LIT – TIPPERARY
ITL–THIOBRAID ÁRANN

Title of Project	Fractured Memories
Student Name & Number:	Thomas Jones K00232078
Student Name & Number:	Jordan O Loughlin K00225361
Student Name & Number:	Elijah Omotosho K00227425
Date	
Word Count (excluding appendices)	

Games 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 Games Design and Development 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: Thomas Jones

Signature of Candidate: _____

Name of Candidate: Jordan O Loughlin

Signature of Candidate: _____

Name of Candidate: Elijah Omotosho

Signature of Candidate: _____

Dated:

Table of contents

Acknowledgements

The team as a whole: Jordan O Loughlin, Thomas Jones, and Elijah Omotosho.

Those who aided in the project with art: Meghan Geoghegan, and Luka Wall

The source creator of the Collision Detection that was implemented: https://github.com/SonarSystems/SFML-Box2D-Tutorials

1 Introduction

1. PURPOSE

The goal of the project is to provide a short adventure styled game that utilizes an android phone in order to activate certain attacks along with puzzles to engage the player at the end of every level. The expected benefits of the project are that this project aids our group in gaining experience on game development by going through each stage of the game development process and giving each of us in our group a taste of what it's like when creating smaller games.

2. BACKGROUND

In this background the team will be looking at Maze puzzle games in general, then the group will be looking at how to make a circular design of a maze, explaining what pixel perfect is and how it will work inside of the group project and finally the group will state the inspirations of the project and what elements of our inspirations are going to be going into the project.

The maze game genre is a video game genre that is described as the game uses quick player actions that required interactions with an enemy by outmaneuvering them or attacking them while navigating their ways out within a time limit or completing a specific objective.

The most famous maze game would be Pac Man which was a game with the titular character Pacman trying to escape from the four ghosts in-game which are Pinky, Blinky, Inky and Clyde. The goal was to collect all the pelts inside a maze before the ghost got to you and took out your lives. It gives the user multiple opportunities to kill the ghost for a limited time.

Branching off to puzzle video games, they focus on more the logical and conceptual challenges, it mainly focuses on solving puzzles as the main and primary activity as gameplay. The gameplay consists of shapes and colors or symbols the players must manipulate in order to directly or indirectly create a specified sequence.

The genre of puzzle games is very vast and requires some level of thinking which may uses numbers, colors, shapes, or complex rules. A common trope in puzzle games is giving the player lives to give them a sense of urgency to complete the number of tries.

The goal in this section for the group is to combine both genres into one and make a game surrounding that incorporates all the tropes and traits of each genre while adhering to the brief given to us for the project.

Moving unto the circular maze design, the group have decided to create a circular maze. To accomplish this assignment, the group are going to acquire the services of the website 'http://www.mazegenerator.net/' to create a randomly pregenerated mazes for us.

After creating the mazes, the team will then transfer the mazes into visual studios and modify them to covert the white pixels inside of the generated mazes into transparent pixels so that the team can place the object and they will stay on the same plane as the pregenerated maze. After all that the team will then apply the pixel-perfect collision detection method to make the maze detect collisions to it.

Pixel perfect collision is an accurate kind of 2D collision detection that uses bitmasks to determine if two objects collide. Bitmasks are used to create every object in our game because it will give the team something which will allow us to accomplish collision detections. The team was primarily inspired by *Zelda: A link to the past* and *Stardew valley*. These games are both top-down adventure games. The difference between the two games is that one game focuses on the action-adventure aspect of the game while the other focuses on the one more role-playing aspect of the game through my reach into both games. But both games have maze and puzzles aspect to them.

Zelda: A link to the past will be the game the group will be mainly taking from since it will have similar but inspired gameplay and it has the more maze section to the game than *Stardew Valley.* The place in *Zelda: A link to the past* that the group will mainly be taking from is the dungeon section of the game since it will be mainly focusing on the maze and in the dungeons inside of the games it also incorporates puzzles which meet our puzzles subgenre of the game. (Anon., 2019)

While in the game *Stardew Valley,* the group decided on taking the general movement of objects and GameCharacter in the project, so this will include the games mechanic like jumping, dodging, moving, attacking, defending and interacting with an object like for example picking up items. (Anon., 2019).

1. BACKSTORY OF GAME

The Back story of the game follows the character, Luna. The character Luna takes inspiration from the character the bride in kill bill. the story of Kill Bill follows the bride going on a journey to get revenge on the people who put her in a coma for 3 years. Luna at the start of the story

takes inspiration from kill bill by opening on her preparing for her wedding day. while she's waiting, she gets a knock on the door. When she opens the door, she meets a hooded figure.

The being approaches her and gives her a warning on her potential future that looks grave for her future life partner. After hearing this she is taken back by the being and assumes that if anything the being in front of her is involved in this attack against her to be a life partner. She boasts to the figure and tells it that she is going to stop any grave thing from happening to her life partner and she's confident in her abilities to do so.

The being tells her that she shouldn't be boastful and that it is not involved in any attack against her life partner. The being repeats again that he is only here to warn her about the coming attack against her life partner and if she feels that she can take care of it hopefully she does it before it's too late.

The being leaves the scene without a farewell. Luna tries to stop it from going and get more information from it, but he disappears before she could. The scene ends with her running out of the room to check up on her fiancé.

The next time we meet Luna she has just woken up inside a cave. She is looking for a way out and she has short memory loss. She ponders to herself for about where she is, what happens to her and why does she have a sharp pain around the back of her head. Once she thinks about it for after a while, she is shown a vivid memory of what had happened before she ended up in the predicament, she is in now.

She remembers that after she left the room, she went to go see how her fiancé was doing. Once she got to the area her fiancé was preparing inside, she saw that the door was opened and looking into the room she saw her fiancé lying lifeless on the ground. Before she could react and do something about it, she is hit with an object that knocks her out. Only catching a glimpse of the item used against her and the face of the person who did it to her. But the face is blurry in her memories, so she was unable to pinpoint who the person was.

With the grief on her shoulders knowing that her husband has passed away she breaks down and takes in the fact that her husband is no longer with her. After a few hours, she picks up herself and starts to make her way through the dark cave. She is optimistic because she is hoping to get out and get revenge for her husband. she encounters the hooded figure one more time and knowing that it was not the being that killed her husband. They exchange a few words before the hooded figure informs her, it pities her and will give her one more chance to write her wrongs by throwing a sword at her and using magic to teleporting her to the first level saying to piece your memory together you will have to traverse the depths.

3. NEEDS STATEMENT

The game is going to be set apart from the others because it will be utilizing circular maze types. Most maze type games that are being produced will be geometrically angular. The team will be using an android controller, which will also set the game apart from its competitors. There is a gap in the market for circular based maze games that use separated controllers and their sensors.

4. SCOPE

The team must produce a game as a component of our third-year project. This project must be submitted as an entry to the Games Fleadh competition in March. The theme for Games Fleadh 2020 is "Maze". The team can use an OO language such as C++, C#, or Lua. It is expected of us to use Simple & Fast Multimedia Library to build a desktop game. The team can use any assets the team creates, and/or any assets that are free to use and to whom you give copyright.

The game needs to have a maze element, as this is the genre of the game. The team would be aiming to use circular mazes rather than a traditional square.

The game needs to include an enemy of sorts, that challenges the player throughout the maze.

Another challenge in the game would be a mini-game system. At certain points in the game, the player is required to complete a minigame to progress in the game.

5. PROJECT MEMBERS

Team Member	Role	Contact Information	Responsibilities
Thomas Jones	Champion	K00232078@student.lit.ie	Makes sure that the project is going into the correct direction and doesn't get derailed.
Elijah Omotosho,	Stakeholder	K00227425@student.lit.ie,	

Jordan O'Loughlin		K00225361@student.lit.ie	
Jordan O'Loughlin	Project Manager	K00225361@student.lit.ie	
Thomas Jones, Elijah Omotosho, Jordan O'Loughlin	Architect	K00232078@student.lit.ie, K00227425@student.lit.ie, K00225361@student.lit.ie	
Thomas Jones, Elijah Omotosho, Jordan O'Loughlin	Analyst	K00232078@student.lit.ie K00227425@student.lit.ie K00225361@student.lit.ie	
Thomas Jones, Elijah Omotosho, Jordan O'Loughlin	Developer	K00232078@student.lit.ie K00227425@student.lit.ie K00225361@student.lit.ie	Coding the Games and writing up documents that are involved with the game.

2 Problem Definition and Background

2.1 What is to be Accomplished

After some initial brainstorming, the scope as defined by the team was to create a Maze game, with a circular maze instead of a square maze. The game would implement an android controller to provide enhanced playability. Part of the gameplay would include a murder mystery to be solved to complete the game. Finally, puzzles could be played between levels to give hints or prevent players from progressing. Introducing these features to the game will present problems in coding and time management.

The issue with a circular maze is that SFML, the library the team will be using works in a 2D cartesian coordinate grid, which means that the maze cannot be perfectly circular. The collision system that the group has been familiar with is designed around square walls that fit neatly to a grid system. This system creates a bounding box around a sprite and checks for collisions between bounding boxes. When the circular wall is created the bounding box, or abounding circle will not suit. To overcome this, the group intends to research different collision detection algorithms.

When attempting to integrate the android controller to the game, the core issue is compatibility between C++ and the Android language. To overcome this, research is to be carried out about the API and any IDEs that could be used to aid the group in accomplishing this task. Another issue with the android controller is the delivery method; will a different application specifically

for the android device to allow users to input controls such as movement or are there already applications dedicating to performing this on the market. If this is difficult to employ, a simple XBOX controller could be implemented from built-in libraries.

A murder mystery element provides many pitfalls including the complexity of coding the solution, enforcing randomisation into the selection and pick up of clues, allowing the user to view the clues that they have discovered, and how this relates to solving the mystery. The strategy for handling these problems is to keep everything simple, not to make the mystery too complex, and to allow a player to be able to solve the mystery every time. Research into mystery type games and how they implement their mysteries and how the user solves them. The randomisation of the mystery and complexity of code will be handled by programming conventions, clear code, and object-orientated programming.

Adding puzzles to be solved in the game, adds a level of extra complexity. The issue with this is that there would need to be several puzzles put in place to increase the replay ability of the game. Each puzzle requires a known solution if the puzzle is randomly created a random solution would need to be created, on the other hand, the puzzle could be hardcoded into the game system itself, as such the solution would need to be hardcoded. Either implementation of puzzles will require a lot of time to perfect and require a lot of testing.

2.2 Problem Definition (Story and Background)

Due to the style of the game, the team believed that the story for the game was crucial for the game to succeed. This was decided at the beginning of the project. Early concepts for what the game story included a version where you play as a man lost in a cave looking for his dog. Another had our protagonist looking for their dog throughout the maze. The initial concept that the team went with was a wedding massacre where in which a heartbroken bride witnesses the murder of her husband and seeks to find the killer.

The story began with the scene opening to a couple celebrating their marriage. While eating and drinking at their reception they are interrupted by a crazy witch who states how if she could not have the husband then no one could and kills the husband. Within in seconds, Luna expresses shock, sadness, fear, angrier raging at the mysterious figure. The goal of the game was to catch this figure and find out why they had killed Luna's betrothed. The game would show Luna seeking through a maze to find who the murder was. The final scene would be Luna finding out that the killer was her sister all along.

After several discussions with the supervisor and the team, it was decided that the plot of the game should be Luna fining her way through the maze all while collecting clues in an elimination-style. Each clue collected would tell you what clue is irrelevant to the solution. The team decided this would be the most fun as it turned the game into a full puzzle game with a maze to create a challenge in finding said clues. A large amount of inspiration came from the board game. The elimination was chosen over the collection style as this would challenge the player more and encourage them to explore each level to the fullest in order to find all the clues to make the solution easier to find.

The character of Luna was to be an ex-adventurer who has settled down and gave up on adventuring and wanted a similar life with a family. She found that in Nate who was a sweet person. Life went on and Luna found herself happy till the day that was meant to be the best in her life. After finding her husband dead she wakes in a gothic cave system. She must try to remember what has happened and who she found in the room with her husband dead on the floor. If she

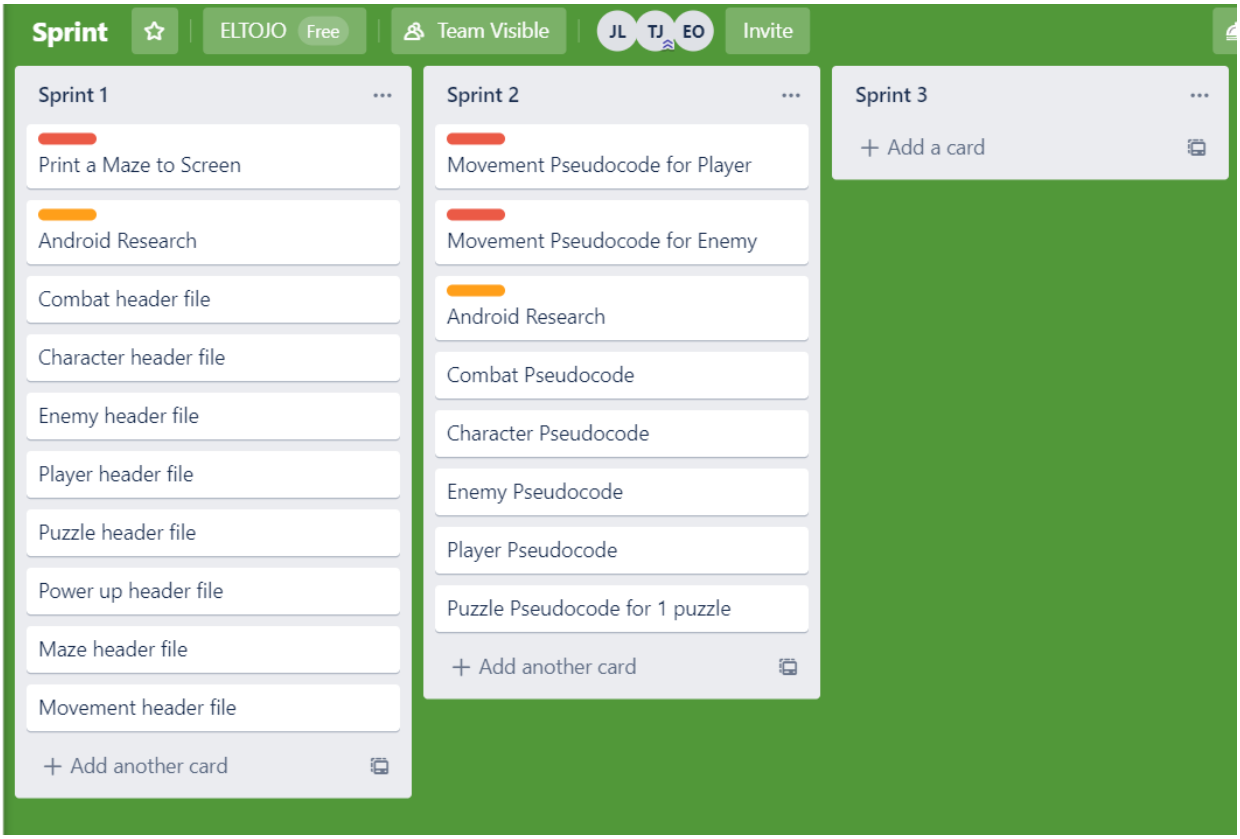
does not the mysterious figure who has put her there (unknown to her) will keep her there for all the entirety. It is up to Luna to act and solve this mystery

2.3 Project Management & Communication

The main objectives for the project are to deliver a game that challenges the player with enemies along with a club system that has them solve a murder mystery. Some of the biggest problems the team believed to have faced at these points include the collision system, club system. These were some of the hardest problems that the team have faced. These problems the team were solved by the team and were correctly implemented into the game. The main issue with collision was due to how the collision piece of code that controls all collision in the game. Due to it being pixel by pixel this would cause the player sprite to get caught inside of the maze walls if the player was to get near the wall and go in another direction. To solve this problem one sprite that was a collective of the player sprite sheet. This would check all the sprites at once. This resolved all issues with the player getting caught in the wall

The plan for project management for the team included a weekly meeting which the team would use to discuss what they had accomplished for that said week. This would allow the following week's tasks to be fitted to what was completed the previous week by the team. This would allow plans to relevant and done in a priority the team assigned to each task.
Along with the meeting, all task was pinned to a board in **Trello.com.** This would allow each member on the team to be able to see at any given point what each of the members were working on ensuring no overlaps in similar tasks being done at once.

Figure 1 - Trello Board



Near the end of the project, the Trello board became must less filled and some tasks were talked about and completed without being added to the Trello board. These would occur whenever there was a coding week due to the team being in the one-room everyone was able to discuss what each member was working on. Along with the Trello tasks a few weeks into the project the team dropped the sprite sheet system due to each week the three would be multiple changes that needed to be made each week with what had to be done for the following week leading the team to scrap this sprite system and in turn, focusing on more what was done and needs to be done on a week to week base.

However, a lot of the ideas for the game had to be eliminated due to time constraints. A meeting which discussed all the features that needed to be added to the game before the deadline. These were sorted into three groups. Essential and nice to have. This focused the team on completing the rest of the game and narrow our vision to something that the team believed to be possible to accomplish in what time was left.
In these last few weeks, the team has decided to commit the project for the maturity of the time remaining in order to ensure its completion.

2.4 Problem Definition Conclusion

By the end of the project, the team had learned a lot about the problems we had predicted to have encountered along with some problems that were unseen at the time of writing the problem definition section. One of the biggest issues that the project quickly ran into was the complexity of implementing an android phone to be used as a controller. This was found out to be a lot more difficult than the team first anticipated. Due to the nature of the C++ language and SFML, the team decided that the time investment into the research it would take to allow this type of controller to be used inside our game would be too large for the time the team had to complete. The team decided to instead further develop upon the clue system instead of the controller as our unique feature. The was clear this was the right choice due to the result of the game that was reached by the end of the project where each member was delighted in the result of the project.

Along with the controller, another problem the team faced was the pixel-perfect collision system that was opted for by the team. This problem turned out to be quite mild and was accomplished quite quickly into the development of the game. The collision system found worked mostly without any problems. The only problems that the group faced were a slight flicking when the player collides with the wall. The team was unable to fix this issue due to the way that the collision system was set up and decided to leave it in the game as it did not cause much of a problem for gameplay. Furthermore, another issue with the collision was due to the multiple player sprites. Once a player had got next to a wall and switched directions and changing the sprite the player would permanently stick in the wall. This was solved by creating a secondary player sprite that was made up of all possible sprites the player could use. This would check the

forest point the player could touch and if there was a collision then the player would be moved back to their previous position.

The clue mystery system was implemented with a little bit of hardship. The goal of the system was to randomly select a set of possible clues and place them throughout each level. Along with this an inventory system that would display each clue and clues that were already picked up. This was created using vectors that would store a set of clues such as four weapons,4 suspects,7 motives. The only part of the system that is not random is the place in which the clues are. Each clue will be in the same position each game however that clue that is picked up at that location is random.
Finally, due to time constraints, the puzzle system was dropped to prioritize more important features that needed to be developed. The starting code of the puzzle has been left inside the game folder to demonstrate the thought process of how the system was going to work.

2.5 Implementation

Many different types of Implementation were tried, In the project and the one that had the most success was making managers for objects inside of the project. The objects that were made into managers were the Clue System, which was made into the ClueManager, the SpriteManager, the LevelManger, the SoundManager, the StateManager and the TextManager. The Manager were implemented into the project to make it more cost effective on memory so that the game can run efficiently without having to worry about lag while spectators play the game. The ClueManager was implemented by taking the clue object that was made inside of the project and making three vectors out of the clue pointer object, by using the spriteManager object and Sprite pointer. Inside the constructor method of the ClueManager, the spriteManager pointer is used to get the Sprite of a pickup, it also used to create four different method used inside of the file to setup the weapons, motives, suspects and sprites of the clues.

```h
#include "Clue.h"
#include "SpriteManager.h"
#include <SFML/Graphics.hpp>

#ifndef CLUE_MANAGER
#define CLUE_MANAGER
class ClueManager
{
public:
    ClueManager();
    vector<Clue*> weapons;
    vector<Clue*> suspects;
    vector<Clue*> motives;
    SpriteManager spriteManager;
    sf::Sprite* pickedUpSprite;

    Clue* RandomClue(string type);
    Clue* RandomClue();

    vector<Clue*> ReturnSolution();


private:
    void SetUpWeapons();
    void SetUpMotives();
    void SetUpSuspects();
    void SetUpSprites();
    int RandomNumber(int size);
};
 #endif // !CLUE_MANAGER


```

```cpp
sf::SoundBuffer* SoundManager::GetAudio(string path)
{ //DONT PUT IN THE WRONG PATH FFS!
    sf::SoundBuffer* audio;
    audio = nullptr;
    if (AudioExist(path) == -1)
    {//if it doesn't exist
        audio = SetAudio(path);
    }
    else
    {//if it does exist
        audio = audios[AudioExist(path)];
    }
    return audio;
}

```

Inside of the RandomClue method that takes in a string type of clue and checks what type of clue it is first, it does this by taking in a char option that checks the first char that was taken into the method. The char is then put into a switch statement. Inside the switch statement it uses the Char Case to check a solution is made for the clue system and it does it by for creating a while loop in the specific case the using the random variable that was created inside the method

to equal a from a method called Random number which takes in the size of the clue and calculates the integer r of the random number between the size of the clue vector and zero. Then an if statement is used to check if the clue is not placing it into the pickupClue pointer. Then a Boolean created earlier is set to false.

```
Clue* ClueManager::RandomClue(string type)
{
    Clue* pickedClue = nullptr;

    char option = toupper(type[0]);
    int random = 0;
    bool incomplete = true;
    switch (option)
    {
    case 'W':
        while (incomplete)
        {
            random = RandomNumber(weapons.size());
            if (!weapons[random]->solution)
            {
                pickedClue = weapons[random];
                incomplete = false;
            }
        }
        break;
    case 'S':
        while (incomplete)
        {
            random = RandomNumber(suspects.size());
            if (!suspects[random]->solution)
            {
                pickedClue = suspects[random];
                incomplete = false;
            }
        }
        break;
    case 'M':
        while (incomplete)
        {
            random = RandomNumber(motives.size());
            if (!motives[random]->solution)
            {
                pickedClue = motives[random];
                incomplete = false;
            }
        }
        break;
    }
    return pickedClue;
}
```

```
int ClueManager::RandomNumber(int size)
{
    int r = rand() % size;
    return r;
}
```

In the next method called RandomClue it takes the same premise as the Random Clue from before but it uses Boolean instead and uses the randomNumber method and takes in 3, it use the switch statement again but instead of the while loop it adds another condition where if weapon is not random and it is picked up, set the clue random number position to the pickedUp

which is a null pointer and set the Boolean to false. Once the while loop is done set the pickedClue to true and return it.

```
Clue* ClueManager::RandomClue(string type)
{
    Clue* pickedClue = nullptr;

    char option = toupper(type[0]);
    int random = 0;
    bool incomplete = true;
    switch (option)
    {
    case 'W':
        while (incomplete)
        {
            random = RandomNumber(weapons.size());
            if (!weapons[random]->solution)
            {
                pickedClue = weapons[random];
                incomplete = false;
            }
        }
        break;
    case 'S':
        while (incomplete)
        {
            random = RandomNumber(suspects.size());
            if (!suspects[random]->solution)
            {
                pickedClue = suspects[random];
                incomplete = false;
            }
        }
        break;
    case 'M':
        while (incomplete)
        {
            random = RandomNumber(motives.size());
            if (!motives[random]->solution)
            {
                pickedClue = motives[random];
                incomplete = false;
            }
        }
        break;
    }
    return pickedClue;
}
```

Next method is return Solution inside this method the solution to the clue are created. This is done by creating a solution vector which takes in a weapon, a suspect and a motive. This feat is accomplished by using a for loop and an if statement which takes the then I potion into the solution, then pushed back the clue object into the solution vector. Return solution.

```
vector<Clue*> ClueManager::ReturnSolution()
{
    vector<Clue*> solution;
    for (size_t i = 0; i < weapons.size(); i++)
    {
        if (weapons[i]->solution)
        {
            solution.push_back(weapons[i]);
        }
    }
    for (size_t i = 0; i < suspects.size(); i++)
    {
        if (suspects[i]->solution)
        {
            solution.push_back(suspects[i]);
        }
    }
    for (size_t i = 0; i < motives.size(); i++)
    {
        if (motives[i]->solution)
        {
            solution.push_back(motives[i]);
        }
    }
    return solution;
}
```

Setup methods are to store the clues into their specific criteria which are weapons motives and suspect which are then it set the solution to a specific clue to true using the randNumber function.

```
void ClueManager::SetUpWeapons()
{
    weapons.push_back(new Clue("Candlestick","The WEAPON couldn't have been a Candlestick, they were all in their right place!","weapon","Textures/Candlestick2.png"));
    weapons.push_back(new Clue("Poison","The WEAPON couldn't have been poison, I remember he WASN'T foaming at the mouth!","weapon","Textures/Poison2.png"));
    weapons.push_back(new Clue("Sword","The WEAPON couldn't have been a sword, there was no stab wounds!","weapon","Textures/Sword2.png"));
    weapons.push_back(new Clue("Club","The WEAPON couldn't have been a Club, I remember it wasn't that type of wound!","weapon","Textures/Club2.png"));

    weapons[RandomNumber(weapons.size())]->solution = true;
}

void ClueManager::SetUpMotives()
{
    motives.push_back(new Clue("Greed", "He couldn't have been killed because of his GREED, he wasn't like that, always giving to charity!", "motive", "Textures/greed.png
    motives.push_back(new Clue("Envy", "He couldn't have been killed because of his ENVY, he didn't really care for material things!","motive", "Textures/envy.png"));
    motives.push_back(new Clue("Lust", "He couldn't have been killed because of his LUST, he was a gentleman who only had eyes for me!", "motive", "Textures/lust.png"));
    motives.push_back(new Clue("Sloth", "He couldn't have been killed because of his SLOTH, he was a dilligent man, who had a good work ethic!", "motive", "Textures/sloth
    motives.push_back(new Clue("Wrath", "He couldn't have been killed because of his WRATH, he was a very understanding and calm man!", "motive", "Textures/wrath.png"));
    motives.push_back(new Clue("Gluttony", "He couldn't have been killed because of his GLUTTONY, he always showed some self restraint!", "motive", "Textures/gluttony.png
    motives.push_back(new Clue("Pride", "He couldn't have been killed because of his PRIDE, he wasn't afraid to show humility", "motive", "Textures/pride.png"));


    motives[RandomNumber(motives.size())]->solution = true;
}

void ClueManager::SetUpSuspects()
{
    suspects.push_back(new Clue("Bob", "The MURDERER couldn't have been Bob, he has no hands!", "suspect", "Textures/Bob.png"));
    suspects.push_back(new Clue("Sam", "The MURDERER couldn't have been Sam, she is a pacifist!", "suspect", "Textures/Sam.png"));
    suspects.push_back(new Clue("Jim", "The MURDERER couldn't have been Jim, I remember the priest kept talking to him!", "suspect", "Textures/Jim.png"));
    suspects.push_back(new Clue("Skylar", "The MURDERER couldn't have been Skylar, he wasn't mentally strong enough for that!", "suspect", "Textures/Skylar.png"));

    suspects[RandomNumber(suspects.size())]->solution = true;
}
```

The set-up sprite method in clueManager is used to set the origin of each clue and for the picked-Up Sprite.

```
void ClueManager::SetUpSprites()
{
    for (size_t i = 0; i < motives.size(); i++)
    {
        motives[i]->clueSprite->setOrigin(300, 300); //Objects are 600x600
    }
    for (size_t i = 0; i < weapons.size(); i++)
    {
        weapons[i]->clueSprite->setOrigin(300, 300); //Objects are 600x600
    }
    for (size_t i = 0; i < suspects.size(); i++)
    {
        suspects[i]->clueSprite->setOrigin(300, 300); //Objects are 600x600
    }
    pickedUpSprite->setOrigin(300, 300);
}
```

3 Literature Review and Research

3.1 Character Archetypes in Games

3.2 Introduction

Character archetypes are important in games as they provide specific roles for the player for example, in Portal 2 the hero needs to escape from a testing facility (Valve Corporation, 2011). The heroes need to escape hands off this need to the player too. Archetypes can be a tool that game creators can use to prevent breaking an immersion from the game and even provide hooks and replay value. It is rare and next to impossible to find games that don't have any basic character archetypes. However, there are some games that don't have some of the core archetypes such as Simulator genre games like The Sims (Electronic Arts Inc., 2000) generally lack a shadow, mentor, or guardian. This lack of archetypes works well for some, for example, the puzzle game Tetris which has no shadow or hero (Pajitnov, 1984), but it is essential to have a very minimum character archetype.

Some interesting points can be examined with archetypes as a focus; the first of which is a minimum archetype for a maze adventure type game and detailing what will be needed and what won't be needed. Secondly, reviewing archetypes in other games and how they can develop the story. Lastly, looking at how interactive gameplay can have an impact on these archetypes and the story they develop.

3.3 Required Archetypes

There are several Jungian Archetypes that apply to video games such as The Self, The Shadow, The Hero, The Trickster, and The Sage (Jung, 1991). These archetypes bring life to the story and draw the player in. These Jungian Archetypes aren't restricted to video games; art, music movies, books all contain these fundamental Jungian Archetypes. These archetypes are applied to provide the audience with a better connection to the story (Novak, 2012).

Novak (2012) goes on to list the main archetypes; hero, shadow, mentor, ally, guardian, trickster, and herald. The hero is the main character, and the most important in most games as this is the player's avatar…" which "the player must identify with…" (Novak, 2012). This hero starts the story in danger of having some form of a major problem to work towards and solve. In games there are many examples of this for instances in The Legend of Zelda, Link is the Hero (Nintendo, 1993); and in Stardew Valley, you are the Hero (Barone, 2016).

The Shadow is the next important Jungian Archetype that is present throughout most video games. The Shadow is the anti-Hero, representing everything that the Hero is not. This is the main villain of the story that is solely responsible for the Hero's problems. The Shadow doesn't need to be a different entity to the hero; it can represent a dark side of the hero (Novak, 2012). In the Legend of Zelda series, for example, one of the villains is Dark Link and is a representation of his own strength being used against him (Nintendo, 1993).

These two archetypes; Hero and Shadow, are the main driving factors in games. Most players will become emotionally attached to the hero of the game and align themselves against the shadow. These will be the main set of archetypes in the maze adventure genre. The hero must navigate the maze, and at the centre of the maze, the shadow or an agent of the shadow such as the Guardian or Trickster archetype will be there to oppose the Hero. There is also the question of whether, in a maze game, a Mentor archetype is needed to get the player used to the obstacles in the maze? The Mentor might not be needed throughout the game but just the start. For example; The Legend of Zelda: Link's Awakening, Link's grandfather was his Mentor, and that is where players first learn how to attack and block (Nintendo, 1993).

Examining these archetypes is key to writing a solid and compelling story that will keep the player interested in the gameplay. Every game should be looking to get the player to be emotionally invested with the hero. Using the Guardian or the Trickster archetype to keep the identity of the Shadow hidden can "add to the story's dramatic tension" (Novak, 2012).

3.4 Examples of Main Archetypes in Games

Reviewing and exploring examples of the archetypes of Hero, Mentor, Guardian, Trickster and the Shadow, in other games can bring light onto what is needed from a maze adventure genre game in terms of the main characters. Analysing these archetypes that work well, and those that don't work well can provide a better knowledge base for archetypes too.

A most notable reference to the Hero and his journey is in the video game Assassin's Creed (Ubisoft, 2007). There is a striking resemblance to Joseph Campbell's Monomyth where the hero's adventure consists of separation, initiation, and finally a return (Campbell, 1975). The Hero, Desmond, is taken and transported to a different land. Reluctant at first, however through the aid of mentoring by some allies, he becomes more willing. After this, Desmond embarks on his journey to find a piece of Eden which will help everyone. In the end, this turns out to be far from the truth and his mentor becomes the antagonist. After the battle is won, Desmond returns to his world. Returning to the world he makes a lucky escape and is rescued by allies, which sets up for a return.

Moving on from looking at the hero, two examples of good villains in video games can be seen in Portal 2 (Valve Corporation, 2011), and Tekken (Namco, 1994). In Tekken, a simple fighting game, the plot is fascinating. The main antagonist being Heihachi Mishima who has done horrible things to his son. His son spends most of the game fighting to get the chance to beat his Father. However, Heihachi has trained someone else as an opponent, which acts as one of the Guardian archetypes of the game (Namco, 1994). In Portal 2 the protagonist is also a Trickster archetype, always trying to hinder the hero in some way with trickery and deceit. Sometimes these tricks work against the antagonist and this is part of the Trickster archetype. This is a form of comic relief in the game too, providing a little more entertainment in the puzzle-solving type game (Valve Corporation, 2011).

In Portal 2, a strange transition of role happens, which seems to be synonymous with other games. In Portal 2's predecessor Portal 1, the villain was GLaDOS. In Portal 2 the game begins with

a mentor and ally named Wheatley. As the story progresses, GLaDOS returns, and Wheatley helps overcome her. However, with this act, Wheatley becomes the villain and GLaDOS becomes a reluctant ally (Valve Corporation, 2011). This change; where the hero becomes a villain, or the villain becomes an ally is seen throughout many games. The Tekken series itself is one of these. This is referred to in The Monomyth (Campbell, 1975) as the return; where the hero has a reason to come back, maybe not as a hero, but an ally, guardian, or even villain.

Examining the archetypes defined by Novak (2012), and the hero's journey as told by Campbell (1975) there is a direct correlation to the enjoyment of a game and its story content. By using the archetypes, heroes and villains can easily be created, and this can be seen in numerous pieces of entertainment media.

3.5 Interactive Gameplay's Effect on Archetypes

The classic archetypes can be set in stone, however in some cases, in interactive games, the villain can change, the ally can change, and even the hero can become the villain. As described by Josiah Lebowitz (2011) there is a spectrum of interactive storytelling. Depending on how open-ended a game is, will impact on the number of possible shadows and guardians on the higher end of the interactive storytelling spectrum.

Singularly the hero will often not be impacted by changes in choice until the very end of a story or game. Using BioShock as an example there were two possible endings, a good ending and a bad ending (2K Games, 2007). In Fable 3 scattered throughout the game are decisions to be made, and those decisions impact the Hero's looks, how they are viewed by NPCs, and the endings of the game (Lionhead Studios, 2010).

Villains in Skyrim V are dependent on the hero's perspective. There are several options in Skyrim to choose a side of an opinion and these are very well played out. This allows for the game to be played in different ways, and each time the villain or the shadow might not always be the same. (Bethesda Softworks, 2011). The story in Skyrim can be considered as an open-ended one or even a true player-driven one on the spectrum of interactive storytelling as described by Josiah Lebowitz (2011).

Using Skyrim (Bethesda Softworks, 2011) as an example of a good open-ended or even player-driven story and Assassin's Creed (Ubisoft, 2007) as a good Interactive Traditional story, comparing the Guardian archetype, it is easy to see some similarities. In Assassin's Creed, there are several mini-boss type characters which prevent the hero from learning more and progressing until they have been eliminated. In Skyrim (Bethesda Softworks, 2011), following a storyline can be similar to that of following a storyline in Assassin's Creed (Ubisoft, 2007); without tracking down a specific item, you won't be able to progress through that storyline. However, Skyrim will allow you to continue with other main quest lines and doesn't specifically require you to complete storylines to complete the game.

When using interactive gameplay, the archetypes in games, such as Skyrim (Bethesda Softworks, 2011), can be very different depending on how each user plays the game. Most notably the Shadow and the Hero take the most impact. Guardian archetypes aren't as impacted, their presence almost remains the same, though they may take different forms depending on the story and the villain.

3.6 Conclusion

In conclusion, it is clear which core Jungian archetypes are necessary for a maze adventure genre game to keep the story interesting and focused. The most obvious is the Hero and the Shadow, after this there is a specific need for a Mentor and a Guardian. The use of an Ally can enable hints

and solutions when Heroes are having difficulty completing the maze. Using a Trickster can provide some comic relief and humour in the game.

These specific archetypes in an interactive game, though easy to craft from a traditional story point of view, needs more thought put into for interactive games that has a branching path, open-ended, or player-driven story. While looking at a truly interactive experience, there is an aim to keep the focus on these archetypes as they drive the game and story, while also drawing connections with the player.

3.7 Branching andNarratives

Introduction

In this part of the research paper, the two topics that are going to be discussed will be **Branching** and **Narrative**. The heading under **Narratives** that are going to be discussed is **Narrative Structures, Techniques and Connection to Branching**. The heading under **Branching** topic that is going to be discussed is **What is Branching and Branching Plotlines in Media**.

After going through both topics inside of this literature review, there will be an evaluation of what has been learned through the topics and headings that accompanies them, with a piece on how what was learned can be incorporated in a gaming environment.

Narrative Structures, Techniques and Connection to Branching

A narrative is an announced or penned explanation of occurrences to be followed by the audience. A successful narrative leaves an imprint on its audience (Raney, 2011) e who explored the role of morality emotional reaction to and enjoyment of media. Enjoyment of the story, or otherwise, is communicated through emotional reactions like high appreciation of the work, high disdain or worst of all indifference.

These emotions can be display by the audience through the fallout games series (Bethesda Softworks, 2011) which demonstrate how the audience can display an array of emotions as stated earlier in this paragraph. This is exhibited through the reactions of critics and fans sites like Metacritic (Metacritic, 2019) which exist as media analysis site for both consumers and professional reviews.

The website gives a broad view on how the general public and the analyst view a piece of media through a critical lens, a case shown of this would be the public perception of different games series. These games are Fallout 3 (Metacritic, 2008), Fallout 4 (Bethesda Game Studios, 2015) and Fallout 76 (Bethesda Game Studios, 2018), they display the different types of emotional reactions already discussed with Fallout 3 representing the game where the audience enjoyed the demonstrating high appreciation for the work, Fallout 4 representing the game where the audience demonstrated indifference to the game and Fallout 76 representing the game where the audience demonstrated high disdain for the work as an emotional reaction.

Since a Narrative is a story, it needs a structure to keep it the audience invested in the content that is being presented them, this would be known as the plot of the story. The Narrative structure is about the plot giving a reason why the character in a narrative faces conflict, this is where the locations of the events will take place and who are the key players inside of narrative which is the main characters, while the plot in relation to a narrative purpose is to display where the conflicts are constructed and concluded. Examples of this can be the Star Wars First Trilogy under George Lucas (Star Wars, 1977), it follows a plot where a hero, in this case, Luke Skywalker go out on a journey with a group of companions to master the force and defeat the evil empire. Narrative Structures follow a **Three-Act Structure** (A.L, 2005), where each act focuses on a specific part of a story. The first act of the **Three-Act Structure** is referred to like the setup of the story, this is where the main protagonists are established in the plot and this is where characters are explored which could be their habits, personality, backgrounds and quirks are. It will also deliver and introduce a difficult issue or situation in the story, such as in screenplay like Alien (O'Bannon, n.d.)**,** the main protagonist Ripley is established as mother figure who loves her daughter and wants to get back to her as quickly as possible but her rank warrant master makes it challenging for her to accomplish it since she is stuck on ship infested with aliens and because of her occupation she is not equipped to

deal with the aliens. After the first act has established the characters and conflict it lends itself to the second act in the **Three-Act Structure**, this is where the main protagonists that were set in the first act, go through transformation involved with the issue established through the form of character development which is more commonly known as a character arc. This character arc is used to show the extensive changes the protagonists go through based on the conflict they go through in the first act, an instance of this that can be seen in a game would be Undertale (Fox, 2015), It is game that can be beaten without doing any harm not any creature inside the whole game. The first establishes the story of the game and depending on the decisions the player make in the middle of the game; the player learn more about the world and a lot about themselves as they play through the game. By piecing mysteries together and trying to solve the conflict established for the player which is to climb out of the demon world without being slain by the king of the domain that the player character happens to fall into.

Then after the protagonists have gone through their character arc, they then make their way to the final act in the **Three-Act Structure** which is the resolution arc. This is where the conflict that was created in the first act and explored more in-depth in the second act is then confronted by the protagonists after they have been forced to by the condition made either in the first or second act. all the components of the issue are brought together and concludes the story wrapping it into a neat bow. An instance of this shown to me in a piece of media would be Of Mice and Men by John Steinbeck **(SparkNotes: Of Mice and Men, 2019**). The resolution act was set in motion once Lennie who was the main character had killed Curley's Wife by suffocation. The forced the story to halt and focus on Lennie trying to get away from Curley and his men who were after his life. He enlists his best friend George and few other characters help to get him to safety, but unfortunately, through several factors, they failed at the task and the book comes to an end in a quiet, but the sombre end.

There are distinct forms of **Narrative Structure Techniques** which are **Linear Narratives** (Sciutteri, 2018), **Nonlinear Narratives** (McIntosh, 2010), **Interactive Narration** (Wand, 2002) and **Interactive Narrative** (Ryan, 2009). **Linear Narrative** is the most frequently used technique and it displays in sequential order and shows the events in the order they happened. a game that follows **Linear Narrative** perfectly would be Super Mario Bros (Nintendo, 1985)which has a simplistic where the player by himself or with a friend plays the characters Mario, Luigi or both to go and rescue the Princess from the evil scoundrel King Koopa.

Nonlinear Narratives is a narrative technique that shows events in a nonsequential order which means that it doesn't follow a traditional flow in terms of a story and has the story broken up in multiple parts. A demonstration of this inside of gaming media would be the game Mass Effect (BioWare, Edge of Reality, Demiurge Studios, Straight Right, 2007)with having whatever decisions the player throughout the game, affecting the ending the player would get.

Interactive Narration is when a user/player choice helps drive the plot forward through the user's interactions. A game that demonstrates this would be Final Fantasy 7 (Square Enix, 1997), by allowing to interact with the narrative without effecting the story like adding optional characters. But keeping the definitive ending of the story no matter what the characters actions.

Interactive Narrative is a scheme about fiction where users can make their own decisions that affect the narrative through their actions which can alter the plots that develop alternating endings. Games like fable (Lionhead Studios, 2010), allow the player's action to not only affect gameplay, but the story being told throughout the game.

The **Narrative Structure Technique Nonlinear Narrative** is essentially what the concept of **Branching** is, which was explored by Sandy Louchart and Ruth Aylett in their paper **A Narrative Evolution** (Sandy Louchart, 2005)**. Branching** in a sense works with **Nonlinear Narrative** to connect plots or potential events, so by bring a real-world example of branching which is **Schrödinger Cat Theory** (Schrödinger, 1935) that states that there if you put a cat and a piece of radioactive materials into a box, it asks the question if the cat is still alive or not leaves two possible scenarios that are explored if either are true.

What is Branching and Branching Plotlines in Media

Branching Narrative (Nelson, 2015) acts as a Structure used for as the **Nonlinear Narrative**. So, looking at the first act, once the protagonists are given a foundation the narrative gives the small decision- making choices which could lead to a new array of choice that has to do with the conflict established. These new arrays of choices can also lead down to their own cluster of which can go on and on and on forever. Entertainment Media which gives creatines to the **Branching Narrative** will be explored in this section, the type of media being looked at will be **Branching Narratives** in **Film**, **Literature** and **Gaming**.

Branching Narratives Structure in **Film** has been shown in a show such named **Bandersnatch** (Bandersnatch, 2018). **Bandersnatch** is a choose your own adventure show like video games like **Zork** (Infocom, 1979) or in modern time the **Walking Dead** (Telltale, 2012) that encourages input from the user based on the option given to them to make for the protagonist of the show. The options can range from picking what type of breakfast the protagonist is going to devour to deciding if the audience wants to end another's characters exist in the show.

Each choice the audience makes over the course of the movie affect what type of ending they could possibly receive and it changes what the tone of the movie the audience watches, it could range from ghost story ending to The **Truman Show** (The Truman Show, 1998) type of ending. This demonstrates that on the entertainment side of the film it can produce a different viewing experience for each time it is view audience showing a strength of Branching Narratives, but its downfall is that repetition can slowly erode interest of the audience since it forces them to replay and redo the same routine until they pick a crossroad where they can select a different decision to their last one.

Branching Narratives Structure in **Literature** has been shown in Books related to **Dungeons and Dragons** (Arneson, 1974). Dungeons and Dragon are role-playing games that incorporate Literature which are used as a foundation to a story where players are dropped down to roam, explore and interact. Inside don't ring and Dragons there is but enforcer called the dungeon master he is the person that creates the world the players are going to be playing in. He also dictates this story so he sets up the conflict and the reason why the players must go on their journey to stop for example an evil wizard or stop a war from breaking up in a continent of his world.

The reason why this **Literature** classified as a **Branching Narrative** is because of the multiple possibilities of the players affecting the world through the choices they make either being minuscule decision to Gigantic decision that make an impact on the world. The change be change through the Literature using dice rolls and the players or Dungeon Masters imagination, which makes it so that if the player or dungeon master can think of it. Making the possibilities endless and run on forever without the restriction of films and gaming.

Branching Narratives Structure in **Gaming** has been shown in gaming by using titles like **Shadow the Hedgehog** (USA, 2005). Shadow the Hedgehog uses a Branching Storyline map which moves to a different event in the story based on how the player interactions with the NPCs inside of the game- world, so for if the protagonist decides to go against human or decides to go against the aliens in the

the world will branch off into two completely different sides of a story. But if the characters decide to play for and against both sides they are then pushed into the middle of the story.
These are all displayed using a **flowchart** that is inside of the game. A **flowchart** is a diagram that tracks a workflow or process. So, in this scenario it can be used to keep track of every key decision made by characters inside of a narrative by making a branch to another event after the outcome of the first event has been reached, so on and so on until it has reached the end of a narrative.
Overall by looking at **Branching Narratives Structure** in media and seeing the similar and constating traits of the Structure between **Film, Literature and Gaming.** You can see that by doing interacting with the medium a branch can be created which conceives a pattern of decisions that leads to multiple outcomes that leave the audience distinct ending based on the decision they had made throughout the narrative.

3.8 Conclusion

In conclusion, the Narrative Structure Technique gave way to the Branching Narrative Structure, This Structure allows the uses of choices to dictate how the protagonists are going to resolve the conflict. This creates repetition inside of a game, but it does enhance the user's experience in the game if their other possible endings that can be reached, by interacting with the game differently in the first and second act before triggering the third act. opening the possibility for the player to experience the game a different way and adding replicability to the game.

3.9 Section C

3.10 Introduction

Game mechanics are the core elements of a game that together create the kind of gameplay that the user will interact with within the game. Game mechanics are responsible for implementing the game's rules, challenges and objectives. When broken down into their individual components, games can often look like tasks. However, that is not how the player will experience the game. They will experience the game mechanics together and if done right they can create a large sense of fun and accomplishment for the player through visuals, accomplishments and challenges.

The first point that will discuss will be a general definition of game mechanics and how they create gameplay. Secondly, how game mechanics can be used to motivate the player to progress and fosters a sense of accomplishment is discussed. Gameplay mechanics are crucial to how challenging the player will find your game and how they will understand how the game works, and in this context how gameplay mechanics are introduced is also considered.

3.11 Description

3.12 What are game mechanics?

The definition of game mechanics is broken down into rules, challenges and objectives. Each will be discussed in detail.

Rules: Game rules dictate how the player can interact in your world. They are one the most important components of any video game as they define how a game world will operate to the player actions along with the winning conditions of said game. Jesper Juul determines "video games rules are activities based on formally defined rules and containing an evaluation of the efforts of the player". Rules, Gameplay and Narratives in Video Games. Simulation & Gaming: An Interdisciplinary Journal of Theory, Practice and Research. This means that rules are based around what the game designers set; however, these rules are useless without the player wanting to do anything. Rules are what bind a game world. Depending on your game your world rules

could be completely different from another game. For example, Red Dead Redemption which requires the player to complete missions in a very specific manner and doing anything too creative or different to how the game designer planned will result in a failed state. Or you could create a more open approach to how a player interacts with these rules allowing for more creative mechanics and actions that the player can take e.g. Zelda Breath of the Wild explains little on what the player can and can't do and drops the player into a wide-open world. Along with this, the game has a vast element system that the player can choose to use or not. Such as using a leaf to create a gust of wind that will drop a bomb onto unsuspecting enemies.
Either approach to games allows for very different gameplay approaches and different experiences neither of which are better than others.

Challenge: Another pillar of game mechanics is the challenging difficulty of a game. Many games find it difficult to find the right balance between baby mode, challenging and frustrating. Many games leave the choice of difficulty to the player with different difficulty modes or enemy scaling or ranking. This can be a great challenge for game designers as ensuring your game challenge is accepted for every player well. is pretty much impossible. Should the player be allowed to skip sections of your game if they find it too much of a challenge? This is where the challenge gets strange. If a player spends

countless deaths and hours trying to beat your game or if they breeze through at their own pace. Which one is more enjoyable to them? Take Dark Souls for example. This is a game that will break your time and time again. It punishes the player for being too confident when thinking that they have finally gotten around to understanding a boss's mechanic. However, this base core of punishment this cruel mistress that never relents and never surrenders is what makes finally defeating that boss you have spent till 3 am trying to defeat all the bittersweet. It gives the player a grand sense of accomplishment due to the extreme difficulty of said boss

However, take a more relaxing game such as Super Mario. This is a game that you can beat on a plane or during any form of travel while having a long conversation and not being totally concentrated. Compare that to Dark Souls where if you blink at the wrong moment it can mean certain death for you. This is another type of experience. As for which Is better it comes down to the player. If there is only one difficulty option for a game and a player finds it frustrating and wants it lowered there isn't much they can do. This leads to excluding some players from what the game, however, certain developers like the makers of Dark Souls believe that the challenge in each section of the game is key to the gameplay and how they want the player to experience the game.

Each game should be experienced in the way the player wants. Players if they want should be allowed to lower the difficulty however when it comes to games that are crafted around a said difficulty such as dark souls that is the way the player should experience it as with many games a higher difficulty doesn't add real challenge whereas just lowering the player damage or increasing enemies health. But certain games are only truly an experience if beaten in the way the developers intended. *The artistic significance of difficulty in video games – Reader's Feature*

Objectives: Creating relative objectives for the player to complete is essential to player motivation. Objectives should relate to the story of the game world and should include short- and long-term goals. *Videogame Objectives*

Short goals should include objectives that the player can complete regularly and keep them engaged such as rewards for completing short missions or levels.

Long goals should be something the player gets once they reach a great milestone e.g. get to a certain point in a story which will create another story hook that will leave the player wanting to know more or learn what happens next. Keep the player guessing. Along with this, the player should receive a reward for getting to this point.

3.13 Player motivation through game mechanics

Along with objectives story and character, progression is what can incentives the player to progress through your game. Motivation can come from short, mid and long-term mechanics
Short mechanics: Theses are events that are designed to occur every few seconds to minutes. Example of getting money in an RPG. These act as a basic motivator for the player. However due to the frequency of the events the player might not take notice of every little growth it does act to progress and acts a form of progression into midterm mechanics.
Midterm goals are that occur once per session. Session being relative to said game as a session can be different for the game genre. For example, let's use an RPG. These will add a permeant presence to the player in world change or player change such as a player level that will allow the player to unlock a new game ability. The difference between short and midterm is that the player will remember the impact of the midterm achievement and look forward to the next one.

Lastly is the long-term mechanics that will happen over several sessions of gameplay. These will occur after completing several midterm goals such as completing a long multi-part achievement. Due to long term mechanics being a lot like midterm just less common the player will think of theses less then midterm mechanics. However, motivation can be also defined in a more abstract way. *Motivating Mechanics in Game Design.*

Motivation can also come broken into Intrinsic or extrinsic. *Motivate player for better engagement and retention*

Intrinsic being motivation is driven by internal rewards that will give the player a sense of satisfaction. This is the pleasure a player gets completing tasks or activities.

Extrinsic refers to the rewards a player gets external of themselves such as getting a new level in a character.

How you introduce theses mechanics can play an essential part in the player's enjoyment of your game.

3.14 Introducing to Game Mechanics

Introduction of game mechanics is essential in how you want to pace your game to the player. Introductions go back to old school gaming where game mechanics were explained externally from the game itself in a game manual. However, with the adoption of digital sales in modern gaming, this has become an obsolete way of explanation. (bma5176, 2014) Game mechanics introduction can be done in three ways: tutorials, gameplay or none at all.

Tutorials are consistent of a prearranged level in which you guide the player through each one of your game mechanics slowly introducing them one by one where at the end the player should be familiar with how the game works and how they are expected to achieve each goal. Good tutorials give the player a simple introduction to game mechanics but leave the players interruption of how to use theses mechanics up to the player's creativity.

The gameplay is teaching the player by inserting them in the level or world and slowly introduce new mechanics at a time. A great example of this is Rayman Legends which has several levels per world. With each new world, each level in the world introduces new gameplay feature to keep the player engaged and always learning such as enemies that the player can ride or moving

lasers they must avoid. Each one of these new mechanics culminates at the end of each world to create a flow of gameplay that the player is familiar which ends with a masterful display of the player's skills that they have learned. Whereas the tutorials show the player how to do actions in a set-piece way gameplay allow the player to naturally learn how to learn mechanics that fill them with a great sense of satisfaction.

Finally, the most hardcore way to approach to introducing game mechanics to the player is by telling them nothing. Dropping them smack dead in the worlds and let the player learn at their pace. This can give the player a great sense of control and freedom. They must learn by themselves how to traverse the world. However, unlike the other two methods, this method can lead to player frustration due to lack of knowledge. This can lead to not learning a mechanism hour into gameplay that would have helped them in previous sections. This would not occur with the other two methods

3.15 Conclusion

Overall game mechanics play the most vital role in player enjoyment. Ensuring the rules, challenges and objectives fit your game world and are fair to the player along with adding subtense to your game is crucial for any game that aims to create a good game. However, which method to pick for each of these pillars is subjective and can often be the hardest part of game design due to the conversation around should the game be harder, or should they hold the players handless.

Along with this ensuring the player is sufficiently motivated through character progression or story hooks are extremely important to give the player engaged and happy to be being your game.

Finally, how you introduce game mechanics can be as important as the mechanics themselves. Without sufficient introduction, the player might be lost or find areas more difficult than they should be not to having appropriate knowledge of the game mechanics.

3.16 Conclusion

Looking at all this, a pattern can be seen in games that can provide good solid gameplay. From one perspective, there are the Jungian Archetypes which can help a player feel a connection to the protagonist, and how these archetypes are dealt with in interactive games can provide a solid understanding of plot in a branching story type game.

Next, looking at these branching stories and how they are constructed by using a flowchart you can use a Branching Narrative Structure to keep track of the actions of the player throughout the interactive game. Which give an excuse for the player to replay the game again and lets them try out multiple story paths in the narrative that leads to a different outcome each. Which can be enhanced by the mechanics in the game.

By establishing that your game has innovative or fun game mechanics, an ample amount of motivation to continually play your game to a certain number of hours or to a certain point in the story and giving the player relevant information that they need to in order to play the game is what helps create solid gameplay. Any game that has a strong stance in each of this three-part will a sure-fire successful among gamers.

4 System Design and Configuration

4.1 Architecture

During the design phase of the project, the team had two options available as a starting point. On one hand, the project could be written from the code of previously coded games that were learned over the course of the semester in college, or, alternatively, built from the ground up. The latter approach would allow for flexibility while coding. The project took flexibility over being confined to a specific type of gameplay. While designing the architecture of the game itself, it was easy to identify certain aspects that could be managed with a specific object that could be then used by the main object. Due to this, one object holds the most functionality. The Game object co-ordinates all the manager objects and conducts the interaction between each object. Each aspect of the game that would be defined as repetitive was outsourced to a manager.

The Simple Fast Media Library was the backbone of most objects, and this library handles a lot of graphics, window controls, keyboard inputs, sprites, text, fonts, and game functionality, for example, the game loop which occurs in a function that runs while the window is open. The team had to learn SFML and get to grips. There were a few issues with this library, however, this was mainly due to installing the library and setting up additional libraries and binaries.

The Collision detection code came from an external source and was part of our research for collision detection on the circular maze as opposed to a solid angled maze. This code was easily integrated into the project and there was no need to modify any of the code provided by the external source.

Everything involved in changing level and levelling up was handled in a Level Manager object. The text box which provides context to the game was handled in a Text Manager object. The loading of sprites and sounds were handled with a Sprite Manager and a Sound Manager. A Story Engine controls the state of the story, which section the player has currently achieved and what to present the player next. A Clue Manager controls how clues are created and hold individual pointers to each Clue object. The State Manager controls the changing of states,

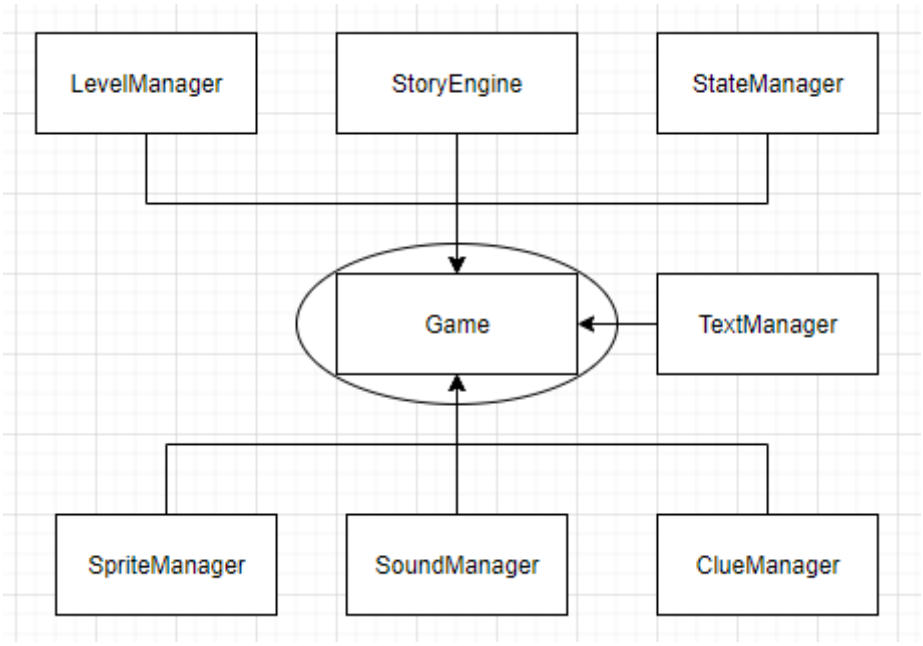
holding an *Enum* of states and sub-states. Finally, an Event Handler to handle events that occur within the game loop.

The game loop itself was designed off Unity's update function. Inside the Game object, the function that loops while the window is open is locked in a separate method called *GameLoop* and every iteration of this calls a blank Update function, so that the update function is clean and easy to read. This mechanic allows for simplicity and prevents any member of the team from accidentally breaking the *GameLoop* and keeps the original *GameLoop* truly clean.

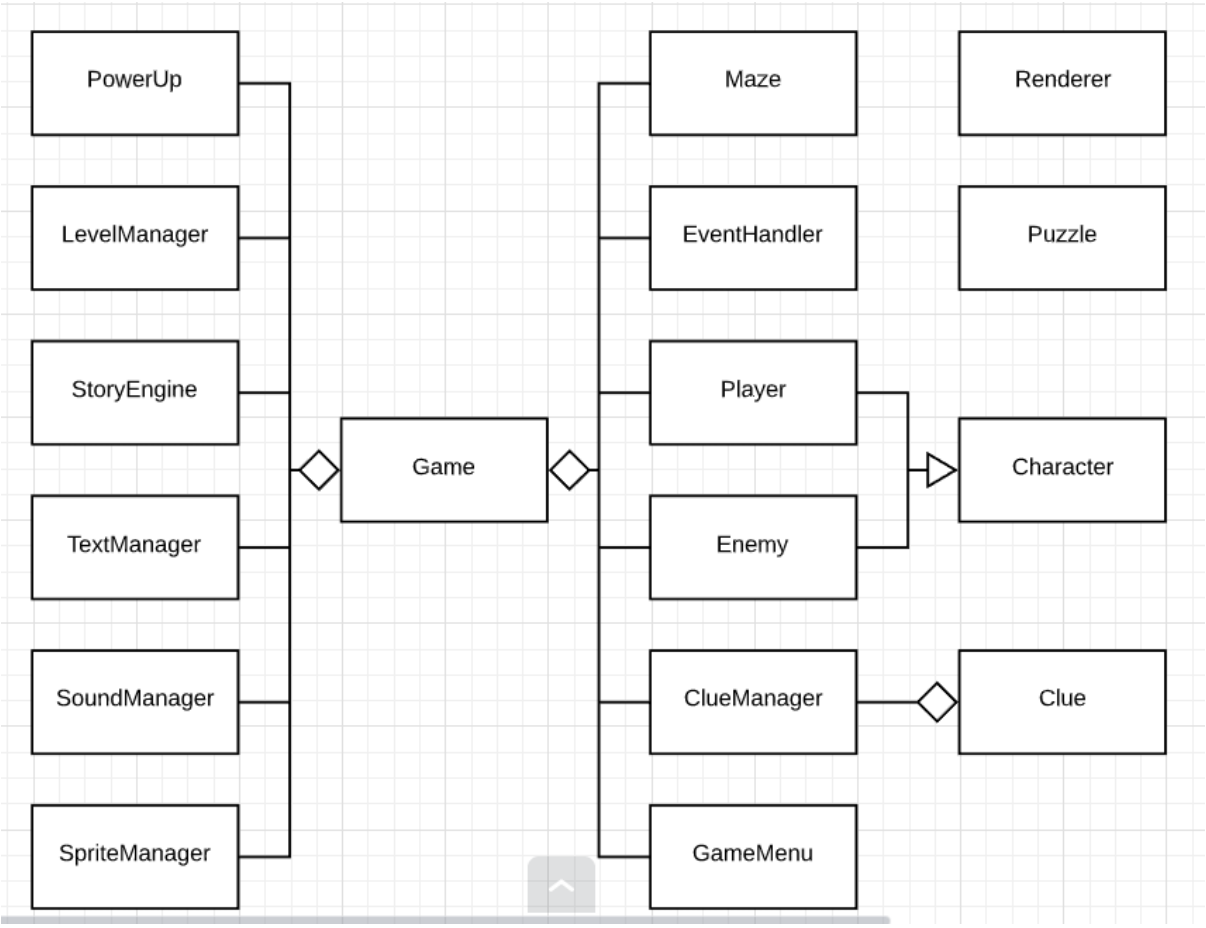
The design of the Game object allows for an easy debugging as any issues generally occur within the managers and finding solutions to issues can be divided out easily from this. The managers also create obfuscation between the main functions and the functionality of those functions.

Describe your approach to solving the problem and, where relevant, include sub-sections on

System Architecture



Class Diagram(s)



Sequence Diagram(s)

Actual code is to be provided on an accompanying and suitably labelled CD.

4.2 System Design (Projected)

During the System design phase in the infancy of the project, the movement and collision was a key aspect. Given that the maze design was circular rather than square, a simple movement and collision system couldn't be used as it had been done in previous projects. The collision was created from a pixel-perfect-collision detection algorithm that was sourced online. This allowed the system to detect if a pixel from sprite A was colliding with any pixel from sprite B ignoring transparency. It was decided that the player would have 8 direction movement as opposed to 4 so that circular corridors could be navigated easier with diagonal movement. Initially, a separate collision handling object would be used as a manager to handle collisions that would occur, so that adding collision checks would be simple.

While designing the system, the initial design was to have three types of enemies with different types of behaviours; up close, ranged, and a boss. The melee enemy would follow the player as the player got within a certain range. The ranged enemy would remain stationary and fire projectiles at the player. The boss was designed to be presented at the end of each level and would have a mixture of ranged and melee attacks. The types of pickups the player could get in each level was decided on at the start of the project. There would be a visibility pickup which would allow the player to see more. A health pickup would increase the player's health. An attack pickup would increase the player's attack strength. An armour pickup would allow the player to take more damage, and finally a clue pickup to help solve the murder mystery.

At the beginning of the project, the death system would put you back to the start of the maze, clearing all pointers and resetting player's statistics, and allow you to see the result of the murder mystery that was put forward to the player during the introduction scene. Tying this to the story initially meant that the player's husband dies as a result. The attack system was designed to be implemented as a Dungeons and Dragons type attacking system in real-time. Where the enemy and player had an armour class, damage dice, and a modifier to the attack. The player and enemy would also have high health which would be displayed as hearts for every 5 health they would have. The attacking function would be called upon pressing the attack button and to be implemented close to the code in the figure below.

```
int attackRoll = rand()%20 + modifier
if(attackRoll > enemy.ac)
{
    int damage = rand()% damageDice + modifier
    enemy.health -= damage
}
```

Producing the architecture of the system during the initial design phase of the project, the main game object would hold the main functionality. The game object would be a class of its own and the main method would create the game object and execute the running of the game. The game object would create the interaction between different objects and contain the main manager objects. The game object is projected to be large but essential.

The changing of states would be handled with a state manager. Each time a state is changed a different scene is rendered. The design of this was a choice between two options, either each state would have its own SFML RenderWindow, or it would use a master RenderWindow

created in the game object. The team decided that each state would use the master Render-Window.

4.3 System Design (Evolution)

As the project grew from its infancy the movement of the player remained as it was initially projected from the design of the system; 8 directional movement. A design decision which wasn't implemented was the handling of collisions in a separate manager object and was opted to have collisions handled close to where it was needed in the system overall. Some changes that were made was that the sprite would change on 4 of the 8 movement axes. This created a slight issue with the pixel-perfect collision as changes in sprites could often cause the player to become stuck. However, this was solved with creating a collision sprite which composed of all states of the player sprite. This was then rendered underneath the back-ground sprite. As shown in the figures below.

```
window->draw(*player.collisionSprite);
window->draw(*background);
```

When time started to become an issue, many of the enemy types were put to the side as a 'nice to have' feature after the very first melee enemy was created. It was essential that the maze had at least one enemy type, but not all. The melee pathfinding was a simple method of finding the difference between the enemy's x and y position and gradually reducing or increasing them as required as shown in the figure below.

```
if (playerY > m_position.y)
{
    m_position.y = m_position.y +
        m_speed * elapsedTime;
}

if (playerY < m_position.y)
{
    m_position.y = m_position.y -
        m_speed * elapsedTime;
}
if (playerX > m_position.x)
{
    m_position.x = m_position.x +
        m_speed * elapsedTime;
}
if (playerX < m_position.x)
{
    m_position.x = m_position.x -
        m_speed * elapsedTime;
}

//Move enemy sprite
characterSprite->setPosition(m_position);
```

The pick-up and power-ups that were decided upon also began to decrease as time became an issue. The attack and defence objects were cut out of the project and only the visibility and the health were implemented along with the clues as this was an essential part of the game, and a core feature that could not be ignored. The clues were handled by a clue manager as were the pickups. The visibility, in the end, used the countdown object that was created for use in the project.

After attempting to implement the attack system, this was chosen to be changed. The Dungeons and Dragons system that was proposed was designed to fit a turn-based attack system and did not suit the real-time combat system that was in place. As such if an enemy collided with a player sprite the attack function would be called, and a cooldown timer would be started. The player and enemy health were changed to smaller numbers in response to this.

The death system was implemented close to the end of the project's life cycle as it required the clue manager and the story to be implemented first. However, the initial design did not change much if at all; the player dies, the end scene is presented showing the player the

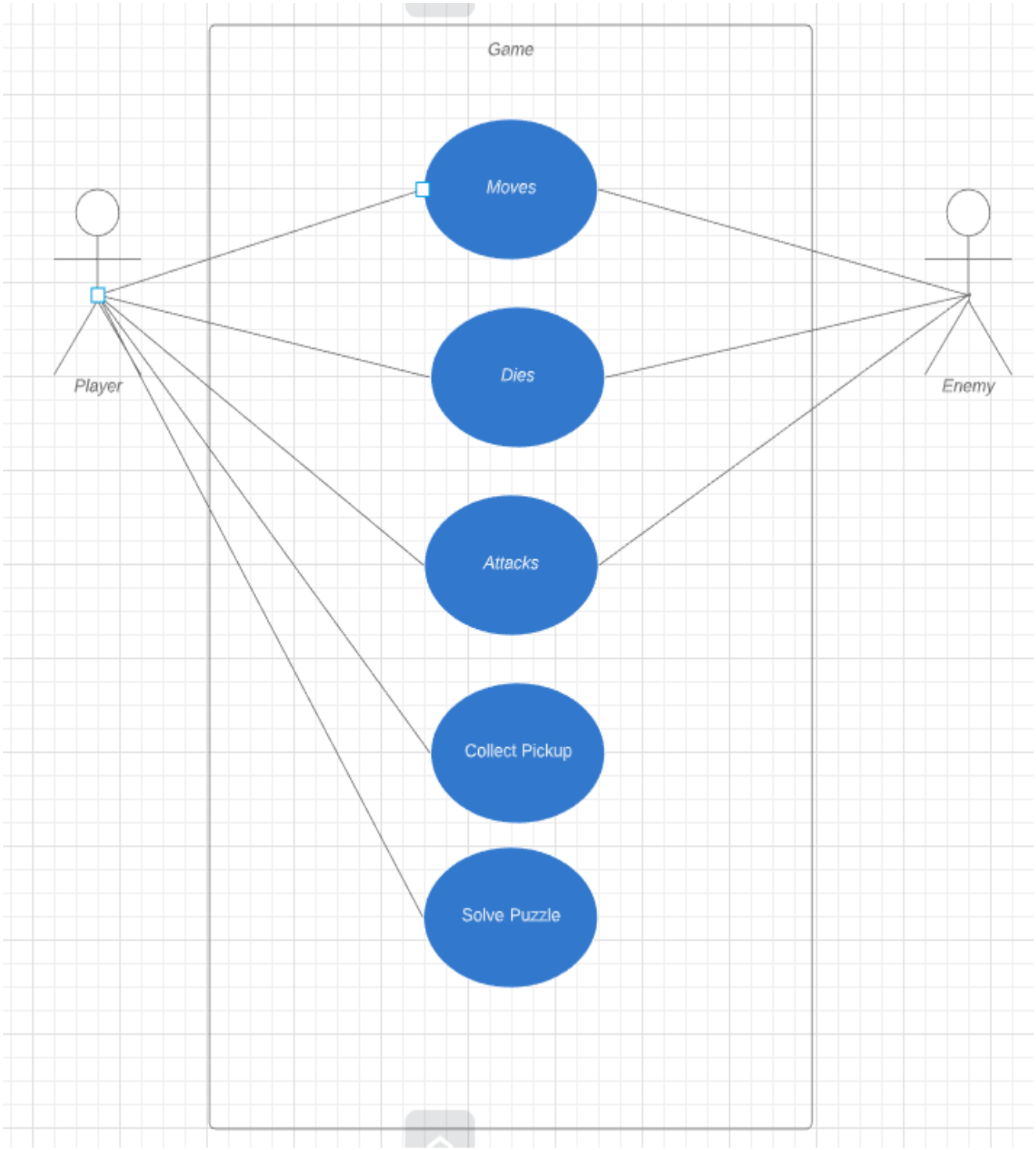
solution and asking if the player wants to play again. The revealing that the husband dies was made to be a story change where it would happen regardless as such the concept of the husband dying if the player dies wasn't implemented.

Towards the end of the project's coding lifecycle, the game object managed to get large as projected, however, the size was bigger than expected. A lot of functionality was placed into the Game object and as such, the different coding styles of each member created messy code. Many functions were separated into different files to allow for work to be completed without needing to interfere with the backbone of the Game object. These were grouped with the naming convention; Game_*.

The state manager did not hold functionality for different states in the end and instead was an object to organise the Enums of the states of the game itself. This made states easier to handle and create on an as-needed basis. The main state Enum held core game states, while a substate held story states as shown in the figure below.

```
enum STATE
{
    MENU = 0,
    PLAY = 1,
    PUZZLE = 2,
    PAUSE = 3,
    HOW_TO_PLAY = 4,
    NEXT_LEVEL = 5,
    GUESS_KILLER = 6,
    STATENUll = 7,
    DEATH = 8

};
enum SUBSTATE
{
    OPENING_SCENE = 0,
    PICKUP_CLUE = 1,
    OPENING_PUZZLE = 2,
    MENU_EXIT = 3,
    NUll = 4
};
```



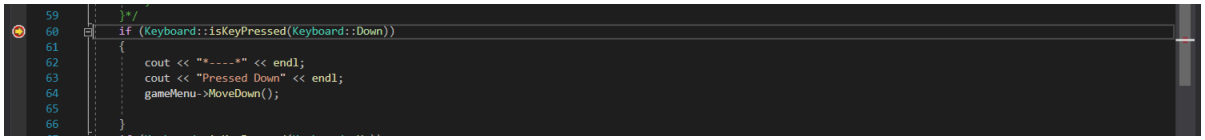
5 Testing & Implementation

Throughout our project, several methods were used to ensure that each feature that was implemented would not cause any errors further on in the project or any runtime errors. Methods that were used include Whitebox testing and Blackbox testing which includes unit testing such as execution operation testing mutation testing,

Code coverage: Whenever a new feature is implemented and needs to be tested each function within the feature is tested. Methods used to test these include utilising the breakpoint debugger in visual studio. This allows us to walk through our program step by step showing in what order the program is running along with what values it changes. The debugger allows us to see any given variable is used in a function and its current value at that point in the program. For example, for the main menu a variable being used to determine what option in the main menu is being selected.

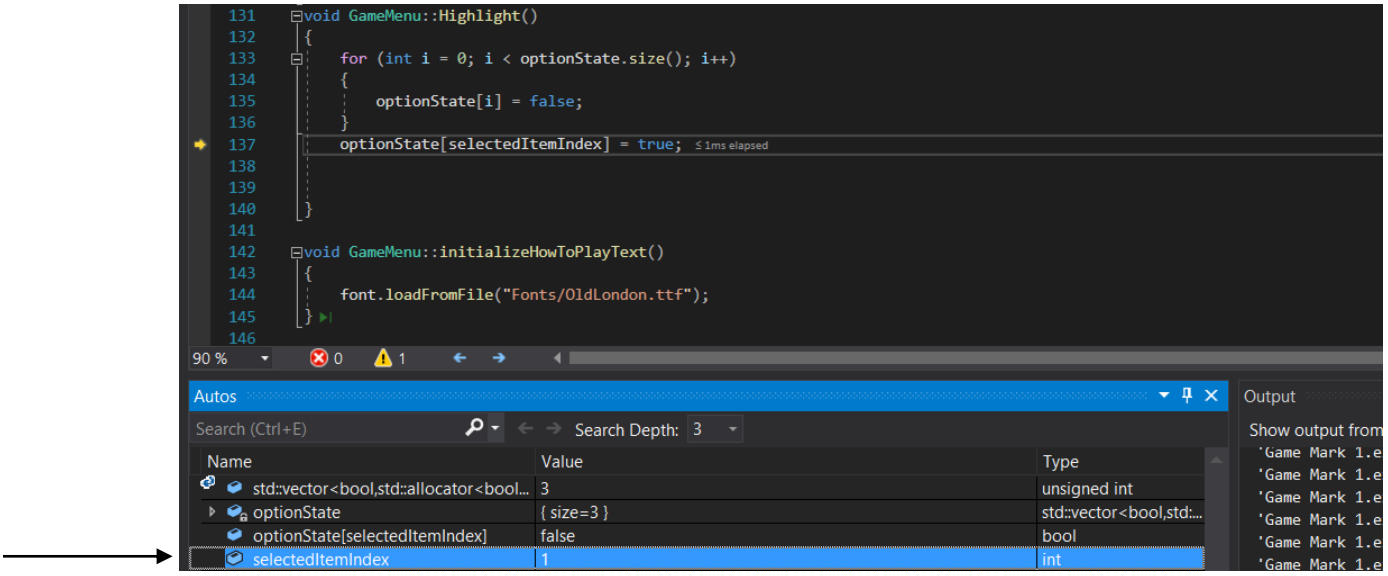
Figure 2- Code Coverage

Breakpoint



Creating a breakpoint here will stop the program and jump back into the visual studio and execute each line of code on each click step in or step over. In this example, the value of *selctedItemIndex* should be and can be checked in the debugger once it reaches the **MoveDown()** function.

Figure 3 - Checking if the variable is storing the variable, that is expected



This shows that the current value of the *selectedItemIndex* is 1. This means options are selected. Once the program loops one more time the highlighted text will change.

Test cases

Along with these, the team created test cases to test if features behaved the intended way. This allowed the team to ensure each feature was working from the user's perspective. Test cases were also used to ensure that the program was working to the specifics that any members of the game had given to the part of the system.

Figure 4- Test cases

No	Action	Input	Expected Output	Actual Output	Test Result
1	Test if the how to play option in the main menu changes screen to how to play	Enter once the how to play is highlighted	Change screen to how to play	Brought to the how to play screen	Pass
2	If pressing b in the how to play menu returns us to the main menu	Press b during the how to play screen	Returned to main menu	Returned to main menu	Pass
3	Check if pressing e near enemy results in the enemy getting hit	Press e close to the enemy	Enemy killed and removed from game	Enemy removed	Pass
4	If pressing enetrer when the play button is highlighted starts the game	Start the opening scene	Prints message to screen and changes scene	Text appears but prints over the main menu	Fail
5	Exit closes program	Press enter when exit is highlighed during the main menu	Program closes	Program closes	Pass

Once a member of the team had run test cases for whatever feature they were working on for any said week there would be a discussion around what errors/bugs said person encountered during their testing. The error would be analysed, and possible solutions would be suggested. Depending on the severity of the error the team would suggest if it was worth the time investment to fix the bug or if it would be better to invest that time working on more

pressing issues. Once this was done the error would be given a level of priority and would be added to the following week's sprint sheet.

Boundary testing was done specific gameplay elements. This type of testing is done to confirm the scope of any said variable. For example, boundary testing was used to test the enemy Ai. Enemies are only meant to approach the player if they get within a certain range in this case it was 75.

Testing allowed for a greater level of quality assurance and ensured the games features did not interfere with others.

The game has potential for further improvements in many areas that the team wish they could have worked further on. Examples of this include the puzzle mini game not being implemented due to time constraints along with other features such as further enemy AI and a longer story.

Given more time the team would expand on the guessing system to allow for more options along with different levels of difficulty for the player to choose from which would increase the damage output of enemies and fewer clues to make the mystery harder to solve.

5.1 Conclusion

In conclusion based on the implementation and test it was found that if we try to test to if the play options is the main change to how to play and you test by press enter into the how to play screen then it will pass the test and it bring the participate to the how to play screen just as they expect result predicted.

The next test done was to check if in the how to play menu by pressing the b button on the keyboard would it return the participant back to the main menu, It is revealed that the test had pass and by pressing the b key it does return the user back to the main menu.

The next test done was by checking if the player presses the spacebar will they be able to kill an enemy and remove them from the game, the test does pass and it is reveal that once the enemy is killed a sound bite is played which tell the participant that the enemy is no longer inside of the game.

The next test done was to see if we press the play button would a textbox pop up, original it didn't work but after time working on the textbox, it eventually worked after adding and removing section from the code until it eventually worked.

The next test was to see if the program would close if we pressed the escape key. After doing the test it was found that yes, the program would exit after the participant pressed the escape key.

No	Action	Input	Expected Output	Actual Output	Test Result
1	Test if the how to play option in the main menu changes screen to how to play	Enter once the how to play is highlighted	Change screen to how to play	Brought to the how to play screen	Pass
2	If pressing b in the how to play menu returns us to the main menu	Press b during the how to play screen	Returned to main menu	Returned to main menu	Pass
3	Check if pressing e near enemy results in the enemy getting hit	Press e close to the enemy	Enemy killed and removed from game	Enemy removed	Pass
4	If pressing enetrer when the play button is highlighted starts the game	Start the opening scene	Prints message to screen and changes scene	Text appears but prints over the main menu	Fail
5	Exit closes program	Press enter when exit is highlighed during the main menu	Program closes	Program closes	Pass

6 Critical analysis

6.1 What Worked

While experimenting with the Clue system we went through multiple interactions, we first wanted the clue system to first work a mini data base inside our project that allows us to pull from the information from the database we made and the randomly create a combinations to allow us solve the mystery inside of the game. It would have been a huge that bases that would have considered the feature of each person who was a potential culprit.

But after a few meetings and getting opinions from our peer outside of the group we decided to downside the clue system and simplify it to the point where it works similarly to the game of Cluedo. We were initially going to have seven individual who had three items on them that would link them to the murder, the three items would be randomly pick from a pool and it would be assigned to a specific individual. But it found to be big and tedious to do.

While brainstorming again, it was found that it would be more beneficial to have it so that the specific number of suspects was shorten and that the number of items was shorten, which allowed more brainstorming. After a few more brain storming session the clue system was finally complete, and this was done by adding the opposite of the seven deadly sins to clue system to give a motive to why a specific individual would go out of them to murder the plot device in the story.

By adding that little bit to the clue system a few goals were accomplished, the clue system was finally at a manage state where it was easier to predict what type of solution would come out, it made easier for the audience of the game to guess who the culprit is at the start of the of the level and finally it gave story and logical reasoning on why the plot device was killed a built a characters after this.

6.2 What Didn't Work

While performing research into the android controller and examining the steps it would take to implement it proved to be a feature that could not be implemented. The interaction between C++ and the Android Studio language was too comprehensive to add. It would have required the team to learn and understand the new API that none had seen before. This would have added a lot more time to the project, and time management was not on our side. In addition, the android controller would have required that an accompanying application to be made to handle user input on the controller side and to send that information to the target machine through Bluetooth or wireless connection. The project works without this feature and is a nice to have addition.

The puzzles that had planned to be involved was a feature that had to be cut out halfway through the project's lifecycle due to time constraints. The group was running out of time, and after a long debate on this, it was decided to, unfortunately, stop working on the feature. The team decided that the game would function as intended without this feature. The variety of puzzles that were required to be implemented as a weak point. The feature would have required more art to be made and more assets that the team did not have the luxury of creating. This was an unfortunate cut as the team had put work into getting this feature completed; the decision to cut it came at a halfway point of the feature's lifecycle.

The melee enemy's pathfinding and artificial intelligence weren't as efficient as the team designed from the beginning of the project. When a player comes in a range of the enemy, the enemy begins to move closer. However, if the player were to move with a wall between them and the enemy the enemy will be trapped as it cannot navigate efficiently around the walls. This was slightly solved by decreasing the range of the enemy's acquisition range. The implementation of a* algorithm was not a good fit for the type of design that the game was, as a*

required the collision detection that was built based on tile-based games. The reduction of the enemy's target range solves the issue from a player's perspective and gives the illusion of a better pathfinding algorithm.

The team was on a constant struggle with finding art assets. One member of the team had an artist who was willing to help, however, they too were on a tight schedule. Though they could produce good art, the team found themselves always having to chase up the art. This proved to have a negative impact as art was a major feature of the game that would bring it to life. The art, in the end, had to be created by those who hadn't a lot of artistic talent or sourced from online free sources. Trying to find these art assets and creating art assets added to some time management issues as it was a necessity to be completed. The art that was in the game at the end was a mismatch of styles that creates a messy look to the game at times.

7 Conclusions

7.1 System Design

Observing the system design's evolution throughout the project the changes made to each part has made an overall positive impact. The movement system works well, though at some point the movement of the player has a slight jitter, and the team isn't sure why that happens. The 8-direction movement works well with the design of the maze walls, and players can effectively move around the circular corridors put in place. Movement of enemies works but could be designed to have better pathfinding capabilities.

Collisions work as intended; any time where a collision between two sprites is needed, the collision method is called, and it will return true if a pixel overlaps another. Though all collisions were designed to be implemented in a collision detection object, this wasn't needed as the team didn't feel like it was necessary due to collisions needed to be handled close to the code that needed the detection.

The lack of different enemy types is something that provides a negative impact on the game. Ideally, there should have been more variety from a player's perspective, however, due to time management issues, the team had to cut out some features and the different enemy

types was one such feature after the initial enemy had been created. The melee enemy has basic functionality and pathfinding that is enough to achieve gameplay.

Another feature that was cut back was the player's pick-ups. Initially, the scope of these pick-ups included some player buffs to their attack and defence. However, as the attacking system and enemy systems were scaled back due to time management constraints the pickups were also scaled back, and those that remained were the visibility and the life pick-ups. These work well with the system and provide functionality to the game which adds an enjoyable experience by allowing the player to see more for a brief period and to gain more life back.

The attack system, though different from the initial design, works efficiently. It can be greatly improved with some animations and visual effects to make the experience a lot more enjoyable. What the attack system lacks is feedback to the player; if the enemy and player were to simply flash, this could provide enough feedback. This was part of a list of features that did not make it into the final iteration of the game due to time constraints.

Another system that required feedback and animation was the death and dying frames. Though the system differed from its projected state it was appealing due to its tied in nature with the story and the murder mystery clue system. With more time, this would have improved on and given a lot more focus and story elements such as the mysterious stranger appearing once more.

The main game object, on analysis, could be constructed in a neater and more functional manner. At the state it got to at the end of the project's lifecycle, it became a large mess of code. Though it was a large mess it was also a stable system. This is due to the many parts which had been put into separate files for the sole purpose of making edits and additions easier.

The state manager became an object to hold Enums of states rather than actual states themselves. Though this worked, it could have performed its role better by holding the render of each state to be applied and other valuable resources to those states. The state manager should have become a state engine rather than a manager, controlling how the game interacted with other states rather than the game controlling the state changes.

8 Bibliography

2K Games, 2007. BioShock. s.l.:2K Games. A.L, C., 2005. Story and Narrative Structures in Computer Games, Visby, Sweden: Gotland University. Ang, C. S., 2005. Ang, C.S. (2005). Rules, Gameplay and Narratives in Video Games. Simulation & Gaming: An Interdisciplinary Journal of Theory, Practice and Research. , City University London: s.n. Anon., n.d. Anon., n.d. Videogame Objectives, s.l.: s.n. Arneson, G. G. a. D., 1974. Dungeon and Dragon. Original ed. s.l.:TSR, Wizards of the Coast. Bandersnatch. 2018. [Film] Directed by David Slade. United States of America: Netflix. Barone, E., 2016. Stardew Valley. Seattle: Chucklefish, ConcernedApe. Berube, D., 2019. Motivate player for better engagement and retention, s.l.: gamasutra . Bethesda Game Studios, 2008. Fallout 3. s.l.:Bethesda Softworks. Bethesda Game Studios, 2015. Fallout 4. s.l.:Bethesda Softworks. Bethesda Game Studios, 2018. Fallout 76. s.l.:Bethesda Softworks. Bethesda Softworks, 2011. The Elder Scrolls V: Skyrim. s.l.:Bethesda Softworks. BioWare, Edge of Reality, Demiurge Studios, Straight Right, 2007. Mass Effect. s.l.:Microsoft Game Studios, Electronic Arts. bma5176, 2014. The best way to introduce controls and game mechanics, s.l.: s.n. bycer, J., 2012. Motivating Mechanics in Game Design., s.l.: Gamasutra. Campbell, J., 1975. Monomyth. Novato, California: New World Library. Electronic Arts Inc., 2000. The Sims. London: Electronic Arts Inc. Erwin, S., 1913. Die gegenwärtige Situation in der Quantenmechanik, s.l.: Springer Science+Business Media. Fox, T., 2015. https://undertale.fandom.com/wiki/Main_Page. s.l.:Toby Fox. Fox, T., 2015. Undertale. s.l.:Toby Fox. GameCentral, 2019. The artistic significance of difficulty in video games – Reader's Feature, s.l.: Metro. Infocom, 1979. Zork. s.l.:Infocom. Josiah Lebowitz, C. K., 2011. Interactive Storytelling for Video Games. 1st ed. s.l.:Taylor & Francis. Jung, C. G., 1991. The archetypes and the collective unconscious. London: Routledge. Lindley, C., 2002. The Gameplay Gestalt, Narrative, and Interactive Storytelling, CGDC Conference: s.n. Lindley, C., 2005. Story and narrative structures in computer games, Visby, Sweden: s.n. Lionhead Studios, 2010. Fable 3. s.l.:Microsoft Game Studios. 16 McIntosh, B., 2010. Nonlinear Narrative in Games:- GameCareerGuide.Com.. [Online] Available at: www.gamecareerguide.com/features/882/nonlinear_narrative_ingames.php [Accessed 31 10 2019]. Metacritic, 2008. Fallout 3. [Online] Available at: www.metacritic.com/game/pc/fallout-3 [Accessed 19 10 2019]. Metacritic, 2015. Fallout 4. [Online] Available at: www.metacritic.com/game/playstation-4/fallout-4 [Accessed 31 10 2019]. Metacritic, 2018. Fallout 76. [Online] Available at:

www.metacritic.com/game/playstation-4/fallout-76 [Accessed 31 10 2019]. Metacritic, 2019. Metacritic. [Online] Available at: www.metacritic.com [Accessed 31 10 2019]. Namco, 1994. Tekken. s.l.:Namco. Nelmes, J., 2011. Analysing the Screenplay. New York: Routledge. Nelson, P., 2015. Designing Branching Narrative. [Online] Available at: https://thestoryelement.wordpress.com/2015/02/11/designing-branching-narrative/ [Accessed 31 Ocotober 2019]. Nintendo, 1985. Super Mario Bros. s.l.:Nintendo. Nintendo, 1993. Legend of Zelda: Link's Awakening. Tokyo: Nintendo. Novak, J., 2012. Game Development Essentials: An Introduction. 3rd ed. s.l.:Cengage Learning. O'Bannon, D. e. a., n.d. Alien. [Online] Available at: www.imdb.com/title/tt0078748/ [Accessed 31 10 2019]. Pajitnov, A., 1984. Tetris. Moscow, Russia: s.n. Raney, A., 2011. The Role of Morality in Emotional Reactions to and Enjoyment of Media Entertainment. Journal of Media Psychology. Ryan, M., 2009. From Narrative Games to Playable Stories: Toward a Poetics of Interactive Narrative. Storyworlds: A Journal of Narrative Studies, Volume 1, pp. 43-59. Sandy Louchart, R. A., 2005. Managing a Non-linear Scenario – A Narrative Evolution. In: G. Subsol, ed. Lecture Notes in Computer Science. s.l.:Springer, pp. 148-157. Schrödinger, E., 1935. Schrödinger's cat. [Online] Available at: https://simple.wikipedia.org/wiki/Schr%C3%B6dinger%27s_cat [Accessed 31 Ocotober 2019]. 17 Sciutteri, M., 2018. Interactive Storytelling: Linear Storytelling. [Online] Available at: gamedevelopment.tutsplus.com/articles/interactive-storytelling-part-2--cms-30273 [Accessed 31 10 2019]. Sparknotes, 2019. SparkNotes: Of Mice and Men. [Online] Available at: www.sparknotes.com/lit/micemen/ [Accessed 31 10 2019]. Square Enix, 1997. Final Fantasy VII. s.l.:Square . Star Wars. 1977. [Film] Directed by George Lucas. s.l.: Lucasfilms Ltd. 20th Century Fox. Telltale, 2012. Walking Dead. s.l.:Telltale. The Truman Show. 1998. [Film] Directed by Peter Weir. s.l.: Scott Rudin Production. Ubisoft, 2007. Assassin's Creed. Montreal: Ubisoft. USA, S. S., 2005. Shadow the Hedgehog. s.l.:Sega. Valve Corporation, 2011. Portal 2. s.l.:Valve Corporation. Wand, E., 2002. Interactive Storytelling:The Renaissance of Narration. s.l.:s.n