

---

# User Interfaces:

An investigation into the effects  
on engagement, caused by  
control scheme simplifications  
in games



---

By Paul Martin W. Christensen



**Semester: MED 10**

**Title: User interfaces: An investigation into the effects on engagement caused by control scheme simplifications in games.**

Aalborg University Copenhagen  
Lautrupvang 15, 2750 Ballerup,  
Denmark

Semestercoordinator: Stefania Serafin  
Secretary: Lisbeth Nykjær  
Phone: 9940 2471  
lny@create.aau.dk  
<https://internal.media.aau.dk/>

**Project period: Spring 2011**

**Semester theme: Master Thesis**

**Supervisor(s):  
Henrik Schønau Fog**

**Project group no.: 20071022**

**Members:  
Paul Martin Winther  
Christensen**

### **Abstract:**

This project is concerning the effects on engagement caused by the simplification of control schemes within games. Through analysing theories of flow and engaging gameplay, a 2D puzzle game is designed which allows two different sets of control schemes to function within the game. The goal in the game for the user is to find an exit to a maze, and while inside the maze cast spells using the different control schemes. Through testing, the two versions were compared with each other while also testing for the presence of engagement.

**Copies: 3**

**Pages: 85**

**Finished: 5/27/2011 09:18 AM**

## Preface

This project has been developed during the 10<sup>th</sup> semester of Medialogy at Aalborg University Copenhagen, in the period from the 2<sup>nd</sup> of February 2011 to the 27<sup>th</sup> of May 2011. The project is based on a personal interest in games and interfaces. The product is based on an investigation into the effects on engagement through the simplification of a control scheme in a game.

The report will detail and discuss the various steps taken throughout the production of the product. The report covers areas such as engagement, game theory, game design and game programming.

## Reader's Guide

This report is organized to follow the structure of the actual work progress of the project. Thus the chapter Introduction and Pre-analysis will explain the investigations and considerations done in order to arrive at the final problem formulation. The analysis chapter will investigate deeper into the areas of interest discovered in the pre-analysis, and the results of this will serve as a basis for creating a set of solution requirements for the design of the product. Once the application had been design and developed, a pretest was conducted which resulted in a second iteration the design and implementation phases and the information gained throughout this pre-test can be found in the implementation chapters. The test chapter explains the final test setup, list the results acquired from the test, as well as analyze and conclude on the result. In the end, the final conclusion is presented.

Every main chapter will contain a small description of what the given chapter will cover as well as how it is structured.

The header contains the group number as well as the project title. The footer contains the current page number in the outermost corners.

The APA Quotation standard is used when referring to sources and quoting: "To be, or not to be, that is the question" (Shakespeare, 1603). When a source is used directly in the text, such as: "(Shakespeare, 1603) is one of the great pieces of literature in history", it means that it should be read as a part of the sentence, but still as a reference to the content of the source. The full list of sources can be found in Chapter 8 *Bibliografy*.

When referencing to other chapters or illustrations the text is italic, as it can be seen in the line above.

Illustrations are numbered X.Y, where X is chapter number and Y is the number of the illustration in the specific chapter. Illustration text is easily distinguishable because the font is blue and the font size is smaller than the body text, which is True type font "*Times New Roman*" in size 11. A full list of illustrations with sources can be found in chapter 9 *List of illustrations*.

Code examples and variables are formatted with `Courier New` and the background is grey. Furthermore when multiple lines of code are represented, a line number can be found on the left side.

Chapter 10 *Appendix* contains the questionnaire, some screenshots showing the various versions of the program and a part of the source code. Additional data including more source code, larger images and the full set of test data acquired can be found on the enclosed CD-Rom.

## Table of Content

1. Introduction .....	4
1.1 Motivation .....	4
1.3 Related work.....	5
1.4 Initial Problem Formulation .....	7
2. Pre-analysis.....	8
2.1 Engaging Gameplay .....	8
2.2 Simplification .....	10
2.3 Game Genre.....	12
2.4 Target group .....	13
2.5 Delimitation.....	13
2.6 Final problem formulation.....	13
3. Analysis .....	14
3.1 Methodology.....	14
3.2 Engagement .....	14
3.3 Game Theory .....	20
3.4 Game Programming.....	23
3.5 Analysis conclusion.....	24
3.6 Solution requirements.....	25
4. Design.....	25
4.1 Game Design .....	25
4.2 Level Design.....	28
4.3 Visual Design .....	29
4.4 User Interface .....	30
4.5 Software Design .....	33
4.6 Design conclusion .....	36
5. Implementation.....	37
5.1 . Game Programming.....	37
5.2 Visual Implementation .....	40
5.3 2 <sup>nd</sup> Implementation iteration .....	42
5.4 Implementation Conclusion.....	45
6. Test .....	46
6.1 Usability testing.....	46
6.2 Engagement testing .....	46
6.3 Test Conduction.....	48
6.4 Test Results .....	48
6.5 Test Analysis .....	56

An investigation into the effects on engagement,  
caused by control scheme simplifications in games.

7. Conclusion.....	59
8. Bibliografy.....	61
9. List of illustrations.....	64
10. Appendix .....	65
Appendix.I - Questionnaire .....	65
Appendix.II - Divided preference graphs .....	66
Appendix.III - Screenshots of rooms.....	67
Appendix.IV - Game.cpp .....	69
Appendix.V - Player.cpp .....	80
Appendix.VI - GameBoard.cpp.....	80
Appendix.VII - EventHandler.cpp.....	82

## 1. Introduction

This project will be working with aspects relating to user interfaces in video games, more specifically with the interaction between the user and the system. This introduction will provide a brief description of the overall idea for the project and why the topic of interfaces in video games is a topic that is worth investing.

While the definition of an interface is straightforward, the term covers many different areas of applications such as the interfacing between different pieces of hardware, software or even man and machine interfacing through peripheral input devices such as a keyboard and mouse combination for personal computers, joystick for video game consoles and in recent times through motion tracking which can be applied to all of the before mentioned platforms. This chapter will look into examples of interfaces in video games in order to determine the state of the art, and the direction in which interfaces in video games are heading towards.

Once the fundamentals of this area have been examined, the chapter will be summed up by stating an initial project formulation for the project that will serve as a guide for the investigations for the following *chapter 2 Pre-analysis*.

### 1.1 Motivation

When the first video games started emerging, as the games started becoming more complex so did the controls - for instance the Atari 2600's joystick had one button while the Super Nintendo Entertainment System had a directional pad and eight buttons - making it more difficult for users to quickly learn how to use the controller and be able to just sit down and play immediately. However in recent years this trend has been reversing with the introduction of more intuitive control schemes such as the dance mats from Dance Dance Revolution, the guitar controllers from the Guitar Hero franchise and with the many games released for the Wii which utilize natural movements of the Wii Remote rather than complex button inputs to play the games (Glinert, 2010). This has also caused the games to shift from a "hardcore" to a more "casual" perspective, allowing people who do not generally play video games to take part of the fun without having to read hefty instruction booklets or long guides to comprehend how to play.

However is it possible that this simplification causes an effect on the engagement experienced by the users when playing the games? While searching for existing theories on the topic, it seemed an impossible task to find one specifically aimed at this area of user interfaces in games. There was a plethora of research on how to create user interfaces that would be accessible for the users and provide good usability. However none of them seemed to be focused on the correlation between experienced engagement and the simplification of user interfaces. As such it would be an interesting area to investigate for this project



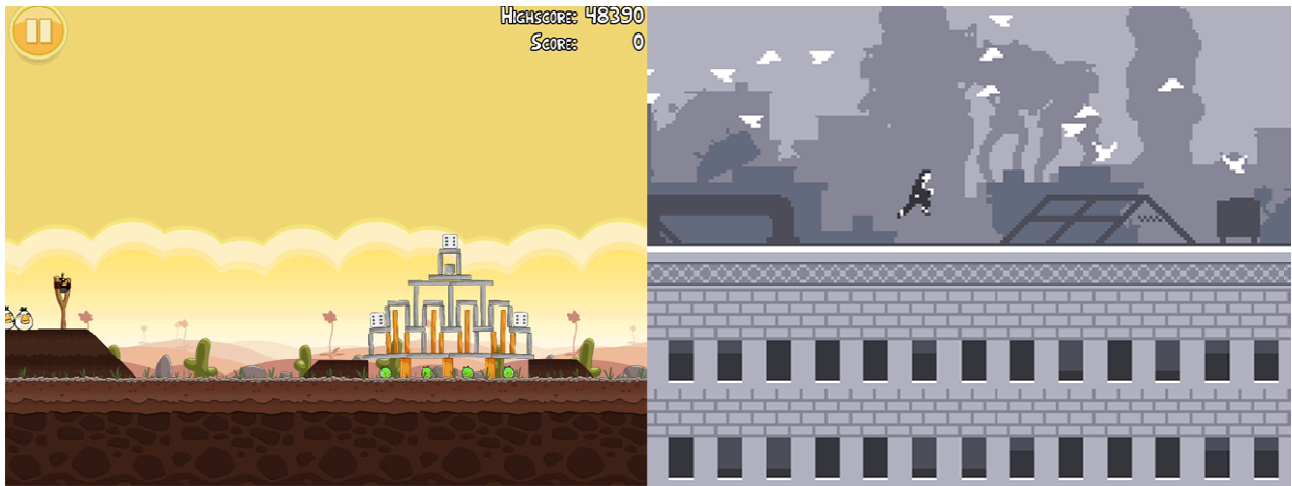
### 1.3 Related work

In order to get an overview of the simplification process that has happened in games within the most recent years, it would be prudent to look at how game user interfaces used to be in the past and the present. With the shift in focus from complexity to simplicity, the games themselves have evolved in a more simplistic direction. Rather than contain complex simulations and sophisticated menus in the games, the games themselves have moved towards allowing quick play sessions ranging from a few minutes to half an hour, based on the low learning curve required.



*Illustration 1.1: Two older games, Diablo 2 (Left, 2000) and Transport Tycoon Deluxe (Right, 1995)*

In *Illustration 1.1* two games that are over a decade old, are shown to illustrate the complexity of games from the past decade. To the left in the illustration is Diablo 2 (Blizzard Entertainment, 2000), a hack and slash type role-playing game in which the player needed to get to know not only the best way to increase the strength of their character, how to play the game and fight the enemies but also the extensive inventory system with items that when combined in different ways provided different beneficial bonuses to the players character. Aside from this, the game used a multitude of keyboard and mouse inputs in order to control the game. Transport Tycoon Deluxe (MicroProse Software Inc., 1996) as seen on the right side of *Illustration 1.1*, was a company management simulation in which the player had to develop and expand their company through transportation of goods from raw material source, to factories and finally to the citizens of the various cities in the game. Other than the financial aspect, the user constantly had to keep an eye on their assets – such as busses – and replace them once they suffered break downs or replacing factories or production pipelines when supply sources went dry. There was also a sense of competition in the shape of either AI opponents or rival human players with their own companies, who would attempt to bankrupt the player in any way possible. Both of these games are more than ten years old, and contain a high degree of complexity despite that they are from two very different genres. The learning curve of both games could be considered steep as there was no instructional guidance as to how to play it – thus making the games leaning towards what was referred to as a more “hardcore” perspective by (Nutt, 2008).



*Illustration 1.2: Two more recent games, Angry Birds (Left, 2009) and Canabalt (Right, 2009)*

In *Illustration 1.2*, two games from within the past few years are shown to illustrate the current trend within many recently released games. Angry Birds (Rovio, 2009) shown in the left of *Illustration 1.2*, is a single button action game in which the player by shooting birds at different structures attempt to eliminate the pigs located within. The game provides a quick instruction on how to fire the catapult with the birds and how to use the special abilities of the birds. Originally Angry Birds was an Iphone game, and the controls were aimed at using a single finger through either presses or by dragging the finger across the catapult in order to pull back the sling. The game was later ported to multiple devices both within the mobile phone area such as for use with phones containing the Android operating system, personal computers using Windows and Mac, but also to consoles such as the Playstation 3 and Playstation Portable. All of these platforms use a similar concept for the control scheme, using a single button for the input and either pulling back on the catapult using the finger, analog sticks or the mouse. Canabalt (Semi Secret Software, 2009) shown on the right of *Illustration 1.2* is a single button game in the strictest meaning of the sentence. The game procedurally generates buildings which the players have to jump between in a side scrolling running game. Originally a flash game, the input used in the game was centered on the left mouse button, having the functionality of making the players avatar jump. In comparison to Angry Birds, there is no objective or levels as such in Canabalt with the only goal to run as far as possible before dying. These games are what would be referred to as “casual” in the previously mentioned gamasutra article by (Nutt, 2008), and are easy to sit down with and contain little to no learning curve. However they still manage to provide a challenge for the users and as such are more approachable for users not accustomed to playing video games. In the cases of the previously mentioned games, both the gameplay mechanics and the control schemes vastly differ in complexity. However there are also games with more similarities such as Battlefield Heroes and Gears of War, in which both of the games are third person shooters. Battlefield Heroes is a game for Windows which utilizes all the numerical keys available in order to change weapons and use items. Gears of War is present on multiple platforms, namely Windows and Xbox 360, and only features two weapons and a grenade button. In order to deal with the lower amount of available buttons on a gamepad, the weapon handling mechanics and control scheme were simplified to fit the platform.

For all the mentioned games, despite their simplification they still managed to become popular games with most of them spanning several sequels. Thus it seems as though if the complexity of the gameplay mechanics



An investigation into the effects on engagement, caused by control scheme simplifications in games.

and the control scheme are scaled in unison, the game will still be engaging for the users. But is it possible that the above games could have been off for better or worse if they had used a simpler or more complex control scheme? What if Angry Birds added an additional button for handling the inputs, would that make the game less or more engaging?

### 1.4 Initial Problem Formulation

Based on an interest into investigation the effects on engagement through the simplification of gameplay mechanics, the initial problem formulation for the project will be as follows:

How can I through the simplification of gameplay mechanics in a video game, determine the effects on engagement?

## 2. Pre-analysis

In this chapter, the key aspects contained in the initial problem formulation will be explored further in order to narrow the scope of the project and create a final problem formulation. The two most important terms to be derived from the initial problem formulation are the words gameplay and simplification. Both of these topics will be researched in relation to engagement. A definition of the simplification process will be defined, and a method for evoking engagement through gameplay in video games will be explored. Different video game genres will be investigated in order to determine a suitable genre for the project scope.

Following the investigation towards the game centered aspects of simplification and engagement, a target group will be defined for the project based on statistic facts related to people of different ages and their experience of playing video games.

### 2.1 Engaging Gameplay

As the initial problem formulation stated, the project will be concerned with the effects of simplifying gameplay mechanics and the effect that this will have on engagement in video games. Thus it is necessary to investigate how to create engaging gameplay.

The word engaged can be interpreted in many different ways according to the Merriam-Webster dictionary.

*[...]to hold the attention of*  
*[...]to take part*  
*[...]to give attention to something*  
*[...]to begin and carry on an enterprise or activity*  
(Merriam-Webster's Online Dictionary, 2011)

The definitions from Merriam-Webster state that engagement is something that requires the user to partake in an activity, which can hold the users attention. This leads to the question of how to create an activity which a user has a desire to partake in. While the motivations for playing video games can be different between each individual user, there exists a common denominator in games which the users always have to face. Every game has some sort of a conflict that the users must overcome in order to complete the game. In Super Mario Bros. (Miyamoto, Shigeru, 1983) the user has to traverse dangerous worlds and fight evil monsters in order to save the princess, in Pacman the user has to eat all the small cheese pieces before being eaten by ghosts (Namco, 1980). The problem with creating these challenges lies with creating a balance between the player skill and the difficulty of the challenges. This is what Mihaly Csikszentmihaly referred to as *flow* in the book *Flow: The Psychology of Optimal Experience* (Csikszentmihaly, 1991), which has served as inspiration to additional approaches on how to create flow (Chanel, Rebetez, Bétrancourt, & Pun, 2008). This theory and how it will be used will be discussed further in Chapter 3.2.3 *Flow*.

Csikszentmihaly theorizes that a balance between the user skill and the difficulty of a challenge is important to keep the user engaged in activity. According to (Salen & Zimmerman, 2003) the challenges have to be meaningful for the player. Meaningful play emerges with the interaction between a user and a system, and as such the input that the user makes to the system should provide a meaningful outcome. Salen and Zimmerman has divided the types of feedback into two separate

An investigation into the effects on engagement, caused by control scheme simplifications in games.

categories. When the system provides immediate feedback from an interaction, it is known as discernible feedback. If the feedback is delayed and will influence the system at a later time, it is called integrated feedback.

Thus the main problem in relation to creating engaging gameplay for the project lies within making gameplay which is both balanced in terms of the difficulty of the challenges as well as making the gameplay meaningful to the player. In order to overcome this hurdle, games consist of different elements such as storylines, visual styles, soundtracks etc. If none of the users actions in a game has any influence on the later parts, it can be difficult to implement integrated feedback. An example of a video game that has a high use of both integrated and discernible feedback is *StarCraft 2: Wings of Liberty* (Blizzard Entertainment, 2010). In *StarCraft 2* the gameplay is mainly oriented around amassing an army in order to vanquish the opposition in a given scenario. From the beginning of a skirmish everything in the early game will have a colossal influence on the later stages. The tactics employed by the player can make or break a victory, as well as the choice of which units to create for the coming battle. Acquiring a second resource pile might ensure you a strong position later in the game, but will leave you vulnerable early on. Discernible feedback is experienced in instances such as attacking an enemy or moving, in which a unit will attempt to eliminate an opponent or immediately move to the desired location.



**Illustration 2.1: *StarCraft 2* (right) is a game which relies on both discernible and integrated gameplay, while *Cogs* (left) mainly relies on discernible gameplay with each level completely separated from the next.**

However not all games rely as heavily on integrated feedback. *Cogs* (Lazy 8 Studios, 2009) is a puzzle game in which the user by relocating small plates with cogs or pipes located on them, can start up various contraptions. Each level features a single contraption and while the user completes the contraption the amount of moves used as well as the time taken is recorded. While a quick completion can net the user a gold medal which will show up on the level select screen, it has no other integrated parts in the game whatsoever.

These games come from two very different genres and with greatly varying aspects. They also serve to provide two examples showing that whether a game is small or complex, it can contain meaningful play provided that the challenges are balanced, and that the game contains discernible, and to some extent integrated parts.

For the scope of this project, building a game such as StarCraft 2 would be impossible due to the sheer amount of resources and developers needed, as well as several years of time to develop the game in. As such it is much more feasible to create a prototype of a game, in which the simplification of gameplay mechanics can be attempted.

## 2.2 Simplification

Ernest Adams, renowned game designer and lecturer at the Game Developers Conference, wrote an article back in 1999 on the simplification in games (Adams, The Designer's Notebook: Simplification, 1999). Here he used the popular board game Monopoly as an example on how simplification can enhance the engagement in a game. In its core Monopoly is a real estate game with features such as trading building lots, constructing houses and hotels and eventually bankrupting the opposition. However it is a simplification of actual real estate, devoid of the less entertaining subjects such as taxes, insurances and building inspections.

*“Reality is complex and difficult. Games are supposed to be easy and fun. To get from one to the other, you have to simplify.”*

(Adams, The Designer's Notebook: Simplification, 1999)

The article also mentions another important reason for simplifying games: To make them more accessible. An example of this is the board game Balance of Power, in which rather than focusing individually on each country, separated them into two groups supporting either USA or USSR. In this way there was only one player and one enemy, making the game easy to comprehend. This provided two buzzwords to remember when the design of the product should begin, find an idea and simplify it and make it easily accessible.

In order to create a clear definition of what the term “simplification” covers for this project, a definition from the *Merriam-Webster* online dictionary will be used:

*“[...] to reduce to basic essentials”*

*“[...]to diminish in scope or complexity”*

(Merriam-Webster's Online Dictionary, 2011)

When talking about video games this definition of simplification is suitable, however some delimitation to the scope will have to be imposed in order to relate it to the video games of today. Reducing a game to its basic essentials can relate to everything from graphics, sounds, gameplay and controls. However not all alterations come in the form of simplification. Two examples of this are the games Doodle Jump (Lima Sky, 2009) for the iPhone and the pc game Police Quest: In Pursuit of the Death Angel (Sierra On-Line, 1992).



**Illustration 2.2: Doodle Jump (left) with its simplistic graphics, and Police Quest (right) which used silence and sound for atmosphere.**

Doodle Jump is created with simplistic graphics resembling those any child would be able to scribble on a piece of paper while playing. It is these graphics that set the mood of the game and its theme and thus calling them simple solely because they are not 3D graphics or push the hardware to its limits would be debatable. Police Quest is an older game which used sound sparingly. Other than to notify the players of events or as feedback from interactions, the game was silent. However this caused a major effect on the atmosphere on the game when pulling over a suspect's car in the game. One minute the siren of the police cruiser is wailing, and in the next minute a deafening silence as the police officer decides how to approach the subject.

These two video games prove that alterations in graphics and different approaches to sound cannot be explicitly reasoned as a simplification, as these changes can be used to directly influence the gameplay experience. As such for this project, other areas for simplification will be looked at in order to determine an area to work within.

In order to overcome this, a few delimitations can be imposed in order to make a clear definition for this project as well as a guideline for the product to be created. First it can be established that a video game needs graphics, no matter how simple it is. One of the last commercial text-based games (not created by a single person as flash games or similar) was Spy Snatcher (Partington & Thackray, 1991). With the current trend of multi-platform video games and free to play massive multiplayer online games, it is highly implausible that a text based game would be accepted. Sound can be implemented if it is deemed necessary, and only in the amount that is necessary. This has been proven an effective method of using sound as seen in the xbox360 platform game Limbo (Playdead, 2010) which used sound sparingly, mainly to depict dangerous elements and the death of the user's avatar. The control scheme is the part of video games which in the past years have seen the most simplification, in order to enable multi-platform releases. This is also somewhat tied up to a simplification of gameplay mechanics. In 2007 a game called Crysis (Crytek Frankfurt, 2007) was released for the pc. In the game the user controlled an avatar equipped with a piece of equipment called a Nano-Suit. This suit allowed them to enhance different aspects of the character such as super strength and speed. Each part could be accessed individually giving the player the possibility to tackle each situation based on their own customized approach. The sequel – Crysis 2 (Crytek Frankfurt, 2011) – was released as a multi-platform game. In order to accommodate joypad controllers instead of keyboard and mouse inputs, a few things were changed in the game. Rather than being able to freely activate strength and speed in the suit, these were altered to automatically activate when jumping or running. While this can be considered a streamlining of the inputs, it also meant that some of the features – such as throwing a people high into the air with super strength – had to be left out of the game. This makes it interesting to see what effect this simplification of



content in the game had on its engagement. As such it would be prudent to focus the direction of the product of this project towards the effects of simplifying the gameplay elements through the controls of the game and test to what extent this effects the engagement of the game. This results in the following definition of simplification to be formed for this project:

**Simplification is reduced complexity of the control scheme and gameplay elements.**

## 2.3 Game Genre

Based on the motivation described in Chapter 1.1 *Motivation*, it is necessary to find a game genre that allows both a highly complex control scheme as well as a more simplistic approach while preferably keeping the gameplay as close to identical between the two versions as possible. Many genres have seen games simplified for use on different platforms. The strategy genre is a genre that has mostly existed on the pc with series such as Red Alert (Westwood Studios, 1996) and Starcraft (Blizzard Entertainment, 1998). However the genre has attempted moves to other consoles through methods mainly focused on alterations of the control scheme such as Tom Clancy's EndWar (Ubisoft Shanghai, 2009) which in order to complement the joypad controllers on the console also allowed the majority of the game to be played through voice commands. The biggest problem with simplification in that strategy games consist of highly complex rule sets that dictate the game.

Role-playing games are usually focused around complex stories and the gameplay can differ highly on a game to game basis. Another factor that these two genres have in common is that due to this complexity the development time for such a title would be impossible to finish within the scope of this project. An exception to this is a 2d sub-genre of Role-playing games mentioned in the book On Game Design (Adams & Rollings, On Game Design, 2003) called Rouge-like. This subgenre of RPGs eliminates the story element from the genre and focus mainly on character development, which is supported in an article by Alison McMahan (McMahan, 2003), in which she states that in order for a game to be engaging, it is not essential for the narrative to be a key component.

The Action genre relies on the skill of the player as well as the speed and the simplicity of the game. These games are usually divided into either First-Person Shooters or Third-Person Shooters and usually involve a storyline. These storylines however are much lighter than those found in roleplaying games, and vary from a mild focus such as Unreal Tournament 2004 (Epic Games & Digital Extremes, 2004) – in which the story is simply to lead a team of soldiers to win a killing tournament – to Quake III (id Software, 1999) which has no story whatsoever. While FPS and TPS action games usually rely on 3d graphics, other types of action games exist which focus on 2d graphics such as the side-scrolling shooter R-Type (Irem, 1987) or the platform game series Super Mario Bros (Miyamoto, Shigeru, 1983).

The puzzle genre relies mainly on the skill of the player. Here the player can usually take their time in figuring out how to solve the challenge they are faced with. Puzzle games don't require storylines or fancy graphics, as long as they can provide the user with a challenging set of tasks to complete. Minesweeper is an example of a classic puzzle game. Through hints in the shape of numbers, the player has to figure out which boxes are mines and which ones are safe until the entire minefield has been cleared. There is no punishment for solving the game slowly, and if the challenge was found to be too easy both the amount of mines and the size of the minefield can be increased.

Due to the low complexity on aspects other than controls and gameplay, the action, rouge-like and puzzle genres should be considered a viable candidate for the product of the project.

## 2.4 Target group

As this project will be working with the effects of simplification on engagement in video games, it would be prudent to define a target group that plays video games on a regular basis in order to achieve valid test results. Furthermore, it would also be prudent to focus the design of the product towards the target group in order to create relevance between the product and the users.

The target group will be defined based on statistical information about people who play video games. The Entertainment Software Association (ESA) releases a yearly report called *Essential Facts About the Computer and Video Game Industry*, that lists different facts about video game players such as age, genre popularity and gender division. The data in this report is based on the American population, of which 67% play video games (ESA, 2010). In the report covering the year 2009, it is stated that the average age of a video game player is 34 year old, with 49% of all video game players being between the ages of 18 to 49.

These numbers provide a usable base for defining the target group. By using the statistics to define a target group aimed at video game players in the age range of 18 to 49 would provide a valid description of the persons that this project will aim towards. It would be possible to use people outside this age range, which also constitutes to 51% of the people who play video games. However at the age of 18 to 49, it is more likely that they are skilled with using computers and more accustomed to playing video games – regardless of the genre. Using too high an age group could result to usability problems due to lack of knowledge of how to operate computers, and using too low an age group would restrict the type of content and difficulty as it has to be understandable by children.

Having decided upon the target group, and delimiting the choice of genres to either action, puzzle or rouge-like, it possible to define a final problem formulation.

## 2.5 Delimitation

Based on the investigations made in the pre-analysis, it is possible to delimit the initial problem formulation into the final problem formulation. Through the investigation into engaging-gameplay the problem area has been narrowed down and can be researched more thoroughly in the analysis chapter. The genres were narrowed down to action games, puzzle games and rouge-like games. For the sake of simplicity, the action genre and the rouge-like genre will be abandoned and the primary focus will be on the puzzle genre. This will allow the product to function in a 2d environment without the necessity of enemy AI as in action games or rouge-like games.

## 2.6 Final problem formulation

How can the effects on engagement be determined, in which the control scheme of a 2D puzzle game is compared to that of its simplified counterpart?

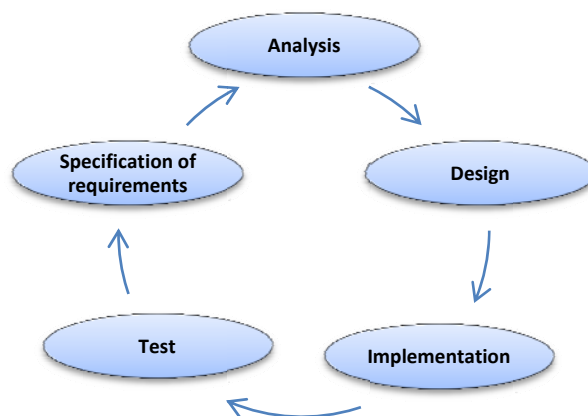
### 3. Analysis

Based on the final problem formulation, the analysis will further investigate the areas of interest, in order to be able to answer the final problem formulation and produce a design. Initially, the analysis will describe the work cycle of the project, to give the reader an understanding of the development of the project.

The term “engaging gameplay” was looked into in the pre-analysis Chapter 2.1 *Engaging Gameplay*, however further investigations into the overall term “engagement” is needed in order to be able to test the engagement of the users. As the product of this project will be a game, theories related to games and meaningful play will also be investigated. In order to create a game, it is also a necessity to acquire technical information related to game frameworks in order to determine which tools and applications can be used in the design of the product.

#### 3.1 Methodology

This project will use an iterative, user-centered development model as shown in *Illustration 3.1*, which loops through the analysis, design, implementation and test until the product suggestion is able to satisfy the final problem formulation. The final iteration’s test will be used to determine the legitimacy of the project and the prototype product.



**Illustration 3.1:** Graphic representation of the iterative model showing the workflow. Illustration borrowed from a previous report (Jensen, Etzerodt, Christensen, & Jørgensen, 2009).

The testing in this project will be used for a multitude of purposes. User testing will be used to support design decisions, should these lack proper argumentation through research as well as obtaining the data that can be used for the proving or disproving the final problem formulation. Usability testing will aim towards revealing any usability errors that the application might contain.

Before tests can be performed and the project re-iterated, it must go through its’ first iteration. This iteration will introduce new theories through analysis of various parts of the final problem formulation. These theories will then be used for design of the product, before this product is subjected to user-testing. The first of these theories will be related to engagement, which will be important in order to determine and test whether or not the product will be engaging.

#### 3.2 Engagement

When working with engagement in the area of media and interaction studies, a tangible way of measuring the results is needed.

An investigation into the effects on engagement, caused by control scheme simplifications in games.

*“In the past few decades, human-computer interaction studies have emphasized the need to move beyond usability to understand and design for more engaging experiences”*

(O’Brien & Toms, 2008, s. 938).

Heather L. O’Brien and Elain Toms, PhD graduates from Dalhousie University are attempting to create a framework and definition of engagement in the paper: *“What is User Engagement? A Conceptual Framework for Defining User Engagement with Technology”* (O’Brien & Toms, 2008). In order to define engagement, O’Brien and Toms used semi-structured interviews as phase one in a multistage research study aimed at the thoughts, behaviours and feelings of the test participants. The test participants were all asked to recall a time in recent memory in which they felt engaged while using an application. O’Brien and Toms also researched theories based within flow, games and information interaction and aesthetics in order to determine which parts of each theory could be related to the term engagement.

The result of their semi-structured interviews led to a framework of key factors that make applications engaging. While the test cannot be considered as a definite way due to the test only being conducted on 17 participants, it can however serve as a guideline for a final test on the level of engagement. The model divides into three threads of experiences focusing on either the sensual, emotional or spatiotemporal. In the framework O’Brien and Toms list different guidelines on how to measure and create engaging products.

Threads of experience	Compositional thread		
	Process of engagement		
	Point of engagement (and Reengagement)	Engagement	Disengagement
Sensual	<ul style="list-style-type: none"> <li>Aesthetic elements are pleasing or attention getting</li> <li>Novel presentation of information</li> </ul>	<ul style="list-style-type: none"> <li>Graphics that keep <u>attention</u> and <u>interest</u> or evoke realism</li> <li>“Rich” interfaces that promote awareness of others or <u>customized views</u> of information</li> </ul>	<ul style="list-style-type: none"> <li>Inability to <u>interact</u> with features of the technology or manipulate interface features (usability)</li> <li>Lack of/too much <u>challenge</u></li> </ul>
Emotional	<ul style="list-style-type: none"> <li>Motivation to accomplish a task or to have an experience</li> <li>Interest</li> </ul>	<ul style="list-style-type: none"> <li>Positive affect: enjoyment, fun, physiological arousal</li> </ul>	<ul style="list-style-type: none"> <li><u>Negative affect</u>: Uncertainty, information overload, frustration with technology, boredom, guilt</li> <li><u>Positive affect</u>: Feelings of success and accomplishment</li> </ul>
Spatiotemporal	<ul style="list-style-type: none"> <li>Becoming situated in the “story” of the application</li> <li>Ability to take one’s time in using the application</li> </ul>	<ul style="list-style-type: none"> <li>Perception that time passed very quickly</li> <li>Lack of <u>awareness</u> of physical surroundings</li> <li>Strong <u>awareness</u> of others when the engagement revolved around social interaction</li> <li><u>Feedback</u> and <u>control</u></li> </ul>	<ul style="list-style-type: none"> <li>Not having sufficient time to interact with or time to devote to the application</li> <li><u>Interruptions</u> and distractions in physical environment</li> </ul>

**Tabel 3.1: Proposed model of engagement and its attributes. (O’Brien & Toms, 2008, s. 948)**

Table 3.1 depicts the points of engagement retrieved through the conducted test. The points generally cover very broad areas, but are still usable as guidelines for possible designs and implementations. The sensual experience thread is primarily focused on the visuals of a product such as graphics and interfaces. The sensual thread matches the decision proposed in Chapter 2.2 *Simplification*, that graphics are an integral part to engagement, in terms of both aesthetics and interface. Problems with the usability of the product are one of the primary disengagement factors.

The model can be used in order to test the engagement of the user through such elements as lack of perception of time or awareness of physical surroundings. The points listed under engagement are viable as

guidelines for the designing of an engaging application by including such points as the “Graphics that keep attention”, “Feedback and control” as well as avoiding the elements listed under disengagement such as “Lack of/too much challenge” or the “Inability to interact with features or manipulate interface features”.

### **3.2.1 An alternate approach to Engagement**

Brown and Cairns describe engagement from a different point of view in their article “A Grounded Investigation of Game Immersion” (Brown & Cairns, 2004), which is based off a grounded investigation. From the results of their investigation, they devised a three part model for immersion. Immersion to Brown and Cairns is used to describe the degree of involvement with a game. The three parts composing their immersion model is engagement, engrossment and total immersion. Between each part is a set of barriers, which prevents the user from proceeding to the next degree of involvement. However Brown and Cairns state that even if the barriers are removed, it does not guarantee that the user will experience a higher degree of immersion. In relation to this project, the most interesting aspect of their theory is related to the initial stage of engagement, and this subchapter will therefore focus on this segment of the theory.

The barriers that prevent the users from entering engagement, is initially according to Brown and Cairns access. If a player does not like the specific style of a game, they will not attempt to engage with it. This can be related to the genre of a game, in which a user who prefers playing fast paced action games might not feel very well at ease if situated with a slower paced strategy or puzzle game. Access also refers to the controls and feedback provided in the game. If the user has no idea as to how they should interact with the game, and it provides no recognizable feedback – the user will not be able to become engaged.

The second barrier to the engagement stage is the investment made by the gamer, in order to learn how to play the game. Without putting any effort into the game, the user will have no idea how to play it and will most likely lose interest very quickly. However if the user puts an effort into learning how to play the game, a reward will be expected that justifies the energy used. According to Brown and Cairns, once these conditions of access and investment have been passed, the user can begin to feel engaged.

This theory contains some similarities with the framework proposed by O’Brien and Toms. Cairn and Brown state in the access barrier that controls interacting with a system and appropriate feedback from these interactions are required to pass the barrier. O’Brien and Toms state a similar fact in that the inability to interact with the system would result in disengagement. While there is a difference in their phrasing, it seems likely that they both consider the interaction and feedback between the user and the system as being an integral part of creating engagement. In the framework by O’Brien and Toms it is stated that the emotional aspects that creates a point of engagement is motivation, and interest. This can be directly associated with Cairn and Browns second part of the access barrier as well as the second barrier of investment, in which only a user with an interest in the style of the game will be motivated enough to invest time in learning how to play the game. Without motivation or interest in the game, they are highly unlikely to create a point of interest and therefore not be engaged.

### **3.2.2 Deconstruction**

The model proposed by O’Brien and Toms provide a general definition of user engagement, defining what the different elements of a media product play on the engagement level. However it would be relevant to deconstruct the term “engagement” in order to provide a clearer guideline to work towards in the design and implementation phases of the project.



An investigation into the effects on engagement, caused by control scheme simplifications in games.

The social aspect of engagement is one that is highly agreed upon, as having positive effects based on its presence in a video game. In an article on the difference between high engagement and addiction to playing online video games by John Charlton and Ian Danforth (Charlton & Danforth, 2007), they discuss how the presence of social interaction is a key point to ensuring engagement in the users. They are however, not the only ones to arrive at this conclusion. While within different fields of research related to engagement, such as online games (Chen, Duh, Phuah, & Lam, 2006), physical activity and games (Lin, Mamykina, Silvia, Delajoux, & B., 2006) and design (Dickey, 2005), these different studies all point towards the beneficial effects of social elements in relation to engagement. In video games the social interaction usually takes place in the shape of playing together with other players directly as seen in online games such as World of Warcraft (Blizzard Entertainment, 2004), network or locally as in games such as Halo (Bungie, 2001) or in a less direct way through high-score tables as found in most types of arcade games or online flash games. All of these point towards the beneficial effect of social interaction in whatever form it may take, has on the engagement of a video game. As such, it would be sensible to include the social element in the product of the project.

The interactive aspect is the interaction between the player and the feedback from the video game. In *Tabel 3.1*, problems related to the interaction between the user and the interface were a direct factor in the disengagement of users. As such it is prudent to consider the interaction between the user and the video game as a component of engagement. In the article by Emily Brown and Paul Cairns (Brown & Cairns, 2004) the initial barrier to achieve engagement is through access, by which they point towards the interaction between game controls and feedback and how these should correspond in order to allow the users to become experts with the main controls. Considering that the final problem formulation for the project relates to working with the simplification of a control scheme, it would be sensible to ensure that the initial interaction with the user and the game is properly functioning.

The narrative aspect of engagement is one that can be debated. As mentioned in Chapter 2.3 *Game Genre* some genres do not even include a narrative, and while McMahan (McMahan, 2003) states that the narrative does not need to be a key point in a video game, it does not mean that it is without its uses. One of the most successful games of all time is the massive multiplayer online roleplaying game World of Warcraft (Blizzard Entertainment, 2004) with over 12 million players (Blizzard Entertainment, 2010). This game is highly based on the narrative, which depicts a struggle between two factions and the player's role in it. According to Michele Dickey (Dickey, 2005), one of the elements that induces engagement is the narrative. This points towards that completely ignoring this component would be an inconceivable act, and that the narrative component should be used if not as a central aspect, then as a supporting component in order to enhance the engagement of the video game.

The visual aspect of engagement is an integral part of any modern video game. O'Brien and Toms depict the visual component as a method of acquiring and then keeping the users attention on the media product. Marshall Jones from the University of Memphis writes in his paper "Creating Engagement in Computer-based Learning Environments" (Jones, 1998), in relation to both sound and graphics state that:

*"It is about how those assets help define, support, and give life to a domain that has no counterpart in the physical world."*  
(Jones, 1998)

Thus it would be prudent to aim towards a visual theme in the product, that can catch the attention of the user while providing them with something they cannot experience outside of this game world.

The motivational aspect of engagement is likely the most abstract. However as stated by O'Brien and Toms, it is a part of the emotional engagement listed on *Tabel 3.1*, and is by many other studies considered an integral part of video games (Brown & Cairns, 2004) (Chen, Duh, Phuah, & Lam, 2006) (Dickey, 2005). Without a motivation towards entering the game world the user is less likely to become engaged, and as such it is the task of the visual component to catch the attention of the users and motivate them to enter the world.

While the above theories cover a wide selection of approaches to engagement, it is by no means a complete guide on how to achieve it. However in order to use the above theories as a guideline for the development and testing of the project as well as creating a simple way of referring back to these theories at later stages in the report, they will be divided into the following components based on their content: The Visual, Narrative, Motivational, Interactive and the Social components.

### 3.2.3 Flow

The Flow theory was initially proposed by the Hungarian psychology professor Mihaly Csikszentmihalyi, which was mentioned in Chapter 2.1 *Engaging Gameplay*. The theory is applicable in various areas, such as the evaluation of interactive media experience.

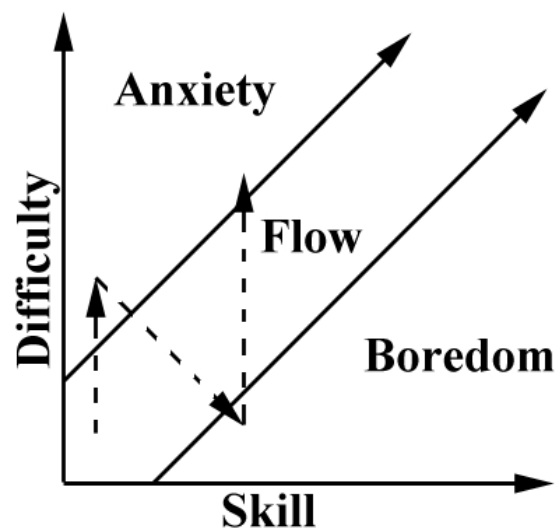
Flow appears when certain conditions (not necessarily all of them), are fulfilled:

1. A task to complete
2. Balance between ability and challenge
3. Concentration on a limited field of attention.
4. Loss of the feeling of self-consciousness.
5. People become absorbed in their activity, and focus of awareness is narrowed down to the activity itself, action awareness merging.
6. Sense of time altered.
7. Direct and immediate feedback.
8. A sense of personal control over the situation or activity.

Freely interpreted after (Csikszentmihalyi, 1991)

The flow theory allows engagement to be tested by measuring how many of the different conditions are present during a testing session. The first condition can be considered as ever present in relation to video games. All games have some task to complete, and therefore condition 1 is an integral part of any video game. Through observation during a testing session the second condition related to the balance between the ability of the user and the difficulty of the challenges, can be observed by examining whether or not the user is able to finish a level in the game. The third condition to the sixth condition are all related to the focus of the user, and can be tested by measuring how much attention the user's is paying to the media. The seventh and eight conditions are related to the usability of the product.

An investigation into the effects on engagement, caused by control scheme simplifications in games.

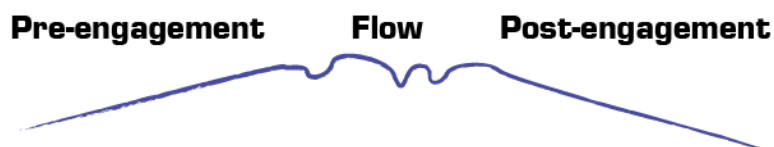


**Illustration 3.2 - Free interpretation of the fluctuating difficulty proposed in the article by Chanel, Rebetz Bétrancourt and Pun.**

An article called “Boredom, Engagement and Anxiety as indicators for adaption to difficulty in games” (Chanel, Rebetez, Bétrancourt, & Pun, 2008) proposes a method maintaining engagement based on the principles originally proposed by Csikszentmihalyi. They propose that maintaining a constant stable increase in skill and difficulty may induce boredom. As shown in *Illustration 3.2*, a method is proposed in which the difficulty will fluctuate between being hard and easy thus making the user experience a wider range of emotional states which in turn will keep them interested in continuing to play the game.

### 3.2.4 Engagement stages

The creation of engagement stages originated in a previously worked on report (Jensen, Etzerodt, Christensen, & Jørgensen, 2009). The theory of *flow* describes the feeling felt by an engaged user during an interaction scenario. However the theory does not describe the state of the mind prior to and following *flow*. I am of the belief that this should be considered as an important part of the engagement feeling, and therefore split the engagement into three different parts as shown in *Illustration 3.3*:



**Illustration 3.3: Graphical representation of the proposed different stages in engagement. (Jensen, Etzerodt, Christensen, & Jørgensen, 2009)**

**Pre-engagement** can best be described as the feeling of anticipation or arousal towards the real interaction (*flow*), based on elements such as the introduction part of a game. A part of the marketing strategy by many developers is to attempt to build up hype by releasing videos or screenshots portraying the game settings and features, prior to the release date of the game, in order to provoke this feeling in the users.

Once **flow** occurs, the users can experience feeling infuriated due to unbalanced *flow*, or the pleasant feeling of successfully completing a challenge and continuing on to the next level. This feeling of **post-engagement** is controlled by the end result of the interaction, and can provoke different types of feelings in different

users. The post-engagement can be seen by users that are longing after the next level in a game and is best described by Brown and Cairns in the below quote. Another possible result from post-engagement could be the feeling of frustration caused by an insurmountable obstacle that prevents progression in the game.

*“An engaged gamer is interested in the game and wants to keep playing.”*

(Brown & Cairns, 2004)

A thesis could be created that states that if the user shows no reaction at all to the game, the user has not been genuinely engaged. If the pre-engagement and *flow* stages are compatible, they should create a post-engagement feeling in the user. However in the case that the pre-engagement level is higher than that of the actual *flow* stage, the post-engagement feeling could be absent. A situation in which such an event could occur could be if a game was hyped as being the best game of all time, and then turns out to be nothing special, the resulting post-engagement could be minuscule.

### 3.2.5 Conclusion

The framework proposed by O'Brien and Toms, supported by the results of the investigation by Brown and Cairns, will work as the guideline for the design and implementation of the product. The framework depicts the importance of a functioning interface, and the necessity of including components that will motivate the users to engage in and complete the tasks presented in the game.

The pre-engagement can be directly used to verify if the graphical presentation of the product is balanced with the actual content of the product. Flow can be measured by examining the user's ability to complete the challenges they are presented with when testing the product. Finally the post-engagement theory can be used as an addition to the flow theory, in order to check whether or not the product mediates flow in the interaction between the user and the computer.

The post-engagement can be tested by inquiring the users whether or not they would like to continue using the product and observing the users body language: Is the user so eager to continue playing that they cannot remain calm, and do they change their posture with every new level?

When utilizing the engagement stages to test on engagement the outcome could, just as with the theory itself, be divided into three parts:

1. To which degree the visual and explicit appearance are engaging
2. To which degree the user is engaged when using the product  
And assuming that flow has occurred:
3. To which degree the user thinks the engagement are lasting after the flow occurs

After having examined the term engagement, this report will proceed to look into game theories as the final problem formulation describes the use of a game in triggering engagement in a user.

## 3.3 Game Theory

The following information has been inspired from an old report (Jensen, Etzerodt, Christensen, & Jørgensen, 2009). The product of this project will be a game, therefore it is important to investigate topics on the theory of games. When tasked with the creation of a product which is linked with a set of success criteria, that may

An investigation into the effects on engagement, caused by control scheme simplifications in games.

or may not be different from the person preferences of the designer, the design should be made to follow a set of rules and guidelines which govern part of the creation of the product.

*“[...]the game starts life as a spark in the designer’s imagination, and the idea is the single most persistent entity in the game development cycle.”*

*(Rollings & Morris, 1999)*

In accordance with the theories mentioned in Chapter 2.1 *Engaging Gameplay*, this chapter will take a look into the theory of meaningful play as presented by Salen & Zimmerman. The creation of meaningful play is essential to ensure that the actions and the feedback between the user and the system support the gameplay and engagement rather than hinder it. In order to create a base for the design of the game world, the theory of the magic circle by Huizinga will be examined.

### 3.3.1 Meaningful play

The following description of meaningful play has been written with inspiration from the work of previous report (Jensen, Etzerodt, Christensen, & Jørgensen, 2009). At some point in everyone’s lives, they’ve found themselves to be playing either by themselves or playing along with others. Play is a part of every childhood and is a natural way of interacting with others, as well as to pass time and have fun. Play is not restricted to people, as most animals also engage in play when they are infants. Play itself is often without a purpose, and can be considered as non-essential. When a child kicks a ball, it can be considered meaningless. In order for it to be usable in designing a game, it has to become more than play. It has to become meaningful. Some humans engage in sports to enhance their skills, at an attempt to become professional athletes. Likewise when animals play, they do so in order to practice skills that are needed for the sake of survival. A look at the definition of a game itself, provides some clarification as to why meaningful play should be present:

*“A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome.”*

*(Salen & Zimmerman, 2003)*

Creating a scenario for a game, in which the conflict is a settlement under attack from bandits. In this scenario the meaningful play would emerge from the resolution of settling the conflict or through the feeling of being within the world of the game.

It is important to realize that meaningful play will present itself through the actions that the user has interacted with the world through, which is then reacted upon by the game. Every action should have a reaction or outcome within the game. This is, in a sense, what Salen and Zimmerman refer to with their definition in regards the creation of meaningful play:

*“Meaningful play occurs when the relationships between actions and outcomes in a game are both discernible and integrated in the larger content of the game.”*

*(Salen & Zimmerman, 2003)*

In a scenario in which the actions of the user has no influence and provides no feedback, it would seem likely that he would feel less connected to the game. This would in term create an uninterested feeling towards the game, and would essentially be what could be constituted as meaningless play.



The actions of the users and the outcomes of the system can be one of two things: Discernible or Integrated, as described in Chapter 2.1 *Engaging Gameplay*. While discernible actions and outcomes are events in which the result can be immediately perceived by the player and can be considered as micro-choices, integrated actions and outcomes work by causing an effect in the long run. These actions can be considered as macro-choices, and an example of this could be a player in a shooter deciding not to pick up an extra box of ammunition, and then twenty minutes later into the game running out of ammo while in the middle of a struggle.

In the context of this project, meaningful play would arise from the players being able to perceive progress towards their goals based on their actions. If the users feel that their actions have no influence on the progress of the game, they will begin to feel detached from game world and risk being disengaged. Every action in the game should be supported such that it provides the user with a clear meaning of the action and reaction relationship between the inputs from the user and the feedback from the game.

### 3.3.2 Magic Circle

*The magic circle* can be thought of as a container in which everything involved in a game can be found. It is the border that divides the world of the game, and the real world. It was proposed in 1955, by a Dutch historian named Johan Huizinga.

*All play moves and has its being within a play-ground marked off beforehand either materially or ideally, deliberately or as a matter of course. Just as there is no formal difference between play and ritual, so the 'consecrated spot' cannot be formally distinguished from the play-ground. The arena, the card-table, the magic circle, the temple, the stage, the screen, the tennis court, the court of justice, etc, are all in form and function play-grounds, i.e. forbidden spots, isolated, hedged round, hallowed, within which special rules obtain. All are temporary worlds within the ordinary world, dedicated to the performance of an act apart.*

(Huizinga, 1955)

In relation to computer games, *the magic circle* includes the world, characters, objects etc. that are present inside the computer game, but also the player playing the game. In relation to real games, such as the sport game soccer, *the magic circle* includes the players, the audience, the stadium etc.

The actual content of the *magic circle*, is only objects that the user is aware of. If he is not aware of an object, he will never use it in the game. This definition makes an interesting point when taking video games into account. In a game such as soccer, every player and spectator is aware of the elements of the game at all times. However when talking about video games, every single game contains a unique world with elements specific to that individual game. While the game world itself might be known, similar to that of the stadium in soccer, the contents of the world can greatly vary. Going by this assumption it would mean that unless the player is specifically instructed on the existence and use of an element or feature in a game, it is not considered as being a part of the game at all. As such it would be prudent to keep this in mind when designing the game, in order to ensure that the players has an understanding of the possibilities that is offered within the game world rather than never experiencing them, which can also impact the flow of the game. An example of this could be a game level with a large chasm with the player on one side and the goal on the other. Unless the user has been informed that they have the ability to jump, they might attempt to figure out if there is another way to get across – which could lead to frustration as well as disengagement due to the lack of possible interactions as proposed in the framework by O'Brien and Toms. In relation to the project, this theory of the magic circle means that the user should be made aware of all the objects, rules and

An investigation into the effects on engagement, caused by control scheme simplifications in games.




interactions possible in the game. Anything the user has not been made aware of is essentially non-existent to them.

### 3.4 Game Programming

When working with a game, there are several approaches one might take when it comes to the programming. One approach is to program a game engine from scratch, which will have the benefit of supporting all the features that the individual project might require. However programming a game engine from scratch would require a gargantuan amount of time and resources, and as such is an impossible feat to accomplish during the scope of this project. However there are two other possible approaches that allow varying degrees of freedom and customization, all of which can be used to accommodate the product of this project. These types can be divided into two categories: Media Libraries and Game Engine SDKs.

#### 3.4.1 Media Libraries

Media libraries are sets of extensions that can be attached to a programming project, which allows handling of graphics, inputs, sounds and similar functionalities to be accessed without having to manually program them. The media libraries listed in *Tabel 3.2* are all based on the Open Graphics Library or OpenGL for short. The main focus of OpenGL is that it is based on the concept of being open source, which allows anyone willing to make contributions to expand the functionality or correct and optimize already existing code.

Name	Description	Platform	URL
<b>GLUT</b>	OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. Last version 3.7 from 1998.	C, C++, FORTRAN, Ada	<a href="http://www.opengl.org/resources/libraries/glut/">http://www.opengl.org/resources/libraries/glut/</a>
	Open Source alternative to the OpenGL Utility Toolkit (GLUT) library. Last version 2.4.0 2005.	C, C++, FORTRAN, Ada	<a href="http://freeglut.sourceforge.net/">http://freeglut.sourceforge.net/</a>
	Cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer	C, but works with C++ natively, has bindings to several other language	<a href="http://www.libsdl.org/">http://www.libsdl.org/</a>
	SFML is a free multimedia C++ API that provides you low and high level access to graphics, input, audio, etc.	Primarily C++, but is also available in C, D and bindings to other languages	<a href="http://www.sfml-dev.org/">http://www.sfml-dev.org/</a>

**Tabel 3.2: These Media Libraries will be compared in order to determine the best choice for the product.**

The first entry in the table, *GLUT*, is more a toolkit than a media library. GLUT is the official frontend for OpenGL, however GLUT has not been updated since 1998 and its' license does not allow anyone to distribute modified library code. The *freeglut* library is open source and contains everything that GLUT has, plus more. Both GLUT and *freeglut* are limited to OpenGL and window and input handling making it less attractive when used for games, not providing for example any sound playback abilities. *SDL* and *SFML* contain almost the same features as GLUT and *freeglut*, but are also capable of playing videos and sounds. *SFML* is written specifically for C++ and features object oriented design and features both advanced graphic options and effects, sound handling (OpenAL), surround sound and network capabilities, making it straight forward to use for this project. Another useful feature about *SFML* is that the library supports sprites, which is a lightweight bitmap image, allowing for easy use of 2d graphics. *SDL* does not have these features making it less attractive than *SFML*. While not listed on *Tabel 3.2*, a new media library called *Cocos2D* has

been released which, as the name indicates focuses mainly on creating 2d applications and games. However as very little information about the framework is available, as well as the official website being highly unstable – this media library has been discarded.

### 3.4.2 Game Engines SDKs

As mentioned in Chapter 3.4 *Game Programming* programming a game engine would take a tremendous amount of time. However some companies exist with the sole purpose of constructing game engines intended for other companies to use to create their games upon. A game engine SDK usually comes with an editor that allows easy handling of project assets such as 3d models or code as well as tools assisting in the construction of levels for the games. Some of the most well known engines currently on the market include Valve's Hammer Editor (Valve, 1996), Crytek's CryEngine 3 (Crytek, 2009), Epic Games' UnrealEd (Epic Games, 1999) and Unity3D (Unity Technologies, 2005). The main benefits of using these engines are that the amount of tools that they provide allows for very rapid prototyping from the initial idea stage to the finished product. Compared to the media libraries, the game engine sdk's provide a faster project pipeline as subjects, such as the import of models and animations are handled internally in the engine without involving the users more than selecting the desired files to be imported. However the downside to using these engines are that they require that the users adapt to the structure of the engine and should the project require the addition of additional features, it will require that the newly written feature conforms with the existing structure in the engine. As such the decision between which to use depends on the skill of the user as well as the time required versus the flexibility desired for the project.

### 3.4.3 Sum up

The decision of whether to work with the media libraries or the game engines comes down to the content and the stated final problem formulation. Aside from SDL and SFML, none of the other media libraries include all the features needed to create a game in its native forms. The game engines sdk's are all plausible for the production of the project within a limited timeframe, and contain all the same elements of the SDL and SFML media libraries. However based on a desire to program all the game functionalities from scratch as well as prior experience in working with both the media library and the programming language C++, the SFML library has been chosen to be used for creating the product of this project.

## 3.5 Analysis conclusion

Throughout this analysis, different topics have been investigated which were necessary for the project to proceed into the design phase. An initial investigation into engagement lead to a deconstruction of the term into 5 components: visuals, narrative, social, motivational and interactive. The theory of Flow by Csikszentmihalyi was also investigated, in which several ways of determining of the user has experienced flow and how this can be used for measuring engagement in the users. The theory of meaningful play by Salen and Zimmerman will be included in order to ensure that the action of the user and the feedback returned by the system is considered meaningful to the user. If the feedback action and feedback seem meaningless, the user will lose interest and become disengaged.

After looking at various approaches to game programming, both through media libraries and commercial game engine sdks it was decided that using the SFML media library would be the optimal choice for this project. The programming language used in correlation with SFML will be C++.

Through the information presented in this chapter, it is possible to create a specification of requirements for the initial design phase.

### 3.6 Solution requirements

These requirements will be elaborated in the sub-chapters of the design chapter of the report. Based on the final problem formulation and the theories investigated in the initial part of the analysis chapter relating to game theory, meaningful play and flow, the following set of requirements have been formulated:

- The theories of flow will be used to enhance the feeling of engagement by attempting to create a balance of skill and challenge in the game.
- The concept of a fluctuating difficult compared to skill level approach should be used.
- The concept of the magic circle should be kept in mind when creating the functionality of the game, and how this is conveyed to the player.
- The actions in the game should all be meaningful and assist the user in reaching the goal.
- Two approaches to a control scheme consisting of different complexity should be created.

The second part of the analysis focused on interaction and programming and lead to the following set of requirements:

- C++ will be used as the primary programming language
- The SFML media libraries will be used for handling graphics and sound

With the solution requirements in place, it is now possible to proceed into the design phase. Every aspect of the design will be based off of the presented solution requirements in order to ensure that the product will adhere to the results of the analysis and thereby fulfill the problem formulation.

## 4. Design

This chapter will focus on the design of the product. As the goal of the project is to investigate the effects on engagement caused by a simplification of a control scheme in a 2D game, it is necessary to design a game in which the investigation can take place. The first subchapter in this part of the report will focus on creating the general gameplay idea for the game. This will be followed by another subchapter in which the visual design of the product will be explained and a general use case scenario for the interaction with the game will be illustrated. The last part of the design chapter will deal with the technical aspects of the solution requirement in relation to the software.

### 4.1 Game Design

The aim of this project is to create a 2D game in which the effects on engagement by simplifying the control scheme can be investigated. In order for the test to be able to display a change in engagement in the game, it is necessary to already have engagement that can be influenced. As such it will be considered as a requirement, that an engaging game is designed. In Chapter 2.3 *Game Genre* the genres were narrowed down to action, puzzle and the rouge-like rpg. Based on that levels in rouge-like rpgs usually consist of small rooms, and the possibility to directly control challenge in the puzzle genre – a hybrid of these two will be used for this project. This sub-chapter will focus on laying out the concept and rules of the gameplay.

#### 4.1.1 Gameplay

The concept of a puzzle is to solve a given task, often with a limitation set in the form of limited moves, tools, time as can be seen in games such as *World of Goo* or *Cogs*. Creating puzzles however is no simple task, and one should keep into consideration the solution requirement from Chapter 3.6 *Solution*

*requirements* that the challenges have to be balanced to the player's abilities. As a means of assisting in the creation of puzzles, Scott Kim has made a guideline named "Scott Kim's Eight Steps" (Adams & Rollings, on Game Design, 2003) with the most important points required for creating a good puzzle:

1. Find Inspiration
2. Simplify
3. Create a construction set
4. Define the rules
5. Construct the puzzles
6. Test
7. Devise a sequence
8. Pay attention to the presentation

### Find Inspiration

Step one relating to finding inspiration is partly covered in the report, based both on the genres chosen. Rouge-like games all take place in a dungeon or cave of some sort. A puzzle can take almost any shape, from pressing buttons in a correct sequence to figuring out the timing to pass over a fire pit. However one specific type of puzzle comes to mind when thinking of dungeons or caves, mazes. All the way back to when games were beginning to test the waters of 3d graphics, even outside of the rpg genre, mazes have been used as puzzles in games. In the first Wolfenstein games (id Software, 1992), the user played a spy attempting to flee the german Castle Wolfenstein. In each level the player had to - aside from killing as many enemies and looting as much treasure as possible - navigate a maze in order to find the keys and then the exit which the key unlocked. Adventure games often take advantage of including a maze puzzle game. The adventure game The Daedalous Encounter (Mechadeus, 1995) included a maze that was randomly generated every time it was encountered. A newer title that also includes a maze puzzle is Darkstar : The Interactive Movie (Parallax Studio, 2010), which indicates that the maze type of puzzle has been around for over a decade and as such can be considered as a usable puzzle type for the product.

### Simplify

Step two refers to the simplification of the puzzle, which goes well with Adams theory (Adams, The Designer's Notebook: Simplification, 1999) on designing games in general. While the puzzle might sound exciting with all sorts of features, it is more likely to be fun when it is cut down to its bare essentials. In the case of the Maze, the essentials are quite simple – as the goal is just traversing the maze. However it might be prudent to add elements to the concept in order to ensure some level of challenge being provided by the game.

As such, aside from the initial task of traversing the maze, the following elements should be added in some form. A trap that will harm the user, ensuring that the possibility of springing a trap is always in the mind of the user – forcing them to plan ahead before venturing further into the maze. Should the user be of lower skill than the difficulty posed, the maze should also contain elements that could assist the user in reaching the exit such as rooms providing clues as to the direction or distance of the exit from the user's current position in the maze. With the possibility of a trap harming the player, an element with the opposite effect should be added for the sake of balance as well as allowing the user to backtrack to a position and retry without permanently being hindered by a mistake.

All of the listed elements require no extra input from the users other than them thinking prior to interacting with the game. As the only input at the current time is the movement keys to traverse the maze, an additional



An investigation into the effects on engagement, caused by control scheme simplifications in games.

element should be added that requires further input from the user – such that this element can be used in testing whether the simplification of the controls has any effect on engagement of the game.

### Create a construction set

The third step involves making a prototype of the game to see whether or not the concept will work. However as mazes have been used in the same shape as intended here, the assumption will be that the concept will work.

### Define the rules

The fourth step is marked in by Scott Kim as being one of the key points in puzzle design. He states that most puzzles are categorized in terms of four things: the board, the pieces, the moves and the goal. The board relates to its real world counterpart, and revolves what the game board is? The pieces relate to the attributes and information that they have been assigned. The moves relate to what moves are possible and legal within the game rules, and if there is more than one moving piece how do they move in comparison to each other? Finally the goal relates to the victory condition, is there a specific goal or can the users win through a partial completion?

For the sake of this project and in order to comply with the second step of simplicity, the board will be grid based as with most board-games. The pieces will be limited to the user's piece, which will be able to move in four directions. The user piece should also have health which can be affected by the traps and the supporting rooms, which goes with the concept of a rouge-like game. Finally the goal for the user in order to fulfill the victory condition is to reach the exit of the maze.

### Constructing the puzzle

Step five revolves around challenging the user in a way that allows the user to find a solution and get the rush from finally figuring out how to get past a tricky challenge. Scott Kim mentions a few possible tools that can be used in order to challenge the users. One of these methods is by making the users make choices during the puzzle, some of which will inevitably lead to dead ends. Relating back to Chapter 3.3.2 *Magic Circle*, one of the important things mentioned by Scott Kim in this step is that the solution to the puzzle must not require obscure solutions using methods that the player is not aware that they are capable of.

### Test

Step six is a necessary step in order to see whether or not the puzzle in its constructed form works as intended, or if it is too difficult or easy to figure out. It also serves as a method of finding any errors in either the implementation or in the rules defined earlier. It also lets people try out the interface, which doubles as usability testing.

### Devise a sequence

Step seven is all about the order that your puzzles will appear in. Scott Kim mentions that the most obvious sequence is a linear sequence going from easy to difficult, however as was also suggested by Chanel et.al (Chanel, Rebetez, Bétrancourt, & Pun, 2008) it is recommended using a sawtooth shape switching between easy and difficult.

### Pay attention to presentation

The eight step is all about the superficial aspect of the puzzle, adding components such as sound, graphical style, animations, user interface elements, storyline and so forth. As the narrative is part of the deconstructed engagement model from Chapter 3.2.2 *Deconstruction* suggested in this project, a form of narrative will

need to be included in the game. Furthermore the game should contain graphics that represent the setting as a means to uphold the engagement as proposed in the framework by O'Brien and Toms. The inclusion of sound would primarily be as a method to support the visuals in setting the theme and keeping the user interested in the game.

### Sum up

Using *Scott Kim's Eight Steps* to creating puzzles, allowed a specification of the content in the game to be made. It was decided that the visual theme of the game would be a dungeon or cave style theme. Also, the type of puzzle that would exist within this theme would be a maze. It was then decided that aside from the main goal of finding the exit, the game should contain rooms with different effects that either support or hinder the user's progress in the game. This would also allow readjusting the difficulty of each level to match the user's ability by changing the amount of each type of room present. It was also decided that the control scheme to be simplified should exist as an interaction with these rooms. The level in the game will be constructed as a board game, minimizing the amount of controls required to play the game. Furthermore the user should be provided with health, that the different rooms can affect, and finally that the victory condition of the game is to reach the exit of the maze. In the event of additional levels, the challenges should interchange between easy and difficult in order to keep them interesting for the user. Finally it was decided that the game should include a form of narrative, and a visual design that follows the chosen theme.

With the game design related to the puzzles complete, the project can proceed to designing the levels and the visual style that will be used in the game. Initially the level design will be covered based on the decisions made throughout this chapter.

## 4.2 Level Design

As previously mentioned, the game will be constructed using a grid as a game board. The level design will be created with keeping this structure in mind, while also allowing control over elements that can alter the difficulty of the game without affecting the rest of the design on a too large scale. By using the grid structure for the game board, it allows for using a modular approach to constructing the individual parts of each level. Similar to how a chessboard is divided into white and black pieces, each point in the grid will be created as a room in the maze. Each room will then contain one of the functionalities mentioned in Chapter 4.1.1 *Gameplay* and by traversing these rooms the user will eventually arrive at the goal. The room functionality will be split as such:

### Trap Room

The trap room will drain the user of half of their current health. This allows the user to encounter multiple traps without dying, which should prevent initial frustration caused by dying with a single wrong move.

### Clue Room

The clue room should present the user with the direction in which the exit is placed compared to the users current position. For instance the clue room could assist the user by informing him that the exit was somewhere to the right of his current location.

### Distance Room

A more subtle hint than that found in the clue room. The user will be informed of the distance between him and the exit, but will not be informed as to which direction the exit is located towards.

### Exit Room

The goal of the maze, reaching this room will fulfill the victory conditions required for completing the puzzle.

### Healing Room

Inspired by the role-playing elements of the rouge-like genre, the healing room will provide the user with a place to replenish any health lost from triggering traps. Furthermore it will allow them to strategically explore the maze by remembering how to return to the healing room, and then scouting the areas around it for the exit.

### Spell Room

A room in which the user can cast a spell that will either aide them or work against them. This will be the primary room for the simplification of the control-scheme. Every time the user enters a Spell Room, he will be prompted to recite a spell. Two versions of this room will be created for the test. In the first version the user should manually input the magic spell based on a predefined template, and in the second version this part of the game will have been either automated or simplified to involving a button click. The primary goal of reaching the exit will not be impacted as the controls related to the movement will remain untouched.

Aside from the difficulty induced by the rooms, the grid structure also allows the challenge of the puzzle to be affected by increasing or decreasing the size of the grid. This allows a simple method of implementing the saw-tooth approach to modulating the difficulty that was recommended by both Scott Kim and Chanel et.al. (Chanel, Rebetez, Bétrancourt, & Pun, 2008). For instance a series of levels could consist of a 10x10 grid, followed by a 12x12 grid and the by a 11x11 grid. With this the difficulty has already slightly moved from easy to hard and back to a medium difficulty level.

## 4.3 Visual Design

The visual design of the game should support the genre and the gameplay of the game, as well as providing engaging visuals to aid the investigation of the final problem formulation. However in order to create an association between the hybrid-genre and the visuals as well including the concepts from the engagement components, it became necessary to create a narrative that could support the game world. In relation to the dungeon filled with traps and the rouge-like approach, as well as the concept of solving puzzles a concept was hatched. The story of the game will focus on a wizard's apprentice who, due to neglecting his studies of spells, ends up casting a spell that transports him down into a dungeon. While the story is not extensive, it is comparable to those found in rouge-like games.

With this decision, the visual elements in the game can now be designed.

### 4.3.1 The Rooms

Each room should have a distinct visual difference that allows the user to - after being acquainted once - immediately identify the room they have entered and know the effects. Identical rooms would only serve to confuse the user as they could be unaware if they had moved to a new room, and if the changes are only subtle, they might mistake one type of room for another. An ideal way to separate the rooms would be to primarily use colors do differentiate between the different rooms. In most situations - both in video games and outside - colors are used as indicators, for instance a traffic light which uses green for safe and red for danger. The same approach could be used in the Trap and Healing rooms. By populating the Trap room primarily with red colors, it would serve as a natural indicator that this room means danger. Likewise by populating the healing room with green would provide the natural indicator of a safe room where the user

can relax. Even if we take out the natural context of the color indicators, the color scheme alone would provide a clear differentiation between the different rooms.

Another way of differentiating the rooms would be by altering the sizes and shapes of the rooms. However this approach comes with a few quirks that have to be taken into account. As the grid based approach with rooms is modular, it also has the requirement that the entrances and exits of every room should match up with each other. The feedback of entering a door located at one point and exiting a completely different place in the new area could confuse the user or make it appear as though some content is missing.

As such the primary approach to differentiate the rooms will be through the use of colors, with the added option to alter the room sizes for added effect – as long as the rooms conform to having identical entryways and exits.

## 4.4 User Interface

The following Chapters 4.4 and 4.4.1 on the topic of user interfaces and innovation has been written using inspiration from a previously worked on report (Jensen, Etzerodt, Christensen, & Jørgensen, 2009).

*“The user interface brings the game to the player, taking the game from inside the computer and making it visible, audible and playable”*

(Adams & Rollings, on Game Design, 2003, s. 224)

The above quote relates to a game being visible, audible and playable. The gameplay of the project has been described in Chapter 4.1.1 *Gameplay* and the visual and audible design of this project has been described in Chapter 4.3 *Visual Design*. The user interface one of the most important aspects to the project, as it is through the user interaction that the change in engagement stated in the final problem formulation should occur.

### 4.4.1 To innovate a user interface or not

When creating a user interface, a single question should be posed prior to beginning the work: ***“Should the user interface innovate or not?”***

In areas such as game design, visual design etc., innovation has always been the forefront in video game production. It is the reason why games such as Dead Space (EA Redwood Shores, 2008) and Tom Clancy’s EndWar (Ubisoft Shanghai, 2008) have become popular games, both prior to and following their releases. These games suggest that innovation is a goal to work towards. However in the case of designing user interfaces, this is not always the case.

*“Although you will want your player to be impressed by the originality of your gameplay, the player will almost certainly prefer a familiar UI.”*

(Adams & Rollings, on Game Design, 2003, s. 227)

Most games are designed to be easy to pick up and play, a trait that is also true for the game being created during this project. In the game, it is the goal to create engagement primarily from the challenges created by the design as well as the rules set for the challenges. Creating a new control scheme will require the users to get acquainted with the new controls, and risks the possibility of making it difficult for the users to access vital information or in the worst case, break the sensual engagement (O’Brien & Toms, 2008). The user

An investigation into the effects on engagement, caused by control scheme simplifications in games.

interface for the game created during this project should as its main purpose support the gameplay, and as such there will be no attempt to innovate the user interface.

#### 4.4.2 The controls

The controls in the game all constitutes meaningful play, since any input from the users side will always be relayed in both discernible and integrated ways throughout the game. Moving from room to room will provide the user with discernible feedback as they will immediately see the result of their action. The effects of the rooms however will provide both discernible and integrated feedback. Stepping into a trap room will cost the player half of his current health which is considered discernible feedback; however the result of this could mean that the user dies shortly after due to stepping into another trap. The effects of the first trap left the player injured, which made it so that the following trap could kill him which is considered as the result of integrated feedback.

The controls can be divided into two categories: Movement and Spell casting

##### Movement

As the quote in Chapter 4.4.1 *To innovate a user interface or not* suggests, users will almost always prefer a familiar interface when playing a game. As I am not attempting to create a breakthrough within interfaces, it would be prudent to attempt to accommodate the movement controls so they match the current standard for most of the present day games. As the movement in the game only will consists of moving in four different directions, the necessary controls can be summed up in a single point:

- Movement will be controlled using either the arrow keys or the wasd buttons.

While the arrow keys can be considered the most intuitive control scheme, with arrows drawn on them that indicate direction, it is common for games today to use the wsad as secondary arrow keys which allow a more natural posture when using both keyboard and mouse. Therefore it would be wise to accommodate both means of interaction.

##### Spell casting

The controls used for spell casting will depend highly on the way this element is integrated into the game. It could be as simple as selecting a spell from a drop down menu to drawing magic runes with the mouse. Also in order to create a similarity between the regular version and the simplified version of the control scheme used in the spell casting, the resulting effect should be identical in both versions.

In order to accommodate this as well as fulfilling part of the final problem formulation in relation to the simplification of a control scheme, the initial prototype of the product will use a one button spell casting system for the simplified version and a multiple button input system for regular version. In order to balance the outcome and prevent the user from always using a beneficial spell, a spell will consist of four words where each word can have a beneficial or damaging effect. By typing in four different words for each spell, the user should be at least to a certain extent unable to identify the effects of each individual word.

This means that the controls for the spell casting will be as follows:

- A spell will randomly be chosen on the simplified control scheme.
- The user will type in 4 words in a predefined structure for the regular control scheme

With the words being magic spells, it is not required that they make any type of sense – and therefore using dictionary words is not required. Any prefix can be used for the spell casting. For the sake of keeping the gameplay simple, the users should not be forced too long words for each part of the spells. Therefore each word will consist of 3 letters, making the grand total of each spell 12 letters.

#### 4.4.3 The interface

Aside from the controls that the user uses to interact with the game world it is also necessary to provide the user with some information about the game world and as a means of feedback from interactions. Taking the grid structure into consideration, one piece of the information that the user might like to know, is where in the maze they are. There are several ways of conveying this to the user, and one of the simplest ways would be to provide the user with a map of the maze. The downside to this approach is that it would eliminate the difficulty of any smaller mazes, as the user would quickly be able to navigate them and know the safe routes back from any position in the maze to a nearby healing room, without giving it a second thought.

An alternate way of providing the information to the user would be to provide them with their current coordinates and perhaps also the size of the dungeon. With an indication of where in the dungeon they are – provided by the coordinates – and an idea as to the size of the dungeon, they would be able to quickly get an idea on how to proceed into the maze. This combined with the already designed rooms that provide clues and distance to the location of the exit, should be sufficient assistance in completing the tasks.

Another aspect that the user needs to be informed of is the state of their health. The most common way of representing this little piece of information has changed greatly over the past few years. In the first person shooter Unreal Tournament 2004 (Epic Games & Digital Extremes, 2004), health was displayed as a number so the user had a clear overview of the exact amount of health remaining. However within the last few years, starting with the release of Gears of War (Epic Games, 2006) the trend switched to health slowly being removed from the interface. Instead of displaying numbers, the screen will just begin to go red as the user nears death. As this project does not require the user to have a specific amount of health, I have opted to go with approach used in the Diablo series of roleplaying-hack'n'slash type of games. The health of the user will be displayed as red liquid in a bottle, and upon taking damage the contents of the bottle will begin to empty out. However it also allows the opposite to happen, if the user receives a beneficial effect from a spell room in which the health of the user is increased by making the fluid in the bottle spill out.

The final piece of information that is required on the interface is a method of conveying the effects of each room to the user. Whether it is the clue from the clue room, the distance from the distance room or the effects of the spell room, they all need to be conveyed to the user. The most appropriate way for doing this is through text. Therefore each time the user enters a room, a line or two of text will appear that describes what has taken place in the room. This will also function as a backup in case that the color scheme approach to differentiating the rooms fails. In order to keep the text as intuitive as possible, it will be located at the top left of the screen in an attempt to mimic the act of reading pages off of the internet or documents on a computer.

#### 4.4.4 The four kinds of users

To add an interesting aspect to this, Kristine Jørgensen from the University of Bergen recently wrote an article (Jørgensen, 2011) discussing the influence of simplifying the graphical user interfaces in various genres of video games. She divides users into three archetypes each with their own agenda in regards to how an interface should look and feel; The Fictionalists, The Systemists, The Relativists. Users from the



An investigation into the effects on engagement, caused by control scheme simplifications in games.

fictionalist want the interface to be gone as much as possible. Anything that intrudes on their experience is an annoyance. As such a user interface that would cater to a fictionalist should blend in seamlessly with the game world. Systemists see's the setting of a game as being irrelevant, and that the interface needs to explanation for being as it is. However the interface communicates is fine with a systemist. Relativists are a placed somewhere between the interface hating Fictionalists and the Systemists who don't care about the interfaces. They prefer elegant solutions which lie in between, and only accept the interfaces as long as they provide useful information.

The point of this study is how to develop interfaces for these three types of users. Jørgensen explains in her article that most users will likely belong to the relativists, having accepted that interfaces are an integral part of the video game industry. Therefore one should not worry too much about breaking the users involvement in a game by adding overlays with graphics or information, as long as the information is useful lot the user.

### Sum up

In this subchapter it was decided not to innovate on the design of the interface as the game should be easy to pick up by the user, and therefore rely on an interface that is known by the users. Movement in the game will happen through the use of either the arrow keys or the WASD keys, which are the most common keys for movement used in modern games. The spell casting system is where the two control schemes mentioned in the final problem formulation will be found. The regular version will require the user to input a number of buttons to cast a spell, while the simplified will only require a single button press to perform the same action. It was also decided that the spells should consist of no more than 12 letters, in order to keep it simple. It was decided not to provide the user with a map, as they would too easily be able to find their way through the dungeon, avoiding obstacles as they went along. The user will be provided with a set of coordinates to figure out where on the board they are located. The health of the user will be provided through the illustration of a bottle filled with red liquid, which is a common method of displaying health within the roleplaying genre. Finally the user will be informed of the events occurring in each room, through text on the screen.

## 4.5 Software Design

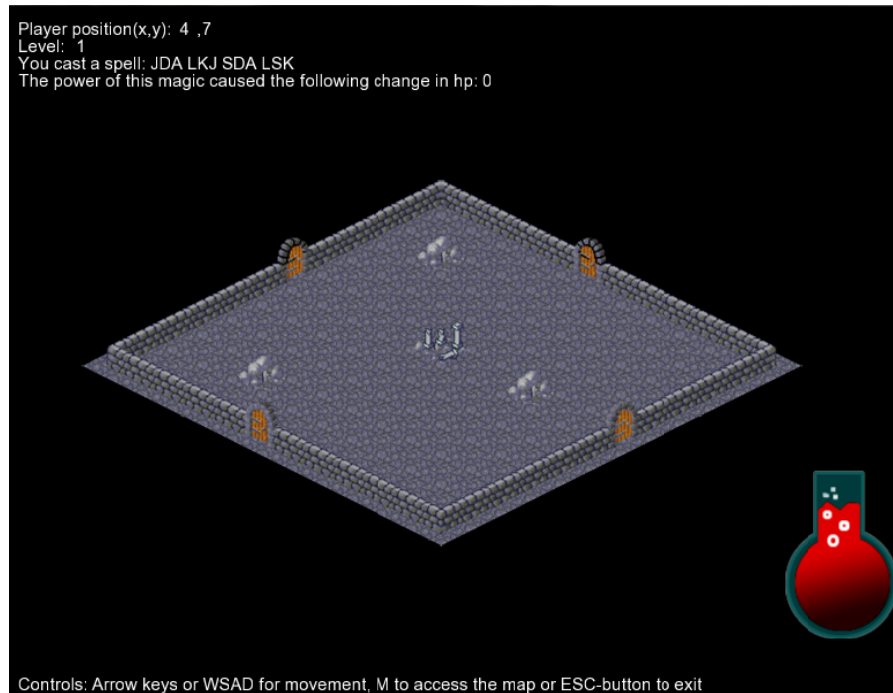
To get a testable product, it is necessary to program an application that users can test on. This chapter will look into the design of this game, in order to determine what is needed before moving into the implementation phase of the project. The overall structure of the program will be created through the use of flow charts.

### 4.5.1 Game flow

The word flow (not related to the flow theory by Csikszentmihalyi) describes how an interaction with the game will function. This chapter will describe how a play trough of the game could occur. A flowchart will also be made to detail the events in the game and what the user can do and when he can do it. When this information has been presented, it is possibly to proceed into the implementation phase.

Based on the information from Chapter 4.1.1 *Gameplay*, the rules of the game has to be clearly defined in order to clarify exactly how the game plays. Not all of *Scott Kim's Eight Steps* will be used, since steps such as constructing the puzzles or testing are described in other parts of the report. The primary step that will be used here, is the step "Define the Rules".

The game will start with the player placed in the maze at a random location, as shown in *Illustration 4.1*.



**Illustration 4.1:** The interface of the game. The user is placed at a random position on the board and has the coordinates of the position presented at the top left of the screen.

### Winning condition

Reaching the exit of the level.

### Starting options

1. The user can move in any of four directions (assuming the user does not start in a corner).

### Clicking a move button

Clicking a move button will cause the scene to change to the corresponding room. Following this the effect of the room will take place. In the case of any other room than the Spell Room, the user will have no further input.

If the room is a Spell Room, the user will be asked to type in 4 words of 3 letters each in order to cast a spell, after which the room will cause its effect.

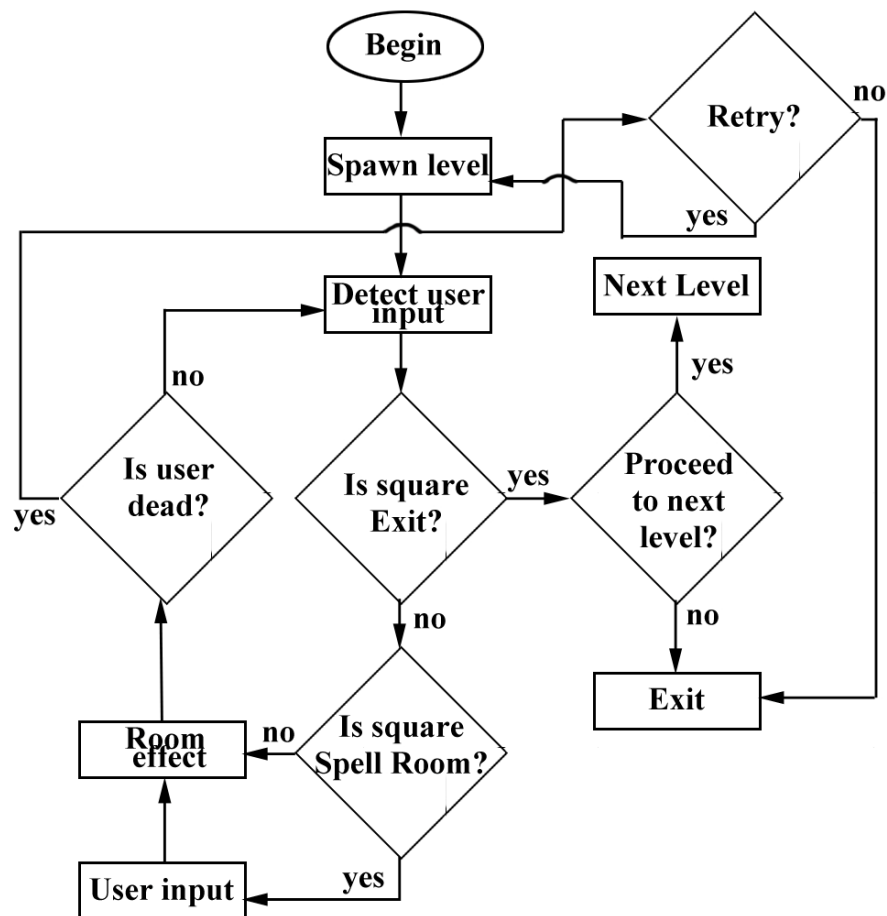
### Reaching the exit

Upon reaching the exit, the game will automatically proceed to the following level.

### Flowchart

In order to get an overview of how the program itself should be structured and how the individual parts of the program should react to the user input, a flowchart has been created.

An investigation into the effects on engagement, caused by control scheme simplifications in games.



**Illustration 4.2:** A simplified flowchart of the game. The game loop goes between the users input and the room effects until the user is either dead or has reached the exit.

### Game Classes

With the flowchart in *Illustration 4.2* as foundation, it is possible to create some initial classes to provide a more detailed overview of what has to be implemented. A main game class needs to be created that will contain the interaction between the user and the game, as well as the main loop which the game will run in. The purpose of this class is also to gather together all the other classes, and create a functioning game from the collaborative functions. A player class will need to be created, which will serve as a controller for the players position and current health. A board class will need to be created in order to create the levels and make the rooms in which the various events of the game will take place. An event class will be created that will handle all the possible events and occurrences in the game, such as re-adjusting the players health after entering a trap room or providing the player with guidance to towards the exit. Finally a class keeping track of the game state needs to be created. This will oversee conditions such as if the user is dead or has reached the exit, and allow the user to retry or continue to the next level. Based on these designs the class diagram created can be seen in *Illustration 4.3* :



Illustration 4.3: The class diagram showing the necessary classes for implementation of the game.

#### 4.6 Design conclusion

The completed design chapter holds a collection of ideas, concepts and decisions made which will help shape the product of the project into a form that will help fulfill the final problem formulation. A hybrid genre was decided upon which mixes elements from the puzzle genre with that of the roleplaying rouge-like genre. In order to help structure and design the game, help was enlisted by using “Scott Kims Eight Steps” (Adams & Rollings, On Game Design, 2003) to creating puzzles, which step by step helped shape the design for the game. In order to be able to differentiate one room from another, each room type would adapt a unique color scheme. When it came to the user interface controls, the decision was made to keep it as intuitive as possible by adhering to the

An investigation into the effects on engagement, caused by control scheme simplifications in games.

current norms of control schemes using the arrow keys or the wasd keys for movement. The interface elements in charge of relaying information to the users were discussed, and based on the decisions made earlier in the chapter, a method was chosen that would compliment level design of the game. In relation to the final problem formulation, the two control schemes will take their appearance in the spell casting rooms. The user will have to push a single button and cast a random spell for the simple version, and type in a sequence of letters to cast the magic spell in the normal version.

Finally the software design contained an example of a use case scenario and how the game would progress through the various events. A flowchart was created in order to visualize how the program should be structured and how the different segments of the code should cooperate.

With the design completed, the project will now move into the implementation phase. The implementation will be based on the ideas and decisions made in this chapter.

## 5. Implementation

This chapter will describe the process of implementing the game that will be used in the test of the project. The chapter will begin by describing the game programming, such as the functions that are required in order for the game to be played, including interface programming, game logic, level design and media handling

Afterwards the chapter will move on to describing the implementation of the visual components, following the decisions made in the design chapter.

### 5.1 . Game Programming

This chapter will cover the implementation of all programming related aspects that were needed in order to make the game design a reality. As a class diagram was created in Chapter 4.5.1 *Game flow* the implementation will follow the classes described, and comment on any additions or delimitations that occurred during the implementation of the classes. The first class to be explained is the main `Game` class, which creates instances of all the other classes and controls the overall gameplay.

#### 5.1.1 The game

As the code in its entirety is close to 1000 lines, this chapter will only provide an overview of the processes that the game goes through in a situation in which the user interacts with the game, in order to provide an example of how the program interacts with the user. The code will explain how the graphical part of the game is created through the use of images and sprites, and how the tileset approach to the graphics decided upon in the design will be handled by the program. An example of code allow interaction between the user and the program will also be presented, as this is an integral part of the game and part of the final problem formulation, without interaction there can be no control schemes. Finally an example will be given as to how sounds and music can be loaded into the game, in order to support the graphical theme. Part of the code used in the project can be viewed in detail in *10.Appendix.IV - Game.cpp*.

While the `Main` function is called from the `WizardMaze.cpp`, the actual game loop happens inside the game object, created from the `Game` class.

## Game Class

The `Game` class can be considered the hub of the entire network of classes. Other than the functionality required in the class to ensure that the game loop can run, it also creates instances of every other class present in the game. Aside from the game loop, the main purpose of the `Game` class is to handle inputs the user feeds to the game, the graphics displayed and the sounds played while the game is running.

The graphics consists of two parts: A window that serves multiple purposes, and a sprite manager that provides the graphics. Both of these are controlled by SFML. The following is an example of the code used to create a window and display a sprite inside the window.

```
1  sf::RenderWindow App(sf::VideoMode(800,600, 32) "Wizards Maze");
2  sf::Image Image;
3  if (!Image.LoadFromFile("../GFX\\RoomsA.png"))
4      Return EXIT_FAILURE;
5  sf::Sprite Sprite(Image);
6  App.Draw(Sprite);
```

The first line creates a window named `App` with the window size of 800x600 using 32bit, and is the window is marked with the label "Wizards Maze". Lines two to four creates an `sf::Image` container, which is the main input of images whether it is from image files or direct webcam feeds. In this case the image is named `Image`, and line three attempts to load an image file from a local folder. Should a problem with loading the image occur, the program will terminate with an error message detailing the problem. Line five creates a `sf::Sprite` container, which is what is used to display the graphics in SFML. Rather than display images directly, SFML copies them into `Sprite` containers for handling and displaying. In this case, the previously loaded image is inputted into the newly created sprite. Finally line six draws the sprite. In order to provide a more detailed overview of how SFML handles images and to avoid rewriting the section once more, Chapter 5.1.3 *Media handlers* is based on a previous project (Jensen, Etzerodt, Christensen, & Jørgensen, 2009) and will explain the process in depth.

In order to avoid having to load multiple files or create huge folders of art contents, all the rooms present in the game has been gathered in a single file called "RoomsA.png". Whenever a new room needs to be drawn, the following line is called:

```
1  Sprite.SetSubRect(sf::IntRect(0,0,800,600));
```

This line calls a function in the sprite that allows the currently shown region of the image to be changed. The four digits determine the top left and bottom right section of the image file that should be shown. By using this method, it becomes possible to use a single large image file to contain graphics that can be used in multiple places or swapped depending on the required situation, without loading additional resources.

The second part of the `Game` class is the event handler, which is related directly to the previously mentioned window. This piece of code handles the inputs from the user and passes them on to the event handler in the game. Whenever a user presses a key that is recognized while the window is the active element – any button presses while the window is inactive will be ignored –, the input will be recognized and passed on to the event handler as such:



```
1  If ((Event.Type == sf::Event::KeyPressed) && (Event.Key.Code == sf::Key::Left)){  
2      SetInput(-1,0);
```

The first line is another SFML function, that checks if a button has been pressed and if the button pressed is of a specific kind, in this case is the left arrow key. If the user input is recognized it proceeds to alter the SetInput functions. The SetInput function is part of the `Game` class which passes the input, which -1 and 0 in this case, to the event handler in the form of an x and a y coordinate. This is later used to calculate valid moves, moving the player and determining which room the player has entered.

The third part of the `Game` class functions used with SFML is the audio handler:

```
1  sf::Music Music1  
2      if(!Music1.OpenFromFile("../Music/bg2.ogg"))  
3      Return EXIT_FAILURE;  
4      Music1.SetVolume(25);  
5      Music1.Play();
```

The main functionality of the audio handler is quite similar to that of the graphics handler. The method in which music is loaded is identical to the image loading process, except that it uses an `sf::Music` container instead of an `sf::Image` container. SFML does not allow any music file formats to be used such as the popular commercial MP3 format; however support for open formats such as the .Ogg format is implemented. Lines four and five are specific to audio and music containers when using SFML, and provide control over aspects such as playing, pausing, stopping or altering the volume, which in this case is lowered to 25% of the original volume.

These three functions from SFML are in charge of handling what the user sees, hears and how they interact with the game. However the program is only at its beginning and it is not until the user has inputted a move that the further events start to take place in the event handler.

### 5.1.2 The event handler

In order to provide an idea of how the game logic progresses, and how it runs similar to that shown in *Illustration 4.2* this subchapter will describe the processes as they happen after the user inputs a command.

#### Board Class

Once the user has inputted a command that is recognized by the input handler, the program continues to run a barrier check. The barrier check is part of the `Board` class, and uses the inputted x and y data to calculate where the user is attempting to move based on his current position. If the move is invalid, the barrier check will return false and the user will be provided with a message on the interface telling him that the desired move cannot be performed. However if the move is valid, the code moves on to the second part, checking the room type. Depending on the room type different approaches are needed, mainly regarding the Spell room.

#### Event Class

If the user enters a Spell Room, a prompt appears asking the user to input twelve letters in order to cast a spell. Once the letters are inputted, they are processed in the `ManualSpell` function which is part of the `Event` class. Here the results of the spell are calculated and returned to the user. In the case of the simplified version of the function - dubbed `CastSpell` -, the program randomly selects four spell words and calculates the result. In this manner, the simplified version treats the Spell room as any other room in the game. Following the event handler, the graphics are changed to match the room that the user has entered.

## Game State

Originally intended as a class for itself, the game state elements were integrated into the `Game` class. This was mainly due to allow easy interaction with the other classes as well as the input handler. Whether the user is dead, or has reached the exit – the procedure that follows is identical. The game enters a loop in which the user is informed that they have either died or succeeded. They are then provided with the choice of retrying or advancing to the next level, or quitting the game. Due to the low implementation time, additional levels were not added. However should the user decide to retry or proceed to the next level – the program will instead reset the player and spawn them at a random position on the game board.

### 5.1.3 Media handlers

The SFML media library can load a wide range of image file formats, which can then be displayed on the screen. When an image file is loaded into SFML, the raw data of the image is loaded into the RAM of the computer. In order to display the image, it is necessary to create a sprite and point it to the image data. Hence the sprite does not own an image; it only functions as a pointer to where the image data is located. This is shown in *Illustration 5.1* where the file `Image.jpg` is loaded into the RAM. The two sprites then point to the image data in the RAM and display the picture from the image file on the screen. This way, the `Image.jpg` file is only loaded once into the memory despite being used in several places.

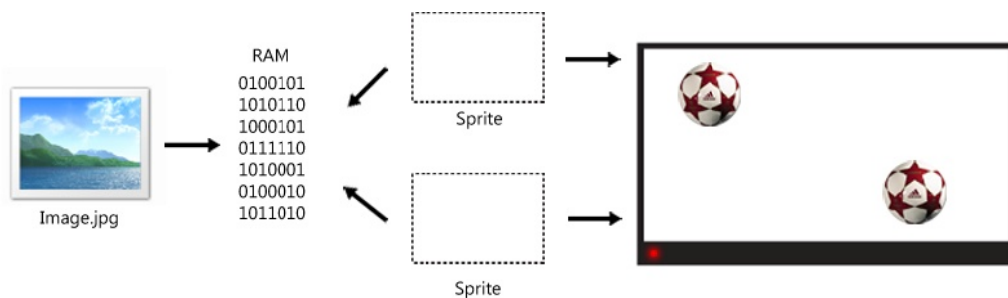


Illustration 5.1: Explanation of how the Image and Sprite handles data

## 5.2 Visual Implementation

This part of the implementation will cover how the design decisions were realized for use in the game. It will provide an insight as to how the different parts of the graphical interface in the game were made.

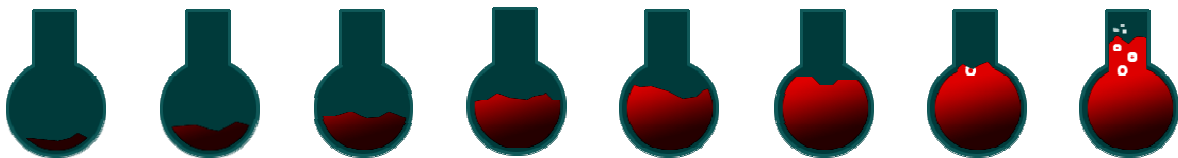


Illustration 5.2: The health bar present in the game

In Chapter 4.4.3 *The interface* it was decided that the graphics for the health system should be a health potion, as seen in so many other games. The graphics for the health potion was created in Adobe Photoshop, using layers to create the bottle effect. Once the initial health bottle was finished - the full one - the rest were made by duplicating the bottle and slowly cutting away of the layer consisting of the red interior. The reason the bottles are all lined up in *Illustration 5.2* is so that they can be used as a tile set in the game, similar to that of the rooms mentioned in Chapter 5.1.1 *The game*. A tile set is a series of smaller images that are joined together in a single large image. The purpose of using a tile set is so that the computer only has to

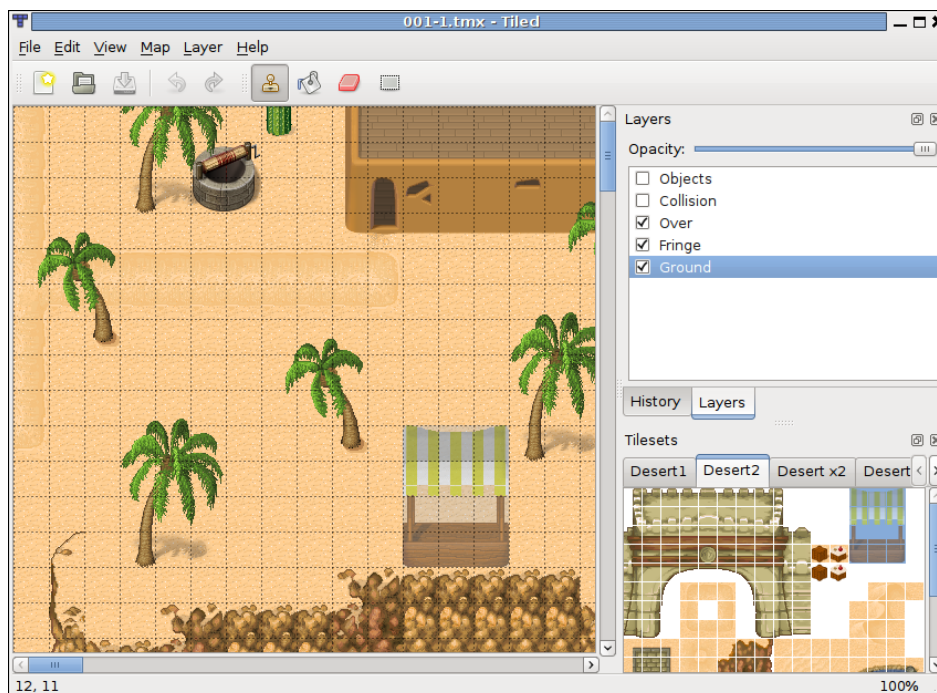
An investigation into the effects on engagement, caused by control scheme simplifications in games.

import a single file and through this file, can provide textures for multiple objects, or it can contain multiple textures used for a single object.

The bottles shown in *Illustration 5.2* are made in small squares that are exactly 150 pixels high and wide. This is done so that when the program needs to transpose the displayed area of the texture, it can simply add 150 pixels to the shown region, to reach the following bottle. This is the same method the game uses for rendering the different rooms of the game. However while the bottles were created in Photoshop, the rooms were created in a slightly different manner using a tool called Tiled Map Editor (Thorbjørn, 2004).

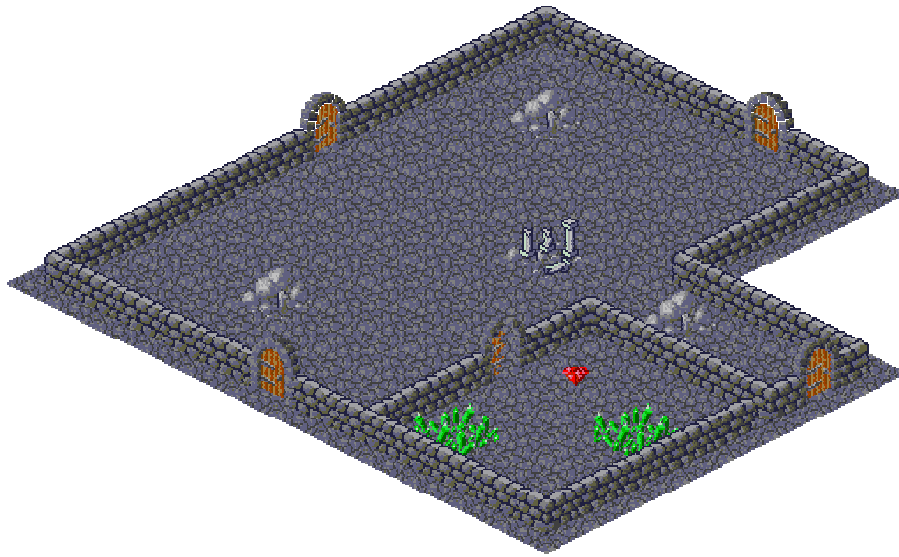
### 5.2.1 Tiled Map Editor

The Tiled Map Editor is a piece of software that allows working with tile sets as construction kits in either isometric or orthographic perspectives.



**Illustration 5.3:** The main screen of the Tiled Map Editor. Notice the tile set in the bottom right part of the screenshot.

Once a tile set has been loaded into Tiled, the user defines how large each individual tile is for the specific tile set and the program automatically makes them into selectable tiles. Once this step is completed, all that's left is to separate each type of tile into its own layer. The ground layer is usually the first to be created, followed by an object layer consisting of objects such as tree's or houses that need to be placed on top of a ground tile. Once a map such as the one in *Illustration 5.3* has been edited to the desired result, it can be exported in either Tiled Map Editors own .tmx format or the map can be saved as an image. The .tmx format is in its essence a modified xml file, with minor changes to support the various information provided with each room or level designed in tiled. The file contains everything from the size of the level, and to the path of the tile set image file used. The second way exports the map into a variation of image formats such as .png or .bmp.



**Illustration 5.4:** An example of one of the rooms created in Tiled for the product, using the Isometric point of view.

For this project the latter method of exporting the rooms for the game. While the .tmx would be far superior to support in the game, it does require the creation of a custom coded .xml parser that can interpret the data and allow a C++ program to utilize it. As I had no knowledge in working with parsers, I instead decided to use the tile set approach for the rooms as well. By ensuring that each room scene was exactly 800x600, the entire range of rooms could be fitted into a single image file, which just as with the health bottle jumps from display area to display area in the image, whenever the user moves to a different room.

The graphical aspect in the project as can be seen from *Illustration 5.4*, was an isometric perspective. The reasoning behind this was an attempt to create a sense of depth, using the pseudo 3d effect that the graphics provide. As one of the important aspects to keeping users engaged is interesting graphics, I decided that using a less traditional approach might entice users to continue exploring the maze. The tile sets for the rooms such as the one shown in *Illustration 5.4*, were acquired from a website called Reiner's Tilesets (Prokein, 2011), which is a source of free to use tile sets for creating games

### 5.3 2<sup>nd</sup> Implementation iteration

Following the original implementation, the product was tested on a few users in order to find any serious usability issues or other aspects that might have been overlooked during the design and implementation phases. These tests were done as a one on one conversation in which the users while they were playing expressed their feelings towards the different aspects of the game. The test group consisted of a total five persons, all belonging to the target group defined in Chapter 2.4 *Target group*, and none of them affiliated with the University campus. The feedback acquired revealed a few problematic issues with the product, and this chapter will detail the issues encountered and explain how they were solved.

The visual aspect of the game was intended to create a semi-3d appearance that would intrigue the player, and keep them interested as explained in Chapter 3.2 *Engagement*. The users all expressed that the graphics were interesting and fit the theme rather well; however one user in particular had a difficult time figuring out navigating the game in relation to the graphics. As shown in *Illustration 5.4*, the exits of the rooms follow the isometric view. However when using controls such as the left and right arrow keys, which door relates to what key? While this was only experienced by a single user, it reveals a possibility that other users might

An investigation into the effects on engagement, caused by control scheme simplifications in games.

also encounter this scenario. As such a second set of graphics which can be seen in *Illustration 5.5* were created with the orthogonal point of view, which can be swapped by the user incase the graphics prove to be confusing.



**Illustration 5.5: The Healing Room from the game, shown from an orthogonal point of view.**

Another issue that some of the players were experiencing was a general problem with finding out where they were compared to the board. Despite the presence of both coordinates to show the users current location, as well rooms that provide help in the form of distance and direction, it was still proving to be too difficult to navigate the maze. After digging some into the problem with the test users, it seemed the problem was visualizing the meaning of the coordinates. In order to combat this issue, a map function was implemented into the `Game` class which displays a map over the currently displayed room as seen in *Illustration 5.6*.



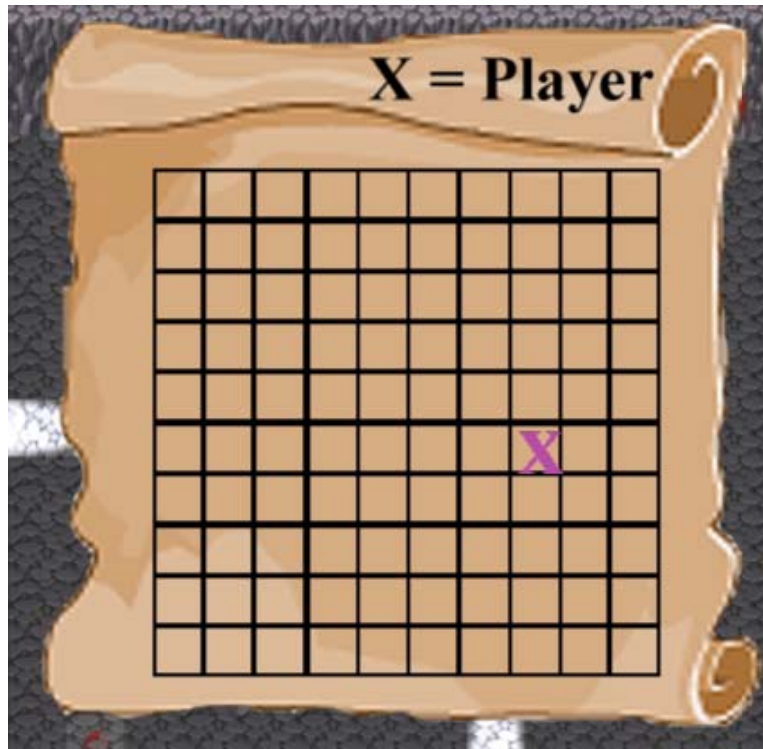


Illustration 5.6: The map overlaying a room displayed behind it. X marks the player's position.

The map contains a visualization of the 10x10 grid, and a representation of where the user currently is by marking the position with an X. While not revealing the contents of any of the rooms, by using the map it becomes possible to visualize where the user is located and recognize which regions have already been explored. With each square on the map being of identical size, the X is being moved by simply multiplying the user's position with the size of the squares.

A third issue also arose related to the spell casting system. The first time a user inputs a spell, they are oblivious as to the results. However once they come across a spell that they can see is helpful. They will stick to that spell for the rest of the game, which was thought to be avoidable with the four words consisting of three letters each. What was not considered was that with the display of the used magic spell as well as the outcome, the users would just copy the last spell. There were several possible solutions found that could solve this issue.

The first solution would be to remove either the entire event message when casting a spell, or parts thereof. The problematic issue with this though is that it goes against the 7<sup>th</sup> point of *Flow*, providing the user with direct and immediate feedback. Also the Spell room would likely become an annoyance as the player would have no idea what the result of entering the room would be or what their interaction caused. Removing part of the event message would be a plausible solution. Changing the part where it displays the spell words, and instead replace them with a message informing that a spell was cast, along with the results of said spell – would prevent users who had just randomly pressed buttons to repeat their spells. However if the user has typed in a phrase or a series of recognizable words, it is likely that the spell would be repeatable.

Another possible solution would be to lower the amount of Spell rooms in the game and replace them with other types of rooms, thereby lowering the impact a repetition of spells would have on the game. However this would also serve to counter the purpose of the project, as the Spell rooms is the room designed with the



An investigation into the effects on engagement, caused by control scheme simplifications in games.

user input simplification in mind. As such this solution cannot be considered viable, unless no better solution can be found.

A third possible solution could be to alter the control input such that the user only has to press a single button to cast a spell, after which the spell will be selected randomly. However it is possible that this will make the difference between the simplified version and the regular version almost indistinguishable.

A fourth and final solution would be to make the program recognize already input spells and request a new one. While this required additional implementation time, it was the most suitable solution to solving the spell issue without taking drastic measures that would somehow interfere with the other aspects of the product. It was implemented such that when the user has cast a spell, the inputted spell is stored and compared with any future attempts at spell casting. If a spell is recognized, the user is prompted to input a new spell as the previous one has already been used once. With this solution, the user will have no other choice than to experiment with the different combinations of spells.

## 5.4 Implementation Conclusion

With the implementation phase completed, the project now has a usable product which will allow various tests to be conducted, in order to determine if the product holds true to the final problem formulation.

The visual implementation was created using free tile sets from the website “Reiner’s Tilesets” (Prokein, 2011), in order to create the appearance of the different types of rooms. Afterwards they were exported for use in Photoshop, where all the rooms were combined into a single large tile set. The interface elements were created in Photoshop, and consisted mainly of creating a health potion to display the user health level.

The programming was done using C++ and the SFML libraries as decided in Chapter 3.4.3 *Sum up*, and followed the structure of the class diagram presented in Chapter 4.5.1 *Game flow*. A minor change to the original class diagram was made when the `GameState` class was incorporated into the `Game` class for simplicity’s sake.

Following the initial implementation, a small user test was conducted in which a few key issues with the prototype product were discovered. The original isometric approach intended to entice the users, turned out to make the navigation of the maze slightly more difficult. The user in question had a hard time figuring out which directional key belonged to which exit on the graphical representation of the rooms. As a solution to this, an alternate set of graphics were created, using an orthogonal point of view. The users also had trouble navigating the maze, as they often got lost and could not quite relate the coordinates provided to a representation of the position. In order to assist with this issue, a map function was implemented in which the users could see their position on a map, along with all the rooms. The rooms however were shown as blank tiles as to not reveal their contents. This allowed the users to get a visual representation of the coordinate data, and help them with recognizing previously visited areas on the map. Finally an issue arose when the users found a beneficial spell, in that they would just repeat the usage of that one spell at every Spell room. Several possible solutions were discussed, and in the end a solution was chosen in which the user can only use a specific spell once.

With the initial and the second implementation iteration completed, the project can now proceed to the testing phase.

## 6. Test

The purpose of the testing is to provide an answer to the final problem formulation, as stated in Chapter 2.6 *Final problem formulation*:

**How can the effects on engagement be determined, in which the control scheme of a 2D puzzle game is compared to that of its simplified counterpart?**

Through the design and implementation phase, an application has been developed that can be tested upon. The test will contain multiple questions in order to obtain a conclusive answer to the final problem formulation. The test will be divided into three areas of interest, based on the theories on engagement discussed in Chapter 3.2 *Engagement*.

### 6.1 Usability testing

An important part of the testing, is the usability testing. This part is tasked with discovering any bugs or flaws in the product that might interfere with or break the users' engagement. This could be related to the interaction method with the product, or with other elements of the game that might seem illogical or prove to be lacking content, or unbalanced in terms of difficulty.

The questions for the usability section of the questionnaire will be divided into smaller components with specific topics, allowing the users to rate each topic on a modified Likert scale from one to seven. The idea for the modified scale was borrowed from Charlton and Danforth (Charlton & Danforth, 2007), in which they use the modified scale in order to achieve a higher degree of versatility on their test responses. One will given to areas that provide the users with major problems and seven will be considered as a perfect situation with no problems appearing. The concept of the game and the project will be explained to the users prior to the testing, such that they have an idea of what they are doing. The questions for the usability questionnaire will be as follows:

- On a scale from 1-7, how would you rate controlling the game?
- On a scale from 1-7, how would you rate the information on e.g. health changes or room events, provided during the game?
- On a scale from 1-7, how would you rate the usefulness of the map?
- On a scale from 1-7, how would you rate the inputting of spells? (normal)
- On a scale from 1-7, how would you rate the inputting of spells? (simple)

The first question will provide an insight to how well the general user input is functioning. Should the first question rate horribly, it is possible that the users' could not figure out how to interact with the game at all. The second question will detail how useful the users' found the data provided with each interaction. This will reveal if it is possible that some of the information is unnecessary, or that more information is required. The third question will reveal if the newly implemented map function has served its intended purpose, or if it needs to be adjusted further. Finally the fourth and fifth questions will reveal if an alternate method of inputting spells has to be found, and also provide an insight towards the complexity of the control scheme allowed in the game.

### 6.2 Engagement testing

In order to have an indication towards whether or not the user has even been engaged at all, as well as data that can be compared across the two different versions, it is necessary to include engagement testing in the

An investigation into the effects on engagement, caused by control scheme simplifications in games.

testing phase. Aside from being the primal factor for fulfilling the final problem formulation for this project, it is also needed to determine if the game itself is able to achieve and sustain user engagement.

The questions for the engagement section of the questionnaire will be divided into smaller components with specific topics, similar to the usability testing. The questions for the engagement testing will be based on the deconstructed engagement term from Chapter 3.2 *Engagement*. As time prevents a test in which aspects such as the social aspect, awareness and perceived time can be conducted, an alternative approach has to be used.

### Sensual engagement

One of the primary focus points in O'Brien and Tom's framework are the visuals. They both serve as an element that can induce engagement and sustain it. As such it would be prudent to include questions regarding the visual aspect of the product in the questionnaire. Following the modified Likert scale, the questions will be as follows:

- On a scale from 1-7, how would you rate the graphics used in the game?
- On a scale from 1-7, how useful were the graphics for the game experience?

The primary question will reveal whether or not the graphics needs to be improved in order to appeal to the users, and the second question will provide an insight as to how useful they were in aiding the player in their play through.

### Emotional engagement

Another of the focus points in O'Brien and Tom's framework, which is testable without being physically present and observing the test participant. If the user feels that they're having fun playing the game, it is a sign according to the framework, that they are feeling engaged.

- On a scale from 1-7, how would you rate the game as being fun?

### Challenge

The testing of challenge is based mainly on the theory of *Flow* from Chapter 3.2.3 *Flow*. If the user is presented with too difficult or too easy challenges, they are likely to be disengaged from the game. As such, any information received related to challenge is usable in the future iterations of the development cycle for the game. The questions are as follows:

- On a scale from 1-7, how would you rate the overall difficulty of the game?
- On a scale from 1-7, how would you rate the difficulty of finding the exit?
- On a scale from 1-7, how would you rate the desire to continue playing in another level?

The first question will provide an insight to the mix of rooms that have been used. If the game proves to be too difficult overall, it is likely that the amount of each type of room needs to be adjusted. The second question is directed more at the challenge of finding the exit, and will provide an insight as to if perhaps the map size is too large or too small. The third question is based on the engagement stages described in Chapter 3.2.4 *Engagement stages*. Assuming that the player has been properly engaged, the post-engagement should provide them either with a feeling of frustration that they could not complete the tasks, or a desire to continue on towards another level.

### Version preference

While all these information will provide an insight to how engaged the users' have been, it is still necessary to compare the two different versions. As such a question should be asked as to which of the two versions the user preferred, which will allow the engagement values of the two different approaches to be put up against each other. The question will be as follows:

- Which of the two versions was your preferred version?
- Which of the two versions did you try first?

Aside from these listed questions – both in regards to engagement, but also usability testing -, each question will be fitted with an optional in-depth question which will allow the users' to explain why they answered what they did. This should provide an additional insight into every aspect of the product and the test results.

## 6.3 Test Conduction

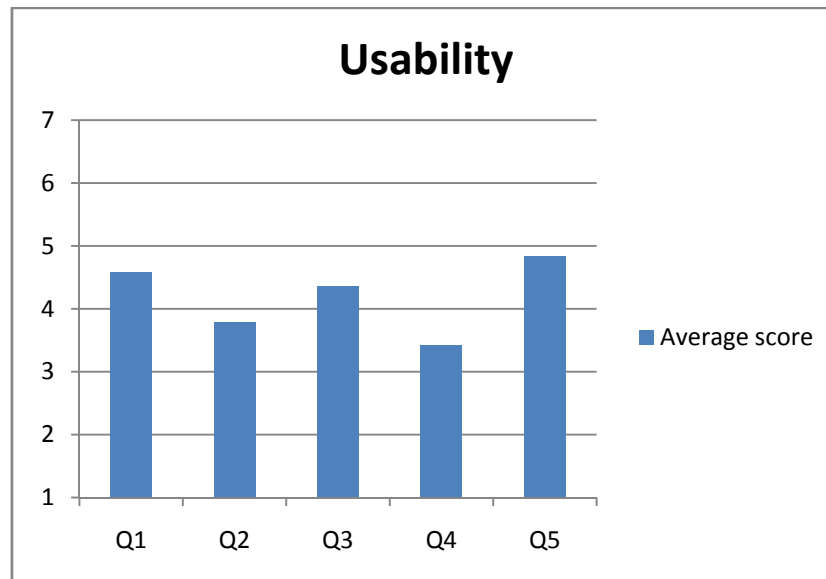
As mentioned in Chapter 6.2 *Engagement testing*, booking a test location within the time scope of the project is not an option. As such an alternative source of test participants will be necessary. Based on the type of the product and the target group defined in Chapter 2.4 *Target group*, a few possible places present themselves. One method is spreading the questionnaires through word of mouth on a social networking site such as facebook. However previous experiences with this have proven to only provide very few responses. As such an additional source of test participants should be used, in order to ensure a sufficient amount of users participate in the test. As such, I will attempt to gain test participants through the largest Danish game oriented website Daily Rush (Daily Rush, 1998). The benefit of this approach with using an online questionnaire and allowing users to download the product from their homes is a high income of test results. However there is also a downside in that some types of results and data cannot be obtained without being physically present to observe during the test.

## 6.4 Test Results

The final test was conducted on 19 persons in the course of two days, with 13 male and 6 female participants. The majority of the participants were within the age range chosen in Chapter 2.4 *Target group*. There were no-one below the age of 18, but there were two participants over the age of 49. The following sub-chapter will describe the results on usability and engagement testing, beginning with the former.

### 6.4.1 Usability results

The usability tests were conducted in order to find any flaws that might cause an influence on the rest of the test, and to see if the changes to usability in the second iteration from Chapter 5.3 *2nd Implementation iteration* has had any effect. As stated by O'Brien and Toms as well as Cairn and Brown, the ability to interact with the game is a crucial part of achieving engagement. The questions from Chapter 6.1 *Usability testing* were answered during the test and the results can be found in *Illustration 6.1*.



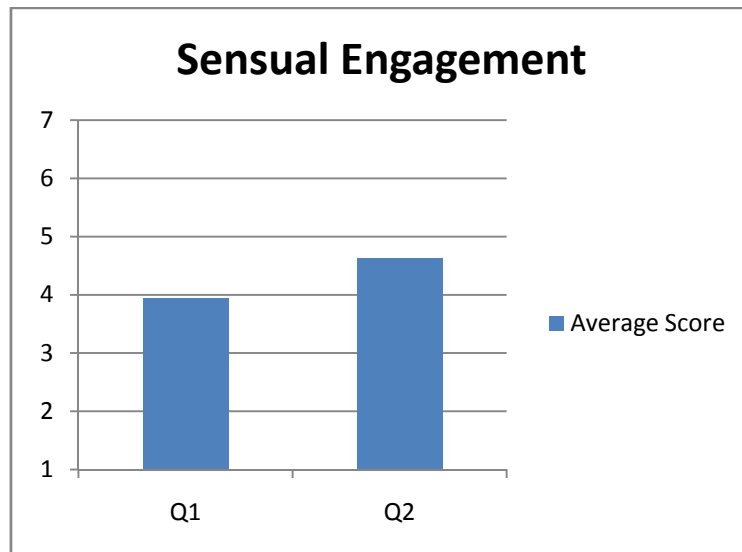
**Illustration 6.1:** Graph detailing the average scores of each of the usability questions. 4 is considered as neither good or bad score, and preferably every aspect should have been above 4 and preferably even above 5.

The test was created using a modified Likerts scale, in which the lowest possible score was one and the highest possible was seven. *Illustration 6.1* shows that the results of the usability provided a varied outcome. Q1 was the first question from Chapter 6.1 *Usability testing* and was related to the controls of the game, the interaction with the system through user inputs. This area has scored slightly higher than average which could indicate that it does not benefit the game particularly in any way; however it should not be considered as an element that breaks the interaction. This tendency follows through with all of the questions scoring just around an average. Nothing stands out as being able to be considered as game breaking, but at the same time there is no aspect of it that works well either. The aspect of the usability that is the closest to reaching a good score is Q5, which is related to the simplistic system of inputting spells. An interesting observation that can be gained from *Illustration 6.1* is the difference in rating between the normal spell casting and the simplified control schemes shown in Q4 and Q5. While the simplified – Q5 – is close to reaching a slightly good rating with the score of 5, the normal control scheme – Q4 – is close to reaching a slightly bad rating with the score of 3. This could indicate a user preference towards the simplified control scheme.

In the qualitative part of the usability questionnaire, users voiced their opinions on how some of these aspects could be improved such as by adding additional features, or readjusting some of the already existing features.

#### 6.4.2 Engagement results

Engagement testing had two purposes for this project. The first was to ascertain that engagement was present in the product, and the second as according to the final problem statement to be able to measure the change in engagement caused by the change in complexity of the control scheme. When the engagement was divided into aspects to test upon, it was divided into the topics of Sensual Engagement, Emotional Engagement and Challenge.



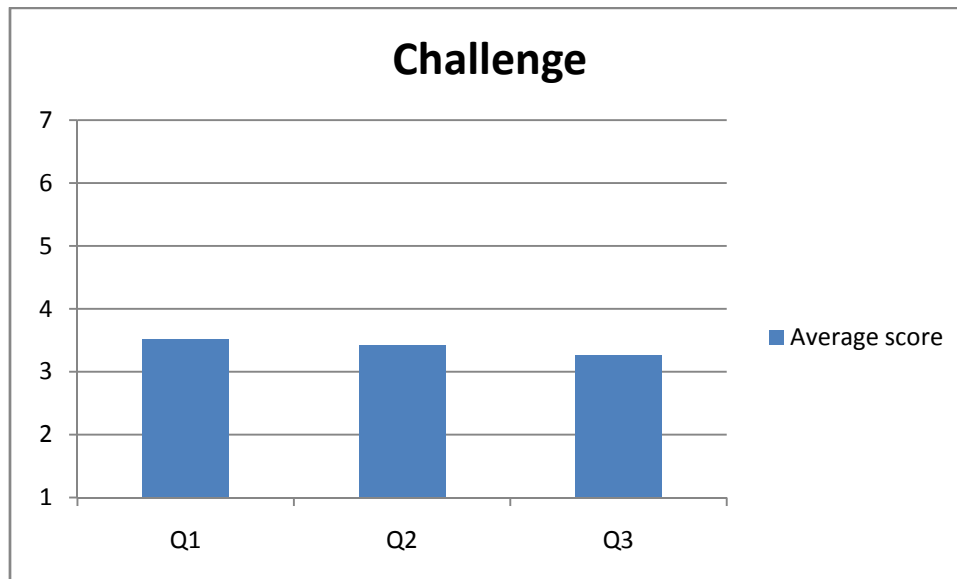
**Illustration 6.2:** Graph detailing the average scores of the Sensual Engagement part of the test, with 4 being an average score.

The result of the sensual engagement part of the test was very similar to the usability results. Once more it appears as though there is no major divergence from the average score. Q1 would seem to suggest that the graphics were usable; however they did not suffice as a method of catching the users' attention and keeping them engaged in the game. Q2 which is related to the information shown in the images, depicts that some information was successfully conveyed – however not enough for the users to consider it usable.

From the qualitative questionnaire on the topic of the visuals, it becomes clear that the visuals were a debated topic. While some users commented on it being great and useful for displaying and defining the type of room the user has entered, others comment on it as being ugly and slightly confusing. Another user mentioned that he was capable of differentiating between the types of rooms, but that the graphics did nothing for the game experience.

The emotional engagement section only contained a single question, literally asking the user how much they would rate the experience that they had just been exposed to as fun. The response from the users ended out in an average score of 2.84, which in the terms of the modified Likert scale means that the game was considered as slightly boring. The emotional engagement should not be considered on its own as an element, but rather as the result of the combined aspects. The gameplay, usability and sensual engagement and challenge all have an influence on this result. It does however indicate that a redesign is required in order to enhance the engagement in the game.





**Illustration 6.3:** Graph detailing the results from the Challenge part of the test, with 4 detailing a mediocre difficulty.

The results of the challenge part of the test can be seen in *Illustration 6.3*. All the results from the challenge questions point towards the game as being slightly difficult. Q1 is related to the overall difficulty of the game, and is based on how often the users died or how hard it was for them to traverse the labyrinth. Looking into the qualitative data once more, a few reasons that might explain the low score can be found. Users mentioned that it was too easy to die, with requests of increasing the starting health as well as lowering the amounts of trap rooms. There was also a comment that the game was too luck based, as there were no way of figuring out the contents of a room prior to moving into it.

Q2 was focused on the difficulty of finding the exit. This scored slightly lower than Q1, indicating that it should be considered as being a difficult task. User comments revealed that the reason for this low score was connected somewhat to Q1, as the trap rooms were a major hindrance in the task of reaching the exit. One user commented that he found the difficult search for the exit as being part of the charm of the game, while another was happy with the clues provided from the clue and distance rooms. A few suggestions on how to ease up on the difficulty were also made, and included adding hints about the exits whereabouts to the map or increasing the amount of rooms providing hints.

Q3 shows the desire of the users to keep playing, which was considered as an integral part of the engagement stages mentioned in Chapter 3.2.4 *Engagement stages*. Looking at the graph in *Illustration 6.3*, the result would indicate that the users did not feel compelled to continue playing after the course of the test. The reasons listed by the users' comments provide an insight as to why they would not want to continue playing:

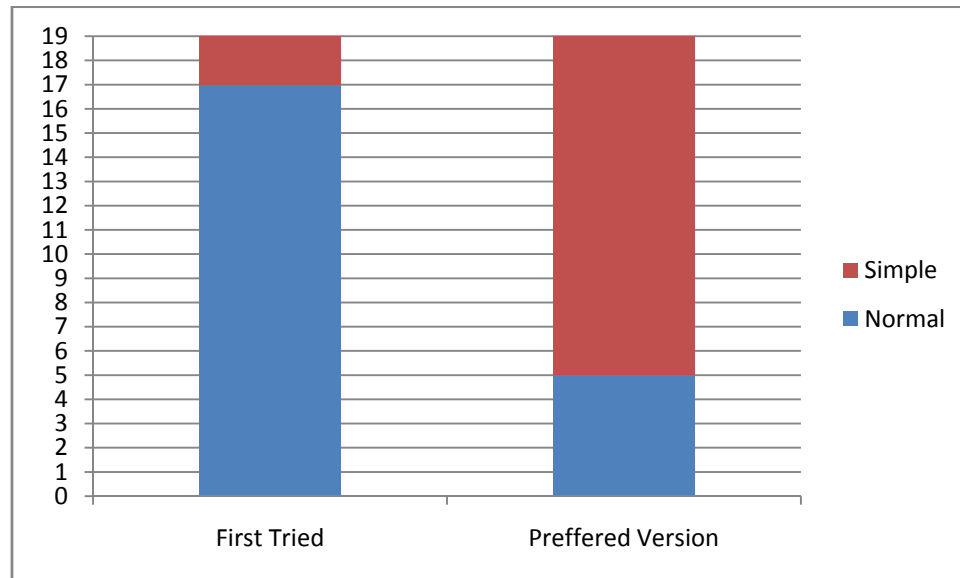
*"It's not fun." "No motivation." "The game type is too repetitive"*  
 (Translated from qualitative data; Test users 1, 3 and 8 respectively)

However some of the other users go against this indication:

*"I would not mind continuing. The game reminds me of a kind of Minesweeper, 1 wrong move and it's the death of your avatar. Big potential as a timewaster."*  
 (Translated from qualitative data; Test user 6)

Which would indicate that there is some users among the test group that go against the majority.

The last part of the questionnaire was intended to discover the users preferred version, as well as seeing whether or not the initial version tried had any effect on the responses.



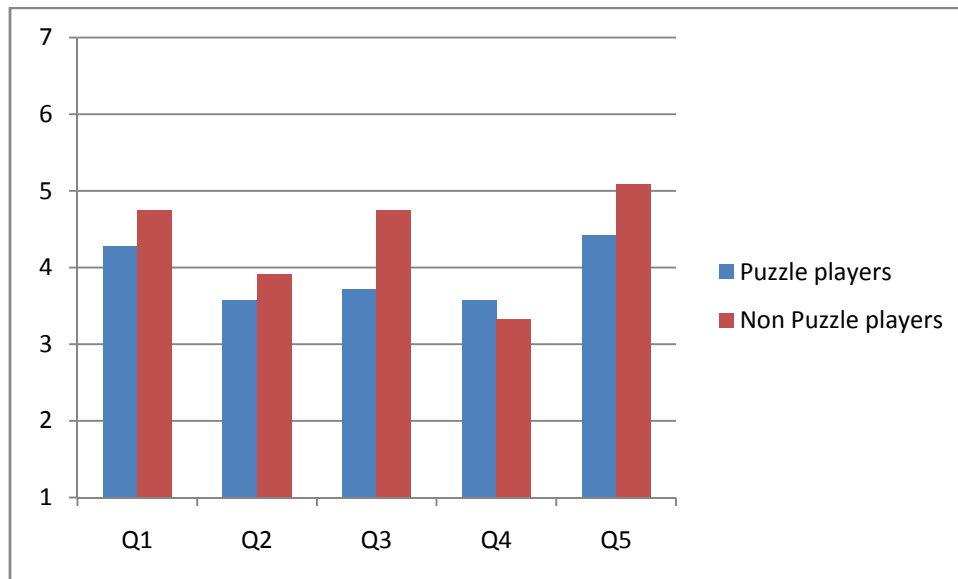
**Illustration 6.4:** Graph detailing the divide of users on which version they tried first, and which version they preferred.

As can be seen in *Illustration 6.4*, the majority of the users tried the normal version of the game first. However after trying both versions out, the majority decided that the simple version was their favorite, which goes well with the indications found in regards to Q4 and Q5 in the usability test results. When looking at the test results from the users trying the simplistic version first, there does not appear to be any significant differences from the results of the users who tried the normal version first.

#### 6.4.3 A second look at the test results

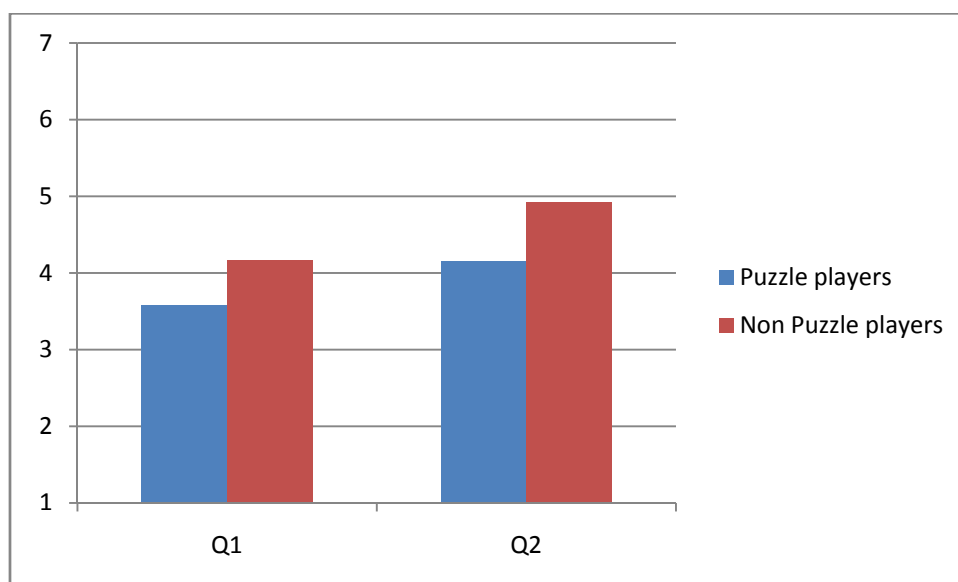
As the various comments from the questionnaire - especially concerning the various engagement questions - suggested that there were mixed opinions in the test group, it was decided to try and divide the group into two separate smaller groups. One of the initial questions asked in the questionnaire was related to the types of games that the various users enjoy playing. As such it seemed like an obvious choice to divide the users into people who preferred the genre of the product game – the puzzle genre – and those who did not play puzzle games on a regular basis.

This created a new set of test results that can be looked upon.



**Illustration 6.5:** Graph showing the usability test results, after the test group has been divided.

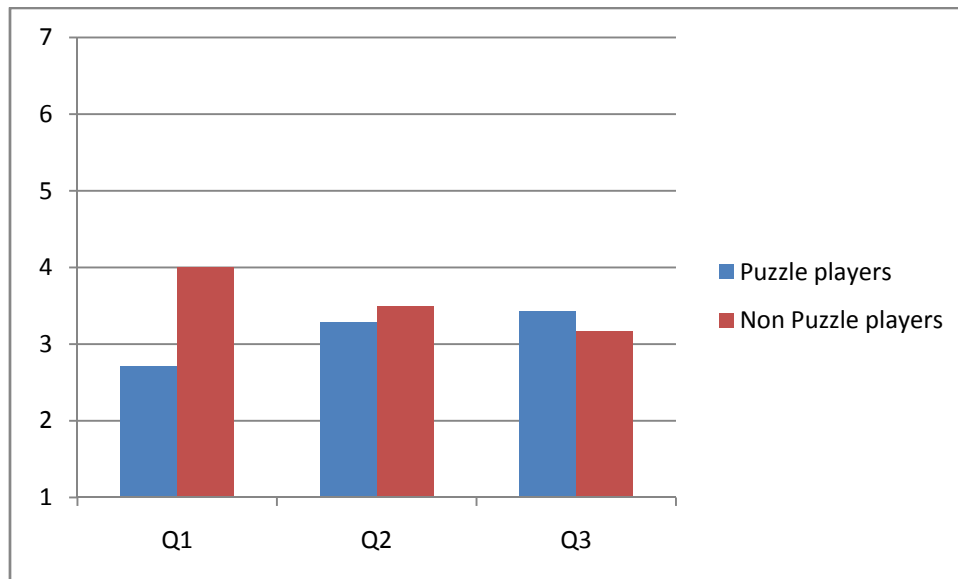
The division of the test group provided a little more insight into the issues with the usability in the product. A general tendency that can be seen in *Illustration 6.5* is that people who play puzzle games are generally rating the usability areas lower than that of non puzzle players. Looking at the test comments by the users, the complaint from one of the puzzle players is that he cannot cast spells whenever he feels like it, while the comments from the non puzzle players requests a better interface. While very subtle, there is an indication to be found in Q4 in which it seems that the puzzle players have less against a more complex manual spell system than the non puzzle players.



**Illustration 6.6:** Graph showing the sensual engagement test results, after the test group has been divided

*Illustration 6.6* shows the changes in the test results after the test group has been divided, and shows some minor changes compared to the group in general. In both Q1 and Q2 there are indications when comparing to the previous sensual engagement test results, that it is the puzzle players that are affecting the result in a

negative way. Non puzzle players are generally more content with both how the visuals appear and also with the way that the visuals present information to the user. Compared to the previous sensual test, both test results illustrate a better result for the non puzzle players, and a lower result for the puzzle players.



**Illustration 6.7:** Graph depicting the results of the challenge part of the test, after the group was divided.

The challenge part of the renewed test results shown in *Illustration 6.7* is also where the biggest differences compared to the previous set of tests can be found. Interestingly enough there is an indication towards the non puzzle players finding the task as being neither easy nor difficult, whereas the puzzle players are finding it more than a little difficult. The non puzzle players do find the task of finding the exit as being difficult – Q2 -, however the puzzle players are still having more difficulties finding it than the non puzzle players. The one place where it switches place is with Q3, depicting the desire to continue playing. Despite the indications of having a harder time, the puzzle players appear to be more inclined to continue playing than their non puzzle counterparts.

This expanded look upon the test results did not display any considerable differences from the original test results, but does provide an alternate point of view for the results that should be considered in further iterations when designing puzzle games.

#### 6.4.4 A third approach to the results

As the dividing of the test group into genre preferences did not provide any significant changes to the results, a third approach was decided upon in hopes that it would either provide a new view on the results or usable insight for future iterations. For this subchapter, users with the preference for the normal edition will be called normal users and the users with a preference towards the simplified version will be called simplified users.

At the end of the questionnaire, the users decided upon which of the two control schemes they preferred to use. As such, the test group can be divided into users preferring one version and another.

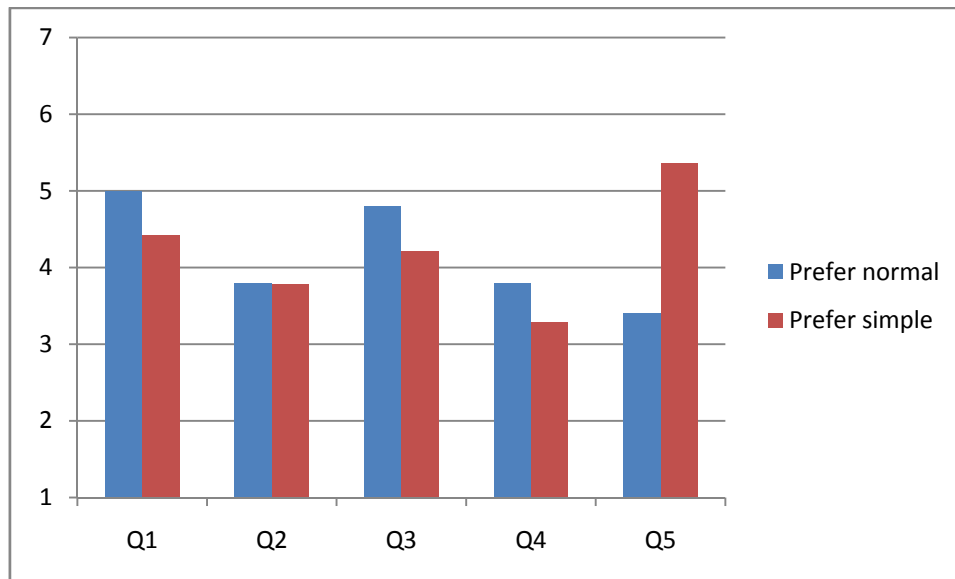


Illustration 6.8: Usability test results after the preference division

The results from the usability part of the preference division can be seen in *Illustration 6.8*. With this division the graph becomes slightly more interesting to observe. The normal users tend to find the controls as being easier to handle, than the simple users – despite having the possibility to use more than twice the amount of keys on the keyboard. This could indicate that the controls in general are not as horrible as could have been expected from the original test results – as there are at least some who find the controls as being slightly good. Q2 regarding the information provided in the game is close to being identical to that of the original test results. It is also slightly noticeable in Q3 that the simplified users find the map less useful than users who prefer the normal version. This could indicate that integrating and automating an even more simplified control scheme would be appealing to the simplified users. The rating in Q4 and Q5 is partially as could be expected, with each group rating their preferred control scheme as the highest. There is however a noticeable indication that the simplified method of casting spells is functioning well with the users, as it has received the highest score of any result thus far.

The remaining results regarding engagement were of too high similarity to those from the two other test result subchapters and will therefore not be described further. They are however available for viewing in *10.Appendix.II -Divided preference graphs* or in full detail on the cd included with this report.

### Sum up

The results of the test displayed a mediocre affinity by positioning all the results between a slightly bad rating of 3 and a slightly good rating of 5. In an attempt to uncover additional results, two approaches of dividing the test participants into groups were attempted. One in which the division was based on the preference of genres, creating a group for people who play puzzle games and one who does not. The other approach was by dividing the users based on their preference of control scheme, creating a group of users with a preference for the simplified version and one for the normal version.

These divides provided an insight into the results of the users, such as the indications that puzzle players generally found the game harder than users not used to playing puzzle games. While not creating any new results, the added insight could prove useful for further iterations of the product.

With the initial results of the test having been gone over, a more in depth analysis can now commence.

## 6.5 Test Analysis

The test analysis will serve as a tool to attempt and create reasoning behind the test results. Each of the test questions will be gone through and discussed using both the quantitative and qualitative data acquired from the test.

### 6.5.1 Usability

Each of the five questions from the usability part of the test will be gone through one by one, and the result will then be related to the theories researched previously in the report. Once a reason behind a test result has been determined, a way of surpassing it will be proposed.

The control scheme was designed in Chapter 4.4.2 *The controls*, with the idea of providing the user with a control scheme that was almost entirely recognizable by the user. This was done so that the user would not need as much time to get to know and master the controls, and could place his effort and motivation towards playing the game instead. In the qualitative questionnaire the functionality of the controls were given a rating of 4.58, which does provide an indication that it was not directly causing disengagement. However based on Brown and Cairns theory discussed in Chapter 3.2.1 *An alternate approach to Engagement*, one of the barriers that stood between the users and the feeling of engagement was access, the ability to interact with the game and be provided with feedback. As the rating in the test is not very high for the controls, it is possible that the barrier of access was not entirely overcome. Looking at the qualitative answers regarding the controls, a few plausible reasons why this could have happened reveal themselves. One user describes a problem that did not appear during the first iteration test of the implementation, but quite possibly should have been considered as being an issue. Allowing the user to control using the WASD as an alternative to the arrow keys, as well as having the map button located on the key M became an issue when entering a spell room. Suddenly the functionality of these keys was replaced with other functionalities. It is possible that this replacement of functionality could have caused confusion among the users, which in turn resulted in a lower control score. The users themselves also provided suggestions to alternatives, such as moving the controls to the numerical keypad or allowing the game to be controlled using the mouse. Another suggestion was to lower the amount of key inputs from the current 12, as it becomes a bit too much when going from spell room to spell room and having to re-input such long spells.

The information in the game was intended as being as easy to comprehend as possible while providing the user with the necessary information about the state of the character, the events of the room entered as well as an indication as to where in the maze the user was located. Several approaches to providing this information were discussed in Chapter 4.4.3 *The interface*, in an attempt to find a suitable one for the product. The test result on the information provided during the game was 3.79, going towards the not very useful part of the scale. Despite the original concept of the informational text being displayed as if reading a book, starting at the top left corner of the screen - some users reported that they did not notice the text immediately, as it was not very eye-catching. Also the coordinate method of showing the position in the map was not very well liked, as the users requested a different way of presenting the information on. Some of the users have commented that they would like more hint squares, while technically being part of the challenge part of the test, it appears that a possible reason for the lower score in the information is based on the desire for additional hints towards an exit. Another feature requested is some form of subtle warning that will alert players that a trap is located nearby, perhaps through visual cues.



The map, as described in Chapter 5.3 *2nd Implementation iteration* was originally intended as an additional method for the users to visualize their location in the maze, based on the results from the second iteration. The test result for the map was 4.37, placing it on the positive side of the scale with an indication that it has been useful at least to some extent. The decision was made to not make the map display the rooms visited, in order not to allow the users to return to a healing room whenever they found a trap room. However the users found the inclusion of the map, without the map filling out as they went along, as being an annoying aspect. While it proved useful to some of the users, the majority of the comments from the qualitative data point toward the map as being more of a useless feature than a helpful one. A common complaint from the users is that the map does not point them towards the exit. One user even commented that he did not even notice that there was a map, despite this being explained both on the front page of the questionnaire and being displayed at the bottom of the game screen throughout the entire duration of the game. This relates to the information discussion previously in this subchapter, that even though the feature had been explained twice using text – it had not been presented in a way that the users expected it to. A user suggestion consisted of filling the map with black boxes, and slowly removing them as the player went along. This could be a possible solution to the map irritation felt by the users, despite going against the original concept by allowing them to backtrack to where the users originally came from such as a healing room.

The last two questions in the usability part of the test were a rating of the two control schemes of different complexity, animated at fulfilling the final problem formulation stated. The results awarded the normal version with an average of 3.42 and the simplistic with 4.84. This initially provides a clear indication that the users favored the simplistic edition based on the controls used, an indication that is also backed up by the majority of the users choosing the simplified version as their preferred version. However due to the mix-up of the controls explained in beginning of this subchapter as well as the users comments on the interaction method being a tad too complex compared to the simple one, it is quite possibly that the score of the normal control scheme could have been much higher. The users comment positively on the ability to cast whatever spell they can think of, but at the same time become displeased with the high amount of inputs required for each spell as well as the lack of a structured approach to casting the spells. The effects of this problem could prove to be fatal for the successful implementation of the project, and the ability to fulfill the final problem formulation.

### 6.5.2 Engagement

In order to test whether or not engagement was present in the product, a series of questions relating to different aspects of engagement were included in the test in accordance with the engagement stages, Brown and Cairns theory and O'Brien and Tom's framework.

The first question was connected to O'Brien and Tom's framework at several levels as well as the theory by Brown and Cairns. The users were asked to rate the graphics, which they did with an average result of 3.95. This indicates that the graphics weren't really as useful as originally planned. The O'Brien and Tom framework use the visuals both to invoke the point of engagement in users, as well as a method to keep them engaged. This relates to one of Brown and Cairns engagement barriers, if the user has nothing that motivates them to play; they are unable to become engaged. The purpose was of the graphics was thus to catch the attention of the user and motivate them to try and play the game, the motivation and engagement would be sustained by the graphics as suggested in the framework by O'Brien and Tom. However with the result of the test, it appears as though the graphics did not successfully catch the attention of the users. The second

question in the sensual engagement was closely related to the first, in that they are both working with the visuals of the game. In the case of the second question, it was how useful the graphics were in providing information to the user. The result of this in the test was a 4.63, indicating that at least parts of the intended pieces of information were perceived. Looking at the user comments for the graphics, the main issues lie with its quality not being top notch, and that it they were unable to understand the concept of the rooms without having an avatar to control within them. The general opinion in differentiating the rooms from one another and remembering their effects, seem to have functioned somewhat. However one user comments that he would like the graphics to more clearly show the purpose of the rooms.

The only question related to the emotional engagement was the users felt that the game had been fun to play. This question received the lowest score of all the questions in the project with only 2.84, lab, below the slightly bad area on the scale. As with the case of the controls, it is highly likely that the results of this question could have been considerably higher if the usability as well as the visuals of the product had been able to create a better result.

Moving on to the overall difficulty of the product, the test result provided a 3.53 placing it somewhere between a medium difficulty and being slightly difficult. As has been briefly mentioned in the other areas of this analysis, some of the elements that the users felt made the game too difficult was the lack of properly conveyed information regarding possible interactions and events upon entering the rooms. If a user has missed this information, it is likely that the hints contained in some of the rooms placed to make the game easier have been overlooked. However taking the theory of *flow* into account as well as Scott Kim's suggestion to pacing puzzles, the difficulty should not be placed directly between hard and easy. While it is true that the ability of the user as well as the difficulty of the challenges should be matched, a saw-tooth like shape of switching between slightly easy and slightly difficult puzzles is needed in order to keep them interesting, an approach that is also supported by Chanel, Rebetez, Bétrancourt, & Pun (Chanel, Rebetez, Bétrancourt, & Pun, 2008) . As such it is possible that the current difficulty level should be kept intact, and based on the user comments spread out the trap rooms a bit more so that they are not connected to one another, without removing them entirely.

The difficulty present in the task of finding the exit, ended with a result of 3,42 – being slightly more difficult than the overall. Once again as mentioned above, the difficulty here could very likely be caused by users not noticing the hints provided by the clue and distance rooms, which naturally makes the game a whole lot harder. However the challenge of the entire game is the task of successfully navigating the maze and finding the exit, and as such making it too easy would ruin the concept of the puzzle. Some users even suggested in the comments, ways to make the game even more difficult by adding monsters wanting to kill the player to certain rooms.

The third challenge question was whether or not the user would want to continue playing the game, and the result for this question was 3.26. According to the post-engagement, the users should either be left with frustration that they could not complete the level or the desire to continue onwards to the next level. This is also supported by the second stage of immersion from Brown and Cairn called engrossment (Brown & Cairns, 2004), in which the user has already invested enough time, effort and attention to make them want to keep on playing. That the result is so low could indicate that not all of the participants felt engaged, and for the ones that did manage to become engaged, it was very lightly. The slightly low score could be expected when the emotional engagement question of fun, scored even lower. In order to understand why the players would want to continue or not want to continue, it becomes necessary to look at the comments. One user's

An investigation into the effects on engagement, caused by control scheme simplifications in games.

comment matches perfectly with the previous statement, stating that he does not want to play as the game is not fun. However some users did express an interest in wanting to continue to play the game, reasoning their decision with things such as the game being fun and quick to complete and with a few more levels would be the perfect time waster.

The next section of the questionnaire was the version preference questions. This was intended to provide an idea as what control scheme version the users preferred to use. 17 of the 19 test participants started out with the normal version. After testing both versions, 14 decided that they preferred to use the simple version over the normal version. This gives an indication that the simple version quite possibly was better suited for this type of game, however one must keep in mind that the normal version suffered from a series of problems relating to the usability of its controls. The graphical influence can to some extent be ignored, as the same flaws influence both the simple and the normal versions of the product.

As there is no direct indication of engagement being present in the product, the closest this iteration of testing can come to fulfilling the final problem formulation is to point towards the simple version being selected by two thirds of the test participants. This could be an indication that if the same, or close to the same game play can be achieved in a video game with a simpler control scheme, one should definitely consider the possibility.

## 7. Conclusion

The motivation of this project was to investigate the effects on engagement that comparing a set of simplified and normal control schemes would have. With everything in the video game world moving towards multi platform, and portable releases – the focus on achieving close to the same with less is becoming a more and more important factor in game development. With inspiration from games from a previous decade as well as more recent games, this project delimited its scope to focusing on investigating the effects caused by the simplification of gameplay mechanics on engagement.

In the pre-analysis the project looked into the theories of engaging gameplay, simplification and a brief look into the different genres of video games. A target group for the project was also determined and further delimitation lead to the creation of a final problem formulation:

**How can the effects on engagement be determined, in which the control scheme of a 2D puzzle game is compared to that of its simplified counterpart**

Through the analysis chapter, a methodology was chosen which allowed the project could go through several iterations in order to refine the product. The term engagement was further investigated with theories by O'Brien and Toms and Brown and Cairns being the central area of interest. A deconstruction of the term was created in order to have a simple way of referring to the individual aspects such as the interactive and social components. The theory of *flow* by Csikszentmihalyi was also looked into, providing possible testing tools for engagement. An additional theory was added that built upon the concept from *flow* of balancing the ability of the user and the difficulty of the challenge, which introduced a constant jagged change in difficulty switching between a little easy and a little hard , in order to avoid the user becoming bored with the product. The concept of engagement stages was revitalized from an old report (Jensen, Etzerodt, Christensen, & Jørgensen, 2009), and used to provide the possibility of additional testing methods. Following this, the analysis chapter looked into the area of game theory, meaningful play, the magic circle and different

approaches that one could take in order to create a video game. The analysis chapter ended with the creation of a list of solution requirements, intended for use with the design.

At the beginning of the design phase, the rules and level design were decided upon, with the game becoming a puzzle game with the goal of escaping a maze. The visual design worked from there in creating the maze as a dungeon, and defining the necessary parts required on the user interface – originally defined by the rules of the game. Here the simplified and normal controls were also decided upon, as being part of the spell casting sequence in the maze. In order to have a clear idea of how the implementation should take place, a set conditions based on the input from the user and the systems feedback were made. This all resulted in a flowchart being made and from the flowchart a set of game classes were defined that would allow easy implementation in the following phase.

The implementation phase followed the procedure described from the design phase, and ended up with a small usability test intended to catch the worst of the bugs. This resulted in a series of changes to be made for the product, in an attempt to ensure that the final product would be able to fulfill the final problem statement.

The final test showed slight indications that the simplistic was preferred over the normal control scheme, both in terms of rating their usefulness and also in relation to preference. However there was no clear indication of engagement being present, and as such creating a correlation between the engagement and the control scheme would be unfeasible.

In conclusion, this project was unable to achieve any results that could provide an insight into the effects on engagement caused by simplified control schemes. It did however create a prototype that following the test analysis can be reiterated, allowing an additional attempt at finding proof for the final problem formulation.

## 8. Bibliografy

- Adams, E. (1999, July 16). *The Designer's Notebook: Simplification*. Retrieved April 5, 2011, from Gamasutra: [http://www.gamasutra.com/view/feature/3356/the\\_designers\\_notebook\\_.php](http://www.gamasutra.com/view/feature/3356/the_designers_notebook_.php)
- Adams, E., & Rollings, A. (2003). *on Game Design*. New Riders.
- Adams, E., & Rollings, A. (2003). *On Game Design*. New Riders Publishing.
- Blizzard Entertainment. (2010, October 7). *Blizzard Entertainment: Press Releases*. Retrieved May 15, 2011, from Blizzard Entertainment: <http://eu.blizzard.com/en-gb/company/press/pressreleases.html?101007>
- Blizzard Entertainment. (2000, June 29). *Diablo 2*. USA: Blizzard Entertainment.
- Blizzard Entertainment. (1998, March 31). *Starcraft*. Blizzard Entertainment.
- Blizzard Entertainment. (2010, July 27). *StarCraft II: Wings of Liberty*. Blizzard Entertainment.
- Blizzard Entertainment. (2004, November 23). *World of Warcraft*. Blizzard Entertainment.
- Brown, E., & Cairns, P. (2004). A grounded investigation of game immersion. *CHI EA '04 extended abstracts on Human factors in computing systems* .
- Bungie. (2001, November 15). *Halo: Combat Evolved*. Microsoft Game Studios.
- Chanel, G., Rebetez, C., Bétrancourt, M., & Pun, T. (2008). Boredom, Engagement and Anxiety as indicators for adaption to difficulty in games. *Mindtrek '08 Proceeding* .
- Charlton, J. P., & Danforth, I. D. (2007). Distinguishing addiction and high engagement in the context of online game playing. *Computers in Human Behaviour* , 1531-1548.
- Chen, V. H., Duh, H. B., Phuah, P. S., & Lam, D. Z. (2006). Enjoyment or Engagement? Role of Social Interaction In Playing Massive Multiplayer Online Role-Playing Games (MMORPGS). *Lecture Notes in Computer Science* .
- Crytek. (2009, October 14). *CryEngine 3*.
- Crytek Frankfurt. (2011, March 22). *Crysis 2*. Electronic Arts.
- Crytek Frankfurt. (2007, November 13). *Crysis*. Electronic Arts.
- Csikszentmihaly, M. (1991). *Flow: The Psychology of Optimal Experience*. New York: Harper Collins.
- Daily Rush. (1998). *Daily Rush*. Retrieved 5 21, 2011, from Daily Rush: [www.dailyrush.dk](http://www.dailyrush.dk)
- Dickey, M. D. (2005). Engaging by design: How Engagement Strategies in Popular Computer and Video Games can Inform Instructional Design. *Educational Technology Research and Development* , pp. 67-83.
- EA Redwood Shores. (2008, October 20). *Dead Space*. Electronic Arts.
- Epic Games & Digital Extremes. (2004, March 16). *Unreal Tournament* . Atari.

- Epic Games. (2006). Gears of War. Microsoft Game Studios.
- Epic Games. (1999, November 30). UnrealEd.
- ESA. (2010). *Essential Facts About the Computer and Video Game Industry*.
- Glinert, E. (2010). *Gamasutra*. Retrieved May 26, 2011, from Designing Games That Are Accessible To Everyone: [http://www.gamasutra.com/view/feature/3538/designing\\_games\\_that\\_are\\_.php?print=1](http://www.gamasutra.com/view/feature/3538/designing_games_that_are_.php?print=1)
- Huizinga, J. (1955). *Homo Ludens: A Study of the Play-Element in Culture*. Boston: Beacon Press.
- id Software. (1999, December 2). Quake III. Activision.
- id Software. (1992, May 5). Wolfenstein 3d. Apogee Software.
- Irem. (1987). R-Type. Nintendo.
- Jensen, M. B., Etzerodt, K., Christensen, M. W., & Jørgensen, H. (2009). *Physicasia*.
- Jones, M. G. (1998). *Creating engagement in computer-based learning environments*. Retrieved March 18, 2011, from ITFORUM: <http://it.coe.uga.edu/itforum/paper30/paper30.html>
- Jørgensen, K. (2011, April 12). *The User Interface Continuum: A Study of Player Preference*. Retrieved from Gamasutra: [http://www.gamasutra.com/view/feature/6346/the\\_user\\_interface\\_continuum\\_a\\_.php](http://www.gamasutra.com/view/feature/6346/the_user_interface_continuum_a_.php)
- Lazy 8 Studios. (2009, April 14). Cogs.
- Lima Sky. (2009, April 6). Doodle Jump.
- Lin, J. J., Mamykina, L., Silvia, L., Delajoux, G., & B., S. H. (2006). Fish'n'Steps: Encouraging Physical Activity with an Interactive Computer Game. *Lecture Notes in Computer Science*.
- McMahan, A. (2003). Immersion, Engagement and Presence – a method for analyzing 3d video games. *The Video Game, Theory Reader*, 77-78.
- Mechadeus. (1995). The Daedalus Encounter.
- Merriam-Webster's Online Dictionary. (2011). Retrieved 5 10, 2011, from <http://www.merriam-webster.com/dictionary/engaged>
- MicroProse Software Inc. (1996). Transport Tycoon Deluxe. MicroProse Software, Inc.
- Miyamoto, Shigeru. (1983). Super Mario Bros. Nintendo.
- Namco. (1980, May 22). Pacman. Namco.
- Nutt, C. (2008, April 7). *Moving The Industry Forward: Peter Molyneux Speaks*. Retrieved May 26, 2011, from Gamasutra: [http://www.gamasutra.com/view/feature/3607/moving\\_the\\_industry\\_forward\\_peter\\_.php](http://www.gamasutra.com/view/feature/3607/moving_the_industry_forward_peter_.php)
- O'Brien, H. L., & Toms, E. G. (2008, April). What is User Engagement? A Conceptual Framework for Defining User Engagement with Technology. *Journal of the American Society for Information Science and Technology Volume 59*, pp. 938-955.



Parallax Studio. (2010, November 5). Darkstar: The Interactive Movie.

Partington, J., & Thackray, J. (1991). Spy Snatcher. Topologika.

Playdead. (2010, July 21). Limbo. Microsoft Game Studios.

Prokein, R. (2011). *Reiner's Tilesets*. Retrieved Maj 2, 2011, from Reiner's Tilesets:  
<http://www.reinerstilesets.de/>

Rollings, A., & Morris, D. (1999). *Game Architecture and Design*. Paraglyph PR.

Rovio. (2009). Angry Birds. *Angry Birds* . Rovio.

Salen, K., & Zimmerman, E. (2003). *Rules of Play: Game Design Fundamentals*. Cambridge, USA: MIT Press.

Semi Secret Software. (2009). Canabalt. *Canabalt* . Semi Secret Software.

Sierra On-Line. (1992). Police Quest: In Pursuit of the Death Angel.

Thorbjørn, L. (2004, June 15). Tiled Map Editor.

Ubisoft. (2007). Assassin's Creed. Ubisoft.

Ubisoft Shanghai. (2008, November 4). Tom Clancy's EndWar. Ubisoft.

Ubisoft Shanghai. (2009, May 12). Tom Clancy's EndWar. Ubisoft.

Unity Technologies. (2005, July 14). Unity 1.0.2.

Valve. (1996, September). Hammer Editor.

Valve. (2007). Portal. Valve.

Westwood Studios. (1996, October 31). Red Alert. Virgin Interactive.

## 9. List of illustrations

*Illustration 1.1* – Own creation based on <http://img182.imageshack.us/i/screenshot020vp9.jpg/sr=1> & [http://sven-slootweg.nl/blog/wp-content/uploads/2009/09/transport\\_tycoon\\_deluxe\\_4.jpg](http://sven-slootweg.nl/blog/wp-content/uploads/2009/09/transport_tycoon_deluxe_4.jpg)

*Illustration 1.2* – Own creation based on <http://theportablegamer.com/wp-content/uploads/2010/04/Angry-Birds-HD-Screen.JPG> & [http://xspblog.com/wp-content/uploads/2009/08/full\\_canabalt2f.png](http://xspblog.com/wp-content/uploads/2009/08/full_canabalt2f.png)

*Illustration 2.1* – Own creation based on <http://brunoberry.wordpress.com/2010/07/22/game-review-cogs/> & <http://beefjack.com/news/lan-denied-starcraft-ii/>

*Illustration 2.2* – Own creation based on <http://www.honestgamers.com/games/15231/Police-Quest-1-In-Pursuit-of-the-Death-Angel-VGA-Remake.html> & <http://www.somegamereviews.com/2010/01/doodle-jump-ipodiphone/>

*Illustration 3.1* – Created by Heino Jørgensen (Jensen, Etzerodt, Christensen, & Jørgensen, 2009)

*Illustration 3.2* – Own creation interpreted from (Chanel, Rebetez, Bétrancourt, & Pun, 2008)

*Illustration 3.3* - Created by Mikkel B. Jensen (Jensen, Etzerodt, Christensen, & Jørgensen, 2009)

*Illustration 4.1* – Own creation

*Illustration 4.2* – Own creation

*Illustration 4.3* – Own creation

*Illustration 5.1* – Created by Mikkel B. Jensen (Jensen, Etzerodt, Christensen, & Jørgensen, 2009)

*Illustration 5.2* – Own creation

*Illustration 5.3* – retrived from <http://www.mapeditor.org/>

*Illustration 5.4* – Own creation

*Illustration 5.5* – Own creation

*Illustration 5.6* – Own creation

*Illustration 6.1* – Own creation

*Illustration 6.2* – Own creation

*Illustration 6.3* – Own creation

*Illustration 6.4* – Own creation

*Illustration 6.5* – Own creation

*Illustration 6.6* – Own creation

*Illustration 6.7* – Own creation

An investigation into the effects on engagement, caused by control scheme simplifications in games.

Illustration 6.8 – Own creation

## 10. Appendix

### Appendix.I - Questionnaire

These were the questions from the questionnaire. While they have been added here, they are available on the cd, in a more structured .pdf file.

#### Demografiske spørgsmål

Denne del af spørgeskemaet er inkluderet, for at få en smule kendskab til hvilke brugere jeg får igennem testen.

##### Hvad er din alder? \*

- ☐ Under 15  
☐ 15-17  
☐ 18-26  
☐ 27-32  
☐ 33-38  
☐ 39-45  
☐ 46-51  
☐ Over 51

##### Hvilket køn tilhører du? \*

- ☐ Kvinde  
☐ Mand

##### Hvor ofte spiller du video spil? \*

Dette gælder både konsol og pc spil

- 1 2 3 4 5  
 Meget lidt ☐ ☐ ☐ ☐ ☐ Rigtigt meget

##### Hvilke genre af spil, spiller du oftest? \*

Vælg gerne flere

- ☐ Action  
☐ Strategi  
☐ Simulation  
☐ Sport  
☐ Adventure  
☐ Puzzle  
☐ Fighting  
☐ Other:

##### Hvor længe spiller du spil af gangen? \*

1 time? 2 timer? femten timer?

- ☐ Mellem 0 og 1 time af gangen  
☐ Mellem 1 og 2 timer af gangen  
☐ Mellem 2 og 3 timer af gangen  
☐ Mellem 3 og 4 timer af gangen  
☐ Mellem 4 og 5 timer af gangen  
☐ Over 5 timer af gangen

##### Hvordan ville du vurdere metoden til at kaste trylleformularer på? \*

Spørgsmålet her er møntet på udgaven hvor du skal taste 12 bogstaver ind.

- 1 2 3 4 5 6 7  
 Meget dårlig ☐ ☐ ☐ ☐ ☐ ☐ ☐ Meget god

##### Hvordan ville du vurdere metoden til at kaste trylleformularer på? \*

Spørgsmålet her er møntet på udgaven hvor du blot skal trykke på en enkelt tast.

#### Brugervenlighed

Denne del af spørgeskemaet omhandler brugervenlighed, og anvendes til at finde eventuelle fejl eller mangler der umiddelbart er at finde i programmet.

##### Hvordan vil du vurdere styringen i spillet? \*

Var det til at finde ud af og fungererede det som det skulle?

- 1 2 3 4 5 6 7  
 Meget dårlig ☐ ☐ ☐ ☐ ☐ ☐ ☐ Meget god

##### Var der noget du kunne forestille dig kunne forbedre styringen, eller måske en helt anden tilgang til det?

##### Hvordan vil du vurdere brugbarheden af kortet i spillet? \*

Kortet kunne frembringes under spillet ved at trykke på M tasten.

- 1 2 3 4 5 6 7  
 Ubrugeligt ☐ ☐ ☐ ☐ ☐ ☐ ☐ Meget brugbar

##### Kommentarer til kortet, og eventuelle forbedrings foreslag:

Kunne du komme på en alternativ måde at viderebringe spillerens position på?

#### Det visuelle indtryk

De følgende spørgsmål omhandler den grafiske del af spillet.

##### Hvordan vil du vurdere de grafiske elementer i spillet? \*

Var de helt grusomme at se på, eller går det an?

- 1 2 3 4 5 6 7  
 Forfærdelige ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fantastiske

##### Hvor brugbar var grafikken til at overbringe information? \*

Fx. Kunne du genkende de forskellige rum's effekter blot ved at se grafikken?

- 1 2 3 4 5 6 7  
 Ubrugeligt ☐ ☐ ☐ ☐ ☐ ☐ ☐ Meget brugbar

##### Kommentarer til det grafiske aspekt i spillet

Foreslag til andre måder at overbringe informationer gennem grafikken?

## Underholdnings værdi

Dette punkt er hovedsageligt for at finde ud af hvor underholdende spillet er på sit nuværende stadie.

Hvor underholdende vil du vurdere spillet som at være? \*

1	2	3	4	5	6	7	
<hr/>							
Ikke sjovt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Umådelig sjovt

## Udfordring

Denne del af spørgeskemaet er til for at undersøge sværhedsgraden af spillet.

Hvordan vil du vurdere den overordnede sværhedsgrad i spillet? \*

Baseret på ting såsom om du døde konstant, eller om det var det for nemt at finde rundt i labyrinten.

1	2	3	4	5	6	7	
<hr/>							
Meget svært	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Meget nemt

Hvordan vil du vurdere sværhedsgraden, omkring at finde udgangen? \*

Fandt du den super hurtigt? eller har du overhovedet ikke fundet den endnu?

1	2	3	4	5	6	7	
<hr/>							
Meget svært	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Meget nemt

Kommentarer til de to ovenstående spørgsmål:

Foreslag til at gøre spillet sværere/lettere. Andre kommentarer er også velkomne.

Hvor meget kunne du tænke dig at fortsætte med at spille spillet? \*

Fx. hvis der var flere baner, og måske andre typer rum på andre levels?

1	2	3	4	5	6	7	
<hr/>							
Slet ikke fortsætte	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Meget gerne fortsætte

Kommentarer til hvorfor du kunne/ikke kunne tænke dig at fortsætte med at spille:

## Udgave preference

Denne del af spørgeskemaet er udelukkende til for at finde ud af hvilken af de to udgaver, er den fortrukne.

Hvilken udgave foretrækker du af spillet? \*

- ☐ Udgaven hvor man selv fik lov til at taste hele trylleformularene ind.
- ☐ Udgaven hvor man bare skulle trykke en enkelt gang for at kaste en trylleformular

Hvilken udgave af spillet prøvede du først? \*

- ☐ Udgaven hvor man selv fik lov til at taste hele trylleformularene ind.
- ☐ Udgaven hvor man bare skulle trykke en enkelt gang for at kaste en trylleformular

Denne boks er til eventuelle kommentarer som ikke er dækket af de forhenværende spørgsmåls kasser. Har du en kommentar eller ti til spillet som du gerne vil af med, kan du skrive det her.

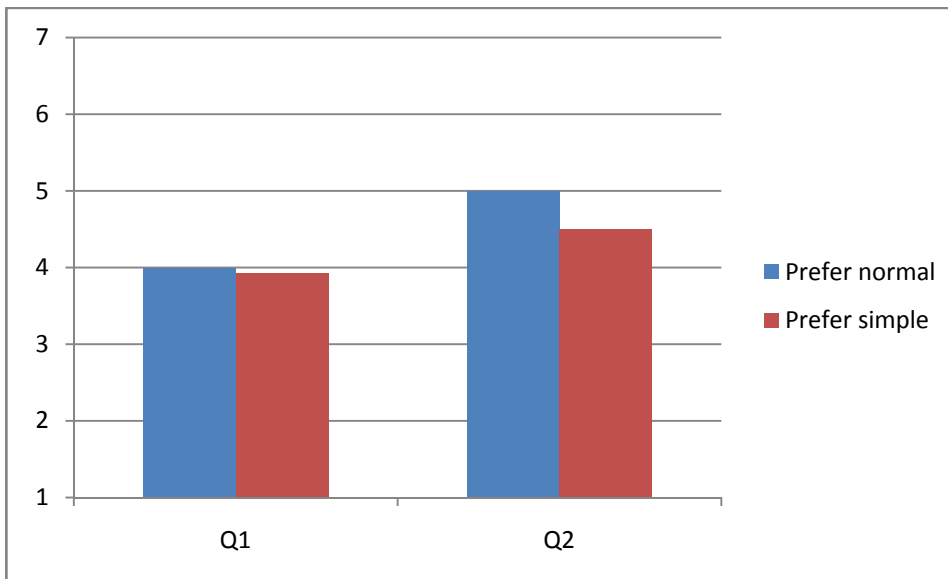
Du kan også inkludere din mening omkring simplificeringen af spil generelt (såsom i tilfældet med Crys og Crys 2 - hvor dragtens stryke og hastigheds funktioner var simplificeret).

## Appendix.II - Divided preference graphs

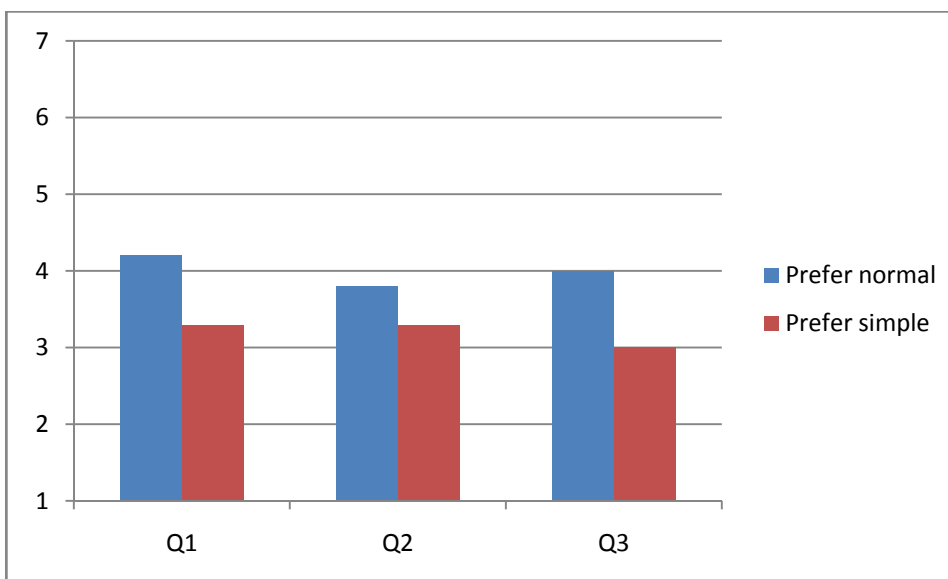
Additional graphs for the preferred version divide of the test group. The full data can be found on the cd.

Sensual engagement:

An investigation into the effects on engagement, caused by control scheme simplifications in games.

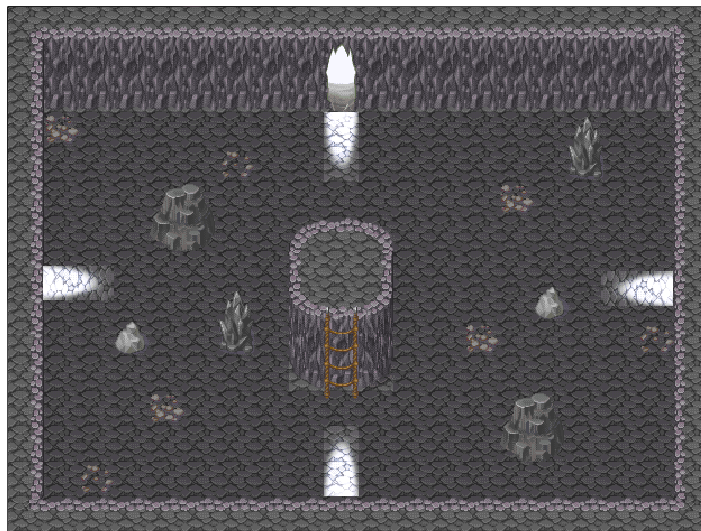


Challenge:



### Appendix.III - Screenshots of rooms

Here is an example of one of the rooms in their original design followed by the redesign made after the second iteration. All of the rooms present in the game, in both old and new form can be found on the cd.



The two above images display the exit in the game, which is the goal for the users to reach in the product. From the original design down to the top down view, a few details had to be let go, resulting in the square frame on the bottom picture.



## Appendix.IV - Game.cpp

The Game.cpp is where the main loop of the game is currently running. Note that it is not entirely cleaned up due to time constraints; it is however in working order. The remaining files, as well as a compiled version of the game - can be found on the cd included with the report.

```

1  #include <string>
2  #include <iostream>
3  #include <fstream>
4  #include <sstream>
5  using namespace std;
6  #include "Game.h"
7
8  Game::Game(int levelselected)
9  {
10     this->rememberedspells = 0;
11     SetLevel(levelselected); //Creates a board, reading the first level file.
12     SetIsRunning(true);
13     LoadSFX(GetLevel());
14     SetInput(0,0);
15     simplified = false;
16     Text[1] = "Player position(x,y): ";
17     Text[2] = "Level: ";
18     Text[3] = "Controls: Arrow keys or WSAD for movement, M to access the map or
ESC-button to exit";
19     Text[4] = "Type in a magic spell using the letters on the keyboard: ";
20     Text[5] = "letters remaining.";
21     TextPositionSet = false;
22 }
23 Game::~Game(void){
24 }
25 void Game::SetIsRunning(bool running)
26 {
27     this->isRunning = running;
28 }
29 bool Game::GetIsRunning(){
30     return this->isRunning;
31 }
32 int Game::GetLevel()
33 {
34     return this->level;
35 }
36 void Game::SetLevel(int number)
37 {
38     this->level = number;
39     Board.CreateBoard(GetLevel());
40 }
41 void Game::LoadSFX(int level)
42 {
43     if(level == 1)
44     {
45         if(!Music1.OpenFromFile("Music\\bg2.ogg"))
46             cout << EXIT_FAILURE << endl;
47         Music1.SetVolume(25);
48         Music1.Play();
49     }
50 }
51 void Game::SetInput(int x, int y)
52 {
53     this->inputX = x;
54     this->inputY = y;
55 }
56 int Game::GetInputX()

```

```

57 {
58     return this->inputX;
59 }
60 int Game::GetInputY()
61 {
62     return this->inputY;
63 }
64 void Game::SetInput(char letter, int number)
65 {
66     this->input[number] = letter;
67 }
68 char Game::GetInput(int number)
69 {
70     return this->input[number];
71 }
72 void Game::SetText(int posX, int posY)
73 {
74     for(int a = 0; a < 8; a++)
75     {
76         TextOutput[a].SetFont(sf::Font::GetDefaultFont());
77         TextOutput[a].SetSize(15);
78     }
79     TextOutput[0].SetText(Text[1]);
80     TextOutput[1].SetText(itos(posX + 1));
81     TextOutput[2].SetText(",");
82     TextOutput[3].SetText(itos(posY + 1));
83     TextOutput[4].SetText(Text[3]);
84     TextOutput[5].SetText(Text[2]);
85     TextOutput[6].SetText(itos(level));
86     TextOutput[7].SetText(Text[0]);
87     if(TextPositionSet == false)
88     {
89         TextOutput[0].Move(10.0f, 10.0f);
90         TextOutput[1].Move(150.0f, 10.0f);
91         TextOutput[2].Move(165.0f, 10.0f);
92         TextOutput[3].Move(170.0f, 10.0f);
93         TextOutput[4].Move(10.0f, 580.0f);
94         TextOutput[5].Move(10.0f, 25.0f);
95         TextOutput[6].Move(60.0f, 25.0f);
96         TextOutput[7].Move(10.0f, 40.0f);
97         TextPositionSet = true;
98     }
99 }
100 }
101 std::string Game::itos(const int &integer)
102 {
103     ostringstream oss;
104     oss << integer;
105     return oss.str();
106 }
107 bool Game::CheckPreviousSpells()
108 {
109     std::string compare = "";
110     for(int a = 0; a < 12; a++)
111     {
112         compare += input[a];
113     }
114     for(int a = 0; a < 100; a++)
115     {
116         if(compare == previous[a])
117         {
118             return true;
119         }
120     }

```

```

121     previous[this->rememberedspells] = compare;
122     return false;
123 }
124 void Game::EmptyRememberedSpells(int amount)
125 {
126     for(int a = 0; a < amount; a++)
127     {
128         previous[a] = "";
129     }
130     this->rememberedspells = 0;
131 }
132 void Game::GameUpdate()
133 {
134     //Last setup
135     sf::RenderWindow App(sf::VideoMode(800, 600, 32), "Wizards Maze");
136     sf::Image Image;
137     Image.LoadFromFile("GFX\\RoomS.png");
138     sf::Sprite Sprite;
139     Sprite.SetImage(Image);
140     sf::Image Healthdisplay;
141     Healthdisplay.LoadFromFile("GFX\\h8.png");
142     sf::Sprite Health(Healthdisplay);
143     Health.SetPosition(650.0f, 400.0f);
144     sf::Image Map;
145     Map.LoadFromFile("GFX\\Map.png");
146     sf::Sprite MapDisplay;
147     MapDisplay.SetImage(Map);
148
149     sf::Image PlX;
150     PlX.LoadFromFile("GFX\\x.png");
151     sf::Sprite XDisplay;
152     XDisplay.SetImage(PlX);
153
154     bool wait = false;
155     bool ShowMap = false;
156     //Game Start
157     App.Draw(Sprite);
158     App.Draw(Health);
159     SetText(Player1.GetPlayerX(), Player1.GetPlayerY());
160     for(int a = 0; a < 8; a++)
161     {
162         App.Draw(TextOutput[a]);
163     }
164     App.Display();
165     while(App.IsOpened() && isRunning == true)
166     {
167         sf::Event Event;
168         if(ShowMap == true)
169         {
170             XDisplay.SetX(142.0f +
171 (Player1.GetPlayerX() * 29));
172             XDisplay.SetY(200.0f +
173 (Player1.GetPlayerY() * 29));
174             App.Clear();
175             App.Draw(Sprite);
176             App.Draw(MapDisplay);
177             App.Draw(XDisplay);
178             App.Draw(Health);
179             SetText(Player1.GetPlayerX(),
180 Player1.GetPlayerY());
181             for(int a = 0; a < 8; a++)
182             {
183                 App.Draw(TextOutput[a]);
184             }

```

```

182 App.Display();
183
184 }
185 else if(ShowMap == false && ((GetInputX() == 0) &&
(GetInputY() == 0)))
186 {
187 App.Clear();
188 App.Draw(Sprite);
189 App.Draw(Health);
190 for(int a = 0; a < 8; a++)
191 {
192 App.Draw(TextOutput[a]);
193 }
194 App.Display();
195 }
196 if(!Board.BarrierCheck((Player1.GetPlayerX() +
GetInputX()),(Player1.GetPlayerY() + GetInputY())))
197 {
198 //Invalid Move
199 Text[0] = "Invalid Move";
200 App.Clear();
201 App.Draw(Sprite);
202 App.Draw(Health);
203 SetText(Player1.GetPlayerX(),
Player1.GetPlayerY());
204 for(int a = 0; a < 8; a++)
205 {
206 App.Draw(TextOutput[a]);
207 }
208 App.Display();
209 }
210 else if(!((GetInputX() == 0) && (GetInputY() == 0)))
211 {
212 //Valid Move
213 Player1.SetPlayerX(GetInputX());
214 Player1.SetPlayerY(GetInputY());
215 SetInput(0,0);
216
217 if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'S')
218 {
219 wait = true;
220 int inputgiven = 0;
221
222 Image.LoadFromFile("GFX\\RoomS.png");
223 Sprite.SetImage(Image);
224 App.Clear();
225 App.Draw(Sprite);
226 App.Draw(Health);
227 if(simplified == true)
228 {
229 Text[0] = "Press
space to cast a spell!";
230 }
231 else if(simplified == false)
232 {
233 Text[0] =
Text[4] + itos(12 - inputgiven) + Text[5];
234 }
235 SetText(Player1.GetPlayerX(), Player1.GetPlayerY());
236 for(int a = 0; a < 8; a++)
237 {
238 App.Draw(TextOutput[a]);

```

```
237                                     }
238                                     App.Display();
239                                     while(wait == true)
240                                     {
241
242     while(App.GetEvent(Event) && inputgiven < 12)
243                                     {
244         if((Event.Type == sf::Event::TextEntered) && (simplified == false)) //Works!
245                                     {
246             SetInput((char)Event.Text.Unicode, inputgiven);
247
248             inputgiven++;
249
250             App.Clear();
251
252             App.Draw(Sprite);
253
254             App.Draw(Health);
255
256             Text[0] = Text[4] + itos(12 - inputgiven) + Text[5];
257
258             SetText(Player1.GetPlayerX(), Player1.GetPlayerY());
259
260             for(int a = 0; a < 8; a++)
261             {
262
263                 App.Draw(TextOutput[a]);
264
265             }
266
267             App.Display();
268
269             if(inputgiven == 12)
270             {
271
272                 if(CheckPreviousSpells() == false)
273                 {
274
275                     rememberedspells++;
276
277                     std::pair<int, string> healthandmessage =
Events.ManualSpell(this->input);
278
279                     Player1.SetPlayerHealth(healthandmessage.first);
280
281                     this->Text[0] = healthandmessage.second;
282
283                     wait = false;
284
285                 }
286
287                 else if(CheckPreviousSpells() == true)
288                 {
289
290                     App.Clear();
291
292                     App.Draw(Sprite);
```

```

271                App.Draw(Health);
272                Text[0] = "You have allready used this spell once
today, please try another.";
273                SetText(Player1.GetPlayerX(), Player1.GetPlayerY());
274                for(int a = 0; a < 8; a++)
275                {
276                    App.Draw(TextOutput[a]);
277                }
278                App.Display();
279                inputgiven = 0;
280            }
281        }
282    }
283    else
284    {
285        if((Event.Type == sf::Event::KeyPressed) && (Event.Key.Code == sf::Key::Space) &&
(simplified == true))
286        {
287            std::pair<int, string> healthandmessage = Events.CastSpell();
288            Player1.SetPlayerHealth(healthandmessage.first);
289            this->Text[0] = healthandmessage.second;
290            wait = false;
291        }
292    }
293    else
294    {
295        //Setting the GFX
296        if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'R')
297        {
298            Image.LoadFromFile("GFX\\RoomH.png");
299            Sprite.SetImage(Image);
300        }
301        else
302        {
303            if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'C')
304            {
305                Image.LoadFromFile("GFX\\RoomC.png");
306                Sprite.SetImage(Image);
307            }
308            else
309            {
310                if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'D')
311                {
312                    Image.LoadFromFile("GFX\\RoomD.png");

```



```

309     Sprite.SetImage(Image);
310                                     }
311                                     else
312     if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'W')
313     {
314         Image.LoadFromFile("GFX\\RoomW.png");
315         Sprite.SetImage(Image);
316                                     }
317                                     else
318     if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'T')
319     {
320         Image.LoadFromFile("GFX\\RoomT.png");
321         Sprite.SetImage(Image);
322                                     }
323                                     std::pair<int, string>
324     healthandmessage = Events.RoomEffects(Board.GetRoomType(Player1.GetPlayerX(),
325     Player1.GetPlayerY()), Player1.GetPlayerX(), Player1.GetPlayerY(),
326     Board.GetWinX(), Board.GetWinY(), Player1.GetPlayerHealth());
327     if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'R')
328     {
329         Player1.SetHealingRoomPlayer(healthandmessage.first);
330                                     //cout <<
331         "healthchange = 7" << endl;
332                                     }
333                                     else
334                                     {
335                                     //cout <<
336         "healthchange += " << healthandmessage.first << endl;
337         Player1.SetPlayerHealth(healthandmessage.first);
338                                     }
339         this->Text[0] =
340     healthandmessage.second;
341                                     //cout << "Leaving Other" <<
342     endl;
343                                     }
344                                     //cout << "Player Health: " <<
345     Player1.GetPlayerHealth() << endl;
346     //Board.Displayboard(Player1.GetPlayerX(), Player1.GetPlayerY());
347     if(Player1.GetPlayerHealth() == 1)
348     {
349         Healthdisplay.LoadFromFile("GFX\\h1.png");
350         Health.SetImage(Healthdisplay);
351                                     }
352                                     else if(Player1.GetPlayerHealth() == 2)
353                                     {
354         Healthdisplay.LoadFromFile("GFX\\h2.png");
355         Health.SetImage(Healthdisplay);
356                                     }
357                                     else if(Player1.GetPlayerHealth() == 3)
358                                     {

```

```

349     Healthdisplay.LoadFromFile("GFX\\h3.png");
350
351     Health.SetImage(Healthdisplay);
352         }
353         else if(Player1.GetPlayerHealth() == 4)
354         {
355
356         Healthdisplay.LoadFromFile("GFX\\h4.png");
357
358         Health.SetImage(Healthdisplay);
359         }
360         else if(Player1.GetPlayerHealth() == 5)
361         {
362
363         Healthdisplay.LoadFromFile("GFX\\h5.png");
364
365         Health.SetImage(Healthdisplay);
366         }
367         else if(Player1.GetPlayerHealth() == 6)
368         {
369
370         Healthdisplay.LoadFromFile("GFX\\h6.png");
371
372         Health.SetImage(Healthdisplay);
373         }
374         else if(Player1.GetPlayerHealth() == 7)
375         {
376
377         Healthdisplay.LoadFromFile("GFX\\h7.png");
378
379         Health.SetImage(Healthdisplay);
380         }
381         if(Player1.GetPlayerHealth() > 7)
382         {
383
384         Healthdisplay.LoadFromFile("GFX\\h8.png");
385
386         Health.SetImage(Healthdisplay);
387
388         App.Clear();
389         App.Draw(Sprite);
390         App.Draw(Health);
391         SetText(Player1.GetPlayerX(),
392         Player1.GetPlayerY());
393
394         for(int a = 0; a < 8; a++)
395         {
396             App.Draw(TextOutput[a]);
397         }
398         App.Display();
399
400         //GamestateCheck
401
402         if(Board.GetRoomType(Player1.GetPlayerX(), Player1.GetPlayerY()) == 'W')
403         {
404             ChangeLevelCheck = true;
405             while(ChangeLevelCheck ==
406             true)
407             {
408
409             while(App.GetEvent(Event))
410             {

```

```

395         if((Event.Type == sf::Event::TextEntered) && (Event.Key.Code == sf::Key::Y))
//Next Level
396     {
397         Text[0] = "Press a direction to begin!";
398         ChangeLevelCheck = false;
399         EmptyRememberedSpells(rememberedspells);
400         //Music1.Stop();
401         Player1.SetRandom();
402         //cout << "Continuing to level 2!" << endl;
403         App.Clear();
404         Healthdisplay.LoadFromFile("GFX\\h7.png");
405         Health.SetImage(Healthdisplay);
406         //App.Draw(Sprite);
407         App.Draw(Health);
408         SetText(Player1.GetPlayerX(), Player1.GetPlayerY());
409         for(int a = 0; a < 8; a++)
410         {
411             App.Draw(TextOutput[a]);
412         }
413         App.Display();
414     }
415     else
        if((Event.Type == sf::Event::TextEntered) && (Event.Key.Code == sf::Key::N))
//Quit
416     {
417         ChangeLevelCheck = false;
418         App.Close();
419         //cout << "Quitting game" << endl;
420     }
421     }
422     }
423     }
424     if(Player1.GetPlayerHealth() <= 0)
425     {
426         ChangeLevelCheck = true;
427         Text[0] = "You have been
defeated. Do you wish to retry? (Y to continue, N to quit)";
428         App.Clear();
429         Healthdisplay.LoadFromFile("GFX\\h7.png");
430         Health.SetImage(Healthdisplay);
431         App.Draw(Sprite);

```

```

432                                     App.Draw(Health);
433
434     SetText(Player1.GetPlayerX(), Player1.GetPlayerY());
435                                     for(int a = 0; a < 8; a++)
436                                     {
437
438                                     App.Draw(TextOutput[a]);
439                                     }
440                                     App.Display();
441                                     while(ChangeLevelCheck ==
true)
442                                     {
443                                     while(App.GetEvent(Event))
444                                     {
445                                     if((Event.Type == sf::Event::TextEntered) && (Event.Key.Code == sf::Key::Y))
//Retry
446                                     {
447                                     Text[0] = "Press a direction to begin!";
448                                     EmptyRememberedSpells(rememberedspells);
449                                     ChangeLevelCheck = false;
450                                     Player1.SetRandom();
451                                     App.Clear();
452                                     Healthdisplay.LoadFromFile("GFX\\h7.png");
453                                     Health.SetImage(Healthdisplay);
454                                     //App.Draw(Sprite);
455                                     App.Draw(Health);
456                                     SetText(Player1.GetPlayerX(), Player1.GetPlayerY());
457                                     for(int a = 0; a < 8; a++)
458                                     {
459                                     App.Draw(TextOutput[a]);
460                                     }
461                                     App.Display();
462                                     }
463                                     else
464                                     if((Event.Type == sf::Event::TextEntered) && (Event.Key.Code == sf::Key::N))
//Quit
465                                     {
466                                     ChangeLevelCheck = false;
467                                     App.Close();
468                                     //cout << "Quitting game" << endl;
469                                     }
470                                     }
471                                     }

```

An investigation into the effects on engagement,  
caused by control scheme simplifications in games.

```

469                                     //cout << "player is dead"
    << endl;
470                                     }
471                                     }
472
473                                     while(App.GetEvent(Event))
474                                     {
475                                         if ((Event.Type ==
sf::Event::KeyPressed) && ((Event.Key.Code == sf::Key::Left) || (Event.Key.Code ==
sf::Key::A)))
476                                         {
477                                             ShowMap = false;
478                                             //SetInput(0, -1); //op
479                                             SetInput(-1,0);
480                                         }
481                                         else if ((Event.Type == sf::Event::KeyPressed) &&
((Event.Key.Code == sf::Key::Right) || (Event.Key.Code == sf::Key::D)))
482                                         {
483                                             ShowMap = false;
484                                             //SetInput(0,1); //ned
485                                             SetInput(1,0);
486                                         }
487                                         else if ((Event.Type == sf::Event::KeyPressed) &&
((Event.Key.Code == sf::Key::Down) || (Event.Key.Code == sf::Key::S)))
488                                         {
489                                             ShowMap = false;
490                                             //SetInput(1,0); //Højre
491                                             SetInput(0,1);
492                                         }
493                                         else if ((Event.Type == sf::Event::KeyPressed) &&
((Event.Key.Code == sf::Key::Up || (Event.Key.Code == sf::Key::W)))
494                                         {
495                                             ShowMap = false;
496                                             //SetInput(-1,0); //Venstre
497                                             SetInput(0,-1);
498                                         }
499                                         else if ((Event.Type == sf::Event::KeyPressed) &&
(Event.Key.Code == sf::Key::M))
500                                         {
501                                             if(ShowMap == true)
502                                             {
503                                                 ShowMap = false;
504                                             }
505                                             else if(ShowMap == false)
506                                             {
507                                                 ShowMap = true;
508                                             }
509                                         }
510                                         else if ((Event.Type == sf::Event::KeyPressed) &&
(Event.Key.Code == sf::Key::Escape))
511                                         {
512                                             if(ShowMap == true)
513                                             {
514                                                 ShowMap = false;
515                                             }
516                                             else if(ShowMap == false)
517                                             {
518                                                 App.Close();
519                                             }
520                                         }
521                                     //cout << "Begining new loop" << endl;
522                                     }
523     }
524 }

```

## Appendix.V - Player.cpp

```
1  #include <ctime>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include "Player.h"
5
6
7  Player::Player(void){
8      srand((unsigned)time(0));
9      this->playerHealth = 7;
10     this->playerX = (rand()%9+1);
11     this->playerY = (rand()%9+1);
12 }
13 Player::~Player(void){
14 }
15
16 void Player::SetPlayerHealth(int health)
17 {
18     this->playerHealth += health;
19 }
20 void Player::SetHealingRoomPlayer(int health)
21 {
22     this->playerHealth = health;
23 }
24 void Player::SetPlayerX(int x)
25 {
26     this->playerX += x;
27 }
28 void Player::SetPlayerY(int y)
29 {
30     this->playerY += y;
31 }
32 int Player::GetPlayerHealth()
33 {
34     return this->playerHealth;
35 }
36 int Player::GetPlayerX()
37 {
38     return this->playerX;
39 }
40 int Player::GetPlayerY()
41 {
42     return this->playerY;
43 }
44 void Player::SetRandom()
45 {
46     this->playerHealth = 7;
47     this->playerX = (rand()%9+1);
48     this->playerY = (rand()%9+1);
49 }
```

## Appendix.VI - GameBoard.cpp

```
1  #include <iostream>
2  #include <fstream>
3  #include "GameBoard.h"
4  using namespace std;
5  GameBoard::GameBoard(void){
6  }
7  GameBoard::~GameBoard(void){
8  }
9
```



```
10 void GameBoard::CreateBoard(int level)
11 {
12     ifstream inFile;
13     if(level == 1)
14     {
15         inFile.open("Data\\level.txt");
16         if(!inFile){
17             cerr << "Unable to open the file level.txt";
18             exit(1);
19         }
20     }
21     char x = 'O';
22     for(int i = 0; i < 10; i++)
23     {
24         for(int o = 0; o < 10; o++)
25         {
26             inFile>> x;
27             board[i][o] = x;
28             if(x == 'W')
29             {
30                 this->winX = i;
31                 this->winY = o;
32             }
33         }
34     }
35     inFile.close();
36 }
37 int GameBoard::GetWinX()
38 {
39     return this->winX;
40 }
41 int GameBoard::GetWinY()
42 {
43     return this->winY;
44 }
45
46 bool GameBoard::BarrierCheck(int x, int y)
47 {
48     if((x <= 9 && x >= 0) && (y <= 9 && y >= 0))
49     {
50         return true;
51     }
52     else if(x > 9 || x < 0 || y > 9 || y < 0)
53     {
54         return false;
55     }
56 }
57
58 char GameBoard::GetRoomType(int x, int y)
59 {
60     return board[x][y];
61 }
62
63 void GameBoard::Displayboard(int x, int y)
64 {
65     cout << endl;
66     for(int a = 0; a < 10; a++)
67     {
68         for(int b= 0; b < 10; b++)
69         {
70             if(x == a && y == b)
71             {
72                 cout << "+";
73             }
74         }
75     }
76 }
```

```

74                     else
75                     {
76                         cout << board[a][b];
77                     }
78                 }
79                 cout << endl;
80             }
81 }

```

## Appendix.VII - EventHandler.cpp

```

1  #include "EventHandler.h"
2  #include <string>
3  #include <iostream>
4  #include <fstream>
5  #include <sstream>
6  #include <cmath>
7  #include <algorithm>
8  using namespace std;
9
10 EventHandler::EventHandler(void)
11 {
12     PreloadFiles();
13 }
14 EventHandler::~EventHandler(void){
15 }
16
17 std::pair<int, string> EventHandler::RoomEffects(char room, int pX, int pY, int
wX, int wY, int pHealth)
18 {
19     int diff1 = abs(pX - wX);
20     std::ostringstream oss;
21     switch(room){
22         case 'R':
23             this->printmessage = "A warm magical spell surrounds
you. You feel refreshed.\n";
24             pHealth = 7;
25             break;
26         case 'T':
27             this->printmessage = "A sharp pain strikes you as you
fail to avoid a deadly trap.\n";
28             pHealth = (floor((float)(pHealth/2))) - pHealth;
29             break;
30         case 'D':
31             diff1 += abs(pY - wY);
32             oss << diff1;
33             this->printmessage = "You are within ";
34             this->printmessage += oss.str();
35             this->printmessage += " moves of the exit";
36             pHealth = 0;
37             break;
38         case 'C':
39             cout << "Win XY : (" << wX << "," << wY << ")" <<
endl;
40             cout << "Player XY: (" << pX << "," << pY << ")" <<
endl;
41             pHealth = 0;
42             if(abs(pX - wX) <= abs(pY - wY))
43             {

```

An investigation into the effects on engagement, caused by control scheme simplifications in games.

```

44         if(pX == wX)
45         {
46             if(wY > pY)
47             {
48                 cout << "1" <<
endl;
49                 this->printmessage = "You sense the exit somewhere to the South\n"; //Checked
50             }
51             else if(pY > wY)
52             {
53                 cout << "2" <<
endl;
54                 this->printmessage = "You sense the exit somewhere to the North \n"; //Checked
55             }
56             else if(pX > wX)
57             {
58                 cout << "3" << endl;
59                 this->printmessage = "You
sense the exit somewhere to the West\n";
60             }
61             else if(pX < wX)
62             {
63                 cout << "4" << endl;
64                 this->printmessage = "You
sense the exit somewhere to the East\n";
65             }
66             else if(abs(pX - wX) > abs(pY - wY))
67             {
68                 if(pY == wY)
69                 {
70                     if(wX > pX)
71                     {
72                         cout << "5" <<
endl;
73                         this->printmessage = "You sense the exit somewhere to the East\n"; //Checked
74                     }
75                     else if(pX > wX)
76                     {
77                         cout << "6" <<
endl;
78                         this->printmessage = "You sense the exit somewhere to the West\n"; //Checked
79                     }
80                     else if(pY >= wY)
81                     {
82                         cout << "7" << endl;
83                         this->printmessage = "You sense the exit somewhere to the North\n"; //Checked
84                     }
85                     else if(pY <= wY)
86                     {
87                         cout << "8" << endl;
88                         this->printmessage = "You sense the exit somewhere to the South\n";
89                     }
90                 }
91                 break;
92             }
93         }
94     case 'W':
95 
```

```

96         pHealth = 0;
97         this->printmessage = "You have reached the exit!\nDo
you wish to continue? (y to continue, n to exit)";
98         break;
99     }
100     return make_pair( pHealth, this->printmessage);
101 }
102
103 std::pair<int, string> EventHandler::CastSpell()
104 {
105     int spellword[4];
106     int calc = 0;
107     for(int a = 0; a < 4; a++)
108     {
109         spellword[a] = rand()%17575;
110         if(magicBufs[spellword[a]] == '-')
111         {
112             calc = calc - 1;
113             //cout << "- ";
114         }
115         else if(magicBufs[spellword[a]] == '+')
116         {
117             calc = calc + 1;
118             //cout << "+ ";
119         }
120     }
121     calc = CalculateBuff(calc);
122     this->printmessage = "You cast a spell: " + magicSpells[spellword[0]] + " " +
magicSpells[spellword[1]] + " " + magicSpells[spellword[2]] + " " +
magicSpells[spellword[3]] + "\nThe power of this magic caused the following change
in hp: ";
123     std::ostringstream oss;
124     oss << calc;
125     this->printmessage += oss.str();
126     return make_pair(calc, this->printmessage);
127 }
128 std::pair<int, string> EventHandler::ManualSpell(char inputtedtext[12])
129 {
130     for(int a = 0; a < 12; a++)
131     {
132         inputtedtext[a] = toupper(inputtedtext[a]);
133     }
134     string spellword[4];
135     int calc = 0;
136     spellword[0] += inputtedtext[0]; spellword[0] += inputtedtext[1];
spellword[0] += inputtedtext[2];
137     spellword[1] += inputtedtext[3]; spellword[1] += inputtedtext[4];
spellword[1] += inputtedtext[5];
138     spellword[2] += inputtedtext[6]; spellword[2] += inputtedtext[7];
spellword[2] += inputtedtext[8];
139     spellword[3] += inputtedtext[9]; spellword[3] += inputtedtext[10];
spellword[3] += inputtedtext[11];
140     int spellsfound = 0;
141     int search = 0;
142     for(int a = 0; a < 4; a++)
143     {
144         for(int b = 0; b < 17576; b++)
145         {
146             if(magicSpells[b] == spellword[a])
147             {
148                 //cout << "Spell was found" << endl;
149                 spellsfound++;
150                 if(magicBufs[b] == '-')
151                 {

```

An investigation into the effects on engagement,  
caused by control scheme simplifications in games.

```

152                                     calc = calc - 1;
153                                     }
154                                     else if(magicBufs[b] == '+')
155                                     {
156                                         calc = calc + 1;
157                                     }
158                                     break;
159                                     }
160                                 }
161                            }
162                            calc = CalculateBuff(calc);
163                            this->printmessage = "You cast a spell: " + spellword[0] + " " + spellword[1]
+ " " + spellword[2] + " " + spellword[3] + "\nThe power of this magic caused the
following change in hp: ";
164                            std::ostringstream oss;
165                            oss << calc;
166                            this->printmessage += oss.str();
167                            return make_pair(calc, this->printmessage);
168    }
169
170    int EventHandler::CalculateBuff(int calc)
171    {
172        if(calc == -4)
173        {
174            return -3;
175        }
176        else if(calc == -2)
177        {
178            return -1;
179        }
180        else if(calc == 0)
181        {
182            return 0;
183        }
184        else if(calc == 2)
185        {
186            return 1;
187        }
188        else if(calc == 4)
189        {
190            return 3;
191        }
192    }
193
194    void EventHandler::PreloadFiles()
195    {
196        ifstream infile;
197        infile.open("Data\\spellbook.txt");
198        if(!infile){
199            cerr << "Unable to open the file spellbook.txt";
200            exit(1);
201        }
202        for(int load = 0; load < 17576; load++)
203        {
204            infile >> magicSpells[load];
205        }
206        for(int load = 0; load < 17576; load++)
207        {
208            magicBufs[load] = magicSpells[load].at(4-1);
209            magicSpells[load] = magicSpells[load].substr(0,
magicSpells[load].size()-1); // Removes the buff from all the spells.
210        }
211        infile.close();
212    }

```