O1 – Methodological Learning Framework

CURRENT TRENDS IN THE DEPLOYMENT OF SERIOUS GAME FOR BUILDING PROGRAMMING SKILLS

Co-funded by the Erasmus+ Programme of the European Union

Document Data

Deliverable: O1/A3 – Current Trends in the Deployment of serious game for building programming skills

Intellectual Output No - Title: O1 – Methodological Learning Framework

Intellectual Output Leader: Univerzza v Ljubjani (Slovenia)

Partners involved: Virtual Campus (Portugal) and University of Thessaly (Greece)

LIST OF TABLES

Table 1 List of programming environments for building programming skills 9
Table 2 List of some serious games for learning programming skills 15

LIST OF PICTURES

Picture 1 Sample Scratch Script 10
Picture 2 Snap! User Interface 11
Picture 3 Alice 3 Code Editor 12
Picture 4 Tynker User Interface 14
Picture 5 Code Combat Gameplay...16
Picture 6 Human Resource Machine Gameplay 17
Picture 7 Lightbot Interface 18
Picture 8 May's Journey Gameplay..19
Picture 9 No Bug's Snack Bar Gameplay 20
Picture 10 Robot ON! Gameplay..21
Picture 11 Educational Pacman Game..22
Picture 12 CMX Game Environment 23

Co-funded by the Erasmus+ Programme of the European Union

TABLE OF CONTENTS

1. Deployment of Serious Games for Learning Programming 5

1.1. Serious Games..5

1.2. Current Trends in Serious Games for Learning Programming......................................7

2. Review of some platforms that can be used for building programming skills.............. 9

2.1. Scratch..10

2.2. Snap! 11

2.3. Alice..12

2.4. Tynker 14

3. Review of some Serious games for Building Programming Skills.............................. 15

3.1. Code Combat..16

3.2. Human Resource Machine..17

3.3. LightBot 18

3.4. May's Journey 19

3.5. No Bug's Snack Bar...20

3.6. Robot ON! 21

3.7. Educational Pacman Game 22

3.7. CMX...23

4. Teaching Programming Skills to Girls 25

4.1. Girls and STEM 25

4.2. Girls Gaming Preferences 26

4.3. Approaches to Teaching Programming to Girls...27

References 29

Co-funded by the Erasmus+ Programme of the European Union

1. DEPLOYMENT OF SERIOUS GAMES FOR LEARNING PROGRAMMING

1.1. Serious Games

Since their apparition on the consumer market less than 50 years ago, computer games have slowly but steadily become a crucial element in our social and cultural environment [1, 2]. The purpose of playing video games already extends beyond their traditional entertainment value and they have been used in many different contexts, like the medical field (e.g. as a tool for repairing social link) or the military sphere (e.g. as a recruiting tool). One of the most common and ever increasing use of video games happens in the field of education, as a learning tool providing significant benefits to all stakeholders in the classroom context and outside of it. It promotes interaction from students with complex, risk-free, skill-demanding practices, it increases motivation and engagement, while at the same time strengthening psychomotor skills, enhancing knowledge retention and decision-making skills, and providing opportunities for repeated practice and immediate feedback [3].

The benefits and usage of playing in the framework of learning predates the invention of video games but the ubiquitous presence of computers in all aspects of our current society has bolstered the apparition and development of a dedicated branch of video games dedicated to be used in an educational setting, called serious games. Although the exact borders between commercial computer games and serious games is sometimes fuzzy and the video game landscape of genre and definitions is always ever changing, some formal descriptions of what serious games are exactly have been put forward. One stipulates that serious games can be defined as games "…that do not have entertainment, enjoyment or fun as their primary purpose" [4]. Another definition can be that serious games are "a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives" [5].

Building on the core principles the design of commercial video games has laid down through decades of iterations, serious games aim at extracting components that make gaming

Co-funded by the Erasmus+ Programme of the European Union

appealing, and combine these with the desired information and knowledge to be transmitted to the user, creating an interactive source for learning that, in turn, motivate each user to extend their own knowledge and deepen their study in a challenging, fun and instant approach [6].

As a study by Freitas et al. [7] concluded, serious games are an advantageous tool in a teacher’s toolbox in the sense that they allow for the creation of content for the purpose of adapted, specific and tailored learning without taking any of the fun out of it.

1.2. Current Trends in Serious Games for Learning Programming

As is it a vast subject that can be tackled in many different fashions, there exist a wide variety of approaches for teaching programming to novice coders. It can be done for example through programming exercises based on visual programming languages. Such languages are especially user friendly as they are both easy to operate (on a drag-and-drop basis) and present content in a very intuitive fashion. Furthermore, object oriented programming is by essence much more easy to introduce and teach in such an environment. Scratch [8, 9], Snap! [10, 11], or Alice 2/3 [12], among others are famous and successful visual programming languages.

Computer games can also be used to teach coding, either by encouraging students to develop and create their very own video games or by allowing them to play serious games whose learning outcomes encompass learning outcomes related to programming.

Students can also create their own basic games through those block-based environments. These drag and drop environments are useful, as they avoid errors regarding syntax and logic [13]. In a study that compared using the Scratch (visual based programming) and Pascal (command line or GUI based programming) coding environment with novice programmers, they found that learners were more motivated to continue their studies in computer sciences when using Scratch [13]. The study also found that the creation of games and stories makes learners more creative and autonomous while learning programming [13].

The idea behind using serious games to teach programming comes from the fact that these, by being more engaging and motivational, allow for students to learn computational thinking and programming skills in entertaining and familiar environments, before transferring those skills to learning a programming language [14].

According to Kazimoglu et al. [14], serious games for learning programming and computational thinking should not only be created for fun, and should also consider a curriculum and skills, making a difference between different programming constructs and encouraging good programming practices (by ways of gamification, like achieving a high score).

Serious games can also be useful for programming by turning unpleasant operations into interesting experiences — Shabanah et al. [15] have created an Algorithm Game Designer aimed at facilitating the creation of algorithm games, in order to improve engagement in algorithm visualization systems. Grivokostopoulou et al. [16] have also developed a game for teaching AI algorithms, based on the Pacman game, so that through algorithm visualizations and animations students could be helped in their learning process, by demonstrating the way an algorithm operates, its functions and how to make proper decisions based on specific parameters.

Following the trends created by commercial video games, serious games have traditionally been deployed in an offline fashion. Technical issues and challenges specific to the world of education have also contributed keeping most of serious games offline. But with the developing widespread adoption of faster internet connection speeds and rising availability of computers in everyone's homes, online game-based platforms like CodeCombat [17], LightBot [18] and Scratch [9] have begun to appear. Combéfis et al. [19] reviewed the main types of online platforms used for teaching programming skills, and have concluded that the following factors can make a serious game more motivating and successful: 1) a feedback and assessment process; 2) aesthetics; 3) collaboration and multiplayer aspects; 4) guidance through the game; 5) no negative consequences; 6) music; 7) a medium-level of challenge to keep the interest of the players. The question of collaborative and competitive games for programming was also highlighted by Miljanovic et al. [20].

Regarding the learning outcomes of Serious Games for Learning Programming, according to an extensive review of 49 serious programming games by Miljanovic et al. [20], most serious games for programming focus on problem solving and fundamental programming concepts, with few games focusing on data structures, development methods and software design.

There are also games that, while they may not teach programming skills, are capable of teaching in an indirect but effective way key computational thinking skills, like abstraction (eg. Tetris), decomposition (eg. Sudoku), algorithm thinking (eg. Puzzle games), evaluation (eg. Memory Games), and generalization (eg. Pattern Games) [21].

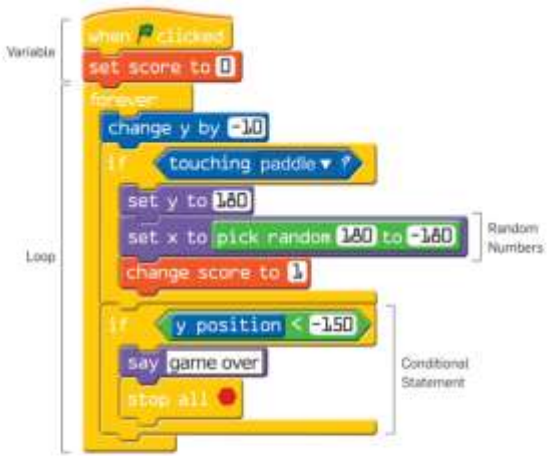
2. REVIEW OF SOME PLATFORMS THAT CAN BE USED FOR BUILDING PROGRAMMING SKILLS

This section will be focused on presenting and comparing different platforms/programming environments that can be used for teaching programming skills to children. The following table is a summary of the different platforms that were considered.

	PLATFORM	TARGET GROUP	LANGUAGE(S)	FREE/PAID	REFERENCES
Scratch	Web-based (but with offline version)	8-16	Visual computer programming language	Free	[8, 9]
Snap!	Web-based (but with offline version)	12-20	Visual computer programming language	Free	[10, 22]
Alice	Windows, Mac OS X, Linux	12-20	Visual computer programming language	Free	[23]
Tynker	Web-based, iOS	7+	Visual computer-programming language, JavaScript, Python	Paid	[12]

Table 1 List of programming environments for building programming skills

2.1. Scratch



```
when [flag] clicked
set score to 0
forever
    change y by -10
    if <touching paddle ?>
        set y to 180
        set x to pick random 180 to -180
        change score to 1
    if <y position < -150>
        say game over
        stop all
```

Picture 1 Sample Scratch Script

Scratch [9] can be used as an introduction to programming, especially among younger students, but also as a way to facilitate the transition to other programming environments, introducing and fostering interest in computer science [22]. One of the benefits of the Scratch website is how it supports a worldwide network of users, hosting a community of programmers which allows users to share projects [22] [24]. Scratch is available in more than 50 languages, including Bulgarian, Croatian, English, Greek, Italian, Portuguese, Slovenian and Turkish. It also has an offline editor that allows for the program to be downloaded to a computer and run without an internet connection. Prior to version 3.0, Scratch was using Flash and is now using HTML 5 and JavaScript.

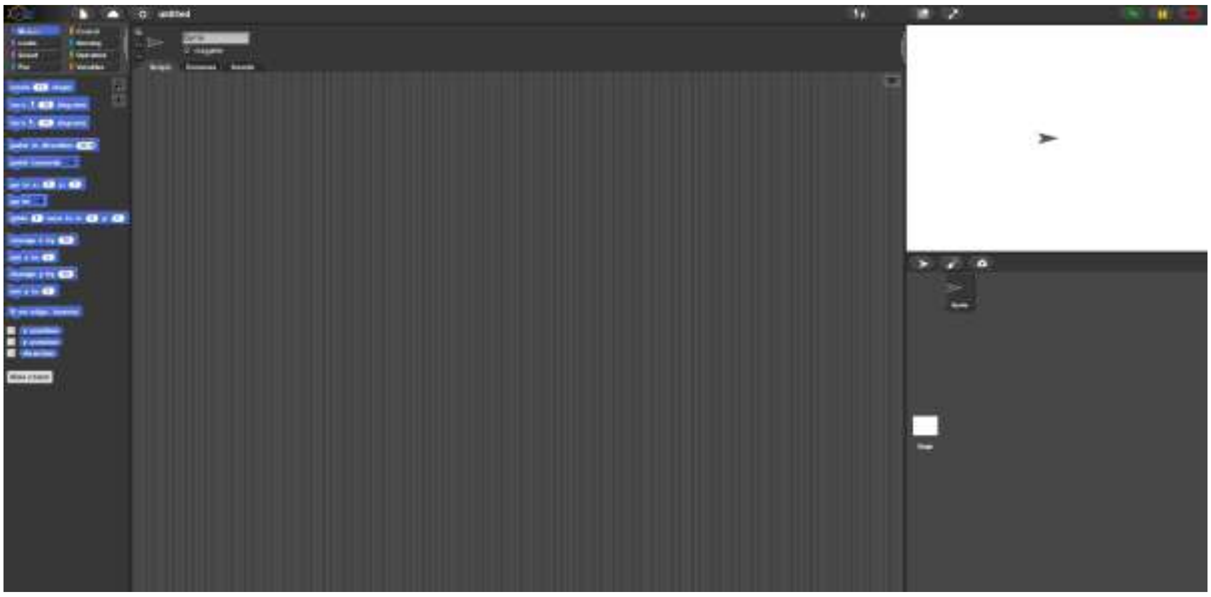
Scratch is a block-based programming environment – these are visual programming languages that work as puzzle pieces, which allow for learners to assemble programs by snapping together the blocks and to receive visual feedback [10]. Another benefit is that it allows for learners to familiarize themselves with the fundamentals of programming without having to worry about syntax. [13]

According to a research study conducted by Meerbaum-Salant et al. [24], most students can achieve a reasonable level of Computer Science concepts through Scratch. However, difficulties arise when teaching specific topics that require a greater level of abstraction. This can, however, be solved if students are accompanied in the process by the teacher. Otherwise, the study concluded that most students will only use it as a tool to create media and not as a way to learn programming [24] as originally planned.

Co-funded by the Erasmus+ Programme of the European Union

2.2. Snap!

Snap! [11] is also a visual block-based programming language whose design is based on Scratch but adding some additional features. Snap!, similarly to Scratch, is web-based, but



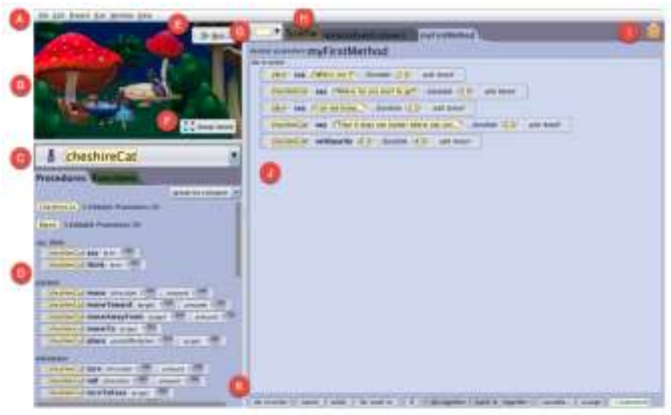
Picture 2 Snap! User Interface

also has the possibility to be run offline through a browser. It is using HTML5 and JavaScript.

Apart from the features of Scratch, Snap! adds first class lists, first class procedures, first class sprites, first class costumes, first class sounds and first class continuations, thus making it more suitable for older audiences and as an introduction to computer science than Scratch [11]. Snap! is available in over 40 languages, including Bulgarian, Croatian, English, Greek, Italian, Portuguese, Slovenian and Turkish.

A study was conducted by Weintrop et al. [10] with high school students using Snap! and Java. The students found the blocks-based approach to be easier than Java – thus, it is in accordance with the view that blocks-based programming (like Scratch and Snap!) is more accessible to novice programmers. According to the study findings, this was due to the fact that blocks are easier to read, due to the visual nature of the blocks that provide cues on how they can be used, they are easier to compose, and serve as memory aids. [10]

2.3. Alice



Picture 3 Alice 3 Code Editor

Alice [23] is a block-based programming environment that aims to motivate learners to program by involving their creativity, through the creation of animations, interactive narratives or simple 3D games. As the other programming environments shown, it removes syntax errors by allowing learners to construct by dragging and dropping code elements [25].

It allows for the creation of virtual worlds with a Virtual World Editor where learners can add 3D objects and add functions and methods to existing 3D objects. Once the Virtual World is created the learner can write code to develop the logic of the game/narrative/animation [26].

Alice is a good programming language for learners with no previous experience, as it allows for them to see as they write their code how the animated programs run [27].

Moreover, a study that focused on Storytelling Alice (a programming environment based on Alice 2) which facilitates the creation of stories by students (with high-level animations, a storytelling gallery, and a story-based tutorial) found that although both Generic Alice and Storytelling Alice were equally successful at teaching programming concepts to girls, with Storytelling Alice, girls were more motivate and increased their time programming, although they found both equally entertaining. Motivation and time spent programming have a positive impact in programming performance – thus making programming more motivating

Co-funded by the Erasmus+ Programme of the European Union

to girl and maximizing the time they spent doing so may be a way to increase women's participation in computer science.

Alice 3, the most recent instalment of the series, is available in 13 languages, including English, Portuguese, Greek, Slovenian, and Bulgarian, although each language might not be available at the same level of completion. Alice 2 is available in English, Spanish, Portuguese and German.

Co-funded by the Erasmus+ Programme of the European Union

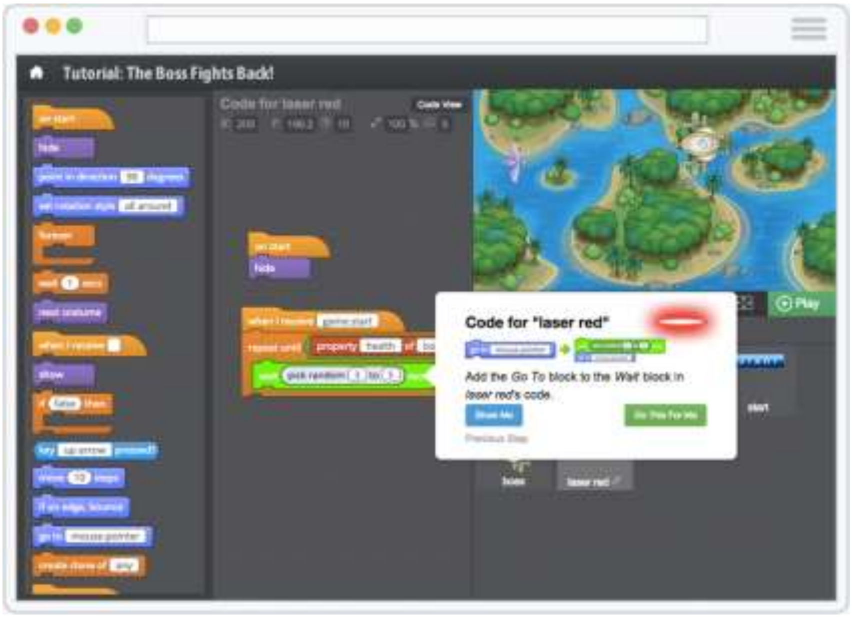
2.4. Tynker

Tynker [12] is an educational programming platform based on a drag-and-drop visual programming language like Scratch. Unlike all the other programming environments presented, this one is a commercial product.

One aspect it adds to Scratch consists in how it approaches coding as a game, in which the users are required to create a program in order to make the screen characters move, interact, and achieve different tasks. It presents problems the players need to solve, making it so that children begin to recognize patterns [28]. Another addition to Scratch is the possibility to see equivalent of each code in JavaScript.

Tynker also provides a built-in tutor to give step-by-step instructions, so that the learner can learn how to apply coding concepts. The learner is also able to visualize the blocks in JavaScript code, enabling them to understand the block in a text-based programming language [12].

Tynker provides over 23 Programming Courses, 11 iPad courses and 2000+ coding activities. It provides individual plans and family plans (for up to 4 members), that can be billed quarterly (20$/month), yearly (10$/month) or onetime payment (240$). It is only available in English.


Tutorial: The Boss Fights Back!

Code for "laser red"

Add the Go To block to the Wait block in laser red's code.

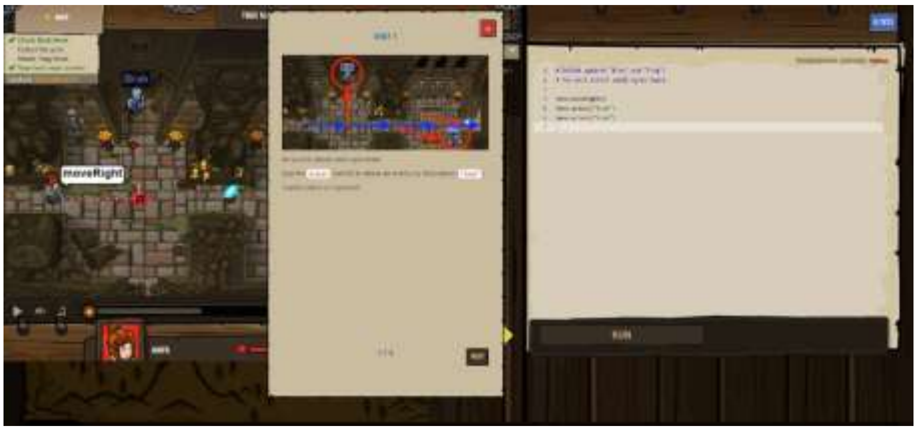
Picture 4 Tynker User Interface

The following table is a summary of the different games that were considered.

GAME	PLATFORM	TARGET GROUP	LANGUAGE(S)	LEARNING UNITS	TYPE OF GAME
Code Combat [17]	Web-based	9+	JavaScript, Python	syntax, methods, parameters, strings, loops and variables, If/else, Boolean logic, relational operators, functions, object properties, event handling, input handling, Arithmetic, counters, while-loops, break, continue, arrays, string comparison, finding min/max, Object literals, remote method, invocation, for-loops, complex functions, drawing, modulo	Commercial Game (*Freemium*)
Human Resource Machine [29, 30]	Windows, Mac OS X, Linux, iOS, Android	9+	Visual Programming Language-based	Input & output and conditionals & iteratives algorithm comparison	Commercial Game (*Paid*) Puzzle Game
Lightbot [29]	iOS, Android, Windows, Mac OS X	9+	Visual Programming Language-based	Conditionals and Iteratives and Recursion Debugging Strategies	Commercial Game *Paid* Puzzle Game
May's Journey [31]	Windows	12-18	Custom Programming Language (inspired from an Object-Oriented Language like Java)	Basic instructions and sequence logic, loops, variables, if statements, comparators and Boolean logic, operations on integers, operations on strings	Research Game Puzzle Game
No Bug's Snack Bar [32]	Web-based	18+	Visual Programming Language	Variable Manipulation, Sequence of Actions, Conditional & Iteratives, Debugging Strategies	Research Game
Robot ON! [33]	Windows	18+	C++	Syntax & Semantics, Variable & Primitive Data Types, Expressions & Assignments, Conditionals & Iteratives, Program Comprehension	Free/Research Game
Educational 'Pacman' Game [16]	Windows	18+	Game for learning AI search Algorithm	Algorithms (Basic textual description of Algorithms and corresponding graphical flowchart along with Pseudocode)	Research Game
CMX [34]	Windows	18+	C	Theory on arrays, outputs of variables after an array's program's execution, syntax of array programs, entry of variable values in an array, calculation of the array's sum, calculation of the maximum value of the array,	Research Game MMORPG

Table 2 List of some serious games for learning programming skills

3.1. Code Combat



Picture 5 Code Combat Gameplay

Code Combat [17] is a web-based adventure game. It has plans for both individual students and classes, providing also resources for teachers to use during classes. It has more than 100 free-to-play and subscriber-only levels. Premium plans are available for $3.99/month or a one-time payment of $39.99. It is available in over 50 languages, including Bulgarian, Croatian, English, Greek, Italian, Portuguese, Slovenian and Turkish.

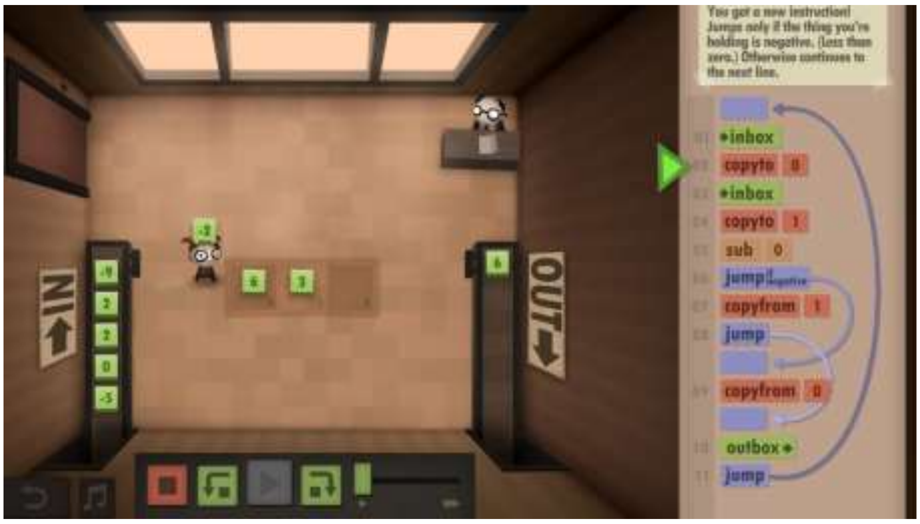
Learners are required to write their own code (either in JavaScript or Python) in order to make the characters move, interact, and achieve different tasks. It is not block-based, requiring learners to write their own code. It provides hints along the way and introduces concepts gradually. The game is divided in 5 different worlds, introducing different concepts as the player progresses through the game: 1) Kithgard Dungeon – syntax, methods, parameters, strings, loops and variables; 2) Backwoods Forest – If/else, Boolean logic, relational operators, functions, object properties, event handling, input handling; 3) Sarven Desert – Arithmetic, counters, while-loops, break, continue, arrays, string comparison, finding min/max; 4) Cloudrip Mountain – Object literals, remote method, invocation, for-loops, complex functions, drawing, modulo; 5) Kelvintaph Glacier – Advanced Techniques.

According to Miljanovic et al. [20], the game's educational content comprises Algorithms & Design (Algorithm Comparison, Problem Solving), as well as Fundamental Programming Concepts (such as Syntax & Semantics, Variables & Primitive Data Types; Expressions & Assignments, Input & Output, Conditionals & Iteratives, Functions & Parameters, and Recursion), Fundamental Data Structures (such as Arrays, Heterogeneous Aggregates, and Abstract Data Types).

3.2. Human Resource Machine

Human Resource Machine [30] is a puzzle game based on a visual programming language. In this game, the player as to automate a task by programming an office worker, being promoted to different levels as they progress through different puzzles [30]. It is a commercial game, being available on Steam for 8.99€.

The player must create a sequence of moves, by dragging and dropping instructions, with new commands being gradually introduced to accomplish more complex operations. The



You got a new instruction! Jumps only if the thing you're holding is negative. (Less than zero.) Otherwise continues to the next line.

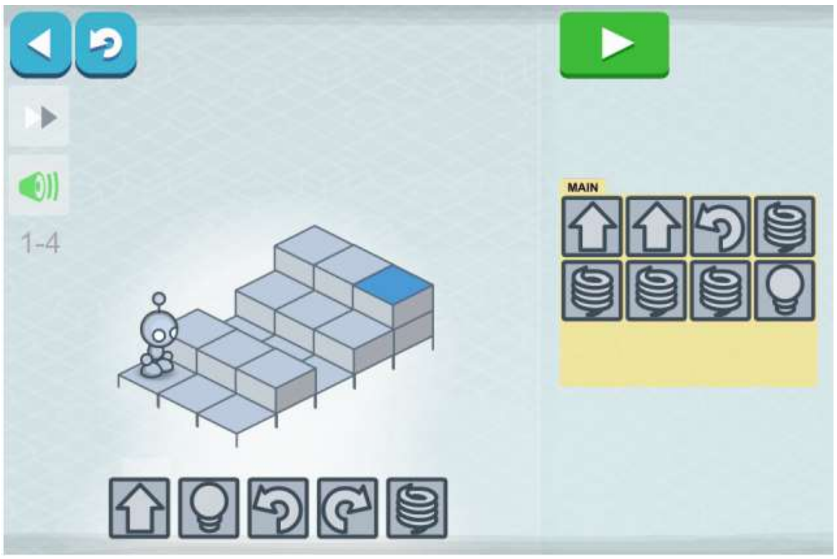
```
inbox
copyto 0
inbox
copyto 1
sub 0
jumpIfnegative
copyfrom 1
jump

copyfrom 0

outbox
jump
```

Picture 6 Human Resource Machine Gameplay

first commands are -> inbox and outbox-> [30].

Through a visual programming language based in blocks, it teaches fundamental programming concepts, like input & output and conditionals & iteratives, as well as teaching algorithm comparison [20]. It is mostly useful for learning how to write assembly-code [30].

3.3. LightBot



Picture 7 Lightbot Interface

LightBot [18] is a puzzle game, with the mechanics being similar to that of Human Resource Machine, requiring programming logical in order to progress through the levels. The player must guide a robot in order to light up tiles and solve levels, featuring 40 levels and 20 challenge stars. The commands used in Lightbot appear as icons.

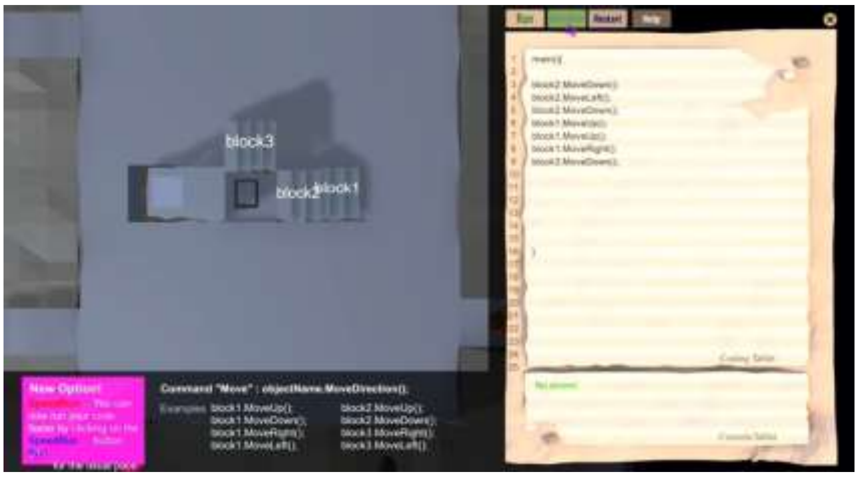
Although in Lightbot, the players do not learn any programming language, “they develop an understanding of sequencing and implementation of algorithms”(p.6), focusing instead on acquiring problem solving skills [20]. Through the game, students learn different fundamental programming concepts, such as Conditionals and Iteratives and Recursion, as well as debugging strategies [20]. As the space for the tiles is limited, the learner must also consider code size [30].

A study conducted by Mathrani et al. [35] using Lightbot with students of a non-university education provider found that students enjoyed playing the games and that this was an effective way of learning programming constructs like functions, procedures, conditionals and recursions. Most participating students agreed that the approach was an effective way for them to understand concepts that would otherwise be more difficult.

Co-funded by the Erasmus+ Programme of the European Union

3.4. May’s Journey

May’s Journey [31] is the only game from this list that is directly targeted toward middle school girls. It is a 3D puzzle game in which the players solve a maze through the means of



```
main(){
block2.MoveDown();
block2.MoveLeft();
block2.MoveDown();
block1.MoveUp();
block1.MoveUp();
block1.MoveRight();
block2.MoveDown();
}
```

```
Command "Move" : objectName.MoveDirection();
Examples: block1.MoveUp();      block2.MoveUp();
          block1.MoveDown();    block2.MoveDown();
          block1.MoveRight();   block3.MoveRight();
          block1.MoveLeft();    block3.MoveLeft();
```

Picture 8 May's Journey Gameplay

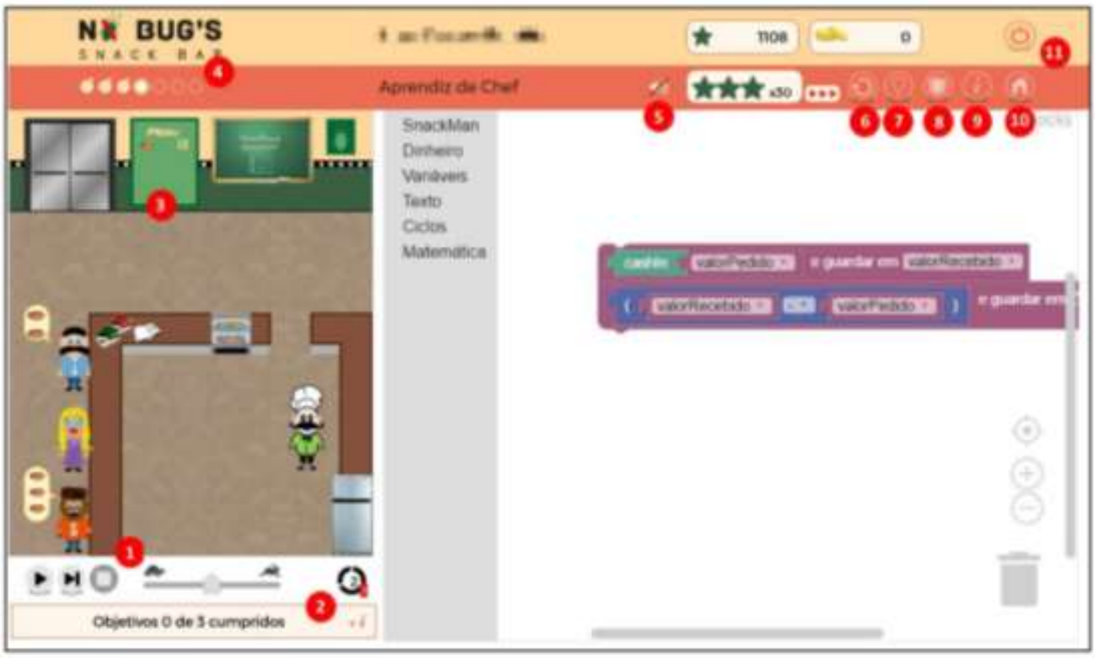
the game’s custom programming language, which is inspired by Java.

The game was designed with middle school girls interests in mind, hoping to attract them to Computer Science by teaching the basics of programming. The developer employs a pseudo-code in order to facilitate the transition between visual programming (drag and drop programs like Scratch) and real programming languages. The teaching content is: basic instructions and sequence logic, loops, variables, if statements, comparators and Boolean logic, operations on integers, operations on strings [31].

There are two phases in the game, an exploration phase with the mechanics of a typical game and a coding phase. In the coding phase, the interface is split in order for the player to type the code, but also get visual feedback of the code. Hints for the code are also provided during the exploration phase. In the story, the hero is a girl who lives in a world that is falling apart and is separated from her friend, needing to fix the game world and solve mysteries. Each part of the mystery is revealed gradually, motivating the players to play more [31]. As pre-teens are the main target group, they made the main character look like a middle school girl, so that the player can project themselves unto the girl. The design also had in mind girls preferences towards design (high lightness, warm colour scheme and less aggressive illustration [31]. The test group seemed also to appreciate the storyline.

3.5. No Bug’s Snack Bar

No Bug’s Snack Bar [32] is a research serious game with a drag and drop block-based approach to make the game independent from learning the syntax of the computer programming language, focusing on problem solving. With a focus on Problem Solving, its learning outcomes are: Variable Manipulation, Sequence of Actions, Conditional & Iterativess, Debugging Strategies.



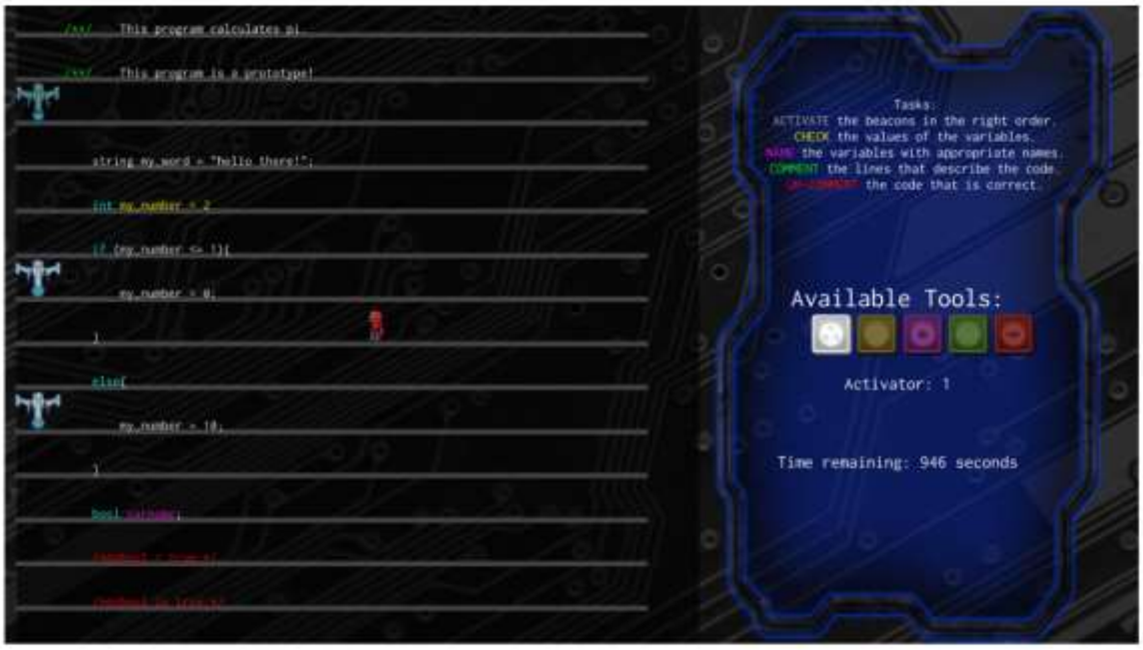
Picture 9 No Bug's Snack Bar Gameplay

In the game, the main character has to work at a Snack Bar, passing through different missions using code. One innovation included in this game is the fact that there’s a tool incorporated for teachers to monitor how students are progressing through the game. There’s also a gamification approach, with the student gaining points as they progress through the games and find ways to better their solutions to the problems presented in order to maximize the points gained.

The study afterwards found that the biggest fault within the game was the absence of a support system, with hints — therefore, the developer incorporated an assistant with hints and an administrative system for the teacher to see who are the learners that need help and to send them personalized hints. The presence of a teacher was, therefore, seen as fundamental.

Co-funded by the Erasmus+ Programme of the European Union

3.6. Robot ON!



```
/**/    This program calculates pi.
/**/    This program is a prototype!

    string my_word = "hello there!";
    int my_number = 2
    if (my_number <= 1){
        my_number = 0;
    }
    else{
        my_number = 10;
    }
    bool [illegible];
    [illegible]
    [illegible]
```

```
Tasks:
ACTIVATE the beacons in the right order.
CHECK the values of the variables.
NAME the variables with appropriate names.
COMMENT the lines that describe the code.
UN-COMMENT the code that is correct.
```

```
Activator: 1

Time remaining: 946 seconds
```

Picture 10 Robot ON! Gameplay

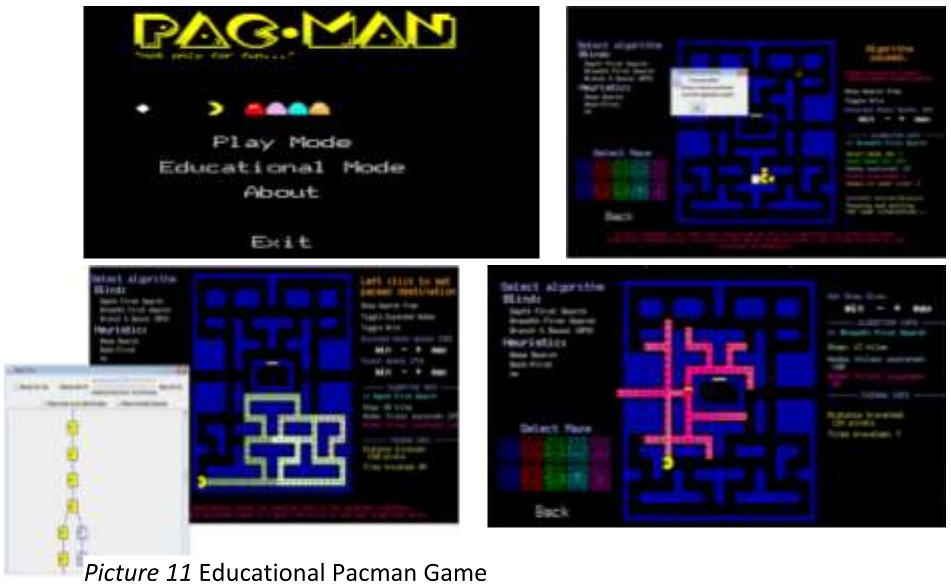
Robot ON! [33] is also a research puzzle-type game aimed at undergraduate students who are learning C++. In this game, the player is a scientist that has to activate a ‘Mech Suit’, doing so through a series of tasks. Once the player finishes a level, he activates a new robot system. It also has a framework in which instructors can create their own levels using other programming languages. The player has different tools, which are colour coded, with the colour on the code corresponding to the different tools. The player is introduced to different tools one at a time, accompanied by tutorials.

Instead of focusing on writing code, this game focus on programming comprehension, in order to teach debugging skills and understand code written by others [33]. As the game progresses, the player is given the following tools: 1) activator tool (to learn control flow); 2) commenter and un-commenter tools (to learn code behaviour); 3) namer tool (to learn variable purpose); and 4) checker tool (to learn data flow) [33].

Co-funded by the Erasmus+ Programme of the European Union

3.7. Educational Pacman Game

Educational Pacman Game [16] is a research serious game designed to teach search algorithms, allowing for students to see how different search algorithms behave and a graphical annotated depiction of them.



Picture 11 Educational Pacman Game

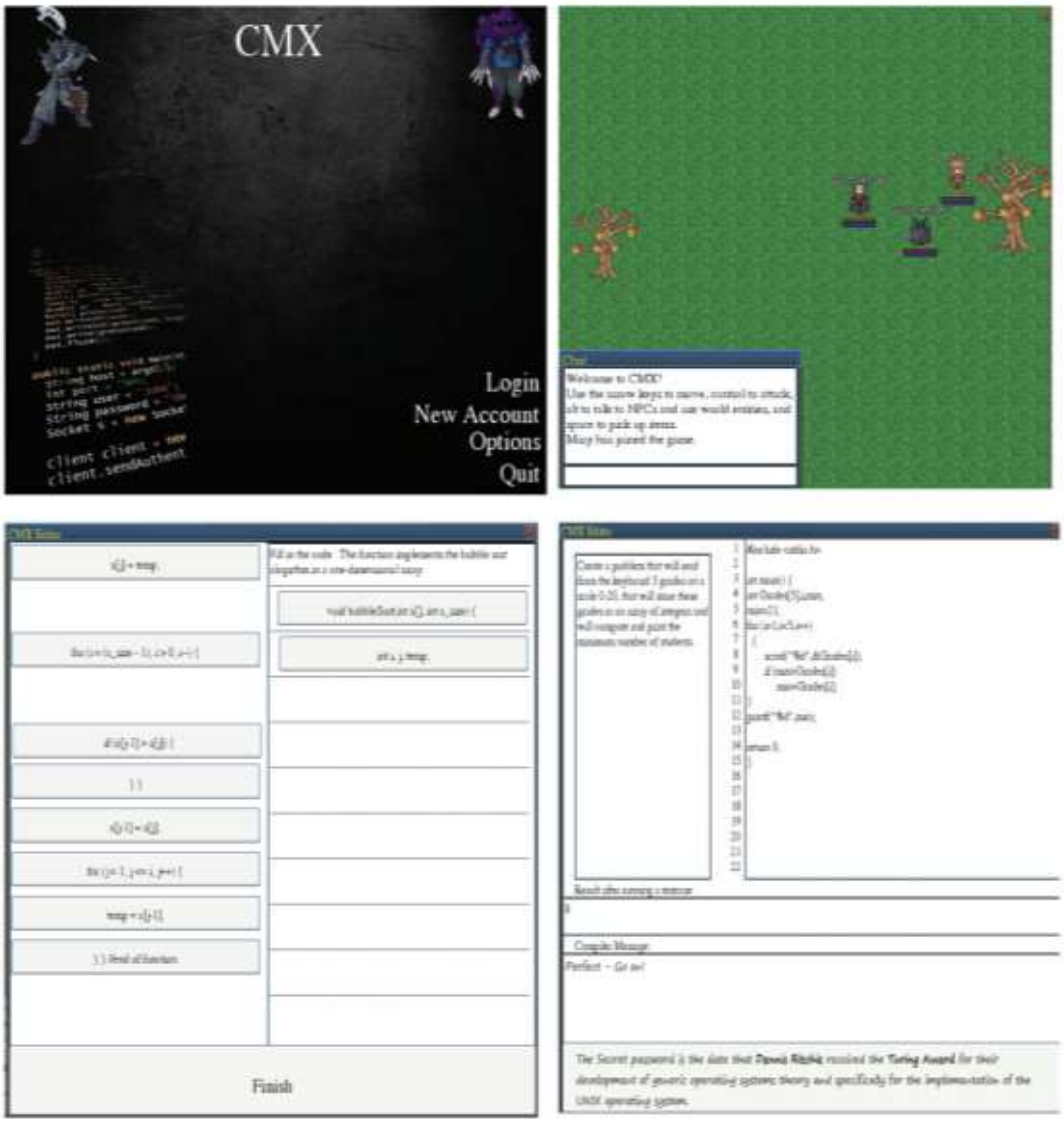
The game has two modes:

1) Educational Mode: the student can read a textual description and the graphical flowchart and pseudocode of an algorithm of his choice. The game also allows for the student to learn the algorithm via visualizations on the different Pacman Mazes [16].

2) Playing Mode: the student has to solve maze levels within different conditions and with a time constraint. These levels are made so that the student has to apply a specific search algorithm to move the Pacman in the maze — the student is asked to, from a random position, reach to a cheery or power-up by moving the character on the specific algorithm [16].

3.7. CMX

CMX [34] is a Massive Multiplayer Online Role Playing Game (MMORPG) for learning programming. In the game there are two teams — crackers and hackers — who compete against each other to find the passwords on a global toxic waste factory. They are assisted by Senseis, that assist them in learning the programming language C. The game includes



Picture 12 CMX Game Environment

three levels of Senseis, with each one holding one password and unlocking another Sensei

Co-funded by the Erasmus+ Programme of the European Union

The students seemed to increase their comprehension level of C — they could not only answer theory questions, but also lay executable programs by dragging and dropping tools, as well as writing programs in C [34].

3. TEACHING PROGRAMMING SKILLS TO GIRLS

4.1. Girls and STEM

The existing imbalance between men and women has been a topic of discussion and raising awareness due to the fact that diversity and more creativity in the workforce by the presence of females is connected with better economic performance for a specific country and also with the idea that everyone should have equal access to the same chances and opportunities [36].

Women and girls participation and uptake in STEM and ICT courses is to this day still significantly poor. This situation may be due to different factors ranging from socio-economic and parental background to attitude towards STEM subjects that, consequently, affect their disposition and engagement with these fields [37, 38]. Any approach to teaching computer science to girls would have to dispel these negative stereotypes and make this field appealing and "cool" to them [39].

The social context of a country also bears influence in career choices and the perpetuation of the existing status-quo, because girls will compare themselves and who they are to stereotypes existing inside the STEM fields and decide that they don't fit in. It is also seen as "natural" for females to not choose computer science fields, which goes against evidence found in different countries where there is a high percentage of women enrolled in these fields [36]. There is a lack of female role models in the public eye to showcase to the remainder of the female population that they are indeed fit for these fields [36].

According to a study conducted by Hosein [37], who analyzed secondary data on the subject, girls who were heavy gamers at 13-14 years old seemed to be more likely to pursue a PTSEM degree. It may be useful to use serious games as a way to motivate girls into computer science and STEM degrees in general [37]. Therefore, it may be pertinent to understand what the gaming preferences of girls are and how this may differ from boys.

4.2. Girls Gaming Preferences

Serious games are games that aren't meant only for entertainment purposes, but also for education purposes: to engage, educate, and motivate students in the learning process. However, the degree in which one wants to play a game varies across gender and the gaming industry has made few attempts at studying female-preferred games [38].

According to the "Gamer Consumer Insights", 46% of gamers across thirteen countries are women, showcasing a growing trend in which women have been becoming more interested in gaming. Likewise, interest in gaming and its connection with gender issues has increased rapidly in recent years. While across all platforms, men tend to prefer strategy, sports, action/adventure, and shooter games, women tend to enjoy more action/adventure, puzzle, strategy and arcade games [40].

Women tend to not be fond of direct competition (conflict or unjust violence settings) and prefer problem resolution [41]. There is also a preference for puzzle games, social games (with a rewarding system), collaborative and exploration games, and virtual life and party games. In regard to adventure games, there is a preference for observing first and playing after the act of observation [38].

It's also important to note the difference in preferences when it comes to platforms: while men prefer to play on the PC or the console (48% and 37%), only 35% and 23% of the women inquired prefer to do so, with this last group playing more mobile games (48%) [40].

4.3. Approaches to Teaching Programming to Girls

Adopting a constructionist approach to games is beneficial for education, because there is not only a focus on providing the game to students, but also providing this group with means and knowledge to develop their own games [42].

A study conducted by Carmichael [39] showcases the beneficial outcomes of combining computer science concepts with video games specifically to a young female target-group. The goal of the one-week course towards twelve girls in grades eight and nine was to teach basic concepts on Computer Science and also to disperse negative stereotypes associated with it. One crucial point to bear in mind when choosing the coding creation software is that educators should conform to a series of requirements: the academic year of the students, familiarity of the instructor(s) and capability of the student in creating a coding project according to the amount of time they will be spending learning the know-how to do it. Some teaching methods used ranged from group activities, brainstorming, reading of relevant articles to interactive demonstrations. An important aspect that Carmichael [39] emphasizes is that during lab time, more than one instructor would've been helpful to, in turn, help all the girls with their doubts about the gaming development phase.

Lastly, Alserri et al. [38] have developed a conceptual model for gender-based engagement in Serious Games, consisting of five elements:

1) Learning Elements: these are the elements that distinguish entertainment games from educational games [38];

2) Female Preferences for digital games: these are the preferences specific to girls, that have to be incorporated into the design in order to motivate and engage them. According to the literature review conducted by the author, these preferences consist in exploration, character customization, storyline, social interaction, collaboration, challenges, fun, control and feedback [38];

3) Flow state theory: some of these elements are also female preference elements. These elements should also be incorporated in order to obtain engagement and motivation: challenges, fun, control, feedback, concentration, clear goals, skill and immersion [38];

Co-funded by the Erasmus+ Programme of the European Union

4) Female game types and genres: according to the authors, these would be fantasy and role-playing games.

5) Social gender factors: parental, peers and teacher influence.

Co-funded by the Erasmus+ Programme of the European Union

REFERENCES

1. Oblinger, D.G. (2004). The next generation of educational engagement. Journal of Interactive Media in Education(8), 1-18. doi:10.5334/2004-8-oblinger
2. Oblinger, D.G. (2006). Simulations, Games, and Learning [Online]. Retrieved from http://net.educause.edu/ir/library/pdf/ELI3004.pdf
3. Tashiro, J.S.D. Dunlap. (2007). The impact of realism on learning engagement in educational games. Paper presented at the Proceedings of the 2007 conference on Future Play, Toronto, Canada
4. Michael, D.S. Chen. (2006). Serious Games: Games That Educate, Train and Inform. Canada: Thomson Course Technology PTR.
5. Zyda, M. (2005). From Visual Simulation to Virtual Reality to Games %J Computer. 38(9), 25-32. doi:10.1109/mc.2005.297
6. Prensky, M. (2003). Digital game-based learning %J Comput. Entertain. 1(1), 21-21. doi:10.1145/950566.950596
7. Freitas, S.S. Jarvis. (2007). Serious games—engaging training solutions: A research and development project for supporting training needs. 38(3), 523-525. doi:10.1111/j.1467-8535.2007.00716.x
8. Resnick, M., et al. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60-68. doi:10.1145/1592761.1592779
9. Scratch - Imagine, Program, Share. Retrieved from https://scratch.mit.edu/
10. Weintrop, D.U. Wilensky. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. Paper presented at the Proceedings of the 14th International Conference on Interaction Design and Children, Boston, Massachusetts
11. Snap! (Build Your Own Blocks) 4.2. Retrieved from https://snap.berkeley.edu/
12. Coding for Kids | Tynker. Retrieved from https://www.tynker.com/
13. Ouahbi, I., et al. (2015). Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment. Procedia - Social and Behavioral Sciences, 191, 1479-1482. doi:10.1016/j.sbspro.2015.04.224
14. Kazimoglu, C., et al. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. Procedia - Social and Behavioral Sciences, 47, 1991-1999. doi:10.1016/j.sbspro.2012.06.938
15. Shabanah, S.S., et al. (2010). Designing Computer Games to Teach Algorithms. Paper presented at the 2010 Seventh International Conference on Information Technology: New Generations, Las Vegas
16. Grivokostopoulou, F., et al. (2016). An Educational Game for Teaching Search Algorithms. Proceedings of the 8th International Conference on Computer Education 2, 129-136. doi:10.5220/0005864601290136
17. CodeCombat - Learn how to code by playing a game. Retrieved from https://codecombat.com/
18. LightBot Retrieved from http://lightbot.com/
19. Combéfis, S., et al. (2016). Learning Programming through Games and Contests: Overview, Characterisation and Discussion. Olympiads in Informatics, 10, 39-60. doi:10.15388/ioi.2016.03

20. Miljanovic, M.A.J.S. Bradbury. (2018, November 7-8, 2018). A Review of Serious Games for Programming. Paper presented at the 4th Joint International Conference on Serious Games, Darmstadt, Germany.
21. Franković, I., et al. (2018). Serious Games for Learning Programming Concepts. Paper presented at the 8th International Conference the Future of Education, Florence, Italy
22. Wolz, U., et al. (2008). 'Scratch' Your Way to Introductory CS. SIGCSE Bulletin, 40, 298-299.
23. Alice - Tell Stories. Build Games. Learn to Program. Retrieved from http://www.alice.org/
24. Meerbaum-Salant, O., et al. (2013). Learning computer science concepts with Scratch. Computer Science Education, 23(3), 239-264. doi:10.1080/08993408.2013.832022
25. Kelleher, C., et al. (2007). Storytelling alice motivates middle school girls to learn computer programming. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, California, USA
26. Sykes, E.R. (2007). Determining the Effectiveness of the 3D Alice Programming Environment at the Computer Science I Level. Journal of Educational Computing Research, 36(2), 223-244. doi:10.2190/J175-Q735-1345-270M
27. Cooper, S., et al. (2000). Alice: a 3-D tool for introductory programming concepts. Journal of Computing Sciences in Colleges, 15(5), 107-116.
28. Geist, E. (2016). Robots, Programming and Coding, Oh My! Childhood Education, 92(4), 298-304. doi:10.1080/00094056.2016.1208008
29. Tomorrow Corporation: Human Resource Machine. Retrieved from http://tomorrowcorporation.com/humanresourcemachine
30. Johnson, C., et al. (2016). Game Development for Computer Science Education. Paper presented at the Proceedings of the 2016 ITiCSE Working Group Reports, Arequipa, Peru
31. Jemmali, C. (2016). May's Journey: A serious game to teach middle and high school girls programming. Worcester Polytechnic Institute, Retrieved from https://digitalcommons.wpi.edu/etd-theses/455
32. Vahldick, A. (2017). Aperfeiçoamento das Competências de Resolução de Problemas na Aprendizagem Intodutória de Programação de Computadores usando um jogo sério digital. Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Retrieved from http://hdl.handle.net/10316/79528
33. Miljanovic, M.A.J.S. Bradbury. (2016). Robot on!: a serious game for improving programming comprehension. Paper presented at the Proceedings of the 5th International Workshop on Games and Software Engineering, Austin, Texas
34. Malliarakis, C., et al. (2014). CMX: Implementing an MMORPG for learning programming. Paper presented at the 8th European Conference on Games Based Learning, Berlin
35. Mathrani, A., et al. (2016). PlayIT: Game Based Learning Approach for Teaching Programming Concepts. Educational Technology & Society, 19(2), 5-17.
36. Bartilla, A.C. Köppe. (2015). Awareness seeds for more gender diversity in computer science education. Paper presented at the Proceedings of the 20th European Conference on Pattern Languages of Programs, Kaufbeuren, Germany

Co-funded by the Erasmus+ Programme of the European Union

37. Hosein, A. (2019). Girls' video gaming behaviour and undergraduate degree selection: A secondary data analysis approach. Computers in Human Behaviour, 91, 226-235. doi:10.1016/j.chb.2018.10.001
38. Alserri, S.A., et al. (2018). Gender-based Engagement Model for Serious Games. International Journal on Advanced Science, Engineering and Information Technology, 8(4), 1350-1357. doi:10.18517/ijaseit.8.4.6490
39. Carmichael, G. (2008). Girls, computer science, and games. ACM SIGCSE Bulletin, 40(4), 107-110. doi:10.1145/1473195.1473233
40. Osborn, G. (2017). Male and Female Gamers: How Their Similarities and Differences Shape the Games Market. Retrieved from https://newzoo.com/insights/articles/male-and-female-gamers-how-their-similarities-and-differences-shape-the-games-market/
41. Vermeulen, L., et al. (2011). Girls will be girls : a study into differences in game design preferences across gender and player types. Paper presented at the Under the mask: perspectives on the gamer, Luton, UK.Retrieved from http://hdl.handle.net/1854/LU-1886961
42. Kafai, Y.B. (2006). Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies. Games and Culture, 1(1), 36-40. doi:10.1177/1555412005281767