a retro 80's game style. Thematic *sounds* and *sound effects* are framing the game not only in the start screen but also in each level. Every level *starts and ends* with appropriate messages and the *dialogues* between the characters are quite amusing.

The *movement* of the characters uses the XY axis system at a normal smooth speed. The *graphics* that have been integrated into the game are 2D while the *optical effects* give a feeling of a 3D environment. All graphics are free for non-commercial use and they have been collected from various sources listed in the Appendix.

3.2.7 Coherence and cohesiveness

"Game of Code: Lost in Javaland" is a fantasy role-playing game that integrates educational content on OOP concepts. The six components of the SGDA framework described in the previous sections compose the main goal of the game (*consistency of individual elements*) which is the transmission of knowledge in basic programming concepts.

The educational content, the mechanics and the imaginary world of the game are directly connected to each other in order to create a balanced system of entertainment and education. Finally, the single-line editor that is used by the player for writing a single line of code or answering in "What will be printed?" activities, as well as the variety of tasks and exercises that are directly connected to the scenario of the game, play an important role in transmitting knowledge (*content - engineering - fantasy world*).

4 Implementation of "Game of Code: Lost in Javaland"

Greenfoot is an open source, multi-platform (Windows, MacOSX, Ubuntu) Integrated Development Environment (IDE) that is based on the well-known educational programming environment BlueJ [25]. The goal of Greenfoot is to support the teaching and learning of OOP through the development of simple games, using Java as a programming language [5].

Greenfoot comes with several ready-to-use classes concerning the graphical interface and the programming behavior of the characters such as movement and collisions. The levels and characters of the game are created as subclasses of the existing World (a two-

dimensional grid of cells) and Actor classes (categories of objects, which exist in the World and have an appearance) respectively, which are provided in each new project created within the environment. In addition, a number of classes can be directly imported through the Greenfoot environment. The classes of the Greenfoot API along with those that can be directly imported are presented in Table 3.

Table 3. *Greenfoot API and imported classes*

Class	Description
Greenfoot API	https://www.greenfoot.org/files/javadoc/
World	Includes methods for creating and managing the game world (levels).
Actor	Includes methods for managing game characters and props.
GreenfootImage	Includes methods for representing and managing images, which can be used either for game levels or for characters, animation and effects.
GreenfootSound	Includes methods for manipulating audio playback.
Greenfoot	Utility class that provides methods to control the simulation and interact with the system.
MouseInfo	Manipulate game events.
UserInfo	The UserInfo class can be used to store data permanently on a server, and to share this data between different users, when the scenario runs on the Greenfoot web site.
Color	A representation of a Color.
Font	A representation of a Font.
Imported classes	*Can be imported through the choice Import Class of the menu Edit*
Animal	This is the base class for all animals. In addition to the standard Actor methods, it provides methods to eat other animals and check for the edge of the world.
Counter	A Counter class allows you to display a numerical value on screen, such as the remaining lives of the player.
GifImage	Set the image of the actor. If the image is a normal picture, it will be displayed as normal. If it's an animated GIF file then it will be displayed as an animated actor.
Label	A Label class allows you to display a textual value on screen.
Map	A class that provides a Google map as an image.
ScoreBoard	An actor class that can display a scoreboard, using Greenfoot's UserInfo class.
SimpleTimer	A simple timer class that allows you to keep track of how much time has passed between events.
SmoothMover	A variation of an actor that maintains precise location (using doubles for the coordinates instead of integers). It also maintains a current movement in form of a movement vector.

Running Greenfoot projects on NetBeans, as well as using it in conjunction with the Microsoft Kinect sensor, the PicoBoard board with several sensors, and the programmable Finch robot that also has a number of sensors is supported (https://www.greenfoot.org/doc).

The Greenfoot API and the classes than can be imported through the environment were heavily utilized for the development of “Game of Code: Lost in Javaland”. Third-party classes from books [26][27] and scenarios from the Greenfoot gallery were utilized (Table 4), while some new classes were also defined.

Table 4. *Game mechanics and Greenfoot classes in "Game of code: lost in JavaLand"*

Mechanic	Class	Functionality	Source
Maze	ScrollingActor ScrollingEnemy ScrollingObstacle TiledWorldPathfinding Tile Point	Set up the environment for a scrolling world.	[26]
Collision	HiddenSprite	Check for collisions by creating a transparent layer on top of the objects.	[26]
Animation	SpriteSheet	Image animation	https://www.greenfoot.org/scenarios/19329
Player's input	TextField Debugger	Single-line code editor	https://www.greenfoot.org/scenarios/7435
Feedback messages	TextPanel	Creates the dialogues, welcome and win/lose messages.	[27]

Based on the experience of using Greenfoot for implementing "Game of code: lost in Javaland" it was affirmed to us that Greenfoot has a very active and devoted community, a fact that is denoted by the numerous scenarios that are shared in the Greenfoot Gallery (https://www.greenfoot.org/scenarios); the lively discussions between the members of the Greenfoot community (https://www.greenfoot.org/topics); and the tutorials and videos that are freely available (https://www.greenfoot.org/doc). Besides the material that is available in the Greenfoot site, a variety of ready-made user interface (UI) and graphics handling classes can be found on the web, as well as in books about Greenfoot. In Table 4 a list of third-party classes that were used in the development of "Game of code: lost in Javaland" are presented. Finally, Greenfoot has a clear and minimal UI and offers visualizations, such as simplified class diagrams with the hierarchy of the World and Actor classes; allows creating objects by interacting with the classes and calling methods through direct manipulation of the depicted objects in order to test aspects of the game during its development without writing code.

On the other hand, Greenfoot is designed to teach programming by creating simple simulations and games. As a result, there are some difficulties and limitations in the development of a game as it escalates. Specifically, in order to implement the effect of movement in a character (animation), the programmer must use either many different images that are alternated or sprite sheet type images. In both cases, several lines of code are needed in order to handle any kind of character animation. Moreover, in Greenfoot the game loop is not handled by the programmer, but instead, in every cycle of the game each actor that has been created in the game executes the code included in its act method. Taking into account that the developer is responsible for discarding and destroying any unused class instances at the end of each level, it is of vital importance for the programmer to handle this issue carefully. Otherwise, it is very likely for a memory leak to happen, causing Greenfoot to crash unexpectedly. Finally, 3D graphics are not supported by Greenfoot, but this is not actually a drawback in the case of SGs.

5 A Survey on the use of Greenfoot

Greenfoot has been successfully used for implementing SGs, such as the one presented in the previous sections. Besides this fact, we aimed to investigate the perceptions of postgraduate students attending a Serious Games Programming course on the usage of

Greenfoot both as a tool for introducing novices to game programming and as a tool for implementing SGs. In the subsections that follow information on the context of the study, the research questions, the participants, data collection, as well as the results are presented.

5.1 Methods

The study was carried out in the context of a "Serious Games Programming" course taught in a Master Programme on Applied Informatics. In the context of this course, Greenfoot is used for an introduction to game programming, after a brief introduction to SGs and SGs design using specialized design frameworks. Greenfoot is used in three out of the twelve three-hour lectures of the course for showcasing basic game mechanics. Moreover, students have to implement their first game using Greenfoot. The part of the course that is based on Greenfoot remains unchanged since 2014 when the course was offered for the first time and is described in [1]. In a previous study aiming at evaluating the overall design of the course, 93% of the students stated that Greenfoot helped them much or very much in comprehending fundamental game programming concepts, while all of them stated that Greenfoot helped them to implement a fully functional game in a short period of time [1]. This motivated the study presented in this article, which was carried out in the context of the same course but utilized a different dataset.

5.1.1 Participants

The thirty-two participants of this study had attended and successfully completed the aforementioned Master course on Serious Games Programming. In the context of this course, various tools and languages are used for introducing students to (serious) game programming. Specifically, Greenfoot is used as a tool for a brief introduction to game programming and simple simulations (a simplified simulation of the solar system is presented in the lectures), as well as a tool used by students for implementing their first game or simulation. The first game can be either an entertainment game or an SG. This is followed by the presentation of even more game mechanics using the programming language C# and a set of ready-made classes, and finally a game engine. Game engines that have been used in the course are GameMaker and Unity. Students can implement their final project for the course, namely an SG of their choice, using either one of the tools utilized in the course or any other tool. Interested readers can find more information on the use of Greenfoot and the overall design of the course in [1].

Consequently, all the participants had experience in implementing games using Greenfoot, ready-made classes in C# and a game engine (either GameMaker or Unity); they had implemented a (serious) game or simulation using Greenfoot; and finally, they had implemented an SG utilizing a tool of their choice as their final project. This means that the participants had good knowledge on all the issues covered in the questionnaire.

5.1.2 Design and materials

A questionnaire designed specifically for the evaluation of Greenfoot as a tool for introducing novices to game programming and a tool for programming SGs was prepared as a Google form. The questionnaire included the following questions:

Q1. What is the most effective way for an introduction to game programming?

- Using an educational programming environment that supports game programming (i.e. Greenfoot)
- Using a game engine (i.e. GameMaker)
- Using a conventional programming language
- Other (please specify)…….

Q2. Is Greenfoot appropriate for an introduction to game programming?

Q3. Is there sufficient material for an introduction to game programming with Greenfoot?

Q4. Is Greenfoot user friendly for programming entertainment games?

Q5. Is Greenfoot user friendly for programming serious games?
Q6. Is Greenfoot user friendly for programming simulations?
Q7. Is Greenfoot effective for programming entertainment games?
Q8. Is Greenfoot effective for programming serious games?
Q9. Is Greenfoot effective for programming simulations?
Q10. Would you use Greenfoot for programming…

- Entertainment games (Yes/No)
- Serious games (Yes/No)
- Simulations (Yes/No)

Q11. In what degree does the Greenfoot API and the classes that can be imported support sufficiently the development of …

- Smooth animation
- Navigating characters through the keyboard
- Depicting score, text, and remaining trials
- Score board
- Dialogue between the players and the system
- Updating and depicting an inventory
- Content frame (welcome & game over screen, educational material screen etc)
- Defining sequence of levels & background music
- Scrolling games
- Embedding video

The first question (Q1) was a multiple choice question offering the ability to enter free text as well. Questions Q2-Q9 and Q11 were closed-type questions utilizing a 5-point Likert scale (1=not at all, 2=a little, 3=adequately, 4=much, 5=very much), while question Q10 was a yes/no question. Question Q1 aimed to investigate students' perceptions on the most effective way for introducing novices to game programming, while questions Q2 and Q3 aimed to investigate the appropriateness of Greenfoot as a tool for an introduction to game programming. Questions Q4 – Q9 aimed to investigate students' perceptions on the degree that Greenfoot is a user friendly and effective tool for programming entertainment games, SGs and simulations, while Q10 investigated their intentions on using Greenfoot in the future for this purpose. Finally, question Q11 investigated the support provided by Greenfoot and the imported classes for implementing fundamental game mechanics.

5.1.3 Procedure

The link of the Google form was sent through an email to students and graduates that had attended the course the last four years. This means that all the participants had successfully completed the course or even the Master Programme on Applied Informatics prior answering the questionnaire. Thirty-two students answered anonymously the questionnaire, giving consent for collecting and analyzing their responses for the aims of the study.

Descriptive statistics were used for analyzing the results. Specifically, for question Q1 the frequencies and percentages for each possible response were calculated, while for all the other questions the mean, standard deviation and median were calculated.

5.2 Results of the study

5.2.1 Using Greenfoot as an introduction to game programming

In this section the results for questions Q1-Q3 are presented. As shown in Table 5, nearly 70% of the participants consider that using an educational programming environment that supports game programming is the most effective way for an introduction to game programming (Q1). We must note that the environment of Greenfoot that was utilized in the course uses Java, a programming language that students had already experience in.

Surprisingly, only 16% students considered the use of a game engine as the most effective way for an introduction to game programming, while the percentage was even smaller for those who believe that the introduction to game programming should be better realized using a conventional programming language. Finally, one student stated that the best way for introducing novices to game programming is with the combination of an educational programming environment and a game engine.

Table 5. *Preferred approach for an introduction to game programming.*

Q1. What is the most effective way for an introduction to game programming?	**Frequency**	**Percentage**
• Using an educational programming environment that supports game programming (for example, Greenfoot)	22	69%
• Using a game engine (for example, GameMaker)	5	16%
• Using a conventional programming language	4	12%
• Other (please specify)…….	1	3%

The results of Q1 were in alignment with those of Q2 (Table 6) that aimed to investigate whether Greenfoot is considered appropriate for an introduction to game programming. In the context of Q2, 72% of the participants stated that Greenfoot is appropriate for an introduction to game programming (median=4).

Finally, nearly the same percentage of participants (69%) considered that the material that is available for an introduction to game programming with Greenfoot (Q3) is sufficient (median=4).

Table 6. *Appropriateness of Greenfoot for an introduction to game programming.*

Question	**Mean**	**Std. Dev.**	**Median**
Q2. Is Greenfoot appropriate for an introduction to game programming?	3.9	1	4
Q3. Is there sufficient material for an introduction to game programming with Greenfoot?	3.8	0.9	4

5.2.2 Using Greenfoot as a tool for programming games

In Table 7, questions Q4-Q9 that aimed to investigate participants' perceptions on the user-friendliness and effectiveness of Greenfoot for programming various types of games are presented along with the results.

The participants perceive Greenfoot to be adequately effective as a tool for programming entertainment or SGs and simulations (median=3). On the other hand, the participants are quite positive regarding the user-friendliness of Greenfoot as a tool for programming SGs (median=3.5) and surprisingly they are even more positive when it comes to entertainment games (median=4).

Table 7. *Appropriateness of Greenfoot for game programming.*

Question	**Mean**	**Std. Dev.**	**Median**
Q4. Is Greenfoot user friendly for programming entertainment games?	3.8	0.9	4
Q5. Is Greenfoot user friendly for programming serious games?	3.6	1	3.5
Q6. Is Greenfoot user friendly for programming simulations?	3.4	1.1	3
Q7. Is Greenfoot effective for programming entertainment games?	3.2	0.9	3
Q8.Is Greenfoot effective for programming serious games?	3.1	0.9	3
Q9. Is Greenfoot effective for programming simulations?	2.9	0.9	3

Besides the aforementioned results, nearly 70% of the participants stated that, based on their experience, they would use Greenfoot for programming entertainment and serious games (Q10), as shown in Table 8. Fewer participants (56%) stated that they would use Greenfoot for programming simulations.

Table 8. *Participants' intentions on using Greenfoot for programming games and simulations.*

Q10. Would you use Greenfoot for programming…	**Yes Percentage (frequency)**	**No Percentage (frequency)**
• Entertainment games	69% (22)	31% (10)
• Serious games	69% (22)	31% (10)
• Simulations	56% (18)	44% (14)

In the last question (Q11) the participants were asked to define at what degree Greenfoot supports programming various fundamental game mechanics through its API and the imported classes without the need to write extensive code. As shown in Table 9, the participants believe that Greenfoot supports adequately all the game mechanics (median=3), while defining the sequence of levels and background music is supported at a higher degree (median=4). It is clear, however, that Greenfoot could be further improved in order to be even more effective as a tool for implementing SGs.

Table 9. *Greenfoot support for fundamental game mechanics.*

Q11. Greenfoot API & imported classes support for...	**Mean**	**Std. Dev.**	**Median**
Smooth animation	3.5	0.9	3
Navigating characters through the keyboard	3.4	1	3
Depicting score, text, and remaining trials	3.4	1	3
Score board	3.4	0.9	3
Dialogue between the players and the system	2.8	1.1	3
Updating and depicting an inventory	2.8	1	3
Content frame (welcome & game over screen, educational material screen etc)	3.3	1	3
Defining sequence of levels & background music	3.6	1	4
Scrolling games	2.9	1.1	3
Embedding video	2.6	1	3

6 Discussion

Greenfoot is an educational programming environment that has been extensively used to teach OOP in Java through the development of simple games and simulations to a wide range of ages, such as high school [18] and higher education students [19]. However, the literature on utilizing Greenfoot for teaching basic game programming concepts [1][20], and even more for implementing SGs is rather limited. The case study presented provides clear indication that Greenfoot can be utilized for implementing SGs. However, a formal evaluation of "Game of Code: Lost in Javaland" by end users is necessary in order to strengthen the conclusions regarding the appropriateness of Greenfoot as a tool for developing SGs. The participants of the survey that was carried out with the aim of investigating the potential of Greenfoot as a tool for an introduction to game programming and as a tool for developing SGs were also positive. It is clear that the

number of participants in the survey was small and the results cannot be generalized. However, they had all knowledge on SGs and various development tools and this provides clear indications that Greenfoot can be utilized in SGs programming education and development.

7 Conclusions

The study presented in this article aimed to investigate whether the educational programming environment of Greenfoot, which has been proved to be effective for an introduction to OOP with Java, could also be utilized as a tool for an introduction to (serious) game programming and a tool for implementing SGs. The results of the survey that investigated the perceptions of graduates who had used Greenfoot in the context of a Serious Games Programming course, show that Greenfoot seems to be effective both for an introduction to SGs programming (RQ1) and for the development of SGs and simulations (RQ2). The development of the "Game of Code: Lost in Javaland" which incorporates various game mechanics (such as elements of action games, mazes, collisions, inventories), as well as typical elements of SGs (such as quizzes, filling in blanks) has provided further evidence that Greenfoot can be used for implementing SGs (RQ2).

Further research could be carried out in this direction and also in extending the capabilities of the open-source educational programming environment of Greenfoot, so as to enrich it with features of an educational game engine as well.

References

[1] S. Xinogalos, "Programming Serious Games as a Master Course: Feasible or not?", *Simulation & Gaming*, Vol. 49, Issue 1, pp. 8-26, 2018, doi: https://doi.org/10.1177/1046878117747014

[2] E. Christopoulou and S. Xinogalos, « Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices", *International Journal of Serious Games*, Vol. 4, Nr. 4, pp. 21-36, 2017, doi: https://doi.org/10.17083/ijsg.v4i4.194

[3] A. Chaffin, and T. Barnes, "Lessons from a course on serious games research and prototyping", In Proceedings of the Fifth International Conference on the Foundations of Digital Games (FDG' 10), 2010, pp. 32-39, doi: https://doi.org/10.1145/1822348.1822353

[4] J. Kirk, "The making of a gaming-simulation course: A personal tale", *Simulation & Gaming*, 35(1), pp. 85-93, 2014, doi: https://doi.org/10.1177/1046878103261780

[5] M. Kölling, "The Greenfoot Programming Environment" *ACM Trans. Comput. Educ.* 10, 4, Article 14 (November 2010), 21 pages, 2010, doi: https://doi.org/10.1145/1868358.1868361

[6] Greenfoot – Teach & Learn Java Programming. https://www.greenfoot.org/door.

[7] D. Michael, and S. Chen, Serious games: Games that educate, train, and inform. Boston, MA.: Thomson Course Technology, 2006.

[8] K. Mitgutsch and N. Alvarado, "Purposeful by design?: a serious game design assessment framework", *In Proceedings of the International Conference on the Foundations of Digital Games (FDG '12)*. ACM, New York, NY, USA, pp. 121-128, 2012, doi: https://doi.org/10.1145/2282338.2282364

[9] P. Brusilovsky, E. Calabrese, J. Hvorecky, A. Kouchnirenko and P. Miller, "Mini-languages: a way to learn programming principles", *Education and information technologies*, *2*(1), pp. 65-83, 1997, doi: https://doi.org/10.1023/A:1018636507883

[10] N. Ragonis and M. Ben-Ari, "A long-term investigation of the comprehension of OOP concepts by novices", *Computer Science Education*, *15*(3), pp. 203-221, 2005, doi: 10.1080/08993400500224310

[11] M. Guzdial, "Programming environments for novices", *Computer science education research*, pp. 127-154, 2004.

[12] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers", ACM *Computing Surveys (CSUR)*, *37*(2), pp. 83-137, 2005, doi: https://doi.org/10.1145/1089733.1089734

[13] M.A. Miljanovic and J.S. Bradbury, "A Review of Serious Games for Programming", In: Göbel S. et al. (eds) Serious Games. JCSG 2018. Lecture Notes in Computer Science, vol 11243, 2018, Springer, Cham, doi: https://doi.org/10.1007/978-3-030-02762-9_21

[14] I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar and S. Lahmine, "Learning basic programming concepts by creating games with scratch programming environment", *Procedia-Social and Behavioral Sciences*, 191, pp. 1479-1482, 2015, doi: https://doi.org/10.1016/j.sbspro.2015.04.224

[15] J. Maloney, M. Resnick, N. Rusk, B. Silverman and E. Eastmond, "The Scratch Programming Language and Environment", *Trans. Comput. Educ,.10*, 4, Article 16 (November 2010), 15 pages, 2010, doi: https://doi.org/10.1145/1868358.1868363

[16] S. Cooper, W. Dann and R. Pausch, "Teaching objects-first in introductory computer science", *ACM SIGCSE Bulletin*, *35*(1), pp.191–195, 2003. DOI: https://doi.org/10.1145/792548.611966

[17] A. Repenning, D. C. Webb, K. H. Koh, H. Nickerson, S.B. Miller, C. Brand, et al., "Scalable Game Design: A Strategy to Bring Systemic Computer Science Education to Schools through Game Design and Simulation Creation", *Transactions on Computing Education (TOCE)*, vol. *15*, pp. 1-31, 2015, doi: https://doi.org/10.1145/2700517

[18] M. Al-Bow, D. Austin, J. Edgington, R. Fajardo, J. Fishburn, C. Lara, ... and S. Meyer, "Using Greenfoot and games to teach rising 9th and 10th grade novice programmers.", *In Proceedings of the 2008 ACM SIGGRAPH symposium on Video games*, pp. 55-59, 2008, doi: https://doi.org/10.1145/1401843.1401853

[19] R. Gallant and Q.H Mahmoud, "Using Greenfoot and a Moon Scenario to teach Java programming in CS1.", *In Proceedings of the 46th Annual Southeast Regional Conference on XX*, pp. 118-121, 2008, doi: https://doi.org/10.1145/1593105.1593135

[20] K. Villaverde and B. Murphy, "Senior project: Game development using Greenfoot", *Journal of Computing Sciences in Colleges*, 27(4), pp. 159-167, 2012.

[21] D. Rick, J. Ludwig, S. Meyer, C. Rehder and I. Schirmer, "Introduction to business informatics with Greenfoot using the example of airport baggage handling", *In Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pp. 68-69, 2010, doi: https://doi.org/10.1145/1930464.1930474

[22] A. Giannakoulas, M. Maraki, C. Tatoglou and S. Xinogalos, "Development of Educational Games for Primary Education and Investigation of Teachers' Perceptions". ,*Proceedings of the 10th PanHellenic and International Conference "ICT in Education"*, Ioannina 23-25 September 2016. (in Greek)

[23] S. Xinogalos, M.D. Gkaidantzi and M. Tryfou, "Development of Open Source Educational Games with Greenfoot". *Proceedings of the 10th PanHellenic and International Conference "ICT in Education"*, Ioannina 23-25 September 2016. (in Greek)

[24] A. Giannakoulas and S. Xinogalos, "A Review of Educational Games for Teaching Programming to Primary School Students". In Kalogiannakis, M., & Papadakis, S. (Ed.), Handbook of Research on Tools for Teaching Computational Thinking in P-12 Education (pp. 1-30). IGI Global, doi: http://doi:10.4018/978-1-7998-4576-8.ch001

[25] M. Kölling, B. Quig, A. Patterson and J. Rosenberg, "The BlueJ system and its pedagogy", *Computer Science Education*, 13(4), pp. 249-268, 2003, doi: https://doi.org/10.1076/csed.13.4.249.17496

[26] M. Haungs, (2015). Creative Greenfoot. Packt Publishing Ltd.

[27] M. Kölling, Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations. Pearson, 2015.

Appendix

Sources of graphics utilized in "Game of Code: Lost in JavaLand":

http://www.clker.com/clipart-alien-13.html
https://openclipart.org/tag/bridge?p=6&query=bridge
http://clipart-library.com/well-cliparts.html
https://www.deviantart.com/schwarzenacht/art/Cabin-old-380515503
https://metalslug.fandom.com/wiki/Slugs
https://openclipart.org/detail/211602/closed-and-open-perspective-crate
https://simpsons.fandom.com/wiki/740_Evergreen_Terrace_(Brown_House)
https://forums.rpgmakerweb.com/index.php?threads/flying-a-bad-ass-dragon-over-town.33993/
https://www.deviantart.com/potatoesenpai/favourites/72629766/rpg-maker
https://opengameart.org/content/hut
https://docs.microsoft.com/el-gr/archive/blogs/cdnstudents/free-space-art-assets-to-help-you-build-your-game

http://www.clipartpanda.com/clipart_images/canoe-image-vector-clip-art-25822555
https://openclipart.org/detail/170762/wood-signal
https://wiki.themanaworld.org/index.php/User:Fother/Pixel_Art
https://gr.pinterest.com/pin/469007748669725644/
https://www.spriters-resource.com/pc_computer/heroesofmightandmagic2/sheet/29410/
https://opengameart.org/content/lpc-animated-water-and-waterfalls
http://rpgmakervxdownloads.blogspot.com/2010/12/charas-8dparamelody.html
https://icon-icons.com/pack/Bumpy-Planets-Icons/1434