

Computer Games Development

Project Report

Year IV

David Whiteford

C00204740

19/4/2021

[Declaration form to be attached]

**Contents**

[Acknowledgements 2](#_Toc54714373)

[Project Abstract 2](#_Toc54714374)

[Project Introduction and/or Research Question 2](#_Toc54714375)

[Literature Review 2](#_Toc54714376)

[Evaluation and Discussion 3](#_Toc54714377)

[Conclusions 3](#_Toc54714378)

[References 4](#_Toc54714379)

[Appendices 4](#_Toc54714380)

# Acknowledgements

I would like to thank the following people who assisted in completing this project including;

John Doe of ACME who kindly agreed to …

I would also like to thank ICME for use of ….

Use this template when writing your research report. As a rule of thumb, the report should be of the order of 10 pages (about 250 words/page).

# Project Abstract

The chosen project will be on procedural generation of a maze-like dungeon system that will be made of rooms and will generate when the player enters a cave in the over world. These will generate a 2D array of different values that will represent the different tiles in the dungeon system.

The maze will be generated with a number of different rooms that the player can enter and these can be connected with a number of corridors that connect the rooms.

The procedural generation system will create rooms, corridors, walls and background tiles to create a dungeon system that will be different each time a user enters these dungeons. These will be used in the game as high loot areas and are procedurally generated so as to have a different layout every time a player enters the tunnels.

I used the version in the game as it creates a layout of different characters. These are laid out to be like rooms and corridors that the player can play in. The use of characters allows a handy way to create these dungeons and are laid in a way that makes them easier to work with.

The system creates these rooms and stores them in a vector and these can easily be transferred to a 2D array that I used to create tile maps and allows the system to be created with a tile map and to then draw these with ease.

The tile map class in the first build took a 2D array and in a switch checked the type of value in the array and depending on the value it created a tile with a certain sprite and placed it in the game.

The procedural generation system creates the values in a certain order so as to make it easier to add them to a 2D array form a vector and to then change it from a char to an int and store them in the 2D array. They are then passed to the tile map to create the tiles and to also draw these tiles so as to make it easier to use.

There will also be functions that will also be a generation system that will decorate the rooms that were generated and this will be done after the dungeon layout has been created. This will create special and uniquely decorated rooms.

The special room will be boss rooms, worship rooms etc and these will have a significant importance to the story of the game for example a worship room might have a statue of the boss that will act as a warning the players. These rooms will be completely changed and the tiles on the background will be different.

The unique rooms will be rooms that will have decorations that will be unique to that room and these are a library with a bookcase and chairs, a feast room with table and chairs, prison with jails, burial room with coffins and others that the programmer can come up with. These will have unique ways to set them up and will have functions in the dungeon class to set these up.

The procedural generation system will loop through each room till most are decorated and at first it will create all the special rooms that make up the boss rooms and other as these are the ones that need the biggest room and so it’s better to create these early so as to not have random décor in them at beginning. These are created and the rooms are done so are deleted from the vector.

Then the unique rooms like the library and the feast room are created. Like the special rooms they can need specific sizes so are done here to allow them to get these and to make decorate their rooms without an issue. Some are deleted when they are done.

Lastly all the rest are decorated at random and have different decorations in them from potions to skeletons and others. They are also deleted so as to prevent them from being decorated again as this may cause some rooms to be not decorated at all.

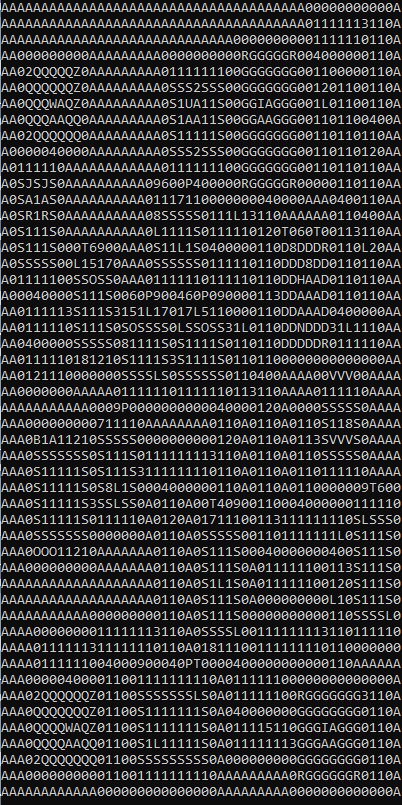
This overall is the main reason that the version of the procedural generation system that I used.

# Project Introduction

The project that I have chosen to do this year as the main is a project that will have a dungeon procedural generation system that will be the focus of the project. The reason for this choice is due to an interest to learn these systems and create a game that utilises a procedural generation system

The game will be designed like the original Legend of Zelda and will be based on a map that the players can explore. The overworld will be the main world that the players can explore and there will be enemies here along with a town where the players can buy better gear and upgrades. This map will use a tile map to generate the world and will also have different terrains with this map.

Figure 1(Overworld)

The procedural generation will be used in the creation of dungeons that the player can find in the overworld and then will explore to find the end where a boss might be there for them to fight. The boss could then drop loot that will make the player have different experience as they play more.

The goal is to give an understanding of procedural generation to myself and anyone who may find this report helpful in the area of procedural generation for the different dungeons that can come from such a system.

My hope is to create a cool dungeon system that will be made of different designs. The dungeon will have walls doors and floors that will make up the dungeon and the system will generate a possible layout for these.

My goal is to do research on the procedural generation of tunnels and to learn the best and quickest way to generate the tunnels in the game so as to make a quick and cool system that can generate the tunnels.

The use of SFML and the creation of my own tile map class to go with the procedural generation will also be included to take the data from the procedural generation and then to create a tile map that will make up the level and will allow the player to explore.

Figure 2(2D Generated Room & Halls)

The system will also include a collision system that will create collisions for the walls and the obstacles and make it so the player can’t phase through these.

I will also create my own spawning system that will use the system from the procedural generator to create different rooms with different decorations and these will be random, special rooms and unique rooms.

There will also be the use of a random generation using the uniform int generation in the standard library and this will generate random values between values that were passed into the function for random int generation.

# Literature Review

# The main goal is to create a procedural generation system within a Legend of Zelda style of game that will generate a dungeon when the player enters the caves / dungeons from the over world.

The goal is to create a system that will have different layouts for the dungeon and each time there will be a different style and layout to the dungeon. There will be different layouts such as a chance that there will be a room or a corridor and these can be changed to allow for more rooms.

(MindControlDx, 2018) has created a system that helped me create the basis for the procedural generation. There is a room chance and a hall chance and this will determine what the next room will be. Then it will start the room and give it a with and a height and this will be a number that was generated between two values.

Then once the room was created the exits of that room are created and based on the direction it will determine the exit. Mine has it so that the non-halls will always have an exit in all directions most of the time while the halls will not be this way.

(MindControlDx, 2018) also has an example of how to place things in an already generated room and I built on this by allowing the rooms to be fully decorated and to also allow the tiles to be completely changed after a room was completed. I also allow the halls to have décor and create a dungeon that will allow the player to spawn and explore the dungeon.

While (MindControlDx, 2018) version stored the data to be outputted as a text mine is instead stored in a vector an passed to a tile map class so as to create the level.

The dungeon will also have loot for the player to use elsewhere in the game and they can also have bosses that the player can fight. These can be unique bosses such as bosses that can be only defeated a certain way.

The loot will be a random spawn that will loop through the rooms and then when it’s created all the special rooms and the unique rooms it will fill these rooms with random decorations some of which will be loot that the player can collect such as money pouches, swords and others.

The dungeon can also be maze like to allow the player to get lost but there could also be a way to create a way to allow the player to keep track of their position and to make it easier to get through the maze.

An example of allowing the player is to light the torch decorations in a room that the player has been in and if these are lit then it will tell the player that they have been here already. All rooms can have torches or some only have them to make it harder and these are placed in the procedural generation system and shaders can be used to create light for these.

There is also a state machine to be put in to allow the animation of the character and the different states that the player may be in. These can be walking to attacks that the player can use in the game.

The state machine will also manage the player movement and the different states that the player will have such as if the player attacks an enemy and various other. This is a minor part of the project to make it feel more like a game.

The world also included the start of a collision system. This was done by giving each of the tiles a collider and then setting a bool to determine if there was a collision. There was also a ray cast class added so as to create rays that will act as the players collision detection.

When a player ray enters a collider the bool that’s related to that raycast will be set as true and the players movement will be stopped and prevent them from moving through the colliders of the tiles that are obstacles.

All tiles that have a tag that is of type obstacle (in this case the number 1) will be given a standard collider of 16 x 16 and any others that have a special tag of 2 will have a collider of the same size as the tile. There is also a string tag that tells us the name of these obstacles and get the size from this. This was a quick collider and can be improved.

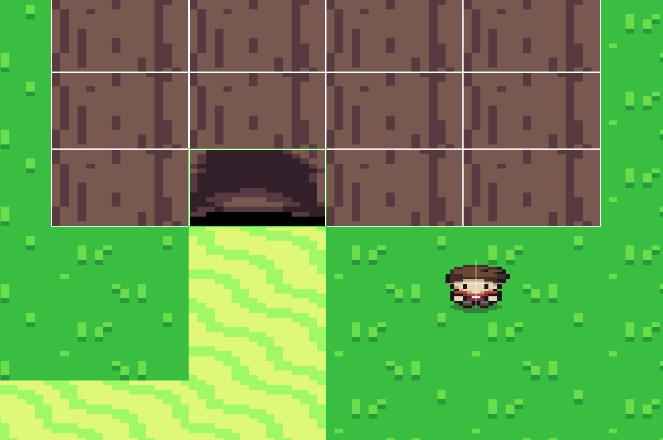
There are also triggers that are made the same way as the above colliders and they also check the number tag and if number 1 then place 16 x16 box collider around it and if 2 then check the type of collider.

Figure 3(Colliders and Ray casts)

This is checked by the string tag and the size is then gotten from this sprite and then the collider is made to be of this size.

This was later changes to that each tile has a collider and the size is given to the tile and passed to the collider when its created and placed.

# Evaluation and Discussion

Replace this text with Results and Discussion.

Describe the results using diagrams such as graphs etc. as appropriate, and discuss what the results mean.

Example: Results indicate that once the threshold gets over a certain point it significantly reduces player performance and player experience

One of the main things tested was the procedural generations effect on the performance of the creation of the dungeon. There were a few that affected this such as the time to create the dungeon increasing when the decorations were included.

There were also time increases when the system had to loop through each room and check the number of decorations in the rooms. This was solved by deleting the room once it had a certain number of decorations.

Doing checks for the creation of the rooms such as checking x, y values or direction also caused it to slow.

Overall, there were issues where the program slowed down but it ran well without match issues and the time to create dungeons was quick overall.

## Project Milestones

### ***The Game Loop***

The first mile stone was to make a game will a game loop and at the very least to have a player character that the player can control and move around. Also needed to animate the player and to get some the states for the player in.

This was to be done by the week of the 16 November 2020. The player moves and is animated using a Finite State Machine and the tile map for the overworld has been started.

More was done than what was expected and the tile map code is also nearly done as all it needs is the positions of the tiles and to also add more tiles to the world. At the moment there is only grass, dirt, water but more are too added to the tile map.

The tile map will be generated when the game starts and will only be done once.

The animations and the player movement were the focus at the start and they have a very basic movement but it works for now as the goal was to get the Finite State Machine working and that was achieved.

### ***The Dungeon***

Started the procedural generation of the dungeon system that will generate a text-based system that represents the dungeon. This was started on the week of the 1st December 2020 and has been made to generate a dungeon.

The start of the dungeon was completed and generating a dungeon each time the game was run. The dungeon has some issues that has to be worked on and these are the likes of the game not generating a dungeon every time the game is run and this will have to be fixed as the project continues.

The goal is to get the procedural generation completed and generating a dungeon for the game as soon as possible and to have it so there is at least a system to show the different way the dungeon is laid out.

There has also to be a bit of work to be done to the finite state machine so as to fix any issues that the state machine has on it. There are issues with it that need to be fixed.

### ***The Dungeon Decorations***

The next major milestone was to create the was to decorate the dungeon and to get there being basic decorations in the rooms. This was to be done after the unique rooms were created but were done without an order at first.

The goal was to create different rooms such as a boss room, worship room etc and then to create the unique rooms such as the library and the coffin rooms that will have different decorations from the other rooms. Then decorate the remainders with random decorations and this would be done last to the remaining rooms.

However, because this was started first all the rooms looked the same when decorated. This convinced me that the different and unique rooms were needed to fill the level out and make it so that there were different looks to some of the rooms.

Each room would have the same decorations and these were things like potions, money, skeletons that will make up the decorations of each of the rooms. Some were given a spawn along the walls and others were given random spawns in the room so as to make it that they didn’t spawn near the walls of each of the rooms. These were completed by 1st February 2021.

### ***Dungeon Generation Refactor***

There were a lot of issues however decorating the dungeon on the one layer and so it was going to be needed to be redesigned. The decorations would need to be stored separate from background tiles and so once the layout of the room was created the vector was copied and used to determine where tiles were and what type was at certain positions.

The background tiles and the décor tiles were stored separate and so it was easy to pass these to the tile map and to then generate the map based on these. The background was rendered first and the decorations was then rendered after to allow the decorations to render before the background tiles.

This made also made it easy to change tiles for the decorations without the need to change the tiles for the background and so allowed these to be kept separate. If a tile was to be checked then the decorations vector can be checked and changed if needed.

When these were then passed and used by the tile map, they were cleared to stop them taking up space in memory as they were no longer needed once the tile map was created in the tile map class. This were completed by 29th February 2021.

### ***Dungeon Generation Different Rooms***

The next types of room that were created were rooms that had unique decorations in them and rooms that were completely different form the original rooms.

The rooms that were completely different were rooms such as a boss room. Worship room and a statue room.

the boss room had a hole in the middle, skeletons and a health that the player can pick up and use. The hole is where the boss can spawn and the skeletons can be of pass heroes that were killed by the boss. This can be done to give it a story and to make exploring the dungeon cool and fun.

The statue room can have a statue of the boss that can act as a warning to the player of the boss and this can also be cool and would add to the story of a game.

These rooms will be generated in the dungeon and will be of different sizes and place at different areas in the dungeon. These can be generated and add to the cool look of the dungeon.

These rooms will be generated at the beginning and they will be got from the rooms vector and these will be given a certain size to be. They can be generated as many times as you want as long as they are within the size required. This were completed by 15th March 2021.

#### **Unique Rooms**

These rooms will be generated after the different rooms and will be made of different décor but will not give the player any information.

These will simply make the dungeon seem different and will make it so there are different room that will make up the dungeon. These will have unique and similar decorations to the other rooms in the dungeon and are there to make the dungeons rooms feel different

These can include rooms like the Feast room, the library, prison etc. These are generated after the boss rom and the before the room decorations are added to each of the other rooms.

They will not have any story to them and are there to simply fill up the dungeon. They were completed by 25th March 2021.

#### **Halls Rooms**

These were decorated will a small number of decorations and were made to have decorations to make them fell like they are not empty and to give colour to them as well. They were now able to have plants, money and potions in them to fill the rooms and to make it feel less empty.

#### **Other Decorations**

There were also other types of decorations added to the dungeon generator such as the ability to have chair next to potin tables and to also have chairs around the feast table.

There also exits / spawns in the walls of some of the rooms. These were in all rooms but were removed from the boss and different looking rooms.

There were also decorations added to the walls and they were the likes of lamps, pictures and chains to make the wall have decorations and to feel interesting looking.

The lamps can be used to create lights to the room and these can have their positions stored in a vector to be used by other objects.

There were also traps that can harm the player and these were set up as triggers and the tiles can be tagged as obstacles if you wanted to create impassable walls and make some of the decorations as also impassable. These can have colliders added to them to do this and can be used as well for a pathfinder. These can be stored and passed around if they are needed by say pathfinders or colliders etc.

#### **Other Work**

There was a main focus on getting the procedural generation focus working to a good degree. The system had a few issues that were ironed out at the halfway point. One of these were rooms spawning on their own with no way to get to these rooms and doors that led into the black void. These were fixed by checking the code and fixing mistakes that were made.

There was also a major refactor after Christmas that I did and this involved changing how the dungeon class returned and pass the data for the dungeon to the tile map class and at first it was a 2D array that was passed into the tile map. I had a reference to the tile map class in passed into the dungeon but that was changed so as to have a getter that returned the stores values to the Game class and then pass them to the tile map class when I set up the dungeon. The 2D array was also replaced with a vector that made it better and allowed easy passing of the values for the dungeon to the tile map.

There was also the setup of the multiple dungeon generation that was needed and this was done so that the old dungeon was cleared from the vector that stored it and everything was reset when the player returned to the overworld. The player could now enter a cave in the overworld that would have them spawn near an exit on a spawn position. When the player left the dungeon, the over world was loaded back in and the player could enter a new cave to generate a new dungeon. This was done to simulate how the dungeon generation would happen in an actual game when players were playing. The dungeon would be different every time they entered a cave. All storage for dungeon were cleared and the old one was reset to have nothing but Unused tiles to allow the generation of a new dungeon.

The decoration system also needed a number of different types of work done to make it generate a good dungeon such as placing décor tiles next to the walls. There was also work that was needed to create multiple of the same tile such as the prison room that needed a loop that started one tile from the left wall and ended one tile from the right wall.

There were also checks done to make sure that the larger tiles had unused tiles under them so as to prevent other tiles from spawning on them as the tile map was 16 x 16 for each tile and so the larger tile covered more than one tile in the dungeon generator and to a function with a nested loop was created to make those tiles in a size of two from first tile all unusable.

All tiles had to be check also if near a door and so moved or not placed at all so as to prevent the tile being placed near the trigger and to allow the player to interact with these. To do this there was a function that allowed there to be a check of all tiles around the x, y position of the current tile in the grid and then place it if it was a good place to place the tile. This was done to make placing the tiles better and o stop blocking of doors and exits.

#### **Collisions**

The collisions were done with rectangles that were around the size of the tile and placed together to create a collider that would stop the player from getting through the walls and act as an example of a collision system that can be used to stop players going through the walls.

The one I used to be a quick creation. Each tile in the tile map was given a tag one for the type e.g., for larger tiles like the table or statue. The other tag was a number ta to determine if the box was a trigger, impassable or if it was passable. These were number 0 – 3 and were used to get from the tile vector all triggers, colliders and larger sprites / tiles.

These were stored in a vector of their own and passed to the player a checked the type and the tag to then give these a collider box. This was changed to have a class of their own and were then check against the player to determine if the player was in the collider.

The player was given rays that were in the 4 directions and their length was the same size as the player. When the endpoint was detected in these the bool for that direction was set to true and all others false and the player was not allowed to move.

The same was done for the triggers and their tags were checked and depending on the tag depended on what happened. All the triggers were set to go to the overworld and allow the generation of a new dungeon.

These triggers were checked the same way as the colliders which was to check if the rays were inside a trigger and then to do an action such as out putting a message and also re set the level to the over world.

The colliders could be also drawn with the debug set to one and also turned off with the debug on. This is to show where the colliders and the triggers were placed. This was done to show triggers for the dungeon exit be at the doors in the walls and others.

The collision system was simple but can be made more advanced with more rays to have a more precise collision detection and also can be made better with more work.

***Procedural Generation Examples***

During the research of the project, I came across a number of different dungeon generation articles that talked about and gave examples of the use of procedural generation in games and more precisely the use of it to generate dungeons.

(Gamasutra The Art and Business of Making Games, 2015) talks about the use of procedural generation and how they use it to create random dungeons and how they their algorithm works. They talk about using the algorithm to randomly generate dungeons using normal distribution in the creation of the dungeons.

(Gamasutra The Art and Business of Making Games, 2015) also talk about the use of tiles in the creation of the dungeon and how they have to keep all room positions and the sizes of these to a certain multiple.

In their example the tile size is 4 so the size they have to keep to is a multiple of 4.

(Gamasutra The Art and Business of Making Games, 2015) next part is to separate the rooms as they were all generated close together and have to be moves away from one another. They use a physics engine to do this over a separation steering behaviour and so they separate the rooms so as to not have these over lapping.

(Gamasutra The Art and Business of Making Games, 2015) designs their rooms in a way that makes them ideal to the type of gameplay that they have in the game and so as to make it more ideal to the game they are making.

They also prevent the generation and separating becoming too tall by generating the rooms positions in a very narrow strip that they can separate from better and not as tall as the other way they did it. They then selected what were rooms and what were not rooms based on how big the size of the rooms was and did this by selecting rooms with a width greater than 30.

(Gamasutra The Art and Business of Making Games, 2015) then uses the Delaunay Triangle + Graph top take in points and then gives back triangles. These were then added to a graph that they had in their project and they also gave the rooms ids. They use the graph to create a minimum spanning tree. This is used to make sure that all their rooms are reachable and that they are not connected to one another and give them a good distance between them.

(Gamasutra The Art and Business of Making Games, 2015) then create hallways that connect the nodes in the graph and they add these if they are close enough on the horizontal and the vertical and add hallways if they are. These start as lines. Any rooms that are colliding with the lines are then added to a structure that holds these rooms. They then add one tile sized grid cells that make up any missing parts.

(Gamasutra The Art and Business of Making Games, 2015) example is a very sophisticated one that had a lot of work in it and ultimately wasn’t used as a reference when I was creating the game.

The example was a bit too complicated for myself and I believed that I would have spent a lot of time simply learning how this works. It was however a good read and allowed me to learn about this approach to procedural generation.

(BlackThornprod, 2018) version of procedural generation that I came across was interesting and was created with unity and involved generating a maze with a beginning and a end to the maze. He created rooms by placing a room and doing a number of checks.

The first he did was to check whether there was a door in the 4 directions. Then placed a position marker in that direction to determine where the position of the new room was. Then choose a door to place and made sure there was a door on the new room that was in the opposite direction of the door that was placing a room in the direction. then if the new room had a door in the right direction it was placed.

In essence it would create the rooms till it reached the end and made sure all doors on each of the rooms had a room it led to and so there were no doors into the void.

(BlackThornprod, 2018) stores the rooms in an array in a script and accesses the rooms form the array depending on the room that’s needed e.g., if the current room has a door that down then he needs a room from the top door array to get a room that has a door at the top. There can be a number of rooms in this array such as a room with doors in all directions, or left, or right, or down etc. They however have one thing in common and that’s there is a door that is at the top of the room. He does a check like this for all the doors that are placed as the script is run.

To prevent a room spawning on another room he adds a box collider to prevent it spawning a new room. The collider is a trigger and when there is a collision then the new room is destroyed if there was a room placed and then there will be no rooms that will spawn on top of already existing rooms.

(BlackThornprod, 2018) then fixes issues with doors having openings into the void and does a check to make sure that the spawner that has a bool that says nothing was spawned and the other spawner that was destroyed also has a spawner that’s false then he creates a bit of wall to close the room. These he calls closed rooms. He then has a problem with there being closed rooms spawning on the start room then he fixes this by destroying these using a script to destroy them

Another thing is that he can have certain rooms spawn more often than others and he does this by adding more of the type he wants more of in the array that it belongs to.

(BlackThornprod, 2018) then creates an exit and does this by creating a list. He adds the rooms to the list as they are created and this means that the last room in the list is the exit room and creates a boss in this last room that he has in the game. This makes it easy to know where the last room is and makes it easier to spawn a boss.

In the end I thought this was an interesting way to do the procedural generation in my own game but due to the lack of corridors I did not use it. I also had planned to use a tile map class early and this was not using a tile map. I preferred a system that would generate the values for the walls, floors and corridors using numbers that can be stored in 2D array and used to create a tile map. I thought it was interesting but chose not to do this way.

**Major Technical Achievements**

Overall, my major achievement was that it created and learned about procedural generation of dungeons and have increased my knowledge.

However, my greatest achievement was the generation of rooms in the dungeon that were decorated fully and made the dungeon felt alive. The creation of rooms that had importance or would have importance to the game was also something I was happy with. These included boss rooms that were completely changed, the worship as well was a different room and were given their own look and feel.

There were also rooms that had their own decorations and these also ma ethe dungeon feel as it were lived in and created a system that was not only able to create random decoration positions but was also able to create specific layouts such as the library that would have seats at the bottom left corner, or the feast room that had table and chairs around it and plants in the room as well.

This took time and was made easier by storing and then deleting rooms from the vector once they were decorated and prevented there being more than one type of room unless the coder wanted there to be more as I have for some of the special rooms in my dungeon decoration generator.

Overall, I was happy with the decorations and how they were laid out in the room.

**Project Review**

What went right? What went wrong? What (if anything) is still outstanding/missing (i.e., still left to do)? If starting again, how would you approach this project differently? What advice would you have for someone attempting a similar project in the future? Were your technology choices the right or wrong ones? If you chose the wrong technology, provide justifications for why you think this. What were the implications of your technology choices?

The project overall went well but there are some changes I would have made had would I do it again.

The first major one is to separate the creation of the decoration generation and the generation of the rooms. This would make it tidier. There is a major issue with the size of the dungeon generation class that is it’s too long and up to 1500 lines of code long and this makes it annoying to deal with.

What I would do is to create separate classes for the different types of rooms e.g., there would be a class for the generation of the special rooms, the unique rooms and all other rooms. The reason I didn’t do this is because I though the functions for the decorating of the rooms would be short but it ended up being long and so the creation of separate classes would make it better.

I am however very happy with the decorations in the rooms as they came out as I wanted them to. The unique rooms are different from one another and allow the user t feel as if they’re in a

lived-in dungeon with beds, feast, jails and coffin rooms along with worship and boss special rooms.

The advice is would give is to separate the classes out as it would make it easier to work with and would make it so that the code can be read easily. Also actually have the dungeon generator create a tile map and use that when testing as it makes it a lot easier than reading 2D array of text and also a lot more appealing to see it be created as a game world.

For the project I used SFML C++ to create the dungeon generation system. Overall, there were little to no major issues and the ability to use vectors over arrays was also a major help as they know their size and made life easier. This was handy as I used them a lot and storing objects in them was good and easy to do and work with.

I also had a debug mode in it that allowed me to turn things like the collisions off and to also output testing values and draw the colliders to show where they were in the world and it made life easier without causing major slow down. I also allowed me to test with ease and not comment a lot of things out when I didn’t want them to output but also didn’t want to delete. Instead, I only changed one value to turn them off.

The use of SFML for me was harder and I learned a lot form it. I think it would be easier to create this in Unity and even if it was, I am still happy I did it in SFML as I learned more about it and was also increasing my knowledge of C++;

# Conclusions

Overall, the work done was to create a dungeon procedural generation system that focused on creating a dungeon room generator that created the rooms within a grid and created as many as possible of different sizes. The dungeon generator was also to create a dungeon that was laid out well with no issues.

The goal was to have it so the rooms were connected and that there were no rooms that had doors that lead to the void outside the dungeons.

There was also a focus on preventing the spawn systems were all in different rooms and that all special rooms didn’t have these in them so as to prevent there being a way for the player to exit during a boss fight.

There also had to be a good system that prevented rooms spawning on their own with no way to access them and this was done and I was happy with it.

These were done right and there were little to know issues with the creation of the dungeon and the layout was good and I was happy with it.

The decoration system was also well done and I was happy with the ways it came out as well. The decoration was made to spawn on tiles that were floors.

There were also checks done to make sure that the tiles around the decorations weren’t meant to be accessible and that it was easy to access in the triggers and doors and that these weren’t blocked. There was also a focus to have décor spawn on tiles and not on the walls, doors or other decorations.

Overall, the creation of the dungeon went well and as long as the above issues were done right it would work out well which it did. In the end I came out with a procedural generation system that created dungeon rooms and decorated them which was the goal.

**Future Work**

Indicate what might be some next steps to try (if a student next year was going to undertake a project in this area what might be an interesting thing for him/her to examine?).

# References

BlackThornprod. (2018, March 13). Retrieved from YouTube: https://www.youtube.com/watch?v=qAf9axsyijY

Gamasutra The Art and Business of Making Games. (2015, March 9). *Gamasutra The Art and Business of Making Games.* Retrieved December 9, 2020, from Procedural Dungeon Generation Algorithm: https://www.gamasutra.com/blogs/AAdonaac/20150903/252889/Procedural\_Dungeon\_Generation\_Algorithm.php

MindControlDx. (2018, 7 31). *C++ Example of Dungeon-Building Algorithm.* Retrieved December 5, 2020, from roguebasin.com: http://www.roguebasin.com/index.php?title=C%2B%2B\_Example\_of\_Dungeon-Building\_Algorithm#Version\_3

# Appendices

Replace this text with Appendices.

This might include ethics application and other relevant material e.g., copy of any questionnaires used.