LIT – TIPPERARY
ITL–THIOBRAID ÁRANN

Title of Project	Snakes and Dragons 2
Student Name & Number:	Sylwester Szwed – K00231357
Student Name & Number:	Martin Colclough – K00227787
Student Name & Number:	Brayden O'Neill – K00224576
Date	13016
Word Count (excluding appendices)	

Game Design and Development

Limerick Institute of Technology (Tipperary)

Certificate of authorship

We hereby certify that this material, which we now submit for assessment on the programme of study leading to the award of [degree title] is entirely our own work, and has not been taken for the work of others save and to the extent that such work has been cited and acknowledged within the text.

Name of Candidate: ______________________

Signature of Candidate: ______________________

Name of Candidate: ______________________

Signature of Candidate: ______________________

Name of Candidate: ______________________

Signature of Candidate: ______________________

Dated:

Table of contents

List of Figures

Acknowledgements

Our team would love to acknowledge the continuous support and help from those around us. Namely Eugene Kenny for giving us valuable advice and feedback on our game from an outside perspective. For allowing us the space we needed for creative freedom and to take the game in the direction we wanted. Also, for understanding our process of work.

We would like to thank Liam Noonan for providing us with the necessary knowledge that we needed to complete the game. Without his classes we wouldn't get to experience the capabilities SFML has to offer and we wouldn't get to upskill our knowledge of C++.

We would like to thank Jacqueline Humphries for providing us with the information we needed to complete this report. She also made sure all of the students knew how to work as a team and what roles each member had in that team.

We would like to thank Rebecca McKenna for lighting numerous candles in support of our team. We would also like to thank her for pushing us towards our target when we got lazy in the last few weeks.

We would like to thank Meghan Geoghegan for doing most of the art work for the game. She spent many hours producing sprites for our game and we feel like she deserves the credit for them.

We would like Elijah Omotosho for providing us with a sound board for the game. He provided his deep voice for the game which creates a humour aspect to the game.

Abstract

This is a document about Snakes and Dragons 2 which is a game that we made for Project and Group Dynamics module. In this document we have discussed the objectives we had for the game, the features we wanted to implement into the game, how we worked together as a team, the coding language we used and how we implemented it, and the outcome of game when we completed it.

We have done this through the document as we discussed the problems and background, including the project objectives, what problems needed to be solved, what needed to be accomplished, the project timeline and implementation and development methodology.

We have also expanded our aims for the game through the literature review. This gave us an opportunity to learn more about certain aspects of the game that we wanted to include in our game such as utilizing the process of procedural generation to implement random walls throughout the game, board and board game mechanics, and traditional ways of playing new board games. Through the literature review we gave you more of an in-depth look into the different aspects that we have included in our game.

Through the system design and configuration, we have clearly laid out from start to finish. This section is used to inform you of the architecture of the game and how to navigate throughout our game. We have also included an example of use to inform you of all the use cases of each feature that the game has. We have done this in a simple way to make it easy for the reader to understand including the use of diagrams.

We have included a section on testing and implementation to explain clearly how we continuously tested our game while creating the code. We used black box and white box testing to see if the game was working correctly. We have also discussed user friendly interface, activation of pickups, integration testing and debugging.

We have also included a critical analysis to evaluate the success of what we have achieved as opposed to what we originally set out to do. We also discussed what difficulties we encountered while creating this game and we discussed what lessons we learned as a team while creating Snakes and Dragons 2. We critically analysed our own work and work ethic in conjunction with the planned timeline and have laid out everything in a concise language for you to read.

We have included a conclusion to this report to discuss how we well we think that we have done together to complete a successful game of Snakes and Dragons 2 that people will find fun and competitive. We discussed the positives and negatives of our work and our work ethic and we also discussed whether the game has exceeded our expectations or not.

1 Introduction

Snakes and Dragons 2 is a two-player split screen game. The two players are placed into two different mazes and must get the key to escape or kill the other player. The players must try to find gold spread throughout the map. The players are able to spawn snakes and dragons for either himself or the opponent while roaming the map.

The inspiration for the game was taken from multiplayer games like Mario Party and Pummel Party. Those games are multiplayer games which involve a group of people playing a game on a board where some tiles have a unique effect. For example, in Pummel Party one of the tiles can change the direction the player can move. In Snake and Dragons 2 we have a maze where the players are able to move in any direction but have tiles that might move the maze around.

Traditional board games are usually played with many players using the same boardgame. In Snakes and Dragons each player has its own playing board with different mazes in them. The focus of this game was to implement a split screen game where choices on one side of the screen will affect the board and the player on the other side of the screen.

Snakes and Dragons 2 was programmed using C++ coding language by each member of the team. We also used the SFML library to help further the development of the visuals in our game. We have also used Visual Studios to aid us in creating the code for Snakes and Dragons 2.

2 Problem Definition and Background

2.1 Project Objectives

The objective of this project was to create a working split screen maze game where the players placed in two separate mazes and both must do what they can to win. We want do have exclusive features in the game that no other team will be implementing. We want to make the game as fun as possible but we also want the game to have a competitive aspect to it. The game will have to include Snakes and dragons that the player will have to battle. The dragons will be tough monsters that will deal loads of damage to you but you will be rewarded graciously. Snakes on the other hand are small monsters that you can battle for a small reward but you can also unleash the snakes onto your opponent.

Beside that this game has to be made for a college module the team goal is to make the game so good that it makes it to Games Fleadh and bring in the glory it deserves. We want to make a game that will exceed our own expectations when it's completed. The game will have to be multiplayer genre as that is the selling feature that we have promised.

The game will have multiple power ups that differentiate from each another. The power ups are good and bad for the player as they will aid him in winning the game but can also slow him down if the wrong power up is triggered.

One of the big objectives we have is to make the game from scratch. We want to produce new and fresh code as that will test our capabilities of coding and the knowledge of game design. We will try to make the code as easy to read as possible and we will try to make it object oriented when possible.

As we want to make the game original we also want to add original artwork into the game. We will try to draw and design our own menus, characters, icons, power ups visuals. At the end of the day the has to be UNIQUE!

2.2 What problems are to be solved

The game will have loads of problems that will need to be solved. Just to start the game we will need to set up SFML with visual studios which can be complicated and not easy. The SFML library has to be installed and the right extensions have to be added to the folder to run the game. If problems are created along the way, that will only slow down the starting progress of the game.

When visual studios and SFML are compatible the game programming can begin. Creating the player class will not be difficult as we have done it 100 of times. Getting the visuals might be little tricky as we will have to learn how to display the whole scene onto the screen in the right order.

The next thing we will be implementing is the player moving. As we have specific movement type (chess movement) it will difficult to implement. The player can only move one square/tile at a time. In SFML the movement is continuous meaning that if you press the right arrow the player will keep moving continuously until you release the button. In our case the player will only move one square regardless how long you hold down the button.

When the player is moving we will then add walls into our game. We will add a wall generation procedure that will create a maze for us. That will probably be one of the most difficult problems that will need to be solved. The generated maze will have to be designed in such way that it will allow the player to reach the shop to buy the key and then allow the player to progress through the map to the escape.

The objects will be a big aspect of the game as they will help you win the battle. The variety of objects will have to include snakes an dragons as that is name of the game. The objects will have to differ from each other and do different task. The aim of the game of is to have at least 7 different power ups all doing something different in the game. Designing the powerups will not be difficult but implementing their interactions with the player will be a challenge. As all objects will be different they will have to be programmed from scratch and code reuse will be ineffective.

The multiplayer part of the game will have to be carefully implemented as the game will have to be relatively balanced. Allowing two players play at once means you have to handle double the amount of input. Every input into the game will have to be read and compared to the other players input. The players should be able to move simultaneously while the while different parts of they game play out.

Snakes and Dragons 2 will have to be a split screen. Player one will be displayed on the left side of the screen while player two will be displayed on the right side of the screen. The split screen will be implemented in a way that the player can see only few squares around him. The split screen will be accompanied by a HUD which will display the player details including but not limited to player health, coin amount, attack and defence stats and whether the player possesses the key or not.

We are planning to implement random player spawning in the game. The players will spawn at random location every time the game is loaded. We will have to implement this in such a way that the players will not spawn on the maze walls nor will they spawn outside the map parameters. Beside having random player spawning we will have random object spawning implemented in the same way. We will implement that when objects/powerups spawn they will not land on top of each other.

We will have to implement collisions for the player. The collision system will have to handle the player movement and make sure the player stays within the allocated space and not cross over the walls. The team will try to make sure the player doesn’t spawn into the walls when the button is spammed. We have a collision system in mind that we previously implemented in prior assignment. The collision will be considered of true or false Booleans as the player will only be able to move across the true tiles. If the player tries to enter a false tile then will be “pushed” back to his recent tile.

Although there are many problems that will need to be solved in the game we first have to solve problems within our team. As it is our first time working in a team that consists for more than two people we have to learn how to function as a strong working unit. We need to learn how to support each other with every task and every problem we will encounter along the way of making this game.

We will have to work on our interpersonal skills including our communication skills so we can efficiently communicate with each other to get the point across between team members. If we can't get our ideas across we will not reach the same end goal. We need to understand what the other team member is trying to describe to us as we will need to implement it into our game. Miscommunication in the team can also lead to arguments, breakdown of process made in the game, blame and most importantly incompletion of the project.

Our last problem in our team is time management. As all of us work at the weekend we can't find time to work on the project. We have to try to complete our deliverables during the week which leaves other college work neglected. If we start to focus on other college work we leave the game being neglected and get into never ending cycle.

2.3 What is to be accomplished

We need to get a proper set up for each team member. Each one of us should be able to program on separate computer and to do that we need to all have Visual Studios and SFML synced together. We need to gain a deeper knowledge of C++ so we can implement the features we want. As C++ is similar to Java we shouldn't find it difficult to learn so we feel we will be able to progress with the game in no time.

We need to understand that beside the game we have other deadlines for other projects and we can't prioritise the game over them but also, we cannot prioritise the different assignments over the game. We have to be aware of game deadlines and assignment deadlines as not watching the timeline will end up in the team falling behind.

2.4 Project timeline implementation

The team has drafted a Gantt chart that we are hoping to stick by. The Gantt chart shows the various number of tasks that are needed to be completed by the members of the team in order to make a successful game. The Gantt chart has twenty-one sprints that vary in different ways from each other. The size of the sprints may vary but the Gantt chart will give an estimate of how long it should take us to implement each feature into a successful game of Snakes and Dragons 2.

Each team member will be assigned a task that they will have to work on over the week or until completion. At the start of every week the team will meet for an hour meeting to discuss the progress within the game and solve any problem that has arisen. We also hope to help each other on regular basis if a team member is falling behind on their work.

The Gantt chart clearly states what feature has to developed and when that feature has to be delivered by. The features will be implemented weekly and tested all the time as it will fit into our development methodology which will be discussed next.

2.5 Development Methodology

Our team thinks that agile development will be the best option for our project, especially "Rapid Application Development". RAD is a type of agile software development methodology that produces a RAPID prototype of software that then gets realised and perfected along the way.

We chose to go with this development methodology because we think that it will suit our team perfectly. We can develop new features while we work on the ones we already have. Once a feature is developed it will need feedback from the group as well as a 3rd party. As features can change or may be harder to implement they will need more testing and more time to perfect.

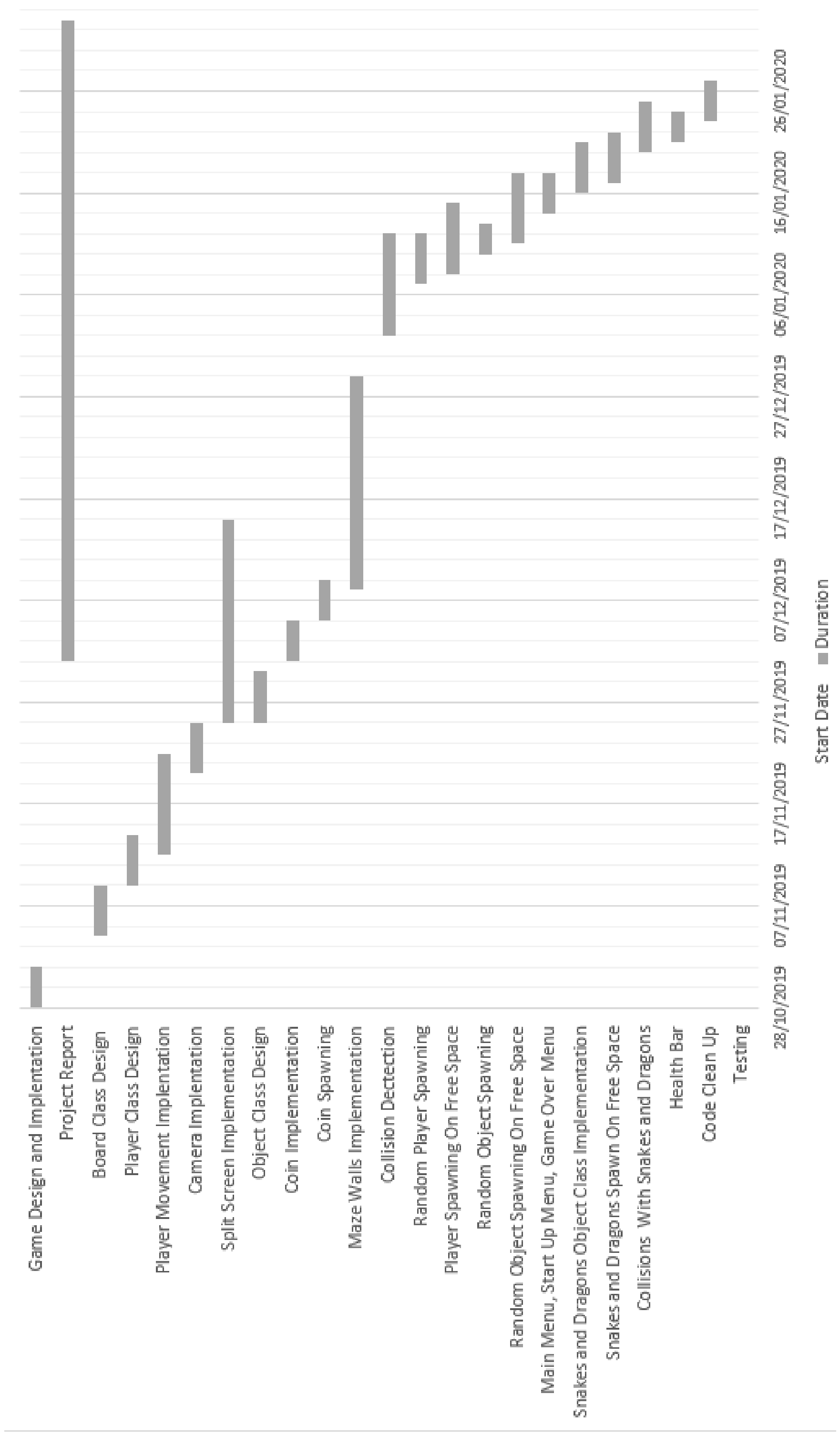


Figure 1.1 Gantt Chart on Project Deliverables

3 Literature Review and Research

3.1 Topic

The topics of our literature review include utilizing the process of procedural generation to implement random walls throughout the game- board, board game mechanics and the new ways of playing traditional board games.

3.2 Introduction

Board games have been around since 3000 BC when Egyptians would play games such as “Senet” [9]. These games were played for leisure by everyone, from the working class all the way to Kings and Queens. Throughout the centuries, new games have come into existence with every region having its own set of unique games… or maybe even the same games as others with a tweak to the rules. Senet is presumably one of the oldest board games in existence, dating all the way back to as far as 3500 BC, followed by the game of Dice in 1400 BC. Even Snakes and Ladders has been in existence for an immense amount of time, dating back as far as 200BC. If you skip about 2000 years traditional games such as Chess or Checkers have made it to the digital world with the first chess computer program being developed in 1957[10]. Since then, countless more games have been seen programmed into the digital world of computing. That is what we are currently working on for our group project. We are aiming to bring a board game concept of snakes and ladders to the digital world in the form of Snakes and Dragons 2.

Snakes and Dragons 2 is an ambitious yet simple maze-game consisting of two players competing against one another in order to secure rightful victory against their opponent. This game will undoubtedly accommodate and abundance of unique and innovative ideas to ensure both an enjoyable and competitive experience for all parties involved. An example of such conceptualizations involves the establishment of intermittent walls which will span throughout the enclosure of the maze. This literature review will explore the various approaches which can be performed in order to achieve this goal.

In conjunction with implementing random walls in Snakes and Dragons 2 we will also discuss the topic of board game mechanics. This research will help to create the core rules of Snakes and Dragons 2 and give the team a greater understanding of how the game should work. The review will explain how mechanics work across other table-top board games and the general rules about them. Information was found for this topic on various articles online as the ones in this review cover a lot of the board game mechanics. However, only the main board game mechanics will be discussed as not all are relevant to Snakes and Dragons 2. We will discuss what mechanics will be used in Snakes and Dragons 2 and what the rules will be for the game.

Traditional board/ table-top games are still as popular now as they were 30 years ago. New games are being developed every year with digital versions coming out not long after. Today's digital world of board games is just as big, if not bigger than the physical world of table-top games. This literature review will explore these two worlds, both how they intertwine with one another and how board games as a whole are moving further and further into the digital world. This study will also compare the difference between Snakes and Dragons 2 to modern digital and non-digital games that are in production. Most importantly this literature review will also examine the multiplayer side of Snakes and Dragons 2.

3.3 Main Body

We will first discuss how to utilize the process of procedural generation to implement random walls throughout the game board. In order to successfully accomplish the goal of constructing random blockades around the board, an appropriate approach would be to take advantage of a process known as 'procedural generation'. However, before this technique can be implemented, it is important to first understand what procedural generation is, as well as what type of procedural generation can prove most beneficial for the game in the long run.

procedural generation is a process used by many modern game developers in order to create vast and unique patterns, antecedently allowing to fabricate virtual worlds that seem more real than ever before. When utilising this technique, an algorithm will be employed to randomly generate geographical locations rather than having a developer manually design static maps. In the case of Snakes and Dragons 2, once the 'move walls' event is triggered, a procedural generation algorithm will be implemented to create a brand new and unique set of walls sprawled across the map. This will combat the initial approach of manually drawing several sets of walls and randomly choosing between them.

There are many benefits of procedural generation. Once subsequently and successfully achieved, procedural generation can lead to endless and unparalleled possibilities, thereby promoting fresh replay ability in a game. This is a very important concept in games development as it can lead to the user wishing to continue playing a game after completing their first playthrough. As well as that, allowing an algorithm to automatically construct scenes can also prove beneficial to the developers as it can spare them from manually doing so themselves, hence leading to a saviour in both company time and resources.

There is a myriad of distinctive examples of procedural generation available which can be utilized in order to achieve the goal mentioned above. However, it is important to first understand which type of procedural generation will prove most efficient as well as most beneficial for this topic. Such examples include but are not limited to: Prim's algorithm and the growing tree algorithm.

Prim's Algorithm is an algorithm that consists of a Minimum Spanning Tree (MST) where its purpose is to connect all vertices together using the minimum number of edges possible. It begins with an empty set that keeps track of vertices already present in the MST. A key must be given to all tiles on the grid and the value of all these keys must be set to infinite except for the first vertex which will be 0. While the set does not contain every vertex, the following process must be carried out:

- Choose a vertex that is not currently in the set that has minimum key value.
- Include this vertex to the set and update the key value of all bordering vertices of this chosen vertex.
- This update is done by integrating through all the bordering vertices and if the total of the first vertex minus the current vertex that is being checked is less the previous vertex that was checked, then update the key as the total of the first vertex minus the current vertex that is being checked.

The Growing Tree Algorithm is very similar to Prim's Algorithm. Here, the algorithm will commence by including a random tile into a list. Next, a random neighbour of the previous tile is selected and added the same list. This random trail continues until it can no longer be extended any further. Once the algorithm finds a cell that has no unvisited neighbours then that square is subsequently removed from said list. After that, it will backtrack to the newest tile and again checking for any unvisited neighbouring tiles. This process is continuously carried out until this list is empty and there are no more squares to check on the board.

In this game, each player will be given the opportunity and ability to freely move throughout the grid in any direction in which they desire, while interacting with unique tiles that trigger different events in the process. For that reason, the Growing Tree Algorithm proves to be the most quintessential approach to generating random patterns of walls throughout the board of Snakes and Dragons 2. This is because it will ensure that most, if not all, tiles can be reached at any moment in time and that no paths will be completely segregated from the rest of the map.

Moving on we will now discuss board game mechanics. Starting with how every board game needs to have a board. There are many types of boards that the board games can be played on. The most basic layout is a normal grid, for example a chess board is an 8X8 grid. Every time you play the game the board is the same layout at the start. But board games can also change every time a new game is to be played. This is called a modular board game, every game in this kind of board game makes each game unique because the layout of the tiles on the map or grid is different and will force the players to make different decisions every time a new game is to be played. An example of a boardgame with a modular board is Star Wars: Imperial Assault.

Movement around the board is also a mechanic that is probably the most important rule in a board game. Different movement mechanisms are applied in different board games. Examples include movement by dice roll which is implemented in the game Monopoly and one move each turn like in chess. There are also rules on how and where a player can move. In chess, for example, the bishop can move anywhere in a diagonal direction and in a game like Snakes and Ladders it is much simpler because the characters can only move in one direction.

A large amount of board games will incorporate a dice roll as a mechanic for the game. This adds an element of randomness to the game. Rolling of dice is used in many ways to make a board game more interesting. The following are some example on how dice rolls are implemented in various board games. In Dungeons and Dragons, dice are used for almost everything in the game. It is used for rolling the players stats like Strength and Constitution, the player needs to roll to check if it hits the enemy and then roll the damage the player hit the enemy with. Dice rolls are also used when players are in conversation with NPC’s to persuade or intimidate them. Other boardgames will use dice as a movement mechanism where the number of the dice determines how many spaces the player moves like in Snakes and Ladders.

Take that is an interesting board game mechanic which means the player tries to put itself into a greater position while at the same time trying to put everyone else at a disadvantage. For example, in the Game of Thrones Board Game the player needs to use tactics to take over most of the board while trying to eliminate other players by making alliances with other players on the board. This makes the board game a lot more competitive and makes it so the player needs to make careful decisions.

The last mechanic that will be discussed is card building which is used in some board games. This is a mechanic where players collect cards to make them more powerful. For example, in the board game Harry Potter: Hogwarts Battle, each player can buy cards from a deck which make them more powerful against the enemies they must defeat to beat the game.

We will now discuss the new ways of playing traditional board games. Most of today's table-top games were published in the 1900s. For example, Snakes and Ladders, which was first published by Milton Bradley in 1943. The game of Snakes and Ladders has gained its popularity throughout the years due to it being a simple game accompanied by elementary rules which can be easily understood by even young children. In most cases, Snakes and Ladders will be one of the first games a child will find themselves playing as it is a great source of grasping some of the basics of Mathematics [11]. As of today, this game can be purchased in 100s of different shapes and forms. Snakes and Dragons 2 will also find itself to be one of these editions. However, it will consist of many never-before implemented features, thereby making the game more compelling and interesting than your orthodox "Snake and Ladders" board game.

One of the most compelling attributes of board games are that they bring people together… the rules, the playing surface and even the tokens all facilitate to positive player interaction [13]. The social interaction that players acquire while playing in the same room or looking at the same board and moving different game pieces is absolutely incomparable to any digital version. In other words, when players share a room, it creates an atmosphere that is just impossible to digitally replicate. You can fight…you can laugh…all the while being in each other's company. When you play a board game across a network, you cannot actually interact with the people you are playing with but rather with the system or the console. You aren't actually moving a player token from one field to another you're telling the computer to do it for you. The barrier between the two worlds is immeasurable, yet we just cannot see it.

"Boardgames, traditionally played in their physical format using boards, cards, dice, playing tokens and the like, are increasingly being translated to digital form for devices such as smartphones, computers, videogame systems and tablets" [12]. But why do we digitalise these games? Well there are several answers to this this question. Firstly, mobility. It is easier for us to carry a game that weighs nothing, is playable anywhere as well as having the ability to play by yourself. Take mobile phones for example. There are hundreds of board games ready to be downloaded… anytime, anywhere. When a game is digitised, for the majority of applications you are given the option to play the game against the computer. This means that you do not require another person to play the game. When we digitise our childhood board games we can either enjoy or hate the new version of that game and the experience it gives us.

A great example of this would be monopoly. This game was published in 1935 on paper/ cardboard with miniscule detail, such as little to no pictures and consisted of only the very fundamental rules of the game. However, it wasn't officially digitised until 1985 when it was released on an Amiga console. In a magazine issue of Amiga Power, the game contained a somewhat basic description of how the game operates on the console. "The transaction is accomplished by picking 'Buy Houses' from the 'Build' menu, moving your cursor to Oxford Street and waiting for £200 to be deducted from your account." [14]. Let's be honest, if a game had a description like that in today's age it would be ridiculed, even though this would have been absolutely necessary 35 years ago. Nowadays, if a game doesn't contain attractive animations, great design, cross-network play or even a reasonable price it won’t be looked at twice.

Look at Snakes and Dragons 2 for example. This is just another version of Snakes and Ladders other than the fact that it is made with new innovative features which are not in any other Snakes and Ladders game. Snakes and Dragons 2 presents new ways of playing board games compared to the old traditional way. In the traditional way, Snakes and Ladders would be played on one board. It simply didn’t matter if the game was digital or not… all players would be positioned on the same playing field. This is in contrary to Snakes and Dragons 2, where the players possess their own playing fields which completely changes the fundamentals of the game.

The game also features split-screen play. This means that two boards will be displayed at the same time, thereby allowing the players to see whether or not they are in danger from their opponent. The game can also be made cross network where each player can only see their own screen and not the oppositions.

3.4 Conclusion

One of the most important components of creating a game is to make it unique… something that will gauge the attention of the people. This can be done in a plethora of ways, be it anything from its vibrant art style, to its smooth and authentic game-mechanics or even just an original and unprecedented game concept. It is absolutely vital that a game holds its own sense of individuality. Otherwise, a failure at launch is imminent, thereby serving testament to importance of originality. In this case, Snakes and Dragons 2’s procedurally generated walls are just one of the countless attention-grabbing features this game has to offer. Every game has the potential to play completely differently and allows for a great use of replay ability so that the players don’t quickly lose interest.

The most important mechanic is the board. Snakes and Dragons 2 will have a 10X10 modular grid board. It will be modular because it’s a maze game and the maze will be different every time a new game is created. The event tiles will also be randomized when a new game is started, or a player stands on the move walls tile which will completely change the maze. The movement around the board will be grid movement. The player can move around the grid as they please but can only go up, down, left and right. They cannot go diagonal as it will not suit the gameplay for navigating around the maze. Rolling dice will be a big part of Snakes and Dragons 2. Dice will be used for player movement (the player moves spaces equal to the dice roll), it will determine who starts by highest number rolled and it will play into the combat. The combat between players will be two dice rolls, an attacker and a defender and the defender loses health based on the attackers roll minus the defenders.

The take that mechanic will also be implemented in Snakes and Dragons 2. The player must make decisions that will benefit themselves and stop the other player from getting more powerful. This is done by the move walls mechanic that will change the maze for the other player and the position of the event tiles. Snakes and Dragons 2 will also have a take on the card building mechanic from board games. So, in card building cards will have items that will make the character more powerful. In Snakes and Dragons 2, the players can buy items such as a sword or a shield from a shop to make them more powerful. The items in Snakes and Dragons 2 can be looked at like cards from a game like Harry Potter: Hogwarts Battle as this board game buys cards from a deck.

The future of traditional board games is changing to a more modern digital world. The games that everyone played together in one room are now available on any electronic device and can be played with family or friends from the other side of the planet. These games don’t require set up or establishing the rules amongst the players. People are now able to jump into a game within seconds. Playing games in the same room will never be as popular as it was 20 years ago. As network play is becoming more popular, we are diving into a more digitalised world. We are subsequently leaving our cardboard platforms we once loved behind, soon to be nothing but figments of our memories. Every year we are developing new ways of playing our old, favourite board games as well improving the various ways we already have done so. This will only continue as we move into a more digital age. Snake and Dragons 2 is an example of how we are moving into a more advanced age of digital board games.

Snake and Dragons 2 can be a basic board game that can be played by people in the same room. We made the game digital, making it more interactive and allowing the game to have more features that would not be possible in its cardboard form. The game will have a board for each player as well as having a board that will have a maze in it requiring the player to navigate throughout the map. The game will have a cut scene for battles not requiring a player input making it more automatic. The game just shows how far the board game technology as came. We started by rolling a dice on a table to now having it done all for us by a computer. We started by dealing out monopoly cash between players to now having the program dividing it between us within few seconds.

4 System Design and Configuration

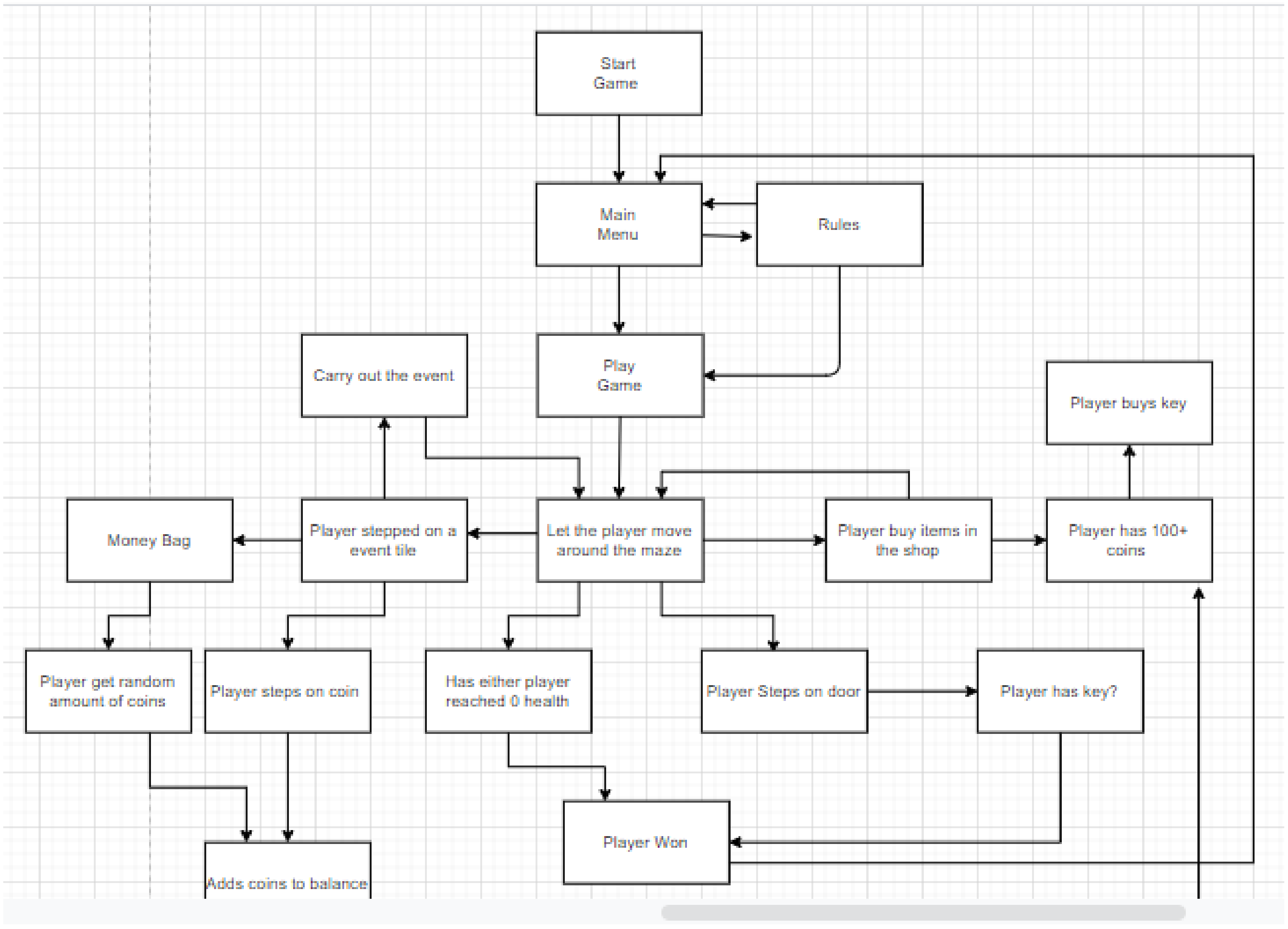


Figure 4.1 Flow Diagram

The above flow diagram explains how the game is laid out from start to finish. Once you launch the application and the game will be started you will be conveyed to the main menu where you can view the rules of the game or launch the multiplayer match.

Once the match is started the players move around the maze collecting coins and triggering different pickups. When player steps on a gold coin 1 coin will be added to his current coin count. If the player steps on a money bag a random amount of coins will be added to his account.

While the player is moving across the maze he can trigger different types of events. Once an event is triggered something will happen within the maze that will affect one of the players. The event can either be friendly and it will benefit you in a certain way or hazardous for the other player.

The game can be won in one of two ways. The player runs out of health due to the damage he has taken from fighting the snakes and dragons or player gathers enough coins to buy the key in the store which he than can take to the exit door.

5 Example of use

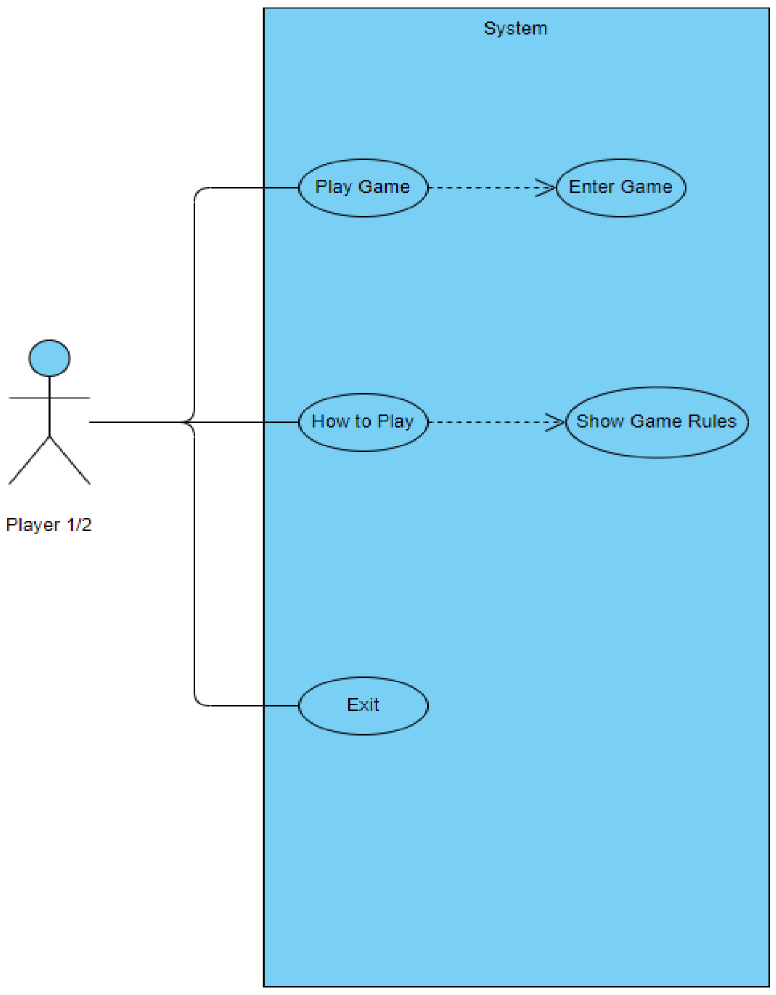


Figure 5.1 Main Use

The menu use case diagram shows the options the players can go through at the start up of the game. If the player chooses play game they will enter the game and can play it from then and on. If the player chooses how to play, the player will be shown the rules of the game and given instructions on how to play the game. If the player chooses exit, the window will close and will need to be restarted to play the game again. Because the game is used on a single computer only one of the players needs to select an option in the menu,

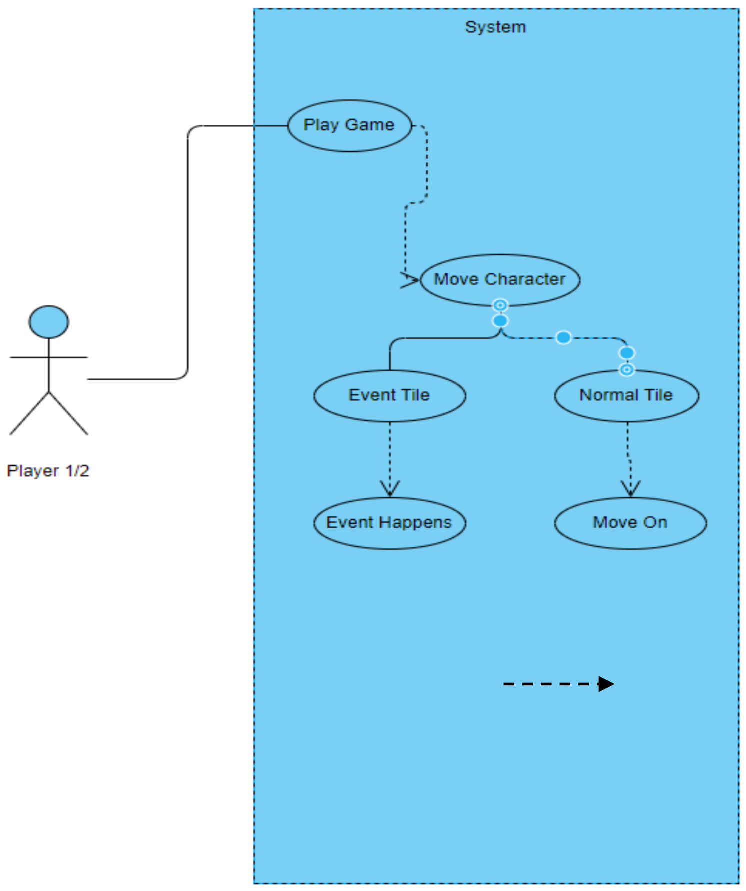


Figure 5.2 Move Use Case

After the player starts the game both players will be allowed to move. Player 1 will be on the left side of the screen and the controls he will use to move are W, A, S and D. On the left side of the screen will be player 2's screen and player 2 moves using the arrow keys. The use case states that when the player is moving around the character, if the player steps on a normal tile nothing happens and the player can just move on. If the player steps on an event tile an event will happen.

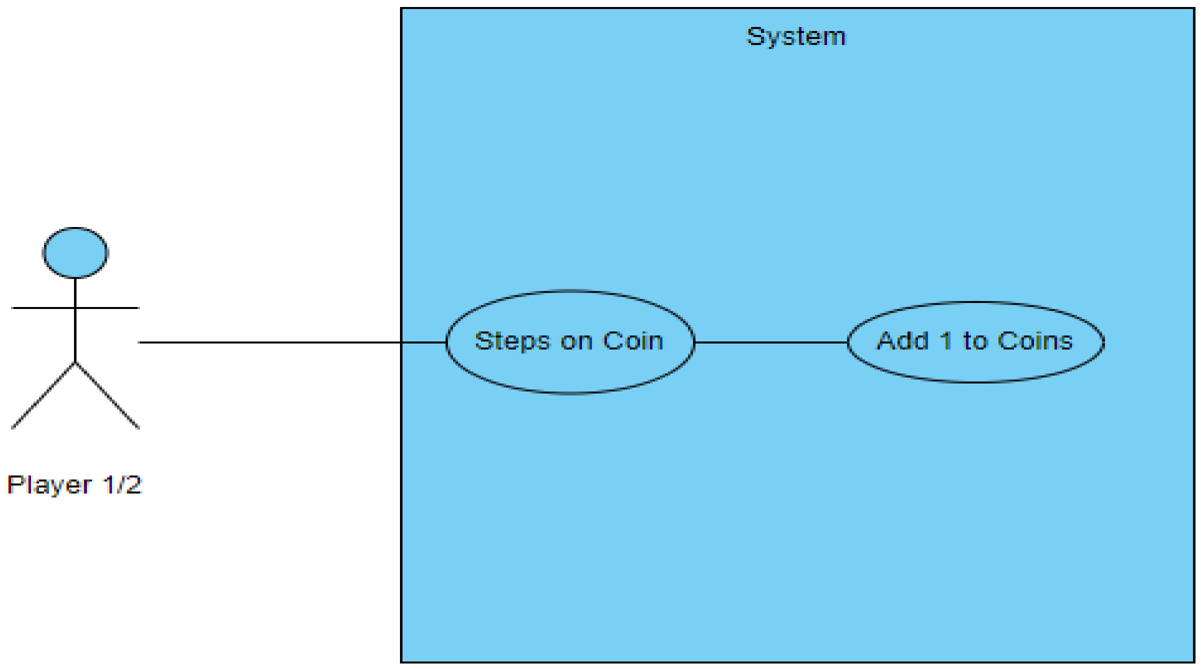


Figure 5.3 Coin Event Use Case

Coins in the game are used for the player to buy from the shop for the player to either upgrade the player or get the key. The use case stated that when a player steps on a tile with a coin on it the system will add a coin value to the players stats.

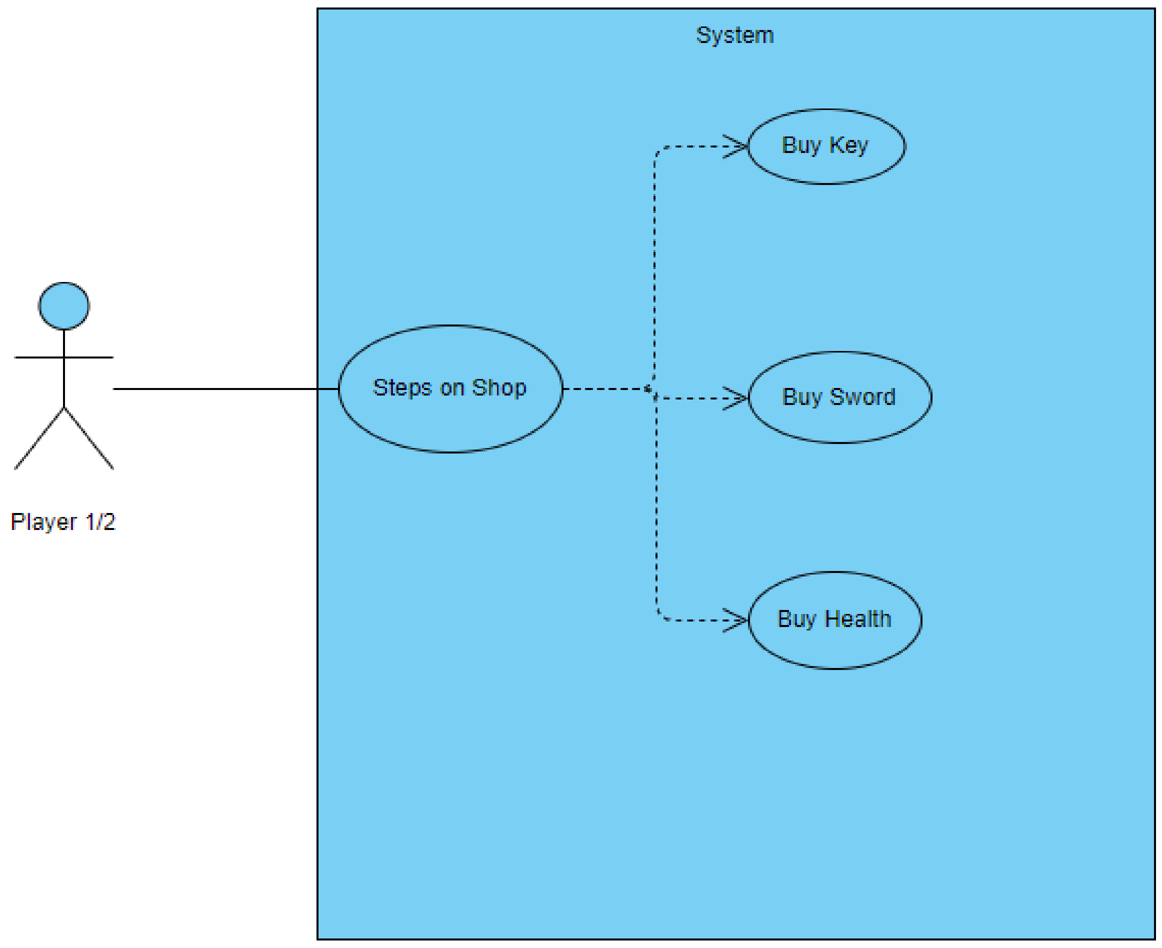


Figure 5.4 Shop Use Case

When playing the game, the players will need to go to the shop to either buy the key to win the game, buy a sword for a greater chance vs. enemies the player may encounter or buy health for future encounters against enemies. Each map will have a shop. The use case diagram states that when a player steps on the shop tile they will be given an option on what they want to buy. They can buy a key, a sword or health. Coins will be used in the shop to buy items.

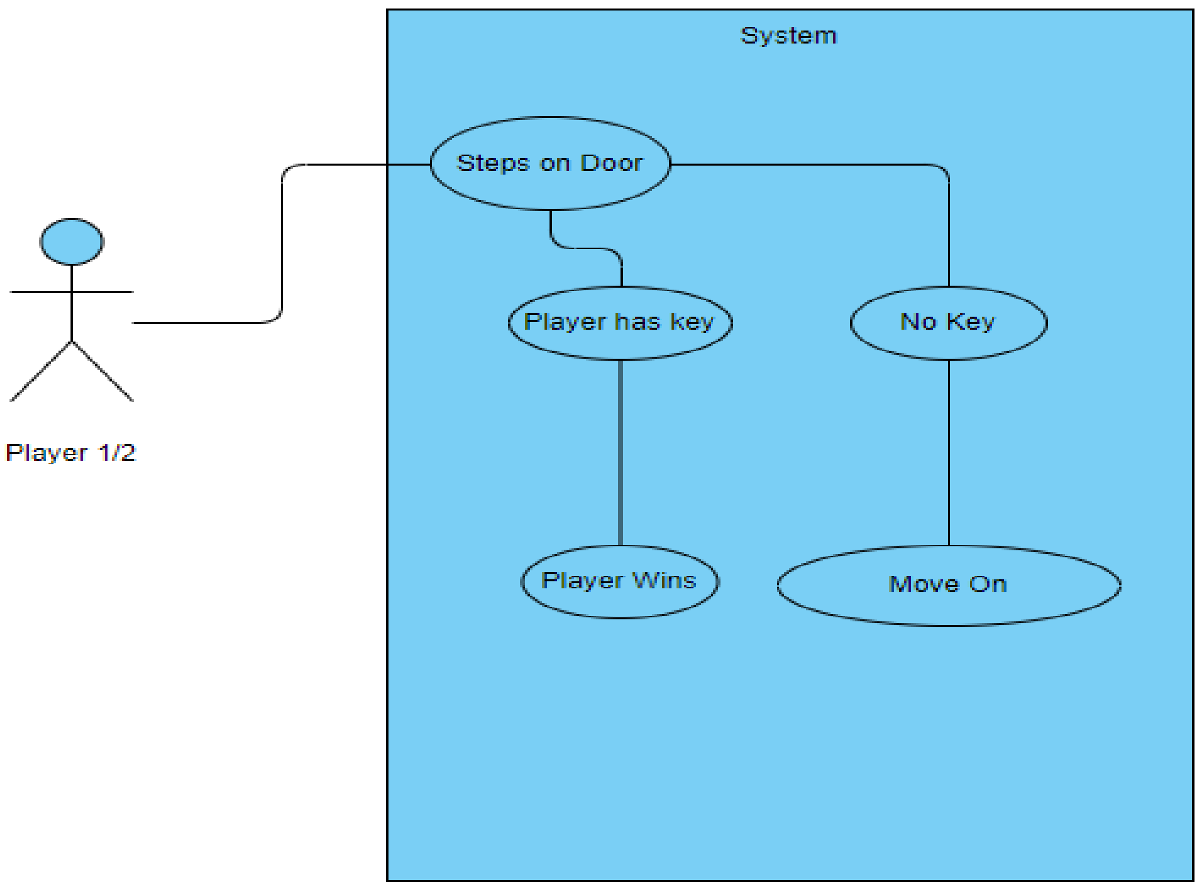


Figure 5.5 Use Case Diagram Explaining Winning Door

For any player to win the game the player must survive longer than the other player or get enough coins to buy the key to open the door. The use case states that when a player steps on the door and the player has the key, that player wins and the game ends. If the player steps on the door and the player has no key the player continues to move and nothing happens when the player steps on that tile.

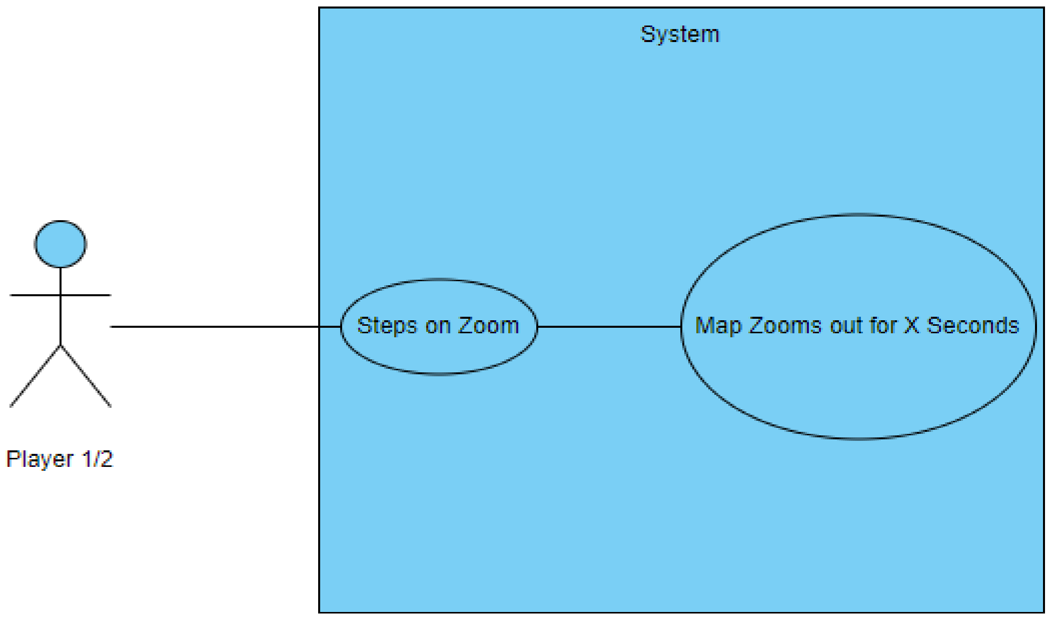


Figure 5.6 Zoom Use Case Diagram

There is a zoom event in Snakes and Dragons 2 which allows the player when they step on a tile too see around their entire map so the player can move while knowing they are going the right direction. The above use case diagram states that when the player steps on the zoom tile the map zooms out for a certain amount of seconds for that player.

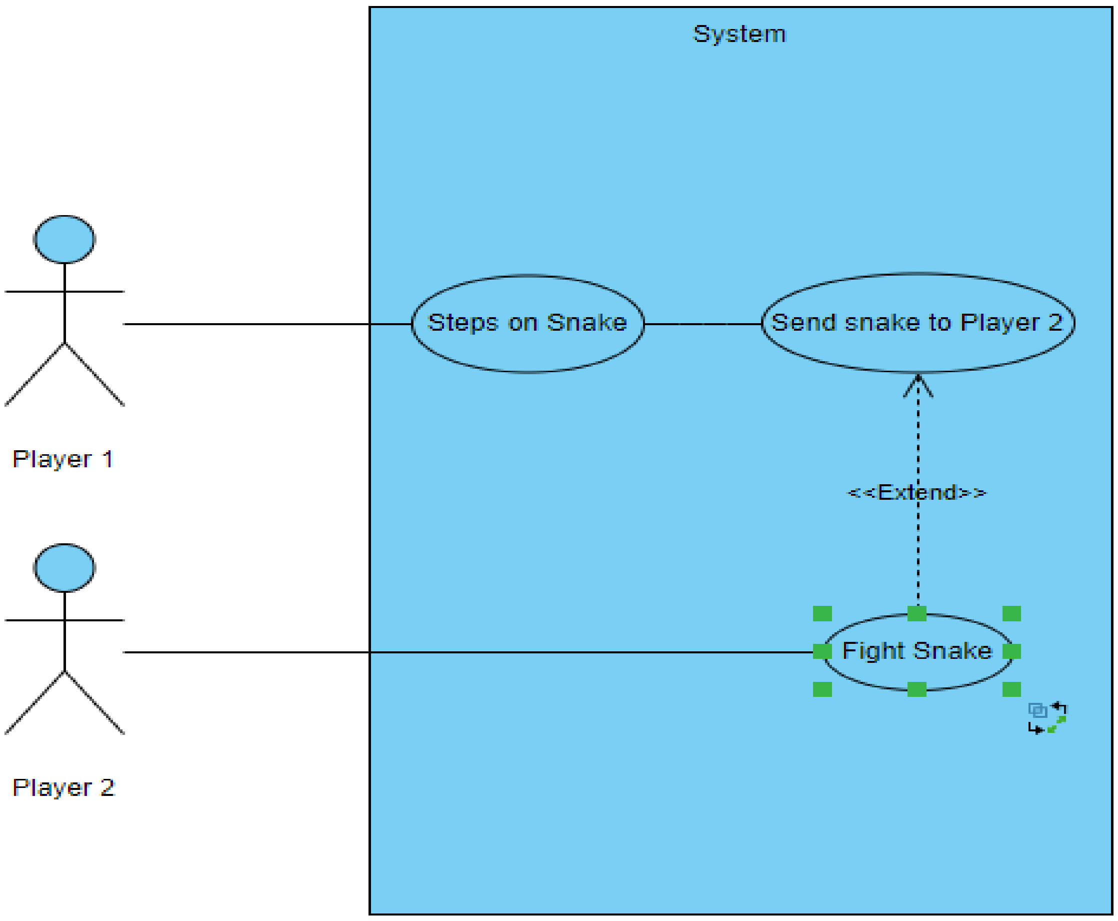


Figure 5.7 Snake Event Use Case

A snake tile will be a help to the player that steps on it because it will send snakes over to the other players maze which then the other player will have to fight it to continue walking through the maze and collect the coins. The use case diagram states that if player 1 steps on the snake tile the system will send snakes to player 2's maze where player 2 will then have to fight the snakes. It is then vice versa if player 2 steps on the snake tile.

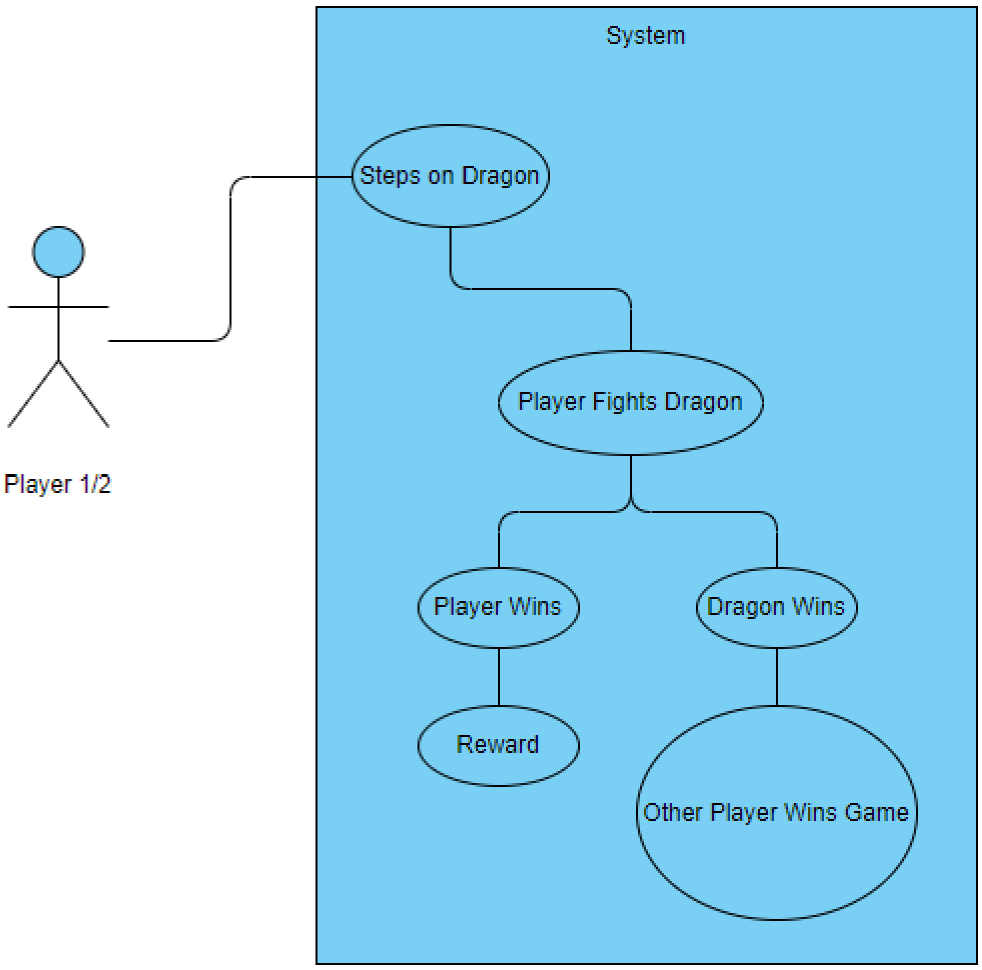


Figure 5.8 Dragon Use Case Diagram

The dragon is the strongest enemy in Snakes and Dragons 2. If the player kills the dragon they will be rewarded, but if they lose they will die and lose the game. The use case diagram states that if either player steps on the dragon tile they will be forced to fight the dragon. If the player wins the dragon will drop a reward but if the dragon wins the fight the opposite player wins.

6 Testing & Implementation

In my opinion, one of the most important parts of Games Development is testing. In order to successfully achieve a fairly balanced and competitive experience for all players involved, vigorous testing was undergone by both ourselves and third-party players. Here, we took the collected data and suggestions into account and used them to implement them into our code as we saw fit. This was undoubtedly an extremely beneficial process as not only did it make our game fairer and better to play in general, it also added to the level of competition between the players, antecedently supplementing the level of fun. Such testing includes, but is not limited to the following:

6.1 Black Box Testing

The purpose of Black Box Testing is to observe the functionality of the games features without looking into the applications code. This was done through the process of allowing third-party individuals to continuously play our game as it is being developed and giving exceptional feedback on a consistent and regular basis. This was a fantastic method of improving the players experience as it helps understand the game's first impressions from a user point of view. Using this method, we took many suggestions into account. For example:

6.1.1 User Friendly Interface

When it comes to the user reading and understanding information, the process of displaying as little data on the screen as possible, while at the same time detailing all necessary information is of vital importance. For the essence of clarity, we implemented a very friendly user interface containing all controls and displaying all needed information in as little words as possible. We found the positive effects of this process as soon as it was added as people spent less time reading and processing information on the screen, allowing for more time to enjoy the game.

6.1.2 6.1.2 Activation of Pickups

When we first implemented pickups that effected the opponent state when triggered, the impact of the pickup would take place without warning for the enemy. We didn't see this as an issue when we were testing the game ourselves as we were already aware of the effects which would take place. However, as we got more people testing our game, it was a common suggestion to add a feature letting the player know what is happening in order to avoid unnecessary confusion. As a result, we added text in the details of the player displaying their current state, such as stunned, rotated etc.

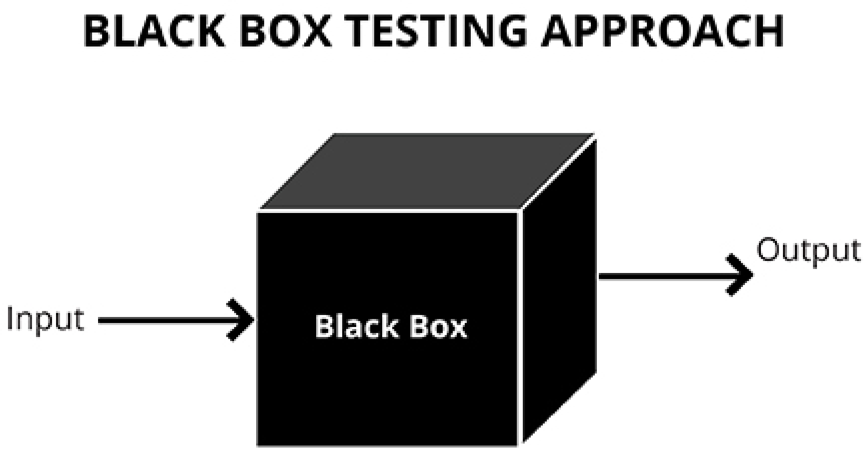


Figure 6.1 Black Box Testing

6.2 White Box Testing

The purpose of White Box Testing (also known as Clear Box testing or Transparent Box testing) is to test the internal design, structure and coding of the application. Unlike Black Box testing, this process is performed by the developers themselves and here the tester can view the code and make observations and changes based on any issues found in the process. Examples of such practices we performed included the following:

6.2.1 Integration Testing

The purpose of Integration testing is to scan the interaction of interfaces with one another. An example of when we used this in our group project was when we pushed and pulled code each other's code to and from GitHub and implemented it into our own files and ensured this new code worked in conjunction with already existing code.

6.2.2 Debugging

One of the most important elements of creating a game is the process of Software Testing. The purpose of Software Testing is not to get rid of all bugs, but to reduce the number of bugs in an application to a reasonably acceptable level. Throughout the process of our game's creation, we performed a myriad of debugging sessions, with the sole purpose of attempting to accomplish a bug-free experience for the player. This included repeatedly playing the game, fixing any issues we found in the along the way. In my opinion, even though some bugs may still be present within our project, we have unquestionably reduced their numbers to an acceptable level.

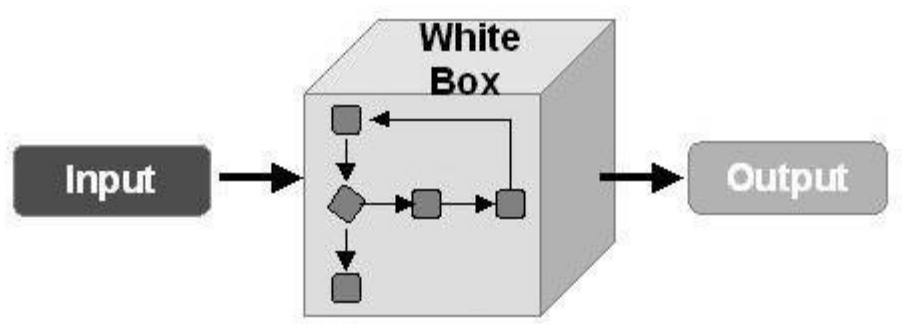


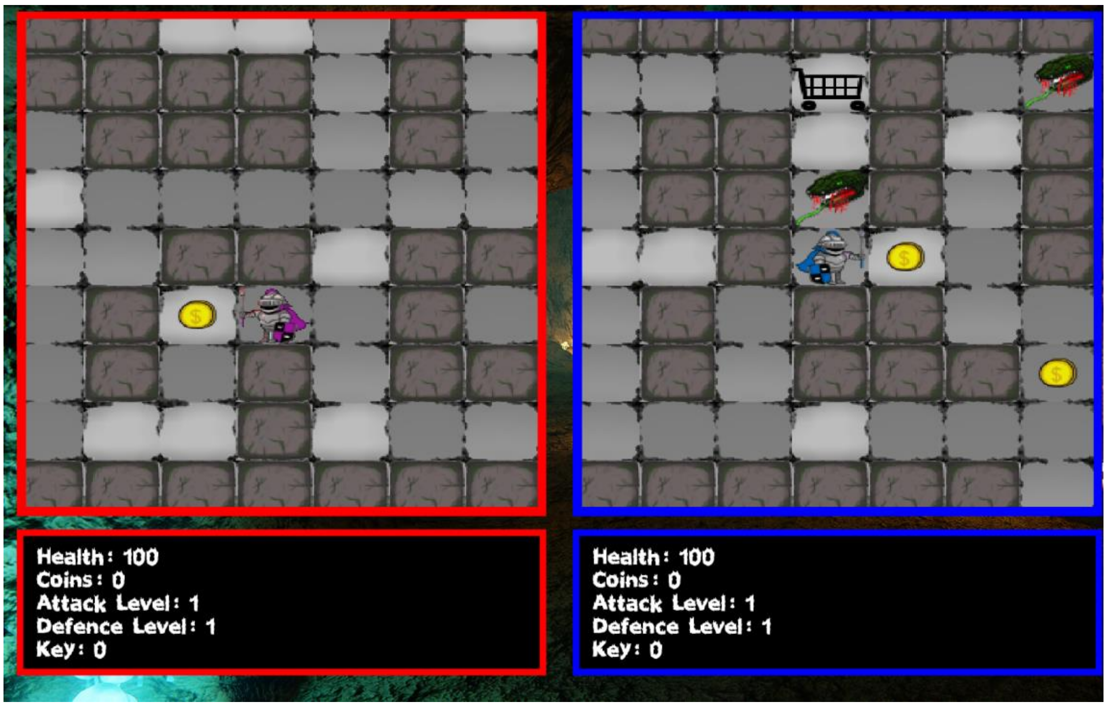
Figure 6.2 White Box Testing

6.3 Reflection of Testing and Implementation

If we were to give an honest and truthful manifestation of our combined contribution of testing and implementation of code, we would most certainly declare our aforementioned testing to be not only a huge success, but also collectively believe it to be a colossal success in the eyes of efficiency.

7 Critical analysis

7.1.1 Working maze game


Health: 100
Coins: 0
Attack Level: 1
Defence Level: 1
Key: 0

Health: 100
Coins: 0
Attack Level: 1
Defence Level: 1
Key: 0

Figure 7.1 Maze Game

We have created a working split screen maze game where the players are randomly placed in two separate mazes. The players have to use the power ups that are placed around the map in random positions so they can out best the opposition in winning the match. You are able to see the visual representation of how the screen looks for in the figure above.

7.1.2 In game features

We have implemented a variety of features into the game. The different features are *quick Sand* which traps you in one spat disabling your movement for 5 seconds, *mouse Trap* which sets a trap on the enemy disabling their movement for a certain amount of time, *rotate/upside down* which rotates the opposite players screen completely for few seconds, *money Bag* which gives you a random amount of coins and *zoom* which increases your field of vision.

Figure 7.2 Money Bag

Figure 7.3 Quick Sand

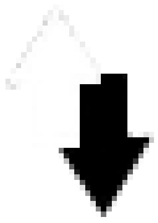


Figure 7.4 Rotate

The above pictures represent the some of the pickups our game has in it. Starting from the left is the money bag, then we have quicksand and lastly, we have rotate/screen flip.

The features we implemented are NOT being used by any other team. We had to make sure that our features are exclusive to everyone else's. For example, one of our early stage features was a visibility dot which only allowed you to see only 2 tiles in every direction. When we found out that other teams were doing it we scraped the feature as it will be basic and that's not how we do things in our team.

We made a game that is very fun to play, it has great visuals and fun audio. This is a game that is very successful with our classmates and they enjoy playing it. We knew it was a good game when we found ourselves playing it for fun instead of to work on the code. The game has a competitive aspect to it as each player has functions at their disposal to hinder the opponent in their quest. We like to think of the game as an argument settling game because it reminds us of call of duty nuketown quick scoping match we all used to enjoy playing when we were younger.

7.1.3 Monsters in game

The game has also monster's that the player will encounter during the game to either help or hinder their play. The snakes and dragons are in place in the game in order to create a battle royale between the player and the game. The player has the options to either fight the snakes and dragons, send them to the opponent or to attempt to leave. The snakes and Dragons have the ability to lower the players health during battle, with the dragon delivering significant damage and the snake delivering some damage.

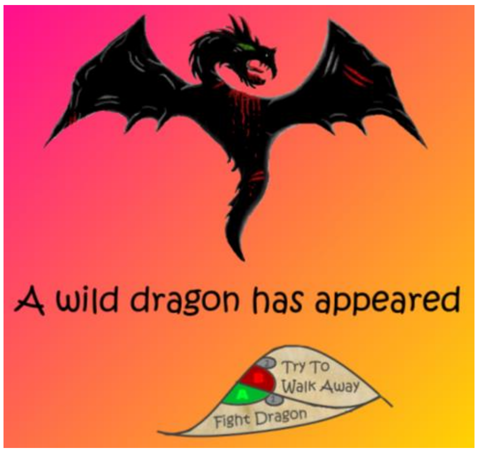


Figure 7.5 Battle with Dragon Screen

The dragon is the most dangerous feature in the game as they have the ability to deliver damage based on their strength and the attack strength of the player. The defence strength of the player also decides how much damage the dragon deals to the player. If the opponent has low health sending a dragon to fight them can be a very strategic way of winning the game. However, upon defeat of the dragon the player wins the battle with the dragon they will be rewarded a large quantity of coins to aid them in their quest.

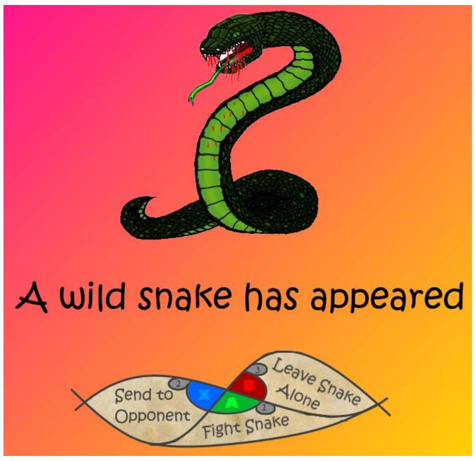


Figure 7.6 Battle with Snake Screen

The snake on the other hand is a small monster and it is not so dangerous to the player. The player has the ability to fight the snake themselves, send them to the opponent or to leave the snake alone. The snake does not deliver the same amount of damage as the dragon but if you encounter enough snakes they can deliver significant damage to the player. If the player has strong attack and defence strengths the snake can do very little damage to the players health. If the player defeats the snake they will be rewarded with a small amount of money.

7.1.4 Completing the set objective

Beside wanting to create a fun and competitive game that people would enjoy, this game was created for the purpose of our college module project and group dynamics. The intended purpose of making this game was to put our theory into practice and create a successful game while also working on our teamwork skills. This also gave us a taste as to what the working world is like and how much work and effort goes into making a game that functions well. The team goal is to have Snakes and Dragons 2 entered into Games Fleadh and to bring home the glory it rightfully deserves. We originally set out to make a game that exceeds our expectations when it was completed. We think that this game has greatly surpassed the expectations we had set out for it.

Snakes and Dragons 2 is a multiplayer game and this is the selling feature of our game. Players have the opportunity to battle it out against each other with one being victorious and the other accepting defeat. The game has many power ups throughout it that differentiates from one another, such as a mouse trap and rotate. Power ups can be very beneficial to the player as they aid them in their quest for the key, such as the money bags. While other power ups can also hinder the player in their mission such as the quick sand, and this can lead to the players defeat.

Our main objective that was set out when we started was that we wanted to make the game from scratch. We used the zombie game that was provided in the game programming module as an example and guideline for where we wanted to go with the game. We wanted to produce fresh and new code and we think that we did that in a way that tested our coding capabilities and practical and theoretical knowledge of game design. We made the code as simple and clear as possible with comments placed throughout the code.

7.1.5 The game art

We really wanted to make a unique game that couldn't be compared to anyone else's based on the visual aspect of the game. We wanted to make an original game that we could be proud of so we added all original artwork and sounds into the game. All the visuals that are in our game are original pieces of art and were created with the intention of being solely for this game.

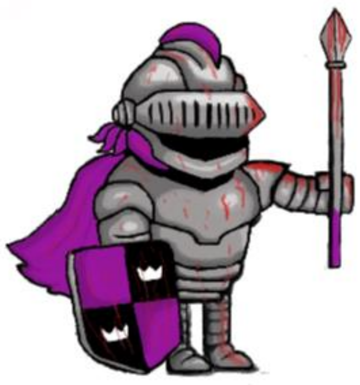
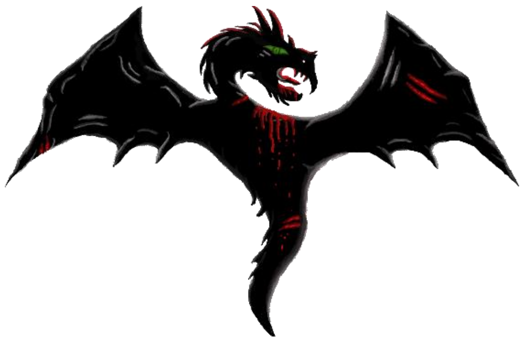


Figure 7.7 Knight Sprite

We created are own player sprites for the game as we tried to keep the art work original. The player is represented as a knight sprite. The knight can be of two different colours. The first player is purple in colour and the second player is blue in colour. The knight sprite has a sword and a shield which can be clearly seen in the figure above.



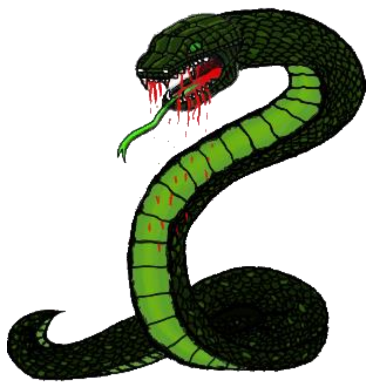


Figure 7.8 Dragon Sprite **Figure 7.9 Snake Sprite**

Original snakes and dragons were created for our game. The monsters were drawn from scratch using Wacom Bamboo touchpad. We made sure that all of available resources were being used. As you can see above the dragon is a 2D drawing that has been made to look scary as hell. There is blood on the dragon giving him post battle effect. The snake on the other side is 3D and is drawn to give the player the feeling that the snake is ready for an attack.

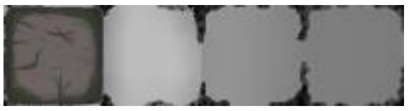


Figure 7.10 Maze Walls and Floor Tiles

The game tiles were also designed to give the game a cave atmosphere. The sprite in the figure above shows 4 different squares. The first square from the left is the sprite for the walls. The other 3 are just different shades of grey to show that the maze is on a chess like field.

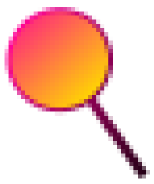


Figure 7.11 Zoom Feature

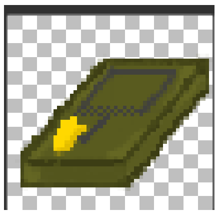


Figure 7.12 Mouse Trap Feature

All of the power ups that are seen in the game were hand drawn to represent different features. We discussed 3 features already in **7.2** but our other two features are zoom power up and mouse trap. The Zoom power up zooms out the camera from the player so the player can see a bigger distance in every direction. if the player is fast enough they can stack the zoom meaning if you pick up two zooms your vision will increase to the first zoom that you collected. The last power up is the mouse trap which simple stuns/traps the opposite player meaning you cannot move from your position for few seconds.

Figure 7.13 Coin Sprite

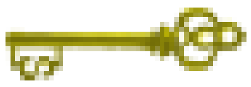


Figure 7.14 Key Sprite

Figure 7.15 Door Sprite

All of these features pictured above are all original art designs that were created for Snakes and Dragons 2. The player needs to collect 100 gold coins in order to buy the key from the shop. The player will then be able to open the door with the key and escape the maze. The player will then have completed the objective of the game.

7.1.6 Game GUI

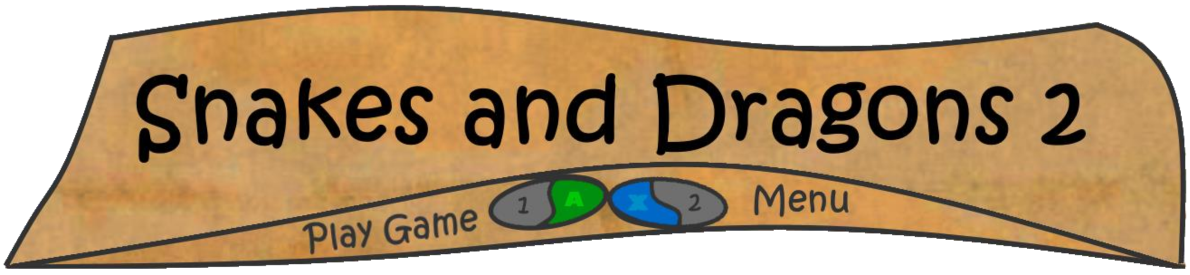


Figure 7.16 Snakes and Dragons Banner

As you launch the game you can already see our own original artwork. The picture above is our banner for our game. The banner will be displayed right as the game launches. As the game supports a controller and keyboard we need to make the GUI user friendly.

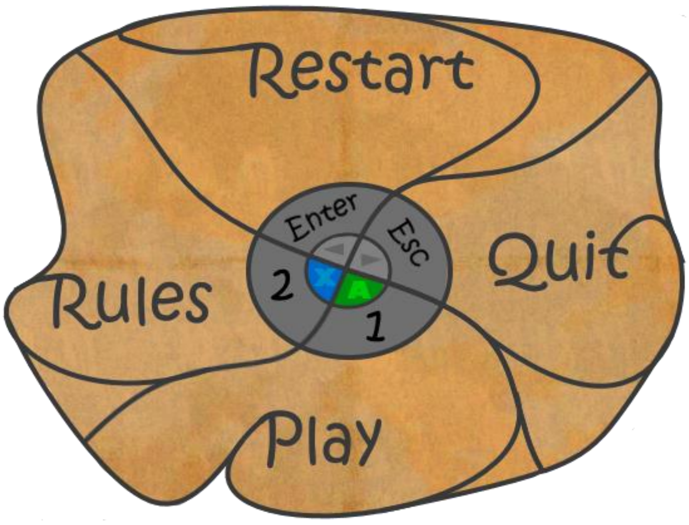
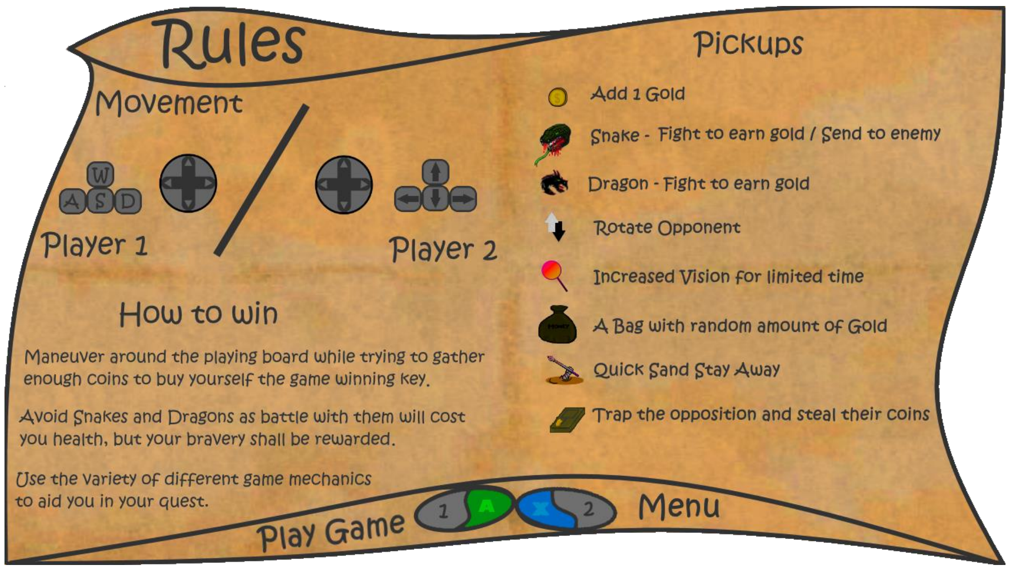


Figure 7.17 Main Menu

After the player presses enter at the start up menu they will be directed to our main menu. Here they have the options to restart the game, quit the game, play the game or to read the rules. This main menu was an original piece of artwork that was created using paint.net. The design of this was to look like crumpled up parchment paper that the player opens to access different things, such as the rules. We have also made this artwork GUI user friendly as the player can see the different buttons they need to use to access the rules or to play the game.



How to win

Maneuver around the playing board while trying to gather enough coins to buy yourself the game winning key.

Avoid Snakes and Dragons as battle with them will cost you health, but your bravery shall be rewarded.

Use the variety of different game mechanics to aid you in your quest.

Add 1 Gold	
Snake - Fight to earn gold / Send to enemy	
Dragon - Fight to earn gold	
Rotate Opponent	
Increased Vision for limited time	
A Bag with random amount of Gold	
Quick Sand Stay Away	
Trap the opposition and steal their coins	

Figure 7.18 Rules Page

The picture above is an original concept that clearly displays the rules of the game. This can be accessed from the main menu as pictured above. The rules clearly states the GUI user friendly movements, how to win the game, and what each of the different pickups are. The artwork was deliberately made to look like an opened parchment that explains everything the player needs to know. This page also shows which buttons the player needs to press to either play the game or to go back to the menu, using either a controller or keyboard.

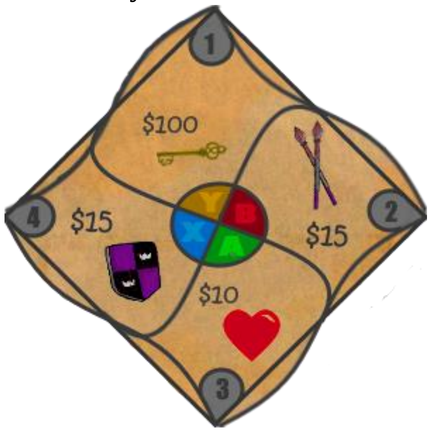
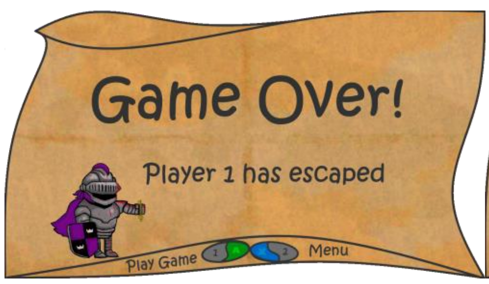


Figure 7.19 Shop

Above is a clearly pictured shop that we have included in our game. This is an original design that was created by our team. In the shop you can buy weapons, health, the key to escape, and defence. The amount each item costs is also on clear display for the player and the controller and keyboard buttons are labelled clearly.



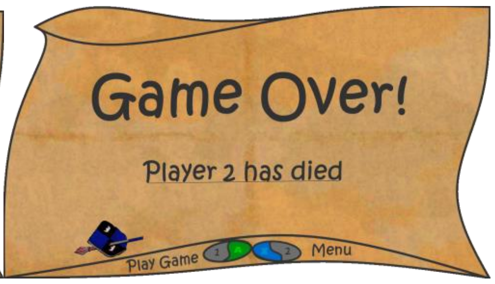


Figure 7.20 Game Over Player 1 **Figure 7.21 Game Over Player 2**

Pictured above is the game over screen, we created multiple different versions of this screen that is dependent on the outcome of the game. The first picture shows the game over screen for if you escape, while the second picture shows the game over screen if you die. We incorporated our sprites of the knights into these screens to make the game over screen more unique. There are also diagrams of what to press for the controller and keyboard users.

7.2 Difficulties encountered

One of the greatest hurdles to face when approaching group project is the difficulties within the group itself on top of all the other complications we experienced. Throughout this project, our team experienced an abundance of such difficulties and we had to tackle them together in order to achieve best results. For example: some of these difficulties included the following:

One of the most difficult complications to deal with was the actual team working together as a group. We found that even though we worked well together when we coding as a group, getting to this point was difficult as it was hard to allocate time to work together while we all had other work / deadlines to do over the weeks and jobs over the weekends. Another problem we encountered as a group was the fact that we were not sticking to the agendas at hand during meetings. This could have been easily avoided if we focused and stuck to the topics at hand.

We experienced a lot of issues with displaying the walls on the board. For example, not reading from the files properly, not being able to print them on the board once they did read from the file, not being able to randomise the file being read for the walls, and finally, not being able to display different sets of walls for each player. We managed to successfully tackles all of these problems except for displaying two different sets of walls for each player. When we attempted to do this, the best we could do was drawing the different sets of walls for each but this would make collisions not work for one of the players. In the end, we decided that it would be best to move on from this topic and just have both players using the same set of walls at any one time so that we have more time to implement more features.

One of the first difficulties we ran into as a team was trying to get the player to move upon each time a button is clicked, moving only the exact size of a tile. Every time we tried to implement this process, the player would either fluidly move instead of moving a certain amount or just not move at all. Thankfully, we eventually overcame this complication and implemented the movement to how we wanted it how to be.

One very annoying complication we ran into during the game's construction, was the fact that we were unable to use certain word styling function such as changing the colour of the words. If this was attempted then the whole game would crash. This was frustrating as we had to stick with white writing even if it didn't suit the game.

Another problem we encountered during the development of our game was the fact that the games HUD was not properly aligned on different screens. This issue was not found until the final few days of the project as we always used the computers in the college. Only when we started using laptops with different screen sizes. The reason this was a problem was because were using fixed pixel measurements in order to properly align all of the HUD elements rather than flexible measurement such as subtracting from the size of the resolution.

At first our game was meant to be a turn-based game where the players were to move based on dice rolls. However, after a lot of testing we decided as a team that it would be of the game's best interest to scrap this idea and make the game a lot more fast paced. In the end, we believe that this was definitely the best course of action as it really added to the competitive nature of the game.

Without doubt, one the most annoying problems we encountered was the error “vector subscript out of range”. This happens when you try to do something to an element that is out of bounds of the vector in question. This was also a very common problem for us as we were dealing with many vectors throughout the entirety of our game.

7.3 What lessons did we learn?

As a team, we learned a lot of lessons whilst working as a group coding a game. This was a new experience for the team which gave us a lot of combined knowledge on how to manage a team and delegate tasks between teammates. The first topics on the lessons we learned is the teamwork component and then the technical lessons learned.

One lesson learned is that the team needs to start as soon as we get the assignment. Once we had our idea, we left it to last minute to do the technical document. After we realised that the document wasn’t where it needed to be, we realised that we needed to start it earlier. We learned throughout doing the project that things need to be done as soon as possible to get the best possible result.

Going with the previous statement we realised that sticking to a good timeline would be more effective than going week-by-week on the work without putting priorities first. If we followed the timeline, we had we would have done all the work without pressuring ourselves and having to stay up all night working on the project.

The timeline would have also helped us to delegate work better between the team. The amount of work done between the team was very inconsistent and involved some people doing more work than others at different times. By not following the timeline and sticking to the plan people were taking over other peoples’ responsibilities.

As a team we could have also stuck more to our meeting schedule (supervisor and team meetings). Before Christmas break the team were attending most meetings and being productive at those meetings. It proved to be very beneficial for the team. After and during Christmas we did not stick to the meetings which led to minimal work being done. We learned not to slack off on meetings as it affects the team's performance in a negative way.

The group also learned what to do to make the team as effective as possible. In general, the team worked well together. Each member listened to each other and knew what roles they had within the team. We let each other voice opinions on what we think about the game and listen to each other just so we know we are on the same page with what we are adding and taking away from the game. A negative of our teamwork is that no one in the team motivated each other at some stages to get work done. We learned that to have good teamwork we need to motivate each other to get things done and we need to help each other when completing tasks.

The communication within the team was good overall. We communicated in college at the meetings and during work sessions together to make sure everyone is on track. We also communicated through messenger on weekends, so we knew what we were doing. The communication sometimes stopped which caused a lack of knowledge on what each person was doing. The team learned that good communication leads to little conflict because everyone knows what they are doing, and it develops a strong relationship with team members.

Now the lessons from the technical component of the project will be discussed. Starting with how we as a group expanded our knowledge about C++ programming and SFML. The group, at the start of the year were just getting back into learning the C++ language and learning a whole new library. We are a lot more comfortable with the code involved as a result of months of work on creating the game. Some of the things we learned are using objects from classes on a different file for code reusability. We learned about the many uses of vectors in which we used for holding the pickup objects which was used for restarting the game by clearing the vector and re-entering them. Using the SFML libraries we realised the effectiveness of game states as it lets us switch between screens in the game. We learned a lot more things along the way with both that will take a lot to discuss.

We also learned how effective clean code and object orientated programming can be. Unfortunately, the team didn't give enough time to clean the code and make it object orientated. We learned as a team that it would have been more efficient to start writing the code object orientated and splitting up files so the code looks a lot nicer and easier to read. The code would be more re-usable and would stop duplication of code.

As a team we wished that we commented more of the code when we were coding. We learned that when we comment code while working as a team, it helps the members understand what the code means that another person implements, it also helps everyone remember what certain variable names mean.

The final thing we learned whilst coding and creating the game is game design and development. We learned how to make a game using C++ with the SFML libraries. The group gained knowledge on how to balance the game so it feels fair for the players against each other and how smooth the gameplay felt. We feel like we achieved invaluable knowledge in game design and development which will undoubtedly help with our future careers.

8 Conclusions

To conclude the report, as a team we are very happy with Snakes and Dragons 2. We feel that we created a very unique game because it is a two player, two board game where certain events on each map can affect what happens to the player on the other screen. Due to the creative thinking of our team and the use of Rapid Application Development we could come up with new ideas while working and try implement them if we feel it is better than other features.

In addition to being a unique multiplayer game, Snakes and Dragons 2 offers original artwork and audio. As a team we decided we wanted all the art in the game to be original, therefore we did not use any sprites from third party websites. We used resources that were available to us to the maximum potential. We got an artist student to draw all of our sprites and load of necessary drawings. The team also decided to make our own audio to make our game as unique as possible. We created our own background sound using beat maker.

Snakes and Dragons 2 has many positive aspects about it. These include Snakes and Dragons 2 being a functional and playable game that players enjoy as it is a very competitive game. We know this because we got people to test our game and received only positive feedback. The feedback indicated that it is a good game for friends to play as we saw the players get really competitive trying to win the game. The game also feels like the game we intended it to feel like but better. It is better because at the very start we wanted player movement by dice roll, but after removing it the game feels a lot better to move around and play the game as a whole.

There were a few negative aspects that we found while coding Snakes and Dragons 2. An example of a negative aspect about the project is our code structure. We decided to try and leave the restructuring the code to near the end. This was not possible because we did not follow a strict timeline and we procrastinated during Christmas where we thought we would get a lot of work done. The game is fully functional with no runtime or syntax errors, but it would have still been nice to re-structure the code to make it a lot more readable.

As a result of completing this assignment, all of the team feel a lot more comfortable coding in C++ and are more familiar with the SFML libraries. We should have started with preparing classes for the code and what exactly was going in them, but at least we now realise this for future projects.

We feel that as a team we completed all of the objectives that we had originally set out upon starting the game. We have also added more features to the game that we thought about while making the code and these have proven to be valuable assets to the functionality of the game. These new features have added to the competitive aspects and make the game more unique and interesting. These functions include quick sand, mouse traps, money bags and rotate. We have also decided which features would suit the game best and any that we felt was too unnecessary we didn’t add. These include the dice roll and player movement by dice roll. We decided not to include the dice roll as it made the game too slow and boring.

At the start of the year Jacqueline asked us what grade we were going to aim for and we simply replied that we just want to “pass”. It is now a few months down the line we can all agree that then answer we provided to her was false and not one bit true. We put in a lot of work into this game trying to make it and we are aiming for the highest grade possible. If she was to ask us this question again we would tell her that we will not settle for anything that is less than 100% as that is how much effort we put into this game!

In Conclusion, Snakes and Dragons 2 is a game that BM Studios is very proud of making and we feel that it has the potential to succeed, with hopefully getting into Games Fleadh. This game has exceeded our expectations and we are very happy with the final outcome of Snakes and Dragons 2. We are happy with the work that we have put into the game and also have new ideas for Snakes and Dragons 2 that will be implemented if our game progresses into Games Fleadh as there are things that we can finesse to make the game more cohesive.

Bibliography

1. Davis, A. M., 1993. *Software Requirements, Revision, Objects, Functions & States.* New Jersey: Prentice Hall PTR.
2. E., G., 1994. *Design Pattern: Elements of Reusable Object Orientated Software.* s.l.:Addison-Wesley.
3. Lake, A., 2011. *Game Programming Gems 8.* 8th ed. Boston: Course Technology, a part of Cengage Learning.
4. Seacord, R., 2005. *Secure Coding In C and C++.* s.l.:Addison-Wesley.
5. Yuzwa, E., 2006. *Game Programming In C++: Start to Finish.* Boston: Charles Media River.

9 References

1. Evens, M. (2019). *What Is Procedural Generation?*. [online] Martindevans.me. Available at: https://martindevans.me/heist-game/2012/11/18/What-Is-Procedural-Generation/ [Accessed 31 Oct. 2019].
2. Tulleken, H. (2019). *Algorithms for making more interesting mazes*. [online] Gamasutra.com. Available at: https://www.gamasutra.com/blogs/HermanTulleken/20161005/282629/Algorithms_for_making_more_interesting_mazes.php [Accessed 31 Oct. 2019].
3. Pcg.wikidot.com. (2019). *Mazes - Procedural Content Generation Wiki*. [online] Available at: http://pcg.wikidot.com/pcg-algorithm:maze [Accessed 31 Oct. 2019].
4. Buck, J. (2019). *Buckblog: Maze Generation: Growing Tree algorithm*. [online] Weblog.jamisbuck.org. Available at: http://weblog.jamisbuck.org/2011/1/27/maze-generation-growing-tree-algorithm [Accessed 31 Oct. 2019].
5. GeeksforGeeks. (2019). *Prim's Minimum Spanning Tree (MST) | Greedy Algo-5 - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/ [Accessed 31 Oct. 2019].
6. Luke, V. (2019). Board Game Mechanics 101: An Introduction To Core Gaming Mechanics. [online] Start Your Meeples. Available at: https://startyourmeeples.com/2018/05/30/board-game-mechanics-101-an-introduction-to-core-gaming-mechanics/ [Accessed 29 Oct. 2019].
7. Boardgamegeek.com. (2019). Browse Board Game Mechanics | BoardGameGeek. [online] Available at: https://boardgamegeek.com/browse/boardgamemechanic [Accessed 29 Oct. 2019].
8. Brandon the Game Dev. (2019). How to Design the Mechanics of Your Board Game. [online] Available at: https://brandonthegamedev.com/how-to-design-the-mechanics-of-your-board-game/ [Accessed 29 Oct. 2019].
9. *A History of Board Games*. [online] Available at: https://web.archive.org/web/20031205225710/http://www.astralcastle.com/games/index.htm [Accessed 29 Oct. 2019].
10. Orr, G. (2019). *The Timeline: Board games*. [online] The Independent. Available at: https://www.independent.co.uk/arts-entertainment/the-timeline-board-games-2346370.html [Accessed 29 Oct. 2019].
11. Arneson, E. (2019). *How to Play the Classic Game Chutes and Ladders*. [online] The Spruce Crafts. Available at:

https://www.thesprucecrafts.com/chutes-and-ladders-snakes-and-ladders-411609 [Accessed 29 Oct. 2019].

12. J. Rogerson, M., Gibbs, M. and Smith, W. (2019). [online] Digra.org. Available at: http://www.digra.org/wp-content/uploads/digital-library/66_Rogerson-etal_Digitising-Boardgames.pdf [Accessed 29 Oct. 2019].

13. Barbara1, J. (2019). [online] Available at: https://s3.amazonaws.com/academia.edu.documents/38786399/Games_and_Culture-2015-Barbara-1555412015593419.pdf?response-content-disposition=inline%3B%20filename%3DMeasuring_User_Experience_in_Multiplayer.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20191029%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20191029T153230Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz Signature=0ac89eee6e5edac24b06edf95e45e184902db5ab1565300424d6b1309b82832b [Accessed 29 Oct. 2019].

14. Archive.org. (2019). *Amiga Power - Issue 03 (1991-07)(Future Publishing)(GB)*. [online] Available at: https://archive.org/stream/Amiga_Power_Issue_03_1991-07_Future_Publishing_GB#page/n71/mode/1up [Accessed 30 Oct. 2019].

10 Appendices

10.1 Sylwester Szwed Reflection

I feel that our group worked well together. We all cooperated with each other, contributed to the production of the game and learned a lot from each other while working on Snakes and Dragons 2.

In particular, I myself have gained invaluable knowledge about coding and I made countless contributions to the game. I have expanded my knowledge of SFML and how it works. I learned more about this library in order to perfect my knowledge on coding language and to aid me in my mission to create Snakes and Dragons 2. I found SFML to be a useful resource to create the game but in my own opinion I found it to be very restricting and limiting as to what I could do with it. I don't see myself using SFML for bigger projects in the future. I gained a more in-depth comprehension of C++ and how to apply it to a successful code. I have found that there are a lot of similarities between C++ and Java but I still have a lot more learning in C++ to do if I want to make it my primary coding language. I have also learned more about cleaning code. I have noticed that using commented code easier to work with when working with a team of people. I learned that separating codes into different files is better as it is easier to find the necessary codes when completing debugging and testing.

I have also learned a lot about teamwork during this project. I have gained new interpersonal skills and have upskilled existing skills that I already have. I learned more about teamwork and how I work as part of a team. I have learned that I like to take charge of tasks and delegate jobs to different people. I have also found that I am willing to listen to everyone's ideas and decide the best one with my team without taking control and choosing the one that I like the best. I have learned that I like to know what is happening within the team, I don't like finding out about changes or problems at the last minute as I find that it hinders the productivity of the group as one singular unit.

In accordance with Dr Belbin’s team roles I have found that I fit quite well into the roles of the coordinator and the implementer. I think that I fit into the role of a coordinator as I am usually the one to delegate tasks to members of the team and I also place a clear focus on the objectives that we had set out for the game from the outset. I think that I am also mature in how I delegate tasks as I ensure to give everyone equal amounts of work and I ensure that I am not leaving any one person with too much to do. I also find that I fit into the role of an implementer as I need to refer back to the plan when I am working on the game to ensure that I have including everything we set out to include in the game. I think that I fit the role of the implementer as I like to take ideas that the team has and ensure that they become a reality and it is possible to include them in the game, while maintaining functionality.

I found that over the course of this project I have upskilled my own personal skills. I have become better at communicating and am more confident when I need to discuss any issues that I have with team members. I have learned how to better phrase my problems with the team as opposed to simply getting angry. I have also become better at communicating praise as it is not normally something I would do. I can better acknowledge when my teammates have done something well and am more equipped to convey my appreciation to them. I have also upskilled my time management skills as they were lacking at the start of the project. Over the course of the project I became more aware of deadlines and created a better running schedule to keep me on track to meet the project deliverables. I have also become more confident in writing reports and how to write them. For this project we had a lot of documentation to do ranging from the proposal to team assessments. Over the course of the project I have gained more writing skills and learned how to finesse a document. I have gathered more information on report writing and am now confident in my abilities to write a satisfying report.

There were also a few problems within the team. I found that for quite a while the team didn’t have their priorities in order when trying to meet the project deliverable deadlines, myself included. I think that during the first few weeks everyone was very focused on the project and creating a successful game but as time wore on everyone let other aspects of their lives take over and became less interested in the game.

Each member of the team works and that effects how we worked on the game when not in college. By the end of the working day everyone is tired and finds it draining to work on the game after doing a full day of work. I, myself have very little time at the weekends to work on the game as I work in the kitchen of a busy hotel. This has also affected my sleeping pattern as I am staying up very late to try and work on the game and am very fatigued because of that.

The team also didn't make the game a top priority in their lives as we all continued to go out and socialise with our friends instead of placing proper focus on the game, whether that was by going on nights out, watching movies or spending too much time with our friends we let the importance of the game sit on the back burner for a while. We got the kick we needed to refocus on the project when we realised how far behind we were than where we had initially planned to be at that time and by mid-January everyone was back on track and the focus was placed back on the game with more passion than we originally had for it. The excitement we had to make an epic game was reignited and everyone put in the work necessary to create the game that we had all envisioned.

I also found that the work division was a bit of an issue within the group. While tasks were delegated and people done them when we had a great focus on the game, it was hard to do so when the game was not everyone's top priority. As I saw less of my teammates I started to do more of the work myself as that way I could ensure that it was being done. I didn't fairly give everyone different tasks and I started on tasks they had been working on as I wanted to ensure that the work on the game moved forward and we could meet the required deadlines. This also put me under a lot more pressure as I was doing a lot of the work on top of working on other assignments.

I had to realise that I wasn't able to do everything on my own and have the game reach its best potential and I needed to delegate tasks more to my team members. I also realised that it wasn't fair on my teammates as I was doing all of the work and not giving them enough to do to be properly involved in the creation of Snakes and Dragons 2. My teammates voiced their concerns to me about their input into the game and told me they wanted more to do and to be in charge of creating more functions and aspects of the game. This allowed me to take a step back and let my teammates take charge of the direction of the game and where it was going. As everyone had their priorities set straight again it was easier to delegate tasks and divide the work up more evenly between myself, Martin and Brayden.

While creating the game I was responsible for many aspects of it. I contributed to over 10 game features in the game which vary from player movement to camera implementation to collisions plus many others. I will now reflect on each one of my contributions and how they impacted the game.

The first contribution I made to the code was player movement. It's something that whole team worked on but I had my input on it. I was the one that implemented Booleans into our movement so every time the player presses the button only the first keystroke will be read in instead of all the keypress. I needed to implement this movement because the game needed a chess like movement.

My next contribution was implementing the game walls including the maze. I created the walls in the game using the notepad. I created a text file with 20 0s or 1s going across and 20 0s or 1s going vertically. I then read in the file and assigned either true or false depending on whether the number was 0 or 1. To do this I created a for loop that created a 2D array of true and false Booleans. I then implemented the wall by using those Booleans and placing walls where every should be.

I contributed in implementing random player movement. I designed the player spawning so that the player will spawn on the map but he will not spawn on the wall. The function I created picks random x and y coordinate and checks whether that square is set to true or false. If the tile is set to true the player will be placed on that field otherwise the function will find new x and y coordinates and check if that tile is set to true. This function will reoccur until a free tile is found.

One feature I am proud of implementing is the split screen feature. I implemented the split screen in a way that the players player view is side by side. Player one is on the left side of the screen and player two is on the right side of the screen. The split screen has a colour border around it to add more aesthetics to the visuals.

The split screen view is set in a way that if player stacks enough zoom features you will be able to see the other players maze map and the other player within it.

The next contribution I added to the game was the camera settings. The camera settings were very difficult to implement as it needed specific calculations to make it look like I wanted. I needed to implement a camera view so that it was centred on the player while the player was between tile 4 and 16 but when the player moved against the walls the camera had to be set to a certain position. Before the camera settings ware implemented the player, platform would move with the player so if the player made it to the corner of the map half of the map view would be outside of the screen.

Another contribution I made to the code was creating the HUD for the game. Snakes and Dragons 2 has a HUD made up of player health, coin count, attack level, defence level and key possession. The HUD is directly under each screen for each one of the players. For player one HUD is under his screen and it is the same for player two. The HUD has a border of the same colour as the player screen border.

One of the features I implemented was the "Zoom" pick up. The *Zoom* pick up is a feature in the game that when stepped upon will increase your field of view. The feature will only last for around 5 seconds but if you can get to another zoom by the time the first one runs out the zoomed-out screen will stay. I had difficulty implementing this feature as the zoom function in SFML is very tricky to get working.

One of my early stage contributions was implementing the coin in the game. I implemented a coin object as the coins will be needed to purchase items from the shop. The coin is really necessary in the game and I had to make sure that the game was fairly balanced so I couldn't add too many coins but I also had to add enough coins so the player wouldn't be looking for them for a long time.

One of my proudest contributions was the player to wall collisions. I used the collision strategy I used in a previous project in game programming which I developed by myself. The collision strategy reads in a text file and created a 2D array of true or false Booleans. If the player moves and he attempts to step onto a tile that is marked as false he will be set back by one tile or 50 pixels. The collisions are made to be so smooth that when you attempt to walk through the wall it just shows you standing in the same position. The collisions ware every hard to implement as there are different collision for each player.

One of my last contributions to the code was creating a second maze so the other player can move around in it. At first, I was going to have the two players on the same map but they would not be able to see each other but then I decide that we promised to deliver two fully working mazes so I had to stick to my word. The first maze starts at the origin of (0,0) and is 1000 pixels in length and height but the second maze starts at (1200,0). The second maze is also 1000 pixels but start 1200 pixels from the left. It was extremely hard to design the second maze as the movement would not work for me but after loads of debugging and testing I finally got the player two being able to move in every direction instead of just horizontally or just vertically.

To conclude my reflection, I can state that I am very happy with my performance and all the dedication I had for this project. I think that I am well capable to work in a team environment and I think that my journey was a total success. I still have many things to improve on but that will come with time and patience. I feel that I exceeded my own expectations when it came to putting in the work and didn't slack too much. I think that I improved my interpersonal skills like I never did before. In my opinion the most important part of this game project was that I had fun with my team.

10.2 Martin Colclough Reflection

In my reflection of this project I will discuss what I have learned as a result of being in a team developing the game, Snakes and Dragons 2. I will also examine how we worked as a team. I intend to inform you about two major problems we had with-in the team and how we managed them. I will discuss what aspects of coding and interpersonal skills that I brought to the team and what parts of the documentation deliverables I completed myself.

First of all, I will inform you of my individual contribution to my team starting with the documentation. In the above documentation, I wrote the literature review about board game mechanics. In the literature review I discussed board mechanics such as the modular grid board, dice and cards in board games and also the take that mechanic. I also made all the use case diagrams for each main feature within the game and gave a description of what each diagram meant. The last thing I included in the documentation is the conclusion where I discussed how the game turned out and what was achieved and failed.

I contributed a lot to the team in the process of making this game such as, the general idea of the game. During the summer, I bought a digital board game called Pummel Party. One day whilst playing the game I thought of the third-year project and an idea was born. The original idea was to make a board game with one board and have two players on it both trying to escape the maze. When I discussed it with my team we thought it was a great idea but added more ideas such as, having it split screen with two different boards. After the idea was accepted, as a team, we brain-stormed all the new features discussed above.

The next contributions I made to my team will be discussed in correlation with my Belbin team roles. After doing the test I figured out that my roles of the team are a team worker, shaper and plant. I am a team worker because I cooperate with the team by doing the most important work required for the project and am willing to help my teammates. I also listen to my teammates which avoids conflict within the team.

Another role I have is a shaper. I have many characteristics of this role including being able thrive under pressure. I know this because when I have a deadline to get something done and I have little time to do it, I can stay motivated until I finally complete the task at hand. One weakness I have with the role of the shaper is that I can sometimes be aggressive with my language while doing the work which could offend my teammates, but luckily my teammates know me and I am not trying to be aggressive.

The final Belbin role I have is that I am a plant. This means that I can have a wide imagination which helps the team in coming up with new innovative ideas. I can also help to solve some problems that the team may be having. The weakness that I bring with the role of the plant is that when I am working, I can become absent-minded and lose focus on the task if I am not under pressure.

Now I will write about the technical contributions that I made to the game. Here is a list of my contributions: board creation, player class, player movement, dice and dice with player movement, shop, obstacles on the board, restarting the game, respawning objects and audio input.

The first thing I did when we started the project was implement the board. As a team we decided to set the board to a 20X20 tile board instead of 10X10 because we realised it was way too small for the game we were trying to make. The outside of the board is surrounded by a wall of tiles that the player couldn't get outside which was implemented before we used the input file of walls.

The second contribution is the player classes, in which I had put in two at the start of making. I then realised that both players used the same rules for movement and spawning so I realised I could make one player class with two player objects. This meant that there was no duplication of code with the player class.

The third contribution is helping code player movement so that the player will move from tile to tile chess movement instead of continuous flow movement like in Pokémon. The team and I figured out that you must move the player by tile size which worked but it went multiple tiles at a time. It was later figured out that to stop this we needed to set a Boolean to set the when a key is pressed to move and change the Boolean back once released.

The next contributions I made was the dice class and have the player move by dice roll. The dice class was meant to be used in the game for various other things such as, battling the snake and dragon, opening a chest and random rewards from certain obstacles. The only thing I implemented in correlation with the dice roll was the player movement. I did this by rolling the dice and add a counter whenever each player pressed a movement button to decrease the counter until the counter hit 0. The next player then got a turn. This feature was later scrapped during a team discussion where we all agreed that the game was much too slow for what we wanted so we later made the players have unlimited movement, so the game feels much more like a race instead.

The shop is another contribution I added. The shop allowed each player to buy a key, shield, sword or health potion. The key wins the player the game, the shield makes the player take less damage from enemies, the attack increases the power of powerups, and the health potion buys the player more health. To implement the shop on the map I made a shop screen when the player stepped on the tile and it disappear when the player stepped off the tile. The player buys from the shop with the buttons displayed on the menu where then coins will be taken from them.

I helped come up with the ideas for the event obstacles on the map. The obstacle I created was the trap obstacle. When player steps on the trap event the other player loses coins and is stunned for a certain amount of seconds. This was done by when one player steps on the trap pickup, I disabled the movement of the player and removed coins from the player. This slows the player from winning the game.

I came up with how to respawn the objects when they are picked up. The first thing I did when doing this was to create a for loop to respawn the object when it was less than what was in the vector. I later realised it was a better idea to use one line of code that spawned an object each time it is picked up (when the player intersects with the object).

I also figured out how to restart the game when it is in two states. For a while in the game, when I would press a button to restart the game, the game will be state it was previously in (the objects stayed in the same place and the player kept spawning in the top left of the screen. To fix this, when either player wins in the game, I set both game stated to game over and deleted every object from the vector and re added them right after. When a player exits the game over screen they are then put back to the menu screen where the player can now start a new game.

The audio I implemented is all original. There is audio for background music, player pickups and the game over screen. The audio's paste and sound needed to suit the genre and setting of the game.

The next thing I will discuss is what I have learned throughout the whole experience of making a game with a group of three people. The main thing I learned about is the language C++ in the use with the SFML libraries. At the start of the year I considered myself a weak C++ programmer. But along with the Game Programming module and working with a team on the game, I expanded my knowledge of C++ and using different classes for different functions. It was difficult for me to get the grasp of SFML at the start, but with help from the Game Programming modules and my teammates I am more comfortable when using the library.

Object-Oriented programming is also a very useful when coding as it adds code reusability. As a team we were meant to insert object-oriented programming when we were finished coding. We realised it was too late to do this and had to scrap that plan. The code would be much easier to read if we added OOP, but unfortunately, we were too late.

I learned how to work in a team more efficiently and effectively to finish the game by the deadline. I think we worked decent as a team as we didn’t have much arguments and eventually came to be on the same page and completed good work as a team without messing up with GitHub uploads. We also had a clear idea of what we were doing with the game very early on.

The final thing I learned is communication is very effective within a team. At the start of doing the project, as a team I thought we didn’t communicate very well. This led to starting deliverables to not be high quality. But when we started having meetings with each other and our supervisor, we got a lot of work done to get back on track.

The last thing I will talk about is some of the problems I felt we had as a group. In particular, our lack of time management and our sections of procrastination.

I found that the main problem we had with our team is time management. When we started writing up the technical proposal for our project, we waited until the night before to complete it. This led to a result of a bad proposal as we underestimated the time we needed to complete it. We also underestimated the time we needed to add more features which led to some features being left out. Another reason time management was a problem is because we did not delegate time to the project properly when we had other assignments which meant all night coding sessions needed to be done to get work done that needed to be done. In the future when doing projects, I now know not to leave things to last minute when doing big projects and focus more on work rather than social life.

Another problem we had as a team in my opinion is procrastination. As I said with time management, we left things to last minute. But eventually, we copped on and realised we had to get work done after the presentation. By the time it came to Christmas break, we had a plan but none of the team stuck to the plan because we kept putting it off until eventually it came back to January and we needed to get work done. As a result of this, some features needed to be left out and we needed to overwork ourselves.

Overall, I am happy with the way our project turned out. We implemented most of the features we wanted including the main features such as the split screen and the interactions between the two players on different maps. I also increased my knowledge on C++ and SFML which will be beneficial in the future life. I learned from my mistakes of not managing my time properly, doing too much procrastination and not starting the code object oriented as we left it too late to implement.

10.3 Brayden O' Neill Reflection

Looking back on our team's contribution and work ethic, I believe that we all pulled our weight and definitely put in as much work as we possibly could. Not only did we perform well in our individual sections of the project, but we also managed to work as a team to accomplish a great final product. At the beginning of this project, I believe we began this project with a limited knowledge of C++ and only a fundamental understanding of the SFML library, but as we progressed towards the end, we definitely finished with a decent comprehension of the language as a whole. If I was to give my own opinion about the languages we used for this project, I would most certainly say that C++ is a very nice language to utilise with SFML to create a fully functional game. I believe that it is a very elegant language and through the use of header files, .cpp files, object-oriented techniques and reusable code, we were able to ensure that we were dealing with code that was not only easy to read and understand, but also easy to manage.

As for the teamwork portion of the project, I believe that our team operated very well together once we got down and worked. At the beginning of the year, we held multiple group meetings in which we didn't yet discuss the code, but together came up with the original idea of the game and all gave our personal inputs and suggestions to incorporate to ensure a fun and competitive experience. When it came to coming up with ideas for the game, we all discussed what we would like to see from in the final product. From here, we then discussed with one another any changes we would like to add to each other's ideas. This ensured that we all were on the same board and really helped tie the game together as a whole. In my opinion, we came up with a great concept for a game in the end, filled with all our original ideas. Once we believed we were finished discussing the project, we then set about coding. Here, I believe that once we all sat down and began to code, we worked nicely as a group. Not only do we all utilise similar coding techniques, but we also had the same idea of how the code should be structured. This in conjunction with having the same vision for the final product, really eased the level of difficulty within the project as we all on the same page. This really showed me the importance of good communication and interdependence within a group as you should always listen to other members' input while at the same time giving your own.

In our group, I believe we did a good job at equally delegating jobs and distributing the work to one another. In my opinion, I ended up contributing an abundance of unique features and elements to the game. Some examples of the additions I put in the group project included the following:

- **Controller Support** – One feature I included in which I believe really added to the quality and immersion of the game was the addition of controller support. Originally, the game was intended to be played on a single keyboard shared by both players. Here, player 1 would use the WASD keys to move and numbers 1-4 to interact with the game whereas player 2 would use the arrow keys to move and number 1-4 on the numpad to interact with the game. I then added the ability for players to have the ability to use controllers if they wanted to. This was done by making every single possible input also be viable on a game controller. Once this was successfully implemented, the game could be played on keyboard, controllers, or both.
- **Opening Screen** – When the game was first opened up, the players were presented with a very dull start-up screen which consisted of the name of the game in normal text as well as a prompt to press enter. I then decided to change this to an interactive screen where the title of the game existed within a nice UI which suited the theme of the game and made it move around the screen in a windows-screensaver styled fashion. In my opinion this really added to the level of immersion and professionalism we wanted to ensure within our game as it showed unique features from the rest of the class from the get-go.
- **Random Spawning** – In our game, it was vital to ensure that all players and pickups spawned randomly throughout the maze. This was done by creating a simple method which dealt with randomising X and Y positions, checking if these positions aligned nicely with the tiles on the map, and then placing them on that tile. This was a very beneficial method to have in our code as it was reused many times, every time something was spawned on the map. However,

when I first made this method, it did not check for collision and so spawned events and players on top of walls. This was fixed later in the game by another member of the group when collisions were successfully implemented.

- **Moving from Game States –** In our project, we included enums which dealt with checking the current state of the game and displaying information on the screen depending on the current state. I believe that this was a very easy process to implement as all that had to be done was change the current state of a player if a certain condition was met such as input or position of the player.
- **Snake Pickup –** The first pickup in which I implemented to the game was the addition of snakes. If a player landed on a dragon, A combat screen would appear on the screen prompting the user to either fight the snake, send the snake to the enemy player, or run away. If the player fought the snake and killed it without dying, the player would return to the game board with a reward of a small amount of gold.
- **Dragon Pickup –** Similar to the snake pickup, if a player landed on a dragon pickup**,** they would be presented with a combat screen prompting the user to either fight the dragon or attempt to flee. The difference between the dragon and the snake is that a player cannot send a dragon to their opponent and the dragon does a lot more damage (but also rewards a lot more gold).
- **Rotate Pickup –** One of the first offensive pickups I put in to the game was the addition of the rotation pickup. Here, if a player landed on a rotation, then the enemies screen would be turned upside down for a brief period of time, depending on the attack and defence levels of the players.
- **Slow Pickup –** If the player lands on a slow pickup, then they would be stunned for a brief period of time, again dependent on the attack and defence levels of the players.
- **Slow Attack Pickup –** Here, I wanted to include as many pickups that influenced the opponents experience and so I added a slowAttack Pickup. This worked the same as the slow pickup but instead made the opponent be stunned for a brief period pf time based on the players' attack and defence levels.

- **Ink Pickup –** When the player lands on an ink pickup, it would make most of the enemies' screen black for a short period of time, limiting their view and disallowing them to judge where they are going.
- **Bug Fixes –** Throughout the creation of the game we were constantly testing the code as a team and fixing any bugs we encountered along the way. This is a great example of agile development and in my opinion, works very well and helps add features in a quick and efficient manner.
- **Win conditions –** Another addition I made to the game was the presence of win conditions. There are several ways in which a player can win the game. For example, if a player saves up enough coins to buy the key from the shop, and they return to the exit, they can escape the maze and win the game. Another way of winning the game is by getting the opponent to 0 health.

Even though I believe we worked well as a group, there are always problems that can arise within a team. At first, our team made a great effort in contributing ideas for the group and really creating a vision for the game but very quickly started to lose motivation as deadlines for other modules approached. If I was given the chance to start the project again, I believe that better time management would be extremely beneficial and would really ease some of the pressure we experienced as the deadline for the game approached. That being said however, once we all eventually sat down and began coding, I believe that we worked extremely efficiently together; designing, testing and implementing new features at the same time.

If I was to summarise my experience of the group project, I would say that I not only had a lot of fun in the process, but also learned a lot along the way. I learned of the importance of good communication within a group, voicing your opinions and being open to as much feedback as possible. This really helps in the production of a team project, allows all members to make an impact and properly create the perfect game. On top of that, I believe that I learned a lot about the different techniques which can be utilised in C++ and obtained a good grasp of the SFML library and how to properly display the game on the screen along with a graphical user interface. Finally, I found that experiencing GitHub as part of a team and seeing how the code is being continuously updated was really fascinating and I believe this will prove very beneficial in a professional sense once we get work experience next semester.