FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Game design for a serious game to help learn programming

Enrique Kato Romo

Master in Multimedia

Supervisor:
António Fernando Vasconcelos Cunha Castro Coelho

July 29 2011

Game design for a serious game to help learn programming

Enrique Kato Romo

Master in Multimedia

Approved in oral examination by the committee:

Chair: Eurico Carrapatoso
External Examiner: Leonel Caseiro Morgado
Supervisor: António Fernando Coelho

23rd September, 2011

Abstract

Serious games are digital games that use technologies and characteristics of video games with other main purposes than to entertain, being education one of the most important. Computer science majors require students to have good programming skills and several projects involving games are being created to tackle this problem in a more entertaining and attractive way. One concern among researchers is that there is not a specific methodology to create serious games for education, since serious games have to be designed for specific users and contexts. This thesis is focused on the creation of a game design for a serious game on programming that can be used in courses of computer science at the University of Porto, which places particular specifications to be met with the project. To achieve this, tasks started with the analysis of the scenario and requirements, followed by the creation of a game concept that could cope with the objectives of the project. The final product, a prototype, demonstrates that it is possible to create a game design and a platform to develop a game that is independent of a specific programming language, making it editable for programming courses that are based on distinct programming languages. The game design was done following the general guidelines that commercial video games usually follow, with the academic activities attached to it since the conception of the gameplay. A detailed design of the game mechanics was done to set the foundation of the gameplay of the serious game and the platform that supports it. At the end a prototype with a single level was generated using the mechanics of programming assignments and tested with students of informatics, receiving positive feedback about the game, finding that is attractive enough to motivate students into programming. Future developments are expected as the game design is an essential version of the final game idea but there are some details that need a redesign and further testing on distinct courses. However the results are promising to continue development of the serious game that would become a successful matured tool that helps in learning programming.

Resumo

Jogos sérios são os jogos digitais que utilizam tecnologias e características dos jogos de vídeo tendo como principais finalidades outras que não o entretenimento, sendo a educação uma das mais importantes. Os cursos de informática exigem que os alunos tenham bons conhecimentos de programação e é assim que vários projectos envolvendo jogos estão a ser criados para resolver este problema de uma forma mais divertida e atractiva. Uma preocupação entre os investigadores é que não há uma metodologia específica para criar jogos sérios para a educação, tendo que estes têm de ser desenvolvidos para utilizadores e contextos específicos. Esta dissertação está focada em criar um game design para um jogo sério de programação que posas ser utilizado em disciplinas de informática da Universidade do Porto, que coloca especificações particulares a serem cumpridas no projecto. Para conseguir isso, foram iniciadas tarefas como a análise do dos requisitos e o desenvolvimento de cenários, seguido-se a criação de um conceito de jogo que pudesse lidar com os objectivos do projecto. Como resultado desta dissertação, foi também desenvolvido um protótipo para demonstrar que é possível criar um design de jogo e uma plataforma para desenvolver um jogo que seja independente da linguagem de programação e personalizável para outros cursos de programação. O design do jogo foi feito seguindo as directrizes gerais que os jogos de vídeo comerciais normalmente seguem, com as actividades académicas ligadas a ele desde a concepção da jogabilidade, como os investigadores propõem. Desenvolveu-se um projecto detalhado da mecânica do jogo para definir a base da jogabilidade do jogo sério e a plataforma que o suporta. No final, o protótipo desenvolvido foi utilizado e testado com estudantes de informática, recebendo um *feedback* positivo sobre o jogo, sendo opinião geral que é atractivo o suficiente para motivar os alunos a programar. É esperado o desenvolvimento futuro deste trabalho, pois embora o game design seja uma essencial para a ideia do jogo final há alguns detalhes que podem ser melhorados com testes mais exaustivos e em diferentes disciplinas. No entanto os resultados são promissores para continuar o desenvolvimento do jogo sério por forma a tornaá-la uma ferramenta madura que promova o sucesso na aprendizagem da programação.

.

Acknowledgements

“A mind that is stretched by a new experience can never go back to its old dimensions.” Oliver Wendell Holmes, Jr. This research project was the last huge step in this marvelous experience. It would not have been possible to complete this last step, one inch at a time, without the support of many people.

I would like to express my love and gratitude to my parents, my brother and sister, who gave unconditional counsel and support during all this experience. I know that even in the distance they are always walking with me.

Very special thanks to my friend Francisco Lopez for his will to help improving my work and to my friend Marcos Anguiano who aided me during the creative process. I thank Denisse Iglesias and Margarita Medina who became my family in this experience, and shared with me the adventure of finishing our projects. I thank the colleagues of my master, who shared the same experiences, especially to Evi Dimakopoulou, for the shared failures and successes along the courses. Also I am grateful to my flat mates that made this experience a really funny one.

Special thanks to my project colleague, Ricardo Gonçalves, for his patience on guiding me into the possibilities of the technological platform and for his ideas for the game concept. Also I thank João Xavier for his help on writing the paper of this project and solving our doubts about the code reviewing component of this project.

Finally I would like to thank my supervisor, Professor António Fernando Coelho, for his helpful guidance, his readiness to solve my doubts and the freedom that he gave me to work on my designs. Without his help I would not have finished successfully this project.

To all those great people that believed in me and supported me through this experience, thank you.

Enrique Kato

Table of contents

List of Figures

List of Tables

List of Abbreviations

SG	Serious game
FEUP	Faculty of Engineering of the University of Porto
GCD	Game Concept Document
GDD	Game design document
UI	User interface
HUD	Head-up-display
NPC	Non-playable-characters

Chapter 1 Introduction

Games have been used for other purposes than entertainment since ancient times. Uses on battle tactics, training and education created several of the traditional games that exist today. This is also happening with digital games and media.

Nowadays the video game industry is one of the biggest entertainment industries. This is seen as an opportunity to use video games on other areas and industries. Since the appearance of digital games, training and education have been the major areas in which games have been used besides entertainment. There are several features of video games that are attractive to players and these are the features that are being used to attract students into activities that otherwise appear as unexciting, difficult to understand or risky to do in the real world. This is the point where edutainment and educational games were created to educate on several subjects.

Educational games involve new research on how to use instructional design to create games that are educative and at the same time attractive to students. Researchers agree that pedagogy has to be embedded into game design if the final product is intended to be successful on education. If a game design does not include instructional design, the outcome could be a fun game without any educational value.

This is also happening on the computer science areas, where game creating practices and serious games are being used to improve learning and tackle the problems of lack of motivation. Research done and the projects implemented show that there are positive reactions to educational games. Still more research is needed on the area of serious games, but the results are promising on the fact that video games are likely to be a great tool to drive education forward and even possibly to change the educational system.

This thesis is directed to create a the game design for a serious game that can help students to learn programming in a more attractive way that motivates them to continue their studies on computer sciences.

1.1 Problem Description

Getting into programming sometimes is seen as a daunting task. Many universities in the United States have seen a decline in demand on Computer Science majors. Researchers have found that one of the reasons is that the programming courses, the foundations of these majors, are seen as difficult, unattractive and boring by new students.

Video games are being used to attract students into subjects that are being seen as boring. This statement brings the next question. Is it possible to develop a serious game that works as an effective educational tool? What about to use a serious game to teach programming?

Research is being done to answer the previous questions. A lot of effort is being dedicated to build a methodology in which instructional design is present while designing serious games. Each of the projects deals with a different situation that has different needs to be fulfilled, making the creation these games a context sensitive process. The specific situation of the project treated in this document generate the question, is it possible the creation of a serious game to learn programming that is editable for other courses and subjects?

The challenge of this thesis is to create a game design that complies with the specifications of the project and serve as a base to develop a serious game that can solve the problem. A game that helps students to understand the fundamentals of programming while at the same time allowing them to enjoy the experience.

1.2 Objectives

The main objective of this project is to create a game design that can be used to create a serious game to learn programming. The game design should also cope with the concepts of instructional design and pedagogy to make the serious game successful as a learning tool.

To show that gaming software can be useful, and not be seen only as a time waster entertainment toy, as efforts are being done using video games as learning tools on other areas.

To prove that good game design can be applied to formal education system, trying to blend what makes the entertainment games so addictive while removing the “boring” brand of the educational games and software.

To check if the resulting prototype shows if the game mechanics can be used to make a game about programming more attractive to students and improve their motivation to continue studying programming.

1.3 Expected Results

At the end of this dissertation it is intended to obtain a game design, with game mechanics that allows the creation of serious game to learn programming fundamentals. This is a part of a project to use serious game inside the Faculty of Engineering of the University of Porto to aid students in their studies of informatics. This game design must comply with the specifications given by the coordinator of the programming fundamentals course to be used on the project. The game resulting will have as the main activity, code programming.

The game mechanics should have embedded the necessary instructional design to build a successful tool for learning. They must be open enough to be used on other more advanced programming courses as well as the one that introduces the fundamentals of programming. The game design has to work as a complement to the platform in which it will be based, be independent of a programming language and include the ability to review the codes automatically.

1.4 Thesis Structure

This document is structured in six chapters, being this introduction the first of them. The second chapter introduces the concepts of serious games and edutainment and how they are used in several areas for training and education., including some analysis on digital tools that use that concepts to tackle the task to teach programming concepts In that chapter is also analyzed the methodologies that are applied to serious games to include instructional design into them and build better tools that aid in education. Chapter 3 describes the specifications of the project that have to be followed by the work on this thesis. This chapter is followed by the description of the process to create the game concept, the game design and the game mechanics that are the base to develop the serious game in discussion. Chapter 5 describes the process of developing the prototype of the project and presents the results of the testing done to it. Finally the last chapter concludes this document giving a picture of the state of the project and the future works intended for it.

Chapter 2 State of the Art

This chapter presents the concepts of serious games and edutainment, followed by an analysis on the related work done in the domain of teaching programming with serious games. It will also be analyzed how instructional design is embedded into video games following some methodologies proposed by researchers.

2.1 Serious games

Society has used games as a learning tool in the past. “Non-digital games are used for learning social, physical, and psychological skills, for example, coping with the emotions after losing, learning the ground rules for appropriate behavior, and modeling the behavior of adults” (Ulicsak, 2010). Traditional games were used as tools for learning before media and digital games appeared. Games had serious themes such as divination, accounting of goods and battle strategies, as their purpose since ancient times. After game mechanics started developing, games began to have a set of rules to be used by the players; this brought more complexity to gaming.

Crawford proposes five main categories in which games may fall: board, card, athletic, children and computer games (Crawford et al., 2003), in all of which a serious objective can be added besides the entertainment one.

Board games consist of a board that represents a landscape in which we find a set of movable pieces that the players control. Early board games were used to represent armies on a field prepared to battle each other, capture the other player pieces or gain control of the board space. Old games like Chess, Checkers and Go, a Japanese strategy game, fall into this situation. The range of this type of games goes beyond the war-like idea, like Monopoly, Scrabble and Dungeons & Dragons[1]. There are also educational themes in games like Arthur Saves the Planet[2] and Cleopatra and the Society of Architects[3], or historical simulations in games like Through the Ages[4].

Card games are those in which a set of cards and the number and combinations of symbols represented on them are the main instruments of the game. Most of the games on this category use the traditional set of 52 cards making variations on the number of cards and symbols usage, like Solitaire or Poker. There are other games that use their own style of symbols and rules to be played like Hanafuda[5] or collective card sets such as the ones used in the game Magic: the Gathering[6].

In athletic games the player uses more skillful body movement and coordination than mental prowess. "The line of demarcation between games and competition illuminates one of the fundamental elements of all games" (Crawford et al., 2003), meaning that there should not be a confusion between athletic game and athletic competition. The author describes this difference as how the player interacts with other players.

Children games are the ones where a group of children joint together and employ a set of rules not entirely pre-defined to bring up a mechanic all can enjoy and participate. Games like hide-and-seek, kick the can, marbles or sardines are part of this category. The main activities in these games are social interaction and collaboration. At school years many of the games given by the teachers to the kids rely on an educational value, at different levels and on different areas of human development.

Computer games are games played on a computer, where the main director and the opponent is often the same computer. Games quickly became part of computers as programmers soon started to create games for that platform. There was an explosion of computer games when the informatics industry became commercially available. Later known as video games, the efforts of people enthusiastic about the idea of playing on computers, created an industry that nowadays surpass other media form of entertainment. From the early games in monochromatic screens that displayed few pixels or text, to the great complex simulations on a 3D environment, the gaming industry has struck on today's games, on both the entertainment and serious side.

[1] http://www.wizards.com/DND/Default.aspx
[2] http://www.freddistribution.com/control/rcn?p=arthur
[3] http://www.daysofwonder.com/cleopatra/en/
[4] http://www.eaglegames.net/ProductDetails.asp?ProductCode=CBG001
[5] http://www.hanafuda.com/
[6] http://www.wizards.com/Magic/TCG/Default.aspx

The main aspects of this form of gaming are the artificial intelligence that the player has to deal with and the eye-hand coordination that those games promote.

All these games can have a different direction than only to entertain, therefore they could be called serious games. The term is often used to describe all those games in the digital world, which are the computer games. For this thesis the term serious game will be applied to the computer games world.

A serious game (SG) can be defined as “a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives” (Zyda, 2005). The definition describes the areas in which a SG can be designed and developed, stating that video games can be used for other purposes than just leisure such as education, through health applications or raising interest about problems on the global world (Magnenat-Thalmann & Kasap, 2009).

A SG is not restricted to the video games environment; it can also be based on simulators since they also use video game technologies. Serious games (SGs) share characteristics of both domains as seen on figure 1 (Muratet, Torguet, Jessel, & Viallet, 2009). In fact a slight majority of experts conclude that there is a significant overlap between simulations and games in the training and education context (Sara De Freitas & Levene, 2004).

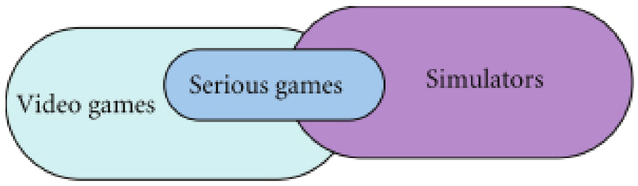


Figure 1. Relationship between SGs on the video games and simulators domains (Muratet et al., 2009).

As computers became powerful enough to manage huge amounts of information and have visual display of the outputs, the US military agencies saw the potential of including an entire war game in digital form, shifting the necessity of manual operation and calculations (Smith, 2009). The creation of such games were used for training in tactics on military institutions, and then applied to the instruction of other services, math models and the use of laboratories. When personal computers became commercially available, the once military exclusive games and simulations opened to entertainment, training in medical, civil and technical education as well as in global communications (Smith, 2009). As of the year 1970 people grew with arcade and console video games available to them, as those systems began to acquire market share above other forms of entertainment.

America's Army[7] is an example of a military game that became very popular among young people; however, its main purpose was to train and recruit soldiers, not to entertain. While it is being used for recruiting and training there is concern that the main attraction of the game is the graphics, rather than the accurate representation of the battle field (Smith, 2009). Training cannot be done entirely on a game or simulation, since full human behavior is still difficult to simulate. There are not any "trainerless trainers", so training still needs a human intervention (Chatham, 2007). Despite this fact, the game has been a great recruiting and training tool with such popularity that the government of China is developing its own game following the United States design, called Mission of Honor (Cifaldi, 2011).

While advances in warfare and military training drive several technologies to development, such as the appearance of the first battle instruction simulators that were game-like systems, other industries and scientific areas also place important interest on what a SG may bring. Simulators partake in other training areas such as medicine, vehicle or equipment use, corporate roles and communications among others, typically characterizing simulations by the roles or responsibilities the gamer takes in a specific context (Charsky, 2010). But SGs are not just simple simulations because they have the power to immerse players into a different world in which they can find themselves in situations they would have to deal on a real job, face challenges, and develop mental and intellectual skills needed to enhance performance (Marfisi-Schottman, George, & Tarpin-Bernard, 2010).

[7] http://www.americasarmy.com/aa3.php

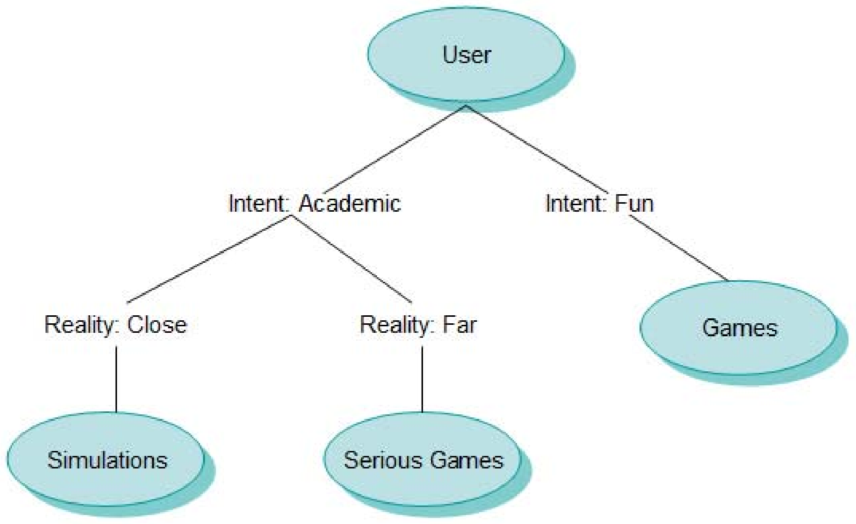


Figure 2. Hierarchical classification of games and simulations based on intent and reality. (Johnston & Whitehead, 2009)

Depending the closeness to reality, specifically on behavior of systems and information not graphics, and the final intent of the player, “a blurry line of user intent separates the categories of games and training simulations” (Johnston & Whitehead, 2009). Figure 2 shows the classification related to intent and closeness to reality. For example Johnston states that Microsoft Flight Simulator[8] is a game based on a flight simulation engine, with nice graphics but a game in the end, whereas other simulators with weaker graphic engines may have stronger simulation engine bringing them closer to reality. Another good example of how intent can change classification of a game is the Civilization[9] series. The games main objective is to entertain. They rely on powerful systems to simulate nation development and it has been found that they may be useful in history courses to help students learn world history (Cifaldi, 2005).

Other sub genres that exist on the simulation games are construction and management, life, medical, vehicle and social simulators. Commercial games may have a simulation engine within them. EA’s SimCity[10] series and the free licensed *Lincity-NG*[11] are examples of construction and management simulations. Biological simulations (i.e. Creatures[12] from Millenium Interactive), god games (like Ubisoft’s From Dust[13]), and social themed games (i.e.

[8] http://www.microsoft.com/games/flightsimulatorx/
[9] http://www.civilization.com/
[10] http://www.simcity.ea.com
[11] http://lincity-ng.berlios.de/wiki/index.php/Main_Page
[12] http://creatures.wikia.com/wiki/Creatures_Wiki_Homepage
[13] http://from-dust.ubi.com/from-dust/en-GB/home/

The Sims[14] from EA) are also use simulation engines. Many of the SGs that rely on simulations, are projects developed on the GNU license by researchers or independent developers, and some of them use simulation engines like SimGear[15], a SourceForge.net project, to power games and simulations alike.

On the medicine field great efforts are done in research about using SGs to improve training and learning of medical procedures and lectures, while other medical projects focus on patient recovery and prevention of diseases. Computer simulations of surgical techniques are becoming more popular, and studies show that players tend to have better chirurgical skills with the use of them (Lynch, Aughwane, & Hammond, 2010). The authors found that there are still some problems with medical simulators like the fact that the field is not completely studied and the lack of tactile feedback which is very important; also the costs to build them as precise and complex as they need to be, represent further challenges. There is however considerable push to do research in this matter as shown in the Games for Health project where the staff is supporting "community, knowledge and business development efforts to use cutting-edge games and game technologies to improve health and health care" (Games for Health Project, 2004).

Other uses of SGs are human behavior changes, environment concern, art and marketing. Figure 3 shows the several approaches in which SGs may be used. They are becoming a massive tool for expression. During the Games for Change conference, Al Gore said that games have clearly arrived as a mass medium, mainly thanks to social gaming brought by social networks and mobile platforms (Leigh, 2011). It will just take small time for other industries to tackle this medium in other different ways.

[14] http://thesims.ea.com/en_us/home
[15] http://simgear.sourceforge.net/

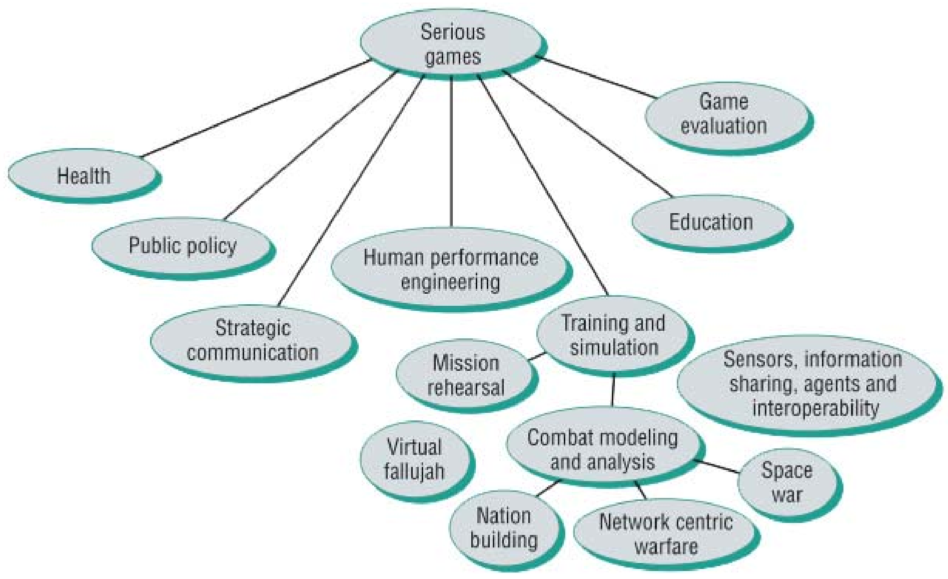


Figure 3. Areas in which Serious Games are being used (Zyda, 2005).

Even more unusual themes such as politics, meditation and gender issues are being targeted for developing SGs. Hidden Agenda[16], a political post-revolutionary game, September 12th [17] and The Redistricting Game[18] tackle political issues on the other side Queer Power[19], a fighting game, uses sexuality as the mechanic; all of the above have little or null leisure characteristics of commercial videogames (Ochalla, 2007).

While there is concern of the overall disbelief that has built in to SGs because of several failed attempts on educational games and the fact that they simply are not focused more on higher mainstream offerings (Ochalla, 2007), research has shown the possibilities of them are growing rapidly. A new ‘serious games movement’ is bringing together game developers and instructional designers to crack the problem of how to create effective serious games applications (De Freitas, 2006). Literature reviews show that education and training are the areas where most efforts are set. The next section is going to be described this is being done.

2.2 Edutainment and Serious games for learning

[16] http://en.wikipedia.org/wiki/Hidden_Agenda_(video_game) The game is no longer being published but a copy can be get if it is asked to the author.
[17] http://www.newsgaming.com/games/index12.htm
[18] http://www.redistrictinggame.org/
[19] http://www.molleindustria.org/en/queer-power

This section will describe the big area of development towards using SG and edutainment as tools to help on training and education. While SGs and edutainment have the same final objective, the pedagogical approach to learning is different between them. Edutainment became less important; on the other side research on educational use of SGs increases rapidly.

“Experiences with digital game natives, those who have grown up playing games, indicated that a game-centered research and educational program could offer many positive benefits” (Zyda, 2005). As there were big advances in the video game and the media industries, a bigger movement emerged towards SGs and edutainment. Researchers are beginning to theorize the cognitive processes during game playing, bringing a better understanding of educational game design and leading to new research (Dondlinger, 2007).

Edutainment and educational games were once praised to revolutionize the education system because their ability to educate and entertain, but they have received bad acceptance due to combining the lowest form of education, drill and practice, masked with boring gameplay (Charsky, 2010). These dynamics of edutainment are also apparent in the e-learning industry, because many of the products that are developed suffer from low budgets, mediocre contents and bad interface decisions (Egenfeldt-Nielsen, 2003) .

The lack of sophistication, poor depth of learning and boring game play offer a point of view from where is possible to analyze the problems and see how games can enable learning (Charsky, 2010). The review of the evolution of edutainment and instructional games shows what the problems are and the advances done through research in that area. The separation of the term edutainment from the research on SGs is necessary since it is not well accepted by scholars.

Edutainment is a broad term which covers the combination of educational and entertainment use on a variety of media platforms including computer games; they have a recognizable reward structure that is separate from the educational purpose (Egenfeldt-Nielsen, 2003). Frequently, edutainment games are those that work on the skill and drill format, in which players practice through repetitive skills and memorization (Dondlinger, 2007). The pedagogy of edutainment started relying on behaviorism (Ulicsak, 2010), on the first generations of the software. This format makes little or no attempt in teaching gamers how to apply knowledge, analyze their understanding, synthesize perceptions, or evaluate learning (Charsky, 2010).

Egenfeldt describes edutainment titles with a series of characteristics that underline the poor use of edutainment for real education; which does not mean that players do not learn anything; instead they do not acquire the intended cognitive skills:

- Extrinsic motivation – this kind of motivation is not related to the game or the learning objective but rather with simple rewards.

- No integrated learning experience – the educative content is just added as more information for the player, instead of being attached into gameplay.

- Drill-and-practice learning principles – the experience for the player relies on memorizing and training rather than understanding.

- Simple gameplay – compilations of mini-games, arcade games and simple adventure games are the examples used by edutainment software.

- No teacher presence – software does not require the presence of a tutor or a teacher assuming that the student learns directly from the screen without human interaction.

The other side of the characteristics stated by Egenfeldt is that edutainment games are easier to create and maintain than games that are more specialized or based on other cognitive paradigms therefore the previous characteristics can show which aspects of the game should be addressed when developing edutainment games.

Games and edutainment, even with the drill-and-practice format, make players learn something. Good video games are not a waste of time as players do learn when playing games (Gee, 2007); they are not useless on the education process, but they are not good teachers per se (Adams, 2005). It is when there is a good balance in game design and the instructional design that real learning happens.

Egenfeldt considers that edutainment is part of the first generation of educative software. It was followed by digital games based on constructive thinking, where the player is centre of attention and acquires knowledge as players analyze problems and apply paste learning as well as learning by making. Third generation software is where good educational SGs stand, using constructionism where learning is reinforced by having to explain it (Ulicsak, 2010). This generations and learning theories are shown on figure 4.

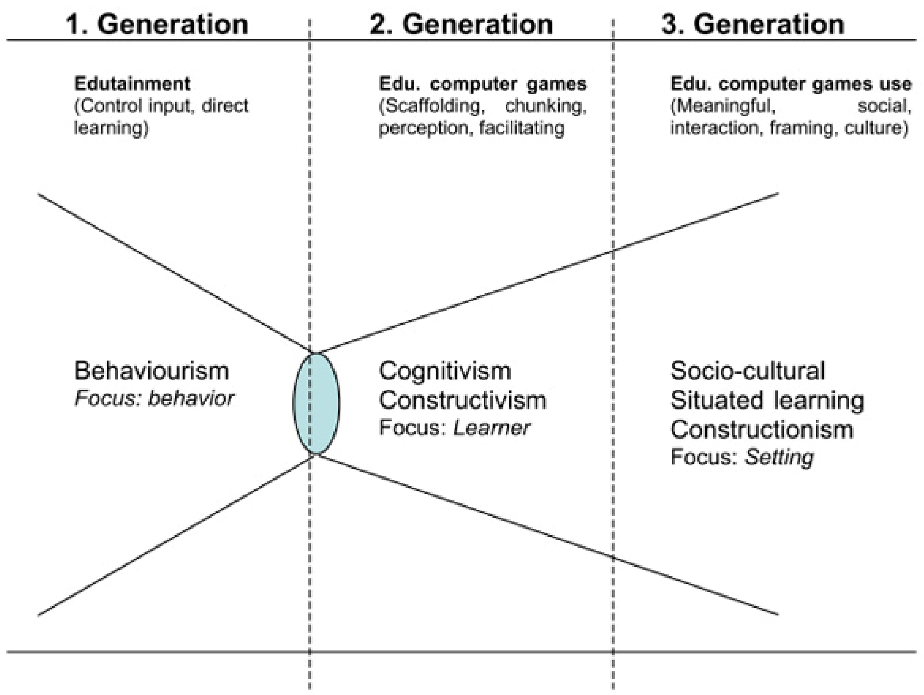


Figure 4. Generations of educative software and games (Egenfeldt-Nielsen, 2003).

Educational SGs make the learner strategize, test their hypothesis, and analyze to solve problems, which require better thinking rather than just memorization or repetition. They motivate the player with a system of rewards and a narrative context in which the activity is situated, and give opportune feedback to prompt learning (Dondlinger, 2007). Good designed games illustrate; they excel at bringing concrete experiences from abstract ideas (Adams, 2005).

Charsky denotes several game characteristics that help on defining a good game design, which later can be translated to a well constructed educational SG:

- Competition and goals – as opposed to edutainment, longer games with expanded win-lose victory conditions can get players involved into achieving a deeper success.

- Rules – simulation elements can be used to create a flexible rule structure, which instead of limiting possible outcomes, allows the player to brake or bend them to get new experiences.

- Expressive choice – choices that are made by the player that, while bringing little educative value, allow greater intrinsic motivation to continue and beat the game.

- Strategic choice – the player can make choices about game attributes like changing the difficulty level, use of in game coaching and debriefing.

- Tactical choice – player makes choices on how to play the game. The player formulates the tactics that he will follow, the paths that he will walk and the strategy to use in order to finish the game.

- Challenges – tasks and activities that are present on all educative software, but it is in SGs with good instructional design that the challenges are seamlessly included into gameplay

- Fidelity – a reality is represented by the use of graphics, audio, video, artificial intelligence and virtual worlds. Designing a convincing reality (not necessarily like the real-world) creates a context that is exciting and immersive.

- Context – authenticity of the game is enhanced by the setting, the narrative, story, scenario and characters that inhabit the game. An evolving and branching game script allows the player to develop different strategies and choices.

These are the characteristics that can be found on most of commercial videogames that offer complex experiences, feature beautifully rendered characters and landscapes (Bellotti, Berta, Gloria, & Primavera, 2009). However a SG is not only story, art and software, they also involve pedagogy. They are games because players play a specific role, carry on challenging tasks and are confronted by the consequences of their actions. They are serious because players need authentic knowledge and skills to address authentic problems and arrive to solutions.

To build a serious game the traditional game team must interact closely with scientists and cognitive experts (Zyda, 2005). Game designers and instructional designers must bridge the gap by identifying those game characteristics and establish a common vocabulary to tap the potential of SGs (Charsky, 2010). It is with this kind of interaction that the academic factor is integrated into the game design, and not just included as a different and extra portion added later to the game. The collaboration between teams tackles learning principles and pedagogy that have to be implemented into the gameplay of a SG.

"Teaching and learning is a matter of three things: the learner must be enticed to try, learner must be enticed to place in lots of effort even if he or she begins with little motivation to do so, and learner must achieve some meaningful success when he or she has expanded this effort" (Gee, 2007). When motivation, action and feedback improve player experiences, active learning happens. When strategic thinking brings new moves and strategies to confront the game as a complex system, then active learning transforms into critical learning.

Active and critical learning are the base of constructivism, the learning theory for educational video games. “The player is immersed in a world that enables them to include feelings and emotions with the social, the player can interact with fellow participants in the virtual environment as well as acquiring and using knowledge gained” (Ulicsak, 2010).

Constructionism is used when video games allow players to be active producers who can customize their own learning experiences, and not just be passive watchers (Gee, 2007). The constructionist approach to learning involves two activities: the construction of knowledge through experience and the creation of artifacts and assets that are meaningful to those creating them (Dondlinger, 2007), for example games where players can create functional objects in their virtual worlds.

SGs can be designed along with those learning principles to promote active and critical thinking. SGs can create a learning space in which the player takes risks and the consequences are diminished in contrast of the risks of real-world actions. This is psychological moratorium, a term devised by psychologist Eric Erickson, where there is a low cost of failure and high reward for success besides the time spent on the game (Gee, 2007). Games are a place where students can act and solve problems without worrying about the consequences of failing that a real world task has.

An assumption is that games do not teach by themselves, instead they help to understand (Adams, 2005), meaning that games are not at the same level as human teachers. Games are tools that help students to explore and understand issues, they train for various situations yet they rely on how they are interpreted (Ulicsak, 2010), and they require de support of a tutor or a teacher. In either way SGs should not be regarded as replacements of traditional learning but rather as a supplement (De Freitas & Levene, 2004).

There is still “little data on how games are used and how successfully they are integrated into teaching practice” (Ulicsak, 2010). But the change in perceptions about gaming brings greater willingness from game developers and educative experts to explore better and more effective game-based learning (De Freitas, 2006).

Games for education have different student targets, which range from primary school to undergraduate programs. They can be designed to help kids and grown up adults into learning subjects of different nature. Targeted public and subject mean that each SG has different design from the others, as the objectives are different in each case. Next section will detail the characteristics of some of the edutainment and SGs that are targeted to computer programming courses.

2.3 Serious games for learning programming

One of the several uses of educational serious games is focused towards the computer sciences. Research shows that there is effort on using serious games to help the student on the several courses that construct a computer science program. This section will review some of the projects that are aimed to the teaching of programming and their main characteristics as interactive software.

Programming, essentially the fundamentals, is hard to learn for novice students (Muratet et al., 2009). Students expect to see immediate results, even on early stages of learning while, real-world programming languages often require a lot of understanding, as well as a lot of work, before useful results are obtained (Anderson & McLoughlin, 2007). During studies immediate feedback while constructing their code allows an easier way for students to make iterative revisions and immediately see how their modifications in the code come along (Phelps, Egert, & Bierre, 2005). Video games and simulations can give this immediate feedback.

Researchers of the area have employed simulation techniques to visualize the results of students using a programming language and becoming familiar with the syntax and paradigms to code statements of a program (Jiau, Chen, & Ssu, 2009). Edutainment, simulations, software and serious games are being developed to help on programming learning.

It is very important to find the 'DNA' of a game, that is, the main gameplay characteristics and mechanics of it. This has to be done by developers when designing their game and when comparing and analyzing the ups and downs of the games of their competition. To better understand the functionalities that lie within various video games to learn programming, commercial and SGs alike, it is important to review their description and analyze their working game mechanics. The next subsections describe mechanics of the projects and software that was reviewed.

2.3.1 Learn to Program BASIC

It is edutainment software developed for PC and Mac that works as an interactive tutorial to learn to program in its own version of a BASIC interpreter. Published by Interplay, it presented BASIC language in a more user friendly interface and well built documentation of the instructions. In-game multimedia cinemas with two characters, “Media Man” and “Goo”, guide and help young people in their beginnings in programming[20]. The software, which is tailored to create games, has a library of images and sounds, as well as image handlers and video players at the disposal of the student.

The software is divided in three main sections: the lectures section, the project section and the freestyle section. The lections part is a step-by-step tutorial on how to use the instructions and functions of the language, guided and demonstrated by the two characters. The project section is where the student embarks into one big pre established objective and makes a program to attain that objective. The last section, the freestyle, is where the student is free to program anything within the abilities of the interpreter, mostly 2D games (Berg, 1999). The language used is not powerful enough to create 3D games.

This program uses the world of edutainment software and media to bring a package to teach programming that is easy to digest and attractive to young students, instead on focusing only on videogame mechanics,. The official site of the game has been removed from the Interplay servers and it is no longer available for download.

2.3.2 ALICE project

Carnegie Mellon University created a free 3D programming environment that as the same time that helps students learn programming lets them generate an animation with narrative[21]. Based on a version of object oriented language which syntax is very similar to C++ or Java, the software has several kinds of objects that the student can manipulate with code to generate a virtual world and animate it to tell a story. The environment allows the student to

[20] http://www.youtube.com/watch?v=0-BxxMp5BRE&feature=related
http://www.youtube.com/watch?v=iSOtdF-mDh0&NR=1

[21] http://www.alice.org/index.php?page=what_is_alice/what_is_alice

simply navigate on a library of objects and using drag and drop manipulate the virtual world and the methods and functions available for each object.

Although this software is not considered a game, the results gotten by testing it on actual courses give feedback about what attracts students to do programming. An interface that is easy to use and improves accessibility to the world of programming, a visual way to input code and review the result and the concept of creating something to share with others are the most important aspects of it.

The researchers and creators of the project aim their studies to find if that programming with a significant purpose, specifically any visual result that shows the output of the code, students would be more attracted to start and maintain their studies in Computer Science. A case study performed by the California Lutheran University did find that the package generates attraction to programming for students in a pre computer science course, but their results show that the program alone is not enough to form students into the computer science major (Klassen, 2006).

There were some drawbacks found by the case study. Students were spending more time focusing on creating a great picture instead of understanding the programming tools they are using. For other students, with some working experience, the concern was that it does not cover real world programming problems (Klassen, 2006).

2.3.3 MUPPETS system

MUPPETS, developed at the Rochester Institute of Technology, stands for Multi User Programming Pedagogy for Enhancing Traditional Study (Phelps et al., 2005). MUPPETS is an interactive platform for learning object oriented design and programming, designed to support the construction of interactive worlds. The system is built completely on the core Java language, making the entire J2SE standard immediately usable, liberating the problems of creating a stand alone language. The system is not a game per se, but has important mechanics that allow the immediate feedback, interactivity and community integration that modern games have.

The platform works on a community of novice and senior students that collaborate on the object oriented programming tasks. It gives the students a functional integrated development environment, advance code creation tools and options to expose created objects to the

MUPPETS community. The developers are concerned about the gap that divides lower and upper division of students, thus the tool allows several form of interaction and communication to make the senior students role models and helpers to the novice.

Students work on an environment that simulates the kind of work that has to be done on real world programming. They have to generate stories with the objects that they create or were created by senior students, while as the same time they have to deal with object oriented programming concepts. The results that the authors found on testing their development on their student community, praise the qualities of the system. MUPPETS intends to integrate cognitive skills used on computer education and promote active and constructivist learning.

2.3.4 Wu's Castle

This game is a project developed at the University of North Carolina under the "Games2Learn" project (Eagle & Barnes, 2009). Concepts of programming are set in an interactive environment in which graphic feedback is given to the students. Undergraduate students have the task to develop small games based on computer concepts that are aimed for novice students. The developer took concepts as arrays and loops, seen as difficult to comprehend by new students, and designed a game to teach about them.

The game is a RPG and was developed in RPG Maker 2XP, in which players manipulate arrays by changing loop parameters and by controlling a character within the levels. The scenarios consist in small levels where the student can experiment with the parameters of loops and control arrays, nested loops and other traditional programming assignments. Those changes on the parameters are reflected on movements of on-screen characters and constant feedback after each attempt, which the authors find is very important to student performance. With this simple mechanic student experiment and learn about two important, but sometimes not so easy to understand, concepts about programming.

It is important to notice that the game focuses only in two main concepts and that is designed with that focus in mind, giving feedback and tips on those matters. The studies done by the authors of the game show that they had an improvement on the final results of first year students after playing the game, and show that those that spend more time with the game got better results.

2.3.5 The Meadow

Developed at the Bournemouth University as its own solution to programming learning, The Meadow is a virtual environment in which virtual beings exist and are controlled by instructions in C-Sheep, a version of the C language (Anderson & McLoughlin, 2007).

The aim of the game is to help students with programming fundamentals on the C language. The authors express that it can also work on graphics programming learning and to experience other languages and compilers. The game uses a 3D engine to create tridimensional visuals and motivate students to use it. On their study they found that novice students are in fact attracted to use a 3D environment, but that more advance users prefer a simpler version to save time.

The students have the task to control the entities living on the virtual world through programming commands using the C-Sheep mini language. The codes generated and compiled on The Meadow, can be compiled on external C/C++ compilers using the library contained on the C-Sheep language. This can help students to pass from a mini language environment to a real world language.

On the case study, it was found that the documentation available was limited and slight incompatibilities between compilers on different platforms. This raised the necessity to continue the work on the platform and create a better learning tool for students at the university.

2.3.6 Epsitec Games

The company has developed games that rely on the edutainment concept and bring fun while learning. Games can be downloaded as a free demo or as a paid package from the company webpage. The games Colobot and Ceebot, in all its versions, share similar mechanics between them but work with different programming languages and gameplay approach. The core mechanic is to program robots to do some jobs for you.

Colobot is a real time game running in a 3D world. The concept is to travel between different planets and solve several missions asked to the player, which range from gathering resources, building structures, commanding robots to perform tasks and defend the land. After starting up the software the players are asked to create their own profile and edit their avatar with a limited selection of features before starting to play. The game is divided by missions, each one of them is done in a different environment or level and it is introduced with a small cinematic that helps to take further the story of the adventure. The story helps in the immersion of the payer into the game and works as a hook to play till the end as the player tries to discover what happens next. There is an option to play specific challenges or scenarios with closed challenges with the objective to review some programming concepts.

The player can play most of the game characters via mouse and keyboard command but there is the option to set specific programmed functions into robots. In this way the robots can do tasks automatically while the player explores other parts of the terrain; while in first levels this can be seen as extra, in later levels it would be a needed action to achieve a multitasking playing. Functions created by the player can be named and saved on each robot created and used later as a fast command option. The game uses its own language that has a similar syntax as C++ and Java, and has its own set of instructions, variable types, and objects; then the game compiles the code and runs it for the player. There is documentation about the language and the gameplay mechanics used in the game, the code editor presents some highlighting on reserved words and limited compiled error messages.

CeeBot, on the other hand, is software specially designed to teach programming tasks while playing small and defined levels, and it is aimed to students from school level to college and university courses. The games are set in a similar concept as Colobot, in presentation and mechanic design, but with a challenge based design to cover programming concepts step by step, being less a full game and more an edutainment course. There is not a specific story or universe to play with; in exchange you get mission objectives that tell the player what to do, and what is happening.

Just as Colobot, the player uses the same method of command of the in game tokens. It uses a similar syntax as the object oriented languages, with its own robot programming language designed by Epsitec. Limited actions with the mouse doing point and click on objects and buttons. There is an extensive documentation of the language of the game along with help and tips available for the player.

CeeBot4 is the most complex of the packages offered by the company, offering student and teacher versions of the software, allowing professors to design and upload some challenges and use them to teach concepts in a visual way and the students can download and try them by themselves. It is designed as a more powerful tool for programming classes, serving as a platform for a small community inside the lessons.

The review of related work reveals patterns that help to further design game mechanics for a similar platform. Cognitive skills and learning techniques have to be designed along with the game mechanics, requiring different development processes than those seen on commercial video games. Next section describes the characteristics of some methodologies that focus on the design of SGs.

2.4 Methods to design Serious Games

The industry and studios that create games for entertainment have already produced several well established methods for their development. The same cannot be said about SGs, there is still discussion about which method is the best to integrate the instructional design with the game design, therefore is an area that is not yet strictly well documented (Mcmahon, 2009).

As seen on the related work, all the projects define different educational objectives that can be applied to their own learning environment. There is need to review some of the methodology used on case studies to understand the steps to develop a successful educative SG. This chapter shows the methods taken as an example to design the game design and the prototype for the project.

The analysis was done on case studies of a version of the DODDEL model enhanced for SGs (Mcmahon, 2009), a process toolkit and methodology developed and used by EMERGO (Nadolski et al., 2007), and a the actors involved on the engineering process of creating a SG presented by Marfisi (Marfisi-Schottman et al., 2010). A four dimensional framework for evaluating the use of a game for education (De Freitas & Oliver, 2006), details other important aspects to consider during game concept and design.

The DODDEL model, short of 'Document-oriented Design and Development of Experimental Learning', seeks to integrate instructional and game design through a document-oriented approach. It is aimed to guide novice instructional designers into the process of SGs design. The model goes from a broad design to detailed specifications; with multiple iterations and a series of document outputs at each stage of the process. A summarized model can be seen on Figure 5, depicting the main steps and their expected specifics of the analysis and design stages.

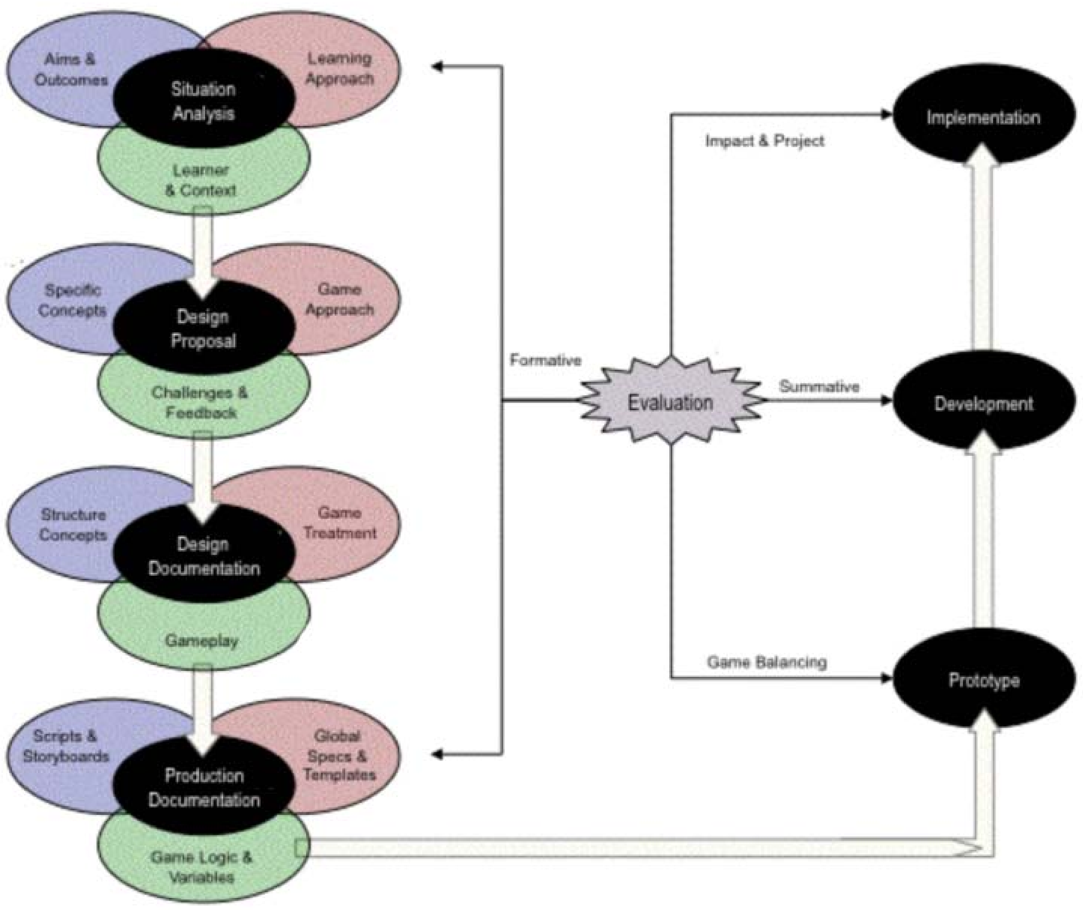


Figure 5. Summarized DODDEL model for Serious Game design course. (Mcmahon, 2009)

The rigorous documentation on the analysis stage let the designers create a feasible and complete analysis before going into the design stage. This is to avoid the jump directly into design without getting a complete concept analysis from novice designers. They find important that the designer team must know what they are dealing with on the educative context.

Since every academic environment is different, the analysis has to be articulated with a learning philosophy that is directed to the needed educational outcomes. Issues like end user experience, game treatment and learning outcomes, are treated at each stage, and defining them let the designers cross to the design stage with a better concept depending on the project.

The model was tested on a SGs course where students had to create a SG for learning. The findings of the author show that students found that the model is easy to understand and that the flow of the model works great into guiding them on each development step. While the initial findings are positive the model has yet to be tested in a commercial or industrial setting.

However some issues also arose during testing. Some students felt confused on what were the necessary outcomes of some of the analysis and design steps. Another issue found is that the model was perceived as limitless on the amount of detail, creating really complex documentation that would diminish resources on the developer teams. The author explains that those issues could be stronger on novice students without background on software engineering and education studies.

The EMERGO methodology is a methodology designed for developing scenario-based SGs and is based on the ADDIE method for designing instructional games. The ADDIE methodology stands for the five steps that form it: Analysis, Design, Development, Implementation and Evaluation; as seen on figure 6. The methodology is used on a specific toolkit that is also offered by EMERGO. This toolkit is formed by a series of widgets that serve for scenario design and testing, requiring that the right information is input at every step.

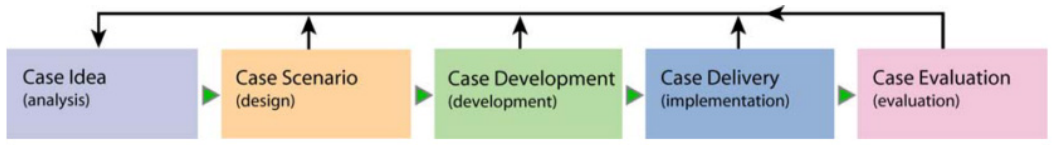


Figure 6. EMERGO’s methodology main steps based on the ADDIE method. (Nadolski et al., 2007)

The development steps are treated like cases, and each one of them works as an input document for the next case. As it can be seen on the previous image the black arrows denote iteration between the developing steps, though it is possible to follow one step after the other (van Zetten, 2010). It is at the Case Idea where the concept takes shape answering questions like: why is it needed, for whom is it meant, what will be in it, how will it be structured? It is also where the main pedagogic considerations take place since the learning principles have to be adapted to the target group, the case scenario and the overall objectives of the game.

The other step where pedagogical concerns exist is the Case Scenario where step designers deal with the learning objectives, the assessment instruments, content and challenges. The result is a detailed description of the activities in terms of tools and resources and how are the students going to carry them out.

The rest of the Cases have to deal with actual game design with some supervision of the pedagogical experts. It is also noted that some similarities with the software development process steps. The toolkit guides and let the developer to assess the results of each step through rigorous input of information and output results.

The model presented by Marfisi presents similar steps to the other two models, but with the added information about the actors that direct the team into those steps. The actors are the important part in the process proposed by the author. It shows how the academic or pedagogical actors interacting with the artistic directors and game designers develop a SG. It is through this collaboration that creates a successful product for education and entertainment. Figure 7 shows the steps of development and how the actors interact with each other.

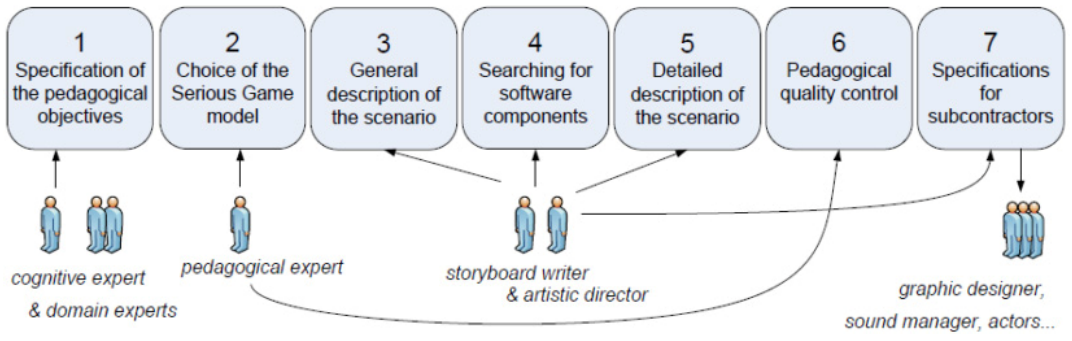

1
Specification of the pedagogical objectives
2
Choice of the Serious Game model
3
General description of the scenario
4
Searching for software components
5
Detailed description of the scenario
6
Pedagogical quality control
7
Specifications for subcontractors
cognitive expert
& domain experts
pedagogical expert
storyboard writer
& artistic director
graphic designer,
sound manager, actors...

Figure 7. Model steps and actors for instructional design proposed by Marfisi. (Marfisi-Schottman et al., 2010)

After the needs of the project are identified the project manager sets a series of tasks and distributes them to their respective members of the team. A cognitive specialist and pedagogical experts have to model the cognitive skills and organize them to form the pedagogical objectives of the SG.

To generate the game concept a writer and the artistic director have to create the scenario of the game, always staying in touch with the pedagogical expert to ensure that the entertainment part of the game gets along with the educative part. In this way the testing phases are minimized as the experts ensure that the several paths of the game design attain the pedagogical objectives. Once the game concept is done, the rest of the tasks involve the steps used on commercial game design. The work is passed to programmers and artists to commence development.

To have and efficient communication and collaboration between the members of the team is a common problem of the developer teams. To address this problems the author proposes an interface that is easy to use, even for the actors that are not computer experts. The interface is composed of a series of widgets that are accessed via drag and drop interactions. Those widgets cover several academic domains and game design domains, allowing the experts to prepare the documentation for the next steps and actors.

These methods present similarities with the processes used for software engineering. The iterative waterfall model shows that constant evaluation and testing is done through each step of software development. The three methods can be dissected into steps like analysis, design, documentation, development, implementation and evaluation; with the respective iteration to ensure a good product.

The framework proposed by Sara de Freitas requires the academic staff to consider four main dimensions before using SGs on the educational practice (Sara De Freitas & Oliver, 2006). While it is not intended to be a prescriptive method, it provides guidelines for the tutors to be more critical in how they use SGs on their courses. It works also as an effective design tool when analyzing and designing an educational game. The review on the concepts treated on each of the dimensions let the designers know what information is needed to generate a complete instructional design that can be applied to a SG.

Each dimension works iteratively with the others and not set to be working individually. The next list shows a brief description of the dimensions of the framework (Ulicsak, 2010).

- Context which covers where the learning occurs - it includes the macro level, so historical, political and economic factors through to micro, the tutor's background and experience, cost of game licenses etc.

- Learner specification, for the individual or group, requires the tutor to consider their preferred learning style and previous knowledge and what methods would best support them given their differing needs.

- Mode of representation, this includes the level of interactivity required, the fidelity, level of immersion produced. It also covers the separation of the immersion aspect with the reflection around the process of playing the game. Most importantly it highlights the potential of briefing and debriefing to reinforce the learning outcomes.

- Pedagogic principles require the tutor to reflect on the learning models which enables them to produce appropriate lesson plans.

Through the analysis of these methods it can be inferred that there is special focus on the analysis and design stages of development where the cognitive skills and pedagogical information has to be included. After the instructional design is finished the actual game design is developed. It is very important the continued communication and collaboration between pedagogical experts and game designers, as seen on figure 8, SGs use pedagogical design to infuse instruction to the game.

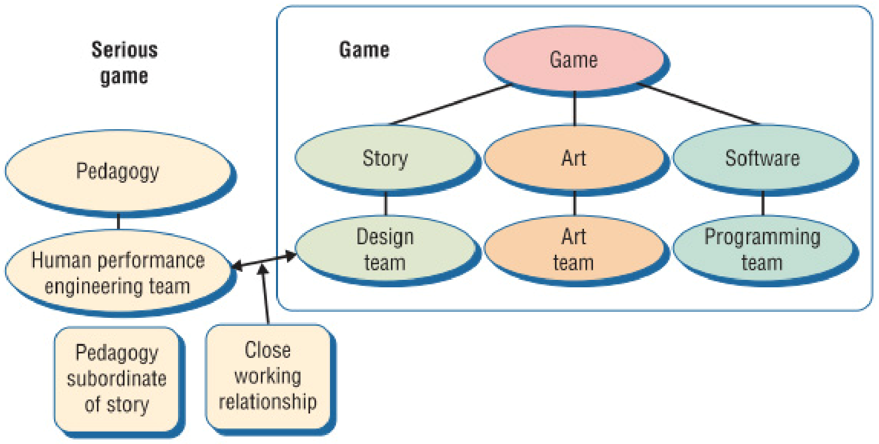

Serious game
Game
Game
Pedagogy
Story
Art
Software
Human performance engineering team
Design team
Art team
Programming team
Pedagogy subordinate of story
Close working relationship

Figure 8. It is shown where in the development process is the connection between educative SGs and video games (Zyda, 2005).

In her book Game Design: From Blue Sky to Green Light" (Todd, 2007), the author presents the general steps that the industry takes to create a game. The steps range from the game's initial conceptualization to game scripting storyboarding; from the functional part of the gameplay to the creation of the game design document and pitch it to a publisher. Specific steps and processes may differ from studio to studio, but the general steps are similar between them.

It would be operatively viable to use interdisciplinary groups to create SGs for high education as an in-house group of development instead of using generic commercial SGs (Rooney, O'Rourke, Burke, Mac Namee, & Igbrude, 2009), since those may not meet the objectives required by an specific situation. Other projects take advance programming courses students to develop games for first year students such as the project “Games2Learn” (Eagle & Barnes, 2009), changing the actors that Marfisi proposes.

As seen during this chapter, SGs can be used as a supplement to help improve normal educational programs. The design of the game has to embed into it the instructional design necessary to obtain the right results in learning a subject. A thoughtful analysis has to be done taking in consideration the target students, the presentation and the context in which the game will run, all of these depending on the needs of the situation. This brings versions of the game that not only may be a better iteration but that complies with one of the fundamentals of the project: a specialized game for any available course. This section closes the chapter of literature review. Next chapter will describe the needs of the specific scenario treated in this thesis.

Chapter 3 Project Specification

The previous chapter brought into account what are serious games, making special emphasis on the educative SGs. To design a successful SG for educations it is needed to perform a comprehensive review of the requisites of the scenario in which the SG will land. The analysis on these specifications is done to build a game and instructional design that comply with the target group and learning principles to which the project is aimed. This chapter will describe the specifications of the scenario in which is based the game design of the SG referred in this thesis.

The project consists in developing an educational SG about programming for undergraduate students. The SG is planned to work as a modifiable platform that complements studies of the programming subjects given at the Informatics degree at the Faculty of Engineering of the University of Porto (FEUP). The academic program of the programming course has to be covered in the game. The professor or tutor will decide how much of the course and what concepts will be included, as well as to make implementation of exercises and give instructional counsel to students.

Following the concepts about educational SGs, the design of this game must include the instructional design (programming concepts) into a game design that can be used and applied at the university.

The game should be built using game mechanics that mix the game design of commercial video games with the design of SGs to bring an effective tool for learning inside informatics academic area. "Students learn much more from actually solving problems, designing algorithms, and writing programs than from attending lectures and taking notes" (Huang, 2001), therefore coding and programming assignments should be the main part of the game mechanics that are the object of this research.

As a SG of educational nature, the project must focus on the instructional design combined with a game design rather than being based only on the common game design used in commercial video games. The entertainment aspect of a game design becomes an interlinked aspect of the project with the educational part that works as the main mechanics of the game. The leisure part comes to improve motivation and concepts understanding, as it is intended that the game make more enjoyable the practice of programming during the course.

The development of the concept should help to develop the main cognitive skills that are considered on the profile of a computer science student. The skills that are considered for a programming environment, which will be the case of study, are to be reinforced during the activities of the game, to ensure that true learning is achieved. Tasks inside the game must set players in an environment where they apply problem solving, planning, logical reasoning and analytical thinking. With the goal to promote active learning the problems of the game have to be designed with these cognitive abilities.

The following points describe the required specifications needed to be solved in the project:

- The game mechanics will be mostly about solving problems via code understanding and input, and then analyze the expected output. Playful mechanics seen in commercial games are to be included to aid in the immersion of the player and improve a continued use of the game, without monopolizing the instruction input mechanics.

- The project can support several programming languages, meaning that there is not a parser that is proprietary or embedded into the game. The parser will be upheld by an independent system already implemented at the University. In this way game will not struggle with different parsers either, it will only detect if the code is correct or not, but must understand the outcome of these solutions. The game mechanics should be designed to be able to sustain this openness of programming languages.

- The game mechanics must be designed to be able to work with a trial and error practices. Considering that during learning periods mistakes and errors are an expected outcome, the mechanics has to support this approach without penalizing the player. The design should ensure that the player keeps trying to solve the exercise, meaning that motivation is at hand and that the challenge does not compromise the advance of the student within the game.

- The game must cover the main concepts of the course. It is intended that the student that finishes determined levels or quests of the game, is able to fare better during class tests.

As said before, the general design of the game mechanics must be open enough to be able to cover other technology courses. Professors teaching other subjects such as networking, databases or advance programming, should be able to modify the game to adapt it to their courses. The design of the game should accomplish to be general and open to let further edition of the game, while keeping the flow and script of the game.

The game should act as a visual platform to understand and comprehend the concepts treated on informatics courses. Assets inside the game can demonstrate the outcome of working with some of the instructions defined by the professor. Game mechanics should allow registering advances made by students in a way both professors and students can review after play.

The whole architecture in which the game will be constructed should permit full or partial edition of the contents from the academic staff. Not only the game should be language independent, but also allow adaptation of new level designs, objects, quests and evaluation and scoring methods. In that way the academic staff can take the game source, re-edit it and deploy a new game with new characteristics in educational matters, while containing the same essence of gameplay and game flow. Architecture in which the game is based is attached to the game mechanics; the design of one must stay inside the possibilities of the other. The architecture for the game also has to be adjustable to meet the objectives of other courses and the system that automatically reviews code that is used at the university.

All the elements described in this chapter serve as basis for constructing the game mechanics and game design needed for the project. The study case in which the prototype will be built on is the fundamental of programming class, meaning that the main function is correct code input and output review. The prototype build at the end of the project will be a demonstration of what a final product may work and look like. Initial tests on it will show information if the initial concept and mechanics of the game is well accepted by students.

Evaluation on the perception of the students is needed to declare that the proposed mechanic will be usable or not in future developments. In the next chapter it will be reviewed the methodology that was followed to create the game concept, an initial game design and the core mechanics in which the game will be based on.

Chapter 4 Scenario Design

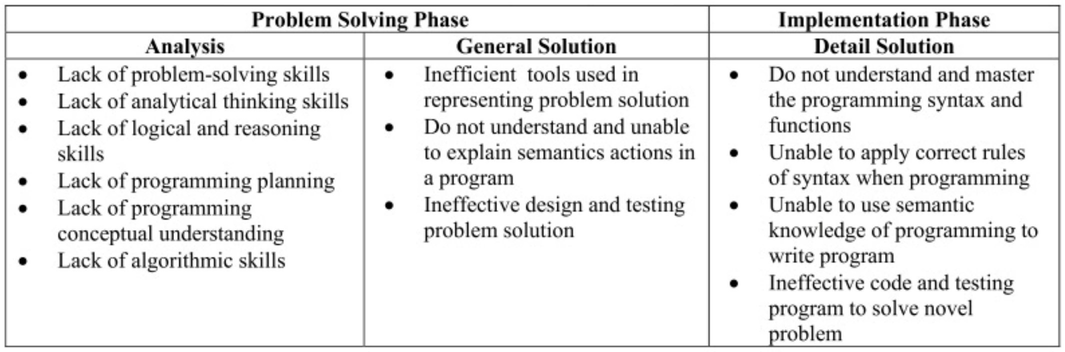
This chapter will cover the process to create the game mechanics and game design that follow the specifications stated en the previous chapter. The steps of development that are described in this chapter are: the game concept and preliminary game design, including the instructional design powering the game. Game mechanics is a fundamental part of the game design so there is a section focusing on the game mechanics designed specifically for the SG. Finally there is a section explaining the steps of the production stage of game development.

4.1 Game Concept

This section will explain the road through the Analysis Stage on how the general game concept was built and which are the outcomes obtained from this step. The specifications of the project described before are the result of the objectives presented by the professor while explaining the problem to solve. The design of the game and the mechanics working for it had to be under these conditions to try to attain the best possible result and develop an innovative tool for the informatics courses.

The main outcomes of this developing stage are the learning approach and a game concept document. Filling selected questions proposed by the EMERGO methodology helps to settle the global scenario of the project, its information and how should it be handled as an academic resource. Lose ends on the conceptualization of the academic idea and objective of the game can be traced through this task, building a better understanding of the learning objectives that have to be reached.

Literature review reveals the main common problems that student have during the fundamentals of programming courses. Experts found that the main problems that the students have to deal and overcome in order to become better programmers along the computer major (ISMAIL, Azilah, & Naufal, 2010). Those problems are showed at table 1. They categorize the problems in four types: lack of skills, ineffective representation of problems, ineffective use of teaching strategies and problems understanding syntax and constructs. The learning approach of the game is intended to target at some degree the solution of these problems.



Problem Solving Phase		Implementation Phase
Analysis	**General Solution**	**Detail Solution**
• Lack of problem-solving skills • Lack of analytical thinking skills • Lack of logical and reasoning skills • Lack of programming planning • Lack of programming conceptual understanding • Lack of algorithmic skills	• Inefficient tools used in representing problem solution • Do not understand and unable to explain semantics actions in a program • Ineffective design and testing problem solution	• Do not understand and master the programming syntax and functions • Unable to apply correct rules of syntax when programming • Unable to use semantic knowledge of programming to write program • Ineffective code and testing program to solve novel problem

Table 1. Main problems found on beginner programming students. (Ismail et al., 2010)

For the learning approach of the game it was considered that practical courses for engineers in informatics are of extreme importance since it is with these practices that students develop problem solving skills that will help them in their future careers (Djurovic, 2010). Practical problems would be the main learning approach decided for the game, in order to develop the problem solving skills of the students. So it was decided that the game would be based on a series of quests that will represent the materials of the academic course. Those quests would be increasing in difficulty as the player completes each area or level as most of the programming concepts seen on later subjects need skills developed on earlier themes.

The SGs medium promotes a visual representation of the problems and possible solutions. It is important that the students change from the initial idea of ‘what’ and ‘how’, that is taught in many introductory programming courses, into a ‘when’ and ‘why’ (Ismail et al., 2010). Students are required to develop the problem solving skills and analytical skills needed to be successful in programming. So practice is important but also, how the game helps them to acquire those skills is important, this is when presentation of the concepts comes at hand.

A visual representation of the theory concepts that construct the course would help on the understanding of the course, and they could be implemented either as visual changes on the game environment or animations depicting those concepts. Feedback is of extreme importance in such cases, and visual feedback brings even more impact into understanding the situation that the students are into. Then it was considered that if the students can watch a visual representation of a concept and how several actions can interact with it, they will be able to abstract that kind of information and have better understanding of such concept.

There is a concern with this approach and the openness desired for the project. To design the levels and a quest system that are linked to the actual representation of what the concept is trying to explain will take a great effort from the designer or design team. Academic staff would have to spend time and resources to design the levels for a version of the game. There must be a synopsis between the assets and their connection between them with the theoretical concepts that are to be taught.

As for the leisure part of the system, the game type, the genre, the setting and the initial story was decided during the first brainstorming sessions. Defining the game was necessary to start development of the technological architecture in which the game would be mounted and continue with the game design in a proper form. This will be a computer game so the platform of choice is either personal computers or laptops.

The pace of the game should be slow enough to let students analyze the situation and come with a solution to the tasks required, so a mixture between a puzzle and adventure genre would fit better with the need. Adventure games lead players into exploration activities to discover what lies within the story and world around them. Puzzle games are mainly constructed with problems that require strategic solutions. The main mechanics of this genre is to solve a group of puzzles and conquer the challenge that is imposed to the player. A mixture of these two genres could work better for the type of experience in which the project is aimed.

The game will be a single player experience. The advantages for this mode rely on the abilities and resources that the staff has to develop the game. While a multiplayer version was not totally discarded, the initial concept would deal only with a single player experience. In her book, Jeannie Novak talks about the differences between generations of players. She describes the next generation of players as social and collaborative, preferring styles of multiplayer cooperation and interaction (Novak, 2008). This is an important aspect to consider in the design of new versions of the game targeted at the next generations to enter to the informatics course.

The type or style of game was decided to go into either a two dimensional game or a three dimensional one. Text based adventures was left out because there is intention from the design team to explain graphically what is going on with the course materials that the student is learning. Newcomers to programming feel more attracted to a 3D representation of the world than a 2D version of it, “as they are most likely to help with meeting the high expectations of the Plug and Play generation” (Anderson & McLoughlin, 2007). Therefore a 3D world was selected as the style of the game. Being in 3D does not mean that the game will be hyper realistic, since that will require huge amount of resources and time form the staff, instead a cartoony style was preferred.

A game setting and a general story was defined after debating several ideas between the participants of the project. The universe would be in a utopian future where the human beings are already ahead in the space age and search about civilizations on other worlds. This scenario applies to the mechanics of programming challenges since coding could be a good way to interact with these worlds. The name elected for the SG is ‘Project Orion’.

The actors of this step of development should be the professors that are on charge of the courses, “as the domain experts” (Marfisi-Schottman et al., 2010). The professor leader of the programming courses settled the specifications of the project, which has as a main objective to be applicable to other courses. If the game needs a new edition concerning other courses than the initial one that focus on the fundamentals of programming, the leaders of the courses must define how the specifications will be for the new version of the game. There will also be need of staff working as game designers should meet with the academic personnel to maintain the game design layout and build a better experience for the students.

The final output for this phase is a Game Concept Document (GCD), in which is resumed all the information for game developers get the main idea of the game. The GCD for Project Orion is located on the Annexes section. After a GCD is created and exposed to every member of the development team; the process continues with the actual design of the game. Next section will detail the process of game design followed for the SG.

4.2 Game Design

After the analysis of the specifications is done, the design process begins, focusing on the game model, the scenario in which the game will run and the design of the game mechanics. The design process starts from the general idea into the specifics, making further description of what the game will be. This section details what are the game design and the game design document (GDD) and how it links with the instructional design to construct the final SG.

Game design is the idea behind the game. It is the process that describes that idea by designing the rules, gameplay, story, environment and artistic assets that will construct the game. Game designers on the game industry conclude that game design could be explained as what makes a game 'fun' (Colayco, n d). The game designers are the main actors of the game design.

Game development is a work done by a team of designers, artists, programmers, musicians, among others; it is a multi disciplinary job. Game design as a step of game development has to involve all the areas in the development team even when the main actors are the game designers. It is the vision of the game designer that has to be communicated to the other player of the team, so they all set effort on understanding and achieving the same goal.

The documentation of the game design is done through the game design document. The GDD is used to organize the efforts of the development team, and while it is guided by the game designers, it is created by collaboration of artists and programmers as well. This document works also as a pathway to create a pitch document that sells the idea of the game to investors and publishers. The GDD should be a living document and accessible to all members of the team.

Since every game project is different there are no strictly defined methodologies to follow, though there are some tools that help into creating the game. In some cases the GDD is created as a team effort at various stages and then someone (game designer, level designer, writer, artist, producer) is asked to situate all information together (Todd, 2007). Then the document must be placed where it is accessible to read and modify as the production advances.

It is during the pre-production state where the game concept, the pitch document, the game design and a small prototype is done. If the pitch document (game proposal) is approved by the game publisher, then full scale development begins, and the game enters to production state. During this stage the GDD can change and grow as the development continues, that is why it is needed to be reachable and of easy lecture.

Game designers have their own ways to manage this document, which can become a volume of great dimensions, but a central database or a wiki to store them is often recommended as they are easy accessible and modifiable. Making the GDD into small useful segments or sections is another way to maintain control over it as it grows. Although a GDD does not remove the need for constant design meetings or electronic discussions; getting opinions and critiques on live about an idea is a faster way to reach team consensus (Ryan, 2010).

There are no exact guidelines of what a GDD should include since development companies have different ways of working and each project carries different objectives to reach. The document segments that the game design documents usually include are:

- Game Concept
    - Introduction
    - Genre
    - Platform
    - Game Description
    - Key Features

- Feasibility Study
    - Market Analysis
    - Financial Analysis

- Functional Specification
    - Game Mechanics and Gameplay Elements
    - User Interface
    - Art and Video and Concept Art
    - Sound and Music
    - Story
    - Characters
    - Level Requirements

- Software and System Architecture Specification
    - Development environment
    - Game code architecture
    - AI
    - Multiplayer (if applicable)
    - User interface

- Story and Character Bibles

- Level Design Document

- Flowcharts, scripts and storyboards

The GDD of Project Orion discusses the game at a pre-production state and will therefore have the necessary sections to describe the idea until that stage. Also with the entertainment factor, the game design has to include the necessary instructional design to create a successful experience on learning programming. The full initial GDD is handled as a second volume of this document, found in the digital version of the thesis. The next subsections refer to condensed explanations of the functional game design, the architecture in which the game will work and the activities that deal with the instructional design.

4.2.1 Functional Specification

The functional specification is the segment of the GDD that describes the overall gameplay mechanics of the game, the setting and how the player interacts with it. The game mechanics are defined on a separate section dedicated to them; therefore this section explains the rest of the essential points of the design of the game.

The interface allows the player to take control of the game; it is what makes the game interactive, compared to other media. The interface must be designed to meet the targeted players, their needs and choices, if not the game will be unplayable (Novak, 2008). The main mechanic of the game requires that the student has immediate access to a keyboard; therefore the best control interface to use is the combination of a keyboard and a mouse. There are no plans to support joystick control pads since there will not be an exhaustive use of movements from the player.

The user interface (UI) is designed to be a clean and simple one. To be in line with the story, gameplay and setting, the interface is designed as a style of the digital and space era. The UI must allow the player easy access to the menus and help system which essential to the gameplay since help and language documentation has to be at hand for the player. The start screen will give access to the student profile, and once inside the profile, the student will be able to start a new game or continue others that were saved before.

The head-up-display (HUD) should also be simple to detect and monitor; it will display information about the score of the level and the hotkeys to access the menus. The interface for the programming mechanic should be designed to be unobtrusive for the student, clean and simple as well. The following image shows the different user interfaces that are planned to be used in the game.

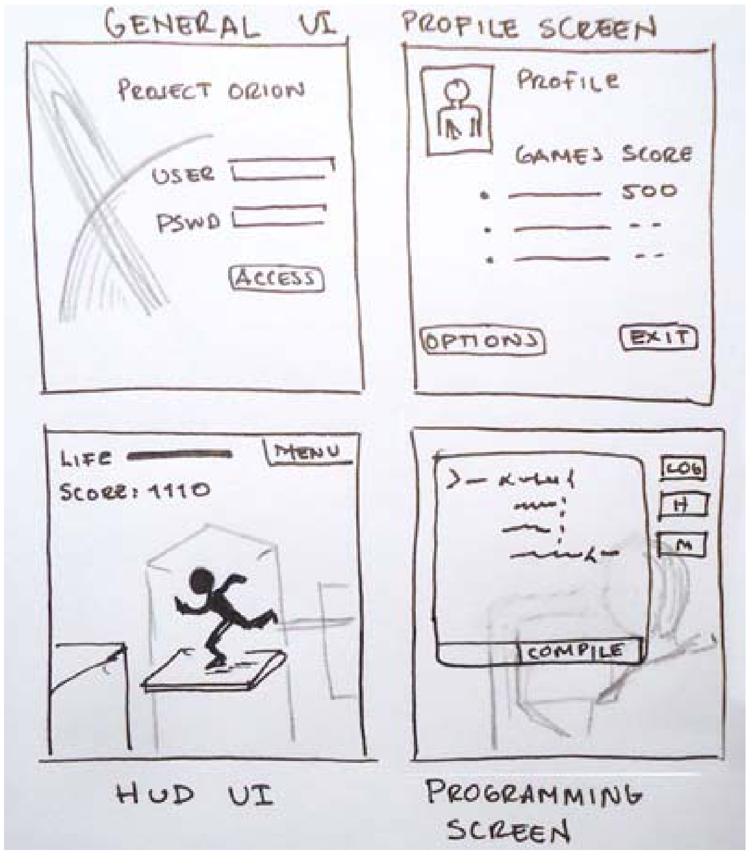


Figure 9. Initial concept of the UI for started screens, HUD and the programming screen.

Usability testing of the interface and navigation between screens is essential. The objective is to allow the student to interact with the game effectively, to keep the interaction simple. The testing has to be done during the game development process, where it can be tested without the distraction of aesthetics.

The next point in the functional specification is to create the vision of how the game will look like. Concept art is used on pre-production processes to set the ideas of the artists and designers on the art and style of the game. It is the guide for the visual aspects of the game. During the creation of the concept, the general consensus of the team was to use a mixed style between realistic and cartoony for the presentation of the game. This selected style is reflected on the concept art generated for the game. The following images show an example of presentation of the game.

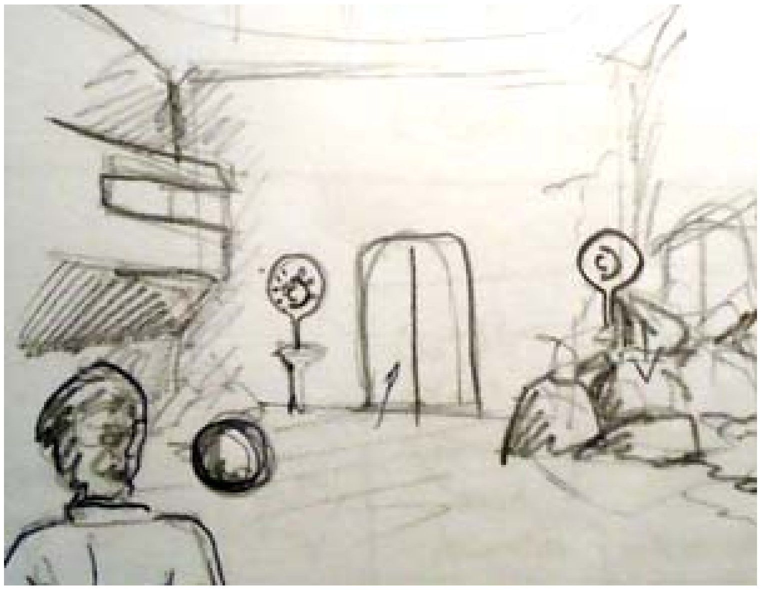


Figure 10. Quick draft for the overall concept of a level in the game

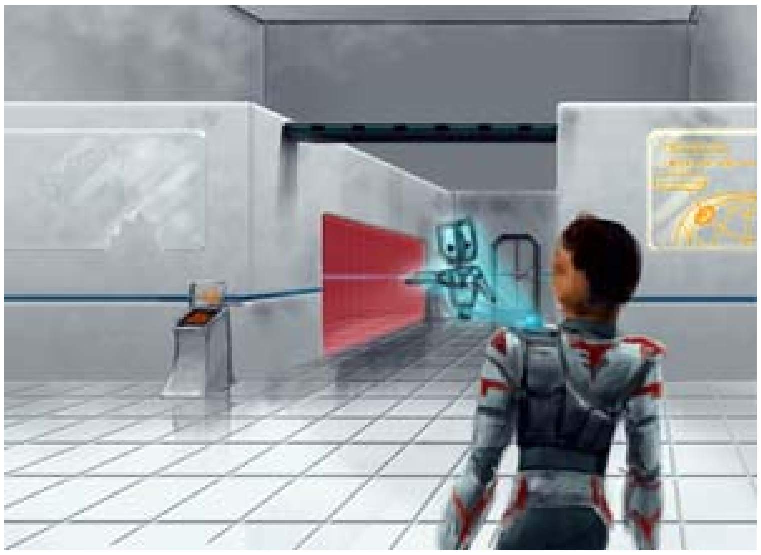


Figure 11. Concept art of a level of the game.

Concept art can be used to describe the setting and the general mood of the game. Artists generate studies on shapes and colors to get the results expected by the lead artists and designers. They also generate the look of the characters and non-playable-characters (NPC) that populate the game. After the initial drafts, detailed drawings are given to the production artists and modelers to construct the characters that will appear on the game.

Figure 12. Concept art for the ambient mood of the world.

Figure 13. Art used for storytelling purposes.

The decision of not including video or complex animation reels is because of the available resources of the academic staff. While they create more immersion and attraction of the game, they are considered unnecessary for the nature of the game. Instead it is better the use of simple animations to tell the story or explain concepts. There is one style of animation called animated stills, the use of still images with limited movement or frame animation. This style can do the job of representing ideas and are a lot quicker to produce than complex CGI or hand drawn animations.

Just as the visual presentation makes the game more attractive, sound and music brings up the immersion of the player into the game. Audio can be categorized in sound effects, voiceovers and music.

Sound effects bring realism to the virtual world, set the mood as well as change it. When audio design is correctly done, can provide gameplay cues (like achieving a milestone or objective), heighten player satisfaction and enhance quality of the game (Novak, 2008). Sound effects give feedback of what is happening.

Voiceovers are used in games for spoken dialogue and narration. As done with animation and cinema, it is usually an outsourced task. Since the narrative of the game is generally composed by sets of one or two lines and are often out of context, the audio director must ensure that the actors read the lines on the tone needed (Novak, 2008).

The music score that plays during gameplay sets the mood and pace of the game. For example action tittles usually have fast paced music, while game simulators have sooth and calm music for long plays. Music can be recorded, this is orchestrated with real instruments, or it can be synthesized with MIDI instruments (Novak, 2008).

It would be recommended to have a score of sound effects and music for the game. Voiceovers are not essentially needed. The lack of staff with the abilities to design quality audio for the game mean that at the moment audio is not included in the game design. However to power the initial version of the game, free or paid tracks and sound effects can be taken from Internet. Or the task could be handled to a sound engineer that can create content for the game.

The story will feed the game with a plot and a final objective for the player. While it is not needed on all game genres, having a good story adds immersion to the game and motivates the player to continue further. The premise of Project Orion is: '*As the space race helped the human kind to make the First Contact, a quest for knowledge and discovery of the galaxy started swiftly. Join to the team of explorers as a neo-archeologist to find and study the fate of other civilizations through the galaxy'*.

The elements of story besides the premise are: the back-story, synopsis, theme, and setting. The overall story of the game is powered by the plot, which tells how the story is unfolding. The plot requires that the conflict within the story is balanced; this speaks about the dramatic tension (Novak, 2008). To tell how the story develops, it is recommended to use the ‘and because…’ way of telling instead of the ‘and then…’ syndrome. In that way the plot becomes more interesting, because cause and effect are introduced into the story (Todd, 2007).

The overall story on Project Orion happens on the future of the human race. The player is sent to a planet where the ruins civilization that was very human-like is found. His task as a neo-archeologist is to discover what happened to that civilization and realize if the same could happen to humans or not. The plot deals with a conspiracy that threatens the expansion of human beings. While it is a peaceful era, there is still corruption and ambition, and the knowledge acquired by the character could send a warning to avoid repeat the future.

The story is tied to the advances of the player through the levels of the game, which brings the design into level requirements. The characters, the story and flowcharts (script) of the game should be written before entering to level design. To aid development it was decided that the plot bits that uncover the whole story should be placed between levels. Meaning that the player sees how the story is unfolding after completing all the obligatory rooms of a level. This is helpful because diminishes the effort of placing plot information during level design, and makes the story more manageable.

Characters can either be the player characters, controlled by the player, or non player characters, which are not controlled by the artificial intelligence (Novak, 2008). Game developers spend a significant amount of time designing their characters; they seek to give personality to the game and create a connection with the player (Todd, 2007). The character bible is the document that contains all the information about the characters of the game. The character bible deals with extensive descriptions, back story and sketches of the main, secondary and supportive characters.

Isaac is the name of the main character of Project Orion. He is a human being from Earth, in his late twenties, whose job is to study other civilizations found on the galaxy. He has a sidekick character called H4L that helps him in his duty. Isaac is controlled by the player; he can interact with the world in limited way. The robot, while is not directly controlled by the player, can be called at any time to aid Isaac on his tasks, as it is the one that interacts with the systems on the planet. The following image shows some concept art evolution of the main characters of the game.

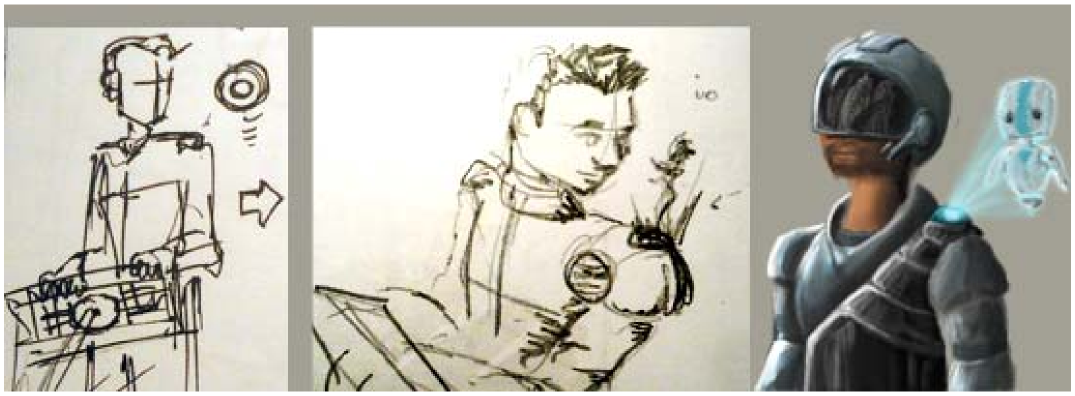


Figure 14. Evolution of the design of the main characters for Project Orion.

It is recommended to have information on the levels and puzzle design covered somewhere in the GDD, before the production stage of the game, because it saves a considerable amount of time (Todd, 2007). Puzzles in level design must always stay organic, ‘loving the player’ instead of ‘killing the player’, to avoid frustration.

In Project Orion, the levels match the material of the programming course, so it was decided to have rooms as the basic structure of the levels. Each of the rooms would handle a set of quests that the player must solve to advance. Those quests should be tied to a puzzle that has to be solved via programming and should also be aligned to the material assigned to the level.

The idea is to have a series of objects that together form a puzzle. The player solves the programming task and the trigger is sent to the objects to react according the programming concept that was tested. In this way the feedback of the code input comes as a visual demonstration, which will help the student understand the concept.

The option to choose different paths is going to be offered for the player, but the mechanics around this is the difficulty of the challenges. Difficulty selection is not going to be available form the main menu, but it can be managed in-game. Depending on the skills of the player, there is the possibility to improve skills on easier paths, if the player chooses so. If the player feels comfortable with his skills he has the chance to get a higher score on an optional way with harder quests. This multipath option between levels is depicted on the image below.

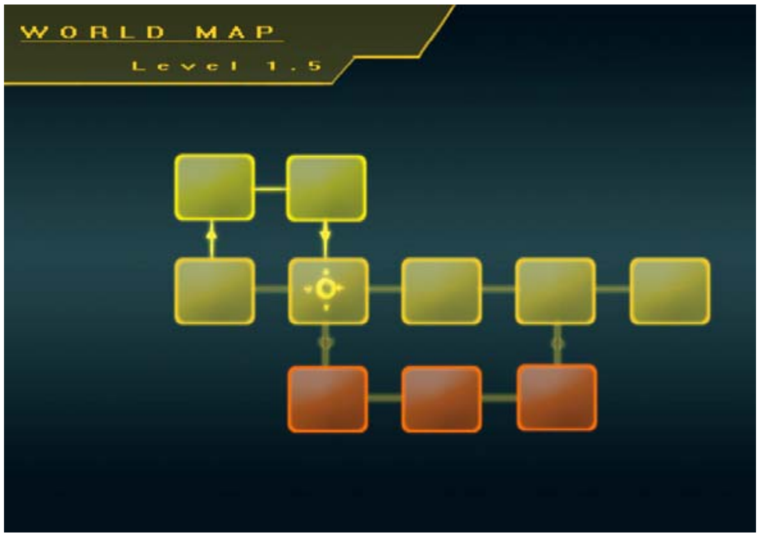


Figure 15. Concept art of the map screen, showing the available paths to travel through the rooms of the level. Shades of colors represent difficulty.

This section described the functional segment of the game design for Project Orion. There are ties with educative activities in some of the points referred in this section. The interface and the level design are linked with the pedagogical activities, and therefore they should be designed according to the original specifications. The other points described, refer more to the presentation of the game, and while they are also tied to the scenario in which the game will be used, there is more freedom into designing them.

4.2.2 System Architecture

Many decisions taken with the game design for Project Orion are aligned on the system architecture in which the project will be standing. The technical specifications set a series of boundaries of what can be done and that is why it is so attached to the game design. Game developer companies create game technical documents to describe the possibilities of the system structure and game engine used in the game (Novak, 2008). This section will explain the architecture used for Project Orion.

The game is designed to work on a client-server architecture, where the three packages in which the game is mounted are connected through either a local or wide area network. Figure 16 shows the components diagram that describes how the packages interact with each other.

"At the core of this architecture the server package manages all aspects of the platform. The client package is the interface which users use to interact with the platform and finally, the DOMjudge package handles all source code evaluation" (Kato, Gonçalves, Xavier, & Coelho, 2011).

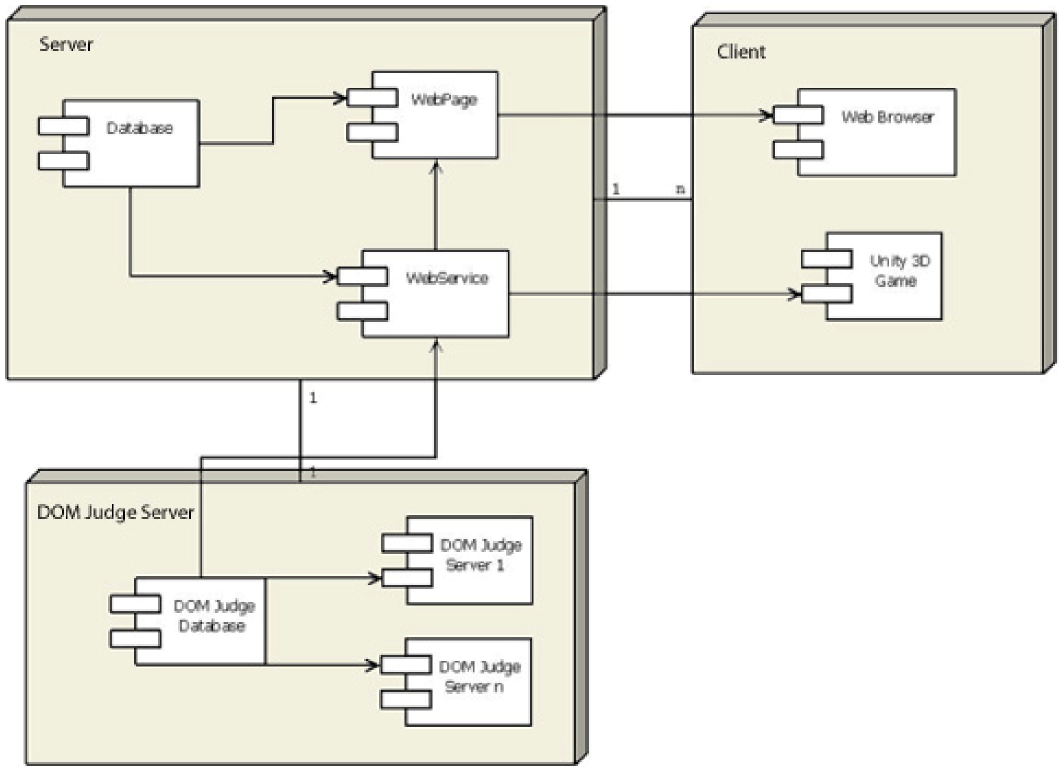


Figure 16. System architecture for Project Orion.

The server can be described as the core of the project; it pumps data to other components and packages. Its mission is to interact with the other packages by managing and processing all information. The game is basically allocated on a web server. All the assets of the game (game engine, textures, objects, exercises) are stored on the server side. The database holds information about the materials for the course. The designers chose from the pool of problems and make sets of similar problems related to a specific programming concept (Kato et al., 2011).

The access the game the student needs a recent version of a web browser. The game engine that powers the game is Unity3D; the client computer has to have installed the Unity 3D player. If this is not the case the student will be prompted to install the player. Development of the assets of the game has to be consistent with what the engine supports.

The DOMjudge Server is the one that holds the process that automatically reviews the code that students input. This package allows to use in the game any programming language supported by DOMjudge, meaning that there is not a dedicated parser for the game.

This architecture also allows the academic staff to edit the game and make new versions of it. The editor has the tools to generate new levels and rooms, include pre made objects and characters and attach quest.

This section described briefly the platform that will support the game. The activities that can be performed by the students and the staff are also attached to the architecture. The next section describes the instructional activities that the student will have to perform once playing the game.

4.2.3 Instructional activities

The design phase of a SG has to result in a detailed scenario of the activities that the students will perform in it. A detailed technical description of the activities can be found on the game mechanics section, where the rules of the game explain the possible activities. This section describes the overall set of activities performed by students and professors and how are they designed to interline with the instructional design.

The student is able to explore the world, collect items, and interact with the objects and characters. These activities are meant to let players immerse themselves in the narrative of the game and increase the motivation to continue the game. Motivation to play is a significant characteristic of educational video games and that effective game design considers both intrinsic and extrinsic rewards for play (Dondlinger, 2007).

Every time the player enters a room, he will find a place with different layouts and objects. The player has the task to explore what is inside the room to discover what has to be solved to reach the next area. In the way of the player, there could be items he could pick or characters to interact with. Depending on the design of the level, these items and characters can reveal information about the plot or about the challenges that have to be solved in that room.

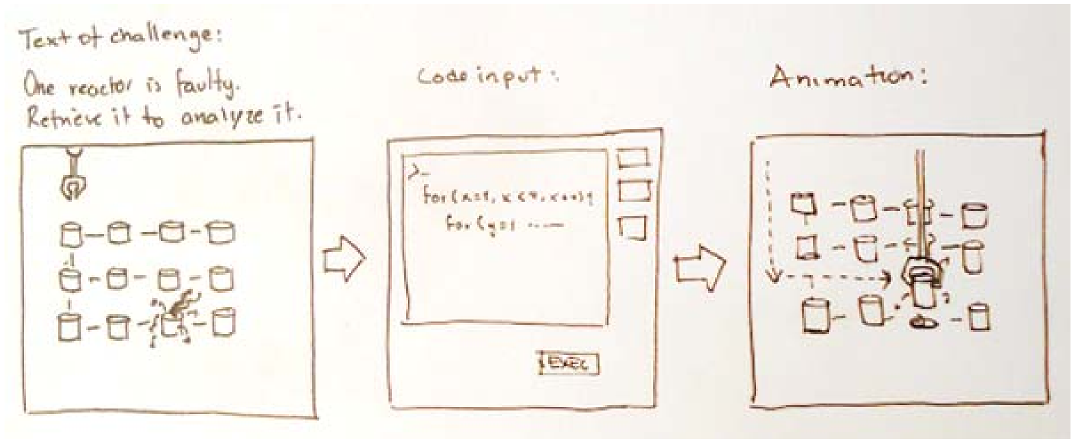
The challenges have a set of points that the player can acquire as he solves them. These points add up in a score level that handles the in-game performance on solving the challenges. The use of leader boards on games motivates players to increase their score so their name stacks higher on the board. Scores of the player will show up in a leader board and in that way the student will see how is he faring against other players. To avoid direct competition with other players, the name of the rest of the players of the board will not be visible to the student that is looking at the leader board through his designated profile.

It is important to note that it is not the intention to use the score as a grade on the course score of the student. The intention is to motivate the students to go for the harder levels and train with more difficult exercises. However, finishing determined number of levels, defined by the professor, can help the student to cover the necessary material to face the real course test.

It has been mentioned on the previous chapters that the main activity that the player is going to do is solve programming problems by actual code input in the game. When the students access a terminal, they will be prompted with the challenge that needs to be solved. The text explaining the task has to fit in the context in which the player is standing, this means that the exercise depends on what is happening on the room in that moment. The objects in the room must be linked with the text that asks the student to perform some action. If the exercise is not related to the actual situation of the player, it will work as a mechanic isolated from the game, like those seen on edutainment software.

It is here where exploration comes handy. The student must study the room and work out a way to go out of there. Objects inside the room may give clue on what is going wrong in that section of the level. Scattered items or characters may have information of what kind of code structure is needed to interact with the terminals. Active thinking is promoted as the player needs to seek for a way to solve the puzzle in the room.

After receiving a positive evaluation of the code, a trigger is cast into the objects to react according what the code is meant to do. For example, if the player is asked to navigate and find information through the index of an array, a correct code can trigger an animation demonstrating how was done the movement inside the array. Figure 17 shows the sequence of this example. The intention is to activate critical thinking on the student as he/she studies what the code does.



```
>_
for(x=1, x<4, x++){
    for(y=1 ---
```

Figure 17. The player reads the problem, writes the solution on console and triggers the animation.

The student can access educational material and documentation at any given time, though it will be available little by little at the same pace as the student advances in the game. The student has a log where he can review hints and tips collected during the game. It is possible to save pieces of code and notes on the log of the character. This will let the student review and use pieces of code that he already mastered in situations in which use of similar code structures is possible. This activity is intended to help the student to think about previous experiences and own thought on exercises done before.

The professor also has a role in gameplay. The professor is able to review the advances of the students playing the game. This allows monitoring if there are any problems with any given section of the game, this relate to determined concept, meaning that further instruction on that concept can be given at lesson time. The professor is the one that broadcasts help messages through the game, as if he was an entity living on the game world, like an artificial intelligence on the planet's system.

The activities described above reflect the educational objectives that are planned for the game. As seen on other methodologies, this step is designed before entering the production stage, trying to fusion the instruction design into gameplay. The next step is to go more specific with the design choices in which tasks "evolve into more detailed depictions of gameplay" (Mcmahon, 2009). This defines the main mechanics of the SG. The proposed game mechanics are explained in the next section.

4.3 Game Mechanics

An integral part of a complete game design and the main aspect to build gameplay is to define the mechanics that will power the game. This section explains what those game mechanics are and afterwards describes the proposed ones to be used on the game design of Project Orion.

Every game has a set of rules that indicates how it is meant to be played by a person. These rules are also called 'game mechanics' and are included from board and sport games to video games and computer SGs. Taking this concept to the digital version of a game, players tend to focus on the elements, which may be perceived as a group of mechanisms or "black boxes", which are there inside the virtual world, visible or not, to allow them to play the game (Fabricatore, 2007). Game mechanics is what makes a videogame playable, and serve as basis for gameplay; if the mechanics are designed in a player focused fashion the final game may have a stronger quality.

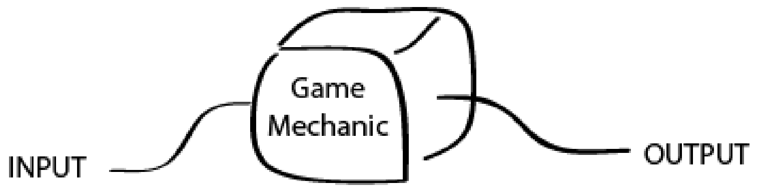


Figure 18. Game mechanics as a black box (Sigman, 2010)

A player or computer action could be a game mechanics at work. The function working under the mechanic 'black box' takes an input and transforms it into an output, as seen in figure 18. This is the feedback that a player expects. After the player performs one activity inside the game, the feedback is output to show that something is going on. The feel of this feedback brings certain satisfaction and motivation to continue operating the game. In instructional design, feedback mechanisms are important in the way that let the user know they are making progress towards some future state and using what they have learned, there is a pursue of additional pleasure (Danc, 2006).

Commercial video games main purpose is to entertain and the mechanics used in them let various modes of play; they incorporate goals, chance, rules, and discernable outcomes along with strong focus on presentation and storyline. Educational games focus on a learning outcome that is dependent upon an appropriate pedagogy and the underlying game mechanics and how the content is integrated into the game so the learning is intrinsic to play (Ulicsak, 2010).

For a SG the game mechanics should apply to the aim of the project, which has to cover the academic design within the concept of a game, without loosing focus that the main objective is to help on learning rather to entertain, using this last characteristic as a helper on the continued use of the software.

The primary core mechanics of a game are defined as the actions of a player that are most repeated during play time to reach a rewarded end-game state (Sicart, 2008). Those mechanics can be described as functions that are available to the player and explained at the beginning of the game, and the main actions that the player will be performing through all the game.

Secondary game mechanics are described as "core mechanics that ease the player's interaction with the game towards reaching the end state" (Sicart, 2008). These mechanics can be available from the start of the game or not, becoming as optional functions that are used occasionally through the adventure, but they will help to reach the end state of the game.

An important part of the mechanics is the player-token. A player-token is described as the main medium in which the player interacts with the virtual world, it could come in a form of an avatar or as an "invisible hand" (Fabricatore, 2007). It is with this token that the player interacts with the mechanics of the game; it represents him on the virtual world. Describing a character or the avatar in which the player will interact with the world is a very important part of the gameplay mechanics.

The first approach for the game design intended for the project was done during brainstorming sessions between the developers of the project and the supervisor about the main mechanics of the game. The main aspects of the game like the universe and plot, movement, and focus on the academic stance, were defined after doing some comparison between SGs in the subject and commercial games. A second approach was determined given the fact that the game should be modifiable by the academic staff of FEUP. This led to a creation of the concept to have a game editor available to professors, which would also have some mechanics attached to it.

The mechanics and rules exposed in this section are the proposed mechanics to meet the specifications of the project and the case of study, a SG for learning fundamentals of programming. These mechanics are divided into three parts which will involve the primary core mechanics for the game as well two separated secondary mechanics, one that describes the general gameplay for the player and the second one about the activities aimed for the professor.

As described on the game concept and design, the game will be a third person adventure game. The player-token for the SG will a 3-dimensional avatar that will represent a character. Accompanying the main character is the robot sidekick, working as a supportive character. This token receives input from the player to interact with the virtual world of the game. The character makes any physical interaction within the world while the robot sidekick is the one that gives advice to the player and prompts the main mechanic of the game: programming.

As part of the core mechanics, the end-game states for the game proposed in this thesis are to get the highest score possible and to acquire knowledge of the whole story of the game. The way to reach those end-game states desirably is to input the correct code during missions. The next sections describe how the primary and secondary mechanics work.

4.3.1 Primary core mechanics

Below are described the main functionalities of the SG and the mechanics that are working on them.

Exploration includes 360° land movement, jumping. Direct character interactions with objects occur while opening, closing, pushing, or pulling things. Character controls mechanics are defined as follows:

Mechanic	Control*	Key	Click
Movement			
Move forward	KB	W	
Move backwards	KB	S	
Turn clockwise	KB	D	
Turn counterclockwise	KB	A	
Move sideways right	KB+M	D	Hold Right
Move sideways left	KB+M	A	Hold Right
Jump	KB	Space bar	
Camera viewpoint	M		Hold Right
Interaction with objects, menu items, and options	M		Left
Open Items menu	KB	I	

Table 2. Table of main controls.
*KB = Keyboard / M = Mouse

Moving the character in the 3D world is necessary to continue advance. It is during exploration that the player can find the terminals for coding, understand the objects and puzzles of the room and to do some looting on items and logs that uncover the lost story. The main way of direct interaction with objects, whether is a terminal, a random object or a NPC, it is done with the left click button of the mouse. This will bring a small context menu of what can be done with the specified object.

Moving from one level to another is part of exploration, though because of the nature of the project, it is limited to single rooms representing one type of challenge per room. These mini levels are interconnected to each other through a series of passages, doors, transporters, and together they construct a level. Each room has several kinds of objects that construct a general puzzle. The character explores this 3D chambers, find objects and puzzles to solve, and then heads to the next room.

The **main interaction is done by coding** through the virtual terminals scattered in the game. When the character interacts with an available terminal, a screen presenting the command prompt appears indicating the instructions of the challenge and the area where the code must be input.

The following diagram represents how the mechanics should work. They areas shaded in grey show the 'black box' surrounding the working mechanics about code input. It is also shown the systems in which the game is running: the platform of the game and the server where the platform communicates with the code evaluating system.

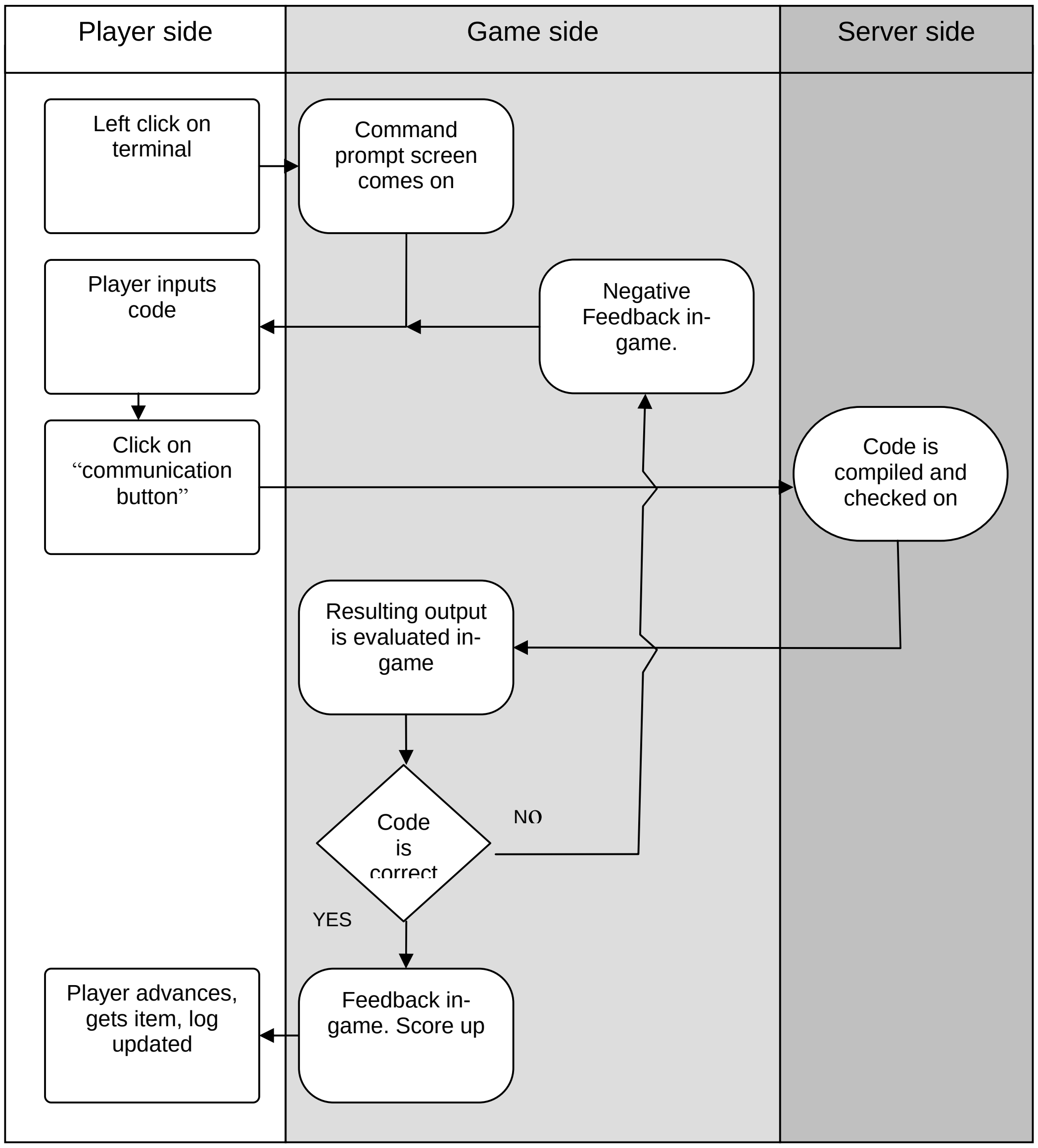

Player side
Game side
Server side
Left click on terminal
Command prompt screen comes on
Player inputs code
Negative Feedback in-game.
Click on "communication button"
Code is compiled and checked on
Resulting output is evaluated in-game
Code is correct
NO
YES
Player advances, gets item, log updated
Feedback in-game. Score up

Figure 19. Flow chart of the coding mechanic.

After receiving information about the problem that has to be solved, the player has the task of input the code on the screen of the command prompt. When it is done, the student clicks on the "communication button" that sends the code to the reviewer system located on a server, the one used at FEUP is DOMjudge, an automated judge system used in programming competitions (Eldering, Kinkhorst, & Warken, 2011). This lets the game be language independent, since it is not the game that compiles and executes the code. The server then sends the flag to the game system as a success or failure, the game takes that answer and reacts to it. The player then gets the answer as a positive or negative feedback. As shown in the diagram, the player may have multiple attempts to solve an exercise.

Help and hints are given through the game using the knowledge of the robot sidekick or manuals scattered in the game. Further theory concepts insight and code clues are also explained to the player through the game after solving main challenges. The student may be able to call the robot sidekick for aiding on any given quest if it is designed that way.

Mechanic	Control	Key	Click
Help			
Game help menu	KB	F1	
Robot aid	KB	H	
Review of Materials	KB	F1	
Logs of quest	KB	L	

Table 3. Table of help controls.

It is important for a SG for learning that there is enough documentation about the game and about the materials of the course covered to be available for the student at any time. There will be help for game use and review of academic materials all reachable from the help menu. The robot companion can prompt aid messages of what is happening in the game, from quests and challenges to the path to follow. Hints of what are the necessary steps to complete a code can be also be given by the secondary character. The log system controls the advances of the player, giving information on what is the last step, quest or room explored.

4.3.2 Secondary game mechanics

Secondary game mechanics describe the rest of the functions of the game. These mechanics are categorized in two groups covering the two parts that compose the whole project: the player part for students and the editor part for the academic staff.

The playable part

The player part of the game is where a student interacts with the software, the playable section of the full package. As the main part of the project it is there where the primary core mechanics are to be implemented along with the secondary mechanics, to complete the required design of the game and the specifications of the project. The activities in this section are:

Each of the terminals presents one or various quests to solve. The problems are assigned during the edition of the game as well with the relationships on other objects and their influence on them. The correct solution of a problem triggers the interaction with the objects on the room. The player gets access to other areas, a hazardous object or zone is cleared, or items and plot chunks are obtained. As described before if the player fails to a challenge he can try again.

The correct output of the code analysis sends a trigger to the object that is liked to the terminal, the object as an asset of the game react to its own functions. There could be a linkage between various objects, like a circuit board, the player may be required to solve more problems to open the doors so the current could flow between objects to solve the whole puzzle.

“The score (which is a great motivator in any game and provides the player with the stimulus for continuous improvement) rewards and penalizes the player” (Bellotti et al., 2009), and as such is a tool to be used to increase motivation of the player.

Player increases his global and level score depending on the quests solved within the level. A value of points is given to every problem to solve, this is the maximum of points that each quest contain. Getting some special items and solving optional quests increases the final score for the player; these activities are harder programming challenges to perform.

The optional challenges are not necessary to advance and serve as motivators to get a higher score by players that have more skills on coding. The value of the score is defined on the editor of the game. The following diagram explains the flow of actions while solving a problem.

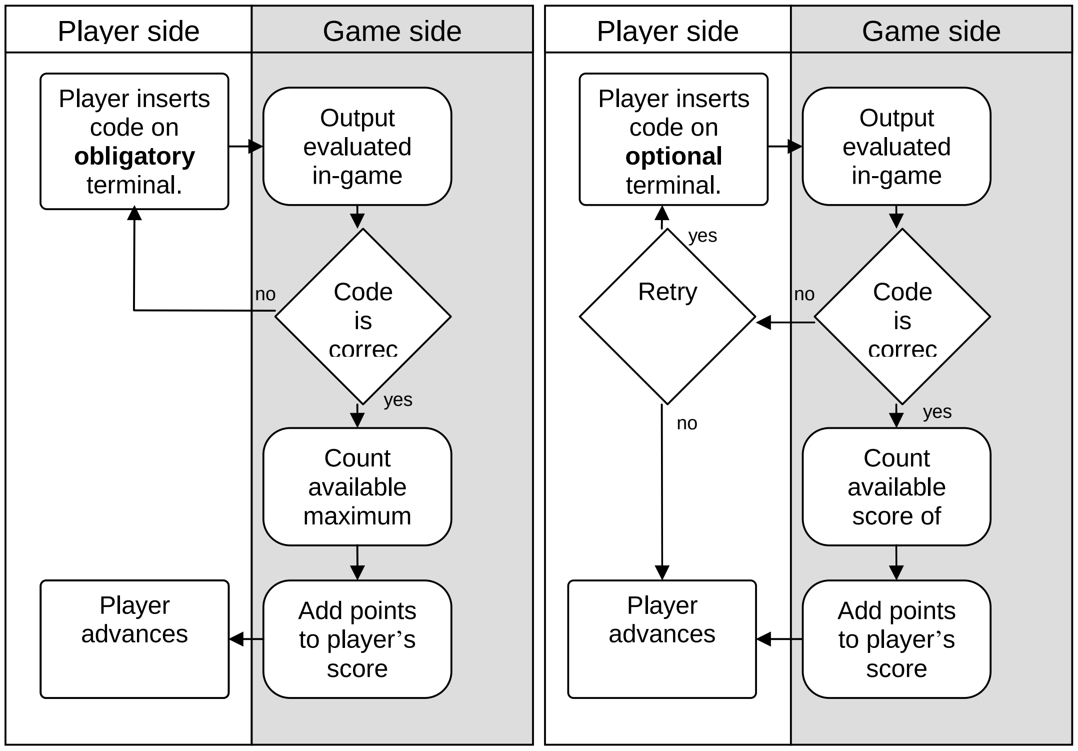


Figure 20. Flow within obligatory (left) and optional (right) challenges.

Award to the player comes as addition of points from the score available for a specific quest. Correct code input performs a limited point addition from the full value of a quest. An incorrect code will maintain the score at its minimum. The errors that are taken in consideration for this penalization are: compiling errors, failed compiled code and unexpected results including memory overflow and infinite loops. A compiling error may be the most common mistake made by students prompting quick help from the game for them to review. For the rest of the errors the mechanic of penalization will be counted as failed attempts of a compiled code without reaching success, Help to solve these problems has to be more complete and deep.

Number of tries, as a number, may not be penalized. After a defined number of tries, help and hints from the game will be available to the player. Even after giving some hints, if the player reaches a limit of attempts, another pathway will then open, redirecting the player to a series of levels with easier challenges to bring up their skills. At the end of this new route, the character gets back the room he was supposed to reach, and try to fulfill the normal difficulty quests.

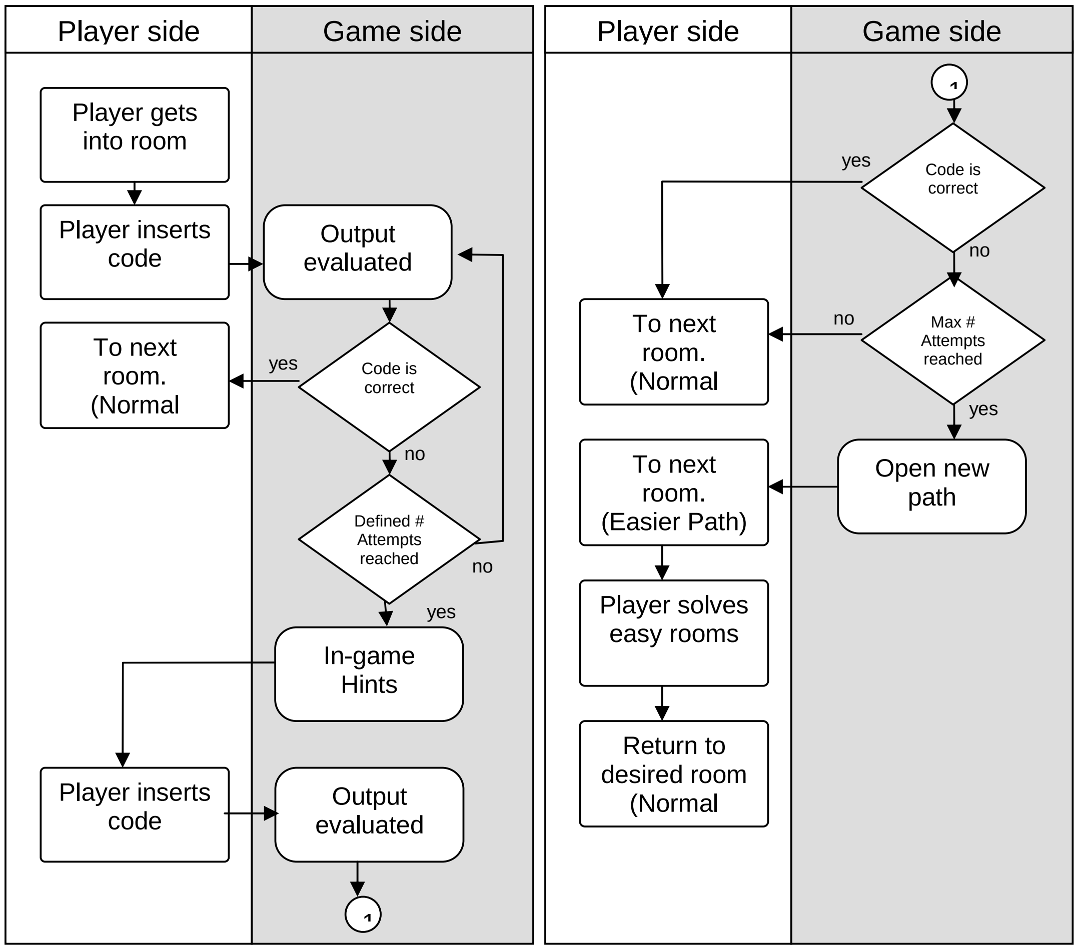


Figure 21. Flow when player fails several numbers of attempts on normal path.

Players can keep a mini inventory of items found during challenge solving or exploration of the levels. Items can be used to aid the player, to complete other quests or could be log or educative information about the problems that are being solved.

Items are categorized in four sets: adventure log items, academic materials, collectibles and achievements. The first two sets are contained on their respective menu and accessible at any time of the game to review either the plot of the adventure or usable information about the programming challenges and theory concepts of the subject. The last two sets are for score and level up of the player and are stores at the player profile and items menu.

The editor

The editor is planned as a complementary section of the full software and will be accessible by the academic staff to generate new games, following the same core mechanics to every game, to attain other final learning objectives.

This chapter will describe this part of the software, which will allow professors to do changes on the overall world of the game. The modifications of the game are done in the editor interface, allowing the academic staff modifies the general educative options of the game. After finishing with the edition the system will allow the professors to generate a package with the input of all information and then the system will use this package to construct a working game according to the selections done by academic design. The main changes that can be done in the editor are: edit room layout and level conformation, categorize problems in pools and choose a selection of problems to be used on levels, modify the contents of the help system and activate or deactivate in-game metrics.

The **edition of the world** and levels allows staff to edit room layout, the objects that are contained on them and the connection between each room. For room edition the editor will represent the space of each room with a finite matrix in two dimensions. The matrix works as the graphical representation of the room will allow the professor to identify what are the objects to interact and places where they are standing to build the challenges that the player has to face. Floors and walls can also be defined into the matrix delimitating the space and shape of the room with the option to define up to three upper floor levels inside a same room. Objects for transportation and passage between the rooms are also placed in this interface.

Objects are defined as assets with modifiable parameters which are changed during their placement on the matrix. The parameters available for each object depend on its classification, whereas on object for a programming challenge would allow setting a problem to it, a transportation object will define the route and linkage between rooms. The paths for an easier route to improve the abilities of a student are also defined here, assigned to an invisible object that will appear when the number of attempts triggers it. An option to test the room before and after saving will allow the editors to try the layout of the room and the relationship of the objects in it.

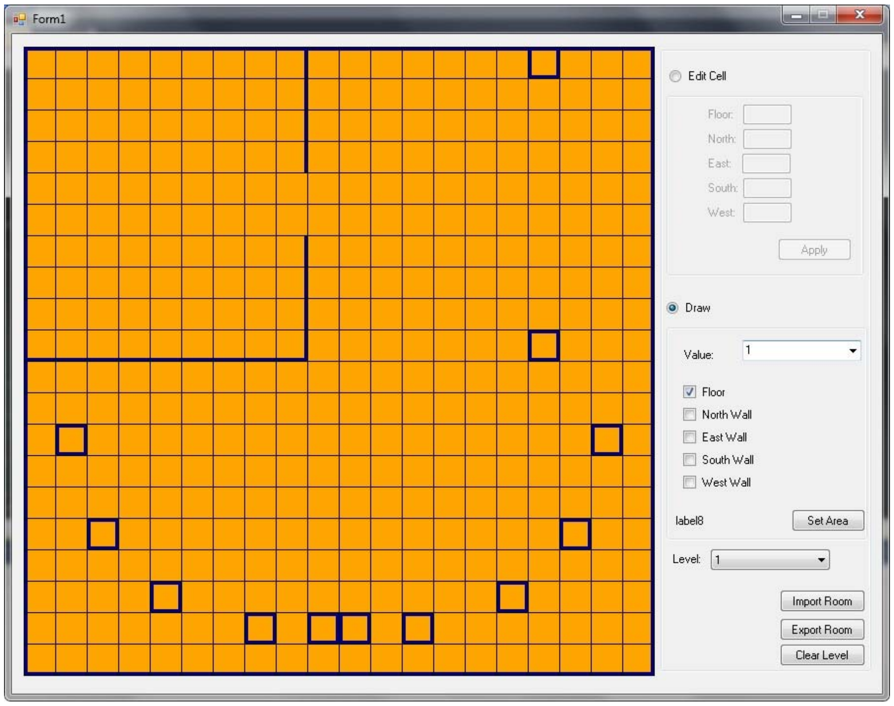


Figure 22. An early build of the graphic interface of the editor (done by Ricardo Gonçalves).

Changes on graphics for assets of the game, such as textures and type of objects can be done and applied at any part of the edition, but depend on the creation of them before intended use. There would be a library where these assets could be chosen from and applied to the rooms and objects of the game. If the staff arranges to have enough variety of this kind of assets, new iterations of the game could be even more varied and distinct of the previous one.

Choosing problems and embed them into specific objects and rooms of the game is an important functionality of the editor. Professors can select problems from a database and categorize them in different pools according difficulty or subject. The categories are important to control because with them the right subjects and difficulty are defined according to the academic course and the levels of the game, ensuring that the pace of the game goes along the materials seen at class. Then it is defined from which pool of problems the challenges are going to be taken and applied to the terminals, the place where the student will solve each task.

A random function will assign specific problems at the moment the player access his game bringing a slightly different experience between students and each time the player replays through the game.

Enhance learning with some other options of the editor such as the easy level routing and help and documentation structures. Alternative paths with easier exercises can be designed and linked to the normal path to help students develop skills for normal difficulty challenges. This has already been explained before, but it is important to the learning curve, that the students have the option to improve their skills in another way if a challenge is too big for them.

A help system can be modified and it is attached to the creation of the levels. The different types of help can be both input to the system as text, images, animation or voice and afterwards categorized according to the plan of the game. Help information can be attached to the objects of the game such as the terminals, log information or items found by the player. In a very similar fashion of that of creating the easy path after a number of failed attempts has been reached, help hints can be triggered.

Just as the visual assets of the game, this information has to be designed and stored outside the game. Then they can be imported and assigned to objects inside the game as mentioned before.

Predefined **metrics can be activated** for a specific game version. Metrics are important for evaluation of the course and how is the student dealing with the game. In the videogame industry especially on the social games branch there is discussion about metrics on those games, whether they are in an ethic problem through monetizing players or just improving the game with the information taken from them. Since this is not a game for economical matters the metrics proposed are those that help either the student or the professor to evaluate advances and uses on the game. They will be treated as information to better understand if the game is working or not, and that's why the metrics should be well designed for the experience.

These four are the main functions of the editor, being the options that the academic staff has available to edit and modify the game for their own course. The core mechanics, the overall flow of the game, its back story and main presentation design are not changed with the editor. In this way a new version of the game can be created quicker and more efficiently than designing a game from the ground up.

The complete package is ready for a major re-design if needed, though it will need more planning and resources, since most of the assets and game scripting would have to be reconstructed from ground-up. The next section explains the development stage of a game.

4.4 Development

Once the pre-production stages are complete, the next step is to develop a prototype to introduce the game mechanics to the people that is going to give green light to the project. This thesis covers up to the pre-production stages and the implementation of a first prototype, however this chapter will describe how further production is intended.

Production is the longest phase of development and it involves all the teams in charge of developing the games. Teams are formed by multidisciplinary people, which have always a leader to guide the efforts of all and communicate advances to other teams. The several roles that those teams cover are: producers, designers, artists, programmers, audio engineers, quality testers and marketing experts (Novak, 2008).

In the production stage there are milestones that are reached as development advances. Novak states that it is not a rule but usually developer studios call them: alpha, beta and gold phases.

At the alpha phase the game is playable from the beginning to the end. The engine and user interface (controls and navigation) are complete, and testing starts to be done on gameplay features and game modules to find gaps and bugs. This is the stage where the game is polished and marks the first time the game is tested by people from outside of the development teams.

When the game reaches the beta phase all the assets including the artistic specifications are complete. Development ceases and the focus of this phase is to find and fix the biggest number of bugs possible. Performance and playability are also tested, and it may include people recruited even from outside the company, in most of the cases via online testing. All the material that will accompany the final build of the game has to be created, this includes: manuals, compatibility of hardware and interfaces, localizations.

After the beta phase the game is considered gold. It is at this phase that the game is sent to be manufactured as copies of a master game disc, after they have been thoroughly tested. Management reviews the product and agrees that the product is ready to enter manufacturing stage.

Project Orion is not intended to match commercial games in terms of production value, but if it is intended to be a great experience for students, its development must be done also by a multidisciplinary team.

The role of the producer must be taken by the professor that is in charge of the course, since it is he that defines which academic objectives have to be covered by the game. As seen on previous sections, instructional design has to be embedded into the concept and design phases of the game. The tutors and professors are cognitive and domain experts and in their role as producers, it is crucial that they keep continuous communication with the development team.

The designers have the task to create the style, the levels and the interfaces of the game; since Project Orion consists on changeable items, all of them can be re-designed if it is needed. The creative style and the interfaces can be handed to artists, but it is the level design that requires great deal of attention from the designers. The previous sections describe that the objects in a level have to be related to the problems that are expected to be solved. To ensure that the best feedback is given to players, the objects must react according to the programming concept that the student is experimenting on any particular level. Other tasks that the level designer must do are object creation, room placement, navigation through levels, items and triggers involving plot progress.

Artists have the task to achieve the creative vision of the designer. A complete version of the game, or further versions, requires a bigger set of visual assets. To create new levels and puzzles, objects in the game have to modeled or arranged from a source like the Internet. To create new environments, textures for the rooms have to be designed. All the created objects and graphics have to be placed on a database that is accessible to programmers and designers. Project Orion does not require complex and organic assets, since it would be difficult to place them in the current version of the editor. However, animators may be needed to enhance the animation clips that are intended, these may include: character movement, feedback of problems, explanatory clips and plot cinematic.

After all the assets are arranged the programmers commence work on scripting the game. The role of the programmers is to make the game work. They will work on the Unity 3D game engine to implement the game and the objects populating it. They are the responsible for the architecture of the system; they have to be sure that the components are working and make further testing any time a new version of the game is created. Programmers may increase the capabilities of the current architecture model, opening new opportunities for the instructional and game designers to implement new features on the game.

Audio is a very important part of the game experience. Feedback is enhanced with audio cues and immersion is greater with music. Audio experts are in charge of composing music and design sounds. They will create pieces of audio with enough quality to improve the experience of students with the game. They must be sure that the music goes according to the setting and that the sound effects are used correctly when needed.

This process is an iterative one. Testing is done through each of the phases of game development. However usability testing is essential, because it can reveal problems that deal not only with the game design but also with instructional design. While doing usability testing students can make comments of what they like and dislike of the game features and place observations if the instructional functions included are enough or not.

This section summarizes what is the production stage who is involved in it. Development teams formed with people of complementary profiles are the responsible for bringing the game to life. Further research on the game development processes can help the development team to create a product that is capable as a tool for learning. It is possible to develop the game 'in-house' with the resources that are available at FEUP. This section finishes with the game development chapter. The development of the prototype and its evaluation is going to be discussed in the next chapter.

Chapter 5 Prototype development and Results

Using the game development process described on the chapter before and the specifications of the project, the next step was to implement a prototype for testing. The prototype was used to test the proposed game mechanics on students of the same major as the targeted students. The next sections will depict the process done to construct the prototype and the assets generated for it. The results of the evaluation of the prototype will also be analyzed according to the specifications described on chapter 3 and the mechanics proposed on chapter 4. It is also explained the cooperation with another dissertation that describes the architecture of the project.

5.1 Prototype Development

The first step taken into the development of the prototype was to design the game mechanics depicted on chapter 4. Of the proposed mechanics only the help system on the primary and secondary mechanics was left out on the prototype because of time constraints. Exploration and interaction mechanics were fully implemented while the secondary mechanics were executed in a limited way.

After the game mechanics were defined, the rest of the game design process was followed. It started with the creation of the assets that were going to be used on the prototype. Seven types of objects were created for the two rooms planned for the demo level, all of them attached to the scenario context:

- Terminal – object that allows the player get into the programming exercises and the screen where the code is input. For the context of the game, the terminal represents the place where the character can interact with the AI in the world.

- Door – navigation between rooms is done by passing through doors.
- Transporter – an object that lets the player reach other areas inside the room that are not accessible by just walking.
- Data pad – this is an object that gives information to the player about the exercises or the story of the game. Important hints on the programming challenges can be found when the player analyzes the data pads.
- Force fields – this object impedes access to a certain area.
- Lights – objects used to light the scene. They can give feedback on accessible areas or doors.
- End object – designed to be a door in the prototype, this object triggers the end of the demo.

These objects have specific functions built into them. When the student solves a challenge correctly, the game receives the trigger from the reviewer package and the objects react to let the player continue with the level. The objects were obtained from free sources on Internet and then modified and retextured to fit the game context. The next group of images show some of the objects generated for the prototype.

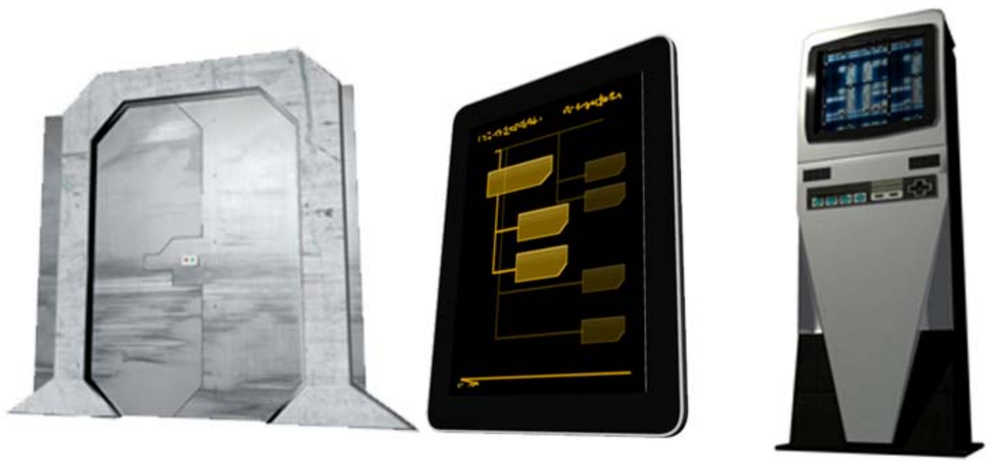


Figure 23. Rendered images of the modified door, data pad and terminal models.

The avatar for the player, Isaac, was also used in the prototype. An animated model that exists in a Unity 3D tutorial was used due to time restrictions and to avoid technical issues regarding to animation. The model was slightly modified and retextured to bring greater contrast between the character and the world, since the original textures were too dark, this can be seen on figure 24. Textures for the walls and floor of the room were also generated for the demo.

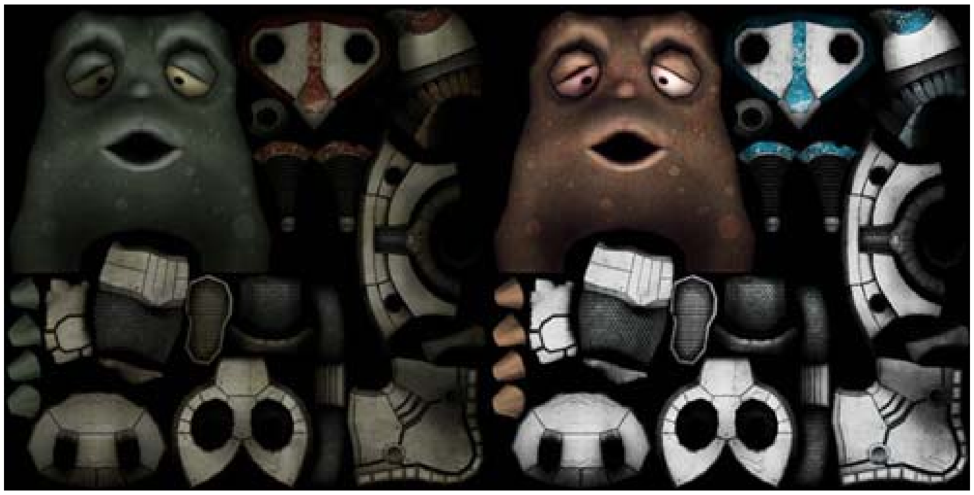


Figure 24. Contrast and color correction (right) done on the original textures (left). Textures taken from the materials of the Unity3D tutorial.

The next step was to create the level of the prototype. The level covers four programming exercises, all of them accessible through the terminals scattered on the rooms. The level was separated into two rooms and the player has to solve two challenges on each one of them. Figure 25 shows the first level design concept for the prototype. There can be seen the starting point for the player, the objects in the room and the path that the player should take to finish the level.

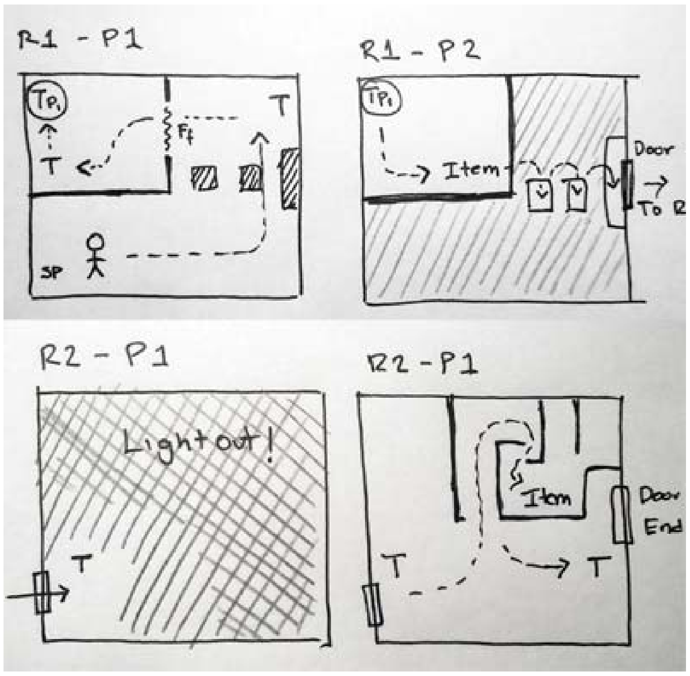


Figure 25. Level design of the first and second rooms of the level.

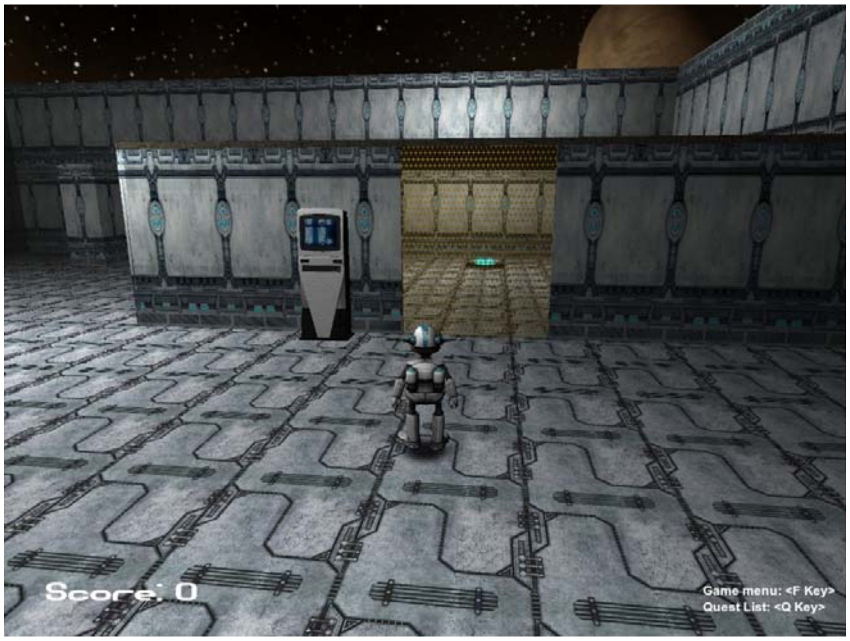


Figure 26. Screen shot of the actual prototype level.

After all the objects are ready the actual prototype level was created on the level editor shaped for the project. Testing of the placement and looks of the assets was done to ensure that the objects were easy to spot. Some of the walls and platforms changed place from the original design during testing to achieve a better relation between exploration mechanics and the challenge mechanics. A final screen of the looks of the level is shown on figure 26.

The programming exercises where based on the use of expressions and navigating through a set of numbers. At the time of design, two types of instructions were written. First the direct instructions of what the student has to do. Second the context driven instructions that related the challenge to the game concept. Table 5 show the texts used to tell the student what were the tasks to be solved. The intention is to attach the exercises to the game context and make the objects react to the outcomes of the code that is input.

The message given by the last terminal shows only <Enter Password>, the player has to find information of what is the password asked by the terminal. On the second room there is a data pad that holds log information about one engineer, whose name is Pherick Iliam, that recorded his activities long before the Isaac arrived to the planet. There is another data pad on the first room with hints on how to do one of the exercises. Both data pads have information tidbits of what is happening in the story at the time represented on the prototype as seen on the table 6.

Quest	Direct question	Context driven question
Q1	Write a program that receives 5 numbers and returns the position of the smallest of all.	This force field receives energy from five power generators. To bring it down, set the value of the five power generators and return the number of generator that has the least amount of energy available.
Q2	Write a program that receives 10 numbers and makes the harmonic mean calculation of them. There is a data pad that has that equation..	It seems the transporter is not calibrated; it would be dangerous to start it on like this. To calibrate it you must set up the harmonic mean of a frequency of ten numbers.
Q3	Write a program that receives 5 numbers and then returns the positions (relatively to first entry) after ordering them from min to max.	The light of this room is out of energy. The problem is detected on the arrangement that has been just done on the five power generators. To restart the power line the generators must be arranged by their capacity values, from minor to major.
Q4	Password unknown Override password. Search for instructions.	<Enter Password>:

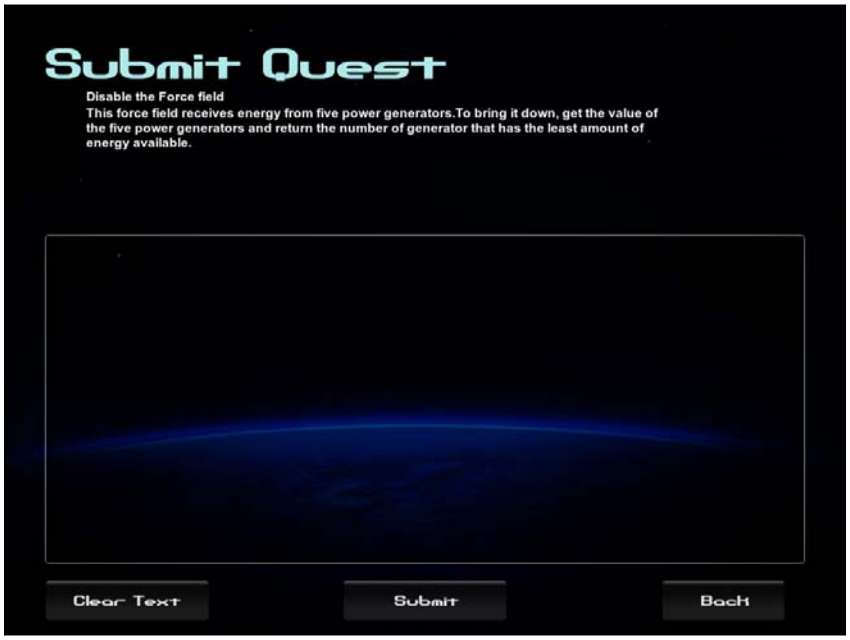
Table 4. Challenge instructions included in the prototype.

DP		Context driven text
DP1	HINT FOR Q2	Pherick Iliam's note log. I will take note of this since it is easy to forget. It should be handful if the teleporter loses its calibration, when all this time of confusion ends and we can settle down again. Harmonic Mean Equation 1/H = 1/ ((1/n) * m) m = (1/a1 + 1/a2 + ... +1/an) or $\sum_{1}^{n}\frac{1}{a}$ a= frequency numbers * It just needs to find 10 frequencies so n=10
DP2	TO SOLVE Q4 Write a program that receives 2 numbers as a min and max of a range of numbers, and calculates the average of the prime numbers within that range. EX: 4 -10 → prime numbers: 5,7 → average of pn: 6.	Pherick Iliam's note log. The situation doesn't seem to recover at all. I can see the horror on people faces. This is not what they told us that would happen. I may not be able to come back, in that case it is better to leave a backdoor to override the password of the access door of this sector. If someone finds it before me, please recover what belongs to us. To override the password, the average of the prime numbers contained on any range of numbers (MIN and MAX), it's needed.

Table 5. Text depicted on the data pads scattered in the level of the prototype.

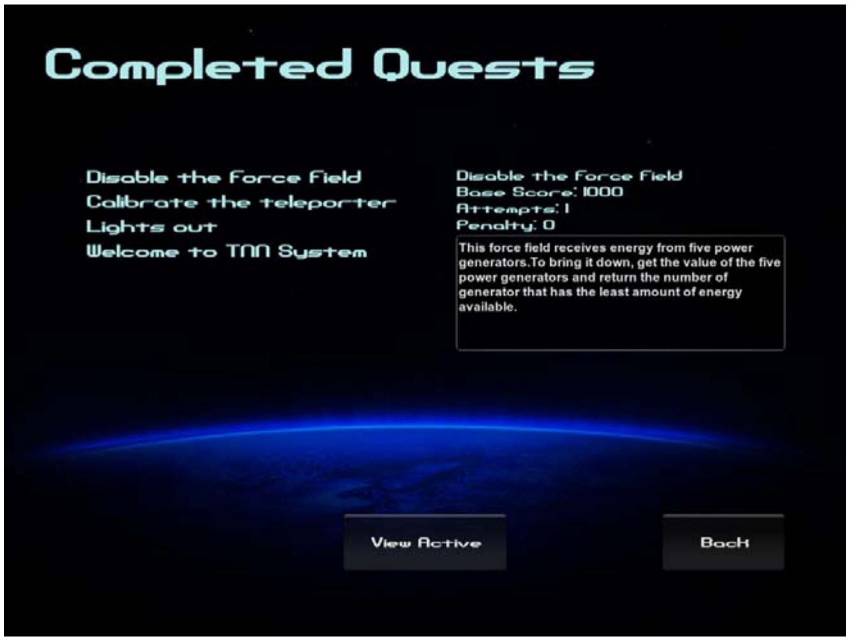
Finally the prototype screens were designed. These screens give access to the student and allow them to create several game files. Two screens explain briefly the story of the game before the player begins game play. And the rest of the designed screens are use while the players are experimenting with the prototype, which include the programming section, the quest screen that describes the task to do and the menu screen. Figures 27 and 28 demonstrate the general interface of the programming and completed quests screens for the prototype.

An error feedback was implemented, prompting a message when the code resulted in a failed trigger from the DOM Judge server. The message is prompted on the programming screen, reducing the available points of the challenge.



Disable the Force field
This force field receives energy from five power generators.To bring it down, get the value of the five power generators and return the number of generator that has the least amount of energy available.

Figure 27. Screen of the programming interface of the prototype.



Disable the Force Field
Base Score: 1000
Attempts: 1
Penalty: 0

This force field receives energy from five power generators.To bring it down, get the value of the five power generators and return the number of generator that has the least amount of energy available.

Figure 28. Screen showing the information on completed quests.

A brief period of testing was done to make sure the prototype was working correctly and the navigation through the screens was right. The implementation for testing was done in one of the computer labs of FEUP since the connection to the web and the DOM Judge servers is required for the game to work. The next section discusses the results of the testing done on the prototype.

5.2 Results

This section will evaluate the final state of the prototype according to the specifications of the project and game design discussed on previous chapters. The results obtained from the testing phase will be also analyzed.

It was mentioned before that during the game design process a concern about the feedback that the game can give to the player. This concern turned in reality while designing the prototype. Since the objects react to triggers that are not game specific they cannot be easily linked together to give actual feedback about what the code does. This could happen with logic errors on the code. For example, if the student writes a piece of code that is compiled but the result is not what is expected, the game will be unable to show this unexpected behavior.

This situation is considered a weakness in the game design because it cannot apply one important pedagogical concept that is needed to implement good instructional design on the game. The game will not give accurate feedback on every case. However this is attached to the type of platform that supports the game, meaning that both game design and platform have to be tweaked to convey a better feedback for the student.

The help mechanics were not implemented as proposed on the game design chapter.

The rest of the game mechanics proposed in this thesis cover the specifications of the project. The game design is open enough to cover several programming languages supported by the DOMjudge server. The game is designed to be editable on secondary mechanics and visual assets, without having to change the core mechanics of the game, which allow the academic staff to produce another version in shorter time. The architecture of the game allows the game design to accomplish this.

After finishing the development of the prototype and implementing it successfully, it was tested by students with education on informatics. While this is not representative of the focus group at which the game is intended, they can compare the game with their own experience built during programming courses and formulate particular opinions. During the testing time with each of the students, usability testing was done via the observation technique. After finishing playing the demo, the students filled in a questionnaire regarding their experience with the game.

The first findings revealed that players had to take some time to adjust to the control scheme of the game. They explained that the control felt weird without being awkward, though after some minutes they where used to it. However the jumping section of the level proved that the controls make jumping to other platforms without falling a very difficult task. A change in control scheme and simpler level design can handle this problem. The ability for the player to replay a specific part of the game after failing is an important gameplay feature that diminishes frustration. In the prototype a shortcut was placed to get to the other side of the jumping part, in order to let the player continue testing.

An important outcome of the testing phase came quite immediately after starting game play. The players were completely motivated to explore the room as they tried to go to the farthest corners, even when they found out that the only objects in the room were the ones visible from the beginning. Students made the observation that the visual aspect of the game was attractive enough, but that greater graphics would make it more interesting to explore.

Students liked the idea of the game story, while at the same time they made the observation that if the story is more complex it would be better for the immersion of the game. They also noted that sound would enhance the experience, but the focus on it was with a lesser degree than graphics. The prototype does not have sound effects or music, so no further opinion was made on this point.

The screens and hotkeys were manageable and did not present any problem for the players. They expressed that the intention of instructions to be aligned to the game context was a neat idea, but that made somewhat difficult to understand the actual programming task they had to do. The prototype only shows the context question for the challenges, so to mitigate this concern the direct question can be also shown to the player. Error feedback was also found to be weak, the students suggested that more information about the type of error would be better to make corrections on the code.

In overall the results of the usability testing and the questionnaire showed that the students believe the game can be used as a tool that can complement programming learning. Computer technologies change rapidly and thus the uses of programming languages can change as well the structure of the programming courses. The comments given by the testing group, graduate students, may differ of what the opinions that a real testing on new generations could bring. The overall mood of the testing group helps set the project on a base and make further adjustments for future testing on the focus group.

5.3 Project cooperation

This project was developed in cooperation with Ricardo Gonçalves, whose thesis is focused on the same project but from the system architecture standpoint. It was through this cooperation that the game design was able to be completed while at the same time aligned to the specifications of the project and the technological tools available for it. Ricardo designed the technological platform that support the game and constructed the prototype, benefiting from the game design proposed in this dissertation.

Chapter 6 Conclusions and Future Work

6.1 Conclusions

Through the writing of this thesis several steps were done:

- A review of the concepts of serious games for education, edutainment and the pedagogical principles that are embedded into development of this kind of interactive software. This includes the methods used by researchers and experts to incorporate instructional design into games and related work done before.

- It was described the process of creating the game concept and the game design for the project following the steps that development studios generally use for commercial games.

- Design the core mechanics according to the research done in instructional design and game design. Proceed to implement the game mechanics on a prototype under the limits of the specifications of the project.

The prototype exposed in the last chapter is the result of a combination of game design with instructional design and made it work with a technological platform, all of them under the guidelines for the project. The results show that indeed the game seems attractive as a tool to help learn programming. Does it answer the question if the proposed game design can make a game that works as an educational tool? The answer is that this initial prototype is not conclusive on wether the game actually helps students to understand programming fundamentals. The actual instructional design mimics the activities that are used during classroom activities, with the added mechanics of a video game, which seems to be able to do the job as the students felt when testing the prototype. However a more robust version of the game is needed to actually test the game on a programming course to show if the game helps on learning.

What the prototype does illustrate is the notion of having an open and editable tool to be used along programming courses, not only with the fundamentals of programming but with more advance courses as well, even courses that does not focus on programming. It was seen during the development of the prototype that the core mechanics support this conclusion. Secondary mechanics and other rules of gameplay are rudimentary at this point but they are a good base to conclude that the game design is sufficient to create a game of this nature. So, is it feasible the creation of a serious game to teach programming that is editable for other courses? It is possible, but additional revision and development on the game design is needed.

It is the openness of the game mechanics and the platform that set new challenges on the game design for this particular project. The changes are needed especially on the control scheme of the interfaces and the functions of the objects that construct the puzzles of the game. The level design has to be done with precision, so the objects used in the challenges react and give feedback to the player about the programming concepts that are being studied. Of course more assets are needed to complement the existing ones, although to do this more resources are needed such as developers, time and development tools.

The game is capable to support other subjects and educational materials. The game design allows the academic staff to make changes on the challenges and review the advances of the students and become assistants to the experiences of the students. The game has potential to become a successful tool to help in the learning of programming concepts.

6.2 Future Work

The main step in the future is to create a version of the game that is more robust in features and test it on the actual programming courses. This would help in improving the game design, but also the instructional design. The testing can be done as a separate activity or within the current evaluation activities of the courses. This may reveal if the instructional design used in the game is completely successful or not for this case scenario. Developers have to remember that “more research is needed to provide empirical evidence for how game-based learning can be used most effectively” (De Freitas, 2006).

The related work about projects of this nature shows differences in game mechanics and features between them and the proposed game. There are interesting features included on those projects that could be beneficial to include into the game design of Project Orion. The creation of assets by the players themselves giving more personalization to the software and enhance creative thinking as they try to create the programming puzzles for the game. A game that is capable of a multiplayer gameplay can open a community for students to work cooperatively and learn from others.

Sound is essential; the staff must include audio experts that can generate content for the game. A platform that handles the story, being either linear or non-linear, would diminish the complexity of level design since the story has to be included into it. It would be interesting to develop a tool to create multiple paths of the story and that the academic staff simply chose which path is going to be inserted in the game.

Improving and making deeper definitions of the game design will make Project Orion a successful as a serious game for education.

References

Adams, Ernest. 2005. The Designer's Notebook: Educational Games Don't Have to Stink! Gamasutra. http://www.gamasutra.com/view/feature/2190/the_designers_notebook_.php. Date Accessed : 13/06/2011

Anderson, E.F., and L. McLoughlin. 2007. Critters in the classroom: a 3D computer-game-like tool for teaching programming to computer animation students. In ACM SIGGRAPH 2007 educators program, 7–es. ACM. http://portal.acm.org/citation.cfm?id=1282048.

Bellotti, F., R. Berta, a. De Gloria, and L. Primavera. 2009. Enhancing the educational value of video games. Computers in Entertainment 7, no. 2 (June): 1. doi:10.1145/1541895.1541903. http://portal.acm.org/citation.cfm?doid=1541895.1541903.

Berg, Peter. 1999. Learn To Program: BASIC Review. Issue #9 of QB:TM. http://www.petesqbsite.com/sections/articles/ltpbrvw.txt. Date Accessed : 04/19/2011

Charsky, D. 2010. From Edutainment to Serious Games: A Change in the Use of Game Characteristics. Games and Culture 5, no. 2 (February): 177-198. doi:10.1177/1555412009354727. http://gac.sagepub.com/cgi/doi/10.1177/1555412009354727.

Chatham, Re. 2007. Games for training. Communications of the ACM 50, no. 7: 36-43. http://portal.acm.org/citation.cfm?id=1272537.

Cifaldi, Frank. 2011. Mission Of Honor Is China's Answer To America's Army. Gamasutra, May 16. http://www.gamasutra.com/view/news/34677/Mission_Of_Honor_Is_Chinas_Answer_To_Americas_Army.php.

Cifaldi, Frank. 2011. E3 Report: Education Arcade - Case Studies: Civilization http://www.gamasutra.com/view/feature/2308/e3_report_education_arcade__case_.php
Accesed: 29/09/2011

Colayco, Bob. So you wanna be a: Game designer. Gamespot. http://www.gamespot.com/features/6129276/so-you-wanna-be-a-game-designer. Date Accessed : 30/06/11

Crawford, Chris, Sue Peabody, The Art, Computer Game Design, World Wide Web, and Donna Loper. 2003. The Art of Computer Game Design. Osborne/McGraw-Hill.

Danc. 2006. What are game mechanics? LostGarden blog. http://www.lostgarden.com/2006/10/what-are-game-mechanics.html. Date Accessed : 05/12/2011

De Freitas, S. 2006. Learning in Immersize Worlds: A Review of Game-based Learning. JISC e-Learning Programme. http://www.citeulike.org/group/2518/article/1277752.

De Freitas, Sara, and Mark Levene. 2004. An Investigation of the Use of Simulations and Video Gaming for Supporting Higher-Order Cognitive Skills. Digital Age, no. Celda: 35-42.

De Freitas, Sara, and M Oliver. 2006. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated? Computers & Education 46, no. 3 (April): 249-264. doi:10.1016/j.compedu.2005.11.007. http://linkinghub.elsevier.com/retrieve/pii/S0360131505001600.

Djurovic, Gordan;. 2010. Implementing Cognitive Theory of Multimedia Learning in Existing Academic Programs. Transforming Engineering Education: 1–7. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5508949.

Dondlinger, M.J. 2007. Educational video game design: A review of the literature. Journal of Applied Educational Technology 4, no. 1: 21–31. http://www.mendeley.com/research/educational-video-game-design-a-review-of-the-literature-1/.

Eagle, Michael, and Tiffany Barnes. 2009. Experimental evaluation of an educational game for improved learning in introductory computing. ACM SIGCSE Bulletin 41, no. 1 (March): 321. doi:10.1145/1539024.1508980. http://portal.acm.org/citation.cfm?doid=1539024.1508980.

Egenfeldt-Nielsen, P.D.S.S. 2003. Making sweet music: The Educational Use of Computer Games. egenfeldt.eu: 1-13. http://www.egenfeldt.eu/papers/sweet_music.pdf.

Eldering, J., T. Kinkhorst, and P. Warken. 2011. DOMjudge - programming contest jury system. http://domjudge.sourceforge.net/.

Fabricatore, Carlo. 2007. Gameplay and Game Mechanics Design: a Key to Quality in Videogames. Citeseer. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.120.5942.

Games for Health Project. 2004. Games for Health. http://www.gamesforhealth.org/. Date Accessed : 11/06/2011

Gee, James Paul. 2007. What Video Games Have to Teach us About Learning and Literacy. 1st ed. Palgrave Macmillan.

Huang, Timothy. 2001. Strategy game programming projects. Small 4, no. May 2001: 205-213. Ismail, Mn, N Azilah, and U Naufal. 2010. Instructional Strategy in the teaching of Computer Programming: a Need Assessment Analyses. TOJET 9, no. 2: 125-131. http://tojet.net/articles/9214.pdf.

Jiau, Hewijin Christine, Jinghong Cox Chen, and Kuo-Feng Ssu. 2009. Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics. IEEE Transactions on Education 52, no. 4 (November): 555-562. doi:10.1109/TE.2008.2010983. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5164890.

Johnston, Hannah, and Anthony Whitehead. 2009. Distinguishing games, serious games, and training simulators on the basis of intent. Proceedings of the 2009 Conference on Future Play on @ GDC Canada - FuturePlay '09: 9. doi:10.1145/1639601.1639607. http://portal.acm.org/citation.cfm?doid=1639601.1639607.

Kato, Enrique, Ricardo Gonçalves, João Xavier, and António Coelho. 2011. Serious Game for Introductory Programming. Lecture Notes in Computer Science 6944: 1-12.

Klassen, Myungsook. 2006. Visual approach for teaching programming concepts. In Proceedings of the 9th International Conference on Engineering Education (ICEE 2006), 23–28. Citeseer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.9440&amp;rep=rep1&amp;type=pdf.

Leigh, Alexander. 2011. G4C: Al Gore Says "Games Have Clearly Arrived As A Mass Medium." Gamasutra. http://www.gamasutra.com/view/news/35310/G4C_Al_Gore_Says_Games_Have_Clearly_Arrived_As_A_Mass_Medium.php. Date Accessed : 11/05/11

Lynch, Jeremy, Paul Aughwane, and Toby M Hammond. 2010. Video games and surgical ability: a literature review. Journal of surgical education 67, no. 3: 184-9. doi:10.1016/j.jsurg.2010.02.010. http://www.ncbi.nlm.nih.gov/pubmed/20630431.

Magnenat-Thalmann, Nadia, and Zerrin Kasap. 2009. Virtual Humans in Serious Games. 2009 International Conference on CyberWorlds: 71-79. doi:10.1109/CW.2009.17. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5279708.

Marfisi-Schottman, I., Sébastien George, and F. Tarpin-Bernard. 2010. Tools and Methods for Efficiently Designing Serious Games. free.iza.free.fr, no. October: 226-234. http://free.iza.free.fr/articles/ECGBL-iza.pdf.

Mcmahon, Mark. 2009. Using the DODDEL model to teach serious game design to novice designers. Action Research, no. Proceedings ascilite Auckland 2009: 646-653. http://www.ascilite.org.au/conferences/auckland09/procs/mcmahon.pdf.

Muratet, Mathieu, Patrice Torguet, Jean-Pierre Jessel, and Fabienne Viallet. 2009. Towards a Serious Game to Help Students Learn Computer Programming. International Journal of Computer Games Technology 2009: 1-12. doi:10.1155/2009/470590. http://www.hindawi.com/journals/ijcgt/2009/470590/.

Nadolski, Rob, Hans Hummel, H. Van den Brink, Ruud Hoefakker, Aad Slootmaker, Hub Kurvers, and Jeroen Storm. 2007. EMERGO: methodology and toolkit for efficient development of serious games in higher education. Organizing and learning through gaming and simulation: proceedings of Isaga 2007: 259.

Novak, Jeannie. 2008. Game Development Essentials: An Introduction. 2nd ed. Canada: Delmar Cengage Learning.

Ochalla, Bryan. 2007. Who Says Video Games Have to be Fun? The Rise of Serious Games. Gamasutra. http://www.gamasutra.com/view/feature/1465/who_says_video_games_have_to_be_.php. Date Accessed : 10/05/11

Phelps, Andrew M, Christopher A Egert, and Kevin J Bierre. 2005. MUPPETS : Multi-User Programming Pedagogy for Enhancing Traditional Study : An Environment for both Upper and Lower Division Students. Education: 8-15.

Rooney, Pauline, K. O'Rourke, Greg Burke, B. Mac Namee, and Claudia Igbrude. 2009. Cross-disciplinary approaches for developing serious games in Higher Education: frameworks for food safety and environmental health education. Higher Education: 3-7. http://arrow.dit.ie/scschcomcon/44/.

Ryan, Tim. 2010. The Anatomy of a Design Document, Part 1: Documentation Guidelines for the Game Concept and Proposal. Gamasutra. http://www.gamasutra.com/view/feature/3384/the_anatomy_of_a_design_document_.php?print=1. Date Accessed : 29/06/11

Sicart, Miguel. 2008. Defining Game Mechanics. Game Studies. http://gamestudies.org/0802/articles/sicart. Date Accessed : 13/04/2011

Sigman, Tyler. 2010. Anatomy of a Game Mechanic. Gamasutra. http://www.gamasutra.com/view/feature/4091/anatomy_of_a_game_mechanic.php?print=1. Date Accessed : 05/12/2011

Smith, R. 2009. The Long History of Gaming in Military Training. Simulation & Gaming 41, no. 1 (April): 6-19. doi:10.1177/1046878109334330. http://sag.sagepub.com/cgi/doi/10.1177/1046878109334330.

Todd, Deborah. 2007. Game Design: From Bluesky to Green Light. A.K. Peters.

Ulicsak, Mary. 2010. Games in Education : Serious Games. Context.

Zetten, Martijn van. 2010. Developing serious games in higher education using the EMERGO methodology. Method Engineering. Ultrecht University. http://www.cs.uu.nl/wiki/bin/view/MethodEngineering/DevelopingSeriousGamesInHigherEducation. Date Accessed : 05/07/2011

Zyda, M. 2005. From visual simulation to virtual reality to games. Computer 38, no. 9 (September): 25-32. doi:10.1109/MC.2005.297. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1510565.

Annex I Questionnaire for Case Analysis

Case embedding
For which courses, curricula and institutions will it be used?
It will be initially used for the programming fundamentals course at the Faculty of Engineering of the University of Porto (FEUP). It could be open to other courses later.

Is it a stand-alone item or used with other instructional materials?
It will work as an stand alone tool for learning, but with shared material from the course.

Case content
What is the main complex cognitive skill?
Problem solving, planning, logical reasoning and analytical thinking

What prior knowledge and skills are expected for enrolled students?
Skills using computer programs
Skills on math courses
Skills on understating algorithms

What are physical locations in the case?
The laboratory rooms for the computer and informatics major at FEUP

What case characters are relevant?
The student as the player.
The professor and tutors as the editors, reviewers and guides.

What kind of activities do students need to perform for acquiring the main complex cognitive skill?
Analyze the situations presented in the game. Understand the requirements of each task. Plan the steps and develop code. Test codes input and analyze the outcomes.

Is redundant information provided, or is everything strictly needed?
Redundant information may be given. Two versions of the instructions for a quest to define it better.

How realistic and authentic is the case?
It will be as realistic as the exercises that are designed for it. It won't have a realistic scenario given on a company environment, but could have exercises developed to attain that scope.

If students can redo a case: will this be the same case or a variant?
The students can redo re-plays. They may be different depending on the randomization of the problems.

Can students undo former decisions?
Students can retry any of the quests until they have solved them correctly.

Are different learning routes and tasks for different students offered?
If several failed attempts are attained, a new path with easier exercises will be available. In some cases the student may choose the easier path.

What kind of cooperation is needed by students?
Playing through the game should guarantee that enough material is covered for the classroom test. Feedback given between them and the professor is contemplated to be disseminated outside of the game.

Do students have different case characters?
No, there won't be different characters in the first design of the game.

Do students have active roles?
Yes. They will have to play the game to cover all the course materials.

Do teachers have active roles?
They could have, as they have the option to create and modify game versions and in-game help material.

What aspects induce and sustain interest and motivation?
Taken from leisure games design, the scoring factor, the story reveal and visual aid are the aspects to induce motivation.

What unforeseen circumstances are incorporated?
Grade of copy between students, and to minimize it there is a random selection of problems. Not enough scenario related problem design to show the potential of multimedia learning.

Is competition incorporated?
Competition is only with the level and final score if it is seen outside from the game.

How do students get rewarded for excellent performance or behavior?
Optional extra quests and greater score points are the main rewards. The design of the game could bring up collectibles to award solving those difficult quests.

Students' progress
How do students discover not yet having acquired the main complex cognitive skill?
If a new path with easier pool of exercises is open, or they failed with several attempts to solve a quest.

How can students monitor their progress?
Under their profile they can see the quests that are already solved correctly, the percentage of advance in the game and the level and final score earned through their play through.

How is it checked if students have acquired the main complex cognitive skill?
Is summative assessment included and are its results used in formative assessment? Which students' progress figures are to be used by teachers during run time?

Contact with peers

Should contact between students be encouraged?
It could be encouraged but out of the game environment. They could aid each other on some tasks.

Should students see if peers are on line, when they have been on line?
Not at this time.

Can students compare their progress with peers?
Not in-game at this time.

Using media
Will existing material be used, is new material needed?
Existing learning material is used mainly for initial design of the quests and levels of the game. Existing problems could be used, though it is recommended to create new problems that align with the assets available in the game.

What media genres are used (e.g., interviews, docudrama, movie, animations)?
Text documentation, images, animations and voice podcasts could be used as explanatory items inside the game.

What media assets are needed and what are their costs?
Most of the text documentation could be taken directly from the course materials.
Images, animations, voice podcasts have to be made to improve the help documentation.
For the game design, new textures and models have to be constructed or acquired and adapted from available internet sites. Sound and music is also important for game immersion, tracks have to be designed and applied to the game environment.
There is an important amount of resources that have to be applied to this project to come as a structured and attractive tool for students. This part is a time consuming task and some assets could involve money costs.

Case delivery
Is the number of students within one run restricted?
The restriction depends on the capacity of the platform. As a single player experience it does not have a limit of players from a game design perspective.

When can students enroll for a run?
Students can first enroll at the beginning of the course. Then anytime during the course they can change profile up to ten times.

Is it possible to change the case after starting a run?
It is possible if the student starts a new profile. Maximum of ten profiles are considered.

(embedded) Support

How will technical support be provided?
There is technical support planned at this step of the project. A website could be uploaded to help students with troubleshooting and documentation. The academic staff could bring a team of designers to help with the usage and well behaving of the system.

How will support be provided for acquiring the main complex cognitive skill?
It will be provided outside of the game during classroom sessions or consulty interviews with academic staff.

IPR
Is it allowed for others to use the case?
It is only considered to be used only at FEUP at this time.

Are materials from other parties incorporated and what are their Intellectual Property Rights(IPR) arrangements?
Most of the materials should be originally designed inside the University, or through student activities.
3D models, textures and sound assets taken from internet could come from free databases, if other assets need to be taken they could be paid and the IPR would be different depending on the database.

Annex II Game Concept Document

Project Orion

Game Name :	Project Orion
Game Logo :	> project orion ();
Document Type :	Game Concept Document
Document Version :	V 1.0

Credit

Document Purpose:	Game Concept
Document Version:	1.0
Working Title:	Project Orion
Game Concept:	Enrique Kato Romo
Game Document Author:	Enrique Kato Romo

Sign-Off

Game Design Sign-off

CEO:	António Coelho
Lead Designer:	Enrique Kato Romo
Lead Programmer:	Ricardo Gonçalves

Introduction

Project Orion is a serious game whose main objective is to help students learn programming. It is a single player adventure game in which the player takes the role of Isaac, a space neo-archeologist, and uncovers mysteries of long lost worlds. Isaac is not alone, he is accompanied by his robot sidekick H4L, who allows Isaac interact with the alien system's artificial intelligence.

Project Orion's design will allow the player to practice and learn about programming concepts during gameplay.

Game Analysis

This is a general overview of the game.

Game Description	
Genre:	• Adventure - puzzle game.
Game Elements:	• Exploration • Collecting items • Removing obstacles • Story Comprehension
Game Content:	• Thriller
Theme:	• Sci-Fi
Style:	• Cartoony
Game Sequence:	• Linear Storylines
Player:	• Single Player

Game Reference	
Game Taxonomy:	• Fictional Game/Narrative
Player Immersion:	• Narrative • Mental
Reference:	• The setting and story has references from novels of Isaac Asimov and Arthur C. Clarke. • Controls and style referenced from World of Warcraft. • Gameplay mechanics referenced from other programming serious games.

Game Technical	
Technical From:	• 3D Graphics
View:	• Third person camera
Platform:	• Unity 3D • *Programming:* Any programming language
Device:	• PC

Game Atmosphere

Game Play

The game is accessed through the network. You have to input your username and password and the game will carry you to your lobby screen. There you can create several game profiles and see the scores obtained on previous plays. Selecting an unfinished profile will take you to the point or level where you were before leaving the game. A new file will start a new game that will direct through the story and instructions screens.

You are Isaac, a space neo-archeologist, who is in charge to do research on worlds that have or had any form of civilization, to improve knowledge of mankind. To aid you in your quest you have a robot companion named H4L, he can communicate with the systems that powered the life of these worlds. It is through him that you will get access to the system and recover the information of what happened with the civilization that populated the last world in your schedule.

Soon you will find that the levels are composed of a series of rooms that you have to explore. Each room has puzzles to solve and items to collect that will help you in the game or will give more information about the plot or about programming instructions. You have to reach the terminals scattered through each room to access the system, and then using your programming skills, communicate and solve the challenges of the room. You finish a level when you complete all the obligatory challenges and while there is no losing, you still have to be careful to get the highest score possible and continue the game.

Key Features

- Number of Levels

  6 Main levels defined by the contents of the course of Introductory Programming.

  The number of sublevels or rooms is defined during the creation of the game, depending on the academic objectives to reach and the materials to cover.

- Number of Enemies/ Characters

  2 Main characters

  1 Main enemy and boss

  5 Secondary characters

- Time of Game Play

  Undefined

- Replay ability

  It is possible to begin a series of new games at any time; there is a random function that will choose available problems from a selected pool, making each play through a little bit different from the last one.

- Audio Specifications

  Basic audio system

- Graphic Specifications

  Basic graphics card

- Device Compatibility

  PC with connection to internet, an internet browser and the Unity player installed

- Number of Players

  1 (Single player)

- Online Activities (high scores, etc.)

  Score board that shows how high you are related to other players

Selling Features

- Marketing Ideas

  Promote the game use through the introductory programming course

- Consumer Group

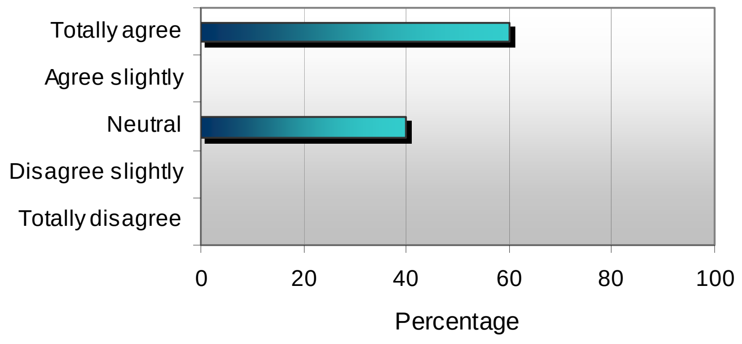
  Students of the first year of the major of Informatics at the Faculty of Engineering of the University of Porto

- Unique Features

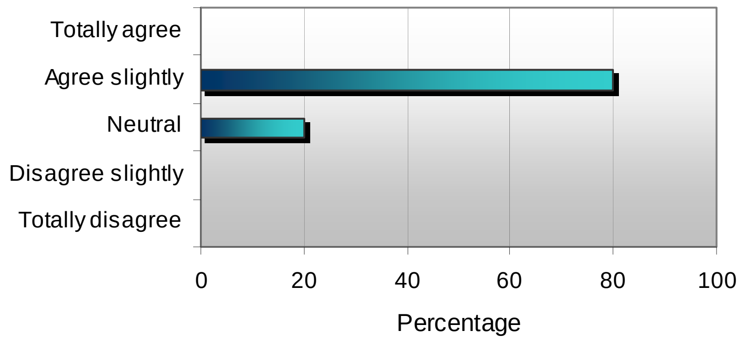
  Programming language-free game
  Editable platform to create, fast and easy, new versions of the game
  3D Environments and visual explanations of concepts
  Enticing storyline
  Score leader boards

Annex III Results of the Testing Questionnaire

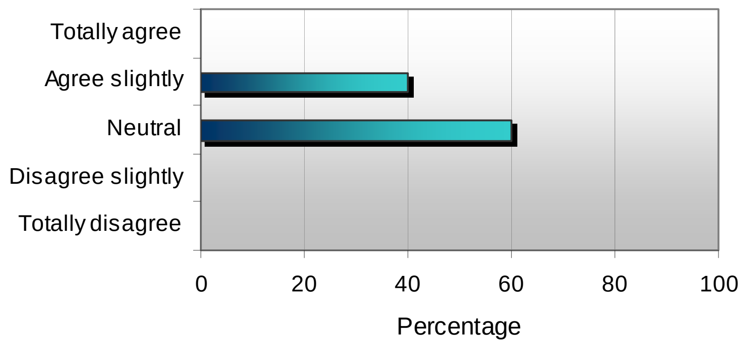
The response time between solution submision is acceptable



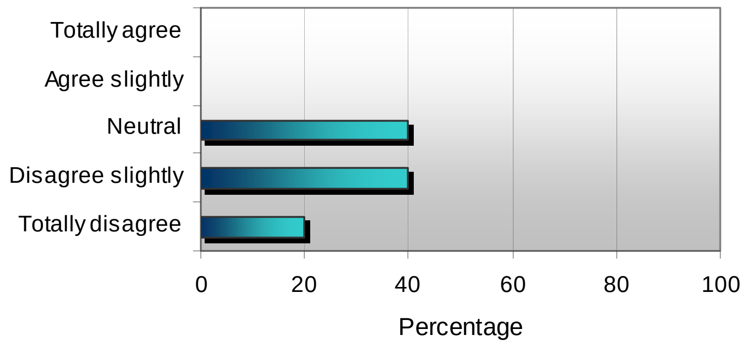
This game is a good complement to learn programming



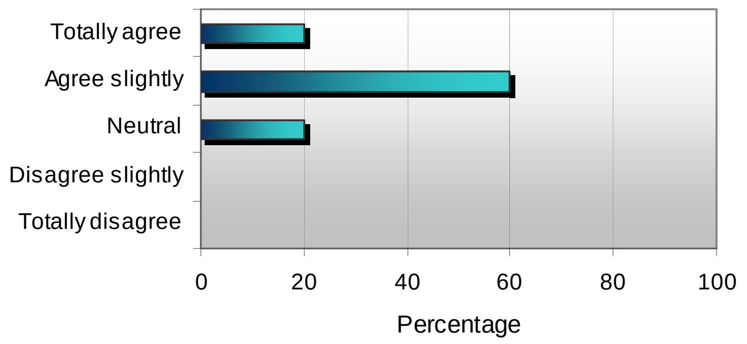
The error messages of code submissions should present more information about the error



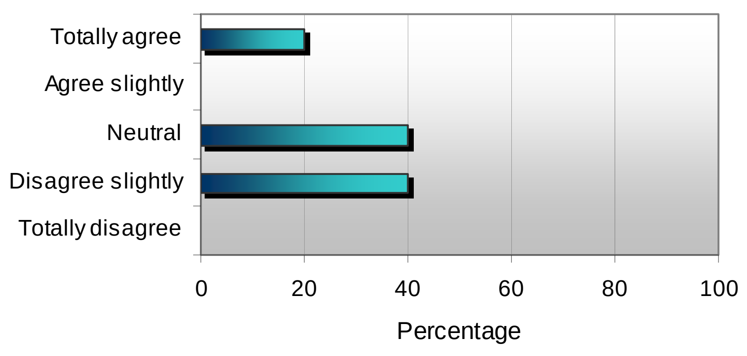
Making submissions of code should only be done on the game terminals , and not through the quest menu accessible at any time.



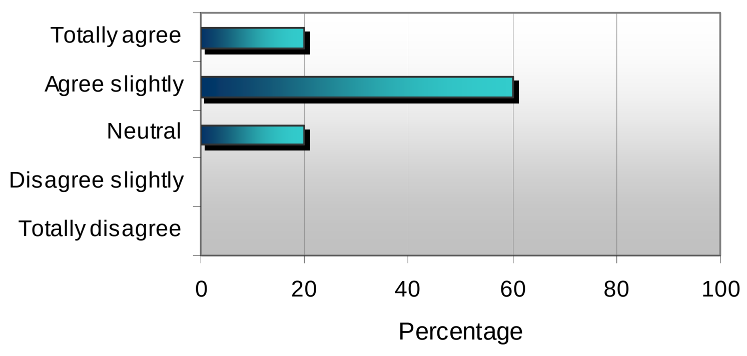
The game is intuitive and could become a fun option to motivate learning



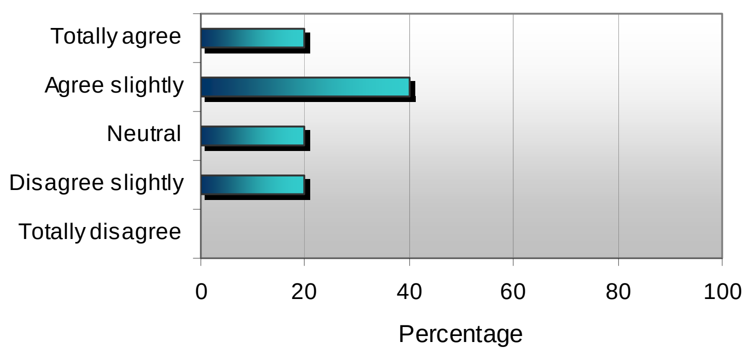
The number of objects to interact with is enough



The interfaces are easy to use



Sound effects and music would improve the game experience


Totally agree
Agree slightly
Neutral
Disagree slightly
Totally disagree
Percentage