UNIVERSITY OF MACEDONIA
SCHOOL OF INFORMATION SCIENCES
DEPARTMENT OF APPLIED INFORMATICS

Serious Games for teaching and learning computer programming: Design, development and evaluation of a customizable educational Massive Multiplayer Online Role-Playing Game (MMORPG)

Ph.D. Dissertation

Christos Malliarakis

THESSALONIKI, GREECE
OCTOBER, 2015

Serious Games for teaching and learning computer programming: Design, development and evaluation of a customizable educational Massive Multiplayer Online Role-Playing Game (MMORPG)

Christos Malliarakis
Bachelor's Degree (BSc) in Applied Informatics, University of Macedonia, 2007
Master's Degree (MSc) in Information Systems (MIS), Department of Computer Science of the Aristotle University, 2010
Master's Degree (MSc), Informatics and Management, offered jointly by the Departments of Informatics and Economics of the Aristotle University of Thessaloniki, 2011

Ph.D. Dissertation

SUPERVISOR
Satratzemi Maya
Professor, Department of Applied Informatics, University of Macedonia

ADVISORY COMMITTEE
Satratzemi Maya
Professor, Department of Applied Informatics, University of Macedonia

Refanidis Ioannis
Associate Professor, Department of Applied Informatics, University of Macedonia

Xinogalos Stelios
Assistant Professor, Department of Applied Informatics, University of Macedonia

EXAMINATION COMMITTEE

Satratzemi Maya
Professor, Department of Applied Informatics, University of Macedonia

Refanidis Ioannis
Associate Professor, Department of Applied Informatics, University of Macedonia

Xinogalos Stelios
Assistant Professor, Department of Applied Informatics, University of Macedonia

Chatzigeorgiou Alexandros
Associate Professor, Department of Applied Informatics, University of Macedonia

Kaskalis Theodoros
Associate Professor, Department of Applied Informatics, University of Macedonia

Sakellariou Ilias
Lecturer, Department of Applied Informatics, University of Macedonia

Economides Anastasios
Professor, Department Of Economics, University of Macedonia

Abstract

Computer programming has for decades posed several difficulties for students of all educational levels. A number of teaching approaches have been proposed over the years but none seems to fulfil the needs of students nowadays. On the other hand, students use computers mainly for playing games and using the Internet, and as quite a few researchers state, this aspect of computers should be taken into account in the way we educate young people. Towards this direction, in this thesis we aimed to initially investigate the educational games developed for and used in the computer programming domain and review to which level they address the aforementioned difficulties. Related work includes a number of games that have been developed towards this goal. However, even though they seem promising examples, they lack features that would allow them to successfully underpin computer programming learning by facing the majority of the identified problems. To this end, we propose a new, advanced Massively Multiplayer Online Role-Playing Game (MMORPG) named CMX that includes the facilitating and positive features identified in existing solutions and incorporates missing elements that will bring forth a new generation of educational games for computer programming. The thesis proceeds to elaborate on the educational features of CMX as well as present a design framework that was devised taking into account previous work in designing educational games. Furthermore, a framework is constructed and presented that will guide the incorporation of learning analytics mechanisms in computer programming education. Moreover, since MMORPGs require significant amounts of resources, such as bandwidth, RAM and CPU capacity to support learning, a new methodology is proposed to achieve monitoring and optimization of the load balancing, so that the essential resources for the proper execution of an educational MMORPG for computer programming can be foreseen and bestowed without overloading the system. Finally, the effects of CMX on teaching and learning computer programming are assessed through a study with undergraduate students. Students evaluated various aspects of CMX by filling in a questionnaire that was based on an evaluation framework, which was devised in accordance with the design framework of CMX. The results show that the majority of the students increased their performance in computer programming, were entertained by playing the game while learning, and felt motivated to continue based on the game's scenario due to the variety of activities included. Furthermore, students stated they had a positive attitude in regards to re-using CMX in the future in order to learn additional

programming concepts. The positive results of this study pave the way for CMX being used in the classroom and expanding the game's functionalities that will further increase students' performance and support teachers in delivering the required knowledge. Moreover, the work reported offers game designers and teachers methodological and empirical results for game based learning in such a difficult domain as is computer programming. What is more, the design and evaluation frameworks presented are general enough that they can be easily adjusted and/or extended for designing and assessing educational games in other domains as well.

Keywords: serious games, computer programming, learning process, holistic frameworks, learning analytics, educational programming environments, design framework, educational games, MMORPG, evaluation framework.

Acknowledgements

My Ph.D. studies were a unique experience in my life. Apart from the scientific knowledge that they undoubtedly provided me with, they expanded my research concerns and strengthened my critical perception and creative thinking. For these reasons, I would like to thank the people who contributed to the successful completion of the dissertation.

Firstly, I would like to express my sincere gratitude to my advisor Prof. Maya Satratzemi for the continuous support of my Ph.D. studies and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD studies.

My co-advisor, Assistant Prof. Stelios Xinogalos, has been always there to listen and give advice. I am deeply grateful to him for the long discussions that helped me sort out important details of my work. I am also thankful to him for the consistent notation in my writings and for carefully reading and commenting on countless revisions of my manuscripts.

Besides my advisors, I would like to thank Associate Prof. Ioannis Refanidis for his insightful comments and encouragement.

I would also like to make a special reference to Mandoulides Schools, where I was fortunate to work in as a teacher in Computer Science. It is a pioneering school that constantly looks for educational innovation and this was a source of inspiration for me. Finally, I would like to thank the Research Committee of the University of Macedonia for the financial support that it provided to me for covering conference-related expenses.

Most importantly, none of this would have been possible without a special person in my life, Ms Maria Zotou. She has been a role model for me as a researcher and a constant source of motivation all these years. I would like to express my gratitude to her as she has aided and encouraged me throughout this endeavour.

Last but not least, I would like to thank my family for supporting me spiritually throughout writing this thesis and my life in general. I thank them for the support, the unconditional love and the constant encouragement to continue my studies at this level. I am really proud of them and I consider a large part of my research belonging to them.

Table of Contents

Figures

1. Introduction

1.1. Field research and thesis objectives

The field of computer programming education has been studied for numerous years and research indicates that significant difficulties and challenges are faced by teachers and students respectively (Koulouri, Stanislao & Macredie, 2014). These challenges regard the complexity of the domain, which includes complicated knowledge with abstract concepts and the demanding development of practical skills, such as logical thinking, problem solving etc. (Moser, 1997). These issues hinder computer programming education, as students usually characterize the courses as boring and difficult to understand. Furthermore, students consider the teaching methods used during these courses as uninteresting and state that they are not provided with enough diverse exercises in order to practice the concepts they have been taught (Roslina & Nazli, 2009).

To this end, researchers have endeavoured to identify and apply solutions that could address these challenges and enhance computer programming education. A representative encouraging suggestion involves the utilization of educational games in the classroom, which are gradually being accepted as a promising educational tool (Johnson et al., 2013), because it is evidenced that they increase students' performance and motivation. The usage of games in education goes back to ancient times, in fields such as battle training, where games were not only preferred because of their entertainment value but also because they captured trainees' attention and allowed them to have fun while learning. The same applies nowadays with the digital games and media, where the game industry is one of the biggest entertainment industries worldwide. This has prompted the usage of computer games in different areas other than entertainment, one of which is education and training. The features of this technology that are attractive to players (e.g. attractive graphical interface, an engaging scenario, tasks/quests, interaction activities, characters etc.) can be incorporated in subjects usually considered boring or uninteresting, such as computer programming.

Particularly, multiplayer online games can provide multiple benefits when utilized in education (Garzotto, 2007); for example, Massive Multiplayer Online Role Playing Games (MMORPGs) continue to be popular as they reinforce motivation and

creativity and allow skill development, such as problem solving and communication (Beedle & Wright, 2007). The incorporation of MMORPGs in education is encouraged by related studies, since they can foster learning by enabling students to practice, think critically and plan strategically (Dickey, 2007).

The successful incorporation of such a technology in education in order to increase students' performances and tackle lack of motivation involves a new and ongoing research on instructional design, that the new educational games that will be designed and developed will offer not only entertainment but also educational value. This statement brings the next questions. Is it possible to develop an educational game that will work as an effective educational tool? How will such a tool be used to teach computer programming? Is it possible to create an educational game that can be configured and adapted for other courses and domains?

Research is being carried out in regards to the previous questions. However, existing efforts usually tackle one of the aforementioned situations, and each situation has a different context, problems and objectives. This makes the creation of an educational game that will have addressed all the identified issues a challenging and not yet resolved problem. Thus, this thesis will focus on developing, using and validating an educational game that will:

a) be designed according to an appropriate design framework that will include entertainment and educational elements,
b) include features appropriate for computer programming courses and
c) allow configuration of its components so that it can be adapted for other courses and domains.

The above, aim to lead to a game that will support students to increase their knowledge on computer programming while also enjoying the new learning experience and will support teachers to properly structure their lessons and monitor the learning process.

1.2. Contribution of the thesis

The thesis focuses on the field of educational games for the learning and teaching of computer programming. However, the contribution of the work done is not limited to

the research area of computer programming, but it involves game-based learning in general. More specifically, the contribution is summarized in the following six parts:

(1) A new design framework is constructed that aims to support the design of future educational games for computer programming and also to support the development of the CMX game in the context of this thesis.

(2) A new evaluation framework was constructed to support the evaluation of the usage of educational games. This framework gathers the main parameters that determine the successful usage of an educational game in the classroom and is based on the initial objectives set in the design framework.

(3) An educational MMORPG was designed and developed named CMX that supports teaching and learning of computer programming. The game was designed based on the constructed design framework and after the thorough review of all the relevant games of the area, gathering the positive characteristics and incorporating innovative elements.

(4) A new mathematical optimization model was constructed to monitor and enhance the game's performance. The model is based on the prediction of the resources required to be bonded, by storing information on the players' behavior within the game.

(5) A new model was constructed for the exploitation of the information that is generated during the game, aiming to assess the students in regards to the achievement of the set educational goals (learning analytics). The model calculates relevant evaluation metrics which also produce informative reports.

(6) Five different versions of the game were designed and three implementations and evaluations of the game were carried out on first-year, third-year and postgraduate students of the Applied Informatics department of the University of Macedonia. The students assessed the game and helped answer fundamental research questions regarding the performance of the game, the educational content and the motivations that drive the usage of the game in the learning process. The results were particularly encouraging and verify CMX as effective and beneficial educational software for the teaching and learning of computer programming.

1.3. Structure of the thesis

The thesis consists of seven chapters. The first chapter is the introduction of the research carried out, while chapter 2 provides background information on related work. More specifically, it presents the definition of serious and educational games, it describes different educational approaches and defines the difficulties, mistakes and misconceptions in learning computer programming. It also describes the traditional computer programming education processes and the computer games as educational software for teaching and learning computer programming.

Chapter 3 aims to present existing frameworks that have been proposed by published works for the design and development of domain-independent and computer programming-specific educational games as well as specific features that should be supported.

Chapter 4 presents a series of educational games that have been developed specifically for computer programming courses. The review of these games was carried out based on the specifications described in the studies, where they were explicitly identified in the relevant literature.

Chapter 5 presents an overview of CMX, including the design framework, by presenting the methodology followed, a brief overview of the concepts included as well as a concise comparison of the framework with all identified existing frameworks, the scenario, the user roles supported and the features incorporated. The chapter also presents the CMX editor and all its functionalities as well as the CMX server administration.

Chapter 6 includes a section that presents the architecture of CMX, a system performance analysis and the corresponding creation of a mathematical model that estimates how active players are hosted within an MMORPG game in a server so that the server administrators can take the necessary measures that will ensure optimized server's performance. Moreover, the chapter includes a process to ensure server's performance optimization. The last section presents the evaluation of this optimization with an experiment using the CMX game.

Chapter 7 describes the evaluation of CMX and presents the data collection and the statistical analysis carried out based on the gathered results and all the corresponding

findings. Finally, Chapter 8 offers conclusions drawn and future work that will increase the game's efficiency in supporting computer programming education.

2. Theoretical Background

2.1. Computer programming education approaches

This section presents the literature review concerning features of traditional computer programming education and the main teaching problem, difficulties and mistakes that are identified. Furthermore, the section provides a brief description of alternate teaching approaches for the computer programming domain that have been developed aiming to support the learning process and address the problems mentioned.

2.1.1. Difficulties in learning computer programming

Learning and teaching computer programming has always been a challenge for both the teachers and the students. Researchers have been trying to investigate these challenges and why they occur for the last two decades (Koulouri et al., 2014; Rajaravivarma, 2005; Chang & Chou, 2008; Eagle & Barnes, 2009; Muratet et al., 2009). A result of the research carried out was the attempt to explain both the causes that make it difficult to learn programming, and the teaching methods applied. Most scientific studies have given special attention to the difficulties, mistakes and misconceptions associated with the structures of programming languages and especially in recording errors for specific syntactic structures of each language. Also, studies propose specific teaching approaches that can be applied in order to address such difficulties, and also to allow students to exploit the skills they acquire to the maximum.

The difficulties encountered in learning computer programming are referred to as a possible reason for the steady low rates within the Computer Science (CS) discipline. Various studies express that these challenges have not been properly addressed (Beaubouef & Mason, 2005; Kinnunen & Malmi, 2006; Fletcher & Lu, 2009). Furthermore, the assignment of learning how to construct a program is often considered as demanding and upsetting, especially by students that are novices in computer programming (Bennedsen & Caspersen, 2008).

Recent studies claim that these reactions are due to insufficient learning strategies, lack of interaction with the learners and absence of interest (Barker, McDowell & Kalahar, 2009; Coull & Duncan, 2011). More specifically, as Guzdial

(2004, pp. 19) mentions, “Students want to work on computational artefacts that have meaning for them, e.g., that are interesting and relevant”. Relevant research studies have also made the connection between learners’ dropout rates and their lack of motivation to learn computer programming, since the methods and activities used in such courses are often viewed as boring and not relevant (Wilson, 2002; Beaubouef & Mason, 2005).

Learners’ low motivation and challenges in learning computer programming are issues that have been reported for many years. More specifically, Soloway (1986) stated that the main issues that novice learners face during computer programming education is “putting pieces together”. Soloway (1986) also mentioned that when a program is provided as a problem (e.g. read three numbers and figure out their average), students with high expertise can remember the appropriate computer programming concepts that should be used the minute they understand what the given problem is and how they can solve it; whereas novice students often fail to form the correct programming structure needed, usually because they lack full understanding of the problem and its possible solution. This is due to these students’ limited abilities and skills in forming abstract reflections of problems and their programs and cannot properly depict the computer programming concepts that are required to create the solution (McCracken et al., 2001).

All the aforementioned problems have been studied by several research groups, which tried to identify why students, especially novice ones, face such difficulties in the field. These studies derived a few subjective results and the most prominent was the nature of computer programming itself (Gomes & Mendes, 2007; Hawi, 2010; Coull & Duncan, 2011). This is because the task of forming programs requires a significant comprehension ability regarding computer science concepts and the skill in sorting these concepts in the correct order with the correct syntax in order to solve a given existing problem. However, novice students usually have a different approach of the field, and they consider computer programming a completely technical task that requires solely a recalling of programming commands (Bennedsen & Caspersen, 2007).

Additionally, students face difficulties in writing a program because they are required to start thinking in a different, more abstract way, as well as figure out the correct syntax that will translate their solution to an actual executable program (Dalal et al., 2009). This is why usually, and without sometimes realizing it, students do not take the time to analyze the given problem and design their solution using abstract and algorithmic thinking; they start tackling the coding from the beginning and try to form

the program by focusing on the commands to be written and their order (Rajaravivarma, 2005).

Consequently, the process of learning computer programming requires the development of numerous skills such as abstraction and algorithmic thinking in order to analyse and design a solution as well as an in depth understanding of the computer programming language and its commands that will allow the translation of the solution into an actual program.

2.1.2. Common mistakes in learning computer programming

2.1.2.1. Common mistakes in Procedural Programming

Numerous research studies have been carried out on common mistakes in specific programming structures in procedural programming (Rowe & Thorburn, 2000; Naps et al., 2003). Such structures include variable initialization, loops and conditionals. More specifically, studies show that students find it more difficult to initialize than to update or test a variable. Also, students often express difficulties with having to update variables within "for" loops or implementing recursion. An interesting conclusion stated that novice students could write recursive functions more easily if they had been taught about iterative functions first, but not vice versa (Kay et al., 2000).

Furthermore, students found it difficult to distinguish among expressions that are very similar to each other syntactically or that have other meaning in different contexts, e.g. different programming languages; for example, the expressions "100" and 100 where one is a string and the other a number, or the word "static" in the C language.

One of the most challenging structures in procedural programming for students as well as teachers seems to be the pointers (McCracken et al., 2001). Other structures that also cause many difficulties are dynamic memory allocation and reference parameters, which, similar to pointers, require a deep understanding of how memory works in computer programming.

The most common and important mistakes that can be noticed in procedural programming are shown in the following table (Pillay & Jugoo, 2006).

Table 2-1: Important mistakes in procedural programming (Pillay & Jugoo, 2006)

Category	*Errors*
I. Variables, Typing and Constants	• Calculations performed as part of variable declarations. • Assigning more than one calculation to a variable. • Variables declared to be of the incorrect type. • Accumulators not initialized. • Incorrect initial value assigned to an accumulator. • Declaration of unnecessary variables. • Unnecessary initialization of variables. • Undeclared variables. • Reference to variables out of scope. • Type of variable not specified in declaration. • Use of variables that have not been assigned values only declared. • Placement of declaration of variables and constants, e.g. before the main function.
II. Arithmetic Operations	• Calculations placed in inverted commas. • Incorrect syntax of equations. • Incomplete equations. • The output format function used instead of the round function. • Incorrect use of a function, e.g., the ceil function. • Difference between / and % not understood. • Integer division performed when not required. • Operation of the ++ operator not understood.
III. Logical Operations	• Incorrect syntax of Boolean equations • = used instead of ==. • & used instead of &&.

	• !== used instead of !=. • Equality of more than one value tested with == instead of a combination of == and ANDs. • ANDs used where ORs should be used and vice versa. • Separate if-statements used instead of using the OR operator.
IV. Input and Output Operation	• Result of program not output. • Each possible value of input read in and stored instead of using one variable and testing the value read in. • Assumption that spaces in program will appear on the screen. • Output not formatted or incorrectly formatted. • Output statements extending over more than one line. • Missing inverted commas in string output. • Syntax errors in the combination of variable and string output.
V. Conditional Control Structures	• Syntax errors such as missing semicolons. • Block of statements to be executed not contained in { }, thus entire block not executed if condition is met. • If - else used instead of an else for the last option. • Condition not specified. • A number of else's without if's. • "if else" used instead of "else if". • Single if- statements used instead of if - else statements and vice versa. • If - statement used instead of a conditional loop and vice versa.
VI. Iterative Control Structures	• Syntax errors, e.g. missing semicolons; incorrect for loop declaration. • Incorrect type of loop used, e.g. while instead of dowhile. • Conditional loop used instead of if - statement and vice versa. • No content in loop.

	• Infinite loops. • Counter variable changed in loop. • Incorrect conditions for conditional loops. • Counter variable used as accumulator. • Accumulator initialized in the loop. • Calculations that should be performed after the loop are performed in the loop. • Loop update not performed for conditional loops. • Incorrect update performed in loop. • Functioning of the different loops not understood. • Scope of variables used in conditional loops not understood. • Scope of the counter variable used in the for loop not understood.
VII. Modularization	• Syntax errors e.g. return statement specified in (), semicolon after function declaration. • Function returning a value declared as void. • Value returned by function not stored or used. • Variable to store function output passed to function. • Name of function returned. • Function defined after class definition. • Function not used although question requires it. • Function included in main function. • Typed function does not return a value, outputs or stores the value. • Values not passed to function. • Return statement used to call up function in main function. • Function performs the incorrect function. • Function name does not reflect the function being performed. • Function stores values passed to function in new variables instead

	of using dummy variables (refer to the parameters specified as part of the function declaration). • Value of the incorrect type returned. • Dummy variables declared in function and function declaration. • Scope of variables not understood, e.g., dummy variables (not declared in main method) used in function call to function. • Attempts to return two values from a function. • Values passed to function overriden.

Two other interesting studies by Hristova et al. (2003) and Ahmadzadeh et al. (2005) categorized the aforementioned mistakes in eight specific categories using three criteria: syntax, semantics and logic. The established categories are as follows:

1. Incorrect transfer of knowledge. This regards the cases where students apply concepts and rules they have learned in previous programming exercises incorrectly to solve a specific problem (Pillay & Jugoo, 2006). Representative examples include the usage of variable names that are not relevant to the specific problem, the incorrect assignment of values in variables based on previous examples, the entering of values when it is not necessary etc.

2. Lack of understanding the application domain. One of the most common and important mistakes is when students do not understand or misinterpret the description of the problem they have to solve by programming or when they do not know anything about the process which they have to program. These mistakes were also noticed by Spohrer et al. (1986).

3. Lack of understanding control structures. These errors occur when students do not understand the function of a control structure (Pillay & Jugoo, 2006). For example, students might update a counter variable within a "for" loop after the condition is checked or within a "do while" loop at the beginning of the loop etc.

4. Lack of imagining the execution of the program. Many students are not able to envision how the program they have to write will be executed, that is how each variable and function will be affected after the execution (Pillay & Jugoo, 2006). This

can lead to students executing different functions or statements in the incorrect order, adding statements in a loop that should be executed at a later time, not posting the output of a variable when required etc. Other studies have also reported that it is very common for students to not imagine what each line of the program is actually doing during execution (Ahmadzadeh et al. 2005; Bailey 2005; Chmiel et al. 2004; Edwards 2004; Lahtinen et al. 2005).

5. Problem solving skills. Students are often unable to properly write a program because they cannot solve the given problem correctly. This is due to their lack of problem solving skills, which prevents them from forming a complete algorithm based on the problem. Other studies confirming this common mistake are Perkins et al. (1986), Fergusson et al. (2004) and Lahtinen et al. (2005). In particular, Beaubouef et al. (2001) specify that this mistake does not reflect students' inability to solve computer programming problems but problems in general.

6. Lack of knowledge and understanding of the programming language. This category regards the fact that some students have not acquired the necessary knowledge of the programming language so that they can correctly use it in practice. For example, students have not understood the syntactic rules of the language or how an operator works or how a loop function is structured etc.

7. Incorrect identification of the control structure to be used. Students often do not realize which of the control functions they need to use in a program e.g. usage of a conditional loop instead of an "if" statement for checking errors or usage of a "while" loop where a "for" loop is needed etc (Pillay & Jugoo, 2006).

8. Inefficient approach to solving the problem. Students often are able to interpret the problem correctly and write the program, however the solution they choose is not effective in some way, e.g. it requires additional memory when this could have been prevented or the program is too long when the lines of code should be fewer (Pillay & Jugoo, 2006). These issues occur when for example students perform more conditional checks than necessary or declare additional variables that they do not use or need etc.

2.1.2.2. Common mistakes in Object-Oriented Programming

Numerous studies have researched and analyzed the common mistakes that occur during object-oriented programming. The most common mistakes identified are as follows:

- **Inability to differentiate between class and object**. Holland et al. (1997) state that some students get confused and cannot differentiate between class and object concepts. This issue is more serious when programming exercises use only one object of each class and therefore the differences are not made obvious (Holland et al., 1997).
- **Inability to differentiate between object and variable**. Holland et al. (1997) state that especially students that are already knowledgeable in procedural programming will make the early assumption that objects are identical to variables, especially in examples where a class has a single variable.
- **Inability to differentiate between identity and attribute.** Holland et al. (1997) conclude that this confusion can lead to several misconceptions, such as:
    - Only one variable can be referred to a specific object at a specific time.
    - When a variable is referred to a specific object, then it will always refer to this specific object.
    - If there are two different variables, these need to refer to two different objects.
    - Two objects of the same class in the same condition are the same object.
    - Two objects with the same value for an attribute that is related to the provision of a name to the objects, e.g. "name", is the same object.
- **Objects are simple records with no behavior**. Students often do not understand that the behavior of an object can change depending on its state (Holland et al., 1997).
- **Optional usage of constructors**. Constructors' main purpose is to initialize the variables of an object (Fleury, 2000), which can also be carried out using a function that initializes the variables of the new object.
- **A method changes the state of an object simply with values assignment**. Usually the first lessons in object-oriented programming include objects with a non-changing state and therefore students do not realize that this state can change with methods that assign new values to the object's variables. Holland et al.

(1997) mention that this mistake is also observed even in the most experienced students.

- **The usage of the dot operator is valid only for methods and not for variables** (Fleury, 2000).
- **The short length of a program is preferable to nesting.** Many students think that reducing the lines of code and the classes more important than nesting. This is also because they find it easier to detect all the different components of their program (Fleury, 2001).
- **Polymorphism under conditions**. Methods of different classes can have the same name only if they have different identities (Fleury, 2000).
- **Methods with parameters**. Students lack understanding of methods with parameters in comparison to methods without parameters (Fleury, 2001).

Carter and Fowler (1998) also researched common mistakes in object-oriented programming, and they stated that students get easily confused when developing programs with more than one classes. More specifically, students cannot determine if they should add all the program's classes in one file or in different files. The study also pointed out that some students are often confused when they are required to use multiple constructors and when using the command "return". More specifically, especially students that do not have a prior knowledge of the language Pascal do not understand how to use the "return" command in order to exit a loop.

Finally, another study mentions that some students are unable to create a concept model for the execution of the program (Milne & Rowe, 2002). The study provided questionnaires online regarding the difficulties that students face in the context of an introductory object-oriented programming course. The statistical analysis of the answers gathered showed that the greatest difficulties occur because students cannot understand the concept of pointers and all the concepts that are related to the memory. This is because students are unable to create the concept model of executing the system in order to comprehend how memory works.

2.1.3. Effective computer programming teaching technologies

Learning computer programming faces as many challenges as teaching it. A number of researchers have investigated these difficulties and ways to tackle them in order to improve education in the field (Lahtinen et al., 2005; Ragonis & Ben-Ari, 2005). The results of the research studies record a variety of educational technologies that can be used to assist learning and teaching programming. These technologies could be applied on programming courses in order to underpin the process and help students develop internal incentives, thus foster the feeling of challenge, curiosity, active control and imagination (Ho et al., 2006). We recognize three main categories of technologies that can be used in such settings, as follows:

Educational Programming Environments are sets of tools supporting the student in the process of learning introductory programming (Brusilovsky et al., 1997). Programming microworlds (Brusilovsky et al., 1997) are learner-centred worlds that are explored by directly manipulating objects in the world with a limited set of simple commands coupled with metaphors to aid in problem description and to exploit storytelling (Long, 2007). Serious Games are computer games that are used as educational tools and provide compelling contexts via interactive, engaging and immersive activities (Gunter et al., 2008).

2.1.3.1. Educational programming environments

Our literature review depicted educational programming environments that teachers incorporate in their programming courses in order to facilitate students' comprehension of difficult concepts. These environments support the teaching and learning of object-oriented programming, which has prevailed as a programming model.

DrJava (Allen et al., 2002) is an example of an environment that aims to give students the opportunity to focus on designing programs, rather than learning the mechanics of using the environment. DrJava is a simple environment with just two panes: the definitions pane for defining classes and the interactions pane for the interactive evaluation of the written code. However, this interaction is based exclusively on text, while no direct manipulation techniques are supported. Furthermore, there is no visualisation of object-oriented programming concepts. The central idea of DrJava is using a *"read-eval-print loop"* (read, evaluate, print, REPL) for incremental program

development through interaction with program components as they are being developed. DrJava requires little programming experience and it is oriented to ages 15-25.

BlueJ (Kölling et al., 2003) is "an integrated Java development environment specifically designed for introductory teaching" and is widely used in object-oriented programming courses. The main window of BlueJ presents students with a simplified UML class diagram of the currently open project. Students can create objects by invoking a constructor through interaction with the classes of the UML diagram, and then interact with the depicted objects by calling methods through each object's pop up menu. Furthermore, students can inspect an object's state during method execution. Visualization of the object – oriented entities, such as objects and classes, is an important feature that aims at the comprehensive presentation of their differences. BlueJ incorporates a simple editor that uses different background colours for highlighting blocks of code, reports explanatory error messages and provides a java interpreter (code pad), as well as a debugging ability (Kölling et al., 2003). Overall, BlueJ is characterized by the simplicity of using all the aforementioned features. Finally, a large number of plug-ins has been developed for providing important features missing from BlueJ, such as a class card for presenting the structure of a class. BlueJ requires little programming experience and it is oriented to ages 15-25.

Another example is JGrasp (Hendrix et al., 2004) where students can create UML class diagrams, objects and classes and interact with them by calling methods using the graphical interface. This tool visualizes the static aspect of programs too: class structure information (fields, constructors, methods) is presented; control structure diagrams (CSD) are used so that students can more easily understand the code structure within classes. These diagrams assist students in understanding the source code better by visualizing the program sequence and structure. JGrasp provides easy compilation and debugging, explanatory error messages and error indication via code line colour change as well as templates and code examples. The interpreter of JGrasp, in contrast with that of BlueJ, updates automatically the graphical interface (for example, a constructed object is depicted). Additionally, JGrasp allows a dynamic manipulation of objects (such as changing the type of an object and/or access modifiers during execution) favouring experimentation. JGrasp is also oriented to ages 15-25.

The Scratch environment developed at the Massachusetts Institute of Technology (MIT) Media Lab and it introduces computer programming to students of ages 5 to 18

(Resnick, 2007). Towards this goal, students can create programs, interactive stories and games through drag & drop tiles instead of writing lines of code, thus avoiding making syntax errors. Furthermore, a student writes a program in Scratch using pseudocode and not a programming language. It should be noted that interaction and scenarios play an important role in this game as they intend to stimulate the interest of young students by engaging them in a series of attractive assignments. The availability of multiple characters (sprites), backgrounds, sounds and images motivate students to create their own animations and games, play on their own and share their creations with their classmates, as well as reflect their creations to others.

MIT App Inventor (Wolber, 2011) is a blocks-based cloud programming tool that enables everyone to develop fully functional apps for Android devices. In App Inventor the programmer develops the program as in Scratch in an interactive way using the drag and drop technique using pseudocode and not a programming language. He patches together blocks with specific meaning like loops, if-then-else statements or variables. The pieces fit in each other like in a jigsaw puzzle. App Inventor consists of three parts: The Designer: where the programmer selects components from a palette (object of a certain class) and drags them to the Viewer (screen layout) to easily build the GUI of his app. Then, the static attributes and properties of the selected component can be changed. The Block Editor: provides blocks of code ("build-in drawers") with specific behavior. The programmer drags & drops them to the blocks viewer and he puts them together so as to define his app's dynamic behavior. The app can respond to various events, such as user events (touching the screen) or system events (a new location). App Inventor provides databases, geolocation, audio and video contents and other features. It is oriented to students that are older than 14 years old and have little or no programming experience.

2.1.3.2. Programming microworlds

Numerous programming microworlds have been suggested over the years for teaching computer programming to novice students (Sanders & Dorn, 2003). Most of these systems visually represent methods and classes for the easier and more direct comprehension of the most difficult concepts. Additionally, microworlds attempt to introduce students to object-oriented programming by simulating actual worlds and

protagonists and assigning tasks to students, providing a visual overview of their progress.

Alice (Cooper et al., 2003) is a 3D interactive, animation, programming environment that aims to teach computer programming to students by visualizing objects and their behaviours. Specifically, students can use/modify 3D objects in order to create their own virtual worlds and write scripts for controlling the objects' appearance and behaviour. Alice utilizes a drag-and-drop smart editor for dragging objects into the editor and drop-down menus for selecting messages to send to an object. User-defined methods and functions are automatically added to the dropdown menus. Students can see immediately how their animation programs run – each action is animated smoothly - enabling them to easily understand the relationship between the code and its result.

Jeroo (Sanders & Dorn, 2003) is a programming microworld for teaching object-oriented programming. This environment intends to teach students how to write programs in a simple manner. Some of the features that can be identified are the visualization of objects as actors (an Australian kangaroo) and their behaviours (movement around the world based on the written code) and the explanatory error messages. Various tasks can be assigned to the actors in a restricted world that consists of flowers, nets and water. Jeroo has the following restrictions: there is only one class; students can create up to four actors; inheritance is not supported; students can extend the Jeroo class with void methods, but not with predicates.

objectKarel (Xinogalos et al., 2006) is based on Karel++ (Bergin et al., 1997) which uses the well-known metaphor of robots carrying out various tasks on a restricted world. objectKarel incorporates a series of e-lessons and hands-on activities for motivating students through the use of concepts before they are asked to implement them. Programs are developed with a structure editor, which has the form of a menu and aims at avoiding syntax errors and focusing on concepts. Highly informative error messages are reported for the few errors that can occur, explaining the source of the error in natural language. Students can run, trace with a predefined speed or execute programs step by step (only forward). Also, objectKarel incorporates the technology of explanatory visualization, which means that students are presented with explanatory messages for the semantics of the statement being executed.

Greenfoot (Kölling, 2010) is considered an extended version of BlueJ and it aims to address the lacking features observed in that environment, such as the direct visualization of the objects' behaviour and state. To this end, it is considered both a microworld and an educational programming environment. It intends to assist in the teaching of object-oriented programming in a simple manner and through the visualization of objects, their state and behaviour. More specifically, this tool was designed in order to support the essential features that were identified in the existing similar tools, (e.g. the visualization and simplicity features included in Karel the Robot), as well as features identified in some educational programming environments (e.g. interaction with objects included in BlueJ) (Henriksen & Kölling, 2004). Additionally, it supports interaction between the students and the object-oriented programming concepts within a specified world, where students assign behaviours to actors by calling methods and observe how the actors behave. It also provides an editor, a compiler that provides explanatory messages as well as a user-friendly debugger and is considered suitable for school age novice to object-oriented programming students.

2.1.3.3. Computer games as educational software for teaching computer programming

The usage of computer games in the learning process is a promising way to attract and engage students to learn. One of the main characteristics of good computer games (Gee, 2003) is that they allow players to think of them as design objects. The studies that suggest using these games during learning mention academic domains such as Languages and Education (Gee, 2003), Maths (Kafai, 1995), History (Squire, 2006) and Science (Barab et al., 2005).

According to a research and a case study (Long, 2007), the teachers that endeavour to improve the learning results could use a game that has been programmed as an educational tool in order to advance the learning experience and enhance students' performances. This study showed that, when the game Robocode was used for a project-based course in class, students enjoyed playing with it while learning and they produced very creative solutions for the given project (Bierre et al., 2006).

A representative encouraging suggestion involves the utilization of educational games in the classroom, which are gradually being accepted as a promising educational tool (Johnson et al., 2012; Johnson et al., 2013), because it is evidenced that they increase students' performance (Papastergiou, 2009; Sadler et al., 2013) and motivation (Connolly et al., 2012; Hwang & Wu, 2012). Studies show that educational games are used by any gender, ethnicity and socioeconomic level, and they also seem to foster skills and learning outcomes such as becoming more active and engaged (Ferguson & Garza, 2011). Additionally, they improve students' focus of attention, including their ability to observe and identify objects in both simple and complex systems (Green & Bavelier, 2007, 2012; Ventura & Shute, 2013) and become more creative (Jackson et al., 2012). Furthermore, educational games support students in becoming more receptive to experiences (Ventura et al., 2012; Chory & Goodboy, 2011; Witt et al., 2011), as well as increasing their scores (Ventura et al., 2012; Skoric et al., 2009), and determination (Ventura et al., 2012). Such games have been developed for many educational domains and usually incorporate a number of interesting features, such as attractive graphical interface, an engaging scenario, tasks/quests, interaction activities, characters etc.

Particularly, multiplayer online games can provide multiple benefits when utilized in education (Garzotto, 2007); for example, Massive Multiplayer Online Role Playing Games (MMORPGs) continue to be popular as they reinforce motivation and creativity and allow skill development, such as problem solving and communication (Beedle & Wright, 2007). The incorporation of MMORPGs in education is encouraged by related studies, since they can foster learning by enabling students to practice, think critically and plan strategically (Dickey, 2007).

2.2. Serious and Educational Games

The term "Serious Game" was first introduced by Clark Abt (1970), who worked in the American research lab during the cold war (Abt Associates, 2005). Abt aimed to utilize games in order to train and educate military officers. Towards this goal, he designed various computer games, such as T.E.M.P.E.R., which was used in order to study the worldwide conflict of Cold War 3. However, according to Abt, the term Serious Games is not solely addressed to digital technologies. The researcher gives as an example non digital games that can be applied in schools for the education of math. The

definition that Abt provides for the term "Serious Games" is "Games may be played seriously or casually. We are concerned with serious games in the sense that these games have an explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement. This does not mean that serious games are not, or should not be, entertaining." (Abt, 1970).

Other researchers that have studied this term provide non-digital examples of Serious Games, such as Jansiewicz (1973) in the book "New Alexandria Simulation: A Serious Game of State and Local Politics". This book includes a game designed to support teaching of the American political processes, and it is still being used in classrooms since 2004 in a non-digital format (Jansiewicz, 2011). The benefits of the incorporation of this game in the classrooms were pointed out by Kahn & Perez (2009), who noted that students of courses such as "Introduction to American Politics" performed better after using the game.

Serious games include a wide variety of activities where students can actively participate in order to fulfil specific or overall educational objectives. The relevant literature states that emotions are closely interrelated to actions (Frijda, 1986; Frijda, 2007), since people's emotions prepare them to act and they usually act based on their emotions (emotion readiness). To this end, we must consider that emotions play a very significant role in the structure and experience students will have when they use educational games.

The categorisation of serious games into a number of markets, provided by Michael and Chen (2005), is adopted. The markets are: military games, government games, educational games, corporate games, healthcare games, and political, religious and art games, (Michael & Chen, 2005). Despite such categorisations, notably many games could belong to more than one category.

Educational games are a novel learning method that has been gradually used for teaching various educational domains. Educational games are considered any serious games that primarily aim to teach instead of entertain (Michel & Chen, 2005) and that support "learning by doing" education.

The basic idea behind the development of educational games is the exploitation of existing successful outcomes, such as novel, state of the art computer games towards the creation of innovative and attractive learning experiences. The most important strength of

educational games is that they can support the educational needs of the "digital natives", i.e. individuals who have been accustomed to using computer games, watching movies and browsing the Internet from a very early age (Prensky, 2001).

Educational games are popular across all genders, ethnicities and socioeconomic lines. Furthermore, their usage seems to be linked with the development of positive skills such as attention and visual-spatial abilities (Green and Bavlier, 2007, Green and Bavelier, 2012 and Ventura et al., 2013), performance in the classroom (Skoric et al., 2009 and Ventura et al., 2012), creativity (Jackson et al., 2012), willingness to new activities and environments (Chory and Goodboy, 2011, Ventura et al., 2012 and Witt et al., 2011), motivation for success (Ventura et al., 2012) and engagement (Ferguson & Garza, 2011).

2.2.1. Emotions and serious games

Serious games include characteristics and features that significantly increase the possibility that students will experience positive emotions during learning, especially in comparison to the usage of more traditional learning technologies.

More specifically, well designed serious games include pleasant and attractive features that increase students' motivation to participate more actively in the learning process. Such features regard impressive graphics, interesting scenario, engaging activities, challenging and motivating tasks that stimulate competitiveness etc. This way, players feel more competent and are more aware of their skills and capabilities when they use serious games often and get familiarized with the different game elements (Anolli & Mantovani, 2011).

Thus, such positive emotions are directly related to the satisfaction players receive during the gaming experience. The different types of satisfaction identified include:

- Sensory enjoyment, which players feel through rich visual stimuli of the game (e.g. colors, forms, dimensions, sizes etc) and by their rhythmic succession (Anolli et al., 2010).

- Aspects related to the execution of the game provide satisfaction to players through the likelihood of successful implementation of the objectives or the achievement of challenges (Rozin, 1999; Anolli et al., 2010).

- Players that succeed in a task acknowledge that they have accomplished the goals of the game and have performed well, as well as the perception of the players that they have control of the situation offers them a sense of security within the virtual space. In this case the pleasure comes from the attainment of a goal that is not totally taken for granted, the so-called "functional pleasure" (Oatley 1996; Anolli et al., 2010).

- The satisfaction players feel is usually linked to curiosity and thus the desire to experience and explore something new (Anolli et al., 2010).

However, players can also experience negative emotions during their experience with a serious game. Such emotions usually regard:

- The possibility of failing to correctly execute activities
- The disappointment and possible frustration in the case of errors
- The anger in the case of dealing with obstacles that are difficult to overcome.

The aforementioned emotions can be successfully addressed with a variety of ways in order to avoid players' negative experience. For example, the simulation of real world activities within the games should be realized with entertaining actions that highlight the real world conditions and immerse students in the virtual world. This way, they can investigate all alternate pathways and solutions that are available in the game, without having to be particularly concerned with the possible consequences that their actions would have in the real world (Klimmt, 2009).

Furthermore, serious games allow players to manage their errors by providing a safe environment and helping them to reduce their anxiety regarding failure (Keith & Frese, 2008). Additionally, the environment's attractive world and graphics increase students' continuous engagement and focus, while the interesting activities included help in improving their performance.

Finally, the provision of feedback supports students by informing them of their mistakes in ways that can assist them to understand why the mistakes were made and help them give correct answers in the future.

2.2.2. Constructivist learning of serious games in education

Constructivism is a theory of knowledge (Piaget & Inhelder, 1967) that argues that humans generate knowledge and meaning from an interaction between their experiences and their ideas. During its infancy, constructivism examined the interaction between human experiences and their reflexes or behavior-patterns.

One of the best cases of constructivism in education is considered issue based learning, according to Savery and Duffy (1996). More specifically, in constructivist learning we can consider that:

a. The central idea in learning is understanding and being connected with the learning environment in order to in depth comprehend what is being taught.
b. Bewilderment is the boost for learning, motivating the need for further exploration of the confusing topic and the need for its clarification.
c. Knowledge evolves through social negotiation and collaboration as well as with the evaluation of what each individual has understood on the topic.

Based on the above, Savery and Duffy (1996) elaborate on issue based learning as an instructional model and demonstrate how this theory follows the principles of constructivism. They also state that the work done by Lebow (1993) significantly clarifies all the learning processes that occur in a constructivist learning environment.

These principles of constructivism are shown below (Lebow & Wager, 1994):

1. Link each learning activity to a specific issue or problem.
2. Support learners when they try to understand the general issue or problem.
3. Design appropriate and reasonable assignments.
4. Design tasks that simulate the real world conditions.

5. Shift the control to the learners so that they are responsible for each process they carry out and the corresponding results / answers.
6. Design the learning environment in ways that it will encourage, support, strengthen but also monitor learners' thinking and problem solving process.
7. Reinforce learners' ability to express their thoughts by presenting different perspectives and enabling collaboration.
8. Provide opportunities and support on the learning material as well as on the learning processes.

As shown above, both the suggestions of Savery and Duffy (1996) along with the principles proposed by Lebow and Wager (1994) can be applied in Games based learning and in the utilization of serious games in education, since they provide clearly defined guidelines on how computer games can be incorporated in a learning setting for computer programming courses.

It is important to emphasize that those learners that are more in contact with the learning content during learning are also more likely to remember the information taught. Hence, serious games should include learning content in all parts of the game experience and provide features such as interaction and feedback that will engage learners with the content in a continuous manner. This way, learners can comprehend all knowledge taught by using all the elements within a game which are organized based on constructivist principles.

Educational games should combine entertainment, creativity and technology so as to become effective tools for instruction. However, educational features (e.g. constructivist characteristics) should also be integrated, so that students will be able to successfully fulfil the set educational goals (Greitzer et al., 2007).

One of the most common and effective ways to incorporate constructivism in educational games is including scaffolding features. According to the constructivist theory, educational technologies should provide maximum scaffolding to students, so that they will be supported throughout their interaction with the tools (Sweller, 1988). That way, students can feel safer getting familiarized with a new environment with multiple learning contents and a variety of activities. However, the more knowledge

students receive and absorb, the more the scaffolding features should gradually decrease (Sweller, 1988).

Scaffolding in educational games can be achieved through providing mechanisms of detecting students' movements across the environment, so that everyone can have a full overview of the course of the game at any time. This also allows any student to ask or provide help to players in need. Other types of scaffolding include specific hints during activities when students fail to progress after multiple efforts, explanatory messages in case mistakes are made, as well as collaboration tools (e.g. chat, forums etc.) that allow easy communication between the players. This way, educational games become more personalized and adaptable, reducing the possibility that students will be displeased by their gaming experience and providing security within the environment.

2.2.3. Educational goals and educational games

According to Benjamin Blooms (1956) taxonomy of learning domains, there are three domains with several learning goals that should be examined across multiple axes so that their specification will be accurate and thorough. More specifically, the relevant literature recognizes three different domains that provide information regarding educational goals from different viewpoints:

- **Cognitive educational goals**: intellectual skills (Knowledge). There are six knowledge categories that represent different difficulty levels regarding comprehension and ease of recognition of knowledge in the long run. These categories are as follows:
    - **Knowledge**: recollection of information.
    - **Comprehension**: perception of the importance, translation and interpretation of instructions and problems. Declaration of a problem with our own words.
    - **Application**: Utilization of a concept in a new situation. We apply what we learned in class in new conditions, e.g. at the workplace.
    - **Analysis**: Separation of materials or concepts in partial components for more in depth understanding of their organizational structure.

- **Synthesis**: Construction of a structure using dispersed components to create a new meaning.
- **Evaluation**: Constructive criticism on ideas and materials of learning.

- **Emotional educational goals**: emotions or emotional areas (Attitude). The emotional domain describes all the ways in which we usually handle situations through our emotions (Krathwohl et al., 1993). The five specific categories of this domain, sorted from the more simple to the more complex are as follows:
  - **Receiving phenomena**: Awareness, willingness to listen, selective attention.
  - **Reaction to phenomena**: Active participation of students, who react to the phenomena presented to them. The stronger the motive to participate, the more intense is the reaction.
  - **Assessment**: The value that a person assigns to an object, a phenomenon or a behaviour (e.g. simple acceptance, full commitment).
  - **Organization**: Organization of values in priorities and creation of a specific and personalized values system, through comparison, correlation and synthesis of values.
  - **Internalization of values**: A person has a value system which controls his/her behaviour. This behaviour is consistent, predictable and distinctive of each student. The educational goals are affected to a significant degree by the different behaviours of students, and based on which ways and rhythms they adapt in new conditions.
- **Psychokinetic educational goals**: manual or physical skills (Competency).

Thus, effective educational games should incorporate cognitive, emotional and psychokinetic educational goals in such a way that they ensure learning units will be taught through the application of theory. This process will be carried out within the game, where learners will be able to view the learning content, and apply it in given activities or tasks that will be engaging, motivating and attractive. This remains a significant research challenge, especially in difficult domains such as computer programming.

2.3. Summary

This chapter analyzed different computer programming educational technologies. Also, the difficulties, and the mistakes that occur during procedural and object oriented computer programming learning and teaching were investigated and described. Finally, the chapter described effective teaching technologies of computer programming that have been suggested over the years, and analyzed computer games as educational software for teaching computer programming.

Furthermore, the "Serious Games" and "Educational Games" terms were described and the learning theory of constructivism was studied, which is strongly inter-related with the usage of serious games in education. More specifically, the relevant literature shows that effective educational games should always include different types of educational goals, such as cognitive, emotional and psychokinetic in order to ensure that all learning units will be taught efficiently through learning by doing activities. Towards this goal, the next chapter presents the design frameworks for educational games that have been studied for this work.

3. Designing educational games for computer programming

The availability of frameworks that provide adequate guidelines for designing and developing games for the educational domain is still considered to be a work in progress where more empirical studies are needed (Fisch, 2005; Dondlinger, 2007; Wong et al., 2007; de-Freitas, 2006). Even though a number of frameworks provide interesting concepts that should be taken into consideration, their practical application in designing educational games for specific learning domains and their corresponding evaluation and adaptation is still lacking.

This chapter aims to describe frameworks that have been constructed to guide the design of serious games. Section 3.1 presents an overview of existing frameworks that have been proposed by published works for the design and development of domain-independent and computer programming-specific educational games as well as specific features that should be supported. Section 3.2 describes the architecture of educational games for computer programming while section 3.3 describes the basic requirements for educational games. The chapter concludes with a summary of the work done.

3.1. Educational Games Frameworks

This sub-section presents a number of frameworks that have been proposed and constructed for the design of educational games, without however focusing on a specific educational domain (Malliarakis et al., 2014a).

3.1.1. Four – dimensional framework

The Four-dimensional framework has been proposed by de Freitas & Jarvis (2006), as shown in

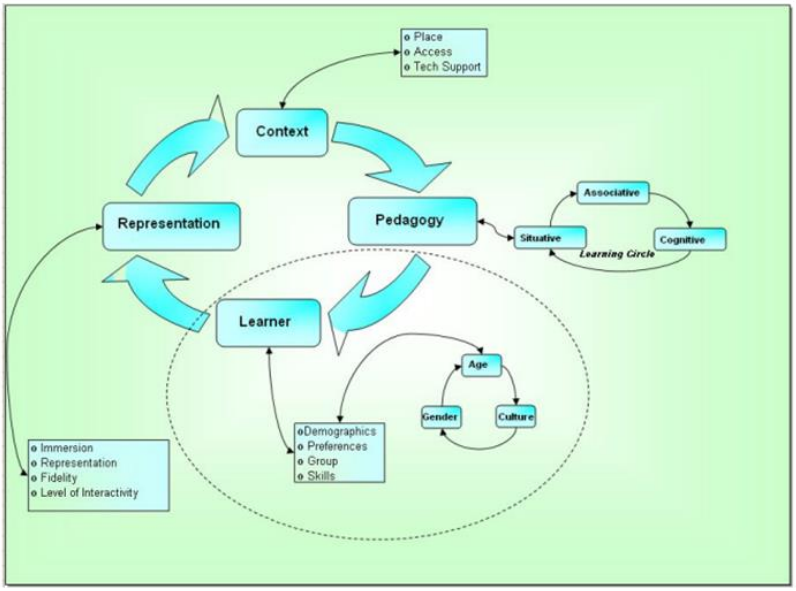

Figure 3-1: Four – dimensional framework (de Freitas & Jarvis, 2006)

The work done and portrayed within the framework shows that this model comprises of four basic principles, as follows:

- **Context**. Each game is characterized by a specific context that will guide the scenarios as well as the ways students and teachers will interact with its features. During the context's establishment, one must define characteristics such as required infrastructure, technical specifications, location of usage, type of game (e.g. role playing, multiplayer etc.), activities to be performed etc.
- **Representation**. This concept refers to all representations that are required to be properly portrayed within the game. For example, each player needs to be represented by avatars that will have specific characteristics based on the context of the game. Additionally, the virtual world should represent interesting scenery that will be integrated with all the features of the game in a harmonized and meaningful way. Successful representation is vital for the increased motivation of students, since they need to be intrigued in order to want to learn by playing. An important metric that can determine this is the quality of the graphics employed during the representation, which needs to be high so that it can create better and more immersive simulations.
- **Learner**. This concept relates to all features corresponding to the learner within the game. Some of these include the ages of the students to be taught, their

preferences, the availability and level of previous knowledge on the specific learning domain, the learning objectives regarding the game's learning outcomes etc. Additionally, it is considered important by the relevant literature to try and promote learning through groups in educational games (Sandford, 2006), as collaborative learning is gradually being employed in education (Kayes et al., 2005).

- **Pedagogy**. The most important factor that distinguishes an educational game from a computer game is the pedagogical aspect, i.e. the fact that the entire game and its activities are developed in order to fulfil learning objectives and to result in learning outcomes. To this end, the development of such games should depend on the study and incorporation of learning strategies. These strategies later on determine how the game will be integrated into the learning process so that it will produce the desired outcomes. Some representative examples of employed learning strategies are problem-based learning or experiential based learning, which allow a variety of pedagogical models for learning processes, especially when they use online technologies (Mayes & de Freitas, 2006). The pedagogies used should promote learning processes that will be constructivist and cognitive so that they will allow the creation of new knowledge by the students as well as their in depth comprehension of the taught material through active and collaborative participation. Additionally, learning processes have to be instructive and associative, that is they are required to progress in a logical manner and provide assistance when needed. Finally, collaborative learning when knowledge is exchanged through practice is also a feature that should be supported by learning processes within educational games (Mayes & de Freitas, 2004).

3.1.2. Conceptual framework

The second reviewed framework has been proposed by Yusoff et al. (2009) so as to provide a reference guide for the design of educational games, as shown in Figure 3-2.

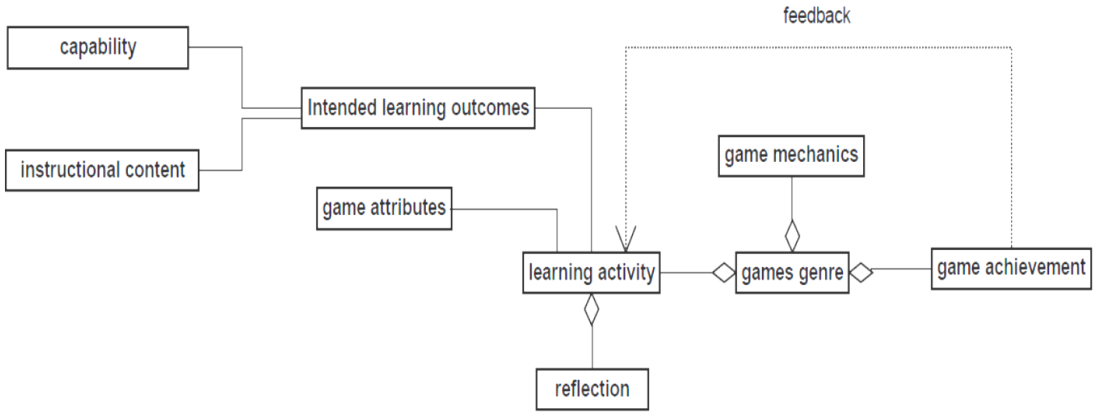

Figure 3-2: Conceptual framework (Yusoff et al. (2009)

A brief overview of the framework's features is provided as follows:

- **Capability**. This relates to the skills students should develop through their interaction with the game within the learning process. Such skills include analysis, recall, evaluation, effective execution of tasks, proper attitude and logical thinking.
- **Instructional content**. The game should be compliant with the educational material that students have to learn while interacting with it; therefore a thorough examination of the units of learning to be studied is essential and will determine the types of activities and assessment methods students will engage in during the game.
- **Intended learning outcomes**. Learning outcomes represent what goals students should be able to achieve once they successfully complete all assigned tasks. As it is depicted in Figure 3-2, a learning outcome is closely inter-connected with the determined capabilities and content. Different learning outcomes correspond to a particular set of capabilities and content.
- **Game attributes**. This concept includes all characteristics that aim to increase motivational and participatory learning. Such are scaffolding, i.e. assistance given to the students when needed; interaction, i.e. types of engagement and feedback required by the game towards the student and vice versa; learner control and sequence, i.e. level in which students can navigate across the virtual world based on assigned activities or on their own without any guidance, incremental learning; i.e. each learning outcome is achieved incrementally

through the execution of a set of activities and not all in the end of the game; rewards, i.e. incentives provided to students that accomplish their goals as an acknowledgement of their successful endeavours and to students that are close to accomplishing them as a motivation to try more; authentic learning, i.e. the virtual world and the activities that will simulate an interesting and attractive environment with which students are already familiar, either from their real life or from their interactions with commonly used computer games.

- **Learning activity**. Each activity plays an important role to the game and focuses on a specific set of tasks that need to be completed. It is important that all learning activities promote motivation so that students will remain interested and immersed in the game's scenario. The design of an activity should take into consideration the learning outcomes that have to result once students successfully execute the given tasks and thus should allow the development of the intended capabilities as well as the comprehension of the corresponding materials. Lastly, the established game attributes will define the activity development.
- **Reflection**. Students should be able to reflect on their experience within the game and be provided with an overview of their progress when requested (Garris et al. 2002).
- **Games genre**. This concept describes the type of the specific game that is to be developed. Specific genres are accompanied by different features so it is important to define from the design phase what type of an educational game this will be (e.g. strategy, open-world sandboxes, role-playing etc.).
- **Game mechanics**. This concept relates to the technicalities (e.g. management of resources, environment layout, database etc.) that should be taken into consideration during the game's development depending on the game's genre, learning activities or instructional content.
- **Game achievement**. The final concept refers to all the ways the game can represent a student's achievement level and is also a significant metric of learning assessment. For example, the final scores each student has gathered along with any additional resources or "rewards" provide an overview of individual achievement and thus show to which level the intended learning outcomes were accomplished.

3.1.3. The Design, Play, and Experience Framework

The "Design, Play and Experience" framework aims to depict the relation between the designer and the player and explicitly demonstrate the concepts that correspond to each layer that is designed depending on the phase and user type. More specifically, the designer focuses on the "Design' phase and has to initially identify the learning objectives that will guide the activities to be designed and that will be used as an evaluation metric once the development of the game is complete. Additionally, all other layers to be designed and shown in Figure 3-3 are initially based on the learning objectives and are later on configured and refined based on feedback from incremental and iterated engagement with the game, a process represented by the "Experience" phase. This is clearly depicted by the arrow connecting the "Experience" phase with the "Design" phase, expressing how continuous and practical engagement can affect the design phase of a next version of the game (Salen & Zimmerman, 2004).

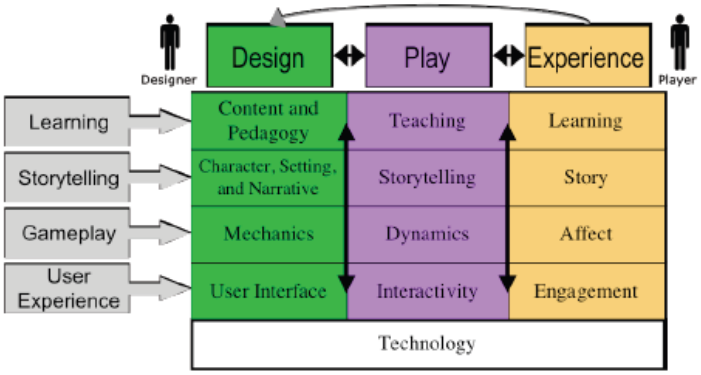

Figure 3-3: The Design, Play, and Experience Framework (Salen & Zimmerman, 2004)

The "Play" phase represents the interaction of players with the game's features. Thus, this phase is closely related to all characteristics of each individual player, such as knowledge background, skills etc. To this end, during the "Play" phase designers must take into consideration the target audience that will use the educational game and that will produce the different experiences during the "Experience" phase.

According to this framework, four different layers guide the design of an educational game. A brief overview of these layers is provided as follows:

- **Learning**. During the "Design" phase, the educational content that will be taught along with the pedagogical theory that will guide the learning process corresponds to the learning layer. In the "Play" phase, this refers to the actual teaching process, which is when students play the game. The "Experience" phase finally represents the learning that is accomplished through the teaching, and thus documents whether the set learning objectives have been achieved as learning outcomes.
- **Storytelling**. Storytelling provides valuable information during the game's design that will guide the virtual world development as well as all the scenarios to be supported within the game. The designer sets the stage by designing the different characters to be included, the overall environment setting, the narrative and the different layouts that will structure the world (Rouse, 2001). Even though the designer sets the story, each player produces his individual storyline through the way he engages with the game's activities during the "Play" phase. Similarly, the final player's story as it will be created after the execution of all assigned tasks will represent the accomplished learning outcomes in the "Experience" phase.
- **Gameplay**. This layer includes all information regarding the players' allowed actions within the game (Adams & Rollings, 2007). Initially, the designer has to define the specific mechanics of the game, such as the learning objectives, the challenges and allowed actions. Once these are integrated in the game, they are represented by the dynamics during game playing, i.e. all the different actions players take in the game that lead to individual pathways and interactions with the game's features. The experience drawn from the play by each user is called an "affect" and represents all emotions players are left with after they are finished (e.g. satisfaction, disappointment, desire to try again etc).
- **User experience**. The final layer of the reviewed framework includes the most visual part of the game, which has to be as entertaining and accessible as possible in order to increase motivation and participation (Saltzman, 2000). The design phase supports the planning of the user interface and aims to provide multiple and easy to use interactivity opportunities during game playing. Finally,

this will lead to experiences that engage students and accomplish in-depth comprehension of the learning material and successful skills development.

3.1.4. Experiential Gaming Model Framework

The experiential gaming model framework describes learning as a circular process that constructs cognitive schemas through activities within the game's world (Kiili, 2005). The experiential gaming model framework aims to help game designers to understand the learning mechanism that should be employed in educational games, as shown in Figure 3-4. According to this model, the direct interactions and experiences players have with the game world create a circular learning process that includes all necessary steps to ensure successful learning objectives achievement. To this end, the model suggests that the activities that will be supported by an educational game should not only be cognitive but behavioural also. This way, players will behave within the activities of the game in a way that will allow them to build cognitive structures.

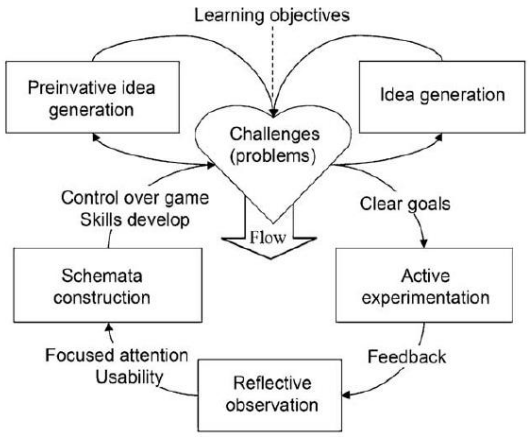

Figure 3-4: Experiential Gaming Model (Kiili, 2005)

This model includes concepts regarding challenges, experiences and ideas. It is often described because of its structure as a human blood-vascular system, with the challenges of educational goals in the heart of the model. This main concept feeds motivation and engagement to the students by providing challenging educational objectives. Students then need to overcome the challenges by generating ideas. The idea generation concept is divided in the model in two different loops, namely the preinvative idea generation (Finke, Ward & Smith, 1992) and the idea generation loop. The

preinvative idea concept regards the initial creative process of students (Maslow, 1963), where the ideas are chaotic, abstract and in general not structured.

According to the model, the challenges provided to the students should be in accordance to their skills and general competences so that it will be easier for them to find appropriate solutions.

According to Kiili (2005), important elements of an educational game should be the scenario that will set up the learning objectives, the challenges students will have to face as well as the flow of the game. Additionally, the world of the game should allow active experimentation with the environment's features so as to develop positive attitudes towards the game. Furthermore, feedback should be provided from the game during interaction that will allow students to reflect on their knowledge gained and to construct proper schemata and evaluate their own performance.

This experiential gaming model emphasizes that deep engagement of students in the learning process should be enabled in all of the game's functionalities; students should always be learning while playing, whether they are chatting with other players or reading an educational material or executing a task or reflecting on their performance etc.

3.1.5. EFM: Model for Educational Game Design

A multi-dimensional model called EFM (Effective learning environment, Flow experience and Motivation) has been proposed regarding the proper design of educational games by Song & Zhang (2008), as shown in Figure 3-5. According to this model, an educational game's environment should be able to support seven fundamental requirements that are highly interconnected with the other axes proposed in the model. These requirements include the availability of appropriate tools within the game as well as the sense of motivation along with the sense of direct engagement to activate students in the learning process. Moreover, the designers should ensure to avoid any possible distractions that could sway students' attention away from learning and enable a constant sense of challenge that will allow them to go through all of the activities scheduled by the teachers and accomplished the clearly set educational goals.

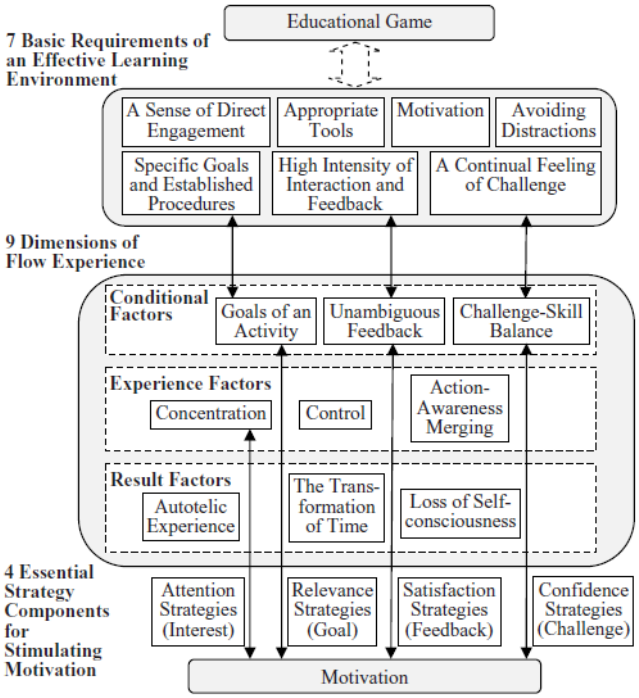

Figure 3-5: EFM: Model for Educational Game Design (Song and Zhang, 2008)

More specifically, the seven dimensions include:

- **Goals**. The educational goals set for the game should be specific, clearly defined and in accordance with the features available within the game, the knowledge and skills required by the students and the graphics of the game's environment.
- **Tools to provide help**. It is important to include hints as motivating aspects in the game in order to stimulate learners' curiosity and ensure they will get help when needed.
- **Avoiding distractions**. Appropriate control of the game can ensure avoiding of distractions so that all elements included in the game will help them get immersed in the scenario and activities and not lose focus.
- **Well defined processes**. All processes included in the game should be appropriate to the scenario with specific and motivating scenes and rules that help the game experience.
- **Sense of deep engagement**. Creating the sense of deep engagement for students will require the usage of realistic, interesting and multi-dimensional plots in the scenario.

- **Constant interaction and feedback**. Interaction and feedback amongst users and amongst users and the environment should be problem-free, continuous throughout the game's lifetime and accurate.
- **Constant sense of challenge**. It is important to constantly adjust the levels of difficulty within the game so that the challenges are appropriate for the specific students' skills.
- **Motivation**. Constant motivation within the game with the usage of features such as assignments, grading, awards etc. is important to ensure the learners' continued interest.

The model also suggests that these seven dimensions of the requirements will determine the experiences within the game and their flow. Finally, the authors indicate four main strategy components that will stimulate motivation for students, namely "Interest", "Goal", "Feedback" and "Challenge". The activities and experiences within the game's world should be designed so as to enable all these four motivation strategies in order to create an effective educational game.

In summary, the model suggests that an educational game can be considered and designed using the same principles and requirements as any effective learning environment that can stimulate motivation and engagement.

3.1.6. Educational Games Design Model Framework

The model proposed by Ibrahim & Jaafar (2009) includes three major factors that should be taken into consideration when designing educational games, as shown in Figure 3-6. The factors are game design, pedagogy and learning content modelling. Each of these is further analyzed within the model to provide analytical information on how to incorporate its features to the game.

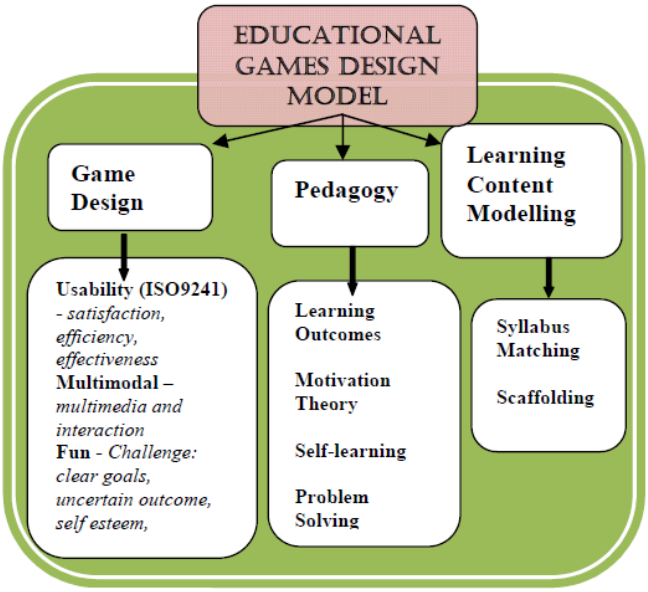

Figure 3-6: Ibrahim's and Jaafar's Educational Games Design Model Framework (Ibrahim and Jaafar, 2009)

Initially, the authors suggest that during game design, the usability of the game should be ensured, and more specifically tested under the ISO 9241 usability standard (Pinelle & Wong, 2008). An educational game is considered usable when it provides satisfaction to its players, it is effective in achieving the goals set before playing and it is efficient in allowing consistent and responsive functionalities. Moreover, it is considered important to allow multimodal content in educational games (e.g. text, audio, videos, animation, graphics etc.) as well as the ability for players to directly interact with this content and receive appropriate feedback. Finally, game design should also support the entertaining element of a computer game, which is essential (Prensky, 2001) because games should be fun to interact with even in educational settings. To this end, designers are expected to include functionalities that will allow teachers to set clear educational goals and activities that will challenge students during playing, will engage them in navigating through the game's features by increasing their curiosity even though they will not know the outcome (Malone, 1980) and will boost their self –esteem.

Furthermore, the authors include the pedagogical factor in the design model, which suggests that the computer games should be designed according to the educational domain and incorporate proper learning strategies to ensure that the game will indeed result in the desired learning outcomes. More specifically, the model is drafted so as to support learning outcomes that belong within the three first levels of the Bloom

Taxonomy, namely knowledge, comprehension and application (Ibrahim & Jaafar, 2009). These three levels of knowledge should be accommodated through appropriate theory availability and modules that motivate learners to reach the 3rd level of knowledge. Additionally, educational games designed according to this model should support self – learning and thus reflection mechanisms; these games allow students to teach themselves by playing, reading the learning materials and assess their own performance. Thus, the authors indicate that such games could be developed as web-based environments so that they can be accessed by learners at any time. The active role of learners in the educational games will increase their competences since traditional learning where knowledge is simply delivered will be replaced by a series of problem solving activities that will stimulate learners' minds and allow them to learn how to solve problems by interacting with materials and tasks within the game. This calls for specific types of learning materials that allow self-learning and problem solving.

Therefore, it is important to also consider the last factor included in the design model, namely the learning content modelling. The content incorporated within the game should be available at specific parts of the game in order to ensure the proper solving of the given problem and thus result in the set learning outcomes. Moreover, features such as a syllabus, terms matching tools and scaffolding mechanisms would increase learners' sense of security since they will mostly be learning on their own, with the teachers available only during lectures time to provide guidance.

3.2. Architecture of educational games for computer programming

The previous section elaborated on the thorough investigation carried out on frameworks and models proposed for the design of educational games. This research identified architectures that are further analysed and formed a list of important features that should be supported by educational games constructed for teaching computer programming concepts (Malliarakis et al., 2014a).

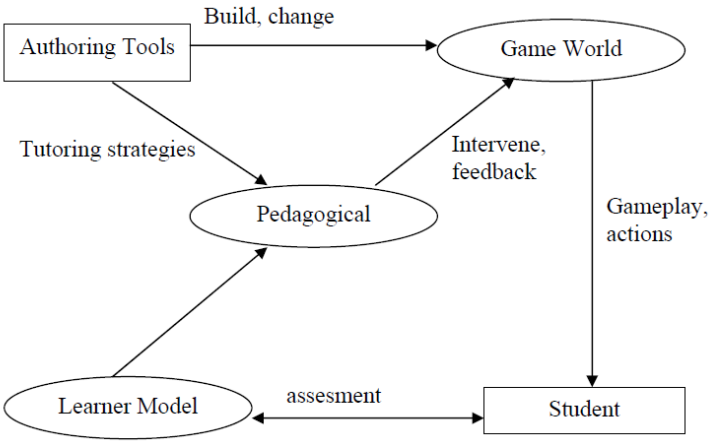

Figure 3-7: Architecture for Intelligent Educational Game (Maragos & Grigoriadou, 2005)

More specifically, the identified architecture shown in Figure 3-7 was proposed by Maragos & Grigoriadou (2005) for the design of intelligent educational games that aim to teach computer programming. This architecture was employed for the design and development of the educational game TALENT, which supports algorithmic if-statements and loops through interactive role-playing.

According to this model, the world of the game is constructed and can be changed through authoring tools by the teachers. These authoring tools should also allow the addition of pedagogies in the game's environment, such as tutoring strategies that support successful learning. This is usually done via a pedagogical agent, which is represented as a character or a tool within the educational game. The agent usually intervenes in the game's process and provides feedback whenever a student with a specific profile requires it.

Furthermore, the game supports multiple learner models that can be assessed according to each student's specific performances and profiles. These models are built in accordance to the pedagogical goals the teachers set while they are constructing the game's world and depending on the educational materials to be taught as well as the activities/ steps that the students have to go through.

All of the above components of the model suggested allow for great configuration rights by the teachers, thus providing abundant flexibility to create multiple different instances of the same game. Teachers can study their students' behaviours and

correspondingly adapt the game's scenario, materials, activities, goals and desired outcomes.

Scratch (Resnick, 2007) was based in a different architecture than TALENT, as it aims to create an environment that would assist creative people to easily accomplish their goals. This aim is also depicted in Figure 3-8.

Figure 3-8: Ultimate aim of the MIT Media Lab (Resnick, 2007)

The Scratch environment was based on the above design principles and towards this goal, students can create programs, interactive stories and games through drag & drop tiles instead of writing lines of code, thus avoiding making syntax errors. Interaction and scenarios play an important role in this game as they intend to stimulate the interest of young students by engaging them in a series of attractive assignments. The availability of multiple characters (sprites), backgrounds, sounds and images motivate students to create their own animations and games, play on their own and share their creations with their classmates, as well as reflect their creations to others.

3.3.Educational games requirements specification

A thorough study was carried out on the case studies that have proposed and developed frameworks for educational games (Becker, 2010; Freitas & Jarvis, 2006; Salen & Zimmerman, 2004; Yusoff et al., 2009; Zualkernan, 2006). According to these works, the development of an educational game should be carried out after the

examination of a number of aspects, where each aspect determines the features that should be supported in an educational game (Malliarakis et al., 2013a). All frameworks include similar requirements as concepts that are considered important. However, this research follows the one suggested by Becker (2010), because its concepts encompass all the ones included in the other frameworks. Thus, it is considered to be a more abstract superset of features that should be supported by all educational games.

Initially, it is suggested that the educational goals should be investigated across two axes by using a different viewpoint, so that their specification will be complete. These axes are:

Cognitive axis, in regards to mental competencies (Knowledge). Educational goals should make sure that the information received by the students begins from the first category in the Bloom's taxonomy (Knowledge) and ends in the final and most complex category (Evaluation) successfully.

Emotional axis, in regards to emotions or emotional areas (Attitude). Educational goals should enable students to handle given situations through the enticement of their emotions. For example, the need and thus the desire to free a prisoner during a game motivates students to solve the assigned task faster and correctly so that they can experience the emotions followed by accomplishing the goal.

Additionally, it is important to select a proper framework that will guide the learning experience with the incorporation of educational games. However, the framework that will be selected or constructed, according to the literature, requires the determination of various elements that together structure the framework's components. For example, the processes of the real world that will need to be simulated into the game (e.g. which movements will be allowed, how will the virtual world be constructed, how will the players be represented etc.) need to be clearly identified. This is a very important step as it determines the educational scenarios that can be supported by the game's environment and thus affects the entire learning process.

Following the framework's specification, an architecture should be constructed based on the components identified that have to be available in an authentic game environment. These components include:

The scenario's space. Students are introduced to the game's storyline once they logon to the environment with a short description of the plot as well as a brief overview of the basic activities they will have to execute.

Relevant cases. Students are provided with a set of pre-solved similar cases from which they can get a better insight of the game's knowledge and skills requirements and thus be better prepared for when it is their turn to solve assigned tasks.

Information resources. Students can access information relevant to the task at hand whenever they are in need of assistance.

Facilitating tools. A set of tools is included that students can use when they are trying to execute a task and that help them build new knowledge. Additionally, the game provides tools that underpin student communication as well as discussion with the teachers regarding any questions, thoughts or reflection on the virtual world.

Also, it is essential to distinguish **information** regarding the student, such as learning goals, learning style (e.g. holistic, analytical etc.) as well cognitive limitations (e.g. behavioural competences that can affect their learning).

Teachers should also be able to set **educational goals** that will have to be accomplished by the students during the game by assigning specific activities for them to participate in. This way, students achieve interim goals by successfully completing tasks that will lead them to absorbing the final learning outcomes set by the teachers.

The above features will underpin the selection of an **authentic scenario** that will provide an attractive story to go along with the game's virtual world. To this end, students should be presented with an interesting and motivating problem that needs solving and it is best if the plot is similar to the ones available in existing computer games. This way, students will be already familiar with the overall concept and the activities they will engage in will seem more like games than teaching exercises. Similarly, the individual **problems** that students will be called to solve during the game should be consistent with the educational goals as well as with any cognitive limitations that may be apparent to the teachers. This is one more reason why both these features are required to be supported in an educational game.

Another important feature is the constant and explanatory **feedback** provided to the students during their navigation through the game's levels. This feedback should be

represented in a form of messages that guide students towards understanding what they did right, what they did wrong and how they can achieve their goals, even though the mistakes they have made. This scaffolding technique ensures that students realize why their actions did not lead to successful task execution and thus they will be able to perform better in their next endeavour.

Finally, a number of **generic conditions** need to be taken into consideration while designing and developing an educational game. Such is the location and type of the education to take place (e.g. offline, online, blended learning). For example, if the learning process will be realized entirely online, then the game requires all the educational material to be uploaded within the environment and communication tools should be very well supported and plentiful. Moreover, the course's duration will determine how many hours students will spend on the game and how many scenarios need to be constructed according to the elements that need to be taught in this duration. All of the above require adequate preparation from the teacher and proper configuration of the environment so as to exploit all of the game's benefits and foster knowledge and skills development by the students.

3.4. Summary

In order to make sure the educational games include all necessary features that will ensure successful teaching and learning, it is essential that they are first designed properly. This indicates the need to follow an analytical guide that depicts all concepts that should be taken into consideration during the design and development of educational games. Thus, in this chapter we reviewed existing frameworks that describe such concepts and presented an overview of existing frameworks that have been proposed by published works for the design and development of domain-independent and computer programming-specific educational games as well as specific features that should be supported. The chapter also included information on architecture features for educational games and thoroughly described the basic requirements for educational games.

Summarizing, the most important features identified in the frameworks and that should be adopted by educational games are:

- Learning objectives and learning goals. They should be very clearly specified.
- Learning strategy that is employed based on these goals.

- Proper organization of the learning material.
- Motivating scenario and representative characters.
- Awards.
- Learning outcomes. They need to be clearly defined, such as what skills and knowledge that players will acquire, through which activities and with which scaffolding methods.

Furthermore, it is important to predict how players will become engaged within the game using a collaborative and interactive infrastructure. Finally, designers should also identify the game achievements and how these will be demonstrated to the players.

4. Educational games for computer programming education

This chapter presents a series of games that have been developed specifically for computer programming courses. The review of these games was carried out based on the specifications described in the studies where they were explicitly identified (Malliarakis et al., 2013a). Moreover, all chosen games have addressed both the cognitive axis and the emotional axis during their development. Thus, in each game students start out by receiving pieces of information, and through their engagement with the game, they move on up the Bloom's taxonomy to the evaluation step by reflecting on their progress and finalizing assignments. Also, the emotional axis is addressed via the game scenarios. All scenarios stimulate emotions that motivate students to go through all tasks in order to win.

Two major categories could be distinguished during the research, which sort educational games based on the educational goals they aim to support. The first category includes games that focus on teaching specific computer programming units while the second category represents games that cover multiple educational goals and thus computer programming material. A review for each category is presented in the next two sections, followed by a comparative analysis presented in Table 4.1.

4.1. Educational games focused on teaching a specific unit of learning

Catacombs. It is a three-dimensional multiplayer game that aims to teach students how to declare variables and use simple and nested if statements and loops. According to the game's scenario, each player represents a wizard that has to rescue two children trapped within catacombs. Towards this goal, the wizards have to answer multiple choice questions trying to solve a given programming code that will help them complete their quests. The answers to the given questions automatically create executable lines of code in a micro-language. If the answers are correct, the wizards progress through the game's levels; otherwise they are given corresponding feedback as to what they answered wrong and are prompted to try again. The game records experience scores for each student and provides explanatory messages as a scaffolding mechanism (Barnes et al., 2007; Barnes et al., 2008).

Figure 4-1: Screenshot of the game Catacombs (Barnes et al., 2007; Barnes et al., 2008)

Saving Princess Sera. It is a two-dimensional game that enables students' scaffolding through explanatory messages directed to the player. Each player has to try and save a princess named Sera who has been abducted by a monster named Gargamel, on her sixteenth birthday. Students are required to complete a number of quests in order to progress in the plot of the game. Towards this goal, they complete lines of code that will result to an executable program or they have to correctly map existing lines of code to their proper position or order within a program employing a drag & drop functionality. This way, students learn the quick-sort algorithm along with simple and nested loops with the usage of a micro-language (Barnes et al., 2007; Barnes et al., 2008).

Figure 4-2: Screenshot of the game Saving Princess Sera (Barnes et al., 2007; Barnes et al., 2008)

EleMental: The Recurrence. It is a three-dimensional game that aims to teach students how to execute recursion and depth-first search transversal using the C# programming language. The player has to navigate across a virtual binary tree by employing the depth-first transversal and complete three quests by applying recursion. Two avatars named Ele and Cera help students during the game in various ways. For example, once the code is written, Ele crosses the binary tree according to how the written code is deployed, while Cera explains exactly what the code is producing at a specific moment (Chaffin et al., 2009).

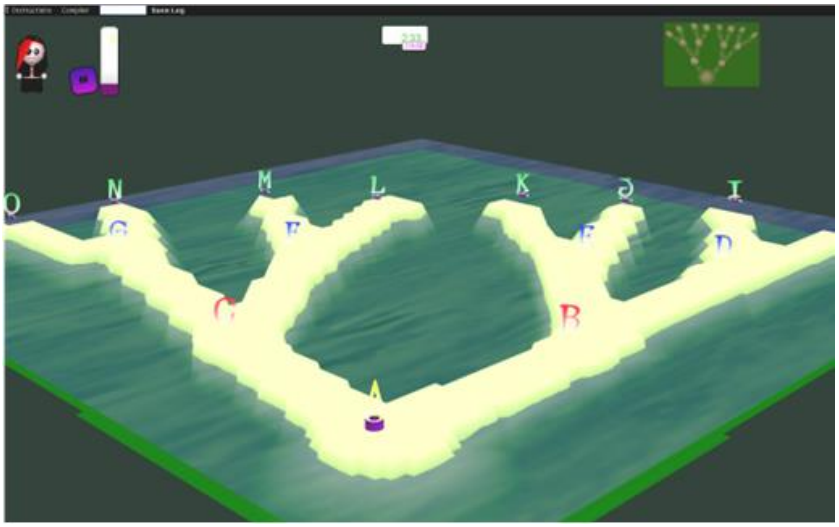

Figure 4-3: Screenshot of the game Catacombs (Chaffin et al., 2009)

Wu's Castle. It is a two-dimensional role playing game that aims to teach students loops and arrays through interactive activities. Each player is a wizard that can control an army of snowmen. Players recognize logical errors at lines of code written in the C++ programming language. The game allows arrays management through changing the parameters inside the loops and movement of the characters through the execution of nested loops (Eagle & Barnes, 2009).

Figure 4-4: Screenshot of the game Wu's Castle (Eagle & Barnes, 2009)

Robozzle. It is an online puzzle game that provides a series of pre-defined commands ready for use and does not show any actual code. According to the game's scenario, users have to build functions that will help them achieve each given task in a grid and tiles virtual world. Users can run their functions and see how their hero will move across the world and can therefore easily detect what mistakes they have made and re-program accordingly (Li & Watson, 2011).

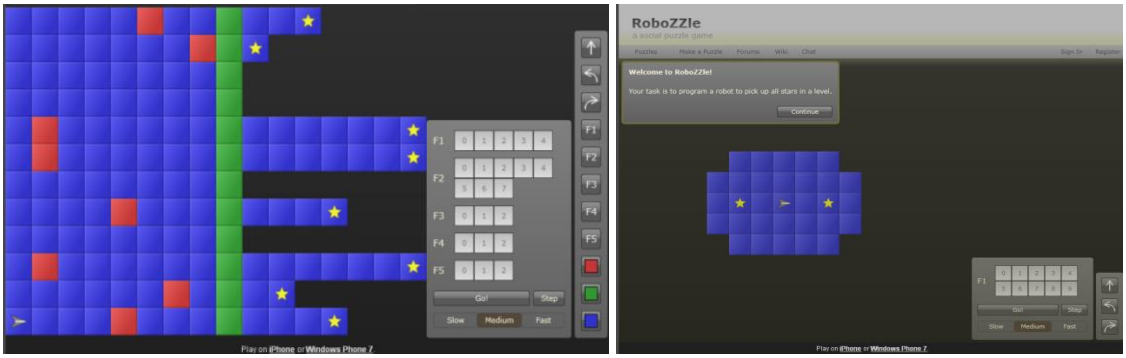

Figure 4-5: Screenshots of the game Robbozzle (Li & Watson, 2011)

LightBot. LightBot (Piteira & Haddad, 2011) is an online puzzle game similar to Robozzle. It includes a series of pre-defined commands and no actual code or programming language. Additionally, users have to complete given tasks by building their own functions and moving the hero across a grid and tiles environment and light all the blue tiles. Once a task is completed, the user can move to the next level, which requires the construction of more complex functions.

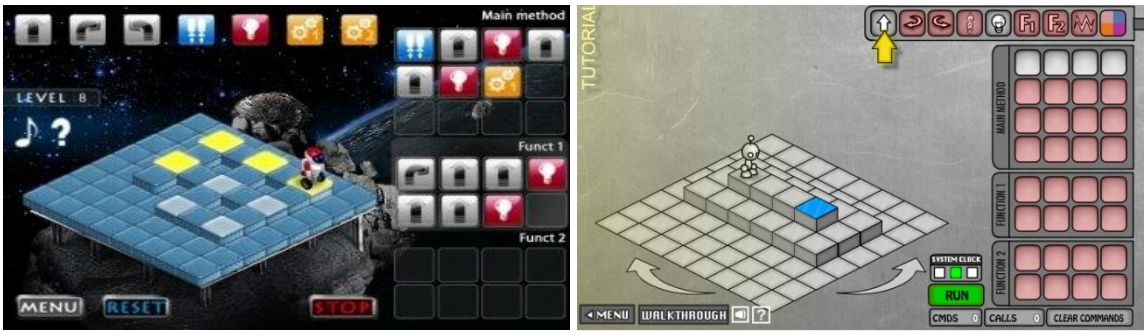

Figure 4-6: Screenshots of the game Lightbot (Piteira & Haddad, 2011)

TALENT. TALENT (Maragos & Grigoriadou, 2011) focuses on teaching if statements and loops in the forms of algorithms by using a micro-language. Each player is an archaeologist that has to navigate across the virtual environment by completing a series of tasks and collect objects that are available at specific locations for their future exhibition at a museum. Towards this goal, students can drag & drop lines of code as well as write them in an editor whenever requested. As a scaffolding mechanism, TALENT provides an agent that acts as a mentor and helps students when needed as well as suggest what their next mission should be.

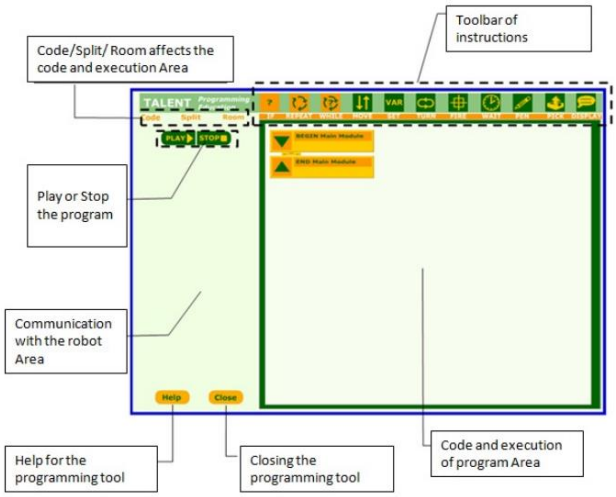

Figure 4-7: Screenshot of the game TALENT (Maragos & Grigoriadou, 2011)

4.2. Educational games focused on teaching multiple units of learning

Robocode. It is a two-dimensional environment that aims to teach computer programming using the Java language. The game comprises of a programming editor,

robots and a virtual arena. Students are required to program a robot that will compete against one another in the arena. Students familiarize themselves with the basic commands of structured computer programming and object-oriented programming (e.g. inheritance, polymorphism) while they try to build a robot ready for combat. During the construction of the program, the robot inherits basic methods that can later be extended by students according to the behaviour they want their robots to have inside the arena (O' Kelly & Gibson, 2006).

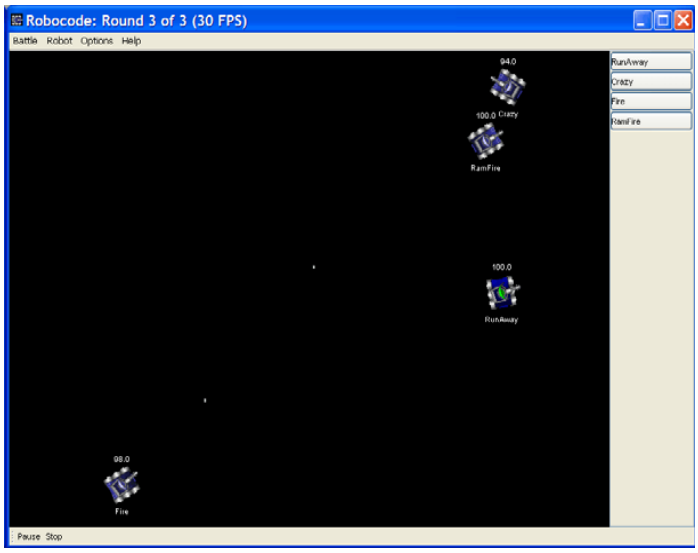

Figure 4-8: Screenshot of the game Robocode (O' Kelly & Gibson, 2006)

M.U.P.P.E.T.S. It is a three-dimensional, web-based and collaborative game that aims to teach object building and in general to familiarize students with the basic concepts of object-oriented programming using the Java language. Students create a robot that has to fight another robot inside a virtual arena, interact with the objects they build and write and compile their lines of code within the embedded development environment that includes a commands console (Phelps et. al, 2003).

Figure 4-9: Screenshots of the game M.U.P.P.E.T.S. (Phelps et. al, 2003)

Prog & Play. It is a library currently integrated in the web-based, real-time multi-player strategy game Kernel Panic, which enables constant interaction amongst users. Students can program their own heroes and form alliances with each other aiming to prevail in the game. Prog & Play allows students to choose the language in which they prefer to code their programs amongst programming languages such as Ada, C, Java, OCaml, Scratch and Compalgo (Muratet et al., 2010).

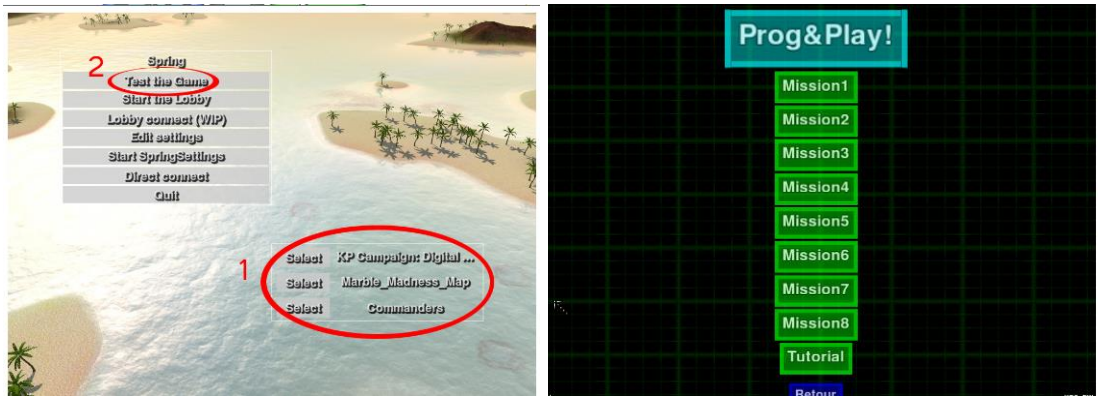

Figure 4-10: Screenshots of the game Prog & Play (Muratet et al., 2010)

PlayLogo3D. It is a three-dimensional, role playing game that allows interaction amongst multiple users and aims to teach basic concepts of structured computer programming. Users are required to program their heroes by writing the corresponding lines of code in the LOGO language, and navigate them across the environment. More specifically, the virtual world consists of the spaceship X-15 located on a constellation of

the Andromeda galaxy, where a contest is held each year amongst pilot-robots (Paliokas et al., 2011).

Figure 4-11: Screenshot of the game Playlogo 3D (Paliokas et al., 2011)

Gidget. It is a web-based game where students can program using a simplified programming language created specifically for the game in order to learn how to design and analyze basic algorithms. A robot named Gidget has problems with a part of his software and thus cannot complete its tasks. Therefore, students are called in to help Gidget by either fixing wrong lines of code, or by completing missing code within given programs. During these processes, students receive constant feedback of their progress (Lee & Ko, 2011).

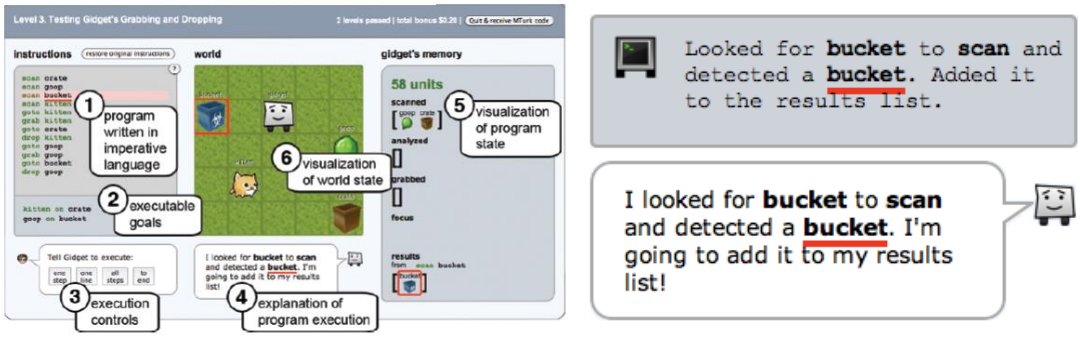

Figure 4-12: Screenshot of the game Gidget (Lee & Ko, 2011)

The following table presents the study with a representation of features supported by the most commonly known educational games for computer programming. The programming elements, characteristics as well programming activities identified can be

considered as concepts that describe the field. The Table summarizes the games that were implemented for the teaching of computer programming by recording features such as programming elements, programming language, programming activities and special characteristics.

Table 4-1: Overview of the educational games for computer programming courses

Game	Programming elements	Programming language	Programming activities	Special characteristics
Catacombs	Variables; Simple and nested if statements; Loops	Micro-language	Multiple choice questions; Filling out lines of code	Three-dimensional; - Multiplayer; Success scores; Scaffolding with explanatory messages from the hero
Saving Sera	if statements; Recursion	Micro-language	Filling out lines of code; Mapping parts of code in corresponding locations; Multiple choice questions	Two-dimensional Scaffolding
EleMental	Recursion; Depth-First Search (DFS) algorithm	C#	Depth-First Search algorithm; Moving the hero on a fantastical binary tree	Three-dimensional; Scaffolding
Prog & Play	Structured computer programming, If-statements, Loops	Ada, C, Java, OCaml, Scratch, Compalgo	Completing eight missions	Multiplayer; Infinite scenarios; Available
Wu's Castle	Loops; Arrays	C++	Arrays management; Movement of the hero; Identification of logical mistakes in code	Interaction; Role playing
Robozzle	Functions	No code used	Creating functions through the hero's movement	Interactive; Available
LightBot	Functions	No code used	Creating functions through the hero's movement	Web based; Available
TALENT	If statements; Loops	Micro-language	drag & drop lines of code; Writing lines of code in an editor	Explanatory messages
Robocode	Structured computer programming& object-oriented programming	Java	Writing lines of code	Every robot stores data of past activities; Available
M.U.P.P.E.T.S.	Three-dimensional objects; object-oriented programming	Java	Interaction with the developed objects; Writing lines of code; Compile	Collaborative; Three-dimensional; Commands panel; Embedded development environment
PlayLogo 3D	Basic concepts and commands of structured programming	Logo	Creation of heroes; Navigation across the "galaxy" by writing commands	Interaction; Multiplayer; Three-dimensional; Available
Gidget	Analysis and design of basic algorithms	Micro-language	Fixing problems and programs; Interaction with a personalized robot	Web-based; Explanatory messages

4.3. Discussion

This section discusses how well the games presented in this chapter fill the requirements that were set in section 3.3. The results are presented below, categorized based on the features identified that should be taken into consideration during the design and development of an educational game for computer programming.

Educational goals. The educational goals seem to cover both the *cognitive* and *emotional axes*. Within the games, these goals are clearly focused in the computer programming concepts the games aim to teach. This is especially the case in the educational games that cover specific units of learning, and thus the desired learning outcomes are more clearly identified. The emotional goals seem to be accomplished through the numerous attractive scenarios available in each game.

The ***problems*** students are required to solve are consistent with the set educational goals and their cognitive limitations. In the educational games focused in specific units of learning, students execute and complete quests that teach them knowledge that is relevant to the programming concepts set in the goals. As an example, the simple and nested loops in Catacombs are taught through the completion of lines of code, and their correct syntax allows students to pass to the next level, while the same concepts are taught in Wu's Castle when students move their characters across the world and recognize logical errors. On the other hand, the second category of educational games (e.g. Robocode, M.U.P.P.E.T.S, PlayLogo 3D) employs problems that allow students to interact with each other and execute multiple tasks that will teach them all the basic concepts of computer programming.

Framework. Educational games that focus on specific units of learning seem to have properly defined a framework for their employment in educational contexts. However, games that teach multiple and more complex units of learning, and thus cover multiple educational goals usually set several frameworks. It should be noted that the games Lightbot and Robozzle do not define any framework.

Scenario's space. All educational games present and work based on a *scenario* in order to attract and motivate students. In some cases *introductory information* is provided to the players in regards to the virtual world (e.g. PlayLogo 3D, Wu's Castle).

Information resources. Most educational games provide *explanatory messages*. The games where this feature is more fully supported are Catacombs, Saving Sera, EleMental: The Recurrence and Wu's Castle. Moreover, *scaffolding techniques* are provided through these explanatory messages that appear while students are trying to solve their quests (e.g. Catacombs, Saving Sera, EleMental).

Facilitating tools. Tools where students can write requested lines of code exist in the Catacombs, Saving Sera, EleMental, Robocode and M.U.P.P.E.T.S games. In addition, multiplayer games (e.g. PlayLogo 3D, M.U.P.P.E.T.S., Prog & Play, Catacombs) include features where students can *communicate and interact* with one another.

Generic conditions. The generic conditions have been taken into consideration in some of the studied cases. This has been carried out more efficiently in the educational games that cover specific units of learning rather than in the ones that teach multiple and complex computer programming concepts. On the other hand, they have not been considered at all during the design of the Robozzle and Lightbot games.

It should be noted that none of the studied games provides ***relevant cases*** that can prepare students for the activities they will be required to execute. The existence of this feature would significantly increase the quality of the games, since it would provide useful tutorials and guidelines for learners.

We also have to mention that many of the aforementioned information regarding these games derive exclusively from the relevant literature, since they are not available for access. This fact also results in our inability to exploit them in the learning process and actually test them against set educational goals in computer programming courses. Summing up, it seems that all studied games include scenarios that motivate learners, clearly indicate the educational goals that need to be reached and include problems that are set up as specified above. Other features, such as facilitating tools, information resources, one or several frameworks and taking into consideration generic conditions are supported by the majority of the studied games. However, none of the games appear to support relevant cases to prepare learners before engaging with the environments or to act as manuals for when learners require guidance.

The main implications derived from the analysis include that most games have been developed to cover programming concepts such as variables, simple and nested if

statements, loops, arrays, functions) with the exception of M.U.P.P.E.T.S. and Robocode that cover more complex concepts such as object oriented programming. We do not consider the fact that the games do not tackle all programming concepts as a disadvantage, since they seem to successfully fulfil the educational goals they set regarding the group of concepts they aim to teach. Also, the study elaborated on the added values of using educational games in computer programming, while a number of interesting principles that help us understand why educational games can improve teaching and learning of computer programming have been derived during our research. For example, games seem to have a facilitating role in the learning process during the teaching of specified concepts and could play a small or a big part in the entire course's implementation process, depending on the generic conditions. More specifically, depending on the nature of the course (online, offline, blended), materials, communications, exams etc could be supported on different levels by the games' environments. To this end, educational games can provide a number of characteristics, such as storytelling, scaffolding and interactivity, which increase motivation for participation in class as well as attract students to complete their tasks through interesting scenarios.

More specifically, the case studies that tested the games based on what educational value they bring forth were examined and the results indicate that they provide students with:

- ✓ Clear educational goals and learning outputs, ensuring that they know what they have to do to achieve the required knowledge and skills.
- ✓ A familiar and immersive environment that attracts students' attention, facilitates their active participation and increases their motivation.
- ✓ Interesting scenarios with comprehensive problems they have to solve, which enable them to learn in a contextual manner (learning specific units of learning periodically).
- ✓ Tools that help them communicate and collaborate with their classmates, improving their group work skills and guiding them through the learning process by explaining the possible mistakes they make. These tools can be applied either with chat functionalities or with different types of interactions between the learners and the game while trying to achieve and fulfil common goals.

Furthermore, the study's findings have implications regarding the design of future educational games focused on computer programming, listing and elaborating on the requirements educational game designers and developers should strive to support and thus setting the foundations for future holistic environments.

Educational games can also assist teachers teach programming in their courses by designing the game and setting up its parameters. For example, teachers can use educational games to plan their courses and monitor students' interactions, progress and evaluation through their activities in the game. The establishment of the educational goals, learning outcomes and the setting up of a scenario that will delineate the curriculum into units of learning also enables teachers to be better prepared and have a deeper knowledge of the materials they teach and get more skilled in course planning.

On the other hand, a significant limitation identified in the existing educational games focused on computer programming courses is the ability of the teacher to configure the environment according to the pedagogical goals related to the respective unit of learning. Additionally, the collaboration concept could be reinforced and better supported within multiplayer educational games so that they can teach more complex programming concepts that will be more efficiently understood through team-based learning activities.

4.4. Summary

In this chapter a series of games that have been developed specifically for computer programming courses are presented. The review of these games was carried out based on the specifications described in the studies where they were explicitly identified.

Moreover, all chosen games have addressed both the cognitive axis and the emotional axis during their development. Thus, in each game students start out by receiving pieces of information, and through their engagement with the game, they move on up the Bloom's taxonomy to the evaluation step by reflecting on their progress and finalizing assignments.

Also, the emotional axis is addressed via the game scenarios. All scenarios stimulate emotions that motivate students to go through all tasks in order to win. However, the comparison of the games using the requirements set in Section 3.3 also identified specific gaps, since none of the identified games supports all functionalities

that lead to an immersive and successful educational game for computer programming. These gaps are aimed to be addressed and fulfilled by the educational game developed in the context of this thesis and thoroughly described in the following chapter.

5. CMX's overview

This chapter introduces the proposed game called CMX, an MMORPG for teaching and learning computer programming. More specifically, it initially elaborates on the methodology that is followed for the development of the game. This methodology derives mainly based on the literature review carried out on related work and aims to address the gaps identified and also include the promising features some of the aforementioned games support. Additional information regarding the game is provided, such as a prototype of its architecture and a few mock up illustrations of its functionalities.

This chapter also describes the methodology followed during the decision making process for developing the game's architecture and scenario. Finally, detailed presentation follows regarding the scenario behind the game and a few of the supporting features, as well as the learning analytics that are integrated.

5.1. CMX Design Framework

This section provides information regarding the design process of our CMX educational game that focuses on the education of the computer programming domain (Malliarakis et al., 2014a). An important step that needs to be realized is the establishment of the game's infrastructure, which follows a standard MMORPG structure. As shown in Figure 5-1, the server administrator communicates directly with the server, which documents and stores the players' status, the world's conditions, all actions as well as any chat logs. Through internet connection, the server transmits instances of the game to multiple individual clients. Apart from the internet connection, a defining factor in the system's performance is the smooth execution of the server, since it transmits and receives a large amount of information regarding each player at any given time that the game is operational.

It is essential to initiate the design of the game by studying its characteristics and defining generic metrics that need to be instantiated based on the different user type. Therefore, a user-centric model was initially defined for the design of educational games, as shown in Figure 5-1. Based on this model, the design phase takes into consideration

each user type's initial aspirations that motivate them to use the game, the specific targets they set to achieve through the game, the metrics that will determine the level in which the targets are achieved as well as the feedback provided to them by the system depending on the corresponding target.

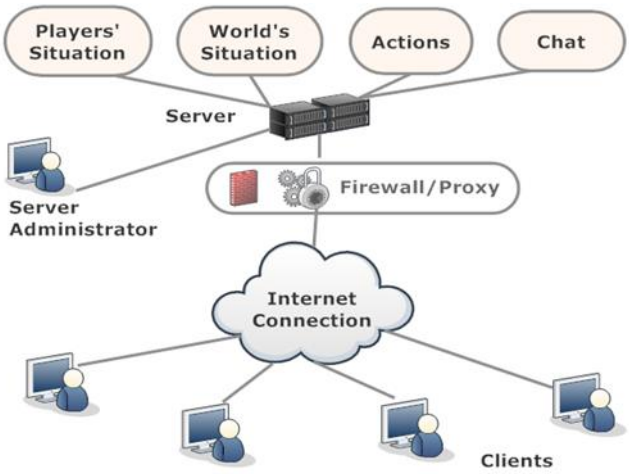

Figure 5-1: MMORPG infrastructure

These inter-connected concepts are different for each user type and therefore relate to different aspects of the game. For example, the game administrator's aspirations can be the game's and server's problem-free operation and the corresponding targets can be the backup of the data produced, the frequent monitoring for possible malfunctions etc. Performance metrics could include the server's database capacity, the cache memory capacity etc., while feedback that will indicate whether the server is working properly can comprise of error messages produced by the game (e.g. in case new data cannot be saved).

Similarly, a teacher's aspirations can be the successful teaching of units on computer programming while the target can look like "at least three of the four game levels on loops will be achieved by all". Thus, the set metrics can be each level's and student's score, log files from the game etc., while the feedback can be a report with the student's progress at a given time/level.

Furthermore, a student's aspiration can be to enjoy his time within the game, to be distinguished through accomplishing the assigned tasks and to learn. Based on these aspirations, they can set their individual or group targets across the overall learning objectives. Their performance metrics can be the score/resources they have gathered,

how many times they have had to retry before writing lines of code correctly etc. Finally, feedback can be explanatory messages from the agents, introductory narrative or video that explains the scenario, errors in the program they're writing etc.

This closely interconnected puzzle is shown in Figure 5-2 and provides a more abstract overview of the initial metrics taken into consideration during design phase, following a user-centric approach. This model takes the form of a puzzle since all user types need to be addressed equally and represent a different point of view of the game.

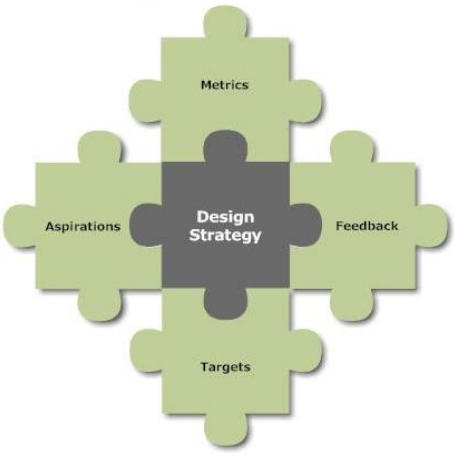

Figure 5-2: CMX Design Strategy Puzzle

The next step that will lead to the proper construction of our framework is the identification of the methodology our game will be based on. More specifically, as shown in Figure 5-3, students will initially engage in actions, which will in turn generate their desire for improvement through attractive scenarios. These actions will help students gain new knowledge regarding the targeted educational content and through collaborative activities, students will be able to produce the set learning outcomes. In the end, the learning objectives will be achieved after the game experience is properly evaluated.

Figure 5-3: Methodology steps of the CMX educational game

More specifically, the game always starts with a specific action. This action is attractive enough that it highlights students' desire to improve by participating on the action's given steps. Through this motivation process, students discover knowledge within the game. This is realized through a series of tasks that provide segments of computer programming information. These tasks are completed through collaborative work. This way, the game not only supports students gaining knowledge, but also the development of various skills (e.g. group work, task allocation, organization etc.) that will benefit them in their future professional lives. The constant collaborative undertaking of these tasks leads to learning of the taught elements. This learning then is evaluated with a number of assessment measures established by teachers. If the evaluation results are above the set average, then it is considered that the targeted learning outcomes are achieved.

For the successful navigation through the methodology showed, all parties affected during the game experience were taken into account in order to try to best support them in achieving their individual goals. For example, students aim to play to win, get distinguished and in parallel to learn programming by enjoying the process. The game's goal is for students to achieve these aims and to be able to communicate and collaborate with each other in an engaging manner towards understanding computer programming concepts. To this end, in the initial version of the game a series of scenarios have been produced for each computer programming unit of learning.

However, the parties affected by the game are not necessarily only the students. Unlike the games reviewed in Chapter 4, teachers also have a very active role in CMX. More specifically, teachers are able to configure parts of the game according to the learning objectives they want to address in each lesson. Moreover, they can set the goals that will determine the winner gamer and will manage the game by setting their own rules to ensure smooth transactions. Teachers also regulate the game so that it serves as supplementary software during computer programming teaching, in accordance to the corresponding taught unit of learning.

Lastly, the game significantly affects the delivered computer programming knowledge and skills. Although these are concepts and not individuals, they should be a part of the methodology followed during the design and development of an educational game. A visual representation of these stakeholders' goals is provided in Figure 5-4, called "CMX Programming Prism".

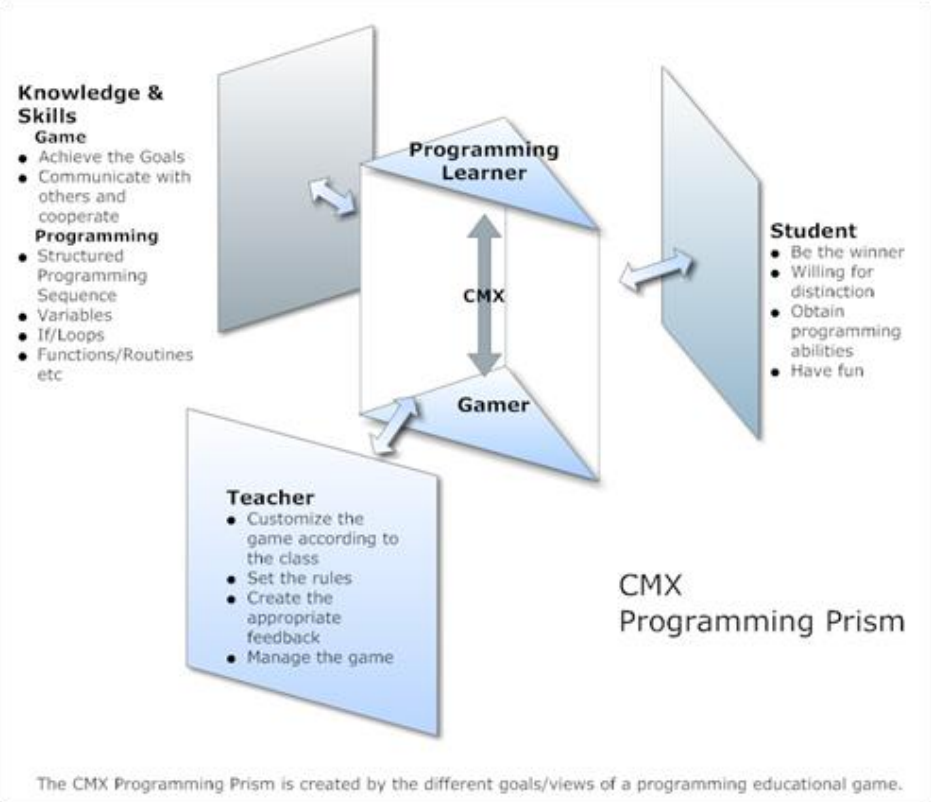

Figure 5-4: CMX Programming Prism

As it is shown in the Figure 5-4, each stakeholder has a different view of the way the game will be used for its corresponding goals' achievement. The game aims to deliver a number of skills to students by enabling easy communication and tasks' execution for goals achievement, while programming knowledge (e.g. loops, if statements, functions etc.) will be delivered during playing.

Finally, based on the review of the existing frameworks that guide the design of educational games in general as well as of the features supported by educational games focused on computer programming education, we move on to propose an extended framework that includes the most commonly identified concepts as well as a series of sub-concepts that provide a more in-depth visualization of the steps to be followed during the design and development process. Based on this framework, which is shown in Figure 5-5, we have designed and developed CMX.

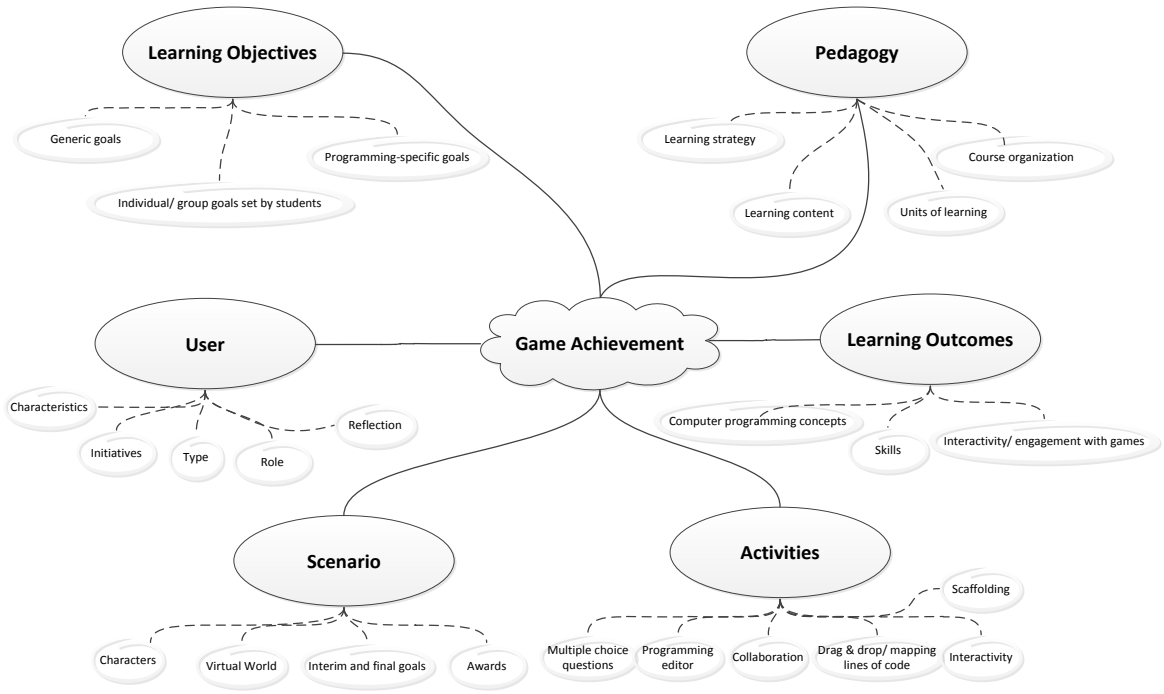

Figure 5-5: CMX Design framework

The CMX Design framework includes concepts that need to be represented within any educational game that aims to teach computer programming. It is abstract enough to be employed by future designers and developers and detailed enough to act as a solid guide without allowing many arbitraries. The most prominent concepts that define the game's design are:

- **Infrastructure**. The design initiates with the establishment of the infrastructure architecture, the technical requirements specification as well as the user interface and concepts visualization design. The infrastructure will have to support simplicity and ease of use.
- **Learning objectives.** Designers will have to initially define learning objectives that the game will be required to successfully support. These objectives can include generic goals (e.g. more than 80% of the students will complete all game's activities within the given deadline) or programming-specific goals (e.g. the 90% of the students will drag & drop correctly the lines of code regarding if statements). Additionally, students can also define their own goals depending on what they desire to achieve through their interaction with the game (e.g. "I have understood if statements so I want to complete all tasks related to loops").

- **Pedagogy.** The game's layout will strongly depend on the learning strategy to be employed during class as well as on the way the course will be organized (i.e. number of units of learning, educational material to be taught etc.). For example, it is possible that the teacher will want to create a level per unit of learning, or merge specific units of learning into one game that will correspond to a certain set of learning objectives. Thus, it is essential to determine these features at an early stage.
- **Learning outcomes**. The determination of the learning outcomes is strongly inter-connected with the set learning objectives. Target outcomes can include the comprehension of the taught material, i.e. computer-programming concepts, such as variables declaration, if statements, loops etc., skills development, such as critical thinking, teamwork, leadership etc. as well as interaction capabilities and engagement with innovative technologies and their features.
- **User.** A user in an advanced educational game can represent different types (e.g. student, teacher, administrator, agent), where each type signifies another aspect of the game's functionalities, as already explained in the CMX design strategy puzzle. Additionally, the specific characteristics of each user need to be determined (e.g. age, prior knowledge, preferences etc.) as well as the aspirations that will drive the user's interaction with the system (e.g. win the game-student, ensure problem free operation-administrator, instruct and assist students-teacher, guide and scaffold players-agent). Finally, each user will need to be able to reflect on the game experience for feedback gathering and future refinement of the game's functionalities.
- **Scenario**. The game's scenario should be thoroughly researched and planned out in order to produce an attractive and immersive virtual world (e.g. fighting arena, castle, forest etc.) with interesting characters (e.g. wizards, robots, mentors, snowmen, prisoners etc.) that are required to complete interim and final goals (e.g. save the princess, put the map's pieces in their correct order, navigate through the tree etc.). Designers have to also define what types of awards players will be granted with during the game, in order to increase their motivation to continue learning.
- **Activities**. The design and development of individual activities is essential and will result to the interested and active participation of students. It is important that students will be able to interact with the world's elements and collaborate

with others towards the achievement of all or some of the goals. Additionally, the environment should provide scaffolding mechanisms throughout all activities that will assist students during challenging tasks. Finally, a number of different ways in which students can contribute their knowledge should be included, since not all students learn better using the same techniques. Thus, an educational game should incorporate a programming editor, along with the ability for students to drag & drop lines of code as well as answer multiple choice questions.

The following table lists all features included in our proposed framework and depicts which features are supported by the corresponding layers of each of the seven frameworks studied during this research (see Chapter 3, Sections 3.1 and 3.2, pp.39-53) .

Table 5-1: Comparison of features supported by frameworks for educational games' design

CMX design framework	Four – dimensional framework	Conceptual framework	The Design, Play, and Experience Framework	Experiential Gaming Model Framework	EFM Model	Educational Games Design Model Framework	Architecture of Intelligent Educational Game
Learning objectives	X (Learner)		X (Learning)	X (Situated learning objectives)			X (Authoring tools)
Generic goals	X (Learner)		X (Learning)	X (Clear goals)	X (Goals)	X (Clear goals)	
Goals set by students							
Domain-specific goals					X (Specific Goals)		
Pedagogy	X (Pedagogy)					X (Pedagogy)	X (Pedagogical)
Learning strategy	X (Pedagogy)					X (Problem solving)	X (Pedagogical)
Course organization				X	X (Dimensions of flow experience)		X (Authoring tools)
Learning content		X (Instructional content)	X (Learning)	X (Design knowledge)		X (Learning content modelling)	X (Authoring tools)
Units of learning							
Scenario	X (Representa-tion)	X (Game attributes)	X (Storytelling)	X (Frame story)	X (Established procedures)		X (Authoring tools)
Characters	X (Representation)		X (Storytelling)				X (Authoring tools)
Awards		X (Game attributes)					
Interim/final goals			X (Storytelling		X (Goals of an activity)		

			)				
Virtual world	X (Context)	X (Game attributes)	X (Storytelling)	X (Implementation)	X (Appropriat e tools)		X (Game world)
Learning outcomes	X (Learner)	X (Intended learning outcomes)	X (Learning)	X (Solution generation)		X (Learning outcomes)	X (Learner model)
Skills		X (Capability)		X (Active experimentation)	X (Challenge-Skill balance)	X (Pedagogy)	
Knowledge			X (Learning)	X (Design knowledge)		X (Pedagogy)	X (Authoring tools)
Engagement		X (Game attributes)	X (User experience)	X (Schemata construction)	X (Sense of direct engagement)	X (Game design)	X (Game world)
Activities	X (Context)	X (Learning activity)	X (Gameplay)	X (Active experimentation)	X (Appropriat e tools)	X (Game design)	X (Authoring tools)
Scaffolding		X (Game attributes)		X (Feedback)	X (High intensity of feedback)	X (Learning content modelling)	X (Game world)
Interactivity	X (Representati on)	X (Reflection)	X (User experience)	X (Feedback, Reflective Observation)	X (High intensity of interaction)	X (Game design)	X (Game world)
Drag & drop						X (Learning content modelling)	
Programming editor							
Collaboration	X (Learner)						
Multiple choice questions							
Infrastructure	X (Context)	X (Game genre)	X (User experience)				
User interface	X (Context)	X (Game mechanics)	X(User experience)				X (Student)
Compiler							
Game achievement		X (Game achievement)	X (Gameplay)			X (Game design)	X (Game world)

The concepts that were similar and included in the frameworks, even if they were not clearly represented by their models, were merged and included in our framework (e.g. the concepts within the Representation, Game attributes and Storytelling that referred to the game's scenario are shown with the "Scenario" concept in the CMX framework). The compiler and the programming editor are suggested specifically for the computer programming domain; however, the framework includes a variety of concepts that allow its exploitation by other courses as well. This framework was employed during the design and consequently development of the MMORPG CMX game and was a valuable support, especially during the requirements and architecture specification processes.

5.2. Scenario

The main environment of CMX replicates a factory which pollutes the ecosystem with toxic waste, and the last remaining piece of earth is also in danger of contamination. In this alternate reality, a team of individuals called crackers are activists who attempt to invade the factory in order to shut down its main server and thus stop it polluting the environment further. However, the factory is equipped with employees named hackers, who are paid to protect the server and the plant's on-going operation. A virus has infected the main server making it vulnerable to attacks. The crackers seize this opportunity to discover the hidden passwords that will enable them to reach the server, enter the passwords, and shut it down. On the other hand, the hackers are also trying to find the passwords so as to destroy the virus.

Students are initially divided into one of the two teams, crackers and hackers. Each team is required to participate in the game's two phases, where they are taught programming language C in order to win. It should be noted that the game can be easily customized to support other programming languages such as Java etc. The first phase consists of a training process, where all players of both teams are trained individually by special characters existing inside the world, who are called Senseis. This training process is essential, as students can learn the theory regarding programming concepts, as well as practice with the help of the Senseis the material they have been taught. In this way, they can accomplish the various missions and proceed to the next levels, where each level provides them with one of the required passwords that brings them closer to the factory's main server. More specifically, there are three different levels that the students need to pass, each represented by a different type of Sensei (Senseis, Iron Senseis and Gold Senseis). Every Sensei provides students with a password that if they execute the requested tasks successfully, unlocks a specific Sensei of the next level. The students initially try and locate the first level of Senseis within the graphical environment. Then, students perform an interactive dialogue with each Sensei, where they are required to respond to multiple-choice and right/wrong questions in order to receive the secret passwords.

As soon as the first set of passwords has been discovered, students proceed to finding the Iron Senseis. Once they have entered the passwords, which enable the Iron

Senseis to identify that each player has successfully passed the first level of training, which comprises the authentication process, the students are required to study a series of tiles of codes that together form an executable C program. They must then construct that code by dragging & dropping and placing the tiles in the correct order, as show in Figure 5-6. The system examines the code submitted and provides corresponding feedback to each student. If the code is wrong, the system presents an error message and allows the student to try again. If the code is correct, the student is given a new set of passwords which he/she has to provide to the Gold Senseis for authentication.

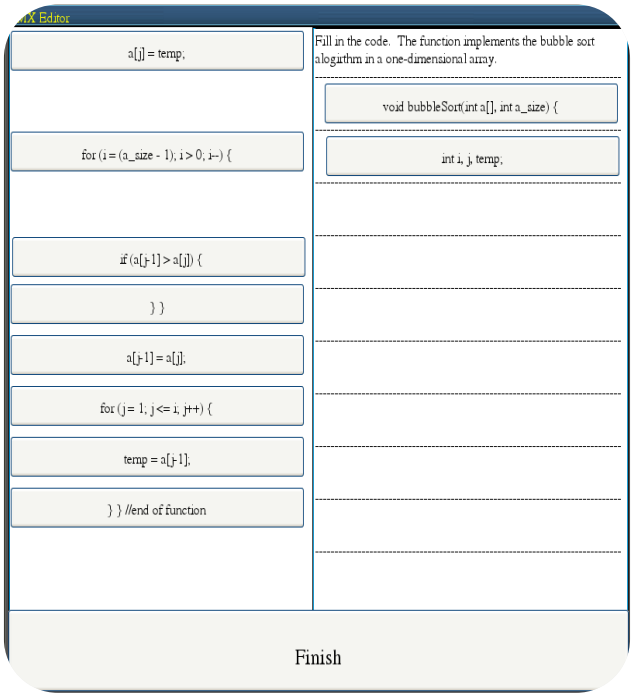

Figure 5-6: Interaction with the game with drag & drop mock up

During the last training phase, students are required to write complete programs from scratch using commands of the C language on a programming editor provided within their interactions with the Gold Senseis, as shown in Figure 5-7. The code is then compiled by an embedded compiler and the result is either a ‘correct’ or ‘error’ message. If a mistake has been made, an explanatory message appears, which enables the student to fix it. Correct code give the students the final password that provides access to the main server.

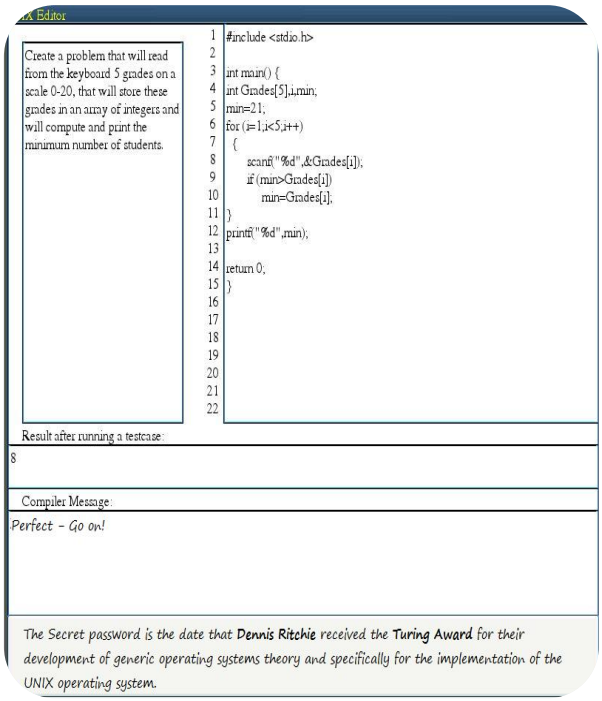

Figure 5-7: Code editor mock up

During training, students can communicate with each other by using the message chat tool, as shown in Figure 5-8, that is incorporated in the game and help each other when they have difficulty comprehending concepts or executing a task.

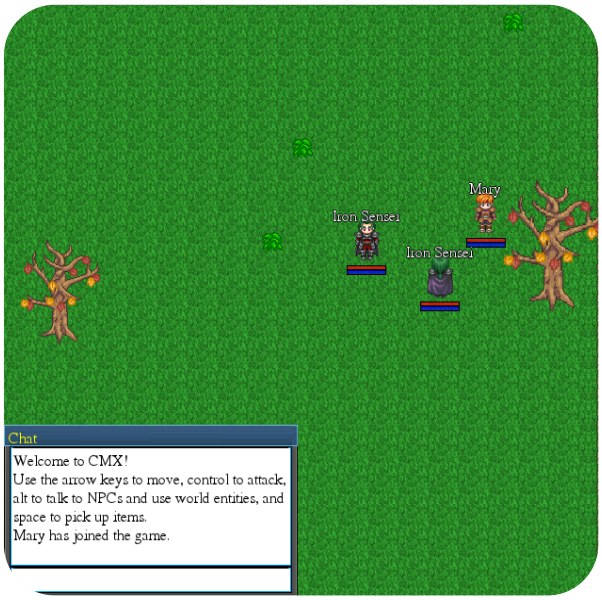

Figure 5-8: Chat tool that assists communication

The first student that manages to find the final password and unlocks the server is the winner. Furthermore, a mini map tool is embedded that displays a micrographic outlook of the environment so that players can have an overview of all players' positions in the game. This feature enhances students' collaboration since they can form alliances based on their respective positions in the game world and endeavour to progress by helping each other.

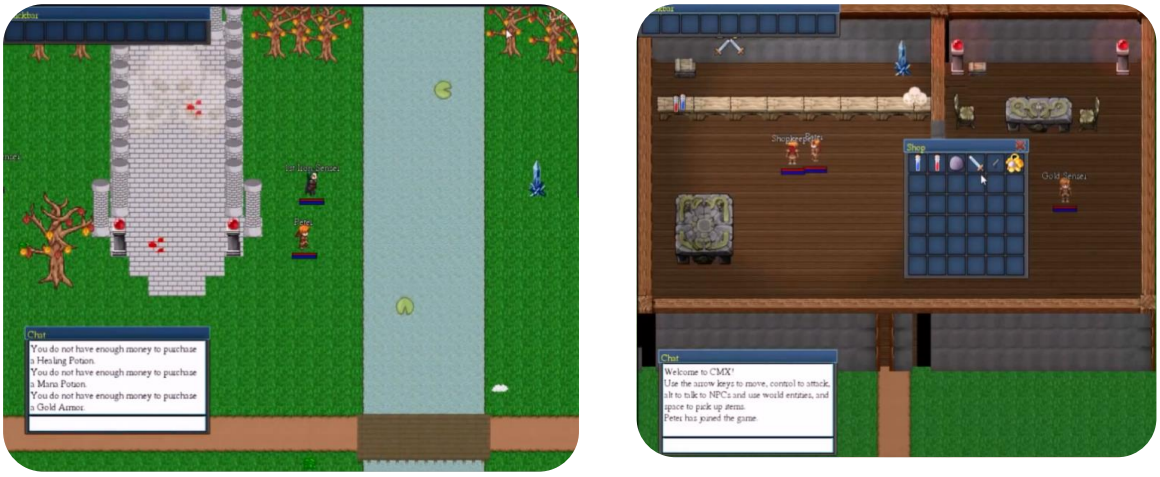

Figure 5-9: CMX environment

Figure 5-9 shows the environment in which the students try to locate and interact with all the Senseis or the other players from the same team during the training phase of the game in order to be properly trained and in which they endeavour to reach the main server to submit the final password.

When the initial training phase is complete, each team moves on to the second phase which involves the efforts made to enter the factory facilities and locate the server in order to enter the final password. Students can protect themselves and beat their opponents by using and winning more weapons, spells and experience points as they fight their way towards the main server.

This training process we have included is an innovative component that we deem necessary to exist in educational games. This way, players are motivated to not only read about the programming elements, but understand them correctly in order to be

competitive enough in the game, since the more weapons they get, i.e. the more questions they answer correctly, the more chances they have at winning.

It should be noted that some tasks during training require the team's collaboration for their successful execution. For example, the group cannot proceed to the Iron Senseis' leader if all the players have not gained weapons from their Basic Sensei. Thus, team work is already promoted even before they start playing the game.

CMX's allowed interactions and supported tasks follow the constructivist theory presented in section 2.2.2, allowing students to create knowledge by actively trying to solve given problems. This way, students can generate meaningful information (e.g. the code to be written, the answer to a multiple choice question, the correct mapping of code etc.) while they interact with the game's elements (e.g. Senseis, activities, avatars) and execute authentic tasks. More specifically, the features the proposed game addresses that provide added value compared to the related work include:

- It is a **holistic game**. Although the game is divided into levels and sublevels, it is designed so that it can be used for all programming teaching sections.
- **Adaptability** of the game interface by teachers. Additionally, teachers can design the game according to their lesson to the class, set the rules of the game, set the goals and finally, decide for the reward of the winners.
- Strong emphasis on **role playing**, which causes students to immerse themselves completely in the game, and thus completing their tasks in a less obligatory manner.
- The game has all the aspects of a multiplayer game so that players should **collaborate** in order to achieve the goals and win the game. There is an embedded chat tool, so that players can communicate with each other and be better organized into their teams.
- The players have a variety of ways they can **interact with code programming** within the game, according to the level of their programming knowledge. They can either answer simple multiple choice questions, or drag and drop tiles to complete the program correctly, or write a program on the editor.
- The game includes a lot of **tutorials with hints and tips** that can help the students achieve the goals and simultaneously teach them the programming aspects they need to know.

- The **scenarios** provided by the game are **infinite** and cover a **variety of units of learning** instead of a single one.
- The game's logic "The best gamer is the best programmer" underpins their **motivation to play more actively** and, by association, to learn how to program more effectively.
- The fact that **this is a MMORPG** removes students' usual initial reluctance to try new and foreign to them educational software. The logic of this game is similar to games the majority of students are already familiar with (e.g. War of Warcraft etc.), so it is easier for them to **find this new technology enjoyable**. Additionally, this reduces the effort it takes for teachers to train students in a novel, complex system, and the time it takes for students to learn all the specificities of such a technology (e.g. menus, console, action buttons, interface layers etc.).

The last point made is very important, as more than 10 million people dedicate multiple hours daily within virtual worlds, collaborating with virtual characters and executing virtual actions (Shanahan, 2009). As an example that was also mentioned earlier, based on stats released by Blizzard, the World of Warcraft game documents more than 11 million players while 40 thousand new players get enrolled every week, dedicating one to twelve hours a day to the game.

While World of Warcraft is an entertainment game all together, there are examples of educational MMORPGs, such as the Second Life world. Second Life is a very well-known game that provides an attractive virtual world with multiple educational scenarios and interesting features that attract a large audience, such as many entities, streaming video and sound, freedom to edit the hero according to one's personal preferences etc. Thus, CMX intends to create a world most students are already accustomed to and use it in educational settings for successful computer programming teaching and learning.

5.3. The CMX roles and features

CMX includes and can be further expanded with a large variety of computer programming concepts without having to focus on only one learning unit. This is one of

the supplementary features included during the design process, aiming to support multiple capabilities for teachers and students. Based on the above, CMX was designed and developed with the intent to draw attention to and engage all roles that are involved in an educational game. To this end, it combines features that support all these roles, as shown in the Figure 5-10.

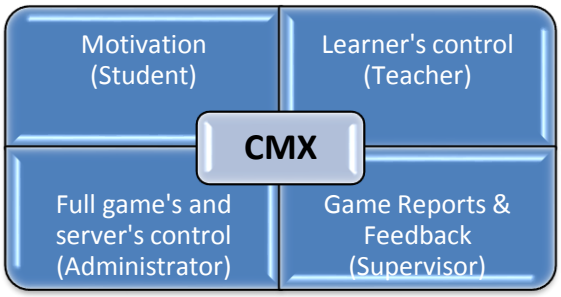

Figure 5-10: User Roles in CMX

More specifically, CMX was developed mainly to support students in learning computer programming, by fostering their motivation and allowing them to be more active while participating in the learning process. This is facilitated with the inclusion of a number of features such as activities/tasks during the training phase and entertaining elements during the second stage (e.g. points, spells, rewards, special weapons, points etc.). Another important feature is the provision of feedback in regards to their progress, the possible mistakes they make during training, their current location or status etc.

Additionally, the game also supports teachers in maintaining complete control over what learning content is taught and what kind of activities the students will engage in during training. The game also allows for the role of administrator (usually but not necessarily the teacher) to configure the entire environment by changing the characters and the graphics, to monitor students' behaviour and their progress, check the server's operation and adapt the game's difficulty levels when necessary. A final role is that of the supervisor, who can be either a teacher or the administrator, and is able to construct reports that can provide insights on the game's status, the overall progress of the participants and other analytics.

Another important concept that, although non-human, is very closely connected to the game is computer programming knowledge and skills. This concept can significantly change the entire game environment since it affects the number and types of

tasks to be included, the units of learning to be inserted, the resources to be made available by the system's server etc. To this end, it is important that the design and development of any educational game takes into consideration the particularities of the knowledge to be taught and the skills to be developed by using the game.

Figure 5-11 shows a list of these features that comprise an educational game for computer programming that supports all phases of education: design, implementation and evaluation.

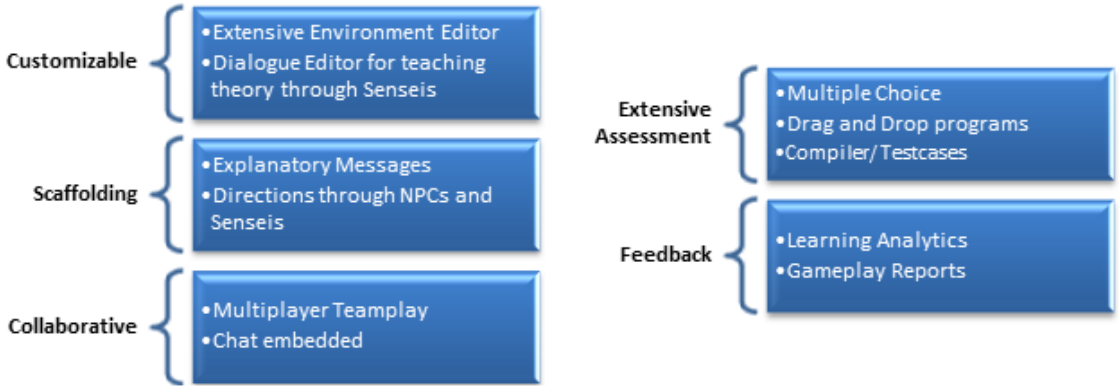

Figure 5-11: CMX features

Specifically, CMX combines entertainment with education: while playing, students are able to perform activities, get feedback on their progress and mistakes, collaborate with their classmates and assess their knowledge comprehension. Also, teachers/administrators are able to customize the entire game, monitor the learning experience, provide scaffolding to students, and include content and activities that will allow the necessary knowledge to be comprehended and skills to be developed. Finally, teachers and supervisors are able to assess all students' activities and receive analytics feedback and evaluation reports from within the game.

5.4. CMX Editor

Another feature is the ability for the teachers to manage the game's environment, monitor students' behaviours during the gameplay and assess their progress. Furthermore, teachers are allowed to determine each of the Sensei's behaviour regarding

how they will guide and engage the students. This way they can simulate their own guiding and scaffolding role into each Sensei character. For this purpose, they can include a variety of tasks, as well as advice, explanatory messages and hints that can assist students when they make a mistake or miscomprehend a concept.

5.4.1. Environment Editor

Teachers can also configure the game's graphical environment, as shown in Figure 5-12. This functionality presents multiple benefits, as it allows the entire game to have a different look for the various classes and students depending on their specific particularities. This fosters and stimulates students' interest and motivation, since the game is re-designed according to their preferences. Figure 5-12 presents a representative screenshot of the CMX editor environment.

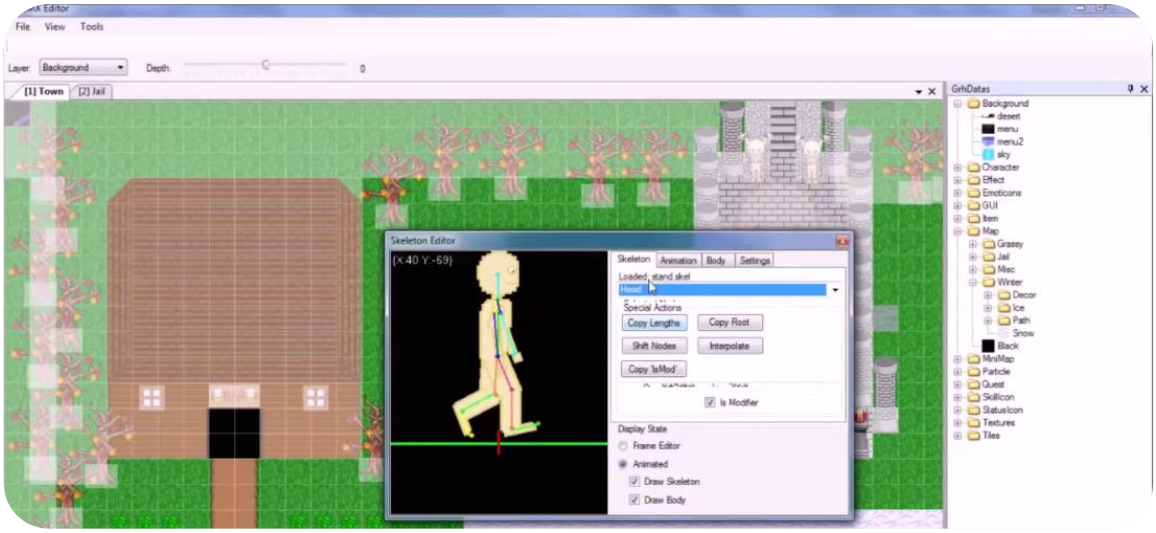

Figure 5-12: Configuration of the CMX environment

Teachers can customize all the graphic elements including the characters and the tiles inside the game. For example, in the next figure we are setting a new graphical environment with two small houses and one tree by dragging and dropping tiles. This is carried by using the special Particle Effects Editor to set dynamic effects, that are graphics that will not have a static nature like images, but will exist in the world as dynamic effects.

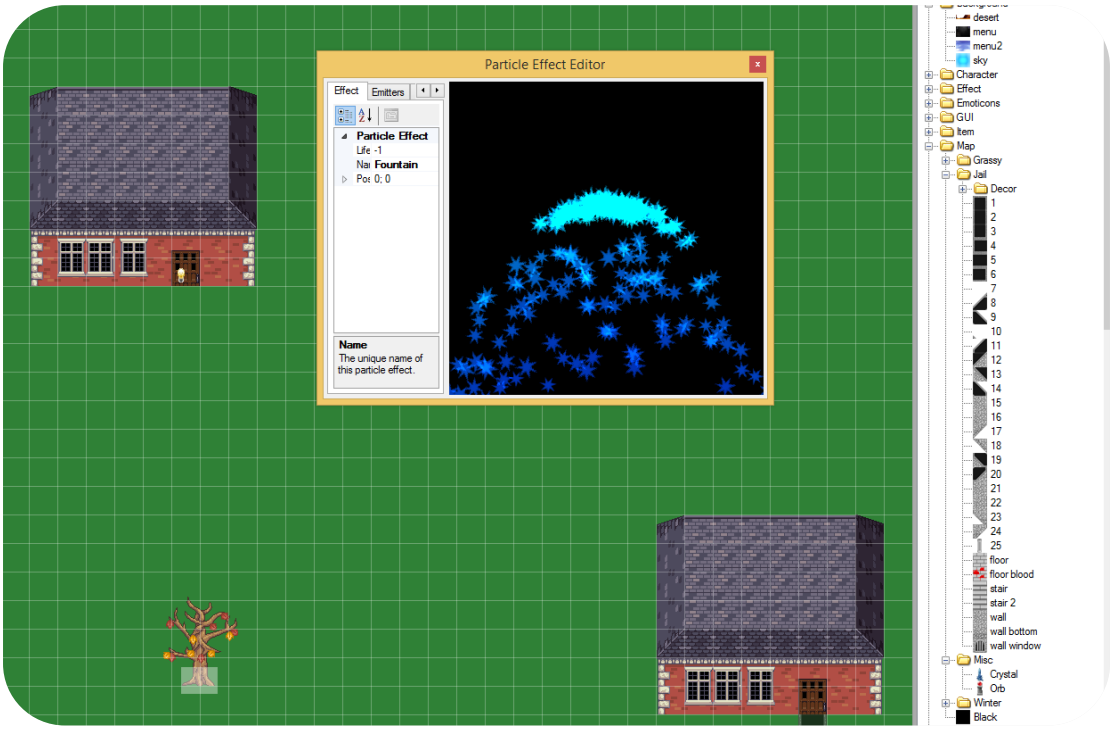

Figure 5-13 Configuration of the CMX environment by placing some special effects

The right sidebar shows all the tiles that are available for use by the teacher in order to draw the environment desired and also add any images. This way, the teacher can create many different versions of the game, simply by changing the graphic environment so that it simulates the world that is set in the teacher's learning objectives.

5.4.1. Dialogue Editor

Teachers also configure the CMX environment by formulating the educational content to be taught in each course section. The teacher uses the game's dialogue editor in order to design the theory units, the practical activities to be added for each theory unit, as well as the sequence flow that will guide students to maximum knowledge acquisition. Figure 5-14 is a screenshot of the dialogue editor that demonstrates each instance of the virtual dialogue carried out between the student and the Senseis as a graphical state with multiple possible answers that determine the progression of the dialogue in different states, as shown by the arrows. This way, each Sensei leads the student from the theory statements to the evaluation states (i.e. multiple choice questions), and based on the student's answers, the teacher provides the corresponding

feedback with congratulatory messages or with hints that should be taken into consideration during the next attempt. Thus, each dialogue with the Senseis is represented by a fully customizable state diagram.

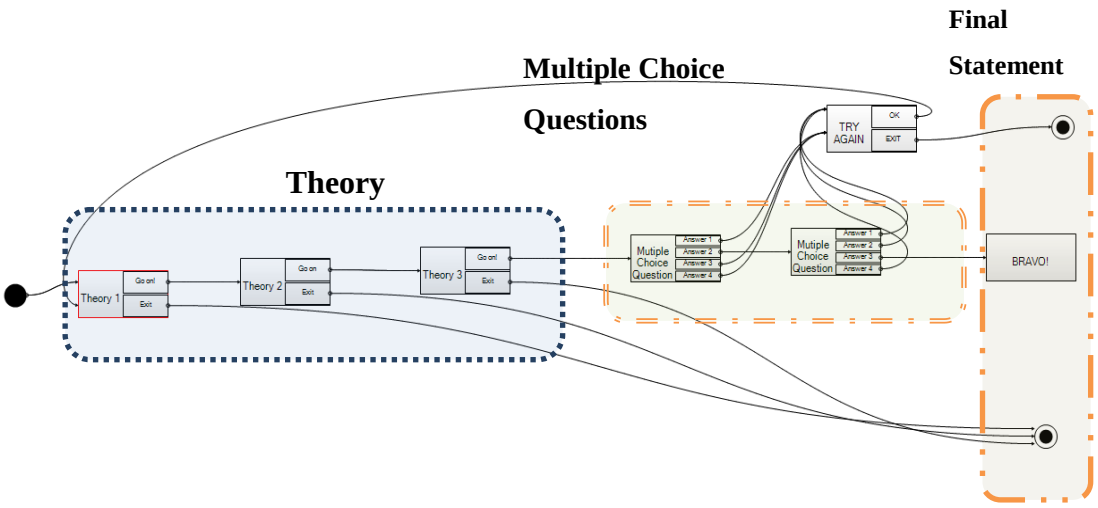

Figure 5-14: CMX dialogue editor

5.4.2. Database Editor

CMX includes a graphical editor where the teacher that wants to generate his own version of the game can configure all dynamic elements of the environment. This Database editor includes templates for items and characters within the game. Using these templates, the teachers can easily add an image in the game and transform it into a dynamic entity, by defining exactly how it would interact with the other entities in CMX.

For example, Figure 5-15 shows a healing potion, where an image is loaded to represent the potion. Also, it is defined that when a player interacts with this item, then 25 health points will be added to the player's profile, and that it can only be used within the game. It should be noted that this way, the teacher can create his own templates and can import the created items multiple times within the game.

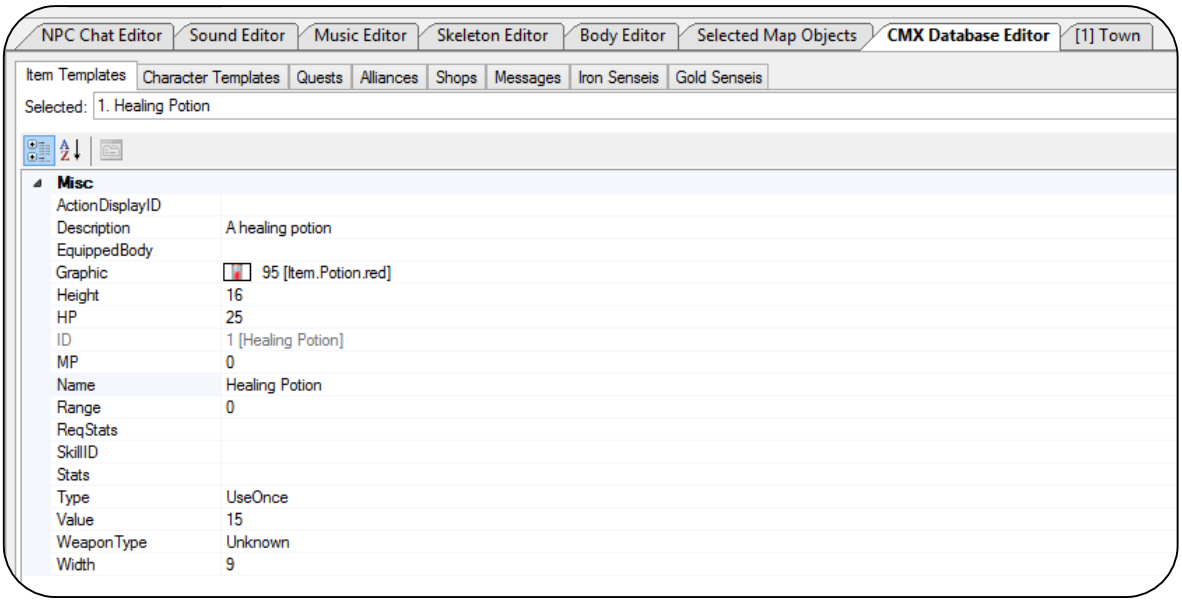

Figure 5-15: CMX database editor – Item Templates

Thus, in keeping with the CMX design framework, the teacher can define the environment that will interact with the user by changing the scenario and the virtual players. Furthermore, the editor includes the ability to define character templates. This way, the editor classifies if the character will have an armor or any sort of weapon, what skills will it possess, and if it can reward a player if it gets eliminated in the game. Also, we can define specific percentages of probability; for example in the following figure, we define that every time the character Guard shows up in the game, there will be 30% probability he will show up with a blue sword, a 20% probability he will show up with a stone as a weapon, 10% probability he will not hold a weapon but a healing potion and a 10% probability he will not hold a weapon but a gold or iron armor. We can moreover determine whether the character can interact and talk with the players and which specific dialogue the character will start when the player will attempt to talk with him.

Also, we can define in which alliance the character will belong to, with what speed he will move, if he will appear again once eliminated and after how much time (respawn) and how powerful of a player will he be.

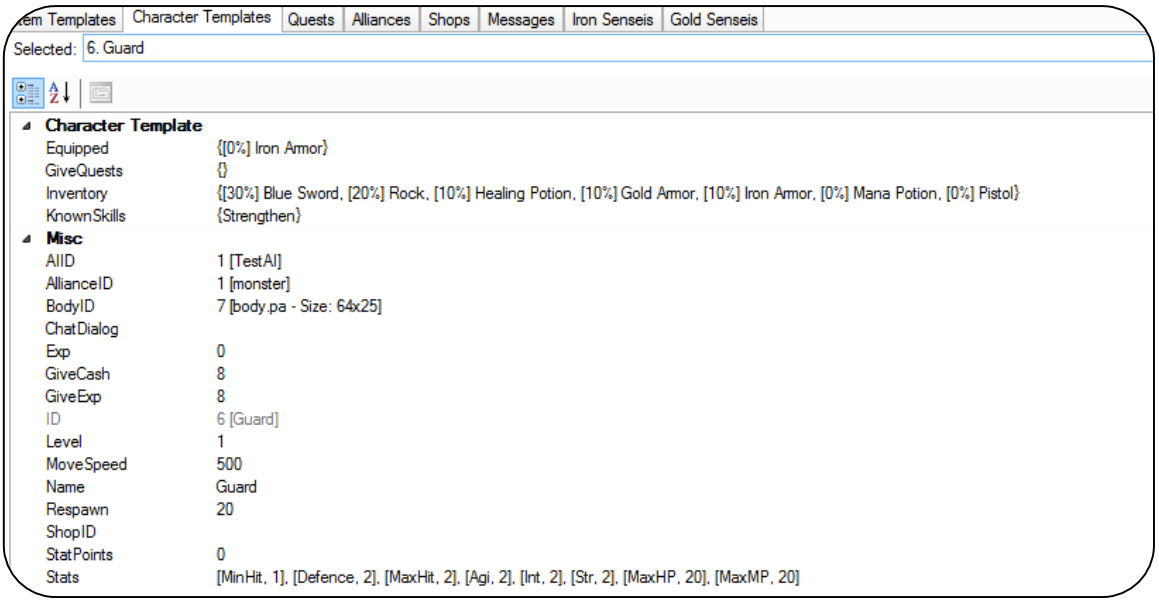

Figure 5-16: CMX character templates' customization

The Database editor includes the option to include smaller, separate missions as quests inside the game, apart from the general purpose of the scenario, as shown in Figure 5-17. Such quests are defined in the editor, where we add the description of the quest, the specific requirements that will determine whether a quest is complete and what rewards will players receive once they complete it.

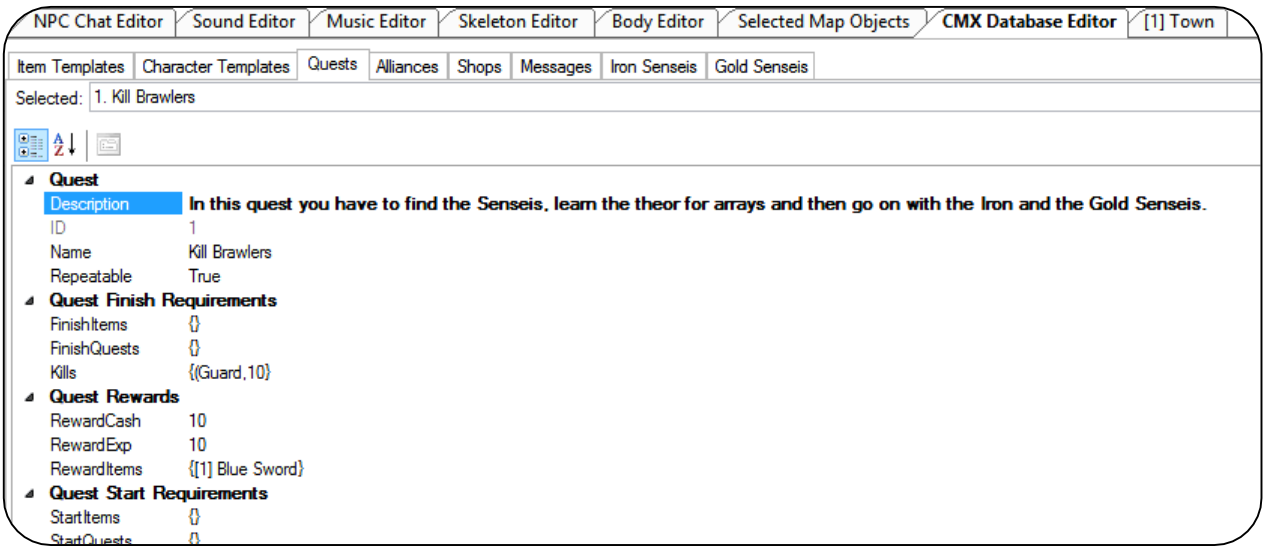

Figure 5-17: Setting and modifying quests in CMX

Moreover, we can set up and configure alliances within the game, between players and virtual players, as show in Figure 5-18. This way, we can create different virtual players that will either be allies or enemies with each other or with the actual players.

Figure 5-18: Setting and modifying alliances in CMX

The editor also allows us to define entities that are called shops as shown in Figure 5-19, a process that is very usual in such games, where users can cash out their points by buying more weapons and powers that will make them stronger in the game.

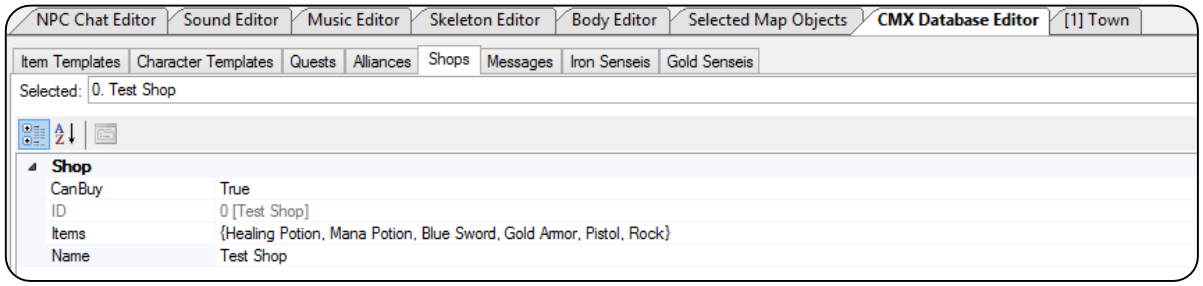

Figure 5-19: Setting and modifying shops in CMX

There is also the possibility to define specific phrases that will be used by the players or the system during the execution of basic processes, as show in Figure 5-20. These phrases can be configured by the administrator or by anyone that wants to translate them in a different language. In the default version of the game there are phrases in Greek and English. It should be noted that the teacher configures all the elements that will facilitate the goal of achieving the learning outcomes that were set with the help of the CMX Design Framework. Language is of course one of those important elements, so that each teacher can make the game more user-friendly.

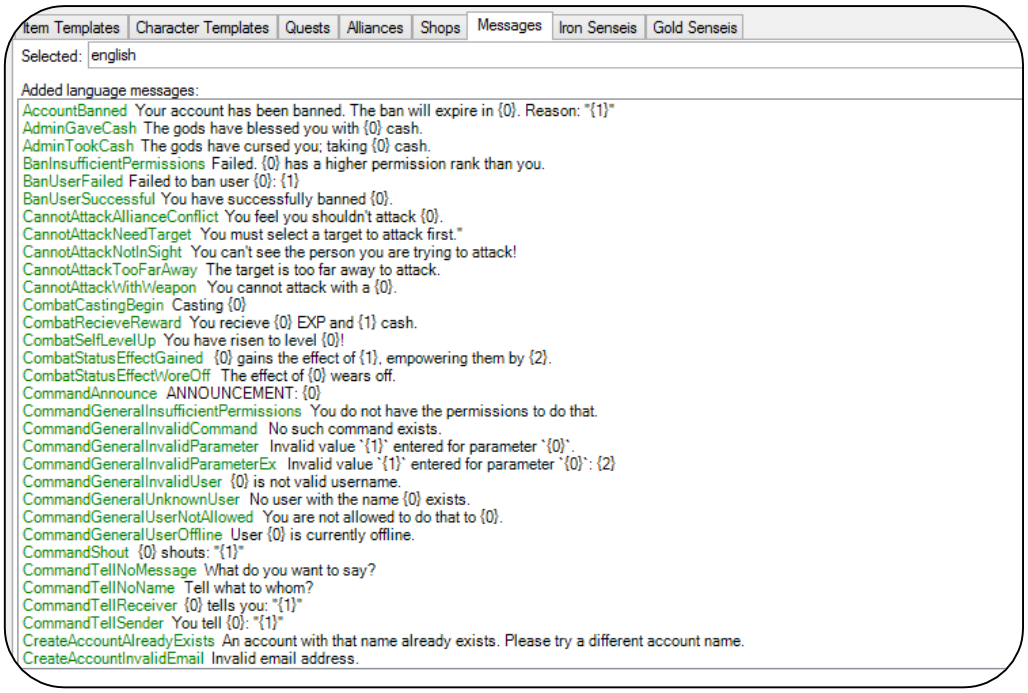

Figure 5-20: Translating system phrases in CMX

Another important configuration feature is the ability to define the activities, as this is a very important element of the CMX design framework, assigned by the Iron and Gold Senseis to the students. More specifically, as shown in Figure 5-21, we can define what problem the Iron Senseis will provide to the players and set the different commands that have to be put in the correct order, as well as show which the correct order is. The players see a random list of the defined commands and have to put them in the order we have specified inside the editor.

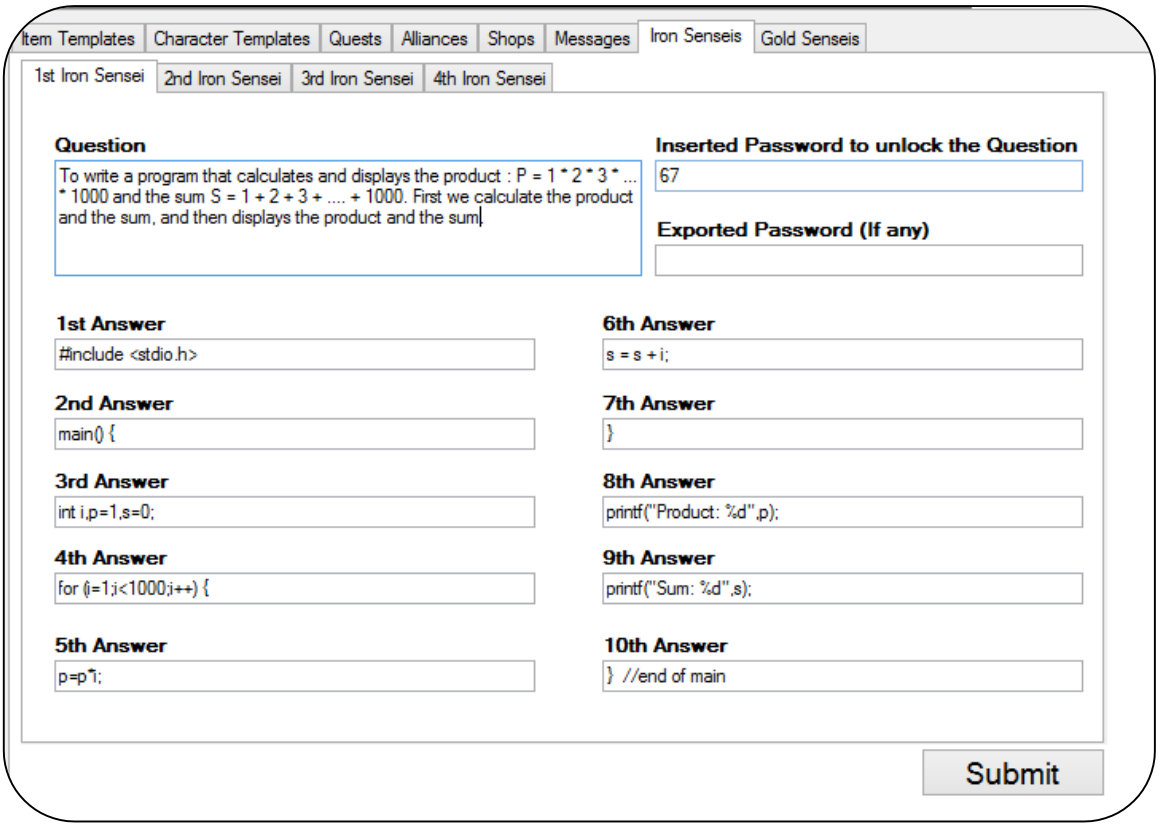

Figure 5-21: Setting and modifying Iron's Senseis activities

Finally, regarding the Gold Senseis, we can define a specific problem and the corresponding testcases that the system will run depending on the inputs provided by the students. In the example shown in Figure 5-22, we ask from the student to develop a program that will read 5 numbers and will compute and print the sum of the even numbers. As first input the numbers 12,12,17,18 and 21 are provided and the program expects as output the number 30.

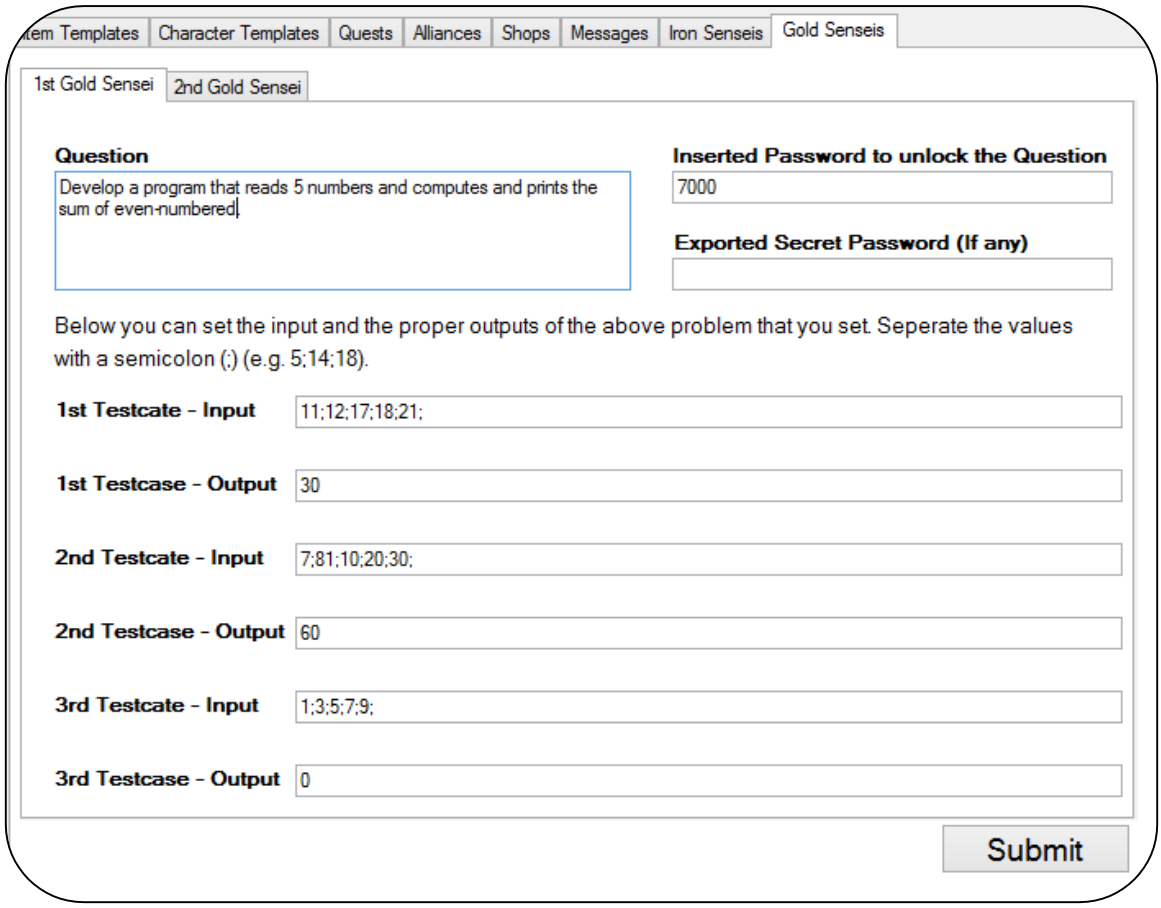

Figure 5-22: Setting and modifying Golds's Senseis activities

Following, the program will run again with numbers 7, 81, 10, 20 and 30 as inputs and it will expect the number 60 as output. Similarly, the 3rd testcase will run the program with inputs 1, 3, 5, 7, 9 and will expect the number of 0 as an output.

So, even if the program written by the student is syntactically correct, it will not be considered correct if it does not produce the expected results.

5.5. CMX Server Administration

The game's administrator has the ability to control the game's server, to manage the players that participate in the game and to have a full supervision of the resources consumed and any problems that may occur.

The following figure shows an instance of the CMX Server, with information like the server's IP, the total CPU usage, the RAM usage by the server and the amount of the

network's bandwidth being consumed. In the specific example shown, the CPU usage is 18%, the consumed RAM is 26MB, while there are remaining 1560MB available for use, and there are 0 users online, therefore no MB are being used for the bandwidth.

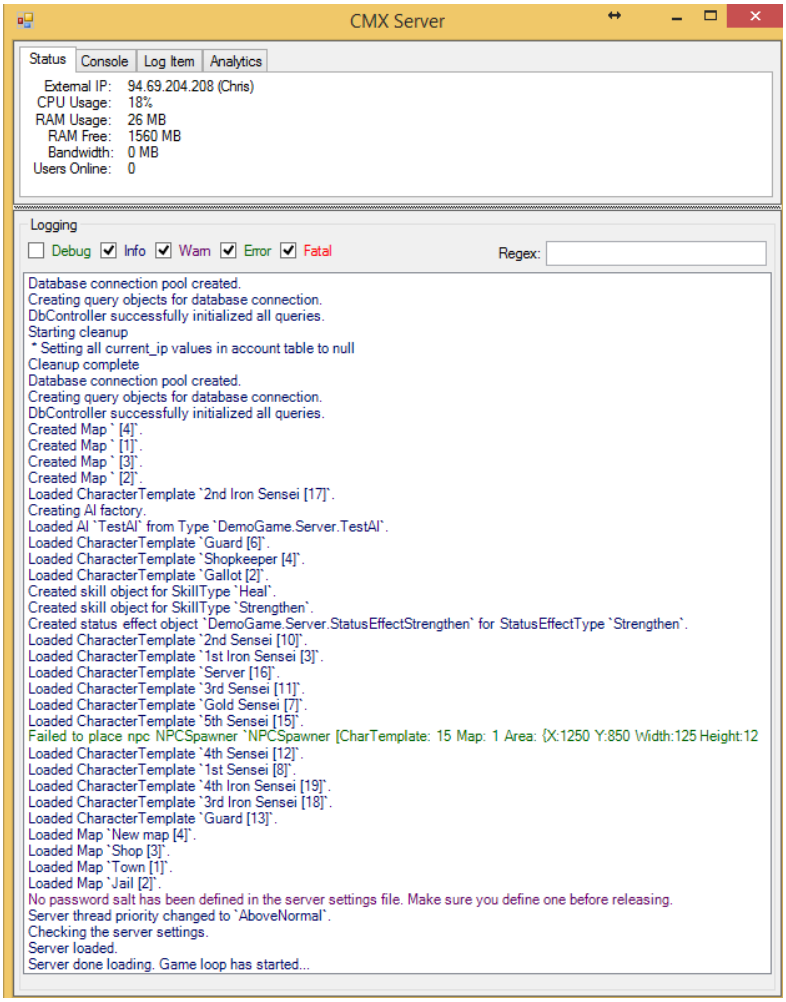

Figure 5-23: Server's administration

The above example was instantiated during the beginning of the game, and before any users had logged on; therefore, we notice that the dynamic elements set within the game are being placed. For example, we can note that the maps are being loaded and the characters are being positioned within the virtual world. We can also see that the positioning of a specific test character named NPCSpawner on the map fails, as its set coordinates are outside the borders of the map that has been designed.

With the above information, it is understandable that the administrator has an overall control and supervision of the game and its normal operation, where he can see

all the errors that can happen and why they happened and he can fix them and re-design the game accordingly.

5.6. Integrating learning analytics in an educational MMORPG for computer programming

Although educational games are an interesting tool that could be used, a significant challenge to be addressed is the identification of all the ways progress and results can be measured for examination. This way, we can determine how effectively the initial educational goals are fulfilled and which activities depict students' performances. This can be carried out with the incorporation of Learning Analytics (LA) in Game Based Learning (Siemens, 2010), a process we refer to as Game-Based Learning Analytics (GBLA) (Malliarakis et. al, 2014d).

LA is considered to exploit features from various fields such as business intelligence, educational data mining and business logic. Scholars have started gaining interest in investigating how the proposed analytics methodologies can enhance learning by gathering and analysing data (Lias & Elias, 2011). The process followed usually includes five major steps, namely capture, report, predict, act, and refine (Campbell, 2007). Each step should be designed so as to lead to meaningful reporting and visualizations that will help teachers to optimize their courses and students to absorb concepts more easily.

5.6.1. Learning analytics and assessment methods in educational Games

Educators of all domains, and especially computer science, are gradually taking into consideration all recent technological developments that could facilitate learning processes, reduce teachers' workload and increase students' capabilities. In this endeavour, they have started to incorporate educational games in order to attract students' attention and transfer knowledge in an entraining manner (Johnson et al.,

2012). Although most teachers use educational games to motivate and challenge their students (Annetta et al., 2009) they do not seem particularly willing to assess students' knowledge comprehension using the games' functionalities or to study students' behaviours within the games. Thus, the majority of academics continue to resort to traditional assessment methods even though games can significantly help evaluate comprehension and skills such as critical thinking, reasoning, problem solving, observation etc. (Del Blanco et al. 2012; Serrano et al., 2012).

The main reasons why academics are hesitant about extracting educational conclusions through feedback from the games is because they either use commercial educational games or because they incorporate or develop educational games with easy-to-use game editors (Torrente et al., 2010). In both cases, educational games do not allow assessment functionalities, and even those that do are difficult to operate because they demand advanced techniques of data mining.

To this end, there is a need to create proper frameworks that will ensure the successful exploitation of all educational data that can be retrieved from educational games. An interesting conceptual framework is proposed by (Greller & Drachsler, 2012), where all concepts related to the application of LA in educational settings are divided into six axes, as shown in Figure 5-24.

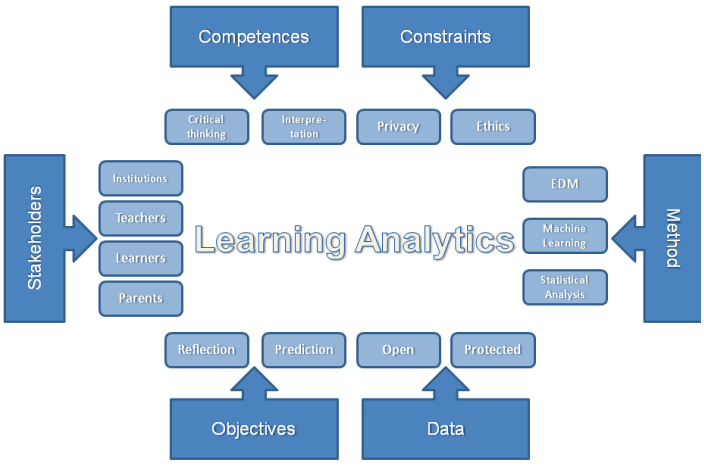

Figure 5-24: The Learning Analytics framework (Greller & Drachsler, 2012)

The axes of the framework can be defined as follows:

- ✓ *Stakeholders*: the contributors and beneficiaries of LA.

- ✓ *Objectives*: set goals that one wants to achieve.
- ✓ *Data*: the educational datasets and the environment in which they occur and are shared.
- ✓ *Method*: technologies, algorithms, and theories that carry the analysis.
- ✓ *Constraints*: restrictions or potential limitations for anticipated benefits.
- ✓ *Competences*: user requirements to exploit the benefits.

The authors suggest that the concepts included in each axis can help extract useful educational conclusions.

Another interesting study mentions that learning analytics is the main mechanism that could allow proper students' assessments (Serrano et al., 2012). More specifically, the authors suggest that high level assessment reports can be extracted with the use of learning analytics tools that could provide useful conclusions regarding students' performance, progress and shortcomings. Towards this goal, a rule-based system is required, which will facilitate interaction with game information such as its structure, specific educational goals set for each course, phases designed, as well as information about characters and objects developed during the game. The teachers can define specific rules on what needs to be analysed and the system uses these rules to retrieve information from the interaction logs. All data is then refined in order to infer conclusions which are reported on high-level terms back to the teachers' interface. The concepts that are used for assessment reports extraction are shown in Figure 5-25.

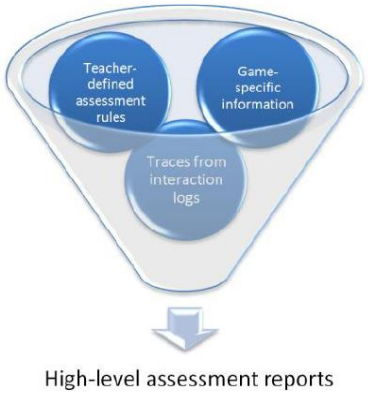

Figure 5-25: High-level assessment reports with Learning Analytics (Serrano et al., 2012)

5.6.2. CMX Learning Analytics framework

The learning analytics theory was taken into consideration during the design phase of CMX. More specifically, the research aimed that the game would retrieve appropriate feedback from teachers and students and would be able to assess whether the educational goals were met. This way, teachers could reconfigure the game if results showed that the progress of students was not successful. Thus, particular focus was given to the game's environment and all its features that could be further analysed and prove useful for the teachers or the administrators.

During this study, these features were grouped in 6 main categories based on their nature. These categories or axes represent the tangible and measurable features of an educational game that should be taken into consideration during the calculation of a mathematical model that measures its efficiency.

Figure 5-26 presents the proposed framework with the six axes (Malliarakis et al., 2014d), where the two ends of each axis include inter-related concepts.

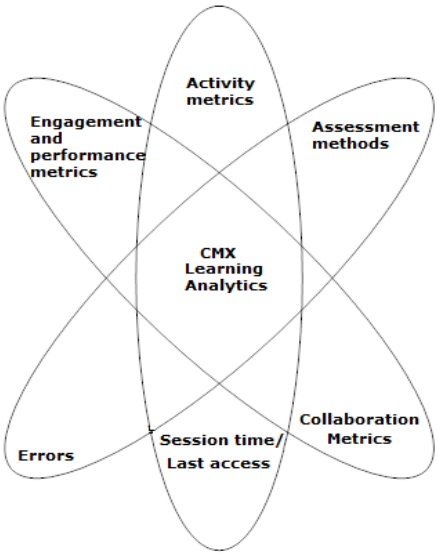

Figure 5-26: Proposed LA' framework

Each axis of the framework is explained below.

1) Activity Metrics

Activity metrics are the ways in which each player interacts with other users and with the game's bots while playing. During all interactions, the database records behaviour intensity, i.e. how active a player is during specific actions. An analytical presentation of all activity metrics can be found in (Malliarakis et al., 2014d), where these metrics are taken into consideration for dynamically engaging the appropriate amount of resources in the server hosting the game.

2) Session time/ last access

An important factor in educational games is session time, which is the time a player starts a game until the time it takes for the game to finish. It could also be considered as the time it takes to complete a task or reach a milestone. These times can easily determine if students utilize the game in an educational manner or if they play randomly because they estimate they will not be able to achieve their goals.

The second time period that should be taken into consideration and recorded as important data is the last access. If a user takes a long time to access the game, there is a chance that the computer programming concepts taught during the previous session have been forgotten. To this end, it would be useful to integrate a reminder mechanism in the game that, if a significant amount of time has passed since a player's last access, it will present a concise overview of the latest materials taught.

3) Assessment methods

CMX provides a number of different interim assessment activities in order to ensure that all players comprehend correctly the computer programming concepts they are taught in each phase of the game. Since computer games can be engaging but also distracting to young students, and also computer programming concepts are constantly required to be used in various tasks, it is essential for the educational game to scaffold learning in every level. In every training level in CMX, players are rewarded for their correct answers with weapons, spells and points, which they can use later on the game.

4) Errors

The concept "errors" represents the number of errors made by a student. We also store information such as what each error was and in which part of the game it was made. This way, we can draw conclusions as to how correctly set up a specific task (e.g. if all students made the same mistakes then maybe the task was designed incorrectly).

Moreover, the recording and analysis of errors and their frequency can assess the educational value of the knowledge taught as well as its impact in proper knowledge comprehension. For example, students can easily complete a task assigned by the corresponding Sensei in CMX if they first properly understand the given guidelines. However, as they move forward into the game's levels and learn new and more advanced knowledge, it remains to be seen whether they will be able to re-use previously learned concepts in more complex tasks, e.g. writing lines of code.

5) Collaboration metrics

Some members may require the team's collaboration for the successful execution of tasks. For example, the players cannot proceed to the Iron Senseis if they don't pass the tasks provided by the Basic Senseis. This collaboration contributes to students' learning because they can retrieve and provide help to members of their team, such as theory hints, links to helpful materials or examples etc. Any such interactions between the players are recorded as collaboration metrics (e.g. chat content, number of chat sessions, passing of game elements etc.) because they are considered an important factor that indicates the game's success and a team's performance.

6) Engagement and performance metrics

The performance of an MMORPG is affected by a number of different factors. These can either include players' behaviours within the environment that has not been previously predicted, or a system fault (e.g. overloading), that could lead to a game failure and downtime. These types of behaviours should also be recorded and made aware of to the teachers when they evaluate the game's performance. Also, the game's administrators are required to view such metrics so that they can fix the errors in the next update.

5.6.3. Implementation

CMX is intended to be played in educational settings, thus by a limited amount of students so that performance results can be more easily and consistently transformed to qualitative and quantitative conclusions on learning progress. However, when an educational game is also a MMORPG, then the data that needs to be gathered and stored in the system's database is more vast and complex. For example, many more students can interact in the game at the same time, thus amplifying the number of interactions that

need to be recorded. Moreover, the data generated (e.g. compiler results, lines of code in the code editor, answers in multiple choice questions, interactions in the arena, chat etc.) need to be constantly analysed and provided as feedback to the teachers. This way, teachers can alter the design of the game's next session depending on the analytics they receive.

Apart from the data that the system can gather based on the interactions and content created within the game, another interesting way to draw conclusions is based on the students' behaviour. For example, an auto-rating mechanism will allow the system to grade players regarding every characteristic of the game that we want to assess, which includes all six axes of the proposed learning analytics framework.

To implement the framework, we created a mathematical model that will automate the process of gathering results and drawing conclusions with the help of LA. This model includes the usage of an algorithm that measures and grades the players' behaviour, i.e. the value changes of the varying factors that we are examining for each player as they are being stored in the system's database. This activity's value is stored as a vector and transferred in the database as shown below:

$$\overrightarrow{S_{CS}(t)} = \langle P^{cs}(t) | W^{cs}(t) | A^{cs}(t) \rangle = \langle s_1^{cs}(t) | s_2^{cs}(t) | s_3^{cs}(t) \rangle \quad \textbf{(5.1)}$$

CS is the indicator of the score that suggests the client-server message transmission, while t is the time parameter, since the values that these variables take change through time. The G^{CS} (Grade) is a vector that corresponds to each player and indicates the educational score the player gets in the game's progression. This score is dependent on a) the player's success percentage during the assessments (variable S^{CS}), b) how many errors the player made (E^{CS}), c) how active the player is (A^{CS}), d) how frequently the player visits the game (F^{CS}), and finally e) how collaborative the player is (C^{CS}). With the recording of all these variables $S^{CS}(t) \mid E^{CS}(t) \mid A^{CS}(t) \mid F^{CS}(t) \mid C^{CS}(t)$, we indicate that in every time instance the variables' values change and each new value is documented in the database over time. It is worth mentioning that the engagement and performance metric is not included in the model, since it is taken into consideration by the administrators regarding the system's performance, and does not provide significant indications on the players' behaviours. Teachers can determine the ideal values for each variable so that they can later on compare them to the actual recorded values.

Alternatively, teachers can determine a specific player's behaviour as ideal, and compare those values to the other implementations. This is shown in (5.2).

$$\overrightarrow{|G_{CS}(t)|} = \sum_{i=0}^{n} \left(\frac{S_i^{CS}(\mathrm{t})}{S_0^{CS}(\mathrm{t})} + \frac{E_0^{CS}(\mathrm{t})}{E_i^{CS}(\mathrm{t})} + \frac{A_i^{CS}(\mathrm{t})}{A_0^{CS}(\mathrm{t})} + \frac{F_i^{CS}(\mathrm{t})}{F_0^{CS}(\mathrm{t})} + \frac{C_i^{CS}(\mathrm{t})}{C_0^{CS}(\mathrm{t})} \right) \quad \textbf{(5.2)}$$

The values of the ideal behaviour are defined as S_0, E_0, A_0, F_0, C_0, while n is the number of transactions per player. Finally, a transaction is considered the completion of a task within a game level. Table 5-2 includes an instance where three players have interacted with the game and the system has rated the model's variables as shown below:

Table 5-2: Database instance of a player's recorded activity

Player ID	S^{CS}	E^{CS}	A^{CS}	F^{CS}	C^{CS}
1	8	5	7	25	5
2	5	8	6	18	3
3	2	9	5	12	2

The above table shows an example where Player 1 in a specific time instance has managed to successfully complete 8 activities, has made 5 errors and has played 7 times the game for 25 minutes each. Additionally, Player 1 has carried out 5 interactions with other players.

In continuance, it is assumed that the teacher determines the behaviour of Player 1 as ideal. In this case, the values of Players 2 and 3 are configured correspondingly. Moreover, all values of the indicators are adjusted between the scale [0.1, 1] in order to normalize the data and ensure easy comparison, so that can equally contribute to the final value of the total grade (G^{CS}).

Table 5-3: Database instance after normalization

Player ID	S^{CS}	E^{CS}	A^{CS}	F^{CS}	C^{CS}
1	1	1	1	1	1
2	0.63	0.63	0.86	0.72	0.6
3	0.25	0.56	0.71	0.48	0.4

It is worth mentioning that these values are gathered for each task and estimated incrementally as each rating vector is gathered and stored within the environment.

In the following figures, we can see the way that a supervisor can receive the results of the game regarding the game's performance, by setting the date and time that the game started. Following, the supervisor opens an analytical Word file which includes

a summary of all the metrics and also a more analytical review of the players' performance.

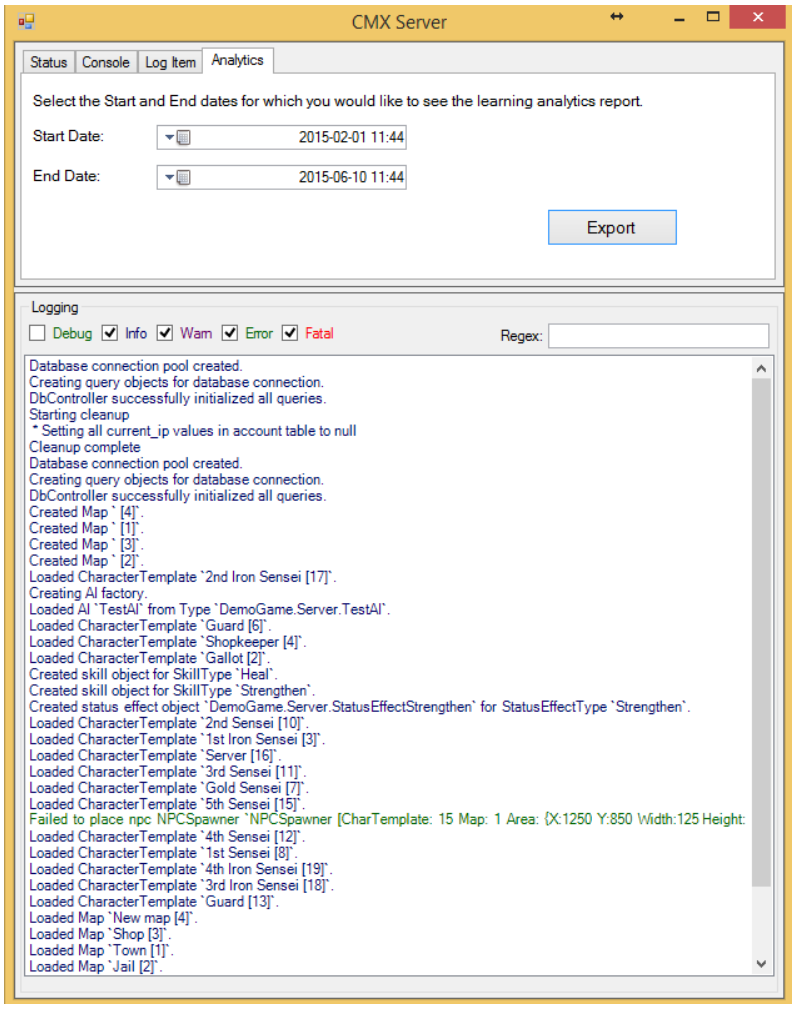

Figure 5-27: CMX monitoring by supervisor

5.7.Summary

This chapter presented the educational MMORPG called CMX. More specifically, the design framework was introduced and analyzed, which led to the design of CMX and the methodology which guided its development. This methodology derives mainly based on the literature review carried out on related work and aims to address the gaps identified and also include the promising features some of the aforementioned games support. Additional information regarding the game was provided, such as an initial prototype of its architecture, the scenario and a few mock up illustrations of its

functionalities. Next, the chapter presented the scenario of the game, the supported roles as well as the specific characteristics of CMX.

The following sections presented the learning analytics mechanisms that have been incorporated in the game and continued on to analyse the learning analytics framework designed for the assessment of CMX's utilization in computer programming courses. The conceptual framework that was proposed and designed comprises of six main axes, namely assessment methods, activity metrics, frequency of errors, session time/ last access, collaboration metrics and engagement and performance metrics. Each axis is presented by describing all related elements recorded in CMX that could be used in learning analytics. Finally, this framework was implemented in a specific mathematical model and exemplary instances of players' recorded behaviours were presented.

6. System's architecture and server's optimization performance

The architecture of educational game systems is fundamental for its performance and the user's final experience within the game. This is even more crucial when the game is an MMORPG, where many players participate at the same time, receive feedback for the position of each player inside the game and interact with other dynamic entities.

One critical problem that presents a significant research interest and remains a challenge is the identification of how we can address the resources consumption in an educational MMORPG in order to ensure its proper operation. In other words, it is important to study how all of the beneficial features of an educational MMORPG can be put to use, while the server hosting the game runs with no problems from beginning to end. This way, teaching and learning is enhanced, students remain immersed in the game's environment, without disorienting their interest or motivation to win, the system does not present delays in its responses and manages to produce the appropriate feedback to the teacher (learning analytics).

For the successful solution of all the different problems that might occur and are crucial to the server system's performance, a number of interesting algorithms have been constructed and are presented in the related work section.

Chapter 6 includes a section that presents the architecture of CMX, a system performance analysis and the corresponding creation of a mathematical model that estimates how active players are hosted within an MMORPG game in a server so that the server administrators can take the necessary measures that will ensure optimized server's performance (Malliarakis et al., 2014d). Moreover, the chapter includes a process to ensure server's performance optimization. The last section presents the evaluation of this optimization with an experiment using the CMX game.

6.1. System's architecture

CMX follows the typical MMORPG architecture, which is a multi-tier client-server architecture. The game has been programmed using the .NET C# programming language, version 4.0, the data has been stored and are retrieved using the MySQL

language and locally in some files using XML, while the game also includes libraries of the OpenGL specification. The usage of OpenGL was supported by the TAO Framework.

C# is a computer language that is used very commonly when developing games because it provides numerous advantages. Such advantages include the existence of the common intermediate language (CIL), which is an assembly language C# compiles the code to. Once that compiling is done, CIL is run on a virtual machine, as shown in the following figure, which in turn creates the machine code. The virtual machine of C# is called CLR, and there we can write the code only one time and the machine will run it on all CPUs that support the CLR virtual machine, for example on a PC (x86), Xenon (e.g. XBOX 365 console) and PS3 Cell (Play Station console). The usage of the CLR machine also helps the system to not crash, since it can detect errors very efficiently and it automatically allocates and frees memory when needed.

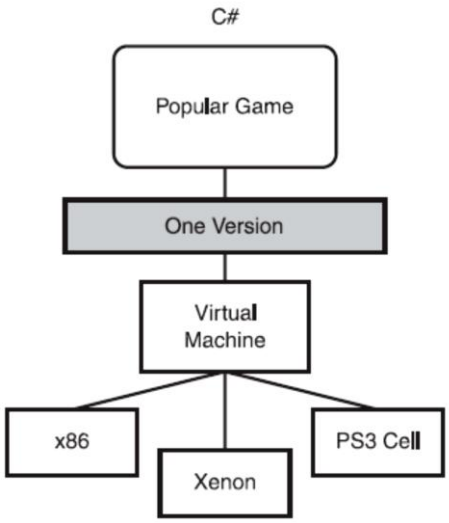

Figure 6-1: Usage of C# architecture for the development of the games (Schuller, 2011)

Two main programming techniques were utilized for the development of CMX; namely KISS (Keep It Simple Stupid) and DRY (Don't Repeat Yourself). These frameworks facilitate the reduction of workload, since when the code is repeated in few places, then

when a change is needed, there will be fewer instances required to change in the entire code.

It should be noted that the development of the game included the usage of the source control system named Subversion[1]. This is a free open source control system that allows problem-free Windows integration and allows the programmer to create branches on complex and lengthy programs such as games. This way, we can divide our programs in different versions and modes. As an example, if we have developed a game that is working with no problems but we believe it would be more efficient if it includes a first-person shooter mode, we can create a branch, where all the changes in the code can be made in that branch's repository and not affect the game's initial code.

Finally, the game development process has used NUnit[2] for unit testing. This helps the programmer test new parts of code in order to test their operation. Unit testing is an intuitive way to assemble all the new parts of code and run them each time the code is compiled. This way, if something happens to not work correctly in a previously working program, we understand that some part of the code is broken.

The following table shows the different tables of the database created for CMX along with their description. As it can be seen, the information regards all accounts created within the game, the characters of the game, some entries in regards to the players' scores, the objects and the maps within the virtual worlds etc.

Finally, the database tables hold information on the Senseis, the Iron Senseis and the Gold Senseis, the activities assigned, and also statistical data and log files that record the players' movements inside the game and the data created during players' evaluation by the Senseis. In Table 6-1 there is a short description from each table of the database of CMX.

[1] Apache Software Foundation, 2015. Apache Subversion 1.9.2. Apache License, Version 2.0. Available at: <https://subversion.apache.org/download.cgi> [Accessed 05 October 2015].

[2] NUnit 2.6.4. Portions Copyright © 2002-2012 Charlie Poole or Copyright © 2002-2004 James W. Newkirk, Michael C. Two, Alexei A. Vorontsov or Copyright © 2000-2002 Philip A. Craig. Available at: <http://github.com/nunit/nunitv2/releases/download/2.6.4/NUnit-2.6.4.msi> [Accessed 05 October 2015].

Table 6-1: Tables from the database of CMX

Name	Description
Account	This table contains user accounts' data.
Active_alliance	Alliances that are created and used inside the game.
Active_character	Characters' data that a user can use.
Attackable_alliance	List of alliances that another alliance can attack.
Ban_account	User accounts that are banned from the game.
Character_account	The characters that are associated with each account.
Equipped_character	Items that a character has already equipped.
Events_guild	Log data from the guild creation.
guild	Guilds that are created from users
Hostile_alliance	Alliances that an alliance is hostile towards by default.
Ip_account	The IPs used to access accounts.
Item	Information regarding the items.
Item_active	Items that are created and used.
Item_rewarded_quest	Items that are given as a reward after finishing a quest.
Items_finish_Quest_	Items that are required to finish a quest.
Items_shop	The items in a shop's inventory.
Items_start_quest	Items that a user should acquire to start a quest.
Map	Information regarding the dynamic entities of the map.
Member_guild	The members that participate in a guild.
Quest	The quests that are set inside the game.
Quest_character_status	Information regarding character's status inside a quest.
Quest_finish_quest	Quests that are required to be accomplished to finish this quest.
Quest_start_quest	Quests that are required to be accomplished to start this quest.
Senseigold_answers	The right answers of Gold Senseis' activities.
Senseigold_answers_player	Players' answers of Gold Senseis' activities.
Senseigold_questions	The activities set from Gold Senseis.
Senseigold_status_player	The status of each player regarding the activities of Gold Senseis.
Senseiiron_answers	The right answers of Iron Senseis' activities.
Senseiiron_answers_player	Players' answers of Iron Senseis' activities.
Senseiiron_questions	The activities set from Iron Senseis.
Senseiiron_status_player	The status of each player regarding the activities of Iron Senseis.
Shop	Information regarding the shops.
Skills_per_character	The skills that are associated with a character.
Statistics _characters	Statistics used for characters.
Statistics_guild	Statistics used for guilds.
Statistics_item_template	Statistics used for item templates.
Statistics_items_buyed	Statistics used for items that have been purchased from a shop.
Statistics_items_consumed	Statistics used for item that have been consumed.
Statistics_items_created	Statistics used for items that have been created.
Statistics_items_sold	Statistics used for items that have been sold.
Statistics_killed_characters	Log of events of user kills
Statistics_map	Statistics used for maps.
Statistics_network	Statistics regarding the network details, bugs and

	latencies
Statistics_quest	Statistics used for quests.
Statistics_quest_accepted	Log of events of quests accepted.
Statistics_quest_cancelled	Log of events of quests cancelled.
Statistics_quest_completed	Log of events of quests completed.
Statistics_shop	Statistics used for shops.
Statistics_shop_buyed	Statistics used for items that a shop has sold.
Statistics_shop_sold	Statistics used for items that a shop has purchased from users.
Statistics_user	Statistics used for users.
Statistics_user_consumed_item	Statistics used for the items that a user has consumed.
Statistics_user_kills	Statistics used regarding users' kills.
Statistics_user_levels	Log of events of user levels.
Statistics_user_shopping	Log of events of buying, using and selling items.
Status_per_character	Status consequences for a character.
Template_character_equipped	Equipped items on a character template.
Template_character_inventory	Items in a character template's inventory.
Template_character_questprovider	Quests provided by character templates.
Template_character_skill	Skills that are associated with the character's template.
Template_item	The templates used for the creation of the items.
Template_per_character	Character templates used for the characters' creation.
Winners	The winners of each game.

Many of the dynamic entities are stored in local files written in XML. This way, static data can be directly accessed without causing further workload on the server and the network's bandwidth. Such data include the maps, the dialogues with the Senseis and different items that can be located in the virtual world. The Figure 6-2 shows a part of the dialogue that has been stored in XML.

```xml
<?xml version="1.0" encoding="utf-8"?>
<SenseiDialogue>
  <ChatDialogs>
    <Count>33</Count>
    <Item0 _extraInfo="516,19;700,83;169,534;1018,322;683,303">
      <ID>0</ID>
      <Title>Inn Keeper</Title>
      <Items>
        <Count>10</Count>
        <Item0 _posX="95" _posY="137">
          <ID>0</ID>
          <Title>
          </Title>
          <Text>Hello! Are you ready to learn about arrays?</Text>
          <IsBranch>False</IsBranch>
          <Responses>
            <Count>2</Count>
            <Item0>
              <Value>0</Value>
              <Page>2</Page>
              <Text>Yes, of course!</Text>
              <Actions>
                <Count>0</Count>
              </Actions>
              <Conditionals>
                <HasConditionals>False</HasConditionals>
              </Conditionals>
```

Figure 6-2: Dialogue stored in local XML file

Similarly, the Figure 6-3 shows a local XML file that stores information on the maps for the virtual worlds of the games.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Map>
  <Header>
    <Name>Town</Name>
    <ParentMapID>1</ParentMapID>
    <HasMusic>True</HasMusic>
    <MusicID>1</MusicID>
    <Size>2000,2000</Size>
    <Indoors>False</Indoors>
    <CustomGravity>False</CustomGravity>
    <Gravity>0,0</Gravity>
  </Header>
  <Walls>
    <Count>787</Count>
    <Item0>
      <Position>-16,1968</Position>
      <Size>2016,32</Size>
      <IsPlatform>False</IsPlatform>
      <BoundGrhindex>0</BoundGrhindex>
      <DirectionalBlock>None</DirectionalBlock>
    </Item0>
    <Item1>
      <Position>0,-16</Position>
      <Size>64,2016</Size>
      <IsPlatform>False</IsPlatform>
      <BoundGrhindex>0</BoundGrhindex>
      <DirectionalBlock>None</DirectionalBlock>
    </Item1>
```

Figure 6-3: XML file with maps of virtual worlds

6.2. Related algorithms for optimization of server system's performance in online multiplayer games

The literature review carried out indicated a limited amount of studies that investigate and measure the server system's performance in online multiplayer games, though no studies were identified for focusing on educational games and especially in the field of computer programming. All these studies take into account that multiplayer games can support either TCP/IP or IPX internet protocols and can be played either over LANs or the internet. The goal in each study was to examine the parameters that affect the server's operation and thus should be taken into consideration while optimizing a game's performance. It should be noted that studies on optimizing applications and mainly games has been a research focus for over two decades. However, there is a distinction between optimizing network resources and optimizing the performance of the packets' transmission on the servers, by monitoring CPU usage and RAM efficiency.

A case study carried out by (Suznjevic et al., 2009) using the "World of Warcraft", identified the specific features of an MMORPG that affect network performance and latency. Their focus was on determining which of the game's actions mobile devices can support; however, they provide useful information that are in accordance with our scope of interest. More specifically, the authors list actions that are usually executed by players in the game across a number of categories, which are deemed to shape the network traffic. A representative overview of these actions' categories is presented in the following figure.

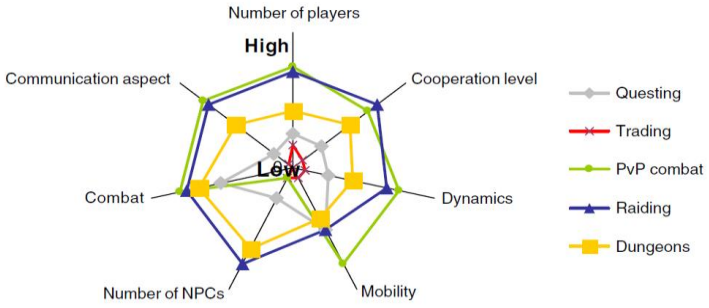

Figure 6-4: Categories of actions within "World of Warcraft" (Suznjevic et al., 2009)

Moreover, (Suznjevic et al., 2009) recorded the specific characteristics that influence network traffic, such as the size of each packet transmitted, the time between each packet's transmissions, the bandwidth that is used and the load of each packet. Other characteristics observed that could affect network performance include the session duration and each player's interest in the actions showed in Figure 6-4. Also, (Borella, 2000) and (Chen, 2006) have created their own models on game network traffic. According to their studies, interactive gaming is extremely delay-sensitive, which in many cases can incorrectly determine who wins or loses a game.

Another research carried out by (Chen, 2006) studies the "ShenZhou Online" MMORPG and indicates that the characteristics that constitute the game's traffic include: the average bandwidth, the volume of the traffic in the client and the server, the time interval status update is carried out (e.g. how periodically is the game's status synched), the overall activity of the game and the game sessions per hour.

To examine the network traffic based on the transmit-receive cycle, Borella (2000) developed models and tested them on the "Quake 2" game. These models indicated that the performance of the game is highly dependent on the clients' machines as well as on the hosts' CPU speed. The exact characteristics that were assessed regarding the network traffic include the server's CPU, Operating System and Random Access Memory.

Taking into consideration these characteristics and measuring the amount of packets transmitted, the study calculated the mean and standard deviation of the transmission volume as well as the time passed between transmissions. This study does not elaborate on possible techniques of optimizing the traffic and thus the game's performance levels, but suggests such actions as future work.

According to Farber (2002), the process followed throughout a game's session includes the transmission of packets from the server to the clients logged on regarding the game's state. Each client receives the data and processes the information by synchronizing their local game's state with the server's. This way, all players in different clients can interact with the game while having the exact same visual representation of the game's status at any period interval. In the study, Farber (2002) used the "Counter strike" game and developed a model to measure its network traffic. This game is ideal for such examinations, since, like most multiplayer games, it is built under client-server

architecture and presents a very high traffic volume available to be examined due to its popularity.

In this model, they studied the server and clients independently, without taking into consideration any possible inter-dependencies between them that could affect the network traffic. This can be also seen in Figure 6-5, where individual measures are taken for server and client, and the characteristics that are evaluated are the size of the packets transmitted and received as well as the time spent between each transmission.

	Server (per client)		Client	
	characteristic	**approximation**	**characteristic**	**approximation**
(burst) interarrival time	peak = 55 ms mean = 62 ms coeff. of variation = 0.5	Extreme (a=55,b=6)	mean = 41.7 ms coeff. of variation = 0.24	Deterministic (40 ms)
packet size	mean = 127 Bytes coeff. of variation = 0.74	Extreme (a=120,b=36)	mean = 82 Bytes coeff. of variation = 0.123	Extreme (a=80,b=5.7)

Figure 6-5: Traffic evaluation for "Counter Strike" (Farber, 2002)

As optimization techniques, the study suggests carrying out predictions on the game's status (e.g. interactions of the players with the game's bots, objects and other players), a method initially proposed by (Bernier, 2001) and (Yu-Sheng, 1997)

An additional study that researched network traffic is (Jehaes et al., 2003). The mathematical model they use to measure traffic delay is shown in the following equation.

$$T = T_{\min} + \frac{S}{R_{eff}} + T_{que,tot} \quad \textbf{(6.1)}$$

The measures included in the above equation include all features that the authors suggest strongly influence a game's performance, where T_{min} is the minimum packet processing delay in a transmission, S is the packet size measured in bits, R_{eff} is the effective link rate in a packet transmission and $T_{que,tot}$ is the total delay while the packets are queuing.

Additionally, Chang & Feng (2003) studied the players' behaviour according to network traffic as well as the factors that affect it. They also chose the "Counter Strike" multiplayer game and examined the way the game's state is constantly updated by the server and transmitted into all the different clients as well as how the players interact

with the game's objects. This study does not focus on the actual game's performance and the characteristics that could optimize the process.

Another research by Chang and Feng (2003) carried out a thorough investigation on the different factors that influence network traffic and multiplayer games' performance. More specifically, the affecting factors were categorized into three axes, namely "Resource Management", "Consistency and Responsiveness" and "Logical Platform and Application".

The quantity of resources transmitted from the clients to the servers and vice versa is a feature that was pointed out by all studies as influential in network traffic. Smed et al. (2002) move on to elaborate on the individual elements that comprise the resources sent in a multiplayer game. These elements have been identified and form an equation called the "Information Principle Equation", shown in (6.2).

$$\text{Resources} = M \times H \times B \times T \times P \qquad \textbf{(6.2)}$$

According to this equation, the different information types that are part of the resources in a game include the number of messages (*M*), the average number of nodes where a message has to be transmitted (*H*), the average of the network bandwidth that can successfully transmit a message without losses (*B*), the time in which each message is transmitted over the network with the corresponding bandwidth to the corresponding node (*T*) and the actual number of cycles the processor has to complete for the successful message transmission (*P*). The equation shows that all these parameters are strictly interconnected; thus, any attempt to lower one in an attempt to enhance the resource transfer will lead to the necessary increase of one or more of the others.

Moreover, Singhal & Zyda (1999) investigated the factors that influence the game's consistency and responsiveness, since both these elements indicate a game's high quality. According to their study, the factors that influence consistency are the bandwidth volume, the latency volume and the number of nodes that are connected to the system. Moreover, the factors that influence responsiveness include fast resources transmission, lack of lost messages and again bandwidth and latency.

Other studies that have been carried out research approximately the same features in assessing multiplayer games' performances and delay factors (Quax et al., 2004; Guo

et al., 2003). The most commonly mentioned factors include bandwidth delay and consumption (Armitage, 2003; Sheldon, 2003). However, all aforementioned studies focus mostly on network traffic of strategy or shooting games and less on MMORPGs that require more resources and in different formats, as it is usually the case in the educational domain. All changes in a player's status (e.g. score, new level, new task achieved) are required to be recorded and presented as a feedback not only to the player in question but also to all other players that could see him/her in their view scope. This way, network traffic in MMORPGs involves changing recordings and multiple message documentations and transmissions to players of the game. This is even more required in multiplayer games that are used in education, since any system delay or failure will not only reduce the quality of the users' entertainment but of the learning process as well.

6.3. System analysis and performance optimization

The architecture on which most MMORPGs are based comprises of one or more servers and several clients inter-connected through internet access. The server is usually protected by a firewall or a proxy server mainly for stability and safety reasons. The server administrator is responsible for ensuring the stable performance of the system and the optimization of the data stored in the server's database. This data is gathered by the game's system, documented in the database connected with the game and it includes information about each player, such as the player's status, the world's condition, the possible actions that can be executed as well as any instances created by the chat function usually embedded within MMORPGs.

More specifically, the player's status includes all elements that relate to the specific player. Such elements can be the player's statistics which are constantly renewed along with his/her corresponding evolvement during the game (e.g. current difficulty level of programming concepts, health points, strength etc.), the player's position in the virtual world and all the game objects under the player's possession. The world's status comprises of all the characteristics of the virtual world, such as the world map, the virtual players (e.g. intelligent agents) that are programmed by the game creators as well as actions that take place by each player. The actions are all the possible moves and operations that a player can realize or that are occurring at a specific moment (e.g. a spell is being cast; a player's character is walking etc.). These actions involve both the ones

carried out by the actual players and agents. Finally, each MMORPG contains groups of players, thus most games include a chat tool that supports the group members' communication through the broadcasting of messages amongst them.

As the number of active players increases, the chat function that transmits simple text messages continues to have a relative small complexity, so it should not be taken into consideration during the search for the system's complexity simplification. Moreover, the internet connection speed will also not be considered as a factor, since it could vary depending on the client and its changes cannot be predicted.

It is important that the system hosting a MMORPG is studied for the identification of all factors that influence its performance and thus could potentially lead to system overload. This way, we can determine which elements to take into consideration during load balancing efforts. To this end, this section will propose a mathematical model for the appropriate estimation of these factors and the subsequent configuration of the system should their values pose a risk to the game's successful execution by learners.

For example, if we assume that we have one player that is playing in a MMORPG, then the server's performance would depend on the player's status, the world's status and the possible actions that are caused or that can be caused. All these elements are recorded and documented in a database installed within a database server inside the main server; so the more the data produced during a game's operation the heavier the capacity transferred from the individual clients to the main server and stored into the database server. Hence, the rate in which the above elements' values change can significantly affect the server's overall performance and potentially obstruct the game's proper execution and subsequently the learning process's successful realization. To this end, we are proposing a model that will estimate the computational cost and the server administrators can take the appropriate actions for possible server upgrading based on these estimations (i.e. dynamic prediction of resources commitment).

This model includes the usage of an algorithm that measures and grades the players' behaviour, i.e. the value changes of the varying factors that we are examining for each player as they are being stored in the system's database. The indicators can take values between the scale [0.1, 10] and are considered to be versus time, which means that

we are recording the activity occurring across time, as transmitted from clients to the server. This activity's value is stored as a vector in the database as shown below:

$$\overrightarrow{S_{CS}(t)} = \left\langle P^{cs}(t) \middle| W^{cs}(t) \middle| A^{cs}(t) \right\rangle = \left\langle s_1^{cs}(t) \middle| s_2^{cs}(t) \middle| s_3^{cs}(t) \right\rangle \qquad \textbf{(6.3)}$$

CS is the indicator of the score that suggests the client-server message transmission, while *t* is the time parameter, since the values that these variables take, change through time. The S_{cs} (Score) is a vector that corresponds to each player and indicates how active the player is in the game's progression. This score is dependent on the player's status' change rate as opposed to a) the other players' status (variable P^{CS}), b) the virtual world (variable W^{CS}) and c) the possible actions that can be executed (variable A^{CS}). With $P^{CS}(t) \mid W^{CS}(t) \mid A^{CS}(t)$, we indicate that in every time instance the variables' values change and each new value is documented in the database over time. We further map the P^{CS}, W^{CS} and A^{CS} variables to the indices $x = \{1, 2, 3\}$ respectively in order to alleviate some of the complex representation of the relation and, thus ratings are finally represented as s_x. For example, in a game, when a player moves towards the opponent, the changes that occur are the following: a) his relative position towards the opponent as he is recorded in his screen, b) his absolute position for the virtual world and finally, c) the possible actions that can be executed and the results that can be presented, e.g. when he is near enough he can damage the opponent, an action that could not have happened if he had been far away. Similarly, in a multiplayer educational game oriented to learning programming if a player tries to make moves or other actions by writing lines of code, the changes that occur are relative to the other players' situation and their code should be configured accordingly.

Moreover, we can estimate that the metric of the activities' score for each player at a t_i time by the *c* client to the *s* server is the sum of the P^{CS}, W^{CS} and A^{CS} metrics. In order to avoid extreme values, we logarithm them, so that they end up being in a [-1, 1] range, where the negative values mean that the player is less active, while the more the values tend towards 1, the more active the player is. This process is mathematically portrayed by the following relation:

$$s_{cs}(t) = \left| \overrightarrow{S_{cs}(t)} \right| = \sum_{i=1}^{3} \left[\log s_i^{cs}(t) \right] \qquad \textbf{(6.4)}$$

Each player's final score is configured by his interaction with the other players as well as by the agents. The difference lies in that we can shape the system players' behaviour so that we ensure optimized load balancing, whereas we cannot intervene in the agents' behaviour, as it is automatically determined by the game system.

Finally, two indicators are defined, namely the u_s indicator that symbolizes the number of the player's interaction with the game's agents and the u_o indicator that symbolizes the number of the player's interactions with the other players. So for a player *a*, the final relation is configured as follows:

$$TS_{cs}(t)=\frac{u_s}{u_s+u_o}\cdot\frac{\sum_{\forall t_i<t}\left[s_{cs}(t_i)\cdot t_i\right]}{\sum_{\forall t_i<t}t_i}+\frac{u_o}{u_s+u_o}\cdot\sum_{\forall j\neq a}\frac{\sum_{\forall t_i<t}\left[s_{ja}(t_i)\cdot t_i\right]}{\sum_{\forall t_i<t}t_i} \qquad \textbf{(6.5)}$$

The above relation estimates the Total Score (TS_{cs}) for a player, which depends on values of the elements stored in our game database. In the above relation of the Total Score (TS_{cs}), the first sum refers to the player's (α) interaction with the system (e.g. movements inside the environment, actions towards game's agents), while the second part of the relation refers to the player's (α) interaction with the rest of the players (j where $j \neq a$) for each previous time instance $\forall t_i < t$). With the addition of the $\frac{u_s}{u_s+u_o}$ in the relation we measure the multitude of the user's interaction with the system while with the $\frac{u_o}{u_s+u_o}$, we measure the multitude of the user's interaction with the rest of the players. It is also important to mention that since these variables change through time, the last values they take (i.e. the most recent) should be considered more in the final value of the player's total score. To represent this, as we show in the relation, we multiply the rating calculated at a specific time instance t_i, with itself. For example, an instance of the database table that holds this information is depicted in the following Table.

Table 6-2. MMORPG database instance of a player's recorded activity

Player ID	P^{CS}	W^{CS}	A^{CS}	t_i	u
2	0.2	0.1	0.6	1	S
2	0.5	0.1	0.4	2	S
2	0.3	0.1	0.3	3	O

Let us assume that we are trying to measure the total score for the player with an ID 2. This player has executed activities during three different time instances within the game as shown in the t_i column. Moreover, the *u* column shows that two of these activities include interaction with the system, while the last one represents interaction with other players of the game. Using the proposed model, the value of the total score will be estimated taking into consideration the fact that the player's interaction with the system's agents (the first two rows that have the value *u*=*S*) has increased in the 2nd time instance, which shows he is becoming more active and thus could need more resources in the future. According to the above table and the relation (5) of the model, the total activity score for the player with ID=2 and instance t_i=4 is $TS_{CS}(4)$=0.44, which means that his behaviour is very good and according to the goals of the game. We could easily understand that without the algorithm we would have to use the same quantum of resources for all the players, but now that we know their behaviour, we could easily customize the game's responses and the corresponding needed resource commitment.

Next, we demonstrate a way to reduce the big computational cost at run-time indicated by the double sums in the last relation of the previous section. The run-time computational complexity of relation (5) is O(*n***y*), where *n* is the number of agents acting in the system and *y* is the total amount of transactions. Both of these numbers increase rapidly in a MMORPG game. More specifically, the $\sum_{\forall t_i<t}[s_{ja}(t_i)\cdot t_i]$sum does not need to be estimated at run time, but incrementally, as each rating vector is gathered and stored. We re-symbolize this sum with the $\sigma_i^{ja}(t)\forall j \neq$ avariable. Additionally, the $\sum_{\forall t_i<t} t_i$ sum can also be estimated incrementally and it will be replaced with the $\sigma_i^{CT}(t)$ variable. This way, relation (6.5) is transformed into the following:

$$TS_{CS}(t)=\frac{u_s}{u_s+u_o}\cdot\sum_{i=1}^{3}\left[\sigma_i^{CT}(t)\right]+\frac{u_o}{u_s+u_o}\cdot\sum_{i=1}^{3}\left[\sigma_i^{jb}(t)\right] \qquad \textbf{(6.6)}$$

This relation's complexity is O(*n*), where n is the total number of agents and players inside the system. The sum in the three dimensions does not count because its limit is stable and already known that it is 3. Thus, we are left with only a sum inside the second addendum. The initial relation (6.5) has a run-time complexity of O(*n***y*), i.e. the number of agents and players (*n*) over the number of transactions (*y*), which is much larger than the complexity of relation (6.6).

In conclusion, the optimization of the algorithm is possible because the player's active score is calculated and stored in the database as a total number, instead of the system calculating the score's value after each player movement.

6.4. Evaluation of the optimization model

In order to evaluate the proposed model, a specially designed test bed is used in a 3GHz Core 2 Duo Server with 4GB RAM. We ran our scenario two times, maintaining the exact same conditions in both cases. However, in the first case, we do not exploit the optimization algorithm and in the second case we do. More specifically for the second case, we investigate if and how much the server's performance improves, while at the same time adding virtual players inside the game.

6.4.1. Test bed overview

We used a special virtual world without any obstacles inside CMX, called "Arena" and we placed many virtual players with behaviours of escalating randomness. This was done in order to simulate the randomness of the players and create a realistic experiment. We moved on to creating additional virtual players with predictable behaviour, so that the appropriate resources can be committed according to their behaviour. More specifically, we created virtual players with predictable behaviour so that we can commit specific resources based on that behaviour, and a group of virtual players with completely random behaviour, which will simulate a possible un-predictable behaviour of an actual player at a specific period.

We continued to load more players in the virtual world and study the server's reaction without and with applying the algorithm. In the first case, as we add the virtual players, equal resources for each player will be committed without any distinctions (e.g. for 100 players, 1% of the entire CPU usage will be used to accommodate the individual traffic). However, this could lead to the engagement of redundant resources because not all players need the same amount of CPU usage, depending on their actions in the game. Thus, in the second case, with using our algorithm, we store in the database features such as how active each player is etc., as mentioned in the previous section. This way, the exact amount of resources needed per player will be calculated and committed.

6.4.2. Use case scenario

The test bed in the experiment was populated with excellent, good, fair and bad virtual players, depending on their level of computer programming prior knowledge. More specifically, 100 virtual players were used, 10 of which were excellent, 30 good, 20 fair players and 40 bad players. By excellent players we mean players with a completely predictable behaviour, where we know for example, that when such a player at a time t_1 is at a specific location in the virtual world in pursuit of a task, this player will continue walking $\forall t_i > t_1$ times until the goal is achieved. A good player is also navigating through the game in a predictable way; however, there is a small chance that some actions will not be predicted by the server, such as a change of course in the Arena. The probability that a behaviour will be unpredictable increases with fair players, while bad players' actions are completely random. These players constantly make non sequential movements and are not aiming to fulfil the educational game's objectives.

We included several players with unpredictable behaviour in our experiment because we wanted to test our algorithm in extreme situations where the server's overloading with resources, since we already can estimate that in excellent and good players, the game's performance will be adequate. Thus, we endeavour to prove that in the case of an excellent or good player, the minimum possible resources are bound in the CPU, while these increase in the case of fair and bad players. As an overall result, the amount of resources needed with the usage of the optimization algorithm should be significantly less than if the same amount of resources had been bound for all players.

We positioned a player on the "hackers" team and one in the "crackers" team in the game's Arena. Following, we equally populated both teams with virtual players that exhibited similar behaviour, while monitoring and recording the sent and received messages packets as well as the CPU usage levels. The players of each team tried to win all other players of the opposite team while executing assigned tasks and gathering weapons and spells. It should be noted that we provided unlimited lives to our virtual players to prevent possible elimination from the game.

During a solely entertaining multiplayer game with the same conditions as the ones we have established in our experiment (e.g. 100 players, unpredictable behaviour of movements, constant transmission of multiple resources), a possible overloading of the system would most likely last a few seconds. This is because, in most cases, the majority

of the students lose and exit a game rather quickly and thus the resources committed get reduced (Chang & Feng, 2003). However, in an educational setting we require the majority of the students to go through all the game's levels in order to obtain the maximum possible knowledge and skills. Thus, any latency on behalf of the server could significantly affect the learning process, the students' motivation as well as the entire game's configuration.

A server's operation is significantly increased when the players interact with each other in the Arena while it is less extreme during the training phase. This is because a large amount of resources are required during the players' confrontations with each other such as each player's absolute and relative position that is occupied in the same screen of the virtual world. This information flow needs to be constant since the data is ever changing, which increases the server's CPU usage.

6.4.3. The results

The experiment was executed without any issues, and we closely monitored all the information recorded in the server's log file. We initially performed the CMX scenario as-is and thereafter performed the use case scenario that involved the usage of the optimization algorithm. The results gathered are presented in Figure 6-6 and they show that when the first 10 players with excellent behaviour are inserted in the Arena, the optimization algorithm seems to reduce the CPU usage by about 5%, since their behaviour has been programmed to be predictable, and thus the algorithm knows what resources they need. The algorithm starts monitoring even more the server's resources when additional players are inserted, even though their behaviour is not completely predictable.

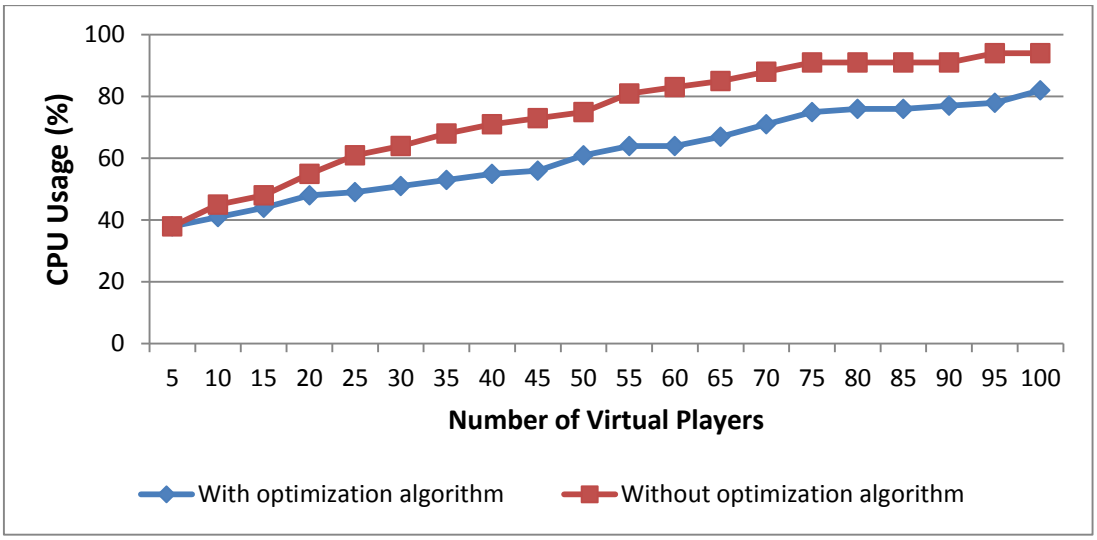

Figure 6-6: CPU Usage with and without optimization

Based on the results, the first 60 players have predictable behaviour, since they play according to the game's rules. With these players, we observe that the algorithm is constantly reducing the CPU's computing cost. More specifically, the CPU usage is reduced down to 20% when 60 players in total are a part of the game's environment. However, when we start inserting virtual players with totally unpredictable behaviour, the algorithm's performance is decreased since it cannot predict what resources should be committed. For this reason, we can see three significant drops in Figure 6-6, which occur when players with more unpredictable behaviour are inserted. Nonetheless, Figure 6-7 also shows that the algorithm manages to maintain an approximately 15% of CPU usage reduction.

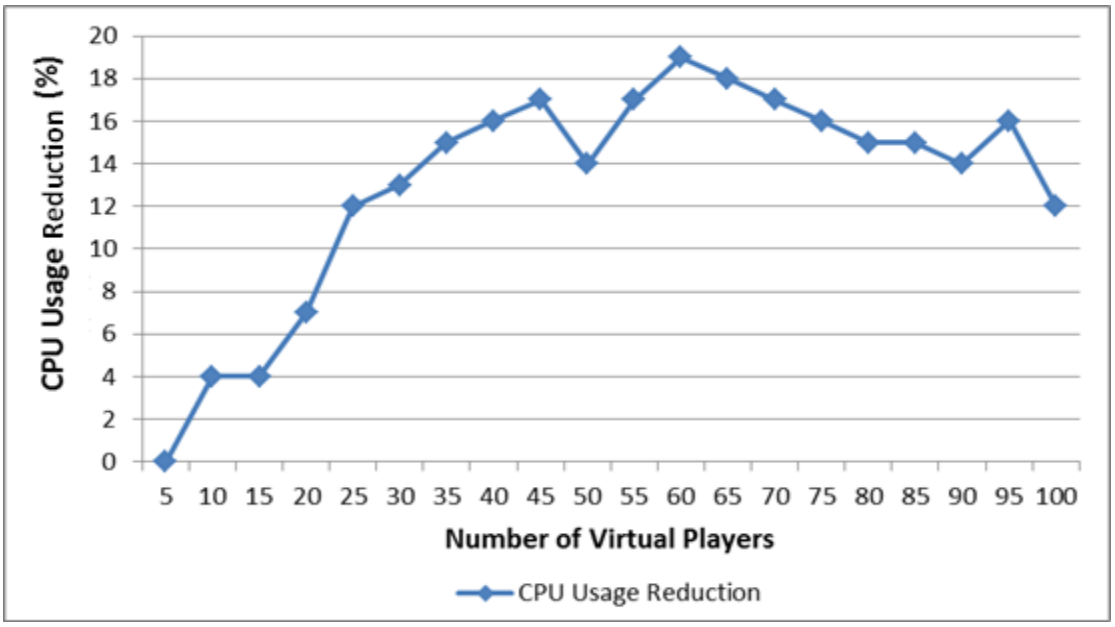

Figure 6-7: CPU Usage Reduction

It should be noted that 15% of CPU usage reduction is a significantly satisfying result for the server's performance, as with the corresponding release of resources, the game can operate smoothly and without performance issues and can also host more players. This optimization ratio will remain stable even if the players are dramatically increased. This is because our algorithm depends only on the behaviours of the players in the game and it already takes into consideration different types of behaviours that new players could show (excellent, good, fair and bad). Similarly, the second diagram shows how the CPU's computing cost reduction is increased. It is worth mentioning that the low reduction values shown in the diagram occur due to the addition of new users with constantly extra random behaviour.

Furthermore, it should be noted that when we use the CMX's server optimization algorithm, the RAM usage is slightly increased since we store more information; however, in an educational game this increase is very low compared to the CPU usage, and therefore it does not affect negatively on the game's operation.

6.5. Summary

Chapter 6 presented the architecture of CMX including a system performance analysis and proposed a new methodology that can achieve optimization of the load balancing in educational MMORPG for computer programming and the bestowal of the proper and only necessary resources without overloading the system. To this end, we analysed existing published works that identified the factors that influence a computer game's performance. These works, however, focus mostly on network traffic of strategy or shooting games and less on MMORPGs that require more resources, as it is usually the case in the educational domain. Thus, the existing studies could not be used as a comparison for the model presented in the current study.

Our model is based on the notion that not all players are equally active in a learning process and in a game's activities, and thus they do not require the same amount of resources committed by the main server to their client. Therefore, we monitor a player's activity along with other parameters within the game. In the end, we manage to commit only the essential resources for each player, where traditionally we would commit the same amount resulting in abundant resources' commitment and thus server overloading.

The novelty of our approach is the cumulative storage of information in the database in an abstract manner, in order to exploit in the maximum degree the gathered information regarding the players' predictable behaviour and thus ensure the proper commitment of CPU and RAM memory. It should be noted that the reasoning behind the exploitation of the cumulative information can be applied in any MMORPG in combination with algorithms that address the optimization of network resources and packet transmissions for the optimum system's performance.

Moreover, the chapter elaborated on the MMORPG CMX and indicated the importance of optimizing such a server's performance to ensure consistent and responsive learning processes. To this end, we applied the proposed algorithm to CMX by creating a use case scenario and inserting virtual players instead of using students. This way, we could control the entire process on the server's point of view, since all virtual players' behaviours were simulated in accordance to the types of behaviours usually presented by students. The results of the test bed indicated that indeed the optimization algorithm managed to decrease the CPU usage by a total amount of 15%,

which should also be the case during the implementation of the game in educational settings. Due to the fact that a more accurate and dynamic prediction of resources needed is carried out with the algorithm, in total less resources are committed than the fixed amount of resources that are usually committed by default.

7. The effects of CMX on learning and teaching Computer Programming

This chapter aims to examine the effects of using CMX on teaching and learning computer programming. The effects of CMX are assessed through three studies with third-year undergraduate students (pilot study), first-year undergraduate students (1st evaluation study) and post-graduate students (2nd evaluation study). All students evaluated various aspects of CMX by filling in a questionnaire designed based on the evaluation framework described in Section 7.2 that is in accordance with the CMX design framework described in Section 5.1. The evaluation process was based on six research questions that aimed to examine all features of the game and how these affect students' learning experience in computer programming. More specifically, the features examined include: the game's performance, its entertaining and motivating elements, the educational aspects of the game, and how students assess their interactions within the game. Finally, the evaluation investigates whether the game's use in computer programming courses can indeed increase students' performance.

7.1. Empirical related work on educational games for computer programming courses

Various research studies have been carried out on games whose features support the teaching and learning of computer programming. The majority of these games include a particular scenario that aims to cover a specific computer programming unit, whereas some games - fewer in number - cover multiple learning objectives and theory units. All the following games are described thoroughly in Chapter 4 and below we focus on the evaluation procedure.

The educational game Catacombs (Barnes et al., 2007) belongs in the first category whose aim is to familiarise students with variables definition and usage of if-statements and loops. The player becomes a wizard and answers questions in order to fill out a code correctly, utilizing a specifically designed mini-language. The evaluation of the game in real world educational settings showed that the students found the experience both entertaining and beneficial, while they found the game to be interesting and attractive. Specifically, two evaluations of the game were conducted with 13 and 8 random students respectively, all of whom had basic knowledge in Computer Science

and were enrolled in the corresponding Department of the University of North Carolina (UNC).

In the game Saving Sera (Barnes, Richter, Powell, Chaffin & Godwin, 2007) the players are required to undergo various quests, either by filling out proper code units in specific exercises or by sorting code segments in their proper place within the program so that they can progress to the next step. These activities enable students to learn and practice on recursion and if-statements. Saving Sera was evaluated by university students, where the evaluation process included the recording of a video as well as an interview after the game. The results indicated that students found Saving Sera entertaining and educational, but would have preferred additional scaffolding. As with Catacombs, the game was evaluated in two rounds, with 13 students and 8 students respectively, randomly chosen with varying levels of computer science knowledge. The students were mainly from the University of North Carolina (UNC) from the Computer Science Department in Charlotte.

Elemental: The Recurrence (Chaffin, Doran, Hicks & Barnes, 2009) is an educational game also developed for teaching computer programming. Initially, a brief demonstration of recursion introduced students to the theory and afterwards students were required to apply the depth first search algorithm by moving the protagonist across a binary tree and complete three missions. Forty-three (43) university students again from the Department of Computer Science at UNC, who were either taking or had completed the "Data Structures and Algorithms" course, took part in the application of Elemental. The purpose of the study was to gather formative feedback and the evaluation showed that the majority of students wanted to learn even more complex programming concepts through the use of a computer game.

Prog&Play (Muratet, Torguet, Viallet & Jessel, 2010) is a web-based strategy game that enables interactions amongst users. In this game students program their characters and try to form alliances in order to win. Fifteen (15) novice computer science students from the Institute of Technology at the Paul Sabatier University Toulouse III in France played Prog&Play. The initial results showed positive feedback regarding the game's entertaining elements and its usefulness as an educational tool.

Robocode is an educational game that aims to cover multiple learning objectives. Robocode (O' Kelly & Gibson, 2006) is a two-dimensional environment that aims to

teach the Java programming language and includes a virtual arena and a programming editor. Robocode was evaluated by 83 participants that were randomly drawn on from the online forum of the Robocode community. During this process, they were asked to write programming code that would solve a specific problem, which involved creating an army of robots that would have to beat their opponents. Students seemed to enjoy playing the game and the overall evaluation results were positive.

All the above educational games present interesting solutions that can increase programming performance. However, they focus on specific units of learning and support limited programming activities.

7.2. CMX Evaluation Framework

Taking into consideration the design framework that was described in chapter 5, an evaluation framework was designed that assists teachers in evaluating all the features of an educational game (Malliarakis, Satratzemi & Xinogalos, 2014b). More specifically, the framework includes 6 different axes.

1. *Students' knowledge in programming and interest in games*
2. *Game's performance*
3. *Game's entertaining and motivating elements*
4. *Game-student interactions and game's difficulty*
5. *Game's educational elements*
6. *Students' performance*

The above axes are described in the following subsections thoroughly.

7.2.1. Students' knowledge in programming and interest in games

The student's profile represents user characteristics such as knowledge, behaviour, and goals (Natkin & Yan, 2006). The CMX design framework has shown that the end user, i.e. the student, should be taken into particular consideration, since the learning objectives and outcomes are directly related to the student's particularities.

Studying student profiles during evaluation of the game by focusing on the interest they have in games, their specific knowledge and skills in the field of computer programming is essential. Overall, this can show whether existing knowledge of a domain or interest in participating in the learning process affects a student's performance in the game.

7.2.2. Game's performance

The performance of CMX during gameplay is a vital factor for its success in education. Proper system function with no delays or bugs can lead to high levels of user satisfaction and immersion in the environment (Desurvire, Caplna & Toth, 2004). To prevent system delays, it is essential to make sure that there are ample resources, such as bandwidth, CPU capacity, RAM etc. One way to ensure players' satisfaction is to sustain optimal server performance, which is especially important in MMORPGs that involve large numbers of users and real-time interactions. Performance can be hampered due to network traffic caused by latencies of the Internet, causing 'lags' which alter the graphical environment, thus disrupting users´ gameplay (Suznjevic, Dobrijevic & Matijasevic, 2009).

The present study researches ways in which resource consumption can be improved appropriately so as to guarantee game performance. This led to the design of the mathematical model that was described in Chapter 6 and that estimates the different factors that affect the server load and disrupt the proper operation of the game. When this model is applied, the server releases only the resources needed for the game and vice versa, and in so doing prevents both overload and network traffic. This allows the students to continue to be immersed in the game, and transmissions of feedback to the teacher (e.g. learning analytics data and results) are carried out successfully (Malliarakis, Satratzemi & Xinogalos, 2014c). The assessment process involves students being asked about their experience of the game as regards its performance. In other words, questions regarding players' satisfaction of movement, interactions and decision-making, as well as message transmission can provide insight that can be used to enhance the game's performance.

7.2.3. Game's entertaining and motivating elements

One of the most significant advantages of educational games is the ability to motivate and engage students in the learning process. This can be achieved by making

the elements of the environment interesting, attractive and entertaining. It is essential that the games capture students' attention from the start and that the students continue to be engaged in the game irrespective of the length of time that involves (Pagulayan, Keeker, Wixon, Romero & Fuller 2003). One way to do this is by providing features that attract students' attention and draw them into the game (Brown & Cairns 2004), such as a stimulating plot that that has a wide range of activities at varying levels (Johnson & Wiles 2003), which although challenge the student, do not discourage them from continuing the game. The activities should kindle students' interest and motivation by being at an appropriate level and in no way should overload their cognitive and memory capacities (Lazzaro & Keeker, 2004).

It is, therefore, essential to investigate students' perceptions of CMX regarding its entertaining features and motivational value. More specifically, students are asked to provide feedback on the degree that they were entertained by specific aspects of the game, which were included in order to increase their engagement and satisfaction. Examples of these are: the plot, the graphics, the learning activities and all other interactions with the game's elements and players.

7.2.4. Game-student interactions and game's difficulty

Students are able to interact with the game and within the game throughout their entire learning experience. Such interactions can include: the tasks they execute, their navigations across the environment, their chat sessions, their usage of the mini map tool that allows them to locate other plays and characters and then navigate to different parts of the world and their attempts to shut down the factory's server. All these features are strongly interconnected with the learning objectives set by the teachers at the beginning of the course.

The game's difficulty is associated to students' attitude towards their interactions with the system. This is highly dependent on the one hand, on the specific skills that they need in order to accomplish an activity, and on the other, on the challenges they encounter while trying to achieve this (Johnson & Wiles 2003; Sharafi, Hedman & Montgomery 2004). For example, if an activity is too difficult for the skills-ability of the student, then the usual reaction is anxiety, if on the other hand, the activity is too easy and the student can complete it quickly, then the most common reaction is boredom (Johnson & Wiles 2003). Thus, it is essential that an educational game has activities at

appropriate levels of challenge for the students. Challenge generates motivation as well as providing entertainment, and thus helps students improve their performance (Lazzaro & Keeker, 2004). For this reason, students need to feel a high sense of satisfaction with the game's activities. This can be achieved by having students: engage in and accomplish tasks with an appropriate degree of difficulty (Lazzaro & Keeker, 2004), confront challenging opponents (Vorderer, Hartmann & Klimmt, 2003), achieve specific goals (Federoff, 2002), develop skills while engaging in situations that have the right degree of danger and intensity (Vorderer, Hartmann & Klimmt, 2003). Students will feel rewarded on accomplishing these challenges and will want to play again in the future (Lazzaro & Keeker, 2004).

Another important aspect regarding difficulty is being able to navigate one's character across the environment without encountering problems and to be able to use the available objects in the world as these were intended, which prevents disruption and thus enables students to accomplish the learning objectives (Gee, 2004). In addition, players should be able to explore the game's world at their own pace (Johnson & Wiles 2003; Federoff, 2002) and in so doing have a more satisfying experience within the environment due to the fact that they can control it themselves.

Students were thus asked to assess to what degree they were satisfied with the various interactions and whether they found them difficult or (un)-interesting. This would help in that if the majority of students, for example, considered a specific task particularly difficult, then that could lead to a lack of motivation to carry out the interaction or even to continue playing the game. CMX's success as an educational game is based on its user-friendly interactions. Students are asked to what extent they found the interaction with the various game elements easy or difficult, whether the game provided appropriate feedback, and whether or not there was consistency in the choices, the situations and the actions that can be taken. Finally, students were asked whether there were elements within the game that supported collaboration.

7.2.5. Game's educational aspects

The elements that differentiate the game from its traditional counterparts are yet another important aspect of the game's evaluation. Since the game's educational content alone does not effectively lead to better learning or teaching (Fisch, 2005) there need to be other features that enhance its educational content, such as, the integration of

scaffolding or appropriate feedback, the ability for players to develop skills and gain knowledge, and the provision of resources that assist students while they play (Werger, 1999).

Students are asked for feedback on the game's efficiency to teach computer programming. It is a key factor to ascertain whether the theory in combination with the exercises given, had a positive impact on students' comprehension level. Additionally, students were asked whether they found the game both enjoyable and educational whether they would like to be taught concepts with this game, and if they found the learning process easier with this game.

7.2.6. Students' performance

The key value of CMX's efficiency is measured by its ability to increase students' performance. If a student becomes a better programmer while playing CMX, then the game has achieved its objective. When the players are totally absorbed in the environment and plot of the game, solving the given problems, then performance can be enhanced (Sweetser & Johnson 2004). It is also of great importance that players feel a sense of control in regards to manipulating their character (Desurvire, Caplna & Toth, 2004) and being able to move freely across the game's world (Sweetser & Johnson 2004). Finally, in order for players to be stimulated and maintain their interest in the game, they should be provided with multiple navigation pathways and different ways of winning (Federoff 2002; Desurvire, Caplna & Toth, 2004).

One way to determine how students' performance can be improved is to study their evaluation feedback, regarding both the entertainment and educational aspects of the game.

7.3. Pilot Study

7.3.1. Methodology of the pilot study

The pilot study was implemented in university students of the Applied Informatics Department in the University of Macedonia that had passed the introductory course in Computer Programming, thus already having a basic understanding and knowledge on programming concepts. The selection of the participants was carried out through a questionnaire, which was designed and disseminated to the students that had

already passed the required “Procedural Programming” course (80 students), in which they assessed their own knowledge by answering how well they were familiar with basic programming concepts. These 80 students were considered as an ideal sample, as they already possessed basic knowledge on the units of the required course, e.g. variables, if-statements and loops. Out of the 40 students that responded to the disseminated questionnaire, a total of 22 students were selected for participation in our case study. The 60% of the selected participants had a 5 or 6 grade in the course and a low score in the self-evaluation, the 20% had a 7 grade and a medium self- evaluation score while the remaining 20% had an above 8 grade and they scored relatively high in the self-evaluation. We chose this specific allocation of students, i.e. majority of students with low to medium performances and minority of students with high performances, in order to determine whether the game will have a positive impact on the first group and if it will maintain (or even improve) the high performances of the second group. It should be noted that 22 out of 40 students were selected because they provide a representative and manageable sample of students for drawing conclusions regarding the educational aspect of the game and the proper performance of the server and clients (e.g. errors, delays etc.). Similar methodologies have also been used for educational game evaluations (Chaffin et al., 2009).

The material and 12 activities included during students’ interactions with the 4 Senseis were as follows:

- Theory on arrays and multiple-choice questions that evaluate the theory comprehension.
- Multiple choice questions regarding the outputs of variables after an array program’s execution.
- Validation questions regarding the correct syntax of array programs.

The material and drag & drop activities included during students’ interactions with the 4 Iron Senseis regarded the:

- Entry of variable values in an array.
- Calculation of the array’s sum.
- Calculation of the maximum value of the array.

- The implementation of a function for the bubble sort algorithm in a one-dimensional array.

The 2 Gold Senseis included two problems that require students to write their own lines of codes. Both problems register variable values in an array, perform various calculations (e.g. minimum, sum and count) and print the results. It should be noted that the system includes a back-end test case that after the program's compilation automatically compares the results of the program with the expected results. A feedback of the compilation and the test case's expected results is provided to the students so that they can identify any mistakes. The study ran two consequent times on the same day to allow easier observation of small focus groups.

7.3.2. Research Questions

The first pilot usage of CMX aimed to evaluate how well the game operates, to determine any problems and weaknesses that need to be solved as well as assess the players' experience and investigate to what degree CMX helped them learn computer programming.

More specifically, the CMX evaluation framework was used in order to evaluate aspects such as game's performance, game-student interactions, CMX's entertaining and motivating elements, CMX's educational elements and students' performance. In consequence, the research questions of the pilot study are the following:

RQ1: What do students think of CMX's game performance?

RQ2: What do students think of CMX's entertaining and motivating elements?

RQ3: What do students think of game-student interactions and the game's difficulty?

RQ4: What do students think of CMX's educational aspects?

RQ5: Does adopting CMX improve students' performance?

7.3.3. Data Collection and Analysis

CMX was evaluated using a series of assessment techniques as identified by the aforementioned related works. To this end, an online questionnaire was constructed that cover the issues examined by existing studies. Apart from the first three questions which investigate students' progress in the training phase, the rest questions are designed using a 5-level Likert scale, where 1 is the most negative attitude and 5 is the most positive attitude.

The results include answers submitted by all participants in the two iterations. These results are then compared to the grades students scored in their exams for the "Procedural Programming" course, in order to determine whether their learning has been enhanced using the game. Moreover, informal discussions carried out during the study as well as the system's log files were taken into consideration when drawing conclusions.

7.3.3.1. Evaluating students' knowledge in programming and interest in games

Based on the evaluation results, it seems that most students spend a lot of time studying computer programming courses and they are already familiarized with playing computer games. This validates the effort of employing games in the educational method as an innovative environment to help students learn computer programming more efficiently.

As it was previously stated the majority of students with low to medium performances and minority of students with high performances was chosen, in order to determine whether the game will have a positive impact on the first group and if it will maintain (or even improve) the high performances of the second group. Students´ interest in computer games shows whether they are already familiar with the features and environment of existing games. Based on the questionnaire results, 36.4% of the students stated that they play games every day or 3 to 4 days a week, whereas 59.1% stated that they play 1 to 2 days a week or very rarely. In relation to the number of hours students spent on computer games, the majority or 59.1% claimed that they played for 2 to 4 hours a week. To the question on the amount of time students spent participating in computer programming courses, 59.1% claimed that it was '3 to 5 hours' or 'more than 5 hours' a week and 31.8% claimed that they spent '1 to 2 hours' a week. These results

show that the majority of students already spent a lot of time playing computer games and were therefore both familiar and comfortable with their features and functionalities. Furthermore, students spent a substantial amount of time studying computer programming, which indicates that there is a need for assisting mechanisms, such as new technologies.

7.3.3.2. Evaluating game's performance

It is important to retrieve players' feedback on their experience with the environment so as to identify any issues that need to be addressed in a future refined version. CMX's overall performance is a high requirement, since it is an MMORPG game and the problem-free usage of the game in educational settings with multiple users is crucial for successful learning (Malliarakis et al., 2013a).The questions included to evaluate the axis are shown in the following table.

Table 7-1: Evaluating game's performance

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.1.1	Response during players' movement	95.5%	4.5%
T7.1.2	Response during message transmission on chat	81.8%	18.2%
T7.1.3	Response during decision making	68.2%	31.8%
T7.1.4	Response regarding players' interactions	81.8%	18.2%

The game's response during players' movements was rated very satisfactory by 95.5% of students, while 4.5% claimed it was satisfactory. Similarly, the system's response during both message transmission and players' interactions were rated very satisfactory and satisfactory by 81.8%. Furthermore, 68.2% rated the response during decision making satisfactory, while 81.8% were satisfied by CMX's response during players' interactions. It should be noted that none of the students posted a negative rate for any of the evaluation metrics regarding the system's performance.

7.3.3.3. Evaluating CMX's entertaining and motivating elements

One of the most significant challenges for an educational game is the successful support of interesting game elements that a) are similar to the ones already encountered in commercial computer games, b) entertain the students during learning and c) motivate

them towards completing their tasks. To this end, this axis includes questions that retrieve students' feedback on how much they enjoyed aspects of the game that relate to increased interaction and engagement.

Table 7-2: Evaluating CMX's entertaining and motivating elements

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.2.1	Evaluate how much you were entertained by the plot of the game in regards to password discovery.	68.2%	31.8%
T7.2.2	Evaluate how much you were entertained by the game's graphics.	68.2%	31.8%
T7.2.3	Evaluate how much you were entertained and motivated by the game's elements (e.g. weapons, health rating, inventory, spells).	90.9%	9.1%
T7.2.4	Evaluate how much you found the dialogue with the Senseis and the multiple question and right/ wrong questions entertaining and motivating to answer correctly?	86.4%	13.6%
T7.2.5	Evaluate how much you found the code writing with drag & drop commands during evaluation by the Iron Senseis entertaining and motivating to answer correctly?	63.6%	36.4%
T7.2.6	Evaluate how much you found the ability to write code and verify it with an automatic test case entertaining and motivating to answer correctly?	86.4%	13.6%
T7.2.7	Evaluate how much the sense of victory enabled you to learn better the programming concepts	63.6%	36.4%

As shown in the above table, 68.2% of the students enjoyed the plot of the game in regards to password discovery and the graphics of the game, while 90.9% enjoyed the various elements of the game (e.g. weapons, health rating, inventory, spells etc). Moreover, the majority of the students (86.4%) found the dialogue with the Senseis and the ability to write code and verify it with an automatic test case to be entertaining and motivating, while 63.6% rated the writing of code using drag & drop commands as satisfactory. Finally, 63.4% stated that the sense of victory enabled them to learn better the programming concepts.

In summary, the activities that the students were required to execute in their interactions with each Sensei level were considered entertaining and motivating.

7.3.3.4. Evaluating game-student interactions and game's difficulty

The students were asked to evaluate the game's level of difficulty, focusing on their interactions with the different Senseis and also assess the technological aspect of the game, focusing on four important system usability heuristics features, i.e. feedback, language, design, consistency.

Table 7-3: Evaluating game-student interactions and game's difficulty

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.3.1	How easy did you find your interaction with the Senseis?	77.3%	22.7%
T7.3.1	How easy did you find your interaction with the Iron Senseis?	68.2%	31.8%
T7.3.3	How easy did you find your interaction with the Gold Senseis?	36.4%	63.6%
T7.3.4	The game provides constant and full information on what is occurring (Feedback).	31.8%	68.2%
T7.3.5	The language used in the dialogue, messages, menu etc. is simple and natural.	90.9%	9.1%
T7.3.6	The volume of the provided information does not create confusion (Aesthetic and minimalistic design).	63.6%	36.4%
T7.3.7	The same choices, situations and actions are expressed/ executed in the same way throughout the entire programming environment (Consistency).	81.8%	18.2%

The majority of the players (77.3%) found that Senseis were easy or very easy to interact with and pass as a challenge. Furthermore, 68.2% answered that the activities they encountered during their interactions with the Iron Senseis were easy or very easy. In regards to the Gold Senseis, only the 36.4% did not face difficulties in the activities that required them writing their own lines of code.

Furthermore, it seems that 68.2% required more feedback during the game regarding the player's current status within the virtual world (e.g. points and experience gained). The vast majority of the students stated that the language used was simple and natural and had no difficulty navigating through the environment's stages, menus etc.

7.3.3.5. Evaluating CMX's educational elements

The participants were asked to assess CMX in regards to its efficiency in teaching computer programming. The results retrieved, indicate that the students were very positively inclined towards all CMX's elements that enable learning programming through having fun (90.9%) and playing (86.4%).

Table 7-4: Evaluating CMX's educational elements

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.4.1	Evaluate how much you found satisfactory the ability to be educated in computer programming by practicing.	90.9%	9.1%
T7.4.2	Evaluate how much you managed to learn computer programming while playing the game.	86.4%	13.6%
T7.4.3	Evaluate how much the educational content was properly combined with game playing.	86.4%	13.6%
T7.4.4	Evaluate how much CMX should be combined with teaching computer programming courses.	63.6%	36.4%
T7.4.5	Evaluate how much CMX enables students to learn by themselves by playing.	63.6%	36.4%
T7.4.6	Evaluate how much the gameplay parts of CMX (points, chat etc) helped you in the educational process.	90.9%	9.1%
T7.4.7	Evaluate how much the learning parts of CMX (hints, theory etc.) helped you in the educational process.	72.7%	27.3%
T7.4.8	Evaluate how much you preferred using games to learn computer programming in comparison to traditional learning methods in class.	63.6%	36.4%
T7.4.9	Evaluate how much you would like for educational games to be used in more computer programming courses.	77.3%	22.7%
T7.4.10	Evaluate how much you would like for educational games such as CMX to be available online for easy access.	95.5%	4.5%

Also, students considered the incorporation of an educational game such as CMX in the learning process beneficial a) for the increase of motivation (63.6%), b) the support of individual self-learning (63.6%) and c) the assistance of professors in teaching programming elements (86.4%). The overall impression of the majority of participants is

that they are positively inclined in using educational games and CMX in particular, for computer programming learning since they have fun while also understanding the different programming concepts.

7.3.3.6. Evaluating CMX's students' performance

This axis regards the performance of students during the study, evaluated by the questions in the following table. Each study session lasted 1.5 hours, a timeframe that was considered satisfactory by 86.4% of the participants.

Table 7-5: Evaluating players' performance

No	Question	0	1	2	3	4
T7.5.1	How many Senseis did you pass?	0%	0%	13.6%	18.2%	68.2%
T7.5.2	How many Iron Senseis did you pass?	0%	22.7%	36.4%	22.7%	18.2%
T7.5.3	How many Gold Senseis did you pass?	9.1%	36.4%	54.5%	-	-

Based on the results, 68.2% of the students managed to successfully pass all Senseis while 77.3% passed at least 2 of the 4 Iron Senseis. Finally, 54.5% of the students succeeded in passing both Gold Senseis and complete the entire scenario.

Finally, we compare students' performance in the game with their grades in the final exams of the "Procedural programming" course. We examine to which extent the students that passed the course (thus scored an equal to or greater than 5 grade out of 10) managed to complete all game's tasks. To this end, we compare each grade to the number of Gold Senseis they passed, since the problems provided by them are the last, and thus more complex and demanding training challenge.

Table 7-6: Comparison matrix of course scores and Gold Senseis passed

No. of Gold Senseis	Score in midterm test of "Procedural Programming" course						
	<5	5	6	7	8	9	10
0	12.5%	0%	33.3%	0%	0%	12.5%	0%
1	50%	100%	33.3%	0%	0%	50%	0%
2	37.5%	0%	33.3%	100%	100%	37.5%	100%
Total	100%	100%	100%	100%	100%	100%	100%

As it can be seen, it is particularly encouraging that 87.5% of the students that had scored 5 in the course managed to program correctly at least one of the problems provided by the Gold Senseis, while 37.5% of these students managed to pass both Gold Senseis and thus complete the game's scenario. Moreover, it seems that all students that scored 6 in the course were able to solve at least one of the problems, while the corresponding percentage in students that scored 7 in the exams was 66.6%. Finally, the students that had scored a grade equal to or greater than 8 passed both Gold Senseis successfully.

7.4. 1st Evaluation Study

7.4.1. Methodology of the study

The CMX game was evaluated by 76 university students of the Applied Informatics Department at the University of Macedonia. The evaluation study was implemented on first year students during the ˝Procedural Programming˝ course. Students were asked to participate in the study without being offered some kind of incentive, such as a bonus grade or without making it required participation. Students enrolled in the study on the basis of their desire to try out a new technology that had the potential to enhance their knowledge in computer programming.

Five different versions of the game were designed, and each version covered a different computer programming learning unit. The first was an introductory unit and taught students the sequence structure and the basic commands of the C programming language. The second learning unit dealt with the selection structure, the third was on loops, the fourth taught students about functions, and the fifth taught students about arrays.

The educational material in each game was supplemented with ten activities that students were required to execute during their interactions with the three levels of Senseis in each game version. More specifically, students initially interacted with four Senseis, where the educational material included:

- Theory regarding each learning unit and multiple-choice questions that evaluate the theory comprehension levels.
- Multiple choice questions regarding the values of variables after a program's execution.
- Validation questions regarding the correct syntax of programs.

Students also interacted with four Iron Senseis through drag & drop activities. During these interactions, students were required to put the lines of code provided in the right sequence that together form an executable C program. Such programs can be either simple or advanced calculations. For example:

- Calculating the sum of odd numbers of a given sequence of numbers.
- Calculating the absolute value of a number.
- Converting months to days and days to hours.

Finally, students interacted with two Gold Senseis through writing lines of code. During these interactions, students were required to construct programs such as:

- Entry of variable values in a program focused on each learning unit.
- Calculation of simple functions, such as finding the maximum or the minimum value, calculating the sum of given elements, and
- Implementation of an entire program, such as the bubble sort algorithm in a one-dimensional array or search algorithms.

The next step of the evaluation process involved the accumulation and analysis of the results in order to answer the research questions presented in the following section.

7.4.2. Research Questions

The main research questions that are investigated with this study represent a different axis of the evaluation framework and are the following:

RQ1: What do students think of CMX's game performance?

RQ2: What do students think of CMX's entertaining and motivating elements?

RQ3: What do students think of game-student interactions and the game's difficulty?

RQ4: What do students think of CMX's educational aspects?

RQ5: Does adopting CMX improve students' performance?

RQ6: Do certain factors affect students' performance in playing CMX? Those factors are: a) gender, b) previous gameplay experience, c) interest in computer programming, and d) students' high ranking of the game's entertaining and motivating elements.

It is important to investigate which factors affect students' performance during gameplay. This study can be carried out through a statistical analysis in order to determine whether specific factors such as students' gender, their interest in computer programming and their previous experience with games have a statistically significant correlation with their performance during their learning with CMX. Another factor examined is whether students that evaluated the game as entertaining and motivating also had increased performance levels.

7.4.3. Data Collection and Analysis

A set of factors are examined that are directly related to the evaluation framework described in Section 7.2. An online questionnaire was created to assess the factors. The questions were further grouped into the six axes of the evaluation framework and the corresponding research questions.

The first four questions of the questionnaire refer to students' progress in the training phase, while the remainder of the questions use a 5-point Likert scale (with 1 being the most negative attitude and 5 the most positive). Additional assessment methods include the log files recorded by the system, as well as any informal discussions carried out in the classroom.

7.4.3.1. Evaluating students' knowledge in programming and interest in games

Students' interest in computer programming was assessed by a self-reflection question, where they had to evaluate their own level of knowledge regarding specific computer programming learning units. The answers provided are shown in the Table 7-7.

Table 7-7: Students' knowledge of Computer Programming in C/C++ (N=76)

Level of knowledge	Basic Syntax	If-statement	Loops	Functions	Arrays
I don't know	13.16%	17.11%	18.42%	22.37%	25.00%
I know a little	25.00%	31.58%	30.26%	30.26%	34.21%
I know enough	30.26%	22.37%	22.37%	21.05%	19.74%
Advanced	19.74%	18.42%	18.42%	17.11%	13.16%
Expert	11.84%	10.53%	10.53%	9.21%	7.89%
Total	100%	100%	100%	100%	100%

The above findings show that a very large percentage of students (70%-80%) did not consider themselves to be advanced or expert in programming concepts, while a significant percentage (40%-60%) stated that they 'know nothing or very little' about these same concepts.

Students´ interest in computer games shows whether they are already familiar with the features and environment of existing games. Based on the questionnaire results, 43.4% of the students stated that they play games every day or 3 to 4 days a week, whereas 56.4% stated that they play 1 to 2 days a week or very rarely. In relation to the number of hours students spent on computer games, the majority or 81.8% claimed that they played for 2 to 4 hours a week. To the question on the amount of time students spent participating in computer programming courses, 68.2% claimed that it was '3 to 5 hours' or 'more than 5 hours' a week. These results show that the vast majority of students already spent a lot of time playing computer games and were therefore both familiar and comfortable with their features and functionalities. Furthermore, students spent a substantial amount of time studying computer programming, which indicates that there is a need for assisting mechanisms, such as new technologies.

7.4.3.2. Evaluating game's performance

Students were asked to provide feedback regarding the game's performance during their learning experience. Such answers can help identify any issues that need to be addressed in a future version of the game to ensure proper operation of the server and the system. Four questions were designed to investigate this axis, as shown in Table 7-8. All these questions aim to test whether students encountered any delays or problems during their time within the game.

Table 7-8: Evaluating game's performance (N=76)

No	Question	Mean	Std.Dev.	1	2	3	4	5
T7.8.1	Response during players movement	4.04	0.871	1.3%	1.3%	23.7%	39.5%	34.2%
T7.8.2	Response during message transmission on chat	3.97	0.711	0%	1.3%	22.4%	53.9%	22.4%
T7.8.3	Response during decision making	4.03	0.832	1.3%	2.6%	17.1%	50.0%	28.9%
T7.8.4	Response regarding players' interactions	4.17	1.076	2.6%	6.6%	14.5%	23.7%	52.6%

Each question on the game's performance gathered well over 70% of students' responses in scales 4 and 5. More specifically, 73.7% and 76.3% respectively, rated the game's responsiveness during movements (T7.8.1) and responsiveness during message transmission (T7.8.2) as either 'satisfactory' or 'very satisfactory'. Almost four-fifths (78.9%) were satisfied with the system's responsiveness during the decision-making process (T7.8.3), while 76.3% stated that they did not 'encounter any performance issues' during their interactions with the environment's elements and characters (T7.8.4). It is worth noting that the percentage of students who posted a negative value for any of the above four evaluation metrics regarding the system's performance was less than 10%, while for the first three (T7.8.1-3) it was less than 5%. The positive student's feedback from these questions was a key element in determining CMX's high performance, which as an MMORPG educational game, is an important requirement (Malliarakis, Satratzemi & Xinogalos, 2013b).

7.4.3.3. Evaluating CMX's entertaining and motivating elements

The evaluation of an educational game's entertaining aspect is threefold as it:

- Supports features that are similar to those computer games users are already familiar with,

- Entertains students during learning with numerous characters, obstacles and challenges, and
- Motivates students to complete their tasks.

The questions included in the questionnaire that aim to evaluate this specific axis are shown in the Table 7-9.

Table 7-9: Evaluating CMX's entertaining and motivating elements (N=76)

No	Question	Mean	Std. Dev.	1	2	3	4	5
T7.9.1	Evaluate how much you were entertained by the plot of the game in regards to password discovery.	4.01	0.966	2.6%	9.2%	14.5%	52.6%	21.1%
T7.9.2	Evaluate how much you were entertained by the game's graphics.	4.22	0.741	0%	1.3%	14.5%	44.7%	39.5%
T7.9.3	Evaluate how much you were entertained and motivated by the game's elements (e.g. weapons, health rating, inventory, spells).	4.00	0.800	0%	2.6%	23.7%	44.7%	28.9%
T7.9.4	Evaluate how much you found the dialogue with the Senseis and the multiple question and right/ wrong questions entertaining and motivating to answer correctly?	3.96	0.871	1.3%	3.9%	19.7%	47.4%	27.6%
T7.9.5	Evaluate how much you found the code writing with drag & drop commands during evaluation by the Iron Senseis entertaining and motivating to answer correctly?	4.17	1.051	2.6%	5.3%	15.8%	25.0%	51.3%
T7.9.6	Evaluate how much you found the ability to write code and verify it with an automatic test case entertaining and motivating to answer correctly?	3.82	1.029	2.6%	11.8%	19.7%	40.8%	27.6%
T7.9.7	Evaluate how much the sense of victory enabled you to learn better the programming concepts.	4.28	0.759	0%	1.3%	14.5%	39.5%	44.7%

As can be seen from the findings, regarding students being entertained by the game's plot for password recovery (T7.9.1), and their enjoyment of the game's various elements (e.g. weapons, health rating, inventory, spells etc) (T7.9.3), the percentages were 73.7% and 73.6% respectively, while a large 84.2% stated that they enjoyed the game's graphics (T7.9.2).

Students were also asked to what extent they were entertained by interacting with the different levels of Senseis and whether they found the corresponding activities engaging and motivating. More specifically, 75% claimed that they were entertained and motivated by their dialogues with the Senseis (T7.9.4), and 76.3% stated that they enjoyed using drag & drop tools to correctly order pieces of a programming code (T7.9.5). A significant number (68.4%) said that they enjoyed and were motivated by being able to write code which can be automatically assessed via the test cases (T7.9.6). The vast majority (84.2%) stated that the sense of purpose (finding the hidden passwords) and of victory (destroying the server) increased their motivation to better learn the programming theory so that they could achieve their goals (T7.9.7). The overall impression of the students´ answers for this axis is positive regarding the interactions and activities of the game.

7.4.3.4. Evaluating game-student interactions and game's difficulty

Students were asked to evaluate their interactions with the Senseis, this time with focus on the difficulty aspect and not on the entertaining level. Furthermore, they were asked to evaluate the technical view of the CMX, by providing input regarding four important system usability features, such as feedback, language, design and consistency. Table 7-10 shows all the questions that have been grouped in this specific axis and the corresponding analytical results.

Table 7-10: Evaluating game-student interactions and game's difficulty (N=76)

No	Question	Mean	Std. Dev.	1	2	3	4	5
T7.10.1	How easy did you find your interaction with the Senseis?	3.93	1.011	2.6%	7.9%	22.4%	35.5%	34.2%
T7.10.2	How easy did you find your interaction with the Iron Senseis?	4.24	0.728	0%	1.3%	13.2%	46.1%	39.5%
T7.10.3	How easy did you find your interaction with the Gold Senseis?	4.01	0.931	22.6%	22.6%	18.4%	23.4%	12.9%
T7.10.4	The game provides	4.00	0.894	1.3%	3.9%	19.7%	43.4%	31.6%

	constant and full information on what is occurring (Feedback).							
T7.10.5	The language used in the dialogue, messages, menu etc. is simple and natural.	3.93	1.181	5.3%	7.9%	17.1%	27.6%	42.1%
T7.10.6	The volume of the provided information does not create confusion (Aesthetic and minimalistic design).	3.80	1.033	2.6%	11.8%	21.1%	39.5%	27.6%
T7.10.7	The same choices, situations and actions are expressed/ executed in the same way throughout the entire programming environment (Consistency).	4.09	0.996	2.6%	5.3%	13.2%	38.2%	40.8%
T7.10.8	How easy could you understand the game's scenario and the tasks you had to do to win the game?	4.26	0.957	1.3%	7.9%	3.9%	36.8%	50.0%
T7.10.9	Could you easily understand how to move your character within the game and how to interact with the other players?	3.96	1.012	3.9%	5.3%	13.2%	46.1%	31.6%
T7.10.10	The mini map included in the game helped me to collaborate with my teammates.	4.24	0.864	1.3%	1.3%	14.5%	38.2%	44.7%
T7.10.11	Features such as the chat tool and the minimap helped me to collaborate more with my teammates.	4.12	0.864	0%	5.3%	15.8%	40.8%	38.2%

As can be seen in Table 7-10, a high percentage of students (69.7%) found that the Senseis were ‘easy’ or ‘very easy’ to interact with and to accomplish the challenge (T7.10.1). This positive response increases significantly to 85.6 in relation to the Iron Senseis (T7.10.2). In contrast, only 36.3% of the participants stated that they found their interactions with the Gold Senseis ‘easy’ or ‘very easy’ (T7.10.3).

From the findings, it can be seen that 75% of students were satisfied with the feedback provided during the game, such as information on each player’s current location in the environment or their status (e.g. experience, points, weapons etc.) (T7.10.4). Also, a large number of students (69.7%) agreed that the language used was fairly simple and natural and they did not have any difficulty in navigating across the different stages and menus of the game (T7.10.5). It is worth noting that a significant 67.1% found that the volume of information provided does not create confusion (T7.10.6), while the majority (79%) stated that the choices, situations and actions were expressed/ executed in the same way across the entire programming environment (T7.10.7). Finally, 86.8% found that they could easily understand the scenario and the tasks (T7.10.8), 87.8% could easily understand how to move and interact with other players (T7.10.9), while 82.9% and 79% respectively stated that the mini map feature (T7.10.10) and the chat tool (T7.10.11) helped them to collaborate with other players.

7.4.3.5. Evaluating CMX’s educational elements

Another important axis that was evaluated concerns the educational aspects of CMX. The participants were asked thirteen questions to assess CMX in regards to its efficiency in teaching computer programming.

Table 7-11: Evaluating CMX’s educational elements (N=76)

No	Question	Mean	Std. Dev.	1	2	3	4	5
T7.11.1	Evaluate how much you found satisfactory the ability to be educated in computer programming by practicing.	4.07	0.957	2.6%	3.9%	14.5%	42.1%	36.8%
T7.11.2	Evaluate how much you	4.05	1.031	2.6%	5.3%	18.4%	31.6%	42.1%

	managed to learn computer programming while playing the game.							
T7.11.3	Evaluate how much the educational content was properly combined with game playing.	4.17	0.915	2.6%	2.6%	10.5%	43.4%	40.8%
T7.11.4	Evaluate how much CMX should be combined with teaching computer programming courses.	4.21	0.805	0%	1.3%	19.7%	35.5%	43.4%
T7.11.5	Evaluate how much CMX enables students to learn by themselves by playing.	4.00	0.800	0%	2.6%	23.7%	44.7%	28.9%
T7.11.6	Evaluate how much the gameplay parts of CMX (points, chat etc) helped you in the educational process.	4.32	0.820	0%	2.6%	14.5%	31.6%	51.3%
T7.11.7	Evaluate how much the learning parts of CMX (hints, theory etc.) helped you in the educational process.	4.08	0.876	1.3%	3.9%	14.5%	46.1%	34.2%
T7.11.8	Evaluate how much you preferred using games to learn computer programming in comparison to traditional learning methods in class.	4.17	0.929	2.6%	2.6%	11.8%	40.8%	42.1%
T7.11.9	Evaluate how much you would like for educational games to be used in more computer programming courses.	4.16	0.767	0%	1.3%	18.4%	43.4%	36.8%
T7.11.10	Evaluate how	4.07	0.822	0%	2.6%	22.4%	40.8%	34.2%

	much you would like for educational games such as CMX to be available online for easy access.							
T7.11.11	Evaluate how much, after the usage of CMX, it will be easier for you to write programs in the future.	4.09	1.085	5.3%	5.3%	6.6%	40.8%	42.1%
T7.11.12	Evaluate how much the process of learning computer programming with CMX will be easy and clear.	4.25	0.954	2.6%	1.3%	15.8%	28.9%	51.3%
T7.11.13	Evaluate how much, while you were playing the game, the activities helped you to use your already gained knowledge in order to successfully complete your goals.	4.13	0.914	1.3%	5.3%	11.8%	42.1%	39.5%

The results demonstrate that the majority of students were very positively inclined towards all CMX elements which allowed them to learn programming by practicing (T7.11.1) and playing (T7.11.2) with 78.9% and 73.7%, respectively. A high 84.2% of students had a positive attitude towards educational content and game playing (T7.11.3), while 78.9% stated that CMX should be incorporated in the computer programming courses (T7.11.4), and if this is done then 73.6% believed that it could support self-learning (T7.11.5). In addition, the majority of students claimed that their educational performance could be enhanced with CMX gameplay components, such as chat tool, etc. (T7.11.6), and CMX learning parts/components, such as theory, hints, explanatory messages, etc. (T7.11.8) with 82.9% and 80.3%, respectively.

It is worth noting that 82.9% of students stated that they preferred learning computer programming through the use of games rather than the traditional learning methods applied in the classroom (T7.11.8), and 80.2% would like educational games to

be included in future computer programming courses (T7.11.9). A significant 79.2% found the programming process with CMX easy and clear (T7.11.12), and a high 82.9% believed that after playing CMX, they would find it easier to write programs (T7.11.11), while another 81.6% stated that the activities on CMX helped them to use the knowledge already gained to accomplish the required goals (T7.11.13). Finally, three quarters (75%) of the students expressed their desire for CMX and other similar games to be available online for easy access (T7.11.10).

7.4.3.6. Evaluating CMX's student's performance

The last axis to be assessed regards the actual performance of the students while playing the game. An advanced performance is considered to require a student passing all or the vast majority of the Senseis and thus retrieving all the hidden passwords.

Table 7-12: Evaluating students' performance (N=76)

No	Question	0	1	2	3	4
T7.12.1	How many Senseis did you pass?	0%	0%	7%	30%	63%
T7.12.2	How many Iron Senseis did you pass?	0%	19%	16%	18%	47%
T7.12.3	How many Gold Senseis did you pass?	14%	28%	58%	-	-

Table 7-12 shows that 63% of students successfully completed the first Senseis level (T7.12.1) while 81% passed at least 2 of the 4 Iron Senseis challenges (T7.12.2). Over half of the students (58%) succeeded in passing both of the Gold Senseis, managing thus to complete the entire scenario of the game.

Although a significant number of students appear to have made substantial progress in the game, with over half finishing all tasks, in order to be able to assess whether or not CMX enhanced their programming performance a comparison is made with the results in their mid-term exam in the "Procedural Programming" course, i.e., prior to being introduced to CMX. Table 7-13 below presents the percentage of students and their scores (out of 10) in the midterm exam (those that got less than 5 are in one group), and the percentage of Gold Senseis that were passed. Only the Gold Senseis results are used in the comparison because these tasks were the most complex and thus

the most demanding. Accomplishing these shows the highest performance students can achieve.

Table 7-13: Comparison table of midterm exam scores and Gold Senseis passed (N=76)

No. of Gold Senseis	Score in midterm test of "Procedural Programming" course						
	<5	**5**	**6**	**7**	**8**	**9**	**10**
0	40.6%	33.6%	14.8%	12.3%	0%	0%	0%
1	34.4%	38.5%	51.1%	53.3%	20%	0%	0%
2	25.0%	27.9%	34.1%	35.3%	80%	100%	100%
Total	100%	100%	100%	100%	100%	100%	100%

It can be seen from the table that over half (59.4%) of the students who had not passed the mid-term exam were able to complete one (34.4%) or both (25.0%) of the problems in the Gold Senseis and write their own code using the C programming language. For students who got a grade 5 in the exam, meaning that they just managed to pass, 38.5% managed to accomplish one task and 27.9% accomplished both tasks and correctly wrote two programs. Together this reached a significant total of 66.4%. Furthermore, a total of 85.2% of the students that got a grade 6, and 88.6% that got a grade 7 managed to write one or both programs correctly, while, 80% of those who got an 8 in the exam, successfully wrote both programs. As was perhaps to be expected those who scored 9 and 10 had 100% success rate.

7.4.3.7. Evaluating the factors that influence the student's performance

The four specific factors that were examined as to whether they influenced students' high performance during the game are: a) gender, b) previous gameplay experience, c) interest in computer programming, and d) students' high ranking of the game's entertaining and motivating elements.

High performance is defined as the students who correctly wrote both Gold Senseis programs and found the final code that shuts down the server, as well as those who managed to complete one of the Gold Senseis programs. A comparison was made using a t-test and the results are shown in Table 7-14.

Table 7-14: Comparison table of factors and Gold Senseis passed (N=76)

Factor	Group	Number	Mean	Standard Deviation	t-value	Significance at p < .05
Gender	Female	28	1.57	0.634	0.622	0.536
	Male	48	1.48	0.618	0.617	0.539
Frequency of gameplay	Frequent	33	1.48	0.667	0.346	0.731
	Not Frequent	43	1.53	0.592	0.340	0.735
Interest of computer programming	Interest	35	1.60	0.604	1.127	0.263
	No Interest	41	1.44	0.634	1.132	0.261

7.4.3.8. Gender

This factor investigates whether the gender of the participants is related to their performance during the study. More specifically, the study examined if males (63%) and females (37%) performed differently when playing CMX. Based on the t-test performed, there is no statistically significant difference indicated between the scores of the females versus the males ($p < 0.05$). Therefore, gender does not seem to be an affecting factor for students' performance. This was a desired result, since any difference in efficiency based on the students' gender is not wanted. It should be noted that in studies performed in MMORPGs, men are significantly more likely to play games because of the potential for achievement and reward, whereas women are more likely to play games because they want to interact with other players (Yee, 2006). Other differences based on gender depicted in relevant studies include the time spent playing, where men play for longer periods of time (Ogletree & Drake, 2007) and that men are more confident in their skills in games than women (Terlecki et al., 2011).

7.4.3.9. Frequency of gameplay

This factor investigates whether the amount of experience playing computer games affects a student to achieve the goals of CMX. To achieve this research, the evaluation questionnaire includes questions regarding the number of weekly hours spent for playing computer games (e.g. arcade, other MMORPGs etc.), as aforementioned. Thus, the results were formed into groups with students that play games frequently and students that do not play computer games frequently and performed a t-test in relation

with their performance in CMX.The results indicated no statistically significant difference between the scores of the students that are already familiar with computer games versus the performances of students that do not usually play games ($p < 0.05$). Therefore, it is concluded that the amount of prior computer gameplay experience does not appear to be a factor that affects a student's ability to successfully complete all the tasks provided in CMX.

A few studies have investigated the possible facilitating role of previous game experience on students' performance in the classroom and their goals' achievement in similar games. More specifically, two studies that researched this specific variable (Kebritchi, Hirumi & Bai, 2010; Miller & Robertson, 2011) also found no statistically significant difference in the outcomes that was relevant to prior levels of computer games skills. This may be considered a surprising result, since prior experience in gameplay was expected to prove beneficial.

7.4.3.10. Interest of computer programming

Another factor examined is students' interest in computer programming outside of the classroom. To this end, questions were included regarding whether they perform any activities related to computer programming because they enjoy them and not as course requirements. The answers provided were then examined, grouped into two categories, i.e. students that are generally interested in computer programming and students that are not, and a t-test was performed to determine whether this factor has any correlation to students' performance. Based on the statistical results, this factor also showed no statistically significant difference between the scores of the students and their interest in programming ($p < 0.05$). It should be noted that students' interest in computer programming is a unique indicator, and hence no relevant previous work carried out was identified that examines this parameter, mostly since such studies are carried out in Computer Science students where they are obligated to engage in computer programming. However, this obligatory engagement is distinguished from a personal, extracurricular interest in computer programming, in order to assess whether such an interest affects their performance when playing.

7.4.3.11. Entertaining and motivating elements

The last factor investigated is if students' performance was affected by their positive evaluation of the game's entertaining and motivating elements. The statistical analysis carried out indicates a significant correlation between students that performed well in the game and that also stated they enjoyed the game's plot in regards to the activities and password discovery ($r=0.726$, $p<0.05$). Moreover, significant correlation was identified between the students that enjoyed the game's graphics and their performance ($r=0.211$, $p<0.05$). Finally, significant correlation was also detected between the students that showed positive opinions regarding the dialogues with the Senseis and the multiple choice questions and their performance ($r=0.231$, $p<0.05$) as well as between the students that enjoyed ordering lines of code with drag & drop commands during their training by the Iron Senseis and their performance ($r=0.650$, $p<0.05$). In conclusion, it is estimated that the majority of the metrics that determine how motivational and entertaining the game was were strongly related to the students' performance during learning and playing. Therefore, based on the results, the more a student enjoys playing a game, the more likely it is that he/she will perform highly.

This relation is particularly positive, since literature indicates that game features that increase students' motivation will also lead to improved training and more substantial learning content recollection (Ricci, Salas & Cannon-Bowers, 1996). Finally, (Garris, Ahlers & Driskell, 2002) indicate that efficient entertaining and motivation elements in a game are crucial for the maximization of learning outcomes and the achievement of all educational goals set in an educational game.

7.5. 2nd Evaluation Study

7.5.1. Methodology of the study

The CMX game was evaluated by 18 postgraduate students of the Applied Informatics department at the University of Macedonia. The evaluation study was implemented on second semester students during the "Serious Games Programming" course. The game was presented in the last lesson of the semester, where it was used by the students and evaluated after the study was complete.

The participants were divided in two groups of nine people and they played CMX in order to learn the specific learning unit of functions. The game was exactly the same as the environment structured in the previous evaluation study, and the participants were asked to evaluate it as knowledgeable on basic computer programming concepts, since they have been taught relevant courses as well as the "Serious Games Programming" course. Thus, these students possess a fundamental experience in previous serious games and in computer programming concepts.

The educational material in each game was supplemented with ten activities that students were required to execute during their interactions with the three levels of Senseis. The next step of the evaluation process involved the accumulation and analysis of the results in order to answer the research questions presented in the following section.

7.5.2. Research Questions

The main objective of this evaluation is the assessment of CMX by postgraduate students that have a basic knowledge of computer programming and have been taught the "Serious Games Programming" course. The questions of the questionnaire were adapted accordingly so as to determine how these more expert students believe that CMX would be perceived by students that are novices in computer programming.

7.5.3. Data Collection and Analysis

CMX was evaluated using the CMX evaluation framework. More specifically, an online questionnaire was constructed that included questions that correspond to all axes of the evaluation framework. Students were asked to respond to the questionnaire, where the first three questions gather information on the students' progress in the training phase, while the rest questions investigate in detail each framework axis and they are designed using a 5-level Likert scale with 1 being the most negative attitude and 5 the most positive.

7.5.3.1. Evaluating students' knowledge in programming and interest in games

This question, which regards students' knowledge and their interest in computer programming, was not included in the specific evaluation process, because the sample consists of postgraduate students, and thus their existing knowledge and interest in the field is considered as granted. However, the study evaluated how much students were interested in and enjoy playing computer games in general. The results showed that 22.2% of the participants play every day, while 66.7% play 1-2 times a week. Regarding the specific times that students dedicate in game playing, 72.2% claimed that they spend 3-5 hours or more than 5 hours playing. These results indicate that students are generally familiarized with computer games.

7.5.3.2. Evaluating game's performance

This question aimed to determine how students evaluated the game's performance. The following table shows the positive results received.

Table 7-15: Evaluating game's performance

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.15.1	Response during players' movement	88.9%	11.1%
T7.15.2	Response during message transmission on chat	83.3%	16.7%
T7.15.3	Response during decision making	77.8%	22.2%
T7.15.4	Response regarding players' interactions	83.3%	16.7%

More specifically, 88.9% of the participants claimed that the game's response was satisfying during players' navigations across the environment. Also, 88.9% stated that the response of the chat tool during messages transition was positive, while 83.3% claimed they could perform interactions with a satisfying response from the game. Finally, 77.8% stated that they experienced no problem regarding the game's response while they were reading the theory provided or writing a program and making decisions on how to proceed in the game's activities.

7.5.3.3. Evaluating CMX's entertaining and motivating elements

Students demonstrated a positive attitude towards CMX's entertaining and motivating elements. Table 7-16 presents the specific results.

Table 7-16: Evaluating CMX's entertaining and motivating elements

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.16.1	Evaluate how much you were entertained by the plot of the game in regards to password discovery.	83.3%	31.8%
T7.16.2	Evaluate how much you were entertained by the game's graphics.	72.2%	27.8%
T7.16.3	Evaluate how much you were entertained and motivated by the game's elements (e.g. weapons, health rating, inventory, spells).	94.4%	5.6%
T7.16.4	Evaluate how much a novice programmer would find the dialogue with the Senseis, the multiple choice questions and right/wrong questions entertaining and motivating to answer correctly.	83.3%	16.7%
T7.16.5	Evaluate how much a novice programmer would find the code writing with drag & drop commands during training with the Iron Senseis entertaining and motivating to answer correctly.	72.2%	27.8%
T7.16.6	Evaluate how much a novice programmer would find the ability to write code and verify it with an automatic test case entertaining and motivating to answer correctly.	83.3%	16.7%
T7.16.7	Evaluate how much the sense of victory would enable a novice programmer to learn the programming concepts better.	88.9%	11.1%

The majority of the students found the plot of the game in regards to the password discovery entertaining (83.3%). Furthermore, 72.2% enjoyed the game's graphics, while 94.4% found the game's elements entertaining and motivating. Students also assessed the levels of entertainment and motivation for a novice programmer. More specifically, 83.3% estimated that novice programmers would be entertained and motivated by their dialogue with the Senseis and the different tasks of that phase (e.g. multiple choice questions, right/wrong questions etc.). Correspondingly, 72.2% considered that novices would be entertained by the Iron Senseis, while 83.3% believe that the activities included

in the Gold Senseis phase, such as writing lines of code, would be motivating and entertaining to novice programmers.

7.5.3.4. Evaluating game-student interactions and game's difficulty

Students were requested to evaluate the game-students interaction elements of the game. The majority of the results appeared positive. Specific detailed are provided below.

Table 7-17: Evaluating game-student interactions

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.17.1	Evaluate how easy a novice programmer would find the interaction with the Senseis?	77.8%	22.2%
T7.17.2	Evaluate how easy a novice programmer would find the interaction with the Iron Senseis?	66.7%	33.3%
T7.17.3	Evaluate how easy a novice programmer would find the interaction with the Gold Senseis?	38.9%	61.1%
T7.17.4	The game provides constant and full information on what is occurring (Feedback).	72.2%	27.8%
T7.17.5	The language used in the dialogue, messages, menu etc. is simple and natural.	88.9%	11.1%
T7.17.6	The volume of the provided information does not create confusion (Aesthetic and minimalistic design).	72.2%	27.8%
T7.17.7	The same choices, situations and actions are expressed/ executed in the same way throughout the entire programming environment (Consistency).	81.8%	18.2%
T7.17.8	How easy could you understand the game's scenario and the tasks you had to do to win the game?	66.7%	33.3%
T7.17.9	Could you easily understand how to move your character within the game and how to interact with the other players?	72.2%	27.8%
T7.17.10	The mini map included in the game helped me to collaborate with my teammates.	77.8%	22.2%
T7.17.11	Features such as the chat tool and the minimap helped me to collaborate more with my teammates.	81.8%	18.2%

The strong majority of the players (77.8%) found that Senseis were easy or very easy to interact with a novice programmer and pass as a challenge. Furthermore, 66.7% answered that the activities that novice programmers would encounter during their interactions with the Iron Senseis were easy or very easy. In regards to the Gold Senseis, only the 38.9% considered that novice programmers would not face difficulties in the activities that required them writing their own lines of code.

Furthermore, it seems that 72.2% required more feedback during the game regarding the player's current status within the virtual world (e.g. points and experience gained). The majority of the students stated that the language used was simple and natural (88.9%) and had no difficulty navigating through the environment's stages, menus etc. (72.2%). Finally, 81.8% considered that there is consistency inside the game in the situations and actions executed inside the environment.

7.5.3.5. Evaluating CMX's educational elements

Students were also requested to evaluate CMX's educational elements.

Table 7-18: Evaluating CMX's educational elements

No	Question	4-5 of Likert Scale	1-3 of Likert Scale
T7.18.1	Evaluate how satisfactory a novice programmer would find the ability to be educated in computer programming by practicing.	88.9%	11.1%
T7.18.2	Evaluate to what degree a novice programmer would manage to learn computer programming while playing the game.	77.8%	22.2%
T7.18.3	Evaluate to what degree the educational content was properly combined with game playing.	83.3%	16.7%
T7.18.4	Evaluate to what degree CMX should be combined with teaching computer programming courses.	88.9%	11.1%
T7.18.5	Evaluate how much CMX enables students to learn by themselves by playing.	77.8%	22.2%
T7.18.6	Evaluate to what degree the gameplay parts of CMX (points, chat etc) would help novice programmers in the educational process.	88.9%	11.1%
T7.18.7	Evaluate to what degree the learning parts of CMX (hints, theory etc.) would help novice programmers in the educational process.	77.8%	22.2%
T7.18.8	Evaluate to what degree novice programmers	72.2%	27.8%

	would prefer using games to learn computer programming in comparison to traditional learning methods in class.		
T7.18.9	Evaluate to what degree novice programmers would like educational games to be used in more computer programming courses.	83.3%	16.7%
T7.18.10	Evaluate to what degree novice programmers would like educational games such as CMX to be available online for easy access.	77.8%	22.2%
T7.18.11	Evaluate how much easier it will be for novice programmers to write programs in the future after using CMX.	83.3%	16.7%
T7.18.12	Evaluate how much easier and clearer the process of learning computer programming will be with CMX for novice programmers.	72.2%	27.8%
T7.18.13	Evaluate to what degree the activities would help novice programmers to use already gained knowledge in order to successfully complete their goals while they were playing the game.	77.8%	22.2%

According to the above table, 88.9% of the participants estimate that a novice programmer would find the ability to be educated in computer programming by practicing satisfactory, while 77.8% considered that a novice programmer would manage to learn computer programming while playing the game. Additionally, 77.8% estimated that the learning parts of CMX would help novice programmers in the educational process and 83.3% assessed that it will be easier for novice programmers to write programs in the future after using CMX. This is also shown in students' estimation that the process of learning computer programming will be easier and clearer for novice programmers if they use CMX (77.8%). The same percentage also claimed that the activities of CMX would help novice programmers to use already gained knowledge in order to successfully complete their goals, while they were playing the game.

Additionally, 83.3% claim that the educational content was properly combined with game playing, while 88.9% stated that CMX should be incorporated in computer programming courses.

7.5.3.6. Evaluating students' performance

The first three questions regarded students' performance during their interactions with the three different levels of Senseis in the game's training phase. Based on the

results, the majority of the students managed to succeed the game's objectives, as shown in the following table.

Table 7-19: Evaluating players' performance

No	Question	0	1	2	3	4
T7.19.1	How many Senseis did you pass?	0%	0%	11.1%	16.7%	83.3%
T7.19.2	How many Iron Senseis did you pass?	0%	11.1%	27.8%	33.3%	27.8%
T7.19.3	How many Gold Senseis did you pass?	5.6%	33.3%	61.1%	-	-

According to the results, 83.3% of the students managed to pass all Senseis, while 88.9% passed at least 2 Iron Senseis and 94.6% passed 1 or 2 Gold Senseis.

7.6. Conclusions

Chapter 7 elaborates on the evaluation of the game with one pilot and two evaluation studies in university students of the Applied Informatics department in the University of Macedonia, which were asked to play the game and practice on learning how to program using the C programming language. More specifically, in the first pilot and the first evaluation study participated undergraduate students to check the game as a learning tool, while the second evaluation study consisted of 18 participants, which where postgraduate students and aimed to evaluate CMX by students that already had a significant knowledge of computer programming and have been taught the "Serious Game Programming" course. This provided feedback on how knowledgeable students estimate that novice programmers would use and assess a game such as CMX.

Once they joined the game, students were trained by specialized game characters named Senseis, where they learned the theory, and they performed a series of different activities in order to gather passwords that enabled them to proceed in the game.

The evaluation of all three studies were carried out based on an evaluation framework that includes six axes, such as students' knowledge in programming and interest in games, game's performance, CMX's entertaining and motivating elements, game-student interactions and game's difficulty, CMX's educational elements and student's performance. This framework was used to set the research questions of our

study and to design an online questionnaire that aims to assess all the aforementioned axes. Furthermore, informal discussions carried out in the classroom and the log files recorded by the system were also taken into consideration while answering the research questions of each study.

The evaluation results of the pilot and the first evaluation study showed that the majority of the students increased their knowledge on computer programming by using the game. More specifically, students that scored a grade of 5 to 7 in a midterm exam of a computer programming course managed to correctly answer multiple choice questions regarding the theory taught as well as correctly place multiple tiles of code in the correct order to make up an executable C code and write their own programs. Additionally, as it was anticipated students with scores equal to or higher than 8 completed all the game's tasks in both the pilot and the first evaluation study.

All the evaluation studies also present the examination of the students' intention to use an educational game such as CMX in the future for computer programming education. The evaluation results concluded that the participants enjoyed the game's entertaining elements and were motivated to progress within the scenario's steps and did not face major difficulties during the learning process. This positive experience is also confirmed by the fact that the majority responded that they would indeed re-use CMX when learning programming in the future.

The following figure provides a holistic overview of the answers for the questions T7.1.1 – T7.4.10 which are designed using the 5-level Likert Scale in the pilot study.

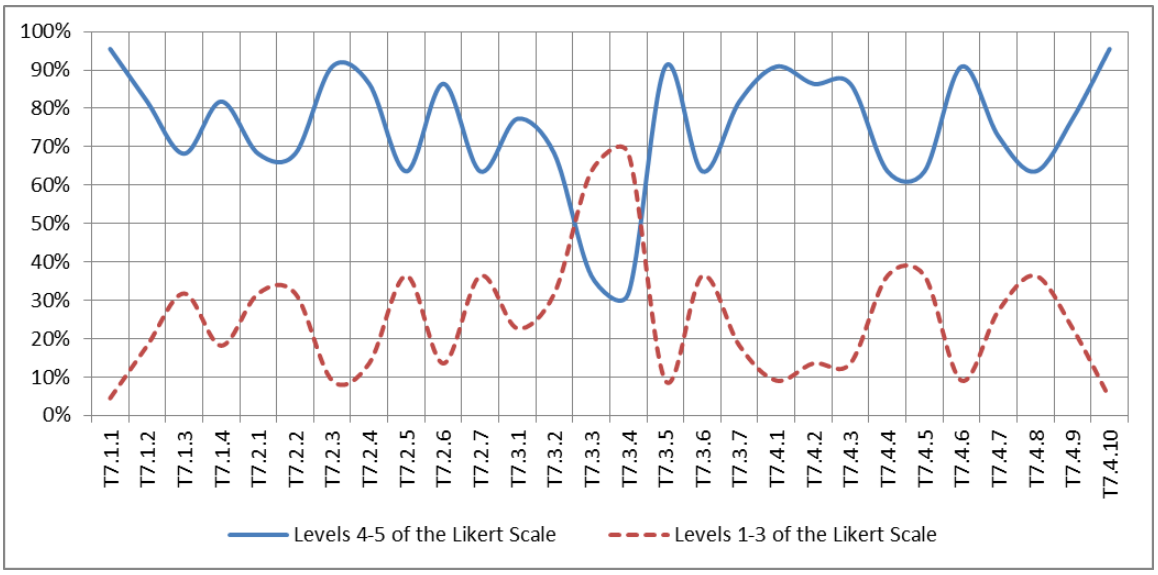

Figure 7-1: Overall students' attitudes towards CMX in the pilot study

The blue solid line shows the percentage of students that responded in each question rating either a 4 or 5 score and thus showing a positive attitude towards the question and the aspect evaluated (e.g. performance, game performance, entertainment and education). As it can be seen in the graph, the majority of the questions depict that students showed positive feedback after their experience with the game. In two of the questions students expressed concerns due to the fact that the Gold Senseis were difficult to pass (T.7.3) and the provision of feedback within the game was not as analytical as they would have preferred (T.7.4).

The Figure 7-3 presents an overall overview of the answers given by the students for the questions T7.8.1 to T7.11.13 as presented in the Tables of the first evaluation study. These questions were designed using a 5-level Likert sale, where 5 represents the most positive attitude and 1 the most negative. Thus, the graph can provide insights on the general attitude of the students in regards to the game and its elements.

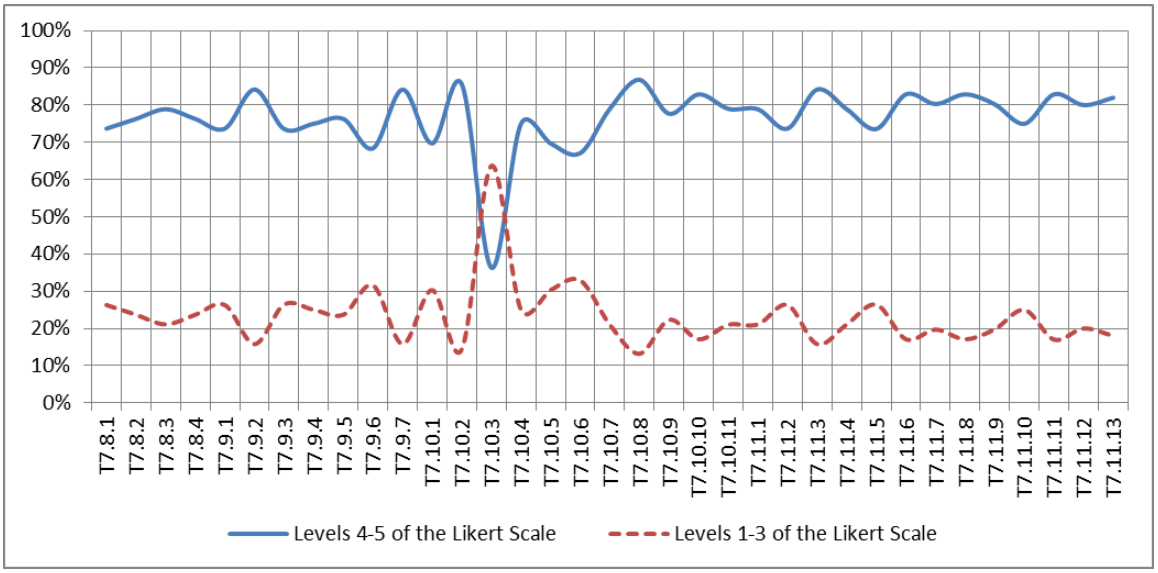

Figure 7-2: Overall students' attitudes towards CMX in the first evaluation study

As it can be seen above, the graph validates all the analytical results provided for each axis in the corresponding section. It should be noted that the only question that the red line is above the blue regards the students' statement that the activities included in the last level of training were difficult to pass (T7.10.3). This is also validated in our evaluation, and should be expected, since Golden Senseis represent the highest level of difficulty within the game, where students need to analyze the two problems provided into code commands and write two executable programs. Therefore, these activities require a very in-depth comprehension of the taught material and is reasonably challenging.

Also, there are two other questions (T7.9.6 and T7.10.6) that have a slightly lower mean than the rest, although much higher than 3.5, which corresponding elements will be improved in the future. More specifically,68.4% rated positively the ability to write code and verify it through the automatic test case (Mean 3.82, S.D. 1.029), while 67.1% stated that the volume of the provided information did not confuse them (Mean 3.80, S.D. 1.033). Thus, the programming editor will be improved, the information provided will be enhanced so that they are not confusing for the vast majority and the game's ability to support teachers during gameplay will be further highlighted.

The overall impression from the implementation and the evaluation of the first evaluation study is that the majority of the students were satisfied by this motivating, entertaining and engaging technology and is positively inclined to use such educational games and CMX in particular in order to learn computer programming.

Finally, a series of t-tests were performed in the first evaluation study in order to assess whether factors such as gender, previous experience in playing computer games and students' general interest in computer programming affected participants' performance. The statistical tests performed did not indicate such correlations. The study further evaluated whether students that evaluated the game's entertaining and motivating elements with high scores also performed well in the game. The statistical analysis carried out showed that there was indeed a strong correlation, indicating that students that enjoyed playing while learning also performed better. This result highlights the necessity for improving the entertaining and motivating features of the game, since it seems that they can lead to increased students' performances.

In the second evaluation study, the provided feedback regarded information on how knowledgeable students estimate that novice programmers would use and assess a game such as CMX. The participants played the game in a version that included the learning unit of functions, as it was designed for the purposes of the previous evaluation study. The students' answers were particularly encouraging, validating the results of the previous two evaluations carried out. The following figure provides a holistic overview of the answers for the questions T.15.1 – T.18.13 which are designed using the 5-level Likert Scale.

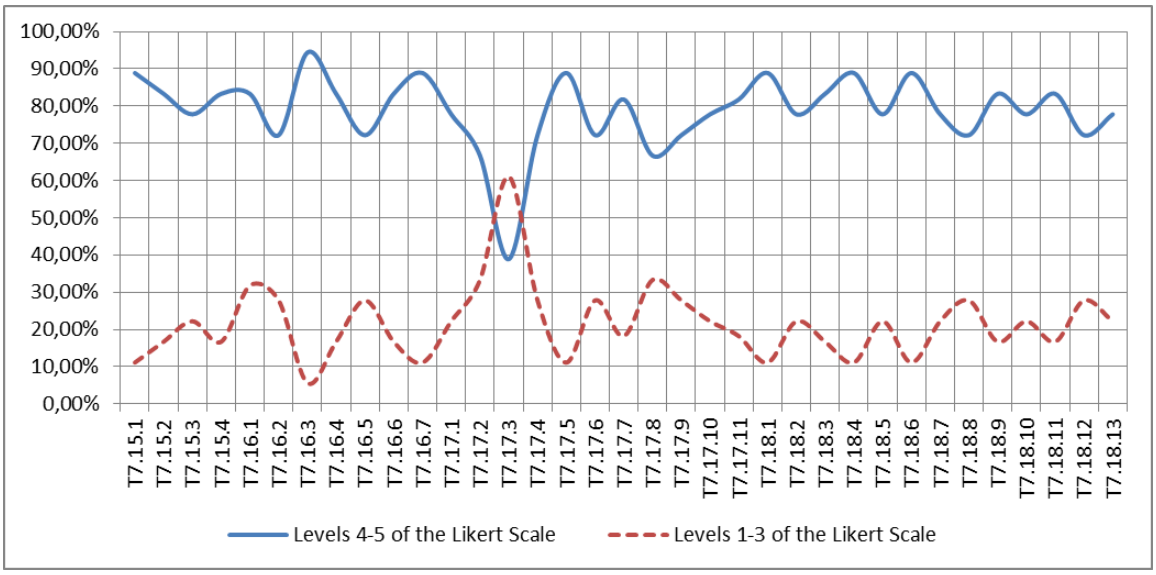

Figure 7-3: Overall students' attitudes towards CMX in the second evaluation study

As it can be seen in the graph, the majority of the questions depict that students showed positive feedback after their experience with the game. The only low point in the graph represents the difficulty that a part of students faced during their interactions with the Gold Senseis and the corresponding activities (e.g. writing their own programs).

8. Conclusions and future work

The previous chapters presented a game-based learning approach that aims to address difficulties encountered in learning and teaching computer programming.

Initially, an overview of the field was carried out through the study of educational games developed to support education in general and computer programming specifically. This study led to the documentation of an extended list with all advantages and limitations identified.

The research proceeded to investigate works that have proposed design frameworks for educational games as well as works that have developed educational games for computer programming. This led to the recording of all characteristics that should be incorporated in a successful educational game in order to fulfil educational goals set by the teachers and students, while being engaging and entertaining in order to enable motivation and active participation during the learning process.

After the accumulation and analysis of all these characteristics, a design framework was constructed, aiming to be applied for the development of an educational MMORPG named CMX, which would support teaching and learning of computer programming. CMX was developed using the .NET framework, the C# programming language, the MySQL database language and the OpenGL graphics library.

The main features that distinguish CMX from the rest of the existing educational games are as follows:

1. The game does not include solely static characteristics. Its graphical environment as well as all educational elements **can be customized.** This allows the game to support potentially infinite scenarios and to be adaptable so as to support the specific needs of any programming course's students.
2. Educators have an active role during the **re-design** of the game as well as in applying and **monitoring the learning analytics** of the game. This way, teachers can view if the game and its application in class is helping students in fulfilling the educational goals set for any unit of learning.
3. CMX includes numerous **evaluation techniques**. Students are assessed through multiple-choice questions, by dragging and dropping tiles of lines of

code in the correct order so as to form an executable program, and by writing and structuring a program with the use of an editor and a compiler.

4. CMX is an MMORPG, and thus includes features such as a chat tool and a minimap tool that allow **students' collaboration** to achieve the game's goals. For example, the chat tool helps the communication throughout the game between all the players, but also between only the players of a specific team. Also, the minimap tool provides information on the location of each player, allowing students to track each other and plan their strategies.
5. It is a **holistic game**. Although the game is divided into levels and sublevels, it is designed so that it can be used for all programming teaching sections.
6. The players have a variety of ways they can **interact with code programming** within the game, according to the level of their programming knowledge. They can either answer simple multiple choice questions, or drag and drop tiles to complete the program correctly, or write a program on the editor.
7. The game includes a lot of **tutorials with hints and tips** that can help the students achieve the goals and simultaneously teach them the programming aspects they need to know.
8. The game's logic "The best gamer is the best programmer" underpins their **motivation to play more actively** and, by association, to learn how to program more effectively.
9. The fact that **this is a MMORPG** removes students' usual initial reluctance to try new and foreign to them educational software. The logic of this game is similar to games the majority of students are already familiar with (e.g. War of Warcraft etc.), so it is easier for them to **find this new technology enjoyable**.
10. The nature and environment of the game **reduces the effort it takes for teachers to train students** in a novel, complex system, and the time it takes for students to learn all the specificities of such a technology (e.g. menus, console, action buttons, interface layers etc.).
11. CMX is a client-server system. To this end, it works using specialized **optimization algorithms that have been implemented and embedded**. These algorithms allow the participation of many users at the game time (real

and virtual users), with a delay <1ms in lan conditions. Thus, any delay that may occur is not perceived by the players.

CMX was utilized in three pilot studies in students of the Applied Informatics Department in the University of Macedonia, and the process was assessed each time by the players. The evaluation was carried out based on an evaluation framework that was constructed in the context of this thesis. The framework includes 6 main concepts that should be assessed in order to determine an MMORPG's success in supporting computer programming education. Six research questions were also drafted reflecting these 6 evaluation axes, as follows:

- *RQ1: What do students think of CMX's game performance?*
- *RQ2: What do students think of CMX's entertaining and motivating elements?*
- *RQ3: What do students think of game-student interactions and game's difficulty?*
- *RQ4: What do students think of CMX's educational elements?*
- *RQ5: Does adopting CMX improve students' performance?*
- *RQ6: Do factors such as students' high ranking of the game's entertaining and motivating elements, interest in computer programming, previous gameplay experience and gender affect students' performance while playing CMX?*

The online questionnaire that was constructed was disseminated to the students after the end of each evaluation study. The results were gathered and analysed for examination. All results that were gathered indicated a positive attitude of students of computer programming courses to learn such concepts using an MMORPG such as CMX. The majority of the students seemed to respond well to the active participation required of them and enjoyed the graphical and engaging environment. This feedback is encouraging towards the further exploration of game-based-learning as a learning method for computer programming education, and more specifically towards the incorporation of CMX and other MMORPGs in the classroom.

Furthermore, useful conclusions were drafted reflecting the initial thesis objectives, as follows:

- *Is it possible to develop an educational game that will work as an effective educational tool?*

The evaluation results of the pilot and the first evaluation study showed that the majority of the students increased their knowledge on computer programming by using the CMX. More specifically, 65% of the participants managed to accomplish at least three tasks and correctly wrote one program, achieving the main game's objectives. Furthermore, all results gathered indicated a positive attitude of students of computer programming courses to learn such concepts using an MMORPG like CMX. The majority of the students seemed to respond well to the active participation required of them and enjoyed the graphical and engaging environment.

– *How will such a tool be used to teach computer programming?*

CMX combines entertainment with education: while playing, students are able to perform activities, get feedback on their progress and mistakes, collaborate with their classmates and assess their knowledge comprehension. Also, teachers/administrators are able to customize the entire game, monitor the learning experience, provide scaffolding to students, and include content and activities that will allow the necessary knowledge to be comprehended and skills to be developed. Finally, teachers and supervisors are able to assess all students' activities and receive analytics feedback and evaluation reports from within the game.

– *Is it possible to create an educational game that can be configured and adapted for other courses and domains?*

CMX was based on a special Design Framework that is abstract enough to be employed by future designers and developers and detailed enough to act as a solid guide without allowing many arbitraries. In CMX, teachers can configure the game's graphical environment and the educational content with special editors, such as the Environment Editor, the Dialogue Editor and the Database Editor.

For future work, all the issues that were identified during the study will be addressed, as well as all the concerns that were raised by the students and the teachers in order to maximize CMX's efficiency. Furthermore, the game's material will be expanded, including more units of learning. Finally, the game will be applied in more real world educational settings in order to further test its validity and gather more data on how its features can be improved. Future research will also investigate further the proper exploitation of the Learning Analytics sub-system, the data recorded and how they can be used for the re-design and monitoring of the course. Finally, future work will involve the

examination and employment of the widely accepted Technology Acceptance Model (TAM) (Davis, 1986) during the game's utilization and the evaluation of CMX based on the framework's concepts, such as system use, behavioural intention to use, perceived usefulness, perceived ease of use etc.

Publications

International Journals

1. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014). Designing Educational Games for Computer Programming: A holistic Framework. *Electronic Journal of e-Learning, 12*(3), pp. 281 - 298.
2. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014). Optimization of server performance in the CMX educational MMORPG for Computer Programming. *Computer Science and Information Systems, 11*(4), 1537-1553.
3. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2015). CMX: The effects of an educational MMORPG on learning and teaching Computer Programming. *IEEE Transaction on Learning Technlogies* (under review).
4. Xinogalos, S., Satratzemi, M. & Malliarakis, C. (2015). Microworlds, Games, Animations, Mobile apps, Puzzle editors and more: what is important for an introductory programming environment? *Education and Information Technologies*, DOI: 10.1007/s10639-015-9433-1.

Book Chapters

1. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014). Educational Games for Teaching Computer Programming. In *Research on e-Learning and ICT in Education* (pp. 87-98). Springer New York.
2. Malliarakis, C., Satratzemi, M. &Xinogalos, S. (2015). CMX: An MMORPG for teaching and learning computer programming. *Developing a Game for the purposes of Learning: A series of Case Histories*. Academic Publishing International. (under review).

International Conferences

1. Malliarakis, C., Xinogalos, S. & Satratzemi, M. (2012). Educational games for computer programming. *Proceedings of the 8th Panhellenic Conference with International Participation "Information and Communication Technologies in Education"*, University of Thessaly, Volos, pp. 28-30, September 2012.
2. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2012). Towards the constructive incorporation of serious games within object oriented programming. In *Proceedings*

of the 6th European Conference on Games Based Learning (ECGBL 2012) (pp. 301-308).

3. Malliarakis, C., Satratzemi, M. and Xinogalos, S. (2013). A holistic framework for the development of an educational game aiming to teach computer programming. In *7th European Conference on Games Based Learning: ECGBL2013 (pp. 359-368).*
4. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2013). Towards optimizing server performance in an educational MMORPG for teaching computer programming. In *11th International Conference of Numerical Analysis and Applied Mathematics 2013: ICNAAM 2013* (Vol. 1558, No. 1, pp. 345-348). AIP Publishing.
5. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2013). Towards a new massive multiplayer online role playing game for introductory programming. In *Proceedings of the 6th Balkan Conference in Informatics* (pp. 156-163). ACM.
6. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014). Integrating learning analytics in an educational MMORPG for computer programming. In*Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference on* (pp. 233-237). IEEE.
7. Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014). CMX: Implementing an MMORPG for Learning Programming. In *8th European Conference on Games Based Learning: ECGBL2014* (p. 346-355).
8. Xinogalos, S., Malliarakis, C., Tsmpanoudi, D. and Satrtatzemi, M. (2015). Microworlds, Games and Collaboration: three effective approaches to support novices in learning programming. In Proceedings of the 7th Balkan Conference on Informatics Conference (BCI '15). ACM, New York, NY, USA, Article 39, 8 pages.

References

Abt Associates. (2005). Biography of Clark C. Abt. Retrieved September 19, 2015, from http://www.abtassociates.com/page.cfm?PageID=104

Abt, C. C. (1970). Serious games. New York: The Viking Press.

Adams, E. & Rollings, A. (2007). Fundamentals of Game Design. p. 277 River, NJ: Prentice Hall.

Ahmadzadeh, M., Elliman, D. & Higgins, C. (2005). An analysis of patterns of debugging among novice computer science students. *ACM SIGCSE Bulletin*,*37*(3), 84-88.

Allen, E., Cartwright, R. & Stoler, B. (2002). DrJava: A lightweight pedagogic environment for Java. In *ACM SIGCSE Bulletin* (Vol. 34, No. 1, pp. 137-141). ACM.

Annetta, L. A., Minogue, J., Holmes, S. Y. & Cheng, M. T. (2009). Investigating the impact of video games on high school students' engagement and learning about genetics. *Computers & Education*, *53*(1), 74-85.

Anolli, L. M. & Mantovani, F. (2011). Come funziona la nostra mente. Apprendimento, simulazione e Serious Games. Il mulino.

Anolli, L., Mantovani, F., Confalonieri, L., Ascolese, A. & Peveri, L. (2010). Emotions in Serious Games: From experience to assessment. International Journal of Emerging Technologies in Learning (iJET), 5 (2010).

Armitage, G. (2003). An experimental estimation of latency sensitivity in multiplayer Quake 3. In *Networks, 2003. ICON2003. The 11th IEEE International Conference on* (pp. 137-141). IEEE.

Bailey, M. W. (2005). IRONCODE: think-twice, code-once programming. In *ACM SIGCSE Bulletin* (Vol. 37, No. 1, pp. 181-185). ACM.

Barab, S., Thomas, M., Dodge, T., Carteaux, R. & Tuzun, H. (2005). Making Learning Fun: Quest Atlantis, A Game Without Guns Educational Technology Research and Development, 1, 86-107.

Barker, L. J., McDowell, C. & Kalahar, K. (2009). Exploring factors that influence Computer Science introductory course students to persist in the major. *SIGCSE Bulletin*, 41(1), 153–157.

Barnes, T., Chaffin, A., Powell, E., Lipford, H. (2008). "Game2Learn: Improving the motivation of CS1 students", *Proceedings of the 3rd international conference on Game development in computer science education*, p.1-5, February 27-March 03, 2008, Miami, Florida

Barnes, T., Richter, H., Chaffin, A., Godwin, A., Powell, E., Ralph, T., Matthews, P. & Jordan, H. (2007). The Role of Feedback in Game2Learn, *CHI* 2007, p.1-5.

Barnes, T., Richter, H., Powell, E., Chaffin, A. & Godwin, A. (2007). Game2Learn: building CS1 learning games for retention. In *ACM SIGCSE Bulletin* (Vol. 39, No. 3, pp. 121-125). ACM.

Beaubouef, T. & Mason, J. (2005). Why the high attrition rate for Computer Science students: some thoughts and observations. SIGCSE Bull., 37(2), 103 – 106.

Becker, T. (2010). The Character of Successful Trainings with Serious Games. *International Journal Of Emerging Technologies In Learning (IJET)*, 5(SI3). Retrieved April 17, 2012, from http://online-journals.org/i-jet/article/view/1498.

Beedle, J. B. & Wright, V. H. (2007). Perspectives from multiplayer vídeo Gamers. *Games and simulations in online learning*, 150-174.

Bennedsen, J. & Caspersen, M. E. (2012). Persistence of elementary programming skills. Computer Science Education, 22(2), 81 - 107.

Bernier, Y. W. (2001). Latency compensating methods in client/server in-game protocol design and optimization. In *Game Developers Conference*(Vol. 98033, No. 425).

Bloom, B. S. (1956). *Taxonomy of Educational Objectives: The Classification of Education Goals. Cognitive Domain. Handbook 1*. Longman.

Borella, M. S. (2000). Source models of network game traffic. *computer communications*, *23*(4), 403-410.

Brown, E. & Cairns, P. (2004). A grounded investigation of game immersion. In *CHI'04 extended abstracts on Human factors in computing systems* (pp. 1297-1300). ACM.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P. (1997). "Mini-languages: a way to learn programming principles." *International Journal of Education and Information Technologies*, 2, 65–83.

Campbell, J. P., DeBlois, P. B. & Oblinger, D. G. (2007). Academic analytics: A new tool for a new era. *EDUCAUSE review*, *42*(4), 40.

Carter, J. and Fowler, A. (1998). Object Oriented Students?, SIGCSE Bulletin, Vol. 28, No. 3, 271.

Chaffin, A. Doran, K., Hicks, D. and Barnes, T. (2009). "Experimental evaluation of teaching recursion in a video game", In Proc. *ACM SIGGRAPH Symposium on Video Games*, Sandbox '09, p. 79-86

Chang, F. and Feng, W. (2003). Modeling Player Session Times of On-line Games, ACM NetGames 2003.

Chang, W. C. & Chou, Y. M. (2008). Introductory C programming language learning with game-based digital learning. In *Advances in Web Based Learning-ICWL 2008* (pp. 221-231). Springer Berlin Heidelberg.

Chen, K-T., Huang, P., Lei, C-L. (2006). Game traffic analysis: An MMORPG perspective, Computer Networks, Volume 50, Issue 16, pp. 3002-3023, doi: http://dx.doi.org/10.1016/j.comnet.2005.11.005.

Chmiel, R. and Loui, M.C. (2004) "Debugging: from Novice to Expert": Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, Norfolk, Virginia, USA, March 3-7, 2004. ACM 2004, ISBN 1- 58113 - 798 - 2, 2005.

Chory, R. M. & Goodboy, A. K. (2011). Is basic personality related to violent and non-violent video game play and preferences? *Cyberpsychology, Behavior, and Social Networking*, *14*(4), 191-198.

Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T. & Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers & Education*, *59*(2), 661-686.

Cooper, S., Dann, W. & Pausch, R. (2003). Teaching Objects-first in Introductory Computer Science. ACM SIGCSE Bulletin, 35(1), 191-195.

Coull, N.J. & Duncan, I.M.M. (2011). Emergent requirements for supporting introductory programming. *ITALICS*, 10(1), 78 – 85.

Davis Jr, F. D. (1986). A technology acceptance model for empirically testing new end-user information systems: Theory and results (Doctoral dissertation, Massachusetts Institute of Technology).

de Freitas, S. and Jarvis, S. (2006). *A Framework for Developing Serious Games to meet Learner Needs. Interservice/Industry Training*, Simulation and Education Conference, 2006. Orlando, Florida.

de-Freitas, S. (2006). "Learning in immersive worlds: A review of game based learning", in JISC elearning progrmme. JISC: London.

Del Blanco, Á., Torrente, J., Marchiori, E. J., Martínez-Ortiz, I., Moreno-Ger, P. & Fernández-Manjón, B. (2012). A framework for simplifying educator tasks related to the integration of games in the learning flow. *Journal of Educational Technology & Society*, *15*(4), 305-318.

Desurvire, H., Caplan, M. & Toth, J. A. (2004). Using heuristics to evaluate the playability of games. In *CHI'04 extended abstracts on Human factors in computing systems* (pp. 1509-1512). ACM.

Dickey, M. D. (2007). Game design and learning: A conjectural analysis of how massively multiple online role-playing games (MMORPGs) foster intrinsic motivation. *Educational Technology Research and Development*, *55*(3), 253-273.

Dondlinger, M.J. (2007). *Educational video game design: A review of the literature.* Journal of Applied Educational Technology, 4(1): p. 21-31.

Eagle, M., and Barnes, T. (2009). “Experimental evaluation of an educational game for improved learning in introductory computing”, *SIGCSE Bull*. 41, 1 (March 2009), p. 321-325.

Edwards S. H. (2004). Using Software Testing to Move Students from Trial-and-Error to Reflection-in-Action. In SIGCSE Bulletin inroads, SIGCSE ’04 Proceedings, 36, 1, pp. 26 – 30.

Färber, J. (2002). Network game traffic modelling. In *Proceedings of the 1st workshop on Network and system support for games* (pp. 53-57). ACM.

Federoff, M. A. (2002). Heuristics and usability guidelines for the creation and evaluation of fun in video games (Doctoral dissertation, Indiana University).

Ferguson, C. J. & Garza, A. (2011). Call of (civic) duty: Action games and civic behavior in a large sample of youth. *Computers in Human Behavior*, *27*(2), 770-775.

Fergusson, K., Darllenfyd (2004). Novice Programming Archive, http://ccse.monash.edu.au/~kef/research/readings/archives/cat_novice_programming.html.

Fisch, S. M. (2005). Making educational computer games educational. In *Proceedings of the 2005 conference on Interaction design and children* (pp. 56-61). ACM.

Fletcher, G. H. L. & Lu, J. J., (2009). *Education Human computing skills: rethinking the K-12 experience*. Commun. ACM, 52(2), 23 – 25.

Fleury, A. E. (2000). Programming in Java: student-constructed rules, ACM SIGCSE Bulletin, Vol. 32, Issue 1, 197-201.

Fleury, A. E. (2001). Encapsulation and reuse as viewed by java students. ACM SIGCSE Bulletin, 33(1), 189-193.

Frijda N. (1986). The Emotions (Studies in Emotion and Interaction). Cambridge: Cambridge University Press.

Frijda N. (2007). The Laws of Emotions. Mahwah: Lawrence Erlbaum Associates, Inc.

Garris, R., Ahlers, R. & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & gaming*, *33*(4), 441-467.

Garzotto, F. (2007). Investigating the educational effectiveness of multiplayer online games for children. In *Proceedings of the 6th international conference on Interaction design and children* (pp. 29-36). ACM.

Gee, J. P. (2003). What video games have to teach us about learning and literacy. New York: Palgrave Macmillan

Gee, J. P. (2004). Learning by design: Games as learning machines. *Interactive Educational Multimedia*, *8*(1), 15-23.

Gee, J. P. (2006). Why game studies now? Video games: A new art form.*Games and culture*, *1*(1), 58-61.

Gomes, A. & Mendes, A. J. (2007). Learning to Program - Difficulties and Solutions, International Conference on Engineering Education – ICEE 2007, p. 283 – 287.

Green, C. S. & Bavelier, D. (2007). Action-video-game experience alters the spatial resolution of vision. *Psychological science*, *18*(1), 88-94.

Green, C. S. & Bavelier, D. (2012). Learning, attentional control, and action video games. *Current Biology*, *22*(6), R197-R206.

Greitzer, F. L., Kuchar, O. A. & Huston, K. (2007). Cognitive science implications for enhancing training effectiveness in a serious gaming context.*Journal on Educational Resources in Computing (JERIC)*, *7*(3), 2.

Gunter, G. A., Kenny, R. F. & Vick, E. H. (2008). Taking educational games seriously: using the RETAIN model to design endogenous fantasy into standalone educational games. *Educational Technology Research and Development*, 56(5/6), p. 511-537.

Guo, K., Mukherjee, S., Rangarajan, S. & Paul, S. (2003). A fair message exchange framework for distributed multi-player games. In*Proceedings of the 2nd workshop on Network and system support for games* (pp. 29-41). ACM.

Guzdial, M. and Soloway, E. (2002). *Log on Education: Teaching the Nintendo Generation how to Program*, Communications of the ACM, 45(4).

Guzdial, M., (2004). Programming environments for novices. In S. Fincher and M. Petre (Eds.), *Computer Science Education Research*, 127-154. Lisse, The Netherlands: Taylor & Francis.

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. Computers & Education, 54(4), 1127 - 1136.

Hendrix, T.D., Cross, J.H., and Barowski, L.A. (2004). An extensible framework for providing dynamic data structure visualizations in a lightweight IDE. ACM SIGCSE Bulletin, 36(1), 387-391.

Ho, P.C., Chung, S.-M. & Tsai, M.-H. (2006). A Case Study of Game Design for E-Learning. In *Z. Pan et al. (Eds.): Edutainment*, LNCS 3942, p. 453–462.

Holland, S., Griffiths, R., and Woodman, M. (1997). Avoiding object misconceptions. In Proceedings of the 28th SIGCSE, 131-134.

Hristova, M., A. Misra, M. Rutter and R. Mercuri, (2003). Identifying and correcting java programming errors for introductory computer science students. Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, (CSE' 03), ACM Press, New York, USA., pp: 153-156. DOI: 10.1145/611892.611956

Hwang, G. J. & Wu, P. H. (2012). Advancements and trends in digital game-based learning research: a review of publications in selected journals from 2001 to 2010. *British Journal of Educational Technology*, *43*(1), E6-E10.

Ibrahim, R. & Jaafar, A. (2009). Educational Games (EG) design framework: Combination of game design, pedagogy and content modeling. In *Electrical Engineering and Informatics, 2009. ICEEI'09. International Conference on* (Vol. 1, pp. 293-298). IEEE.

Jackson, L. A., Witt, E. A., Games, A. I., Fitzgerald, H. E., von Eye, A. & Zhao, Y. (2012). Information technology use and creativity: Findings from the Children and Technology Project. *Computers in human behavior*, *28*(2), 370-376.

Jansiewicz, D. R. (1973). The New Alexandria Simulation: A Serious Game of State and Local Politics. Canfield Press.

Jansiewicz, D. R. (2011). The Game of Politics. Retrieved September 19, 2015, from http://www.gameofpolitics.com/

Jehaes, T., De Vleeschauwer, D., Coppens, T., Van Doorselaer, B., Deckers, E., Naudts, W. & Smets, R. (2003). Access network delay in networked games. In *Proceedings of the 2nd workshop on Network and system support for games* (pp. 63-71). ACM.

Johnson, D. & Wiles, J. (2003). Effective affective user interface design in games. *Ergonomics*, *46*(13-14), 1332-1345.

Johnson, L., Adams, S. & Cummins, M. (2012). NMC Horizon Report 2012 Higher Education Edition, vol. 2012, no. 2 March 2012. The New Media Consortium, 2012, p. 42.

Johnson, L., Adams, S., Cummins, M., Estrada, V., Freeman, A. & Ludgate, H. (2013). The NMC horizon report: 2013 higher education edition.

Kafai, Y. B. (1995). Minds in play: Computer game design as a context for children's learning. Mahwah, NJ: Lawrence Erlbaum Associates.

Kahn, M. A. & Perez, K. M. (2009). The game of politics simulation: An exploratory study. *Journal of Political Science Education*, *5*(4), 332-349.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. & Crawford, K. (2000). Problems-based learning for foundation computer science courses. Computer Science Education, 10(2), pp. 109-128.

Kayes, A. B., Kayes, D. C. & Kolb, D. A. (2005). "Experiential learning in teams" Simulation & Gaming 36(3): 330-354.

Kebritchi, M., Hirumi, A. & Bai, H. (2010). The effects of modern mathematics computer games on mathematics achievement and class motivation. *Computers & education*, *55*(2), 427-443.

Keith, N. & Frese, M. (2008). Effectiveness of error management training: a meta-analysis. *Journal of Applied Psychology*, *93*(1), 59.

Kiili, K. (2005). Content creation challenges and flow experience in educational games: The IT-Emperor case. *The Internet and Higher Education*, *8*(3), 183-198.

Kinnunen, P. & Malmi, L., (2006). Why students drop out CS1 course? Paper presented at the Proceedings of the second international workshop on Computing education research, ICER'06, 97-108.

Klimmt, C. (2009). Key dimensions of contemporary video game literacy: Towards a normative model of the competent digital gamer. *Eludamos. Journal for Computer Game Culture*, *3*(1), 23-31.

Kölling, M. (2010). The greenfoot programming environment. Trans. Comput. Educ., 10:14:1--14:21.

Kölling, M., Quig, B., Patterson, A. & Rosenberg, J. (2003). The BlueJ system and its pedagogy. Computer Science Education, Special issue on Learning and Teaching Object Technology, 13(4), 249-268.

Koulouri, T., Lauria, S. & Macredie, R. D. (2014). Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches. *ACM Transactions on Computing Education (TOCE)*, *14*(4), 26.

Krathwohl, D. R. (1993). *Methods of educational and social science research: An integrated approach*. Longman/Addison Wesley Longman.

Lahtinen, E., Ala-Mutka, K. and Jarvinen, H. (2005). *A Study of Difficulties of Novice Programmers. In: Innovation and Technology*, Computer Science Education, p. 14–18.

Lahtinen, E., Ala-Mutka, K. & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin* (Vol. 37, No. 3, pp. 14-18). ACM.

Lazzaro, N. & Keeker, K. (2004). What's my method?: a game show on games. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems* (pp. 1093-1094). ACM.

Lebow, D. (1993). Constructivist values for instructional systems design: Five principles toward a new mindset. *Educational technology research and development*, *41*(3), 4-16.

Lebow, D. G. & Wager, W. W. (1994). Authentic Activity as a Model for Appropriate Learning Activity: Implications for Design of Computer-Based Simulations.

Lee, M.J. and Ko, A.J. (2011). *Personifying Programming Tool Feedback Improves Novice Programmers' Learning*, Conference on International Computing Education Research (ICER), August 8-9, Providence, Rhode Island, USA, p. 109-116.

Li, F.W.B. and Watson, C. (2011). *Game-based concept visualization for learning programming*, Proceedings of the third international ACM workshop on Multimedia technologies for distance learning, p. 37-42, December 01-01, 2011, Scottsdale, Arizona, USA.

Lias, T. E. & Elias, T. (2011). Learning Analytics: The Definitions, the Processes, and the Potential.

Long, J. (2007). Just For Fun: Using Programming Games in Software Programming Training and Education. *Journal of Information Technology Education: Research*, *6*(1), 279-290.

Malliarakis, C., Xinogalos, S. & Satratzemi, M. (2012a). Educational games for computer programming. *Proceedings of the 8th Panhellenic Conference with International Participation "Information and Communication Technologies in Education"*, University of Thessaly, Volos, pp. 28-30, September 2012.

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2012b). Towards the constructive incorporation of serious games within object oriented programming. In *Proceedings of the 6th European Conference on Games Based Learning (ECGBL 2012)* (pp. 301-308).

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2013a). Towards a new massive multiplayer online role playing game for introductory programming. In *Proceedings of the 6th Balkan Conference in Informatics* (pp. 156-163). ACM.

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2013b). Towards optimizing server performance in an educational MMORPG for teaching computer programming. In *11th International Conference of Numerical Analysis and Applied Mathematics 2013: ICNAAM 2013* (Vol. 1558, No. 1, pp. 345-348). AIP Publishing.

Malliarakis, C., Satratzemi, M. and Xinogalos, S. (2013c). A holistic framework for the development of an educational game aiming to teach computer programming. In *7th European Conference on Games Based Learning: ECGBL2013 (pp. 359-368).*

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014a). Designing Educational Games for Computer Programming: A holistic Framework. *Electronic Journal of e-Learning*, *12*(3).

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014b). CMX: Implementing an MMORPG for Learning Programming. In *8th European Conference on Games Based Learning: ECGBL2014* (p. 346-355).

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014c). Integrating learning analytics in an educational MMORPG for computer programming. In *Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference on* (pp. 233-237). IEEE.

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2014d). Optimization of server performance in the CMX educational MMORPG for Computer Programming. *Computer Science and Information Systems*, *11*(4), 1537-1553.

Malone, T. (1980). *What makes thing fun to learn? Heuristics for designing instructional computer games.* ACM.

Maragos, K. and Grigoriadou, M. (2011). "Exploiting TALENT as a Tool for Teaching and Learning", *The International Journal of Learning*, Volume 18, Issue 1, pp.431-440.

Maragos, K. and Grigoriadou, M., (2005), "Towards the design of Intelligent Educational Gaming systems" Proceedings of Workshop on Educational Games as Intelligent learning environments, Artificial Intelligence in Education, University of Amsterdam, Amsterdam

Mayes, T. & de Freitas,S. (2006). Learningande- learning: The role of theory. In H. Beetham & R. Sharpe. Rethinking Pedagogy for the Digital Age. London. Routledge.

Mayes, T. and de Freitas, S. (2004). Review of e- learning theories, frameworks and models. JISC e- learning models study report. London. The Joint Information Systems Committee. See: http://www.jisc.ac.uk/elp_outcomes.html.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, SIGCSE Bulletin, 33(4), pp. 125-180.

Michael, D. & Chen, S. (2005). Serious Games: Games That Educate, Train, and Inform (1er ed.). Course Technology PTR.

Miller, D. J. & Robertson, D. P. (2011). Educational benefits of using game consoles in a primary classroom: A randomised controlled trial. *British Journal of Educational Technology*, *42*(5), 850-864.

Milne, I. and Rowe, G. (2002). Difficulties in Learning and Teaching Programming – Views of Students and Tutors, Education and Information Technologies 7:1, Kluwer Publishers, 55-66.

Moser, R. (1997). A fantasy adventure game as a learning environment: why learning to program is so difficult and what can be done about it. In *ACM SIGCSE Bulletin* (Vol. 29, No. 3, pp. 114-116). ACM.

Muratet, M., Torguet, P., Jessel, J. P. & Viallet, F. (2009). Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, *2009*, 3.

Muratet, M., Torguet, P., Viallet, F. & Jessel, J. P. (2010). Experimental feedback on prog&play, a serious game for programming practice (educational paper). *Eurographics (EG), page (média électroniq ue), http://www. eg. org*.

Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Velázquez-Iturbide, J.Á. (2003), Exploring the role of visualization and engagement in computer science education, SIGCSE Bulletin, 35(2), pp. 131-152.

Natkin, S. & Yan, C. (2006). User model in multiplayer mixed reality entertainment applications. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology* (p. 85). ACM.

O' Kelly, J. & Gibson, P. (2006). RoboCode & problem-based learning: A non-prescriptive approach to teaching programming. *ACM SIGCSE Bulletin*, 38, 3, p. 217-221.

Oatley, K. G. (1996). Emotions, rationality and informal reasoning. Mental models in cognitive science: Essays in honour of Phil Johnson-Laird, 175-196.

Ogletree, S. M. & Drake, R. (2007). College students' video game participation and perceptions: Gender differences and implications. *Sex Roles*, *56*(7-8), 537-542.

Pagulayan, R., Keeker, K., Wixon, D., Romero, R. & Fuller, T. (2003). User-centered design in games. *Human-Computer Interaction Handbook: Fundamentals, Evolving Techniques and Emerging Applications*. J. A. Jacko and A. Sears (eds.). Lawrence Erlbaum Associates, Mahwah, NJ, 883-905.

Paliokas, I., Arapidis, C. & Mpimpitsos, M. (2011). PlayLOGO 3D: A 3D interactive video game for early programming education. *Third International Conference on Games and Virtual Worlds for Serious Applications*, p. 24-31.

Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education*, *52*(1), 1-12.

Phelps, A, Bierre, K, and Parks, D. (2003). *MUPPETS: multi-user programming pedagogy for enhancing traditional study*, Proceeding of the 4th conference on Information technology education , October 2003, Lafayette, Indiana, USA, p. 100-105.

Piaget, J. and Inhelder, B. (1967). The Child's Conception of Space. See especially "Systems of Reference and Horizontal-Vertical Coordinates." p. 375-418. New York: W. W. Norton &Co.

Pillay, N. and Jugoo V. R. (2006). An Analysis of Errors Made by Novice Programmers in a First Course in Procedural Programming in Java, full research paper in the Proceedings of SACLA 2006, Van Belle J. and Brown I. (eds.), 11 - 20, ISBN: 0-620-36150-6.

Pinelle, D. and Wong, N. (2008). *Heuristic evaluation for games: usability principles for video game design*. in *Proceedings of ACM CHI 2008 Conference on Human Factors in Computing Systems*.

Piteira, M. and Haddad, S. (2011). *Innovate in Your Program Computer Class: An approach based on a serious game*. OSDOC - Open Source and Design of Communication Workshop.

Prensky, M., (2001). *Digital Game-Based Learning*. New York: Mc Graw Hill.

Quax, P., Monsieurs, P., Lamotte, W., De Vleeschauwer, D. & Degrande, N. (2004). Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. In*Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games* (pp. 152-156). ACM.

Radosevic, D., Orehovački, T. & Lovrenčic, A. (2009). Verificator: educational tool for learning programming. *Informatics in Education-An International Journal*, (Vol 8_2), 261-280.

Ragonis, N. and Ben-Ari, M. (2005). *A long-Term Investigation of the Comprehension by Novices*, Computer Science Education, Vol. 15, No. 3, 203-221.

Ragonis, N. & Ben-Ari, M. (2005). A long-term investigation of the comprehension of OOP concepts by novices.

Rajaravivarma, R. (2005). A games-based approach for teaching the introductory programming course. *ACM SIGCSE Bulletin*, *37*(4), 98-102.

Resnick, M. (2007). *All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten.* Proceedings of the SIGCHI Conference on Creativity and Cognition.

Ricci, K. E., Salas, E. & Cannon-Bowers, J. A. (1996). Do computer-based games facilitate knowledge acquisition and retention? *Military Psychology*, *8*(4), 295.

Roslina, I. & Nazli, Y. (2009). Development and effectiveness of educational games for learning introductory programming. CTL Research Report, UTM, Malaysia, Skudai, Johor.

Rouse, R. (2001). Game Design Theory and Practice. Plano, Texas: Wordware Publishing.

Rowe, G., Thorburn, G. (2000). VINCE - an on-line tutorial tool for teaching introductory programming, British Journal of Educational Technology, 31(4), pp. 359 - 11p.

Rozin, P. (1999). Preadaptation and the puzzles and properties of pleasure. Well-being: The foundations of hedonic psychology, 109-133.

Sadler, T. D., Romine, W. L., Stuart, P. E. & Merle-Johnson, D. (2013). Game-Based Curricula in Biology Classes: Differential Effects Among Varying Academic Levels. *Journal of Research in Science Teaching*, *50*(4), 479-499.

Salen, K. & Zimmerman, E. (2004). *Rules of play: Game design fundamentals*. MIT press.

Saltzman, M. (2000). Game Design: Secrets of the Sages, Second Edition. Indianapolis, IN: Macmillan Publishing.

Sanders, D., and Dorn, B. (2003). Jeroo: a tool for introducing object-oriented programming, ACM SIGCSE Bulletin, 35(1), 201-204.

Sandford, R. (2006). Teaching with COTS. Paper presented at the JISC Online Conference Innovating e-Learning 2006. March 30th.

Savery, J. R. & Duffy, T. M. (1996). Constructivist learning environments: Case studies in instructional design.

Schuller, D. (2011). *C# Game Programming: For Serious Game Creation*. Cengage Learning, pp.4.

Serrano, Á., Marchiori, E. J., Blanco, Á. D., Torrente, J. & Fernández-Manjón, B. (2012). A framework to improve evaluation in educational games. In*Global Engineering Education Conference (EDUCON), 2012 IEEE* (pp. 1-8). IEEE.

Shanahan, J. (2009) Students Create Game-based Online Learning Environment that Teaches Java Programing, ACMSE.

Sharafi, P., Hedman, L. & Montgomery, H. (2006). Using information technology: engagement modes, flow experience, and personality orientations. *Computers in Human Behavior*, *22*(5), 899-916.

Sheldon, N. Girard, E., Borg, S. Claypool, M. Agu. E. (2003). The Effect of Latency on User Performance in Warcraft III, Technical Report WPICS-TR-03-07, Computer Science Department, Worcester Polytechnic Institute.

Siemens, G., (2010) What are Learning Analytics? From http://www.elearnspace.org/blog/2010/08/25/what-are-learning-analytics/

Singhal S. and Zyda, M. (1999). Networked Virtual Environments: Design and Implementation. Addison Wesley, ISBN 0-201-32557, ACM Press

Skoric, M. M., Teo, L. L. C. & Neo, R. L. (2009). Children and video games: addiction, engagement, and scholastic achievement. *Cyberpsychology & behavior*, *12*(5), 567-572.

Smed, J. Kaukoranta, T. Hakonen, H. (2002). A Review on Networking and Multiplayer Computer Game, Turku Centre for Computer Science.

Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. Communications of the ACM, 29(9), 850-858.

Song, M. & Zhang, S. (2008). EFM: A model for educational game design. In *Technologies for e-learning and digital entertainment* (pp. 509-517). Springer Berlin Heidelberg.

Spohrer, J. C. & Soloway, E. (1986). *Novice Mistakes: Are the Folk Wisdoms Correct? Communications of the ACM*, Vol. 29, No. 7, pp. 624 - 632.

Squire, K.D. (2006). From content to context: Video games as designed experiences. Educational Researcher, 35(8), 19-29.

Suznjevic, M., Dobrijevic, O. & Matijasevic, M. (2009). MMORPG player actions: Network performance, session patterns and latency requirements analysis. *Multimedia Tools and Applications*, *45*(1-3), 191-214.

Sweetser, P. & Johnson, D. (2004). Player-centered game environments: Assessing player opinions, experiences, and issues. In *Entertainment Computing–ICEC 2004* (pp. 321-332). Springer Berlin Heidelberg.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning.*Cognitive science*, *12*(2), 257-285.

Terlecki, M., Brown, J., Harner-Steciw, L., Irvin-Hannum, J., Marchetto-Ryan, N., Ruhl, L. & Wiggins, J. (2011). Sex differences and similarities in video game experience, preferences, and self-efficacy: Implications for the gaming industry. *Current Psychology, 30*(1), 22-33.

Torrente, J., Blanco, Á. d., Serrano-Laguna, Á., Vallejo-Pinto, J. Á., Moreno-Ger, P., Fernández-Manjón, B. (2014). Towards a Low Cost Adaptation of Educational Games for People with Disabilities. Computer Science and Information Systems, Vol. 11, No. 1, 369–391.

Torrente, J., Del Blanco, Á., Marchiori, E. J., Moreno-Ger, P. & Fernández-Manjón, B. (2010, April). < e-Adventure>: Introducing educational games in the learning process. In *Education Engineering (EDUCON), 2010 IEEE* (pp. 1121-1126). IEEE.

Ventura, M. & Shute, V. (2013). The validity of a game-based assessment of persistence. *Computers in Human Behavior*, *29*(6), 2568-2572.

Ventura, M., Shute, V. & Kim, Y. J. (2012). Video gameplay, personality and academic performance. *Computers & Education*, *58*(4), 1260-1266.

Ventura, M., Shute, V. & Zhao, W. (2013). The relationship between video game use and a performance-based measure of persistence. *Computers & Education*, *60*(1), 52-58.

Vorderer, P., Hartmann, T. & Klimmt, C. (2003). Explaining the enjoyment of playing video games: The role of competition. *In Proceedings of the Second International Conference on Computer Games* (Carnegie Mellon Univ., Pittsburgh, PA).

W. Greller and H. Drachsler, Translating Learning into Numbers: A Generic Framework for Learning Analytics. Educational Technology & Society, 15 (3), 2012, pp. 42–57.

Werger, E. (1999). Communities of Practice, Learning, Meaning, and Identity, *University Cambridge Press*, Cambridge UK, 1999.

Wilson, B. C. (2002). A study of factors promoting success in Computer Science including gender differences. Computer Science Education, 12(1-2), 141-164.

Witt, E. A., Massman, A. J. & Jackson, L. A. (2011). Trends in youth's videogame playing, overall computer use, and communication technology use: The impact of self-esteem and the Big Five personality factors. *Computers in Human Behavior*, *27*(2), 763-769.

Wolber, D. (2011). App inventor and real-world motivation. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 601-606). ACM.

Wong, W.L. L.N.; Cuihua, S; Eduardo, C; Fei, T; Shiyamvar, B; Harishkumar, N; Hua, W; Ute, R. (2007). *Serious Video Game Effectiveness*. In ACE 07, Salzburg, Austria. ACM.

Yee, N. (2006). The demographics, motivations, and derived experiences of users of massively multi-user online graphical environments. *Presence, 15*(3), 309-329.

Yu-Sheng: Designing Fast-Action Games for the Internet, Gamasutra (Online Game Developer Magazine), NG. (1997)

Yusoff, A., Crowder, R., Gilbert, L. & Wills, G. (2009), A Conceptual Framework for Serious Games, in The 9th IEEE International Conference on Advanced Learning Technologies (ICALT 2009) vol., no., pp.21-23, 15-17 July 2009 doi: 10.1109/ICALT.2009.19.

Zualkernan, I. A. (2006) A framework and a methodology for developing authentic constructivist e-Learning environments. *Educational Technology & Society*, 9 (2), 198-212.

Appendix A - Questionnaire of 1st and 2nd evaluation

Name

Email

I. Students' knowledge in programming and interest in games

1. Have you ever played video games? *

	Yes
	No

2. If you are still playing games, how often do you play? *

Daily	1-2 times a week	3-4 times a week	Rarely

3. What kinds of games do you prefer? *

Arcade	
Sports	
MMORPG	
Serious Games	
Strategy	
Retro (tetris, snake, pacman…)	
Social Multiplayer (Facebook, Farmville)	
Other	

4. How much time do you devote in playing each time? *

0	<1 hour	1-2 hours	2-3 hours	3-4 hours	>4 hours

5. Do you study programming except for school? *

	Yes
	No

6. How many hours do you devote in programming besides the courses? *

0	<1 hour	1-2 hours	3-5 hours	>5 hours

7. How many hours did or do you devote weekly for each programming course in your studies?

0	<1 hour	1-2 hours	3-5 hours	>5 hours

II. The game's performance

8. Evaluate CMX's response during players' movements.

Poor				Excellent
1	2	3	4	5

9. Evaluate CMX's response during message transmission on chat.

Poor				Excellent
1	2	3	4	5

10. Evaluate CMX's response during decision making.

Poor				Excellent
1	2	3	4	5

11. Evaluate CMX’s response regarding players' interactions.

Poor				Excellent
1	2	3	4	5

III. The game’s entertaining and motivating elements

12. Evaluate how much you enjoyed the plot of the game in regards to password discovery.

Poor				Excellent
1	2	3	4	5

13. Evaluate how much you were entertained by the game's graphics.

Poor				Excellent
1	2	3	4	5

14. Evaluate how much you were entertained and motivated by the game's elements (e.g. weapons, health rating, inventory, spells).

Poor				Excellent
1	2	3	4	5

15. Evaluate how much you found the dialogue with the Senseis, the multiple choice questions and right/wrong questions entertaining and motivating to answer correctly.

Poor				Excellent
1	2	3	4	5

16. Evaluate how much you found the code writing with drag & drop commands during training with the Iron Senseis entertaining and motivating to answer correctly

Poor				Excellent
1	2	3	4	5

17. Evaluate how much the sense of victory would enable you to learn the programming concepts better.

Poor				Excellent
1	2	3	4	5

IV. Game-student interactions and the game's difficulty

18. Evaluate how much the sense of victory would enable you to learn the programming concepts better.

Poor				Excellent
1	2	3	4	5

19. Evaluate how easy a novice programmer would find the interaction with the Iron Senseis?

Poor				Excellent
1	2	3	4	5

20. Evaluate how easy a novice programmer would find the interaction with the Gold Senseis?

Poor				Excellent
1	2	3	4	5

21. The game provides constant and full information on what is occurring (Feedback).

Poor				Excellent
1	2	3	4	5

22. The language used in the dialogue, messages, menu etc. is simple and natural.

Poor				Excellent
1	2	3	4	5

23. The volume of the provided information does not create confusion. (Aesthetic and minimalistic design)

Poor				Excellent
1	2	3	4	5

24. The same choices, situations and actions are expressed/executed in the same way throughout the entire programming environment (Consistency).

Poor				Excellent
1	2	3	4	5

25. How easily could you understand the game's scenario and the tasks you had to do to win the game?

Poor				Excellent
1	2	3	4	5

26. Could you easily understand how to move your character within the game and how to interact with the other players?

Poor				Excellent
1	2	3	4	5

27. The mini map included in the game helped me to collaborate with my teammates.

Poor				Excellent
1	2	3	4	5

28. Features such as the chat tool and the mini map helped me to collaborate more with my teammates.

Poor				Excellent
1	2	3	4	5

V. The game's educational aspects

29. Features such as the chat tool and the mini map helped me to collaborate more with my teammates.

Poor				Excellent
1	2	3	4	5

30. Evaluate to what degree a novice programmer would manage to learn computer programming while playing the game.

Poor				Excellent
1	2	3	4	5

31. Evaluate to what degree the educational content was properly combined with game playing.

Poor				Excellent
1	2	3	4	5

32. Evaluate to what degree CMX should be combined with teaching computer programming courses.

Poor				Excellent
1	2	3	4	5

33. Evaluate to what degree CMX enables students to learn by themselves by playing.

Poor				Excellent
1	2	3	4	5

34. Evaluate to what degree the gameplay parts of CMX (points, chat etc) would help novice programmers in the educational process.

Poor				Excellent
1	2	3	4	5

35. Evaluate to what degree novice programmers would prefer using games to learn computer programming in comparison to traditional learning methods in class.

Poor				Excellent
1	2	3	4	5

36. Evaluate to what degree novice programmers would like educational games to be used in more computer programming courses.

Poor				Excellent
1	2	3	4	5

37. Evaluate to what degree novice programmers would like educational games such as CMX to be available online for easy access.

Poor				Excellent
1	2	3	4	5

38. Evaluate how much easier and clearer the process of learning computer programming will be with CMX for novice programmers.

Poor				Excellent
1	2	3	4	5

39. Evaluate to what degree the activities would help novice programmers to use already gained knowledge in order to successfully complete their goals while they were playing the game.

Poor				Excellent
1	2	3	4	5

Comments

40. Do you have to make any comments?

Appendix B - Questionnaire of 3rd evaluation

Name

Email

I. Students' knowledge in programming and interest in games

1. Have you ever played video games? *

	Yes
	No

2. If you are still playing games, how often do you play? *

Daily	1-2 times a week	3-4 times a week	Rarely

3. What kinds of games do you prefer? *

Arcade	
Sports	
MMORPG	
Serious Games	
Strategy	
Retro (tetris, snake, pacman…)	
Social Multiplayer (Facebook, Farmville)	
Other	

4. How much time do you devote in playing each time? *

0	<1 hour	1-2 hours	2-3 hours	3-4 hours	>4 hours

5. Do you study programming except for school? *

	Yes
	No

6. How many hours do you devote in programming besides the courses? *

0	<1 hour	1-2 hours	3-5 hours	>5 hours

7. How many hours did or do you devote weekly for each programming course in your studies?

0	<1 hour	1-2 hours	3-5 hours	>5 hours

II. The game's performance

8. Evaluate CMX's response during players' movements.

Poor				Excellent
1	2	3	4	5

9. Evaluate CMX's response during message transmission on chat.

Poor				Excellent
1	2	3	4	5

10. Evaluate CMX's response during decision making.

Poor				Excellent
1	2	3	4	5

11. Evaluate CMX’s response regarding players' interactions.

Poor				Excellent
1	2	3	4	5

III. The game’s entertaining and motivating elements

12. Evaluate how much you were entertained by the plot of the game in regards to password discovery.

Poor				Excellent
1	2	3	4	5

13. Evaluate how much you were entertained by the game's graphics.

Poor				Excellent
1	2	3	4	5

14. Evaluate how much you were entertained and motivated by the game's elements (e.g. weapons, health rating, inventory, spells).

Poor				Excellent
1	2	3	4	5

15. Evaluate how much a novice programmer would find the dialogue with the Senseis, the multiple choice questions and right/wrong questions entertaining and motivating to answer correctly.

Poor				Excellent
1	2	3	4	5

16. Evaluate how much a novice programmer would find the code writing with drag & drop commands during training with the Iron Senseis entertaining and motivating to answer correctly

Poor				Excellent
1	2	3	4	5

17. Evaluate how much the sense of victory would enable a novice programmer to learn the programming concepts better.

Poor				Excellent
1	2	3	4	5

IV. Game-student interactions and the game's difficulty

18. Evaluate how much the sense of victory would enable a novice programmer to learn the programming concepts better.

Poor				Excellent
1	2	3	4	5

19. Evaluate how easy a novice programmer would find the interaction with the Iron Senseis?

Poor				Excellent
1	2	3	4	5

20. Evaluate how easy a novice programmer would find the interaction with the Gold Senseis?

Poor				Excellent
1	2	3	4	5

21. The game provides constant and full information on what is occurring (Feedback).

Poor				Excellent
1	2	3	4	5

22. The language used in the dialogue, messages, menu etc. is simple and natural.

Poor				Excellent
1	2	3	4	5

23. The volume of the provided information does not create confusion. (Aesthetic and minimalistic design)

Poor				Excellent
1	2	3	4	5

24. The same choices, situations and actions are expressed/executed in the same way throughout the entire programming environment (Consistency).

Poor				Excellent
1	2	3	4	5

25. How easily could you understand the game's scenario and the tasks you had to do to win the game?

Poor				Excellent
1	2	3	4	5

26. Could you easily understand how to move your character within the game and how to interact with the other players?

Poor				Excellent
1	2	3	4	5

27. The mini map included in the game helped me to collaborate with my teammates.

Poor				Excellent
1	2	3	4	5

28. Features such as the chat tool and the mini map helped me to collaborate more with my teammates.

Poor				Excellent
1	2	3	4	5

V. The game's educational aspects

29. Features such as the chat tool and the mini map helped me to collaborate more with my teammates.

Poor				Excellent
1	2	3	4	5

30. Evaluate to what degree a novice programmer would manage to learn computer programming while playing the game.

Poor				Excellent
1	2	3	4	5

31. Evaluate to what degree the educational content was properly combined with game playing.

Poor				Excellent
1	2	3	4	5

32. Evaluate to what degree CMX should be combined with teaching computer programming courses.

Poor				Excellent
1	2	3	4	5

33. Evaluate to what degree CMX enables students to learn by themselves by playing.

Poor				Excellent
1	2	3	4	5

34. Evaluate to what degree the gameplay parts of CMX (points, chat etc) would help novice programmers in the educational process.

Poor				Excellent
1	2	3	4	5

35. Evaluate to what degree novice programmers would prefer using games to learn computer programming in comparison to traditional learning methods in class.

Poor				Excellent
1	2	3	4	5

36. Evaluate to what degree novice programmers would like educational games to be used in more computer programming courses.

Poor				Excellent
1	2	3	4	5

37. Evaluate to what degree novice programmers would like educational games such as CMX to be available online for easy access.

Poor				Excellent
1	2	3	4	5

38. Evaluate how much easier and clearer the process of learning computer programming will be with CMX for novice programmers.

Poor				Excellent
1	2	3	4	5

39. Evaluate to what degree the activities would help novice programmers to use already gained knowledge in order to successfully complete their goals while they were playing the game.

Poor				Excellent
1	2	3	4	5

Comments

40. Do you have to make any comments?

Appendix C - Example game – Learning unit of functions

A. Setting the environment

The first step of setting up the game is to configure the environment using the editor, as shown in Figure C-1. There, we can use drag & drop movements to arrange the arenas, either using the numerous images that have already been imported in the game, or loading our own images. The environment is divided into tiles sized 25x25 pixels each.

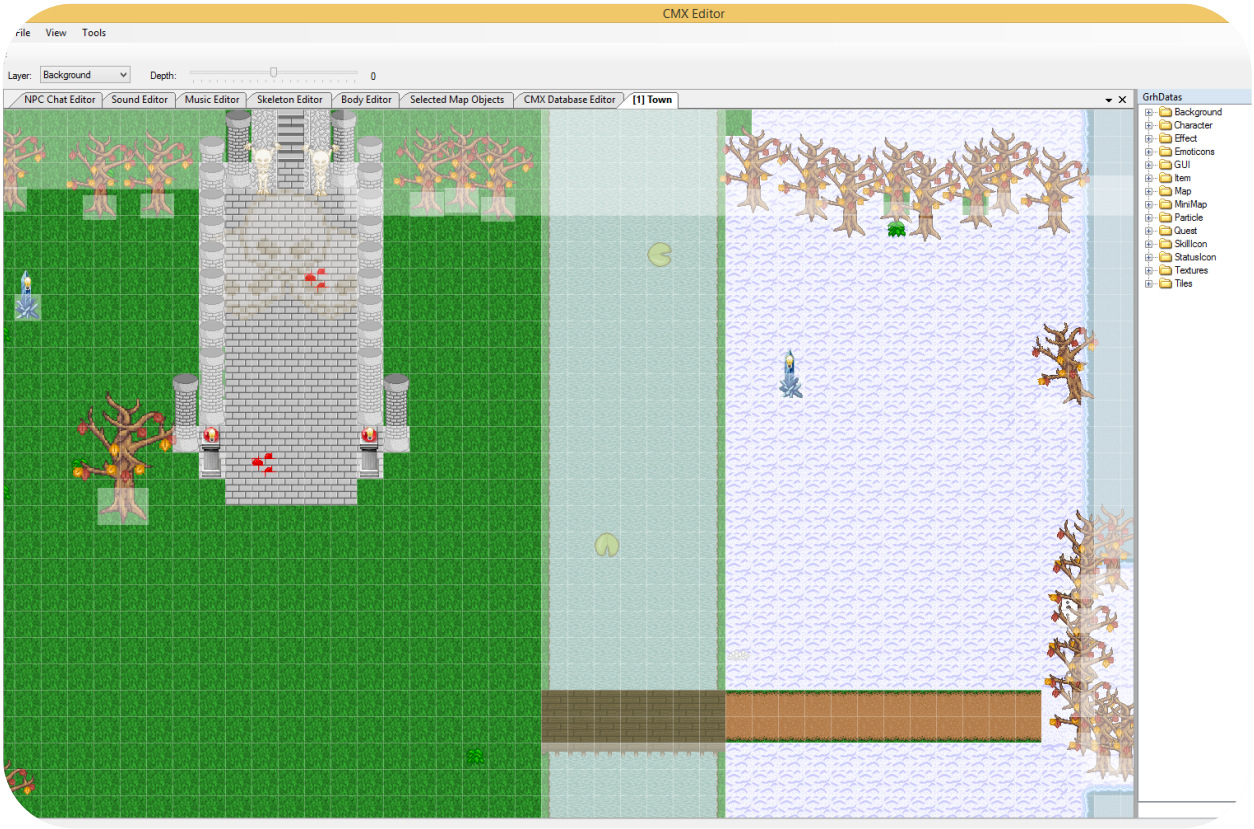

Figure C-1: CMX editor's environment

It should be noted that in the above virtual world there are three different layers; the background, the foreground and the dynamic layer. The background is located behind the players and usually represents the floor, the foreground is located above the floor and usually represents an object that is located on top of the background, while the dynamic shows anything that is located on the same level as the player, over the background which usually has some abilities and the player cannot pass "through" it while navigating the world (e.g. a tree, a river, a house etc).

Once we define the size of the world and we draw it based on our needs, we are ready to proceed to the next steps with setting the character and item templates, that is with creating the dynamic entities that own some interactive characteristics within the game.

B. Setting the characters' templates

The game's Database editor gives the teachers the opportunity to define specific characters that will participate in CMX. In the specific example shown in Figure C-2, we create a template we name "guards". Once we create the template, we can add as many guard characters in the world as we want, without having to use separate template for each one.

During the template configuration, we define the template's name and whether the character will possess a specific object such as equipment, weapons etc. also, we have the ability to define percentage probabilities in regards to how the character will be shown whenever he appears in the game. For example we can say that there will be 30% probability the guard will appear with a blue sword, a 20% probability he will appear with a stone, 10% probability he will appear with a healing potion etc.

Furthermore, we can define in which alliance the character will belong, e.g. if he will be hostile or friendly towards other players, what will he look like and how many points will he provide to the players as rewards if he is eliminated. Also, we can determine whether the character can interact and talk with the players and which specific dialogue the character will start when the player will attempt to talk with him.

Finally, the editor defines if the character will re-appear when eliminated and after how long (respawn) and with which speed he will move.

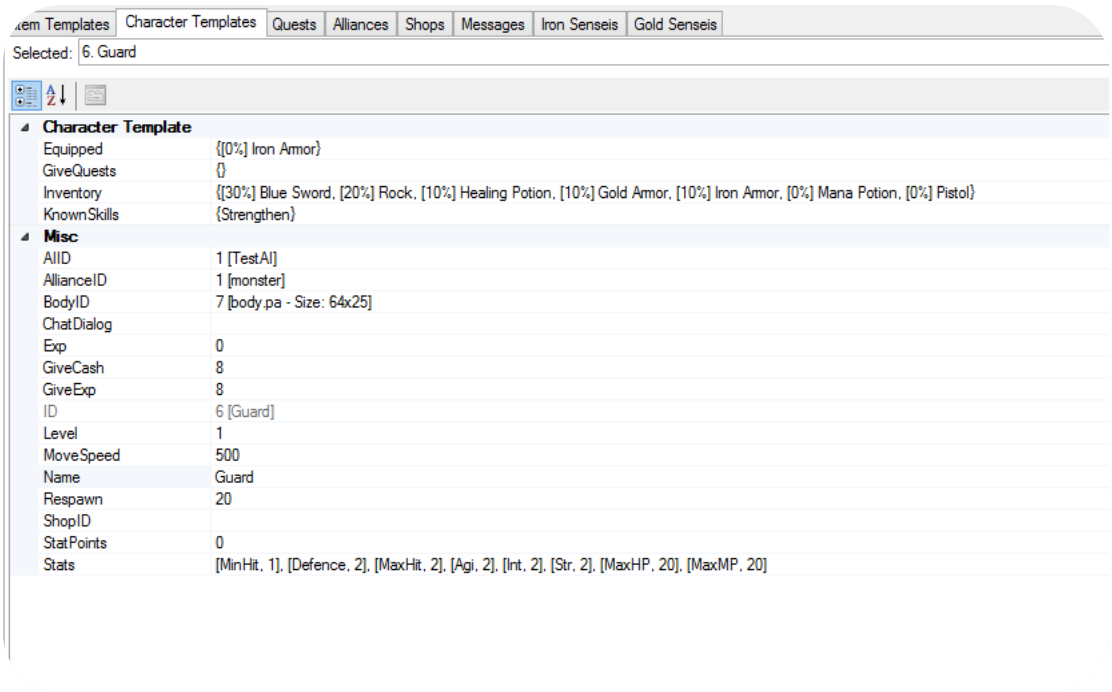

Figure C-2: Setting up character templates

Similarly, we define the templates for the training character, namely the Senseis, the Iron Senseis and the Gold Senseis. The following figure shows that based on the setting up parameters we have entered, when the user interacts with this Sensei, the dialogue for functions will start.

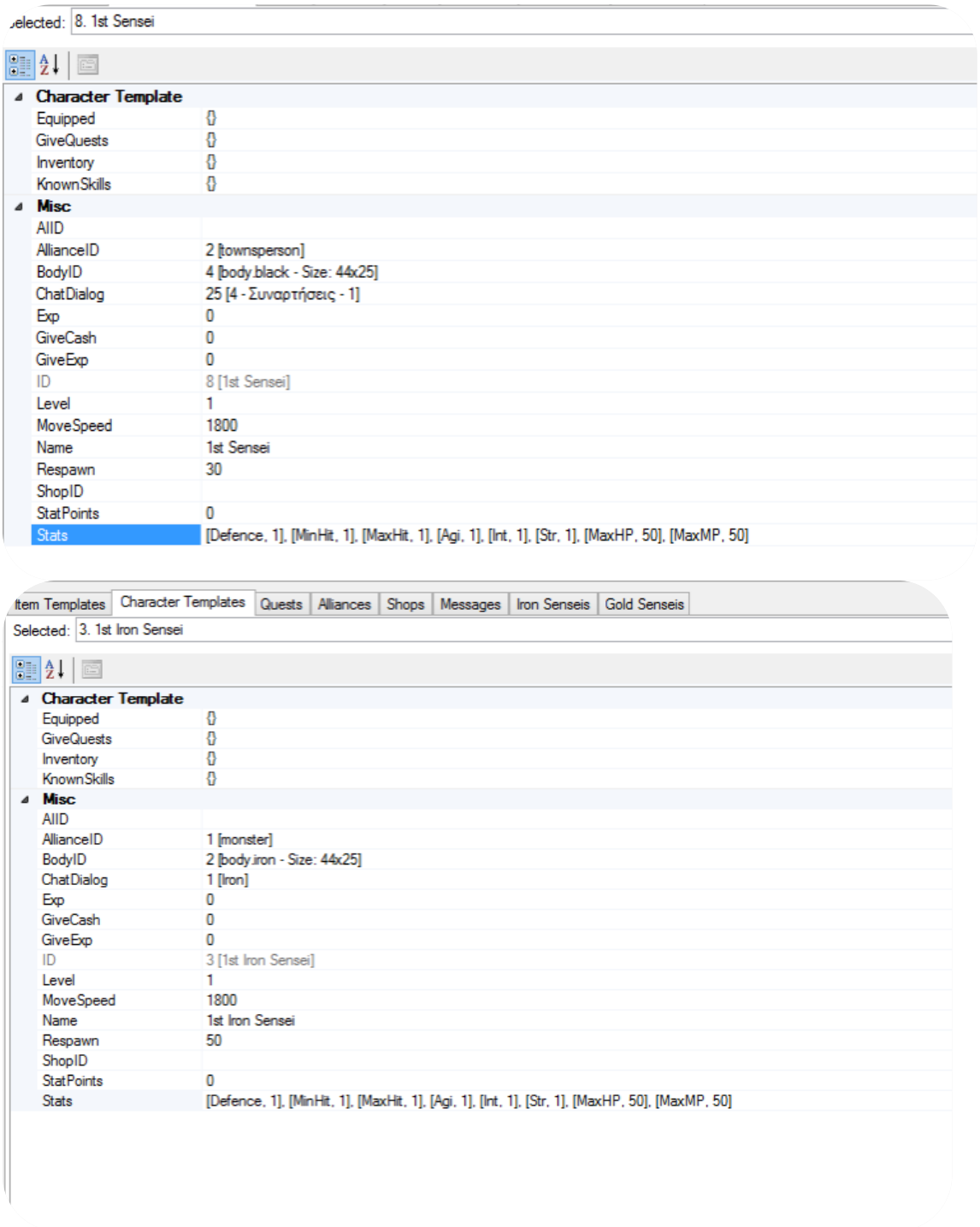

Figure C-3: Configuring Senseis characters

C. Setting the dialogues with the Senseis

Once we have created the templates for the game's characters, we create the dialogues for the Senseis. For this process, we open the Sensei's Dialogue Editor and start configuring the different nodes and respective arrows and creating a state diagram.

In the example shown in Figure C-4, we add four Senseis and thus create four different dialogues.

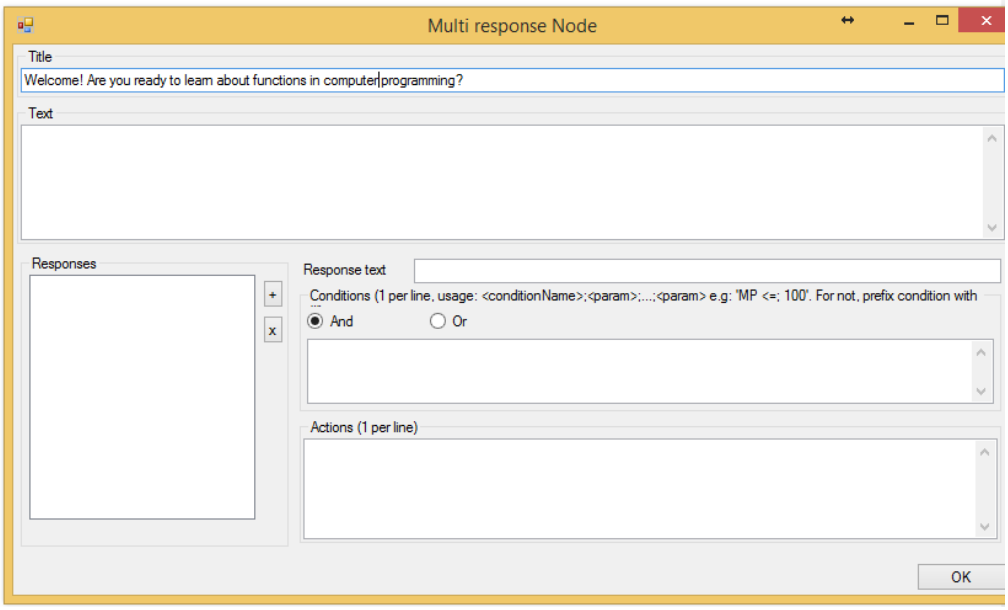

Figure C-4: Setting up dialogues for Senseis

D. Setting the activities with the Iron and the Gold Senseis

We move on to define activities for the remaining two levels of Senseis, namely the Iron and Gold Senseis. For example, we define four activities for the Iron Senseis, and providing the secret password that unlocks each activity. The password is the one users have retrieved from the first level of Senseis. Following, we insert the description of the activity and the required answer that is the solution of the activity with the commands of the program in the correct order, as shown in Figure C-5. Students are presented with those commands in a random order and have to arrange them so that they make an executable code.

The activities we include for the Iron Senseis are as follows:

1. Create a function that transforms the number of a month (from 1 to 12) to the count of days that the specific month has, so that it can be used in a program handling calendars. The months should be checked in an ascending order.
2. Create the following functions that will calculate the sum(add), the difference (sub), the result of multiplication (mult) and the integer quotient (divd) of

two integer numbers. The implementation of the functions should be done with order specific above.

3. Create a program where the function main() will call the function total() and will show her results for the value 100. The function total() will accept a number and will return as value the sum of the numbers from 1 until the inputted variable.
4. Create a function with the name Valid_Time which accepts 3 integer numbers in a row that correspond to hours, minutes and seconds and will return TRUE or FALSE depending on whether that time is acceptable or not.

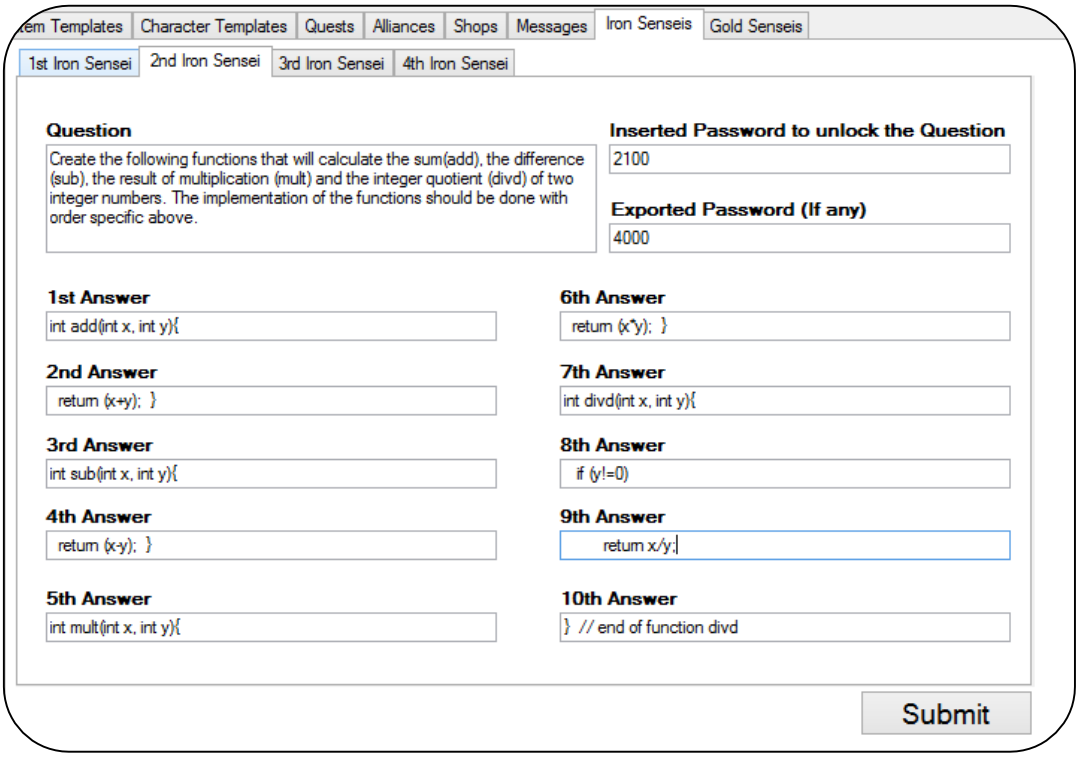

Figure C-5: Setting the dialogues for Iron Senseis

Similarly, we set up the dialogues for each Gold Sensei using the corresponding tab, as shown in Figure C-6. More specifically, we describe the two problems that te students have to write programs for and define the respective testcase examples and the expected outcome from each testcase.

The activities we include for the Gold Senseis are as follows:

1. An Internet café charges its customers with the following process: Minimum charge: 2€ for each game where the user stays until one hour. For each extra hour the charge is 3€. The maximum charge per customer cannot be more than 10€. Write a program with a function calc_charge(), which will accept the time a customer spends at the Internet café (integer value) and will return the charge in Euros.
2. Create a function that will calculate the total cost after a discount is given for the purchase of products. The main program will accept the count of products bought and their price. If the total cost is lower than 100€, then the discount will be 15€, if the total cost is lower than 400€ then the discount will be 20€, whereas if the total cost is higher than 400€ the discount will be 25€.

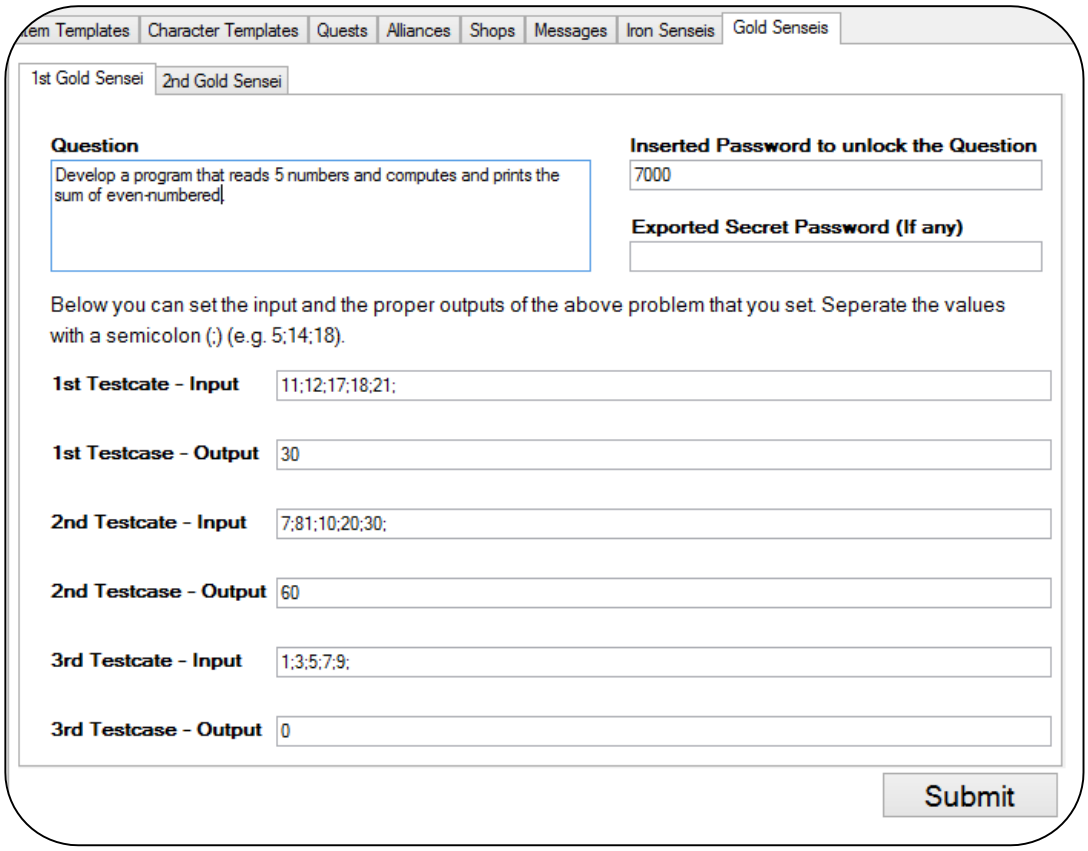

Figure C-6: Setting the dialogues for Gold Senseis

Once we have set up all the templates for all the characters that will participate in the version of the game we have designed, we proceed to add instances of the characters within the environment.