May's Journey:
A serious game to teach middle and high school girls programming

by

Chaima Jemmali

Zijian Yang

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Interactive Media & Game Development

April 2016

APPROVED:

Dr. Jennifer deWinter, Major Advisor

Prof. Dean O'Donnell

Dr. Mark Claypool

Abstract

May's Journey is a game where you help a video game character, *May*, finding her friend and repairing the broken game world. This is a 3D puzzle game in which players solve an environmental maze by using the game's pseudo code to manipulate the environment. The game is aimed at 12 to 18 year old girls and the purpose is to attract them into Computer Science fields by teaching them basics of programming by focusing on logics and concepts while still asking them to type simple instructions in our programming language. Players do this in a compelling environment, with characters they can identify with, embedded in a relevant story.

Our design process was based on our research on young female preferences in games and current teaching techniques for programming. Each decision we made whether for the teaching content, the art style, or the game mechanics and the techniques used to develop this game are motivated by the goal of making programming more appealing and interesting for girls. For this we developed our own pseudo-code language in order to provide an interface that bridges the gap between drag and drop approach and real programming and introduce typing as part of the experience.

We tested our game with 10 teenagers aged from 14 to 17 years old for educational content. We were pleased to see how engaged with the game they were. Overall, the testing results were mostly as expected. The players liked the game (rated 4.8 out of 6) and all of them wanted to play more of it. They all felt that they learned something and 8 of them expressed the will to learn more about programming. Unfortunately, the sample of players is too small to generalize our results so we plan to take the feedback into account, iterate and test it again with a larger study group and get conclusive results.

Working on this project has allowed us to understand the importance of iterative design and early playtest feedback. We have also learned the importance of tutorials in games and how that might completely change the users' experience. Finally, a crucial point was the importance of the UI helpers and targeted feedback in serious games.

Acknowledgement

We would like to sincerely thank Professor Jennifer deWinter, Professor Dean O'Donnell and Professor Mark Claypool for their guidance, understanding and patience. Their mentorship were the reason that we were able to finish this project.

We would also like to thank Professor Alexandrina Agloro. Without her help, we wouldn't have been able to test our game with our target audience.

Both of us would like to thank the other person in this team. It is the comprehension, supporting and compromise we provided to each other that made this collaboration successful.

We would also like to thank everyone who participated in the user study session, iteration testings and user testings. Thank those who provided us feedback, suggestions and technical assistances. We would not have accomplished this project without your generous help and spiritual encouragement.

Table of Contents

Chapter 1: Introduction 1

Chapter 2: Literature Review 7

2.1. Targeted Game Design 7

2.2. Games for Girls 9

2.3. Coding Games 9

2.4. Teaching Content 14

2.4.1. Teaching methods 14

2.4.2. Making programming more appealing to girls 16

Chapter 3: Game Design 18

3.1. Gameplay 18

3.2. Story 20

3.3. Characters 22

3.3.1. May 22

3.3.2. The Oracle 23

3.3.3. The cat 24

3.4 Visual Design 25

3.5 Level Design 33

3.6 Additional Gameplay Elements 37

3.6.1 Collectible gems 37

3.6.2 Secret levels 38

3.7 Teaching Content 39

3.8 Programming language design 42

3.8.1 Defining Grammar 42

3.8.2 Parser 44

3.8.3 Interpreter 46
3.9 Game UI 48
Chapter 4: Evaluation 50
4.1 Iterative Design 50
4.2 Targeted Testing 51
4.2.1 Methodology 52
4.2.2 Results and analysis 54
Chapter 5: Conclusion and Future Work 61

Table of Figures

Figure 1.1 Percentage of women in Computer Science compared to other major fields [National Science Foundation] 2

Figure 2.1: CodeCombat 11

Figure 2.2: Codemancer 12

Figure 2.3: Machineers 14

Figure 2.4: The process of teaching programming to beginners 16

Figure 3.1: A representative graph of the Hero's Journey's stages 20

Figure 3.2: May visual design (version 1 to the left, version 2 to the right) 23

Figure 3.3: May bone animation 23

Figure 3.4: The Oracle visual design references 24

Figure 3.5: The Oracle visual design 24

Figure 3.6: The Cat visual design (version 1 to the left, version 2 to the right) 25

Figure 3.7: The Cat bone animation 25

Figure 3.8: Alto's Adventure visual design 27

Figure 3.9: illustration by Jona Dinges (Jona Dinges, 2015) 27

Figure 3.10: Monument Valley visual design 27

Figure 3.11: Low poly concept design 28

Figure 3.12: level_1_1, first version 29

Figure 3.13: level_1_1, second version 29

Figure 3.14: level_1_1, final veion 30

Figure 3.15: Crystal Hallway 30

Figure 3.16: Desert Chamber 31

Figure 3.17: Windy Garden 31
Figure 3.18: Lounge Bridge 32
Figure 3.19: Water Palace 32
Figure 3.20: Game Design level 0 33
Figure 3.21: Game Design level 1 34
Figure 3.22: Game Design level 2 35
Figure 3.23: Game Design level 3 36
Figure 3.24: Game Design level 4 37
Figure 3.25: A screenshot of May collecting gems 38
Figure 3.26: Some screenshots from how to get and use the secret code 39
Figure 3.27: TokenIcer interface 45
Figure 3.28: Class Diagram ..... **Erreur ! Signet non défini.**
Figure 3.29: CodeCombat UI design 48
Figure 3.30: May's Journey UI design 49
Figure 4.1: Some paper prototypes of our level design 50
Figure 4.2: 2D prototype of level one with the coding interface 51
Figure 4.3: Answers to the question "How often do you play video games?" 54
Figure 4.4: Comparing feelings about programming before and after playing the game 56
Figure 4.5: Comparing feelings about programming as a career before and after playing the game ..... **Erreur ! Signet non défini.**
Figure 4.6: Feelings conveyed by the game 59

Chapter 1: Introduction

This paper is about "May's Journey", an educational game that teaches programming through puzzle solving and storytelling. This game is targeted at middle school and high school girls as a way to introduce them to programming and computer science in general.

Programming has become an essential skill. Studies predict that by 2020, employment in all computer occupations is expected to increase by 22% (Thibodeau 2012). By that that time, there will be around 1.4 million computing jobs in the U.S (Landau 2015). However, statistics show that in 2013, only about 26 percent of computing jobs in the U.S. were held by women, (Peck 2015; Khanna 2013). This number is very low considering that women represent 57% of the workforce.

This hasn't always been the case. In a "Cosmopolitan" story back in 1967 Adm. Grace Hopper, a computer science pioneer, explained that "It's just like planning a dinner," stating that women are just 'naturals' at computer programming (Lewis 2011). At that time, 11 percent of computer science majors were women. In the late 1970s, the percentage of women in the field approached and exceeded the same percentage as today which is 25 percent. The portion of women earning computer science degrees continued to rise steadily, reaching its peak 37 percent in 1984. By 2006, the portion of women in computer science had dropped again to 20 percent (Lewis 2011).

In the graph (Figure 1.1) we can observe the number of women studying computer science was growing fast until 1984, where something changed. The percentage of women in computer science

flattened, and then plunged, even if the share of women in other technical and professional fields kept rising (Henn 2014).

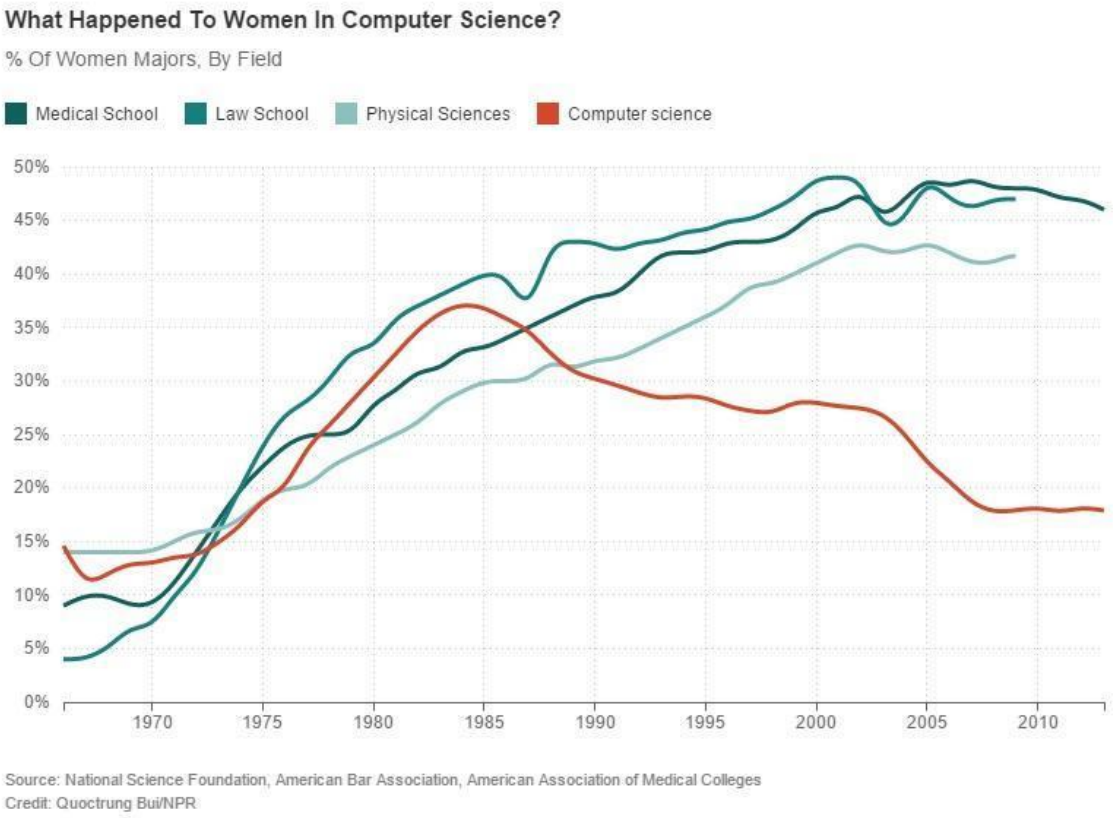

Figure 1.1 Percentage of women in Computer Science compared to other major fields [National Science Foundation]

Programming used to be a field that attracted women, even when society was substantially less friendly to the idea of women pursuing lifelong careers in the sciences. Like many industries after World War II, the industry faced a dire shortage in programmers and systems analysts, roles that involved designing programming instructions. Computer science needed manpower, and women counted as manpower. This need led to many undergraduate women flocking to computer science classes. Men had not yet entered computer science in significant numbers. They, too, were just

beginning to respond to industry demand and had not yet dominated the field (Lewis 2011; Henn 2014; Fessenden 2014).

All these facts and numbers throw a question on how women stepped back from a field where they used to be present. There are many speculations about this phenomenon, especially about what happened in 1984. Many studies relate this drop to the rise of personal computers (Fessenden 2014, Conditt 2014). In fact, in the early 80's Commodore 64s, Radio Shack TRS-80s and others were marketed to boys. In 1983, Asteroids was launched in Vectrex and the commercials show two boys playing the game. That was only one example but many other commercials for games were targeted at boys, so computers had entered the category of "boy's toys" (Conditt 2014) and computerized technology in general had became masculine (Kocurek 2016). Other fields picked up on the trend, with movies (*Weird Science 1985*, *Revenge of the Nerds 1984*, *War Games 1983*), journalistic pieces and books that painted men, not women, as computer geeks (Fessenden 2014). That period was the starting point of what we call the "tech culture" and the definition of "geek".

At the high school level, the number of women and men studying computer science is equal, however, as they progress through the pipeline of more advanced study, the percentage of women drops at each educational level (Frenkel 1990; O'Rourke 1993; Pearl et al. 1990). In 1995, UCLA senior researcher and author Jane Margolis conducted a study about women in Carnegie Mellon's Computer Science path. She found out that half of the women ended up quitting the program. Most surprising is that more than half of those who dropped were doing well. It was just a belief that no matter what men are just better at this. (Camp and Gurer 2001)

The underrepresentation of women hurts the technology companies themselves since half of their users are females. Having more women in the workforce will help represent this underserved part of the market. This is even more relevant in the world of video games where half of the gamers are females (Statista 2015; Jayanth 2014; ESA 2014) while the games that have historically enjoyed the biggest budgets and the highest returns are *Call of Duty* (2003-2015, *Activision, Tencent Holdings, Square Enix, Aspyr Media, Konami*), *Grand Theft Auto* (1997-2013, *Rockstar Games*), *Halo* (2001-2016, *Microsoft Studios*) and other similar games that are aimed largely at a young male demographic (Alexander 2014).

A study from 2013 was interested in the time spent playing for different devices. It shows that men spend 32% of their time playing on gaming consoles against 19% for women. However, women spend more time than men playing on smartphone, tablet or laptop. For other devices, the percentage was almost the same (Deloitte 2015). Considering the types of games by gender, a study that involved 341 people involving 252 male and 89 female, indicated that more men (84.92%) than women (46.07%) reported that they played violent games (Phan et al. 2012). The study also reports the genre of games played by each gender. Male gamers reported that they tend to play games from the Strategy, Role Playing, Action, and Fighting genres more often, while female gamers lean towards games from the Social, Puzzle/Card, Music/Dance, Educational/Edutainment, and Simulation genres (Phan et al. 2012).

Games, being a gateway to programming motivations, it's important to have more computer games aimed at girls that will have them exposed to computers at an early age which is very important. In fact, women who succeed in computer Science are likely to have been introduced to the field early in life (Camp and Gurer 2001). Games and gender work as a cycle. If the games

feature more males, they are more likely to attract more young males to play. Those males grow up and become more likely to become game makers than women (Williams et al. 2009). Since women and men are interested in different genres, we expect the industry to take into account this difference in the design of their games. However, diversity and inclusion is not always present in current video games. Of the 8725 characters surveyed in "The virtual Census" only 15% were female and less than 11% of player-characters were female (Williams et al. 2009). The result of this lack of diversification is not only cloned games but cloned creators, cloned characters and attempt to clone players (Potain 2010).

Getting women into computer science related industries such as the video game industry starts by retaining the interest of girls starting from middle school (Camp and Gurer 2001). At that age, girls are old enough to intellectually handle the concepts of programming, and are young enough to not have already decided against pursuing math and science subjects, specifically computer science (Kelleher 2006). We are seeing more and more tech camps for girls that will help demystify this field and make it look more accessible and more welcoming for women. We can name "Girls Who Code", "Black Girls Who Code", "Geek Girl Camp", "Microsoft Digi Girl and High Tech Camps" and many more. In the same spirit, the idea for this project arose.

Our intention is to design and program a video game that will teach middle school and high school girls basics of programming in a compelling environment with characters they can identify with, embedded in a relevant story. We need to consider major points in our design: How will this game draw girls' attention and form their enthusiasm for programming? How will the game teach programming in a meaningful way? What contents does this game need to teach according to the learning objectives?

In this report, we start by the literature review where we look at designing games for girls, at different ways of teaching computer science and also at previous games that teach coding or programming logic. In the second chapter, we discuss the design choices for the game and how it relates to our research. The third chapter describes our testing protocol and our results. Finally, we discuss our findings and suggest the future development of the game.

Chapter 2: A Review of Games for Girls, Educational Games, and Teaching Programming

Before designing our game, we need to understand what our target audience looks for in a game. We also need to evaluate other games that teach programming and how previous work can influence our design choices. Finally, we want to research the most efficient ways to teach programming to beginners.

2.1. Targeted Game Design

We conceived an educational game aimed specifically at girls rather than to all kids. The reason is that there are not enough "good" games out there that are designed for girls. In an interesting study about the effect of designer's stereotypes on the software they design, educators with programming skills were asked to design different teaching softwares; one designed for 7th grade girls, one other for 7th grade boys and a last one for 7th general students. The software designed for girls resembled "learning tools" while the ones for boys and for general students were very similar and more game-oriented (Huff & Cooper 1987). The result suggests that even when designing a game for both genders, programmers have unconsciously boys in mind, which creates an intrinsic bias in the tools. On the other hand, when tasked with designing games for girls, the programmers lacked an in-depth understanding of girls' preferences in games.

Girls value narrative and characters, but will also play violent games with strong narratives and characters (Laurel, 2000). And while social beliefs suggest that girls just disliked violent games, Brenda Laurel (2000) points out that girls didn't mind violence as much as they disliked

the lack of good narratives and characters. Those misunderstandings and misbeliefs come from the fact that women are not well represented in the process of design and creation of video games. The result is that we have been caught in a male dominated industry that creates content for males (Williams et al. 2009). In a study that researches gender difference in designing game, they created same gender teams of girls and boys (5^{th} and 8^{th} grade) that they taught space content and game design for two weeks and they asked them to envision a game concept that would make kids "just like them" become interested in space exploration. Promos of the game were then showed to middle school children who rated them without knowing the gender of the design teams. The results showed that Girl games were perceived as better for learning. Females liked the games designed by girls better and males liked the games designed by boys better. Also, females rated girl games closer to gender neutral than boy games, but interestingly all of the games were perceived as for boys or for everyone rather than for girls (Heeter et al. 2004).

The creation of a more inclusive content for girls will make them feel more comfortable with being identified as gamers and later as programmers. This will potentially help breaking the cycle and resulting in more diverse video games and software content in the future.

In the next section, we look into what elements girls like in games and what are the gameplays they enjoy the most.

2.2. Games for Girls

A number of published research articles concerning preferences agree on these points about pre-teen and teen girls' preferences on games.

- Myst environment with no guided paths to follow, no points, and no time limit. (Leach 2013)
- Graphic, audio and atmosphere are considered as highly important.(Leach 2013)
- Untraditional design paradigms which strictly emphasis on levels and tasks.(Leach 2013)
- Strong, relatable characters that provide someone to empathize with. (Cassel and Jenkins 2000)
- Complex and rich relationships between characters.(Cassel and Jenkins 2000)
- Puzzle-solving skills rather than their eye-hand reflexes.(Cassel and Jenkins 2000)
- Collaboration over competition.(Leach 2013)
- Social, Puzzle/Card, Music/Dance, Educational/Edutainment, and Simulation genres. (Yee)
- Experiential path and things happening right away. (Cassel and Jenkins 2000)
- No starting over when dying (they think it's stupid and intolerable) (Cassel and Jenkins 2000)
- No interest in beating the clock (Cassel and Jenkins 2000)
- No steep learning curves just to be able to say they achieved or learned something (Cassel and Jenkins 2000)
- Inner world construction (love, caring, nurturing behavior) (Cassel and Jenkins 2000)

2.3. Coding Games

Although we didn't find a coding game with that target audience in mind, there have been a number of coding games made available for public and scholastic use since Logo Programming (1967) added a game-like turtle to its educational interface. Based on our review, we provide a

comparative matrix (Table 1) of the following games: Codemancer, CodeCombat, Machineers, SpaceChem and Codewars. This table depicts the game genre, the language used to teach programming, target users, teaching methods (using real code or coding concept) and a small description of the games.

	Game Genre	Language	Target Users	Uses a programming language?	Introduction
CodeCombat	2D RPG	java, python	13+	yes	builds on older learn-to-code systems level and task based
SpaceChem	2D puzzle	programming concept	10+	no	using programming concepts like in-order execution, loops, branching, and subroutines task based
Codewar	puzzle	CoffeeScript, JavaScript, Python, Ruby, Java, C, C++	14+	yes	solving programming tasks more serious coding and less entertaining good evaluation system multiple solutions
Codemancer	2D RPG	programming concept	9-14	no	Fantasy story and female protagonist, story about a girl trying to grow up and do well despite incredible obstacles.
Machineers	2D puzzle	programming concept	8-14	no	construction puzzle game metaphors

Table 2.1 Comparative Matrix for existing coding games

CodeCombat is a well-polished traditional 2D RPG game whose target players are exactly pre-teens. In their official website, the development team specially mentioned that they had

considered how to draw players' attention and it seems they accomplished this goal according to users' feedback.

The game builds on older learn-to-code systems such as Rurple and Karel. Commands have to be typed correctly to control the avatar, and incorrect code will cause it to lose hit points and eventually die. In each level, the player is assigned a set of tasks. The player is gradually introduced to new commands like loops, conditionals and variables. CodeCombat begins with a smooth learning curve well suited to players with no coding experience. As the player progresses, the tasks involve more complex programming concepts. Most importantly, the levels themselves become more complex due to more possible interaction with the objects in the game world: fences can be built, fire traps can be set, enemies can be lured into minefields, special weapons allow special attacks with a cooldown timer, etc.

However, CodeCombat is still a gender-neutral game for all pre-teens with a male incline. When we take this as a reference, we need to distinguish designs which are not popular amongst girls and replace them with girl-centered design.

Figure 2.1: CodeCombat

Codewars is a more mature version of CodeCombat. Students aren't guided through lessons, but instead confronted with programming tasks. In contrast to CodeCombat, the teaching must happen before Codewars is used, or a student must have the skill and self-discipline to learn necessary coding skills other ways.

In Codewars, each programming task is in reference to a Japanese martial art called Kata. They include a short task description, a set of input data, and the desired output data. The student is tasked with writing a function in his preferred programming language to transform the given input into the desired output.

It’s not actually a perfect reference for our game design because its target players are older than pre-teens and its design is more serious and less entertaining. The design emphasis on learning more than entertaining so it’s not good for drawing pre-teen girls’ attention. However, There are two elements that we can take from it. First, Codewars’s mechanic is good for evaluating how players managed coding skills so that we may use this in our user testing design. Second, in later stage of Codewars, it tells players that for problems there can be countless different solutions. It’s a fascinating idea and we may also implement this concept into our design.

Figure 2.2: Codemancer

Codemancer will be released in fall 2016 and will be available for PC, Mac, iPad and Android Tablets. The game is publicized as an educational game teaching the magic behind programming designed for 9 to 14-year-olds. Players will code their way through a fantasy world full of rival sorcerers and their minions. (Codemancer website)

The game is trying to be as inclusive as possible with a gender-neutral fantasy setting, a female protagonist and a narrative backbone. 'Codemancer' features a moving story about a girl trying to grow up and do good despite incredible obstacles. Aurora's Father's life is in her hands, and that's a lot of pressure for a child just learning to be independent. (Codemancer website)

Since the game is not out yet, we can't really judge the teaching content within the game. However, from the interface and the interactions portrayed in the videos we can infer that the players use visual programming and blocks to drag and drop as opposed to typing real code like in CodeCombat for example.

Machineers is a puzzle game where you are equipped with a toolbox full of gears, belts, and electric wires that you use to build and repair crazy contraptions. Each machine you fix, brings you closer to becoming a true Machineer. The environment is grim and resembles a rusty broken city where only robots live. Despite the female protagonist, the graphic design doesn't seem very inviting for young females. To solve the puzzles, we use visual programming and it's hard to judge how successful *Machineers* is at "teaching" programming.

Figure 2.3: Machineers

Although some references are masculine and not always attractive for girls, we still can extract some elements from these games and apply these to “May’s journey”. For instance, the teaching content and sequence in CodeCombat, the evaluation system in Codewar and the storytelling and protagonist design in Codemancer. Also, in our game, players can manipulate characters or objects using both coding and clicking. In this part, we can use SpaceChem and Machineers (These two games uses metaphors to teach programming concept) as our references.

2.4. Teaching Content

In our game, we are not only trying to teach programming but are also trying to make programming more interesting and more attractive to girls.

2.4.1. Teaching methods

Teaching programming is difficult and there have been many research papers from many different perspectives devoted to the topic during the past couple of decades (Robins et al. 2003;

Pears et al. 2007), but there is still no consensus on what is the most effective way to teach programming (Vihavainen et al. 2011)

Many researchers seem to think that the problem is not to learn the syntax or semantics of individual language constructs, but to master the process on how to combine constructs to meaningful programs (Winslow 1996; Spohrer & Soloway 1986; Robins et al. 2003). They believe it's more valuable and more efficient to learn the logic of programming and understand the algorithms and the code design which is similar in all programming languages before learning the syntax specific to one programming language.

More and more drag and drop programming softwares are emerging. Scratch or The Hour of Code use blocks to introduce kids to coding. Learning through playing is considered a great way to introduce kids to programming and make it seem more accessible. In fact, to those unfamiliar with it, programming can seem very mysterious and difficult to learn. This idea can form a significant barrier to getting more kids interested in CS (Ritchie 2015).

Mark Engelberg, former NASA programmer and inventor of our newly released coding game 'Code Master' has used Scratch in his classroom and agreed on its very positive impact. He used it in his class and has noted how kids got excited about programming and how this kind of software has helped them to think of themselves as creators of programs and not just consumers. However, Scratch and other great tools won't make the kids programmers or give them any knowledge of how programming languages work (Ritchie 2015).

Between real programming languages being too syntax heavy for beginners, and block-based programs that don't give any knowledge about programming languages, there needs to be a bridge to fill the gap and facilitate the transition between drag and drop coding and real coding.

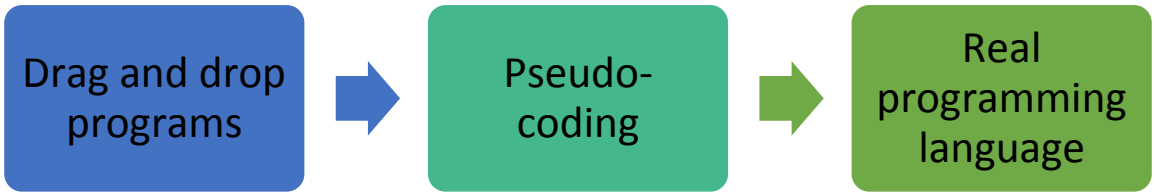

Figure 2.4: The process of teaching programming to beginners

2.4.2. Making programming more appealing to girls

As already mentioned earlier in this section, we want to form girls' enthusiasm for programming. We want them to feel more interested in learning programming beyond the game.

In an interesting paper that focuses on how storytelling can make computer programming attractive to middle school girls (Kelleher 2006), the study conducted was comparing the experiences of girls introduced to programming using two different versions of Alice. Alice is an innovative programming environment that supports the creation of 3D animations. The software provides tools and materials for teaching and learning computational thinking, problem solving, and computer programming across a spectrum of ages and grade levels. (Alice website)In the study, some Participants used Storytelling Alice while other used Generic Alice. For learning basic programming concepts, the results were similar for both groups. However, users of Storytelling Alice showed more evidence of engagement with programming. Storytelling Alice users spent

42% more time programming and were more than three times as likely to sneak extra time to continue working on their programs (Kelleher 2006).

It is also important for girls to know that coding is creative. Pamela Fox, co-organizer of the San Francisco Chapter of "Girl Develop It", a series of workshops to teach women about web development, likes to emphasis the incredible power of creation that coding can put in the hands of young women (Schwartz 2013).

In this chapter we have discussed current programs and games that teach programming. We have also looked into current programming teaching practices and their advantages. We will use these findings in the next chapter where we present our design choices for the teaching content, the environment, the graphics, the story and the characters.

Chapter 3: Game Design

May's Journey is a 3D puzzle game where solving puzzles is done through programming. The main character "May" finds herself trapped in the game world separated from her friend. She asks the player for help and that's how the game starts. The game involves two main phases; exploration and coding and both phases are closely linked together.

3.1. Gameplay

This is a 3D puzzle game where solving environmental mazes is done through programming. We have chosen this game genres because exploration games usually allow to create rich stories with multiple game characters and NPCs. We have chosen to link the programming with puzzle solving because that's what appeals more to our target Audience. Finally, we chose a 3D environment because it enriches our puzzle possibilities and allows us to use lights and shadows to create attractive environment for girls.

The win condition of the game is to fix the game world using the master code.

There is no real lose condition in the game, unless a player gets stuck and doesn't know how to solve a puzzle.

There are two phases in the game:

- An exploration phase with a top down view where players walk around with the hero character and interact with other objects and characters in the game.
- A coding phase that will change the interface of the game, splitting the screen into two major parts; one for coding and one for seeing the feedback of your code. In the coding part, players solve puzzle that will help them advance throughout the game. The

programming problems will be related to the story and they can actually find hints during the exploration phase.

We have four main levels in our prototype. This levels will be explained in detail in the level design section of this Chapter.

Level 0: players are introduced to the game world, they interact with and NPC (the Oracle) for the first time and they interact with a coding interface for the first time.

Teaching content: Main function, simple instruction, movement

Level 1: players need to move more blocks in each sublevel. They have also new game mechanic introduced like pressure plates, locked doors, keys and multiple coding interfaces. In this level, players can also find a secret code that leads them to a secret level.

Teaching content: Main function, multiple instructions, order of instructions, movement

Level 2: players discover again new game mechanics. This level introduces the wind and the fire tiles. The wind blocks the way but can also be used to light the fire tiles. When all the fire tiles in a sublevel are lit, the key to open the level 3 door appears.

Teaching content: Main function, multiple instructions, order of instructions, movement, rotation

Level 3: players encounter a trapped cat that they need to save a cat by solving some more difficult puzzles. In the next levels, the cat can be used to solve puzzles.

Teaching content: Main function, multiple instructions, order of instructions, movement, rotation, loops, cat manipulation

3.2. Story

Our Story follows the narrative pattern of the Hero's Journey. The pattern appears in drama, storytelling, myth, religious ritual, and psychological development. It describes the typical adventure of the archetype known as The Hero, the person who goes out and achieves great deeds on behalf of the group, tribe, or civilization (Vogler 1985).

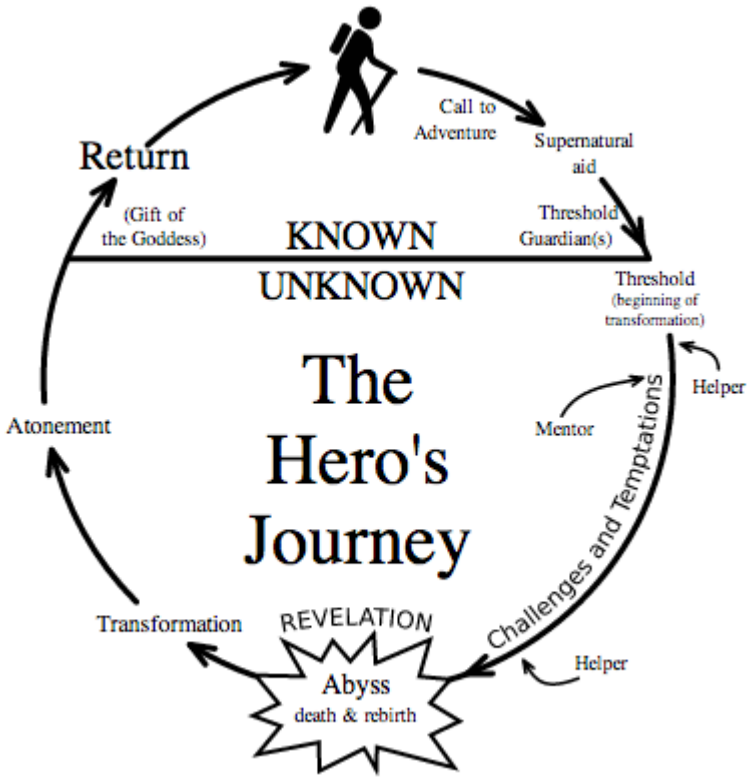

Figure 3.1: A representative graph of the Hero's Journey's stages

In our story the hero is "May", a girl who asks for the player's help because the world she is living in is falling apart. She got separated from her friend and she needs to find him before it's too late. She asks the player to follow her into the Game World "The unknown" to begin the journey. Entering the game world as May, the player makes the first encounter with the oracle "The guardian". She will tell them a little bit about this new world. It is world that was coded to

be in perfect balance but now the balance is lost and the paths are crumbling, and they need to code their way through it. This will mark the beginning of the transformation.

During the challenges and temptations stage, the players will hone their programming skills, solve increasingly complicated puzzles and discover parts of the story through exploration. The parts of the story found are not always in order. Some of them that will arise new questions and theories and will aliment a will to know more.

For example in level 1_4, May finds an ancient book talking about a master code that is both able to save the game world but also destroy it. At that point, the player doesn't know what it is, how to use it or where it comes from, but they know it's important.

During the Journey, you will meet with different game characters "helpers" that will help her through her quest or give her advice in exchange for her help.

The first encounter is a cat that informs her about the danger that threatens the village. He needs help to save the villagers from game viruses that are destroying everything on their passage. This is the first time the viruses "the enemy" are introduced. The player doesn't know yet who are they or what motivates their behavior. Upon accepting to help saving the village, the cat tells May that he might know someone who will help her find Juno, her friend.

Keeping the mystery and revealing one small part at a time, motivates the players to play more, to discover what happened and to encounter more game characters. May encounters many obstacles and meet with multiple characters that will reveal more and more about what is happening to the world.

Halfway through the game, when she finally meets Juno, we hit our revelation when she discovers that if she just escapes the game world, she will leave this world to decay and abandon all the characters to a terrible fate.

Therefore, we enter the transformation stage where they both decide with the help of other characters to find the master code before the viruses, fix the game world and save all the game characters.

3.3. Characters

3.3.1. May

May is the main character in this game. She is a pre-teen girl, the same as our target audiences. Our intention is that May could be the projection of players themselves so that they will have more emotion connection and empathy on the character. (PC Plus, 2010) In the gameplay, players can use the WASD keys to move May.

According to players' feedback, the visual design of the first version is too dark. We made her dress lighter, bright-colored and saturated in the second version.

Figure 3.2: May visual design (version 1 to the left, version 2 to the right)

Figure 3.3: May bone animation

3.3.2. The Oracle

The Oracle is an old and knowledgeable lady. This NPC acts as a tutor of May. She appears in the beginning of some levels and give some tutorials to May. These tutorials include the background story, coding knowledge and new game mechanics. The art style of the Oracle is a combination of ancient Egyptian priest and ancient Chinese clothing.

Figure 3.4: The Oracle visual design references

Figure 3.5: The Oracle visual design

3.3.3. The cat

The cat is designed as a companion of the May. As our research indicated above, a cute sidekick which follows players all along this game will enhance the emotional bond between players and the characters. Also, we designed the cat to build some levels which need more than one character to solve. In the gameplay, May can manipulate the cat to help her with some puzzles. Players need to control the cat using code. We built the first version according a real cat, however, in the user testing, the feedback shows that players thought it's too realist and not cute enough.

Thus, we referred to the real cat and Pokemon when we design the cat and added pointing ears, big tail to make it cute and unrealistic.

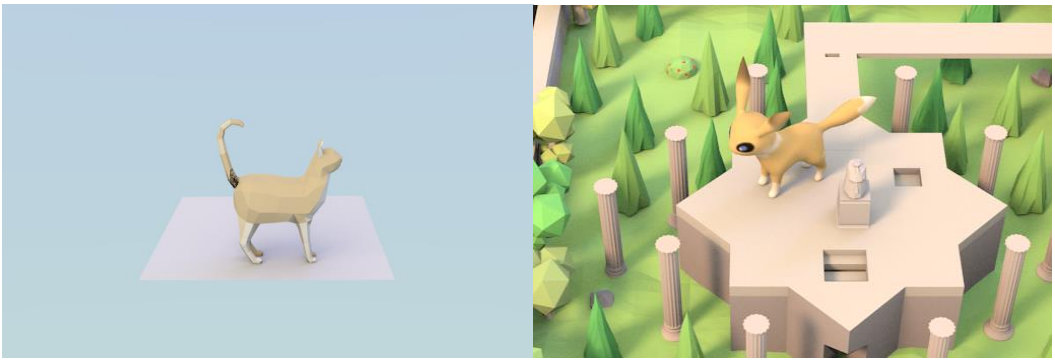

Figure 3.6: The Cat visual design (version 1 to the left, version 2 to the right)

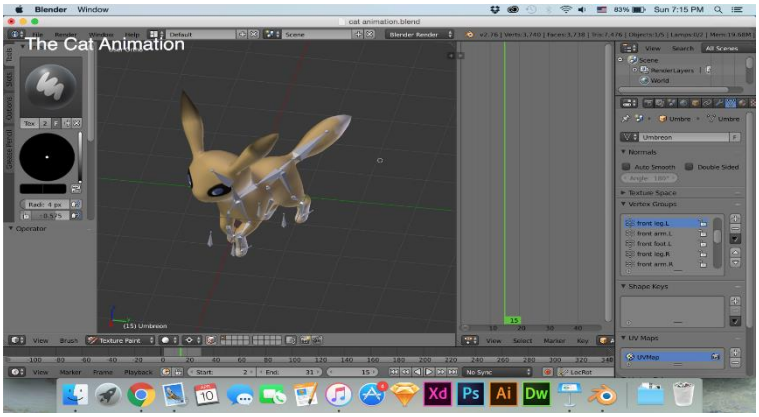

Figure 3: The Cat bone animation

3.4 Visual Design

As we mentioned earlier in this paper, there are some crucial differences between girls' and boys' preference in game design. Girls placed a high emphasis on graphic quality. According to the study's article, "Girls' Preferences in Software Design: Insights from a Focus Group," voice quality, music, and atmosphere were all elements that the players described as "highly important" (Leach 2013).

According to our research, girls have distinctive preference on graphic styles. Generally speaking, comparing to boys, girls prefer tender, graceful, serene and less aggressive illustration as well as High-lightness, warm and moderate saturation color scheme.

In the article "Design Stereotypes: Masculine and Feminine Design Technique" (Cousins 2012), the author did some quick breakdown on design elements. The following visual elements are considered as Feminine:

- Babies, puppies and overall cuteness
- Flowery landscapes and trees
- Food
- Clothing and shoes
- Script fonts and Curved-edge serifs
- Cursive

Also, we researched some games with graphic designs highly evaluated by female players or with visual designs corresponding to our research on female preferences. We chose Alto's Adventure (Harry Nesbitt, 2015), Jona Dinges' illustration and Monument Valley (Jem Yoshioka, 2014) as our reference.

Alto's Adventure is a 2D screen-scrolling game, the minimal graphic design, gentle tone and soft music compose an elegant and delightful game world. The visual assets are simple geometry graph. It's an efficient and smart way to deliver a serene and clean experience which corresponds female preference.

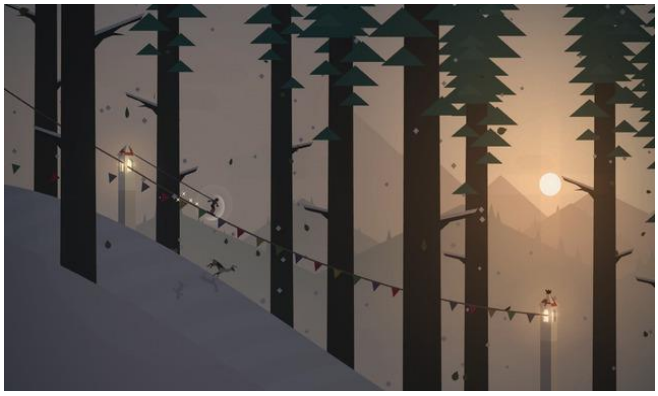

Figure 3.8: Alto's Adventure visual design

We also referenced Jona Dingse's illustration. This group of works managed to create a mysterious atmosphere using 2D low poly art style. We can see how she use colors to achieve the continuous relationship between lights and shadows.

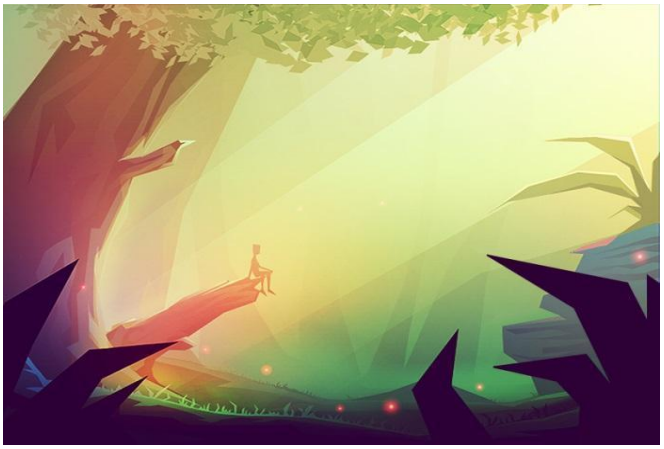

Figure 3.9: illustration by Jona Dinges (Jona Dinges, 2015)

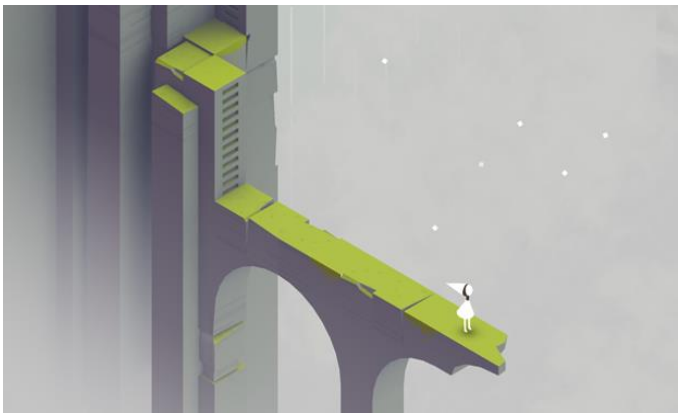

Figure 3.10: Monument Valley visual design

Monument Valley is a geometry puzzle game where low poly and minimalist are regarded as the fundamental art styles. It's designed for everyone; however, it earned excellent reputation among female players. Besides the subtle storytelling skill and elegant storytelling skill, the game art design is the most admirable aspect of this masterpiece. We learnt a lot from how the designers use color scheme and how to use low poly properly to build a 3D game world.

These games are inspired by Myst (Mark J.P. Wolf 2011). They use the effect of light and shadow as well as contrast between tiny characters and mighty environment to create an immersive and Zen experience. Minimalism is applied on both graphic design and storytelling.

To produce that effect, we tried multiple graphic styles to illustrate our game. After several attempts, we found Low Poly a suitable style for our game. First, it's an excellent way to illustrate lights and shadows, hence an elegant atmosphere. Second, most low poly arts are minimalistic, which is exactly what we are looking for. After trying to build some parts using Photoshop, we realized that it's more time-consuming and complicated to create low poly game in a 2D layout. Therefore, we decided to create 3D assets in Blender but keep a fixed top-down view.

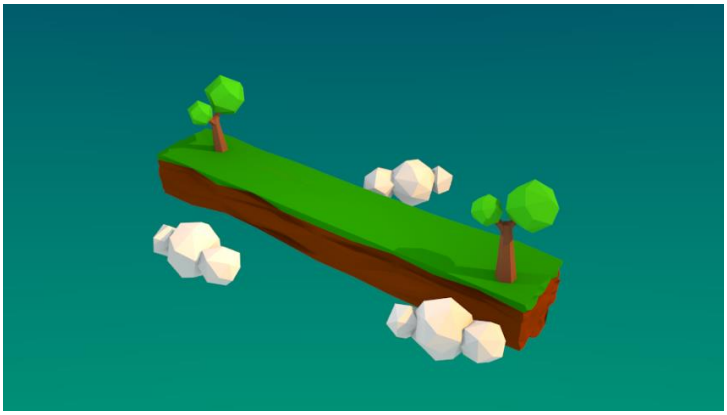

Figure 3.11: Low poly concept design

After seeing the possible of low poly and choosing Blender as our art development environment, we began to build our levels.

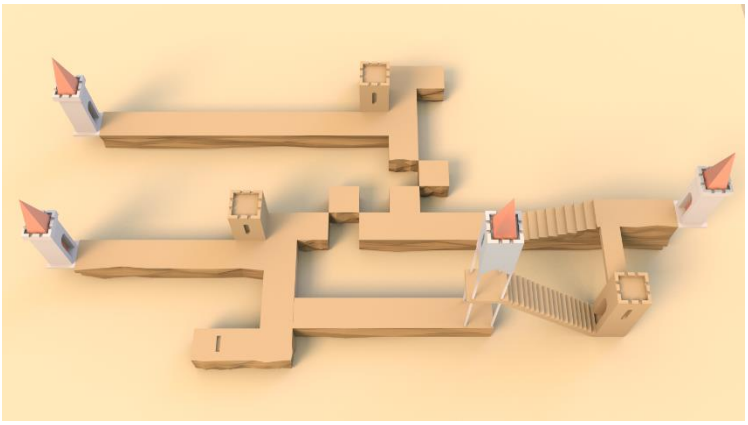

Figure 3.12: level_1_1, first version

Figure 17 illustrates the first version of the level_1_1. We sent this design for user testing and the feedback was that it was too simple and looked too much like Monument Valley. Players wanted to see more details, so we added more details to depict a collapsing game world.

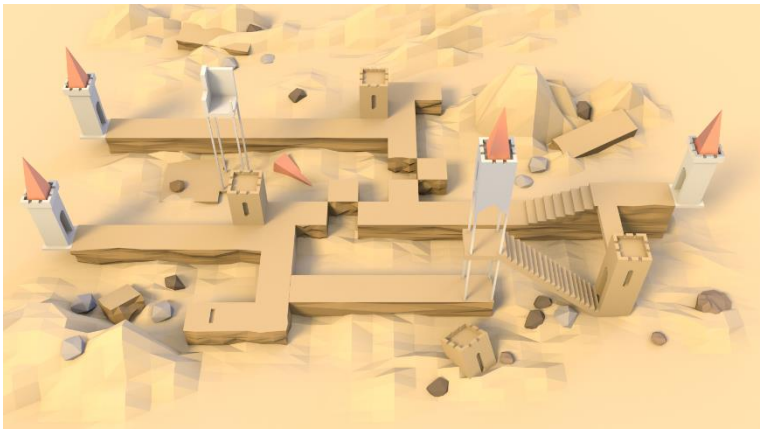

Figure 3.13: level_1_1, second version

After having the second version (Figure 18), we sent it to players again for feedback. Players considered this version visually better than the previous one, however, they felt that the design doesn't match a castle setting, but more of a desert setting. We did some research on how games

form a cognition of indoor space. Based on that, we added pillars, walls and a dark background. Then we have the final version of our game art (Figure 19).

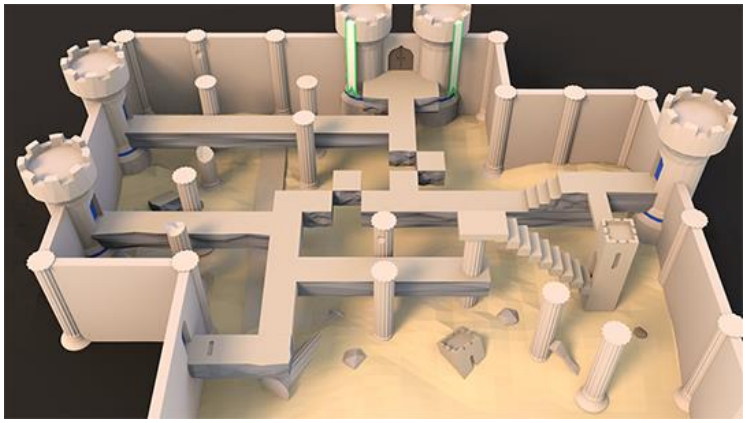

Figure 3.14: level_1_1, final version

The first stage of this game is in a castle. We extracted some elements from the real-life castles and castles in video games. We designed 5 levels in the castle. The *Crystal Hallway, Desert Chamber, Windy Garden, Lounge Bridge and Water Palace*. Each level and the sub-levels underneath have the same visual style.

Figure 3.15: Crystal Hallway

The *Crystal Hallway* is designed as a dark and long corridor where player will meet the oracle for the first time. It's the first scene players see in this game. Since this is a game world with some

“magic”, we tried to deliver a mysterious experience on player first glimpse of this game and let them reveal this game world level by level. To do so, we made the first level dark, with floating glowing crystals and pavilion. The lights in this level attract players’ attention to the path and oracle so that player won’t be distracted by other game assets.

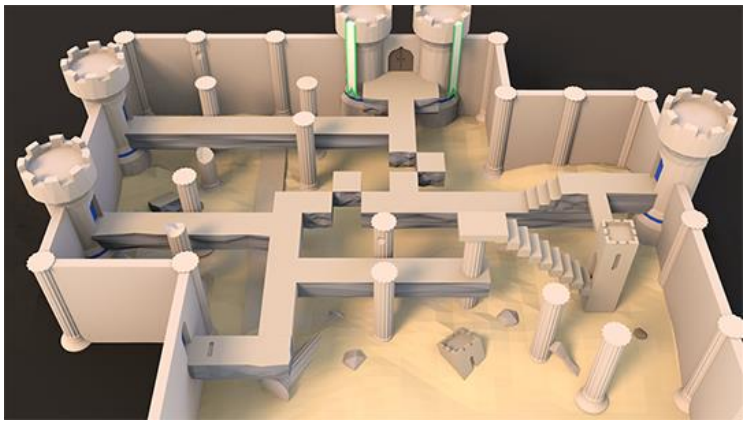

Figure 3.16: Desert Chamber

The *Desert Chamber* is some abandoned lobbies in the castle. We used some shattered path, cracked pillars and sand-covered floor to create the old and broken atmosphere.

Figure 3.17: Windy Garden

The *Windy Garden* is the imperial garden in this castle. In this level, we tried to build something different. The first reason is that we need to ensure players won’t get bored by similar

indoor scenes and we want to construct a multifarious game world. It is an outdoor level with plenty of nature elements like trees, rocks and water. We used high brightness color plan to make this level elegant and delightful.

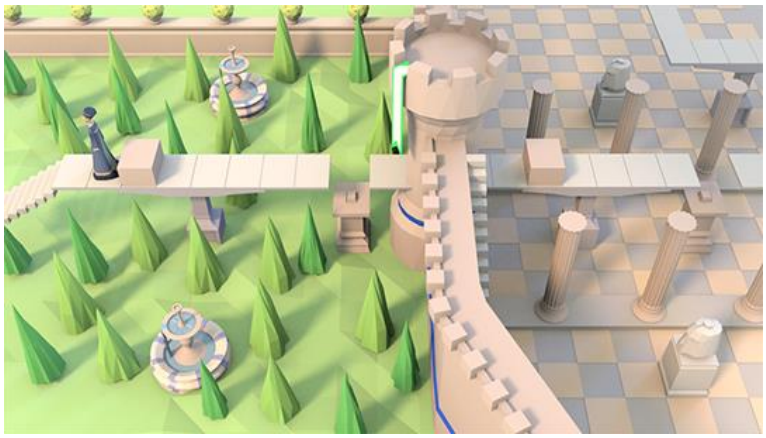

Figure 3.18: Lounge Bridge

The *Lounge Bridge* is a level where May enter the indoor scenes again. Since the puzzles and assets in this level is more complicated than the *Desert Chamber*, we decided not to make the visual too distractive. We did not make these halls broken, instead, we built a neat and clear space to keep the focus on the puzzle solving.

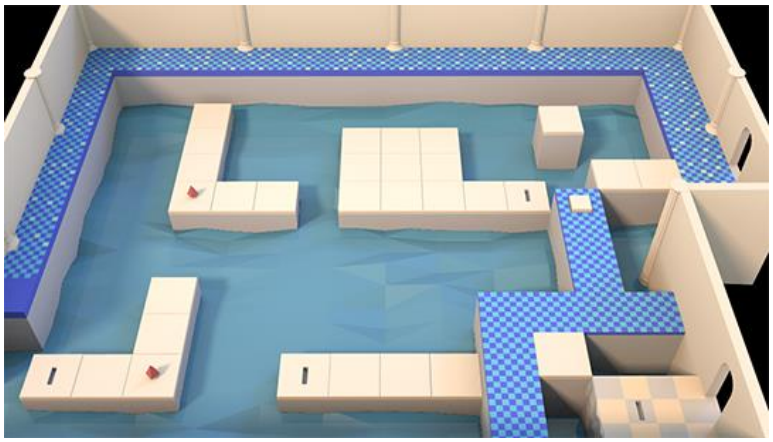

Figure 3.19: Water Palace

The *Water Palace* is referred to the Roman bath houses. We still tried to build visually different levels while the main art style is kept consistent. It is a level in the water. The low poly water is inspired by the Monument Valley.

3.5 Level Design

The levels in our game are divided into groups (main and sublevels). Each group is meant to practice one teaching content at a time, using repetition and logic variation. If we breakdown the levels, we can associate each one with its specific teaching content and its new game mechanics.

Level_0: This is an introductory level where the player has only to move one block one time. This level serves as a tutorial that teaches the coding mechanic.

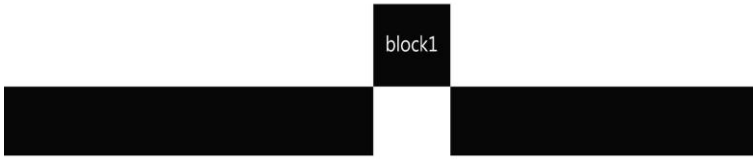

Figure 3.20. Game Design level 0

Level_1: (4 sublevels): In this level, the player will manipulate more than one block, in different directions. Some other gameplays will be introduced:

-Collecting keys from level 1_3 and 1_2 to open a locked door that leads to level 2_1

-Having three coding interfaces in level 1_4, so the player can step on the movable block and move with it.

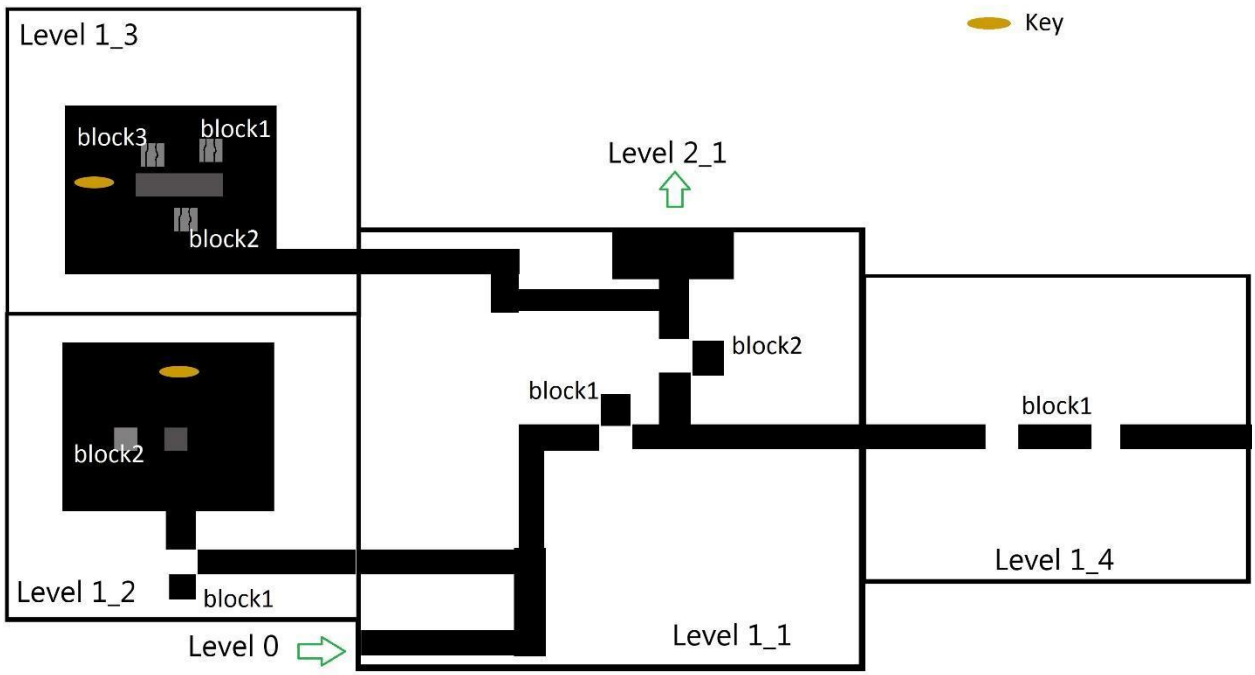

Figure 3.21: Game Design level 1

Level_2 (3 sublevels): In this level, we introduce rotation. We also introduce a new game mechanic "wind" that can block you or can also be used to light fire stone.

-On level 2_1, the player needs to block the wind with a normal block to be able to pass.

-On level 2_2, the player needs to move the wind blocks so they light all 4 fires. When the fires are lit, the platform with the key becomes accessible.

-On level 2_3, the player needs to rotate the block to pass, and then rotate it back so it lights the fire and they get the key. They will also need 2 keys to go to level 3.

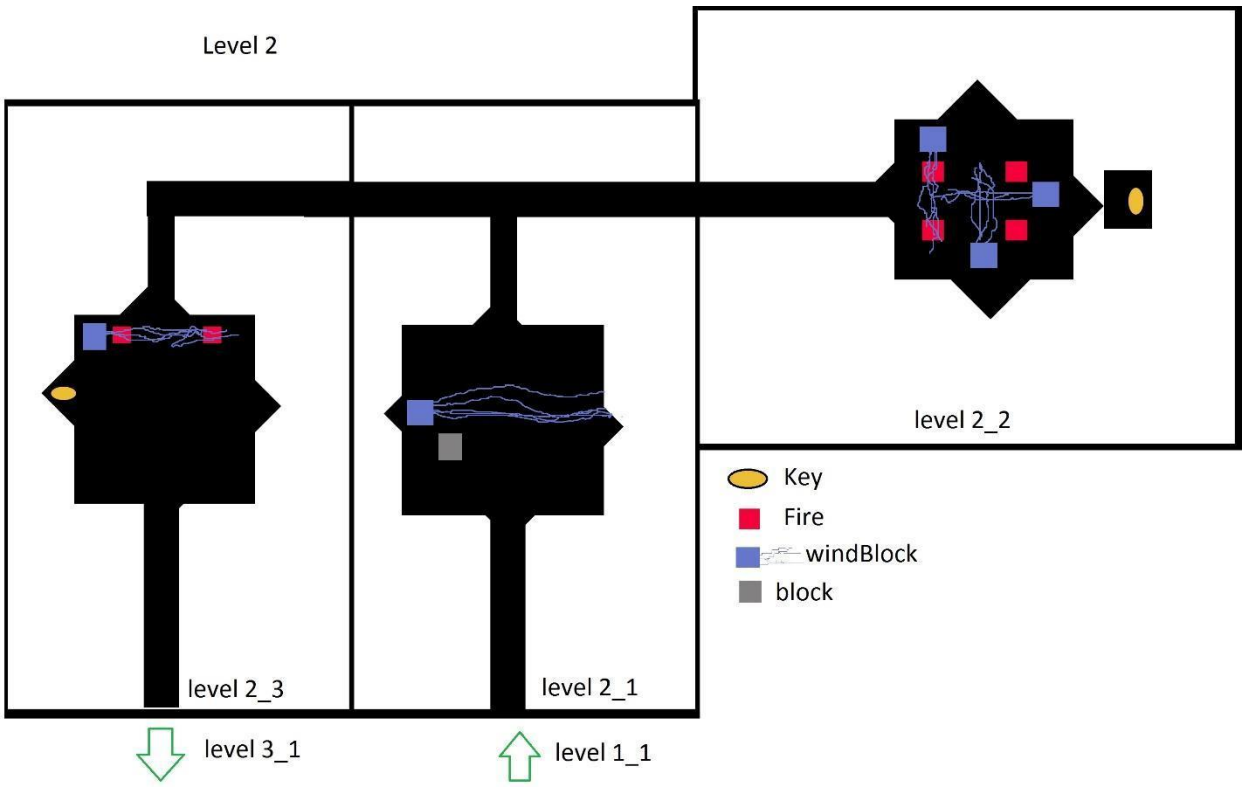

Figure 3.22: Game Design level 2

Level_3 (3 sublevels): In this level, we introduce loops. We also introduce the cat companion which can also be coded to be used to solve puzzles.

-On level 3_1 players need to rotate 3 blocks, the first one will block the cat, the second and third one, will either let them pass and block the coins (the coins are collectibles that will be explained later)

Before leaving the level, the players need to unblock the cat and go talk to him so that he joins them and starts following them.

-On level 3_2 we introduce a simple loop to move 2 blocks 5 times to the right.

-On level 3_3, players need to use loops and code the cat to step on a pressure plate to lower a platform and move a block.

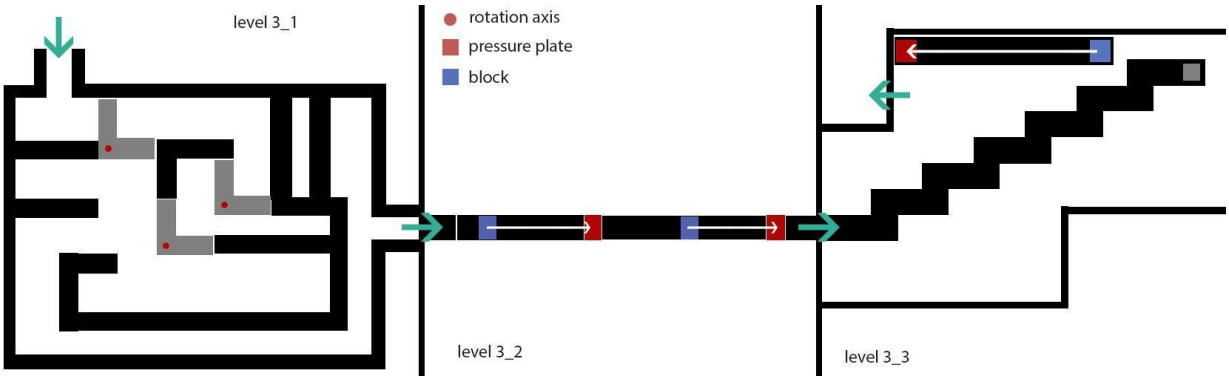

Figure 3.23: Game Design level 3

Level_4 (2 sublevels): In this level, we introduce a new game mechanic revisiting all previous teaching contents. In level 4, players will learn to use different coding interfaces to manipulate different objects.

-On level 4_1 players need to rotate 2 blocks using 2 coding interfaces, they need to rotate the first block and use the coding interface on this block to rotate the second block.

-On level 4_2 players need to rotate 3 blocks with help of their cat, also, they need to think about the sequence. They need to rotate these blocks in order to get both them and the cat through this level.

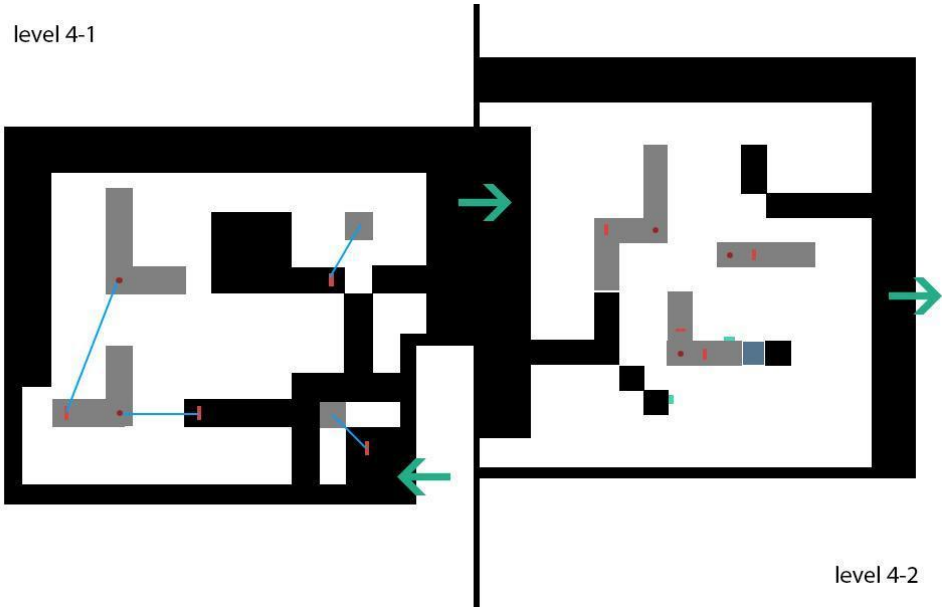

Figure 3.24: Game Design level 4

3.6 Additional Gameplay Elements

While playtesting the game, we realized that there is too much walking for no reason. The game felt empty between the coding parts. That's the reason why we have decided to add more gameplay elements that will encourage exploration and will make walking around the levels more interesting.

3.6.1 Collectible gems

According to our research, collection system helps increase a game's user viscosity. Acquiring is often associated with positive emotions, such as pleasure and excitement, motivating individuals who experience these emotions while acquiring to keep acquiring (Graft, 2009). Therefore, we added a gem collection system in our game.

These gems have no real use for now, but we are planning on adding unlockable items that players can buy with them. The gems also serve as a guide to the player when it's not obvious where they need to go. Finally, the gems serve as a bonus for extra programming efforts. For example, there are two ways of solving level 3. One easy way, gets you through the level and one more complicated will get you more gems.

Figure 3.25: A screenshot of May collecting gems

3.6.2 Secret levels

Research shows shortcuts, mini games, secret levels could be good rewards for players (Ernest Adams, 2014). In order to provide players the experience of surprise and encourage players to explore this game world, we added a secret sub-level in level 1.

Figure 31 illustrates the different levels in which the secret code is involved. On level 1_1, the players can see the secret code but they can't reach it. When they go to level 1_4, there is a pressure plate that will unlock the way to the secret code. Once they get it, they can use it in level 1_3 to type it in and get to a secret room full of gems. This part of the game is not necessary and players can skip it. It is a reward for exploration and perseverance.

Figure 3.26: Some screenshots from how to get and use the secret code

3.7 Teaching Content

As our research suggests, it is better to teach logic rather than syntax for beginners. However, drag and drop approach doesn't really teach programming. This is why we have opted for pseudo coding as a solution to fill the gap between drag and drop and a real programming language. In order to keep a level of abstraction suitable for beginners, we chose to use a custom programming language that is inspired from an Object Oriented language like Java but without the heavy syntax that comes with it.

We want the players to type their own code and there are a few reasons that motivated this design choice. First, we wanted the programming part to “feel” like programming. The players are writing their own code and what they wrote will modify objects. The feeling differs from using a code that is already written for you. Also, one of the biggest challenges when learning programming is how to debug the code. That’s why debugging will be part of our teaching content through simple error messages. Another reason to type the code is to be able to assess the teaching value after playing our game by giving players some written samples of the content introduced in the game and see if they can solve the problems.

In the first stages of the game we will teach basic instructions and sequence logic using methods that are defined within the context. In later stages, we will introduce more syntax and evolve into a fully functional programming language. For that reason, we have first narrowed down the teaching content to the following:

- Basic instructions and Sequence logic
- Loops
- Variables
- If statements
- Comparators (<, >, ==, !=) and Boolean logic
- Operations on Integers
- Operations on Strings

This choice has been made after reviewing several applications that teach programming namely; Code Academy, Hour of Code, Code.org, programming basics.org and many others. The content is usually the same but the order sometimes varies. In some applications, they introduce

variables before commands. We have chosen to introduce basic commands before the variables because they allow us to provide immediate visual feedback on how the code affected the game.

A breakdown of our teaching content, in order of appearance, looks like follows:

Knowledge point	**Description/Examples**
Main Function	-The instructions must be written inside the main function to be executed. -The first times the main function will be written. Later, the players should write it themselves.

Simple instructions to manipulate objects	object.moveRight;
	object.moveLeft();
	block.Rotate("Right");
	door.Open();

Loops	-First, the player is given a number of repetitive instructions to write.
	-Then we introduce the loop to show how it can reduce the commands to type.
	For (0 to 6){
	block.moveRight();
	}

Variables	Integer, float, Boolean, string and GameObject.
	Ex:
	int weight = 50 + block1.getWeight();
	bool open = false;
	String password = part1 + " " + part2;

Conditions	-Some objects will have random behaviors, they will turn either right or left, they will have

	random states… The player needs to cover all the cases to solve the problem.
	Ex: if (block.position == certain position) {
	//do something
	}
	else{
	//do something else
	}

Functions	-The player is given repetitive sets of instructions but only the parameters vary. -They cannot use a loop because the parameters will change. -Then, we introduce how to create a function with parameters and how to call that function in the main function.

Table 3.1: Knowledge points and their appearance in the game

3.8 Programming language design

For the coding part in our game, we are using a custom programming Language. To create our language we need three steps, defining the grammar, the parser and the interpreter.

3.8.1 Defining Grammar

According to our teaching content, we don't need a fully functional programming language but rather one that encompasses the teaching point we need.

a. **Types:**

We use 4 types in our language: integer, float, Boolean, string and an implicit type GameObject.

The types are defined this way:

"[0-9]+\.+[0-9]+" FLOAT

"[0-9]+" INTEGER …

b. **Variable:**

A variable can be written in different ways. For a new variable:

Type identifier; or type identifier = value;

If it's a variable that is already declared then it's

Identifier = value; identifier = operation; (operations will be explained later)

c. **Command:**

Our commands are written this way:

ObjectName.MethodName(); or ObjectName.MethodName(arg1,arg2,arg3…);

Which means: identifier.identifier()

The argument could be either a variable (identifier) or a variable type (string, integer, Boolean…)

d. **For loop:**

Our for loop is defined this way: for (0 to 5){ ……..}

Which means: for (int to int){ ……. }

Inside the loop we can have our commands.

e. **If condition:**

For the condition there are also different ways to write it, the generic one being:

if (condition){

}

else {

}

The condition can be

- A Boolean
- A combination of booeans (not, or, and)
- A comparison between two variables, or a variable and a value.

f. Operations:

The operations affect the variables. So we have different operations for each type of variable.

- For integers we need to have (+,-,/,*) and also parenthesis.

- For strings the operations will be defined as commands: stringVariable.method(args);

- For Booleans we have the operations (not, and, or)

3.8.2 Parser

To create our Parser, we used the software TokeIcer. This program was created by Alan Bryan in 2011 and can be downloadable for free. The software allows users to create their own grammar and generates a class for them that uses that grammar to transform an input string to a list of Tokens. This software helped us a lot in the process of creating our programming language, since it made us gain precious development time.

TokenIcer provides a nice easy to use GUI that serves as an editor for language rules, as well as a test bed for testing the syntax of our rules. In addition, once our parsing rules have been defined, TokenIcer can create a parser class, based on our rules, in either C# or VB.NET. In our case we used C#.

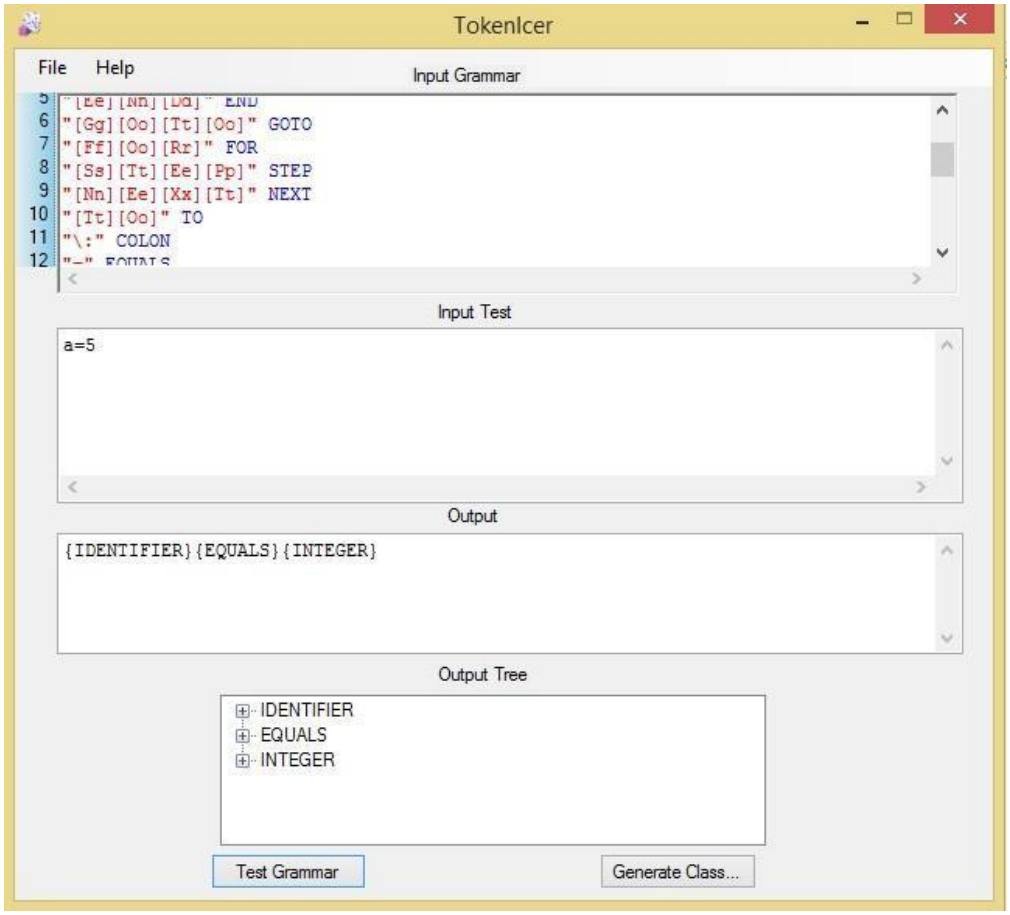

Figure 3.27: TokenIcer interface

We type our input grammar to define each of our tokens. The tokens defined at the top, have a higher priority which means the order in which we define our tokens matters. When we Press Generate Class, we generate a class that defines all the tokens, with some implemented methods:

- **GetToken():** This method to retrieve the next token from the input string. The GetToken()method returns a Token object which contains the TokenName (which is an enum defined earlier like INTEGER, WHITESPACE, FLOAT) and it contains the TokenValue. The TokenValue will be the value retrieved from the parser. For example, an INTEGER might return "8" for example.

- **Peek():** This method will return the next token that GetToken() will return. It allows to look ahead in the token buffer without actually pulling anything off the queue. The return value of Peek() is aPeekToken that contains a Token object and an index. By callingPeek() and passing a PeekToken as an argument to Peek(), Peek() will return the Token that is returned after the last Peek() call. In this way, it's possible to peek ahead several tokens deep.
- **ResetParser():** This method resets the parser. After calling this, the InputStringproperty must be set to a string again and then we can call GetToken() or PeekToken() as you normally would to parse a new string (Bryan 2011).

3.8.3 Interpreter

To implement the logic of our language, we need to find patterns that will lead to something. For that, we created different methods to check the code. First, we need to make sure that the syntax is correct; the types are written correctly, there are no semicolons or curly brackets missing… then we check the logic, for example, when you declare an integer, you can only give it integer values. You can't use a variable that was not declared before….

Once we make sure the syntax and logic is correct, we create objects for them. For example, we will have loop class, condition class, command class…

Figure 33 represents a simplified class diagram of our language parser.

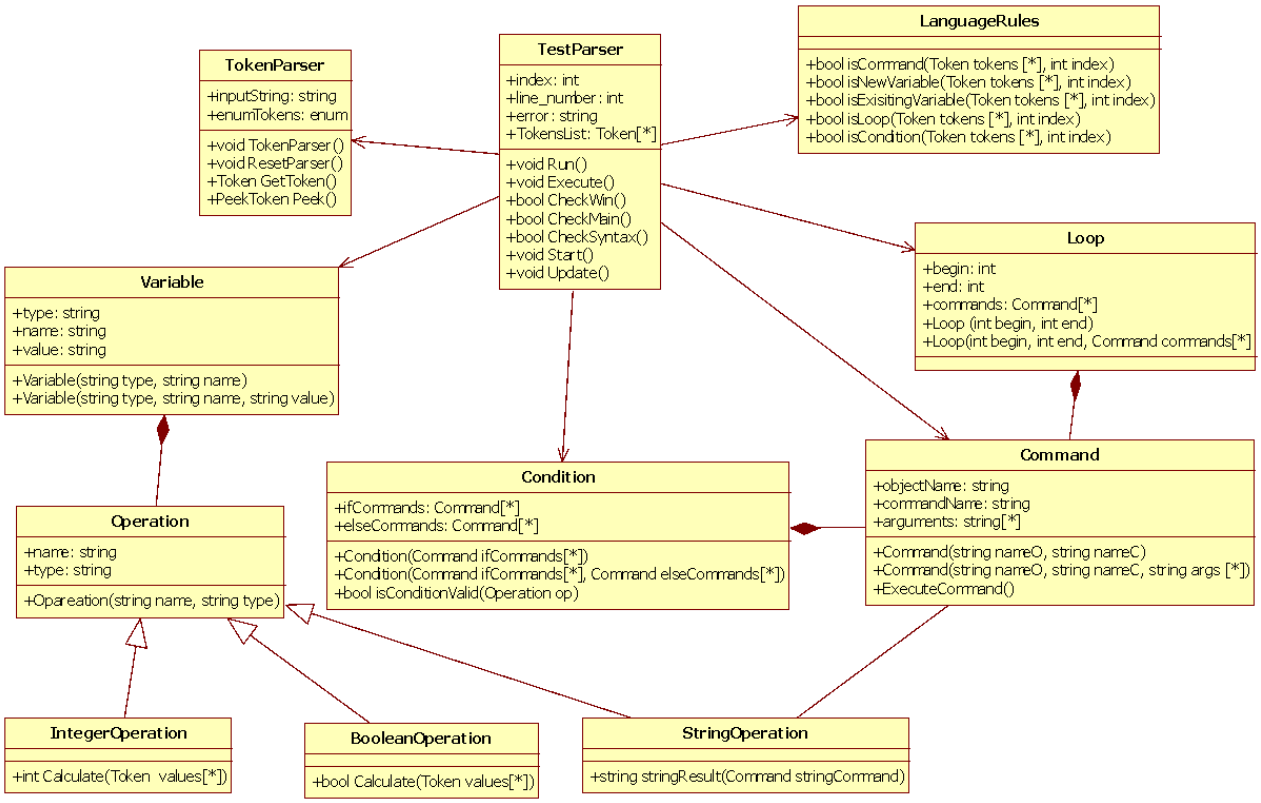

Figure 3.28: Class Diagram

The program runs as follows:

- TestParser will get the Tokens from the inputString of TokenParser.
- It will check use the CheckMain() function to see if the main function is written correctly.
- If the main function is correct, it will continue to check the syntax using CheckSyntax() method else it will stop and return an error.
- CheckSyntax() will use the methods from LanguageRules class to check the input (isCommand(), isLoop()…) and create the appropriate class each time (loop, command…)
- If CheckSyntax() is correct, TestParser will call the Execute() method which will execute the commands from the loop, condition or other commands in the order they were written in by the player.

- By executing the methods, the GameObjects in the Unity Scene that were called in the code will be modified accordingly.

3.9 Game UI

The 3 most important elements of UI design are affordance, feedback and revocability (Alen Cooper, 2007). Affordances are hints in games which imply to players the consequences of their operations. They make the gameplay predictable. Feedback helps players to make judgements and decisions and it gives players a sense of control. Revocability enables players to redo their operation after making mistakes.

Our game will be a top-down 3D game. According to the information architecture of this game, we need to arrange the game UI and coding UI on the screen simultaneously to enable players to intuitively manipulate the game objects using code and get immediate feedback. Although CodeCombat's gameplay is more masculine, the main UI frame of CodeCombat is a good reference.

Figure 3.29: CodeCombat UI design

In our game UI design, players will have a 45 degree top-down view in every level's beginning. Once players enter the coding mode, the perspective will automatically switch to the 90 degree top-down perspective so that players are able to see the whole map while manipulating the objects. (Of course players are able to switch back to the normal perspective) In the coding mode, there is a coding tablet and a console tablet on the right of the screen. Players can write their codes on the coding tablet and read information (like errors) on the console tablet. For example, if a player makes a syntax error, they will be able to recognize the error and locate the line where it occurred. The reason we use this UI layout is that this coding-console structure is applied in most coding software. Our intention is that when our target users get will switch to real coding environments in the future, they will feel familiar with the layout.

On the bottom of the screen is the information bar where players are able to find the affordances. When players hover their cursor on the manipulatable objects, corresponding information and hints will be displayed in this bar allowing players to easily figure out what they can do with these objects or what objects in this level they can manipulate. We want players to focus on solving puzzles and writing codes instead of spending too much time on figuring out what to do.

Figure 3.30: May's Journey UI design

Chapter 4: Evaluation

To make sure our game satisfies the goals we set for it, we needed to submit it to two major playtest sessions. The first session is actually continuous throughout the whole development process and allowed us to iterate on our design at every level. The second session is testing the final prototype with our target audience and get results and efficacy, enjoyment and transference.

4.1 Iterative Design

To design the different levels of our game, we first started with paper prototypes that we tested with different people to adjust the difficulty level. Our testers were mainly WPI students from different fields.

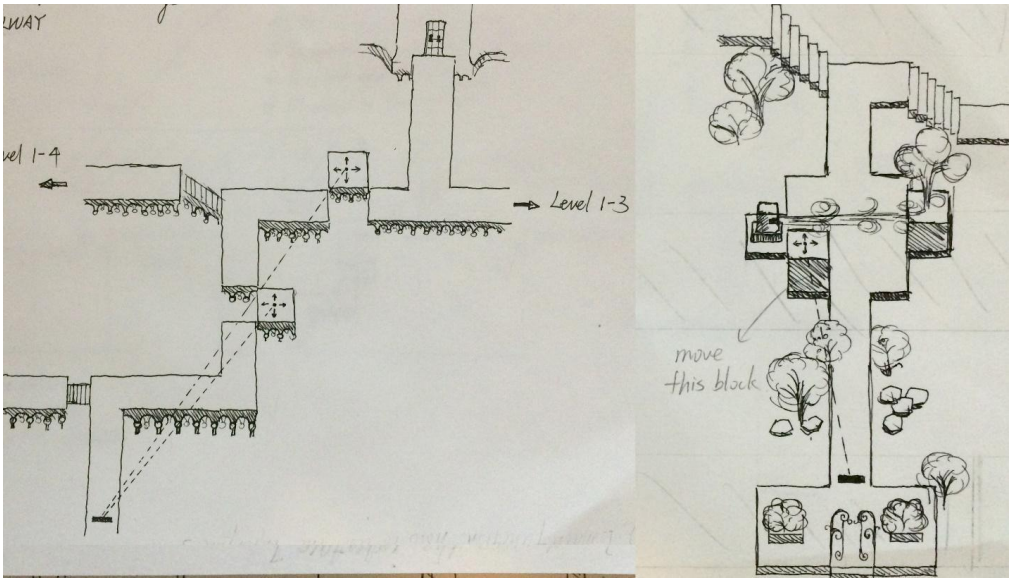

Figure 4.1: Some paper prototypes of our level design

Once the level of difficulty and the transitions were set, we built a 2D digital prototype of levels 0 and 1 that we also sent out for testing to a larger group of testers including WPI students and other participants interested in being play testers. We sent a downloadable PC build along with a google doc asking if they have finished the level, how long did it take them to finish, how easy/hard was it to figure out the mechanics, what do they think of the difficulty curve, also if they liked/disliked any sub-level in particular. Finally, we had an open-ended question about general feelings and suggestions on how to improve the general experience. We were also looking into possible coding bugs and some incoherencies in the gameplay. Figure 4.2 shows a screenshot of the coding interface in level 1_1.

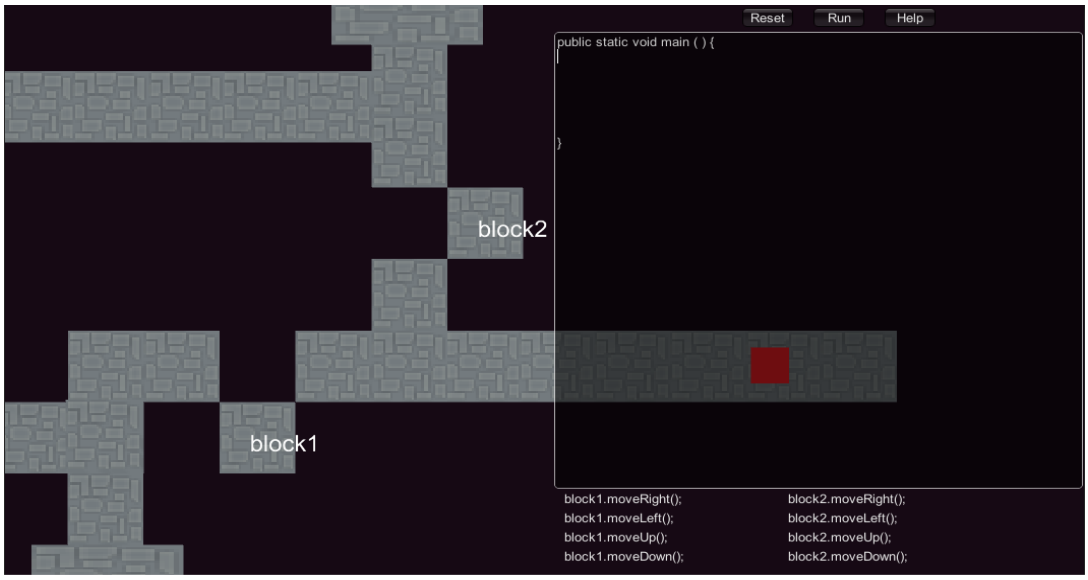

Figure 4.2: 2D prototype of level one with the coding interface

The main objective of this approach is to provide a continuous identification of design issues early in the process. So for our final testing, we focus on educational content with our target audience.

4.2 Targeted Testing

In this playtesting session, we were more interested in assessing our research questions. We wanted to verify three major points.

- Was the game engaging and fun to play?
- Did the game increases girls' interest for programming?
- Did the game teach some programming knowledge?

To answer these questions we used the following methodology for our testing

4.2.1 Methodology

Serious games require unique usability challenges that are not necessary for traditional games. The reason is the additional objective of knowledge discovery through exploratory learning. (Moreno-Ger et al. 2012). We tried to follow the methodology presented in the research article "Usability Testing for Serious Games: Making Informed Design Decisions with User Data" for our testing (Moreno-Ger et al. 2012). Since we didn't have access to recording devices to collect information about all of our users, we changed that part with in-game observation.

For the educational part we know that we can't just merely present the players with a subject and assume they have learned. Assessment is crucial in order to determine that the students have understood the material and can be expected to recall and use the material appropriately (Michael & Chen 2005). This is why our post-game test include small programming exercises similar to what was presented in the game.

Our testing Protocol included three major parts; a pre-game form, in game observations and a post-game form.

The in-game observations were as highly important as the questionnaire. In fact, in a study that compares usability testing methods for video games shows that think-aloud protocol is both more effective and cost effective, especially when a small amount of participants is available (Theodorou 2010).

Our tests were conducted at Youth in Action, a place where youth share their stories, practice leadership, and create change in their communities. The testing session was held in their locals in Rhode Island.

a. Pre-game form

This form (Appendix A) covered three major areas. The first one holds demographic questions (age and gender). These questions are important for us to understand if girls actually like the game more than the boys. We are also interested in checking if the game is more suitable for a certain age range.

The second part of the form included game related questions like game genre preferences, gameplay preferences and frequency of playing games. These questions will help us assess if the game is more successful for gamers who like puzzle and storytelling games more than others.

Finally, the last part tackled programming background and general feelings about programming; if they think it's hard, important or boring. The later questions will also be asked after the game to see if playing the game somehow influenced their view about programming.

b. In game observations

During the game, we were observing their reactions, level of attention, excitement and engagement. We were also writing any noticeable behavior or statement. When the players are stuck or ask for help we offer our help and note the level and the part that causing the confusion.

c. Post-game form

This form (Appendix B) also covers three major areas. First, we ask about the overall appreciation of the game, the environment, the characters and the story. We also ask if they would play the game again and why.

The second part includes questions about programming, if they felt like they learned something. If they liked the programming part in the game. Also, we ask the same questions asked in the pre-game form about their feelings towards programming. Then, we ask about their intentions to pursue learning programming.

The final part is a transference test. Since in our prototype we didn't implement all of our teaching content, we only had three small programming questions in this test.

4.2.2 Results and analysis

a. Participants background

Surprisingly, while our data shows that about half of the gamers were females, 7 out of 8 of our female participants answered that they played games very rarely. The boys answered that they played on weekends.

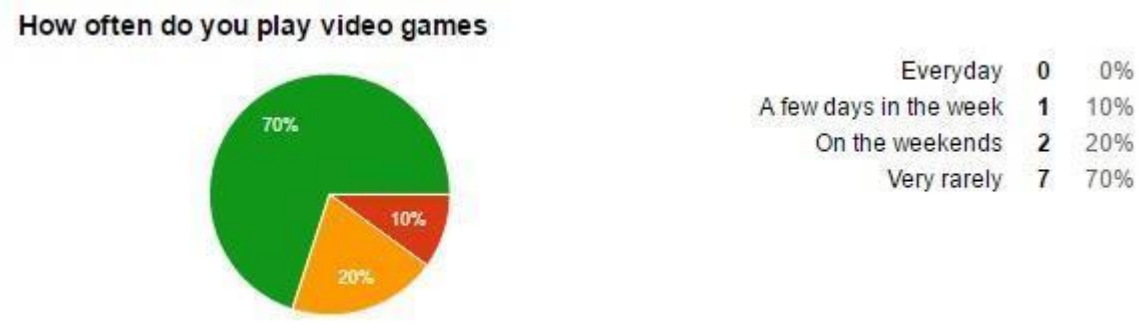

Figure 4.3: Answers to the question "How often do you play video games?"

From Figure 4.3 we can infer that we have a relatively non-gamer crowd. Also, 8 of our participants answered that they had never programmed before. One girl had programmed before using Hour of code and one boy had tried with online tutorials.

About the game preferences, we wanted specifically to know if the players liked puzzle games and games with stories to know if they will be more inclined to like our game. We asked to rate how do they like puzzle games for scale from 1 to 6 (6 being they like them a lot). The average was 4.3 with a standard deviation of 1.07. Some of the reasons why they liked puzzle games were "they make me think and sometimes gets me frustrated but I keep playing", "What I like about them is that they're challenging but I hate when they're too difficult to solve", "I like the ability to use your mind". The reasons why they don't like them is if they are too hard and too challenging.

For stories, 8 of them liked games that include a story. The kind of stories they liked were too broad to be conclusive; it ranged from fantasy to drama, action, realistic stories, suspense, murder and more.

b. Programming results

Now, if we take a look at what our players thought about programming Figure 4.4 before (to the left) and after playing the game. All of them thought that programming was important. And 8 of them already thought that programming was not boring. Moreover to the open question "what do you think about programmers?" most of the participants valued programmers as very smart people, geniuses, very patient and hard working.

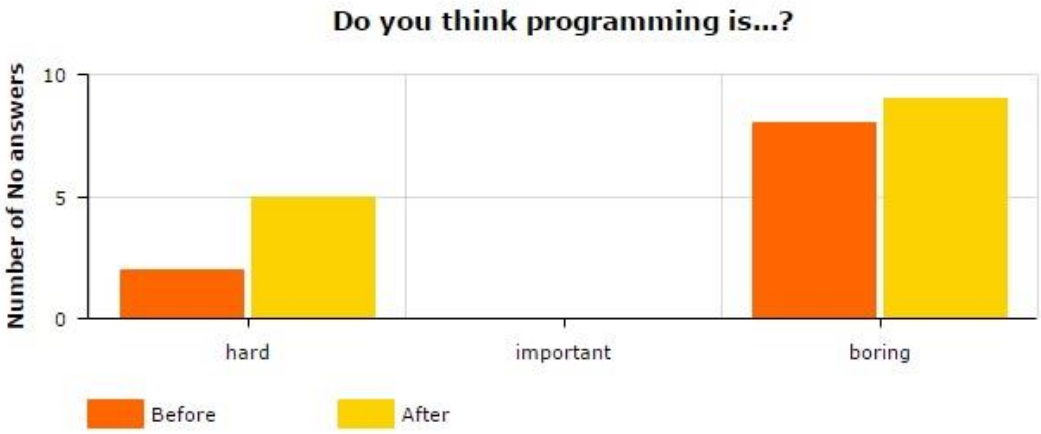

Figure 4.4: Comparing feelings about programming before and after playing the game

The interesting part is that after the game the number of participants thinking that programming wasn't hard increased by 3. Which suggests the game made them feel that programming is more accessible than they thought. If we consider the girls, before the game, only one of them thought programming wasn't hard and after the game 4 of them thought it wasn't hard. This finding is very important to us because research has shown that women avoid computer science fields because they think it's too hard for them. Allowing them to see that it is in fact accessible can probably make a difference in their decision to pursue computer science.

8 of the participants also answered that they would want to learn more about programming. To the question on a scale from 1 to 6 (6 being I liked it a lot) "How did you like the programming part in the game?" the average answer was 4.5 with a standard deviation of 0.74. One of the reasons mentioned was "this is much better than someone telling you do this and do that". We consider this score very positive since programming is the major part of the game and we wanted it to be enriching the game experience rather than a task you "have" to do in order to progress.

Regarding programming as part of their career, before the game 3 of the participants didn't know if they would consider it because most of them have heard little to nothing about programming. After the game, we didn't get any "I don't know" answer, however they split equally between yes, no and maybe. But overall, the percentage of participants who would consider a job that includes programming increased by 2.

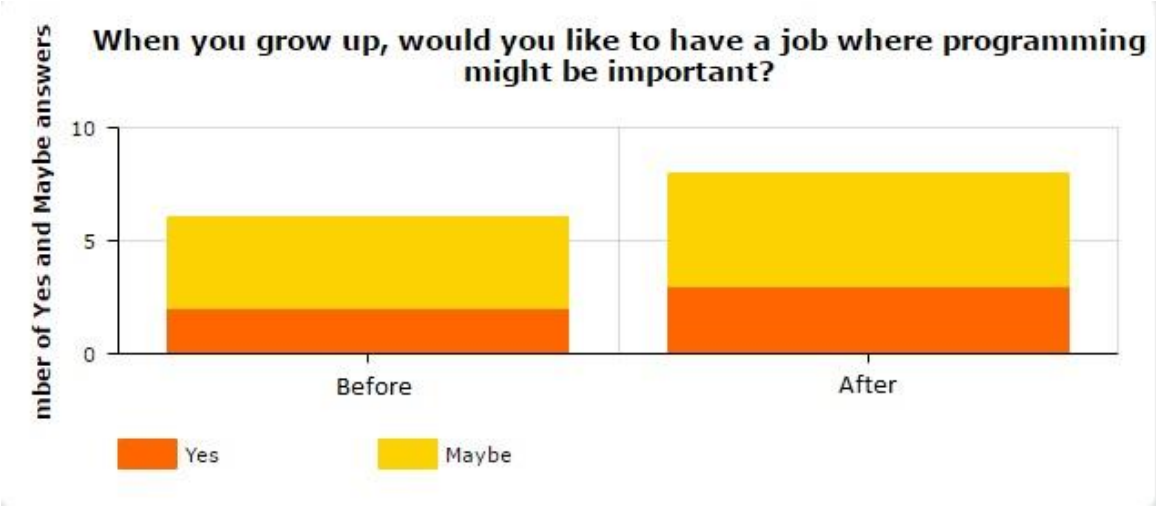

Figure 4.5: Comparing feelings about programming as a career before and after playing the game

All the players answered that they felt that they learned something from the game and 8 of them wanted to learn more about programming. 6 out of 8 girls wanted to learn more about programming. We consider this result very positive since one of our main goals was to make programming more interesting and appealing to girls.

c. Gameplay results

We were also interested to know what they thought about the game in general because we think that the game itself can influence their feelings about programming. If they consider the game bad, they might project that feeling to programming. So we asked them to rate on a scale from 1 to 6 how they liked the game, the graphics, the characters and the story. The results were

an average of 4.8 (standard deviation 0.77) for the game, 4.2 (standard deviation 0.77) for the environment (the environment scores might be lowered because the image quality wasn't optimal on the computers used), 4.5 for the characters (standard deviation 0.74), 3.5 for the story (standard deviation 1.05)

The game rated very high which may have influenced feelings about programming. Moreover, when asked "would you like to play more of this game?" all of our participants answered "yes". The reasons varied; 5 of them thought it was fun to play, 2 mentioned learning programming as a reason to continue and 4 were motivated by the story and what will happen to May. On one hand, the story had an average score, but on the other hand, players were motivated to know more about what will happen. We think that the storyline is good but the dialogs and the way it is presented could be changed to be more captivating.

The game engendered many feelings amongst the players. From the observations, we could note their engagement, frustration and concentration. All the participants seemed very engaged. They were very loud especially when they figure out something. We heard comments like "this game is hard", "this is confusing", "I love this game", "this game is too much", "oh my god the cat talks", "oh my god I have a cat", "the cat is following me, how cute, are you jealous?" Some of them were also comparing how many crystals they collected.

The answers on the survey also confirm those observations.

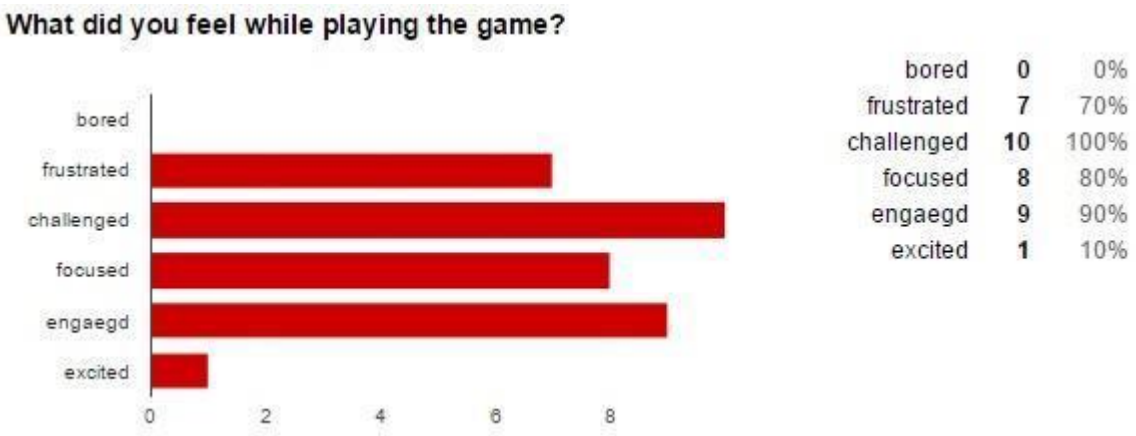

Figure 4.6: Feelings conveyed by the game

All of the players were challenged, 9 felt challenged, 8 felt engaged and 7 felt frustrated.

To make sure that the game is balanced we also asked about the level of difficulty in the game. We asked about the first level which is meant to be easy and serve as a tutorial. We asked them to rank the difficulty from 1 to 6 (6 being very hard), the average was 2.3 with standard deviation of 0.75 which means that the level was mostly considered of medium difficulty. This was already observed during playtesting. The plan was not explain how to play the game as it includes a tutorial level itself, however most of the participants had trouble understanding what they were supposed to do for the coding part. We were walking around and made sure they understood what they were supposed to do. We have also noticed that they helped each other a lot when one of them got stuck.

About the general level of difficulty in the game, we asked if the players would have preferred the game to be easier, harder or if the level of difficulty was good for them. 9 answered that they the level of difficulty was good. 1 wished it was easier. Even though the players approved the level of difficulty of the game, most of them needed assistance to figure out the first level. Once they got it, their progress was smoother. So we believe that the tutorial level needs more guidance. The whole game in general needs more UI helpers.

We also wanted to check if the gameplay additions added to the experience. Only 2 players used the secret code. Most of them didn't know how to use it so we might rethink the hint system to use it. However, 8 of the players tried to collect the gems. Since we didn't add a usage for them yet, we asked the players what they would expect to do with the gems collected. Most of them thought they could buy stuff for the game, unlock items, secret codes, new levels, new companions and power-ups.

Chapter 5: Conclusion and Future Work

The goal of this thesis was to build a game that will spark girls' interest in programming and Computer Science in general. Women being not very present in the Computer Science field, many initiatives are trying to reach out to young girls and make programming accessible to them. In that same spirit, the game teaches basics of programming through puzzle solving. The gameplay and the teaching content are closely linked to create a full immersive experience.

We first started by thinking about an interesting story that would draw girl's attention. That's how we came up with May and her lost friend Juno. We wanted the programming to make sense in the game which drove our environment choice towards a broken game world that can be fixed with coding. While designing the game, we made sure that we had always our research on girls' preferences in mind.

After an iterative game design and development in which the levels, the characters, the story and the environment have evolved, we had finally built a robust first prototype to test with our Target Audience.

We tested our game with 10 teenagers aged from 14 to 17 years old. We were pleased to see how engaged with the game they were. Overall, the testing results were mostly as expected. The players liked the game (rated 4.8 out of 6) and all of them wanted to play more of it. They all felt that they learned something and 8 of them expressed the will to learn more about programming. The storyline was generally appreciated, however the dialogs might need some tweaking to sound more exciting and appealing.

One of the major problems of the game is the tutorial level. We could overcome this problem during our playtesting because we were there to help, but we want a player to be able to play our game without any assistance so that we can reach a larger group of girls who won't be introduced to programming otherwise.

Unfortunately, the sample of players is too small to generalize our results so we plan to take the feedback into account, iterate and test it again with a larger study group and get conclusive results.

References

Alexander. 2014. “Sexism, lies and video games: The Culture War Nobody is winning”. Time http://time.com/3274247/video-game-culture-war/

Alan Bryan, icemanind. 2011. “easily create your own parser”. Code Project http://www.codeproject.com/Articles/220042/Easily-Create-Your-Own-Parser

Anna Lewis. 2011. “When computer programming was ‘women’s work’”. The Washington Post http://www.washingtonpost.com/opinions/when-computer-programming-was-womens work/2011/08/24/gIQAdixGgJ_story.html

A. Robins, J. Rountree, and N. Rountreem. 2003. “Learning and teaching programming: A review and discussion. Computer Science Education”. University of Otago http://home.cc.gatech.edu/csed/uploads/2/robins03.pdf

A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. 2007. “A survey of literature on the teaching of introductory programming”. In ITiCSE-WGR ’07: Working group reports on ITiCSE on Innovation and technology in computer science education. http://delivery.acm.org/10.1145/1350000/1345441/p204-pears.pdf?ip=130.215.220.2&id=1345441&acc=ACTIVE%20SERVICE&key=7777116298C9657D%2E71E5F5E88B9A3E17%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=606997071&CFTOKEN=73515802&__acm__=1461829308_0601eb3e65aa6024b98e2f6542b1cd31

Arto Vihavainen, Matti Paksula and Matti Luukkainen. 2011. “Extreme Apprenticeship Method in Teaching Programming for Beginners”. University of Helsinki

http://delivery.acm.org/10.1145/1960000/1953196/p93-vihavainen.pdf?ip=130.215.220.2&id=1953196&acc=ACTIVE%20SERVICE&key=7777116298C9657D%2E71E5F5E88B9A3E17%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=606997071&CFTOKEN=73515802&__acm__=1461829373_b7e822bbbdb65a48797cdde0eda7b2f8

Carrie Heeter, Kaitlan Chunhui Chu, Rhonda Egidio, Punya Mishra, Leigh Graves-Wolf. 2004. "Do Girls Prefer Girl-Designed Games?". Michigan State University

Charles Huff & Joel Cooper. 1987. "Sex Bias in Educational Software:The Effect of Designers' Stereotypes on the Software They Design". Princeton University https://www.stolaf.edu/people/huff/info/Papers/SexBias87.pdf

Carrie Cousins. 2012 "Design Stereotypes: Masculine and Feminine Design Techniques". Designmodo nhttp://designmodo.com/masculine-feminine-designs/

Caitlin Kelleher November. 2006. "Motivating Programming: using storytelling to make computer programming attractive to middle school girls". Carnegie Mellon University http://www.csta.acm.org/Research/sub/Projects/ResearchFiles/kelleherThesis_CSD.pdf Alice Website http://www.alice.org/index.php

Codemancer website http://codemancergame.com/

David Michael, Sande Chen. 2005. "Proof of Learning: Assessment in Serious Games". Gamasutra http://www.gamasutra.com/view/feature/130843/proof_of_learning_assessment_in_.php

Deloitte. 2015. "Of the time you spend playing games (all types of games), what percentage of time do you play games on the following devices?". Statista

http://www.statista.com.ezproxy.wpi.edu/statistics/318196/time-spent-gaming- device-gender-usa/

Deb Miller Landau. 2015. “8 Tips for Teaching Kids to Code”. IQ Managing Editor http://iq.intel.com/8-tips-for-teaching-kids-to-code/

Denise Gürer & Tracy Camp. 2001. “Investigating the Incredible Shrinking Pipeline for Women in Computer Science”. National Science Foundation http://womendev.acm.org/archives/documents/finalreport.pdf

Derrek Khanna. 2013. “We Need More Women in Tech: The Data Prove It”. The Atlantic http://www.theatlantic.com/technology/archive/2013/10/we-need-more-women-in-tech-the-data-prove-it/280964/

Efstratios Theodorou. 2010. “Let the Gamers Do the Talking: A Comparative Study of Two Two Usability Testing Methods for Video Games”. University College London https://www.ucl.ac.uk/uclic/studying/taught-courses/distinction-projects/2009_theses/TheodorouE.pdf

Emily Peck. 2015. “The Stats On Women In Tech Are Actually Getting Worse”. The Huffington Post http://www.huffingtonpost.com/2015/03/27/women-in-tech_n_6955940.html

Entertainment Software Association. 2015. “Distribution of computer and video gamers in the United States from 2006 to 2015, by gender” Statista http://www.statista.com.ezproxy.wpi.edu/statistics/232383/gender-split-of-us-computer-and-video-gamers/

Entertainment software association ESA. 2014. “Essential Facts About the Computer and Video Game Industry”https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=13&ca

d=rja&ua
ct=8&ved=0CGMQFjAMahUKEwijtILNoYTIAhVL2B4KHcqdCAc&url=http%3A%2F%2Fwww.theesa.com%2Fwp-content%2Fuploads%2F2014%2F10%2FESA_EF_2014.pdf&usg=AFQjCNHjQU2vBXksq_sBNNIM8i7X6fRGg&sig2=VAzDptJbPyr3pmcU8RPF8A&bvm=bv.103073922,d.dmo

Erik Frekjmr, Morten Hertzum, Kasper Hornbmk, 2000, "Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?". University of Copenhagen. Riso National Laboratory http://www.diku.dk/~kash/papers/CHI2000_froekjaer.pdf

Ernest Adams. 2014. "Fundamentals of Game Design". Pearson Education https://books.google.com/books?id=L6pKAgAAQBAJ&pg=PA442&lpg=PA442&dq=why+build+secret+levels+in+games&source=bl&ots=1BMiRFSYZk&sig=OJ9y25dAkiRmzD6nkdk3R7Ug0mM&hl=en&sa=X&ved=0ahUKEwiBt7ya2a_MAhUD2B4KHV3MsQ6AEIUzAI#v=onepage&q=why%20build%20secret%20levels%20in%20games&f=false

Jessica Conditt 2014 "What happened to all of the women coders in 1984". engadget http://www.engadget.com/2014/10/20/what-happened-to-all-of-the-women-coders-in-1984/

J. C. Spohrer and E. Soloway. 1986. "Novice mistakes: are the folk wisdoms correct?". ACM Digital Library http://delivery.acm.org/10.1145/10000/6145/p624-spohrer.pdf?ip=130.215.220.2&id=6145&acc=ACTIVE%20SERVICE&key=

%2E71E5F5E88B9A3E17%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=606997071&CFTOKEN=73515802&__acm__=1461829542_54c80c002c365579b191659a803ed32c

Jona Dinges, 2015, illustration https://dribbble.com/jonadinges

J. O'Rourke. 1993. "Mentor Project Targets Female Undergrads". Computing Research News

A. Pearl, M. E. Pollack, E. Riskin, B. Thomas, E. Wolf and A. Wu, 1990 "Becoming a Computer Scientist", Communications of the ACM

Justine Cassell and Henry Jenkins. 2000. "From Barbie to Mortal Kombat: Gender and Computer Games". The MIT Press

Karen A. Frenkel. 1990. "Women and Computing", Magasine Communications of the ACM http://delivery.acm.org/10.1145/100000/92756/p34-frenkel.pdf?ip=130.215.220.2&id=92756&acc=ACTIVE%20SERVICE&key=7777116298C9657D%2E71E5F5E88B9A3E17%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=606997071&CFTOKEN=73515802&__acm__=1461827695_4457a626304bfb05dd6bf39cce8653aa

Katrina Schwartz. 2013. "How to Grab and Keep Girls' Interest in Computer Coding". KQED News http://ww2.kqed.org/mindshift/2013/09/20/how-to-grab-and-keep-girls-interest-in-computer-coding/

Kris Graft. 2009. "Analysis: The Psychology Behind Item Collecting And Achievement Hoarding" Gamasutra http://www.gamasutra.com/view/news/114668/Analysis_The_Psychology_Behind_Item_Collecting_And_Achievement_Hoarding.php

L. Winslow. 1996. “Programming psychology - a psychological overview”. University of Dayton. ACM Digital Library http://delivery.acm.org/10.1145/240000/234872/p17-winslow.pdf?ip=130.215.220.2&id=234872&acc=ACTIVE%20SERVICE&key=7777116298C9657D%2E71E5F5E88B9A3E17%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=606997071&CFTOKEN=73515802&__acm__=1461829450_af6c8ad5ba065e6caf204d68ac4a4c99

Meg Jayanth. 2014. “52% of gamers are women - but the industry doesn’t know it”. The Guardian http://www.theguardian.com/commentisfree/2014/sep/18/52-percent-people-playing-games-women-industry-doesnt-know

M. H. Phan, J. R. Jardina, & W. S. Hoyle. 2012. “Video Games: Males Prefer Violence while Females Prefer Social”. Software Usability Research Laboratory. Wichita state University http://usabilitynews.org/video-games-males-prefer-violence-while-females--social/

Mike Ritchie. 2015. “Can Scratch & Hour of Code Help Your Kid Learn Programming?” , ThinkFun http://info.thinkfun.com/stem-education/can-scratch-hour-of-code-help-your-kid-learn-programming-thinking-skills

Nick Yee. “Unmasking the Avatar: The Demographics of MMO Player Motivations, In-Game Preferences, and Attrition”. Gamasutra http://www.gamasutra.com/view/feature/130552/unmasking_the_avatar_the_.php?print= 1

Owen S. Leach. 2013. “PARALLEL HEARTS MATHEMATICS GAME: USING EDUCATIONAL GAMES TO ADDRESS THE STEM FIELD GENDER GAP”

https://www.wpi.edu/Pubs/ETD/Available/etd-042513111837/unrestricted/Leach_ParallelHeartsMathematicsGame.pdf

Pablo Moreno-Ger, Javier Torrente, Yichuan Grace Hsieh, and William T. Lester. 2012. "Usability Testing for Serious Games: Making Informed Design Decisions with User Data". Advances in Human-Computer Interaction http://www.hindawi.com/journals/ahci/2012/369637/

Patrick Thibodeau. 2012. "IT jobs will grow 22% through 2020, says U.S." Computer World. http://www.computerworld.com/article/2502348/it-management/it-jobs-will-grow-22--through-2020--says-u-s-.html

Robin Potain. 2010. "Forces in Play: The Business and Culture of Videogame Production" NHTV University of Applied Science, Breda, The Netherlands and University of Adelaide, Adelaide, South Australia

Steve Henn. 2014. "When women stopped coding". Planet Money http://www.npr.org/blogs/money/2014/10/21/357629765/when-women-stopped-coding

Usability Testing Methods for Video Games". University College London https://www.ucl.ac.uk/uclic/studying/taught-courses/distinction-projects/2009_theses/TheodorouE.pdf

Williams, D., Martins, N. Consalvo, M. and Ivory, J.D. 2009. "The virtual census: representation of gender, race and age in video games'. New Media Society