UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

PLAYING GAMES STUDIOS

WSOA4088A - Games & AI

Game Design Document

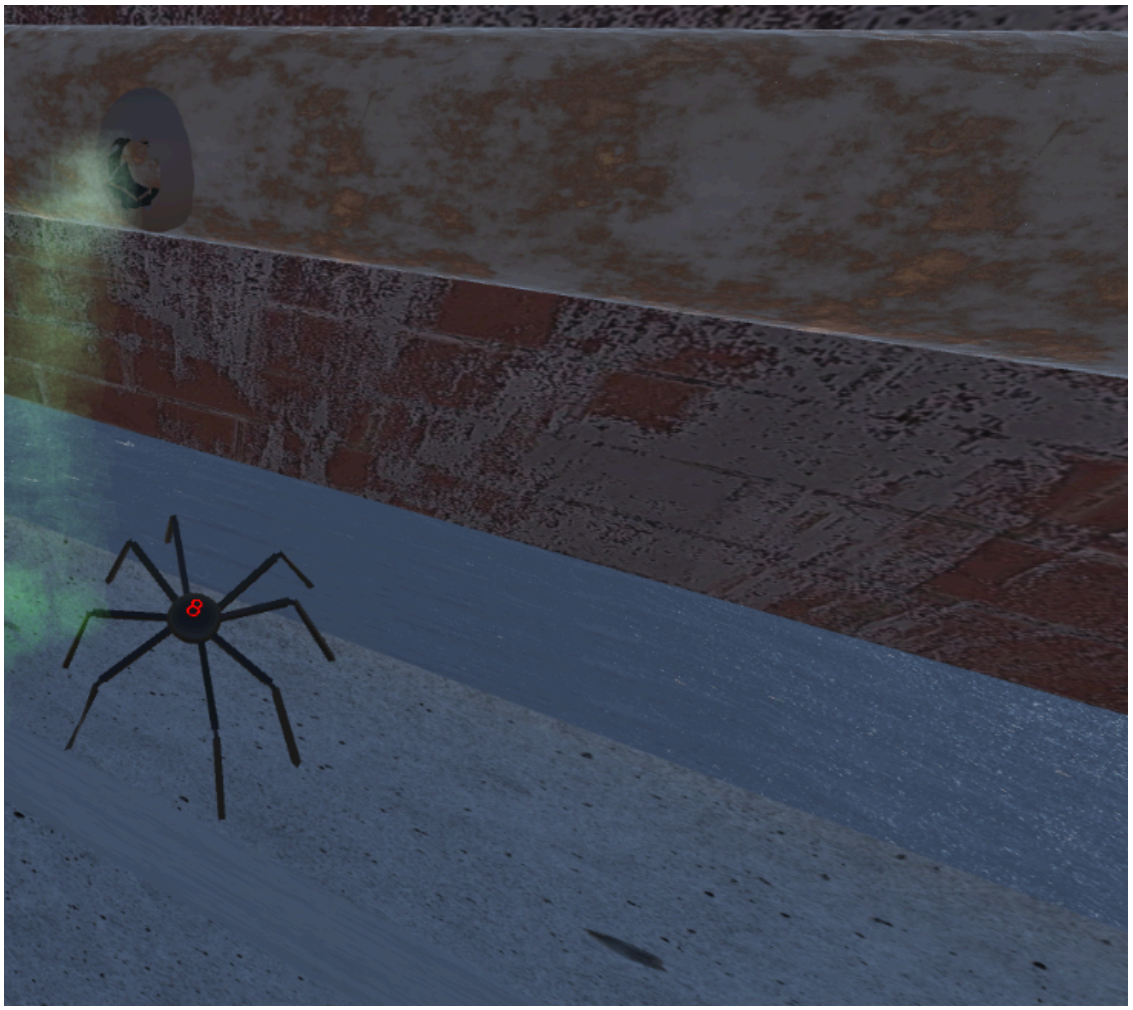
Mikhail Govind	Jean-Francois Retief	Erin Harper	Dylan Cairns	Malakai Braam
2425225	**2458318**	**2445245**	**2344382**	**2457821**

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

PLAYING GAMES STUDIOS

ITSY BITSY SPIDER

The Procedural Dungeon Crawler



Introduction

This document serves to cover the various aspects of game development relating to WSOA4088A - Games & AI Project. This document will cover each team member's various goals, the sub-teams, genre-analysis, inspirations, an overview of the game as well as the team's design goals, details on each AI feature implemented in the game and personal reflections of each team member.

The Team

Mikhail Govind **2425225**	❖ Procedural Content Generation Team ❖ Level Design ❖ Shooting & Other Mechanics Team	SUBDIVISION PCG Procedural Content Generation
Erin Harper **2445245**	❖ Procedural Content Generation Team ❖ UI Lead - Mini Map to help players ❖ Shooting & Other Mechanics Team	SUBDIVISION PCG Procedural Content Generation
Jean-Francois Retief **2458318**	❖ Project Management ❖ Procedural Dungeon Generation Team ❖ Procedural Reactive Animation Team ❖ Shooting & Other Mechanics Team	SUBDIVISION PRA Procedural Reactive Animation
Dylan Cairns **2344382**	❖ Procedura Reactive Animation Team ❖ Asset Creation Team ❖ Enemy AI and Animation	SUBDIVISION PRA Procedural Reactive Animation
Malakai Braam **2457821**	❖ Procedural Reactive Animation Team ❖ Asset Creation Team ❖ Enemy AI and Animation	SUBDIVISION PRA Procedural Reactive Animation

*Note the colour coded logos for the different teams - these will also appear on pages written by those specific teams about their work

1. Genre

This game is a dungeon crawler, where the player needs to navigate a procedurally generated maze/dungeon/sewer. It could also be described as a corridor-shooter.

2. Inspiration

- **Final Fantasy 16** - Abilities used for combat but for puzzling too.
- **Insomniac's Marvel's Spider-Man Series** - Spider's web shooting abilities and "blocking burst pipes with webs".
- **Where's My Water** - Sewer inspiration.
- **SilverlyBee's** Dungeon Generator Tutorial[1] and Extras[2] videos.
- **Codeer's** Procedural Animation Tutorial video[3].

3. Game Overview

Concept: You play as a spider that washed into a sewer. This allowed us to make the game a literal dungeon crawler.
Setting: Sewer complex (generated as a maze the player has to traverse through)
Objectives: Navigate through the maze, while defeating enemies and reach the main boss crocodile. Defeat the main boss and plug the burst sewer pipes. Both of these objectives can be achieved with the same interaction - aiming and shooting webs
Enemies: Crocodile (final boss), rats (mini boss), roaches, nats (flying enemies), parktown prawn special enemy (immortal, so you just avoid this enemy)
Abilities:
Hold Right Mouse Button to **aim**
Press Left Mouse Button to **shoot** webs (either at enemies or bursts in the pipe system).

4. Design Goals

Our main goals were to create a *procedurally generated* dungeon, as well as have *reactive animation* that changes due to the environment, via inverse kinematics and other algorithms. These will be discussed in detail in the section below.

5. AI Features

At the start of development, the group has split up into 2 teams, namely the Procedural Content Generation Subdivision and the Procedural Reactive Animation Subdivision. This allowed the different teams to focus on different aspects and simultaneously work on both AI features.

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

PLAYING GAMES STUDIOS

PLAYING GAMES STUDIOS
SUBDIVISION PCG
Procedural Content Generation

5.1. Procedural Content Generation

Mikhail Govind

2425225

Jean-Francois Retief

2458318

Erin Harper

2445245

Our Plan / Hypothesis / Design Goals

Procedural Maze Generation: i.e. procedurally interconnected corridors

- Hand-crafted “rooms”, connected via an algorithm
- Original video we found enables and disables doors, to make connected rooms, but our version enables and disables **walls** instead - to create a maze-like structure (connected corridors or sewer-pipes made out of “room-sized” tiles that connect seamlessly)

Original idea for Algorithm for Room-by-Room Procedural Generation (Not used in final product)

- Check the location of both prefab's doors
- IF (compatible) THEN place
- ELSE grab a random other prefab
- Repeat until end condition
- IF (limit reached) THEN spawn end-boss fight

"Compatible" - Each room is on a tile, with a door at a certain point. If these points coincide between two rooms, it can be placed.
"limit reached" - random number that will limit the amount of rooms in the game - to avoid near-infinite generation.

1 7	2	3	4	5	6 8
9					10
11					12
13					14
15					16
17 19	20	21	**22**	23	18 24

Figure 1: Visual depiction of a room, with original idea of algorithm

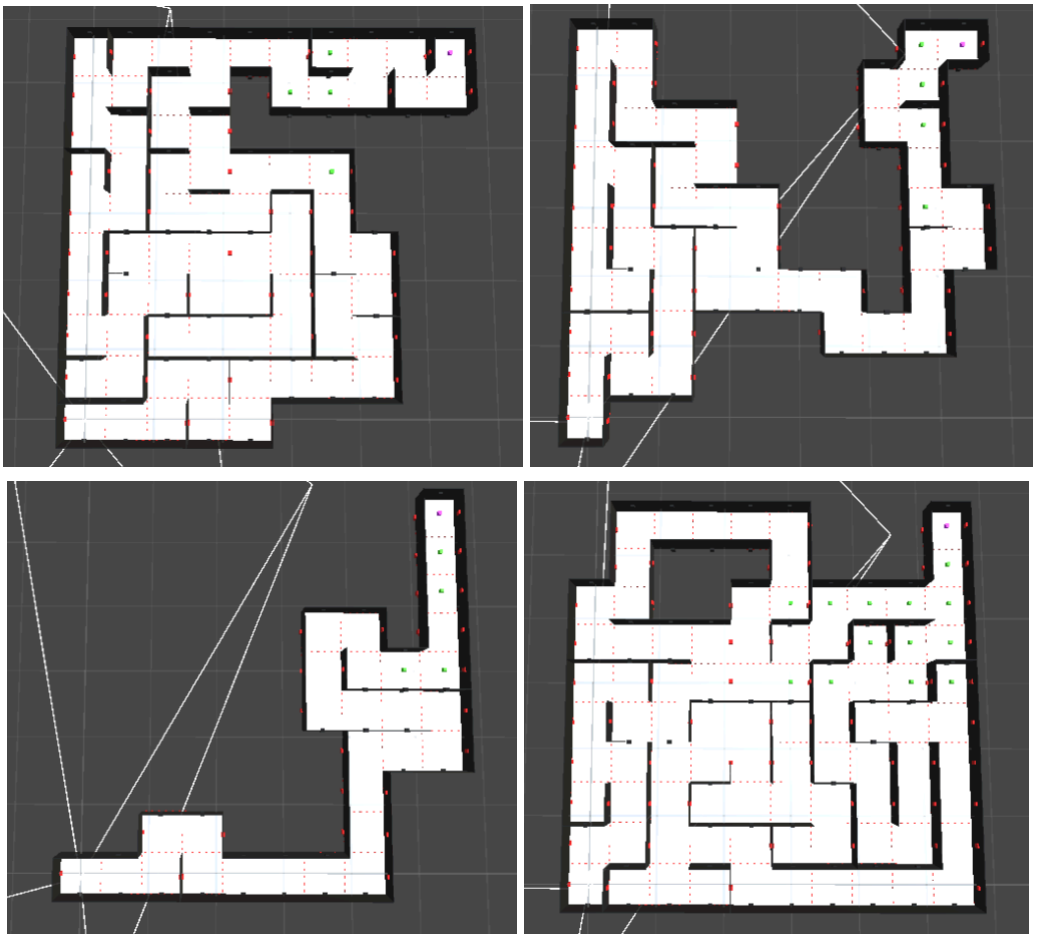
Example:
Room_1 = 9 and Room_2 = 10, then compatible
Room_1 = 22 and Room_2 = 4, then compatible

Figure 1: Screenshot of original game design document, made while we were planning our the game

The above algorithm would've taken too long to implement, so we looked for more practical potential solutions. One of these was an algorithm for creating a maze-like structure, in which we can add our dungeon elements after the base shape of a level is generated.

Algorithm we ended up using:

We found and used an algorithm that uses the *breadth-first search* methodology to check neighbours of a room and procedurally generate a maze-like structure. This algorithm is used to form the basic shape of each level/dungeon, and other methods are used to fill the dungeon with various enemies, puzzles, and clutter - with increasing difficulty (see green and pink indicator squares at later stages of the maze in the figure below).



Figures 2, 3, 4, and 5: Screenshots of early prototype of maze generator

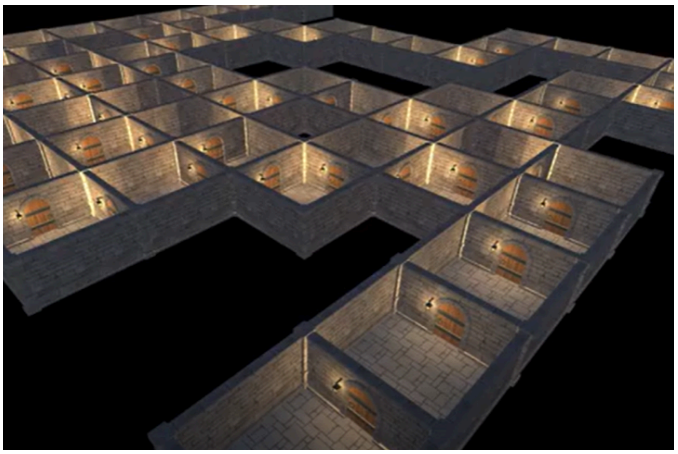
The main difference between our generator, and the one from the video, we took inspiration from, is that instead of keeping each “room” walled off and procedurally activating / deactivating doors, we adapted the code to procedurally activate/deactivate walls. This creates a procedurally generated maze of corridors (for our sewer system), instead of a grid of rooms that have procedurally generated door-connections between rooms.

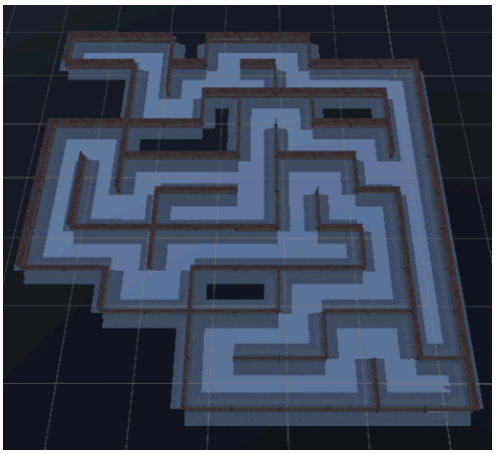
```
public class RoomGenerationScript : MonoBehaviour
{
    public GameObject[] walls;

    1 reference
    public void UpdateRoom(bool[] status)
    {
        for(int i = 0; i < status.Length; i++)
        {
            walls[i].SetActive(!status[i]);
        }
    }
}
```

Figure 6: Room Behaviour Script

Simply changing the room's behaviour, so that the array of walls is being changed, instead of an array of doors, creates a vastly different experience. From a claustrophobic grid of rooms to a sprawling maze. Both easy to get lost in.





Figures 7 & 8: Image of SilverlyBee’s Dungeon Generator Tutorial Video[1] and an Image of Itsy Bitsy Spider’s Maze Generator during runtime, respectively

Room Prefabs

In the maze, each room is a prefab, with other walls that spawn in / don't spawn in scoring to the generation algorithm. To create the look of a sewer, each wall has a step and a ramp leading up to it. This quarter pipe ramp allows the spider to clamber over it. Each prefab is then filled with enemies and/or extra pipes in the centre - to provide variety from corridor to corridor. This shows off the reactive procedural animation, to be discussed in the next section.

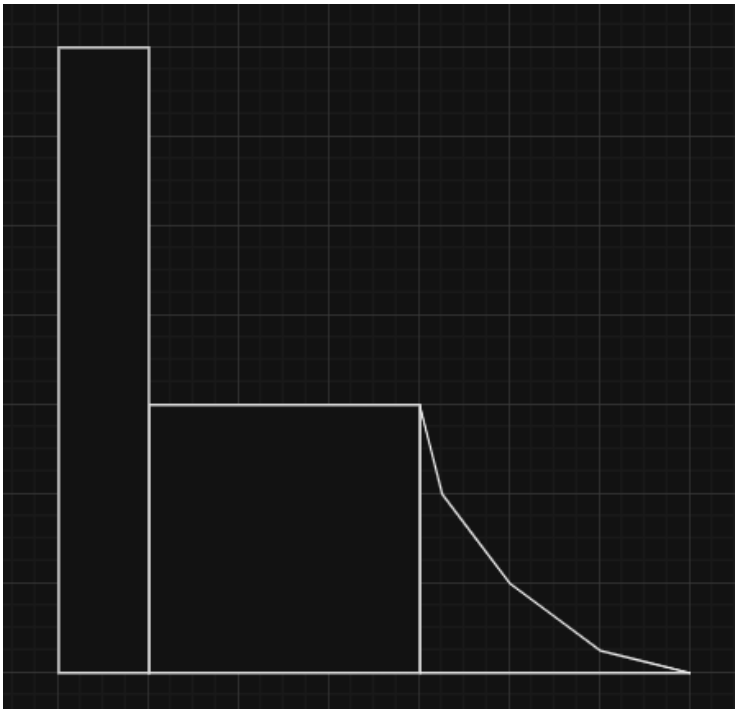


Figure 9: Blueprint of sewer wall within each prefab

Note: Textures used for the walls, ground, water and pipes are sourced from the website TextureCan[5] and Unity Asset Store[7]. We were able to find brick, concrete and metal textures that work for our sewer system.

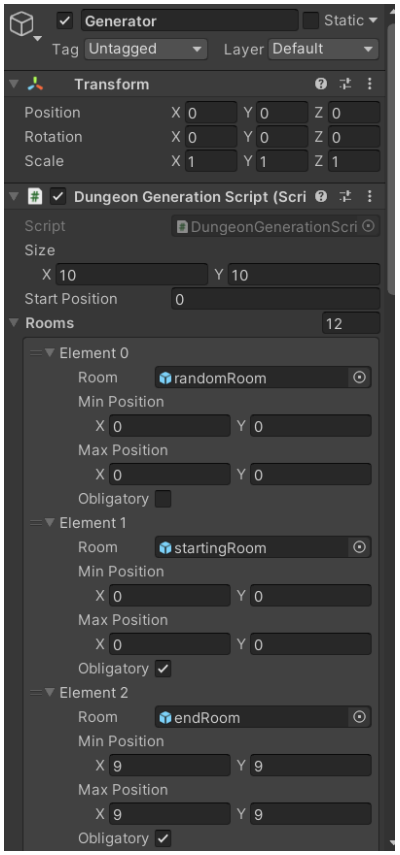


Figure 10: Inspector View of the Generator Object

Different room types (prefabs) are used as input to the system. Namely rooms containing other enemies and/or archways - which varies the play experience from room to room.

Some rooms, like the starting room (which must be empty) and the final room (which must have the final boss) are set as “obligatory” to ensure that it spawns in each instance of the maze.

Aspects, like the breadth-first search to create the maze, as well as the rulesets that enable us to create obligatory rooms, are explained well enough in the videos (SilverlyBee’s tutorial video[1] and extras video[2]) that inspired us to create this system. Consequently we won’t regurgitate the explanations in this document. Our understanding of the code does not go much further than the video’s explanations and our abovementioned change to fit the procedural generation to the purposes we had for our game, i.e. to make it a maze generator.

5.2. Procedural Reactive Animation

PLAYING GAMES STUDIOS
SUBDIVISION PRA
Procedural Reactive Animation

Dylan Cairns
2344382
Jean-Francois Retief
2458318
Malakai Braam
2457821

We took inspiration from a video, from the Youtuber Codeer[3], showing a quadruped character with procedural animation, specifically a walking animation, using inverse kinematics and an algorithm, that makes the limbs dynamically interact with a change in elevation on the ground (avoiding “floaty” walking on slopes). We sourced the code from an open-source github repository from metapika[6].

Our methodology for including Procedural Reactive Animation into the game:

- Importing Unity Inverse Kinematics Packages[4]
- Using inverse kinematics, raycasts and distance algorithms
- Keeping the “torso” a set distance from the “ground” layer (raycast)
- Steps from above video on procedural animation

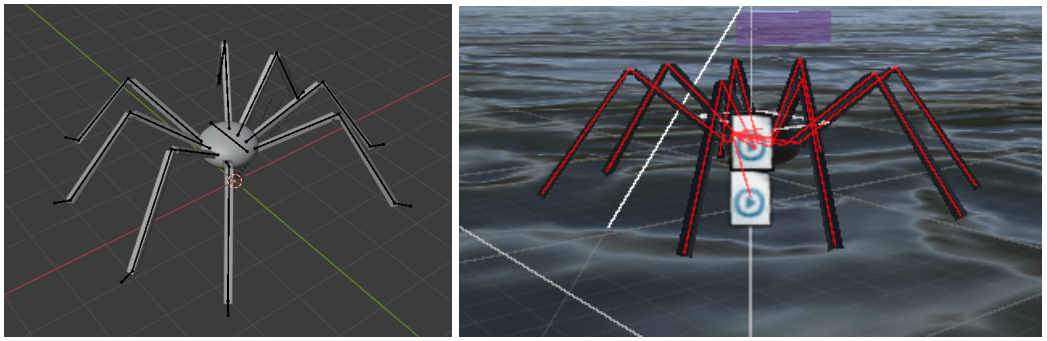


Figure 11 and 12: Armature structure in Blender and Unity respectively

The spider can clamber over geometry on the “ground” layer, if the player speed and/or the slope/step of the geometry is applicable. For practical use inside the game, the spider will dynamically clamber over the sloped surfaces on the walls of the sewer.

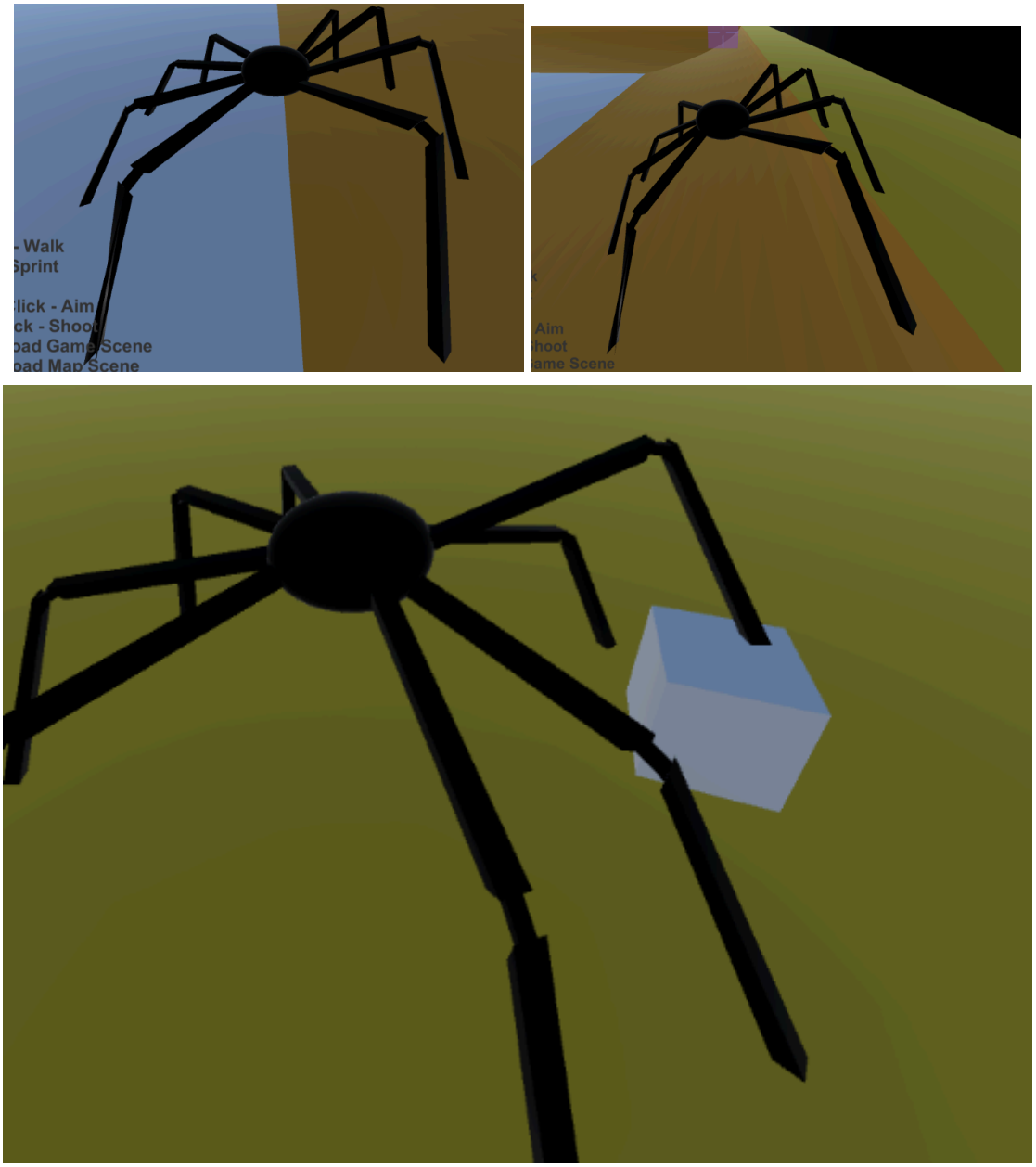


Figure 13, 14 & 15: Spider legs interacting with geometry

The spider model used (Rotation Issue)

When rotating the spider body, the legs wouldn't rotate realistically with it, so the spider's body was changed to be symmetrical in all four directions and the spider no longer rotates at all. Instead the camera rotates, along with the "Gun" the spider has.

Figure 16: The spider with circular body

Enemies - Forward Kinematics

The enemies that appear in the game are:

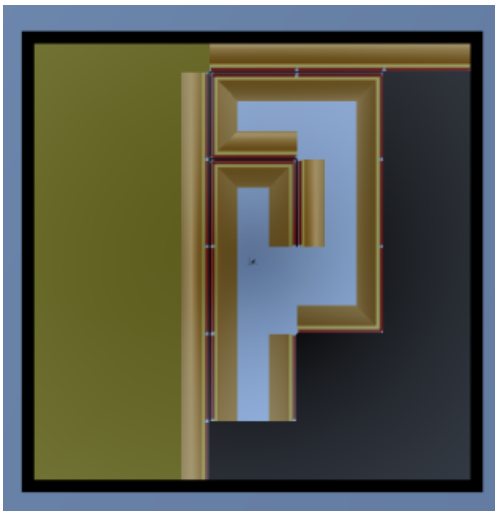
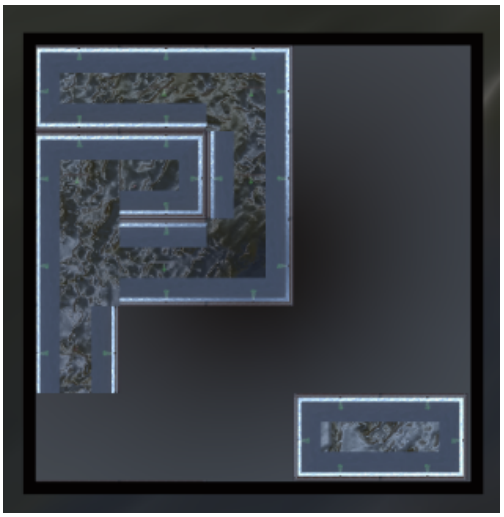
- Gnat
- Roach
- Parktown prawn
- Rat
- Crocodile - Final Boss

All of the enemies use traditional forward kinematic keyframe animations created in Blender for idle, walk and attack behaviours respectively. This was done to show off the difference between the walk animations during runtime, as well as to contrast the spider's procedural movements to show off the natural leg and joint movements that this affords.

6. Other Features

6.1. UI

The minimap shows off the area of the maze you have explored, and the greyed out areas are unexplored tiles.

Figures 17 and 18: Mini-Map (prototype and Final Version, respectively)

6.2. Shooting Webs

From just above the spider, the player can shoot out webs to attack enemies or to block drains. The purple reticle, is an actual 3D object in the scene - that for coding and testing purposes is used a a “gun” asset (visually keeps track of the projectile spawn point)

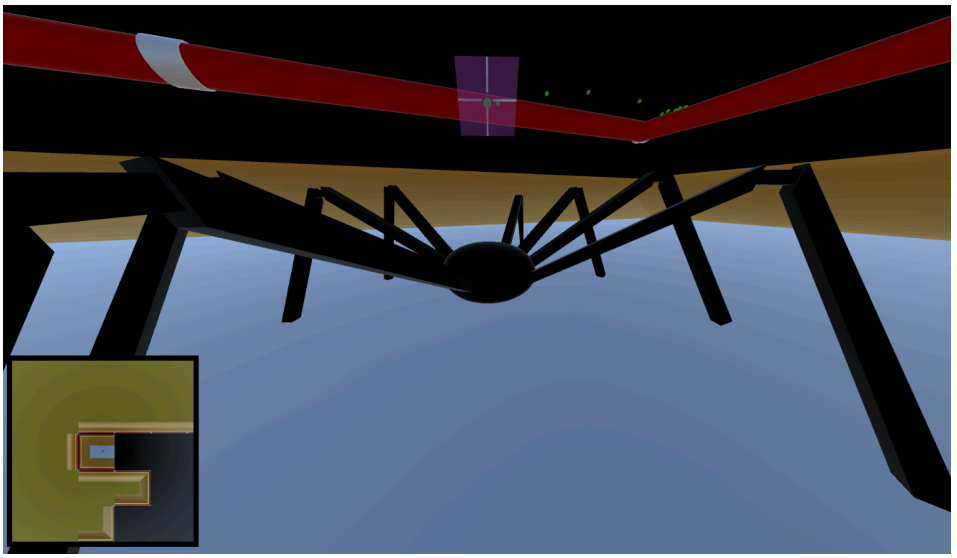


Figure 19: Shooting and reticle (prototype version, since it reads better in still image)

6.3. Enemies

Enemies are discussed above in the animation section, but here is some stats that could prove useful to players (also included in the main menu):

- Gnat Health = 1
- Gnat Damage = 1

- Roach Health = 2
- Roach Damage = 5

- Parktown prawn Health = 9999 (just run)
- Parktown prawn Damage = 10

- Rat Health = 10
- Rat Damage = 10

- Croc Boss Health = 20
- Croc Boss Damage = 20

Note:

- Player Health = 100
- Player Damage = 1

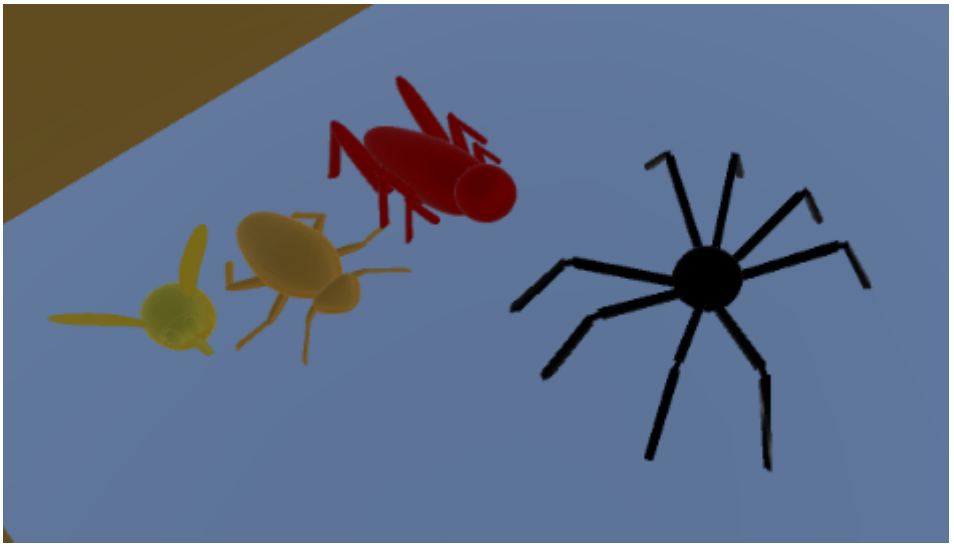
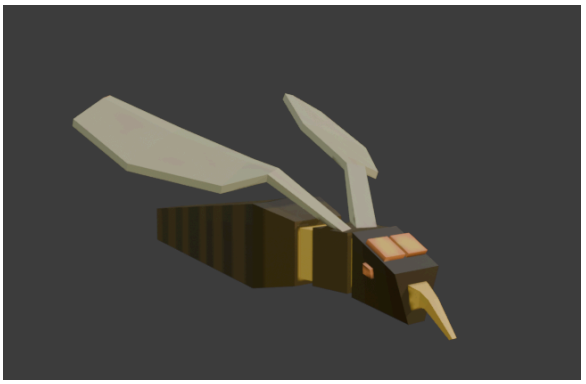


Figure 20: Prototype Enemies and Player Character (spider)

As mentioned above, all the enemies use forward kinematics. It is easier to use than inverse kinematics, but far more time consuming.

Figures 21 and 22: Final Textured Crocodile and Prawn

Figures 23 and 24: Final Textured Rat and Roach

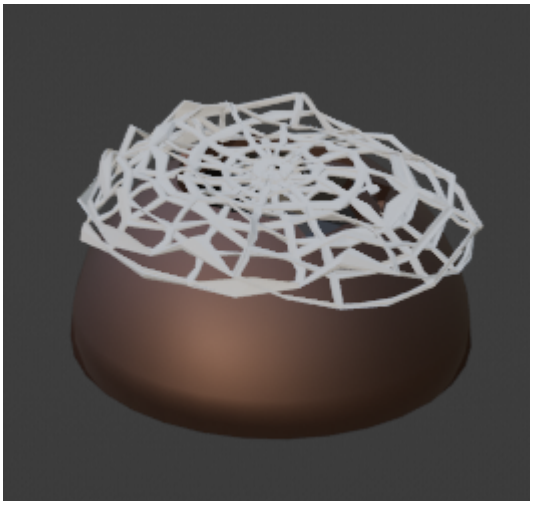
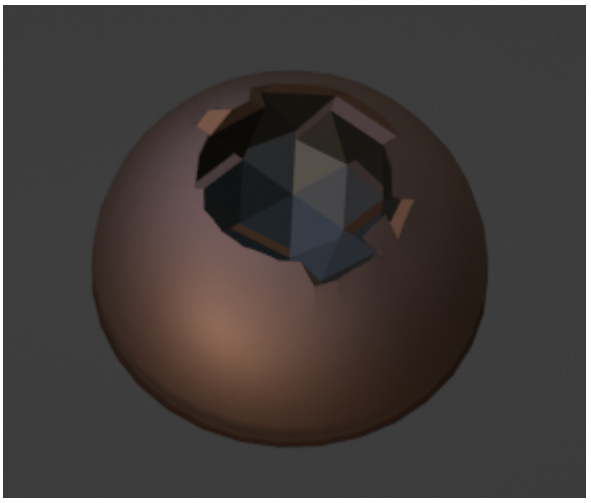


Figures 25: Final Textured Gnat

Most of the enemies use basic NavMesh components to facilitate movement, while the crocodile moves between two set points using the lerp function. The hitboxes (rectangular) for the enemies mostly are situated around visuals and scale of the enemy, but playtesting revealed that the short roach was too difficult to hit, and thus its hitbox was made to be significantly taller than the visual representation of the enemy.

6.4. Blocking Pipes

The player is able to block burst pipes by shooting web projectiles at it. A trigger swaps out the gameobject's visuals when a projectile tagged as a "web" collides with the hitbox.

Figures 26 and 27: Blender View of Closed and Open Pipe Assets respectively

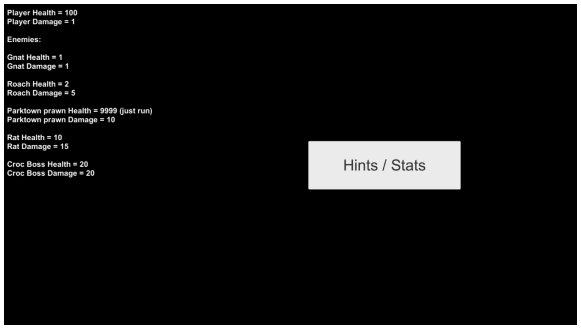
6.5. Demo Scenes - For 30 August Demo

For the early demo (30 Aug) this menu and two demo scenes were created to show off the reactive spider legs, shooting mechanic and map generation.

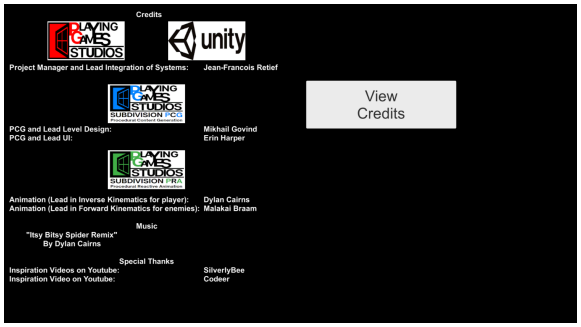
Figure 28: Demo Menu

6.6. Final Menu

The final menu includes options for 3 different size levels (mazes are different sizes). All the levels use the same generator, but the seed size, as well as the room's relative available spawn locations are set to different values in the generator's inspector view.



The player and enemy stats are also viewable at a button press. This helped playtesters to grasp how many times they'd have to hit certain enemies. This was especially helpful for the final boss crocodile in a level.



Credits for the teams are also included in the main menu. This was mainly added due to habit (we always include credits in our games)..

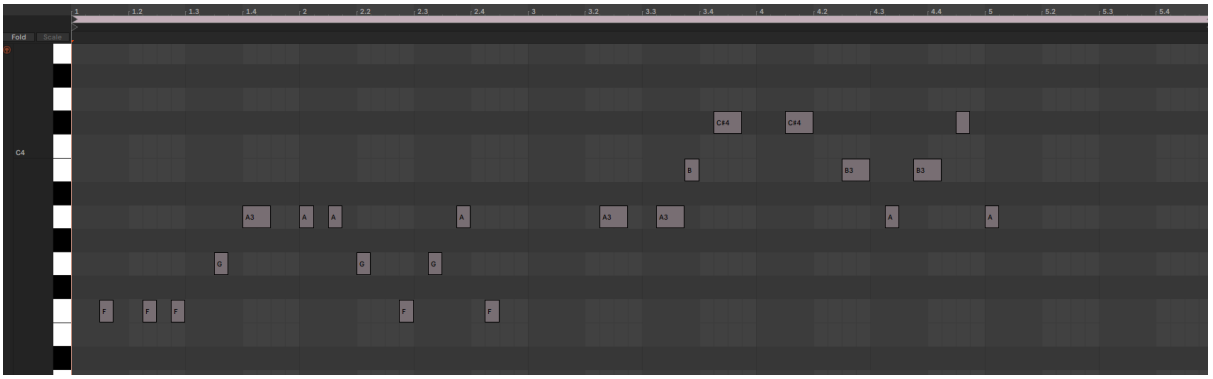
Figures 29, 30 and 31: Main Menu Screenshots

6.7. Playtest feedback (at GameDev Meetup)

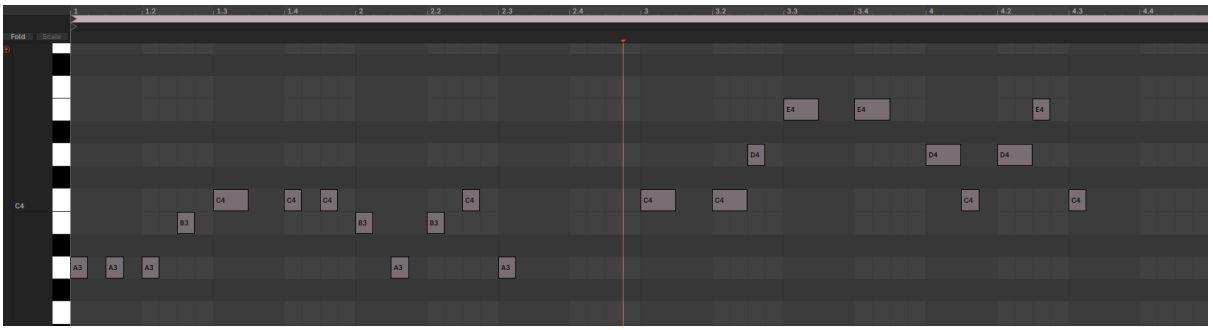
On the 13th of September, we presented a prototyped version of Itsy Bitsy Spider at the GameDev Meetup. We had a total of 7 people play the game even though we had little to no logic for the enemies, textures and materials as well as sound implemented yet. However, the main AI systems, PCG and PRA, were working in the game. Players' overall feedback was that the concept was really good and the PCG was really cool. The overall mechanical feedback and movement was adequate for the players as well. Due to this meetup we have aimed on making the game more aesthetically pleasing along with providing the player with sufficient visual and audio feedback.

6.8. Sound Design and Music

The main soundtrack is a rework of the ‘Itsy bitsy spider’ song. I found the midi clip for it online and then modified it into a minor scale melody.



Figures 32: Original midi clip



Figures 33: Modified midi clip

I made the soundtrack while keeping in mind the atmosphere of the game. Being sewer based and dealing with murk and water, I focused on using sounds that convey that ominous, wet, dank atmosphere that sewers are known for. I also made the soundtrack loopable, so the start and end blend seamlessly allowing it to be played continuously without interruption, almost sort of adding to the dread that the sewer is ‘never-ending’ and finding your way out is imperative, but also spooky.

7. Reflections

7.1. Mikhail Govind's Reflection

This is the second time I am working with this team on a project and it has been amazing. Compared to the first project where we were still getting an initial understanding of each other's strong points, how to split the roles, and each other's workflows; here we have more experience with each other so the workflow went smoother and more efficiently.
Obviously, every single time we collaborate, we'll get better with working together as a team but since this was the second project, that allowed us to focus heavily on applying each of our own specific skill sets to the best of our abilities.

The main takeaway I had from our team's previous project was the importance of intention and having an overall goal of what we are trying to achieve and every choice we make is to achieve that goal. This project's case is a focus on integrating AI techniques. In the beginning phases of this project, we planned to have the water pipe burst puzzle system to have much more mechanics and aspects implemented in it but once we fully planned out the AI techniques that we wanted to develop and integrate along with exactly how, we then decided to simplify the water burst puzzle system so we could place as much attention on the AI techniques as that is what this project and course is all about.

I am part of the Procedural Content Generation subdivision of this group. This AI technique was interesting to learn and implement as it was mainly algorithm focused which before this project has not been something I have delved much into. A lot of maths was involved in this code which is not my strongest coming from the BA side of Digital Arts, although tough, it was a worthwhile experience finally diving into this avenue of development, fiddling with the code until it worked the way we wanted for our game and understanding how we got it to function, and just how it functions in such a manner. Along with this, my strengths lie in forming systems and designing levels, which is exactly the role I took up, like planning and developing the water burst puzzle system and how that would work with the shooting mechanic. The other interesting thing is developing and planning the mechanics and systems of this game around the procedural content generation. The puzzle system, the shooting mechanic, the geometry of the rooms, the way enemies would spawn and function and etc. were all planned with this being a procedural dungeon crawler in mind and the AI technique used to create such an experience. This has been the closest experience of what I could imagine it to be working in, perhaps, a small indie studio and has been vital to improving my teamwork skills along with also revealing new knowledge to me and even knowledge I have gained from my group like learning about the Procedural Reaction Animation done by the other half the team which I didn't even know existed until those teammates proposed such a technique. Always something new to learn from other people. It was then constructive, implementing the different AI techniques together into one project and in the way of creating the cohesive experience between the techniques which I believe we have achieved. The PRA technique mimics and gives the feel of a spider's movement and the PCG technique creates the dungeon crawler experience we aimed for and in context of the Itsy Bitsy Spider inspiration we drew from.

7.2. Malakai Braam's Reflection

My role for the project consisted of the creation of the enemies, projectiles and broken/fixed pipes and assisting in the research and implementation of the procedural animation. Initially I had modelled a spider that resembled a semi-realistic spider with single object legs. Through discussing our found research between articles and youtube videos, we realised that we needed a player spider with separate leg objects to parent bones to. Dylan created what we called the "Mentos" spider and we played around with the amount of separate bones the spider needed in order for the animation package within unity to function as intended. Eventually Dylan found the sweet spot, by having two bones within the spider and the animation worked successfully. When this worked, he and I sat back out of straight disbelief. A game that we are creating with our team now has animation that does not consist of manual keyframed positions but an algorithm that moves the bones for us.

Our personal work process as the PRA (Procedural Reactive Animation) department consisted of us joining a discord call, showing our assets that we have created whether it be enemy related or player, tweaking and editing the textures, animations and meshes together and implementing them into the game. This work ethic really allowed us not only to grow our understanding and skills of animation and asset creation within Blender and Unity, but to also see what we have created or animated from different perspectives. I noticed this more when displaying the forward kinematic animations that I created for the crocodile as well as the overall asset. I am not usually one that enjoys the low poly aesthetic in games and so when the team agreed on having a PS1 style for the assets I doubted my abilities highly. I presented the crocodile expecting the team to ask me to subdivide it or smooth it out, however they seemed very happy with it along with a very simple animation

This project has taught me through learning and researching Procedural Content Generation and Procedural Reactive Animation, that the implementation of AI within a game is extremely doable. I believe that the skills that I have learnt through this project will last me a lifetime.

7.3. Jean-Francois Retief's Reflection

Our team wanted to seek out two different challenges, namely the procedural maze generator and the procedural animation. The respective team members wanted to tackle these challenges, because they were new to us. We have yet to work on games with any procedural generation, as well as animation using inverse kinematics within Unity. These challenges sounded like fun nuts to crack for the first time, and we wanted to feature a game that features both. As the project manager, I had a hand in both sub-teams. I joined meetings of both teams, and worked to integrate the generator, animation and other game mechanics into the game. Some examples include, helping the animation team import different animation and logic into the game, creating mechanics such as the aim-and-shooting mechanics, mixing the animated enemies into room-prefabs that I put into the PCG team's generator, etc. This project was great practice for group work, project management and integrating team members work to create a cohesive final product. This will be invaluable to my future as a game designer/developer.

In terms of the project's correlations to the research project 'Games and AI' and what I learnt from the course's perspective: it was very interesting to focus on generative mechanics. As an engineering student, I am used to mechanic-centric game design (i.e. building a game idea around certain mechanics, rather than a narrative or theme), however I have never chosen such complicated mechanics to build the game around. It was also interesting to have split the goal of the game in two parts from the very beginning (namely maze generation and procedural animation). For quite a large portion of the development time, the two parts of the game were not connected, so when we integrated the two parts together - it created a third main challenge for the group. Luckily, most of what we planned in terms of integrating the animation (with inverse kinematics) and the maze geometry worked out.

Overall, I am happy with the final product and what I learned from the project. Through the course of this project, my skills in the following areas have improved: coding with procedural/generative algorithms, implementation of logic in animation systems in games, pathfinding systems within Unity, general knowledge of animation (such as implementing inverse kinematics in a game engine) and PCG (such as the method of maze generation, using breadth-first search method I had learned about in my second year course, COMS2004A - Data Structures and Algorithms, from the BEngSc side of Digital Arts).

7.4. Dylan Cairns' Reflection

Out of everything required of me during the creation of this project, the thing that stressed me out the most was implementing procedural animations. Having no prior knowledge to the topic and having to implement it into a game meant that it was a very daunting task. At first, I created a spider character with 8 legs, each leg made up of 5 joints, similar to a real spider. I did not realise this, but Unity's animation rigging package contains components for legs with two bones, and legs with more than two bones respectively. I tried using the chain constraint component (more than two bones) for each leg and came up empty handed. The IK worked but I could not get it working through scripting. After this, I created a new spider with 8 legs, each made up of two bones, which ended up working and that is the spider present in the final game. This was such an enormous relief for me because something that was so daunting to me actually came out right. I actually became so comfortable with implementing procedural animations that I ended up using them for every movable thing in a game jam creation, which made me feel even more secure. Procedural animations really breathe life into a project. It feels like your creations can work together with code to become real livable things. Many people think that letting the computer do most of the work for you results in unnatural and biassed outcomes, because computers are never wrong. In this case though, I think that letting the computer do the animating for you results in realistic outcomes specifically for procedural animations which are highly configurable.

I was also in charge of the sound design and composition. There was nothing too complicated here, other than trying to rework the 'Itsy bitsy spider' theme into an ominous sewer soundtrack. A lot of the attack noises for each enemy came out robotic and a bit comedic, which made me laugh quite a lot.

Finally I was also tasked with creating some of the assets for the game. As I mentioned, I created the spider character, as well as some of the enemies too. This was a cool challenge, as I was able to put all of my topology, rigging, weight painting, and animation skills to the test, which turned out awesome. Implementing the enemies inside of Unity was a breeze too due to the way that the enemies were handled inside the project.

7.5. Erin Harper's Reflection

To reflect upon the creation of our game, the main goal for this project was to create a dungeon crawler, in this project I had the opportunity to play a role in the Procedural Dungeon Generation, UI and Puzzle/Shooting Team.

When delving into the Procedural Dungeon Generation part of the game, our focus was to create a dungeon crawler that the player would need to navigate through with a procedurally generated maze-like dungeon. This therefore creates a sense of unpredictability and exploration, as the player is required to navigate through the maze (depending on what kind of maze is generated) and complete the tasks provided to them on the screen such as defeating enemies, plug burst sewer pipes, etc. thereafter defeating the boss. With the procedural dungeon generation I would say one of the issues we face were the placement of pipes with water bursts, as we initially wanted the player to cover all the pipes that the generator would display however, with the rooms being procedurally generated some of the pipes were on the outside or in places where the player couldn't reach so we had to change it from showing the amount of pipes cover and the total required to just the amount so the player does not get confused if there are pipes missing that they did not manage to cover.

Moving on to the UI role, in this section, our focus here was to create a mini map that can display the pathway the player will take, while showcasing the current location of the player and any enemies along the way that they may encounter. Therefore offering just enough information to the player without giving away the complete map layout to the player. With the creation of the mini map, there were not as many issues or challenges faced in this task, as I created this in a previous project and I managed to learn from my past mistakes so I would not encounter these issues in this task.

To conclude, I am happy with the result of the project, and I think we worked together great as a team. This has been our second time collaborating with each other on such a big project and I feel that we have learnt what areas we all thrive in which made it easier to work on this project. From my side, I must say it was my first time working on a project that implements a form of AI into a game, and I was a bit nervous as to how our final game project would turn out, however I am ecstatic with the results and proud of all of us for creating this project. This project has taught me a lot and I feel I have enhanced my skill as a game designer.

References

[1] SilverlyBee (2021) “Procedural Dungeon Generator in Unity [TUTORIAL]”, Youtube, Available at: https://www.youtube.com/watch?v=gHU5RQWbmWE (last accessed: 27/07/2024)

[2] SilverlyBee (2021) “Procedural Dungeon Generator in Unity [EXTRAS]”, Youtube, Available at: https://www.youtube.com/watch?v=gI6G49t--RY (last accessed: 27/07/2024)

[3] Codeer (2020) “Unity procedural animation tutorial (10 steps)”, Youtube, Available at: https://www.youtube.com/watch?v=e6Gjhr1IP6w (last accessed: 30/07/2024)

[4] Unity Technologies (2024) “Inverse Kinematics”, Unity Manual, Available at: https://docs.unity3d.com/Manual/InverseKinematics.html (last accessed: 30/07/2024)

[5] TextureCan (2024) “630+ Free PBR Textures (CC0 Textures)”, Asset Reference, Available at: https://www.texturecan.com/ (last accessed: 20/09/2024)

[6] Metapika (2021), Github Repository: metapika/unity-procedural-animation, “Unity Spider/Bug Procedural Animation”, Available at: https://github.com/metapika/unity-procedural-animation (last accessed: 27/08/2024)

[7] A dog's life software (2016), “Foam Textures and Materials”, Unity Asset Store, Available at: https://assetstore.unity.com/packages/2d/textures-materials/water/foam-textures-72313 (last accessed: 20/09/2024)