

Progress report marking sheet

Student's name: Matthew Taylor	Student's Reg: 100151729	
Supervisor: Dr Rudy Lapeer	Date:	Signed:
2nd marker:	Date:	Signed:
Title: Third Year Project: Report		

Agreed Marks

Description of project: aims, motivation	/10
Description and understanding of issues and problems addressed in the project	/20
Design and planning	/50
Evaluation of progress	/10
Appropriate use of L ^A T _E X	/10
Overall mark	/100

Comments

Report and completed mark sheet should be returned to the LSO.

Third Year Project: Report

Matthew Taylor

Registration: 100151729

1. Introduction

1.1. Aim of the project

The project aimed to implement a first person video game which uses procedural generation to provide a novel and interesting game experience. The project will explore the technical aspects of using PCG as a core feature of a game, specifically, implementation and performance.

1.2. Motivation

Procedural generation is not a new field, especially in video games, but there are interesting areas that are seldom explored. One of these areas is real-time procedural generation.

The motivation behind the project's aims were to explore new and interesting concepts, namely, using PCG as a core feature in a game and using real-time procedural generation to provide an interesting experience.

1.3. Literature

The project also drew inspiration and guidance from existing PCG games, as described in papers by Spufford (2003), Welsh (2016) and Brown (2016).

The project also took into account the wealth of research done on PCG techniques, as described in papers by Perlin (1985), Ebert and Musgrave (2003) and Dormans (2010).

2. Issues and problems

This section will detail the issues identified at the project commencement and then detail how they have been addressed.

2.1. Project scope

Procedural content generation is a large field that has been explored in the literature since the 1980s. The paper by Hendrikx et al. (2013) details this, providing a survey of PCG in games and the many techniques that have been used. A potential issue of the project is attempting to explore and implement too many of these techniques and ending up with a diluted project of little technical interest.

To address this, the project scope was focussed on some areas identified as being under-explored in the survey - level generation in real time systems. This will allow the

project to explore several ideas in this domain and provide a comprehensive implementation in a fully featured game.

2.2. Designing 3D games

Designing 3D games and the assets required to populate the levels created could take a long time, especially if much of the functionality is created from scratch.

To address this and enable the project to be achieved in a realistic timeframe, the Unity game engine was chosen. This tool enables relatively rapid production 3D game environments and assets, as well as being well documented and supported.

2.3. Performance

A game's performance was a critical aspect of its success. If the game cannot be played in a sustainable framerate on common hardware, it could be considered a technical failure. Performance is a broad term, so the project will specifically focus on the following two aspects of performance. The first aspect is **level generation speed**. Each time the game loads, a new level is generated, so achieving a suitable speed in generating the levels and performing the associated work, like loading geometry, was a key consideration. The second aspect is **framerate**. The project will procedurally load a considerable amount of geometry in varied conditions. As much of the geometry will not be static, it will be required to load a large number of meshes into memory, which could affect the performance.

2.4. Evaluation of success

A key issue with this project is evaluating if it is successful. If there is no clear notion of what success looks like, the project could progress in unproductive directions. Below are the key measures of success chosen at the start of the project to measure success against.

2.4.1. Convincing levels

The levels must appear convincing. This is difficult to quantify, but nevertheless is an important attribute. The levels (at least in the initial design) will be mazes generated in a grid, with the grid sections subdivided in different ways.

This could easily create a boring level, where the grid pattern is discernable and the rooms are boring and procedural to traverse. Successful generation of levels will avoid these issues.

2.4.2. Convincing decoration

The decoration in each individual room will contribute to how convincing and hand-crafted each level looks. As described in Germer and Schwarz (2009) and Taylor and Parberry (2010), there are several challenges to making room decoration appear to be hand-made when it is procedurally generated. Of particular focus will be generating furniture and objects in positions that feel realistic and unique.

2.4.3. Performance

Performance is another key component of the project succeeding. If the game responds poorly, it will be frustrating to play. Performance will be measured in a few ways:

- **Framerate:** must be at least 30 frames per second
- **Initial loading time:** must take less than 30 seconds
- **Smooth level re-generation:** no dip in framerate below 30 frames per second

3. Design and planning

3.1. Implementation methodology

The project development operated under an Agile methodology. Agile does not require that all requirements (or “user stories”) are defined up front, instead it is encouraged that they are developed and refined during the lifecycle of the project.

The project itself was divided into Agile “sprints”. It was decided to deliver a sprint every two weeks, as this fit with the development time given to the project and providing updates to the project supervisor.

While all the stories were not required at the outset, as part of the planning, it was decided to plot rough timescales against Agile “epics”. These epics provide a high-level view of functionality and are detailed in Table 1 in Appendix D.

The epics also contain very high-level user stories, which provide a little more detail. These were decided at the project outset, then refined during development. Before any user story can be developed, it was required that more detailed requirements were written. These were then recorded and tracked on Trello.

3.2. Implementation plan

With the high-level design documented in the epics, planning of these epics against timescales was required. This timeline plan is visible in Figure 1 in Appendix A. Progress was then tracked against this Gantt chart to ensure that progress was being made and that the project could be achieved as planned.

As the project was run using Agile principles, the Gantt chart was revised to reflect changes to the plan. These changes are visible in Figure 2 in Appendix A.

Changes to the Gantt chart include the addition of a Level Generation task - the Level Re-Generator. This reflects a focus of the project on procedurally re-generating levels in real-time, so this has been split out as a separate task and given a shorter timescale, reflecting the importance of the task to the project.

The gameplay section was split into two tasks, reflecting the focus on implementing core gameplay as a priority, with enhanced gameplay being desirable but not as critical to the evaluation of success of the project.

3.3. UML

As the project began, a UML diagram was designed to provide a high level framework to work towards. Unity does not prescribe much structure, so a UML design to begin with was useful. The initial UML design can be seen in Figure 6 in Appendix B. The iterated design can be seen in Figure 7 in Appendix B, which shows the definition of classes being enhanced with the additional methods and attributes required to support the design, as well as the addition of the **Level Re-Generator** class.

3.4. Requirements and prioritisation

The project defined epics (or, high level requirements) at the outset, with more detailed requirements written at each stage. This allows the project to be flexible, with lessons learned in the prototyping incorporated into future requirements. It also saves time, because requirements do not need to be written in advance when they would likely be changed anyway.

The epics and requirements written for the project so far are available in the appendix in Table 2 on 18.

3.5. Design prototypes

3.5.1. Guaranteed path generation

The first part of the design to be implemented required a guaranteed path through a maze to be generated. This was achieved by using a random walk - a stochastic process implemented in two dimensions to provide a definite route through the maze.

The random walk is implemented using an agent based approach. The agent can choose from three random directions to travel in - left, right or down. These probabilities are detailed in Table 3. When the agent has chosen a valid direction, it randomly selects a valid tile.

The agent has methods of checking where it has been and where it is going next, so that it picks valid tiles that have openings in the directions it has come from and intends to go. The agent is given bounds and restrictions while it is performing the random walk, which provides a method of parameterising the level and deciding on an endpoint.

3.5.2. Decoration of rooms

Rooms are decorated with interior sub-dividing walls and other elements to provide a convincing and interesting environment to traverse. Decoration is achieved by populating each room with a random pre-defined set of geometry, examples of which can be seen in Figures 8, 9 and 10, in Appendix C.

In the prototype design, a single point is used to populate all geometry from the centre of the room. However, iterations on the design will use several points which an agent will scan through to populate things like sub-rooms, corridors, walls and desks with appropriate decorations, providing considerable variation for little asset generation cost.

4. Evaluation of progress

4.1. Progress against plan

As detailed in the revised Gantt chart in Figure 2 in Appendix A, progress is broadly on plan. The prototyping of the level and room generation is complete and suitable methods have been implemented. Much of the remaining work on level and room generation will focus on refining the performance, achieving higher levels of complexity and improving performance.

4.2. Changes to plan

The plan has changed slightly since the project began. As can be seen in the differences between Figure 1 and Figure 2 in Appendix A, two new tasks have been added.

The **Level Re-generator** task has been added and given a shorter timeframe to provide focus on this task. The re-generation aspect of the project is important to consider when evaluating performance. It also provides a novel feature not seen in many computer games, so this was considered important to focus on.

The **Add Enhanced Gameplay** task was also added, to provide a marked difference between the basic implementation and a more detailed gameplay experience. It will not require much effort to provide a basic experience, which will be important to implement early. The project will then consider enhanced gameplay elements (more enemies,

pickups, keys for locked doors) towards the end of the project, after the main aims have been achieved.

4.3. Evaluation against success measures

In Section 2.4 on page 3, three measures of success for the project were defined.

Against the **Convincing Levels** measure, the project has the potential to succeed. The guaranteed path/tile filling combination creates large levels, with a path through them. The project does not currently take steps to block off many of the non-guaranteed path tile transitions, so the tile structure is discernable - it is quite easy in parts to see how the tiles are laid out. Future

Against the **Convincing Decoration** measure, the project again has the potential to succeed. The prototype of the room decoration system works, but would require quite a lot of manual content generation. However, the project has also successfully implemented agent based approaches to creating levels, so a combination of hand-crafted content and procedural generation should enable rooms to be decorated in ways that are procedural and unique, but also in ways that make sense to humans.

Against the **Performance** measure is where the project has faced the most challenges. The level generation algorithm is quite efficient. As presented in Figure 4 in Appendix D, even when large sizes are used the generation and instantiation time does not increase by much. Indeed, the main constraint around generating large levels is not the time to generate them, but the real-time performance of the levels.

The real-time aspect suffered when large levels were generated. Average framerate was low (around 10 FPS) when levels and decoration was applied to suitable level sizes for a playable game. This resulted from a lack of occlusion culling and a large draw distance. This was overcome by reducing the viewing frustum size from its default size to one more suitable for navigating a closed maze-like level, as demonstrated in Figure 5 on page 5. Global shadow quality was also reduced. These changes resulted in a change of the average framerate from 10 FPS to 60 FPS - well above the threshold required by the performance metric.

5. Conclusion

To conclude, the project is progressing well against the plan. The project is also meeting the success measures decided at the start of the project.

The project has so far just implemented prototype systems, but the systems are extensible and scalable. The framework implemented will easily allow gameplay elements to be added and for increased graphical fidelity to be achieved.

This provides a high level of confidence that the project can achieve the objectives of being interesting, fun and of high graphical quality.

References

- Brown, M. (2016). How (and why) spelunky makes its own levels | game maker's toolkit.
- Dormans, J. (2010). Adventures in level design: Generating missions and spaces for action adventure games. pages 1:1–1:8.
- Ebert, D. S. and Musgrave, F. K. (2003). *Texturing & modeling: a procedural approach*. Morgan Kaufmann.
- Germer, T. and Schwarz, M. (2009). Procedural arrangement of furniture for real-time walkthroughs. *Computer Graphics Forum*, 28(8):2068–2078.
- Hendriks, M., Meijer, S., Van Der Velden, J., and Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 9(1):1:1–1:22.
- Perlin, K. (1985). An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296.
- Spufford, F. (2003). *Backroom Boys: The Secret Return of the British Boffin*. Faber and Faber.
- Taylor, J. and Parberry, I. (2010). Computerized clutter: How to make a virtual room look lived-in.
- Welsh, O. (2016). No man's sky review (eurogamer.net).

A. Gantt charts

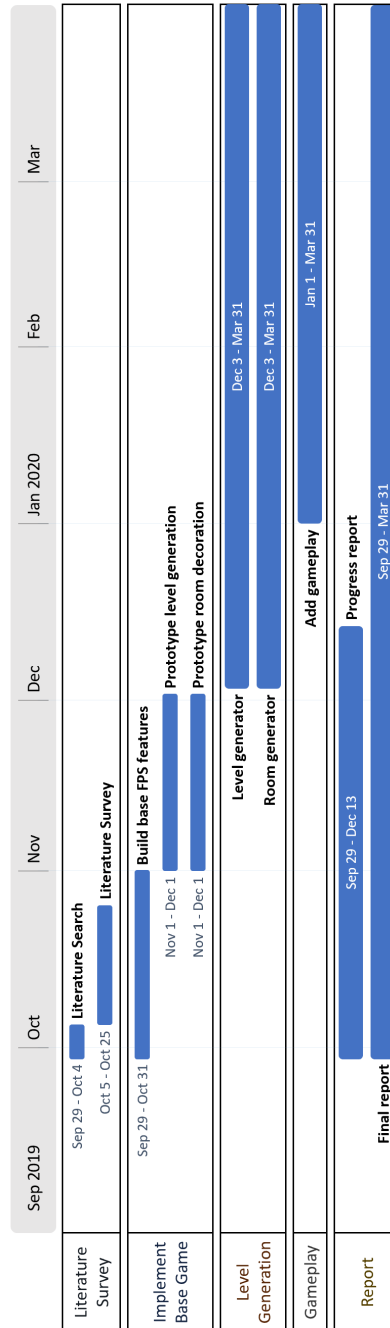


Figure 1: Gantt chart of original plan

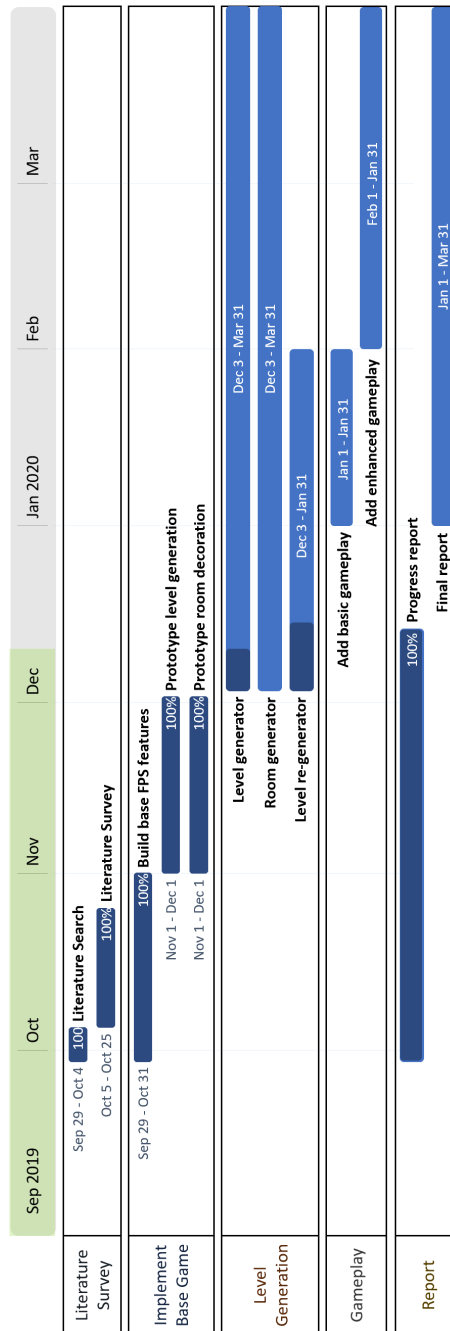


Figure 2: Gantt chart of updated plan

B. Diagrams

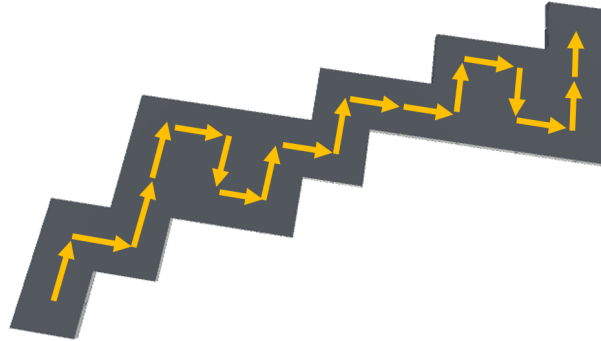


Figure 3: A top-down view of the guaranteed path generation, showing how the PCG algorithm moves in three directions to create a route through a maze

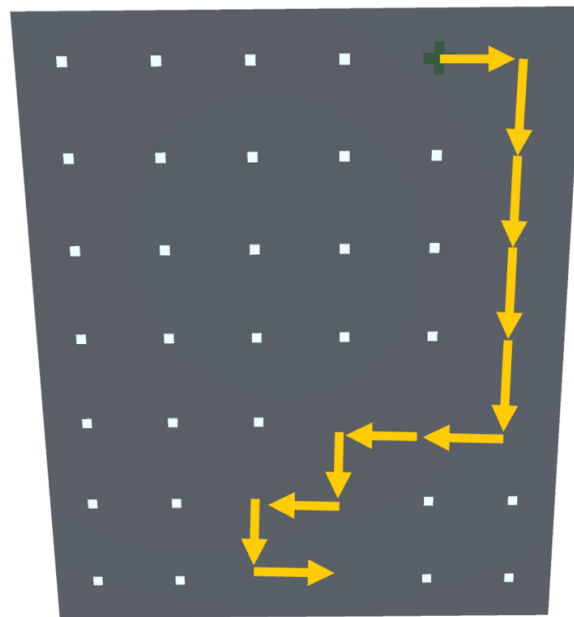


Figure 4: A top-down view of the guaranteed path with remaining tiles filled in, demonstrating how the filling algorithm works

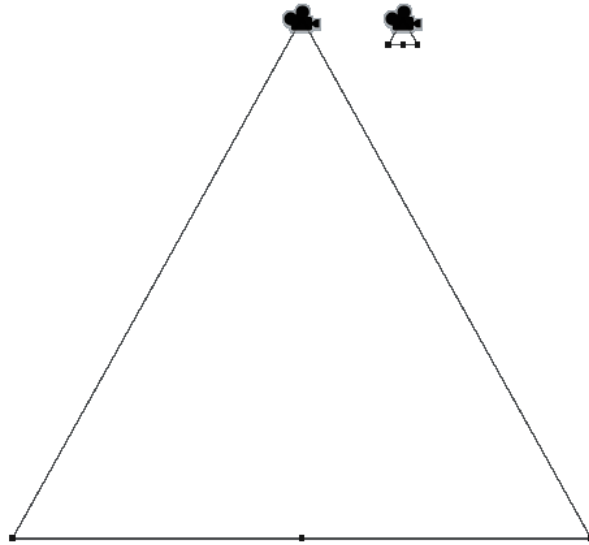


Figure 5: A top-down of the difference in default and optimised viewing frustum sizes, with Unity's default frustum size on the left and the optimised size for the generated mazes on the right

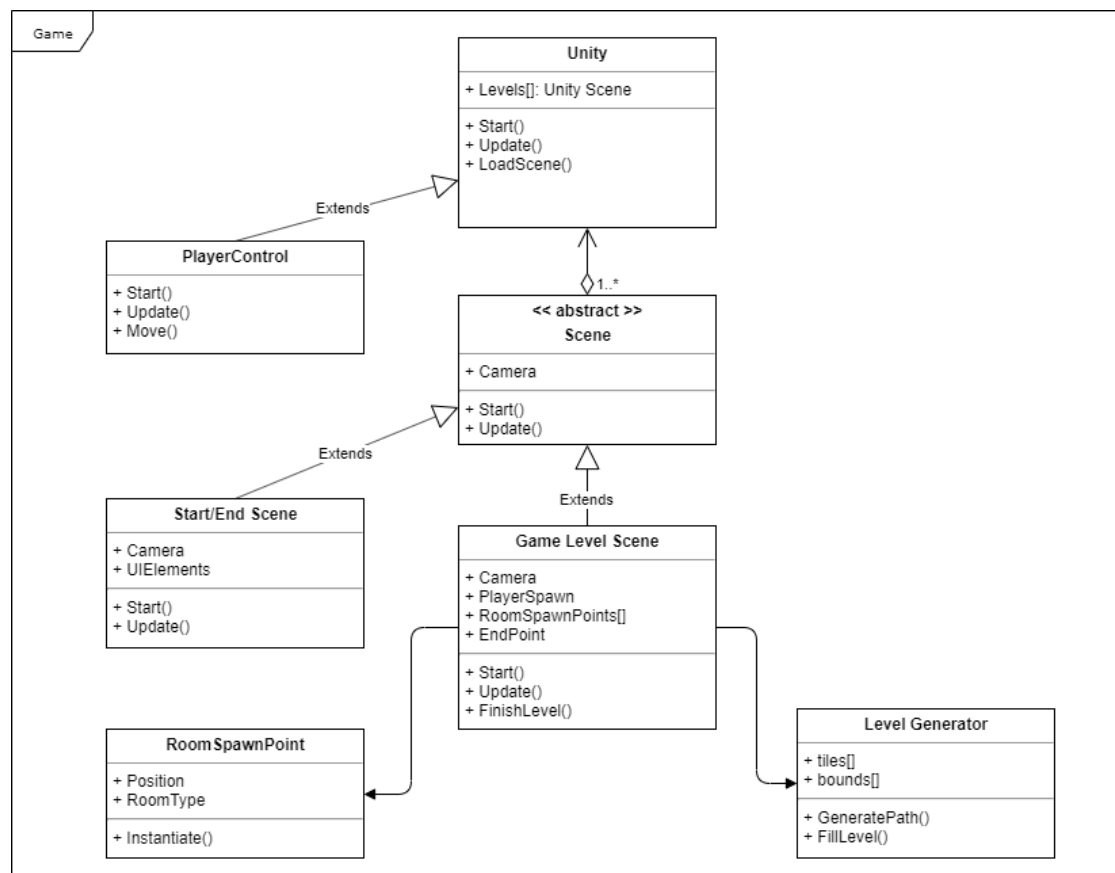


Figure 6: Initial UML diagram

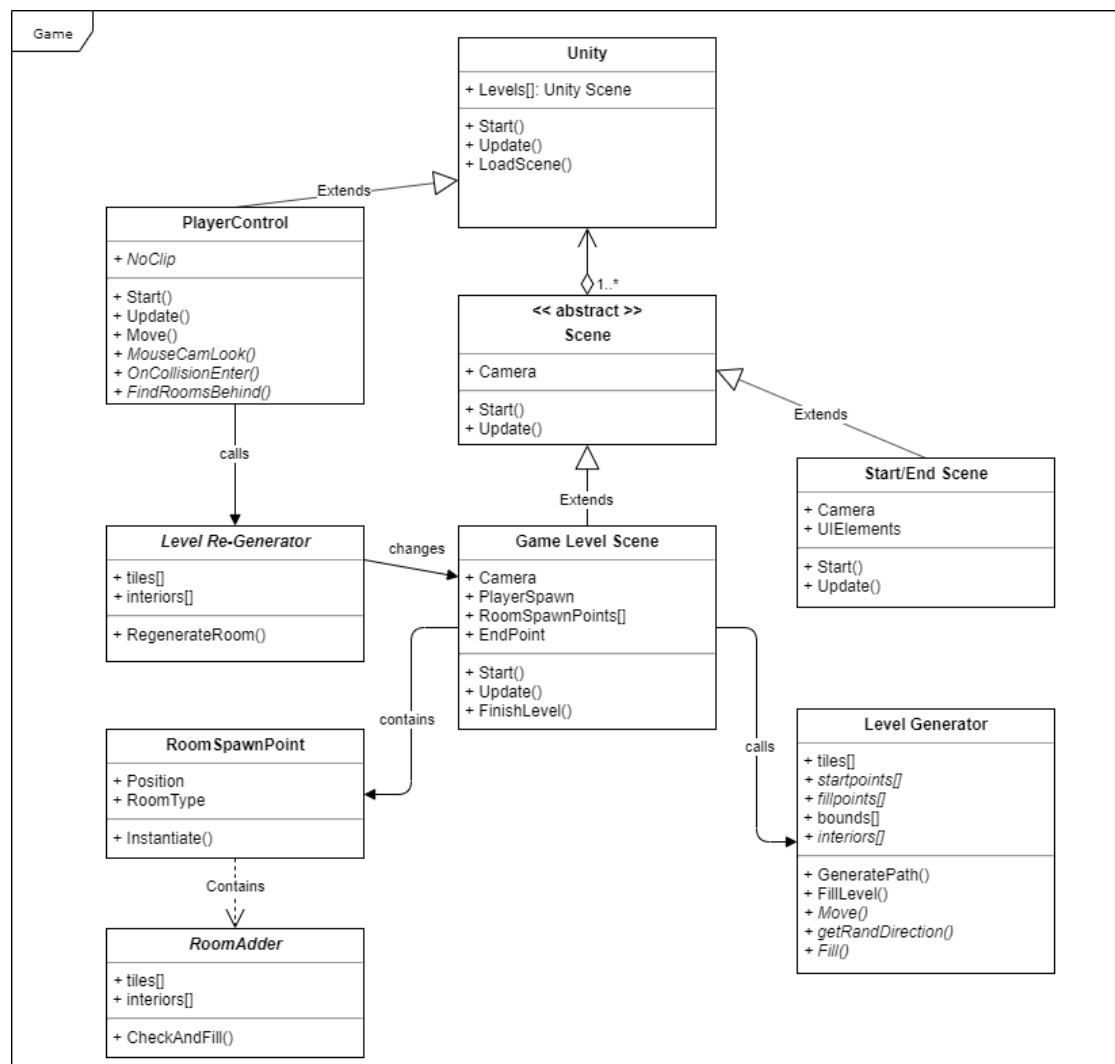


Figure 7: Redesigned UML diagram

C. Screenshots

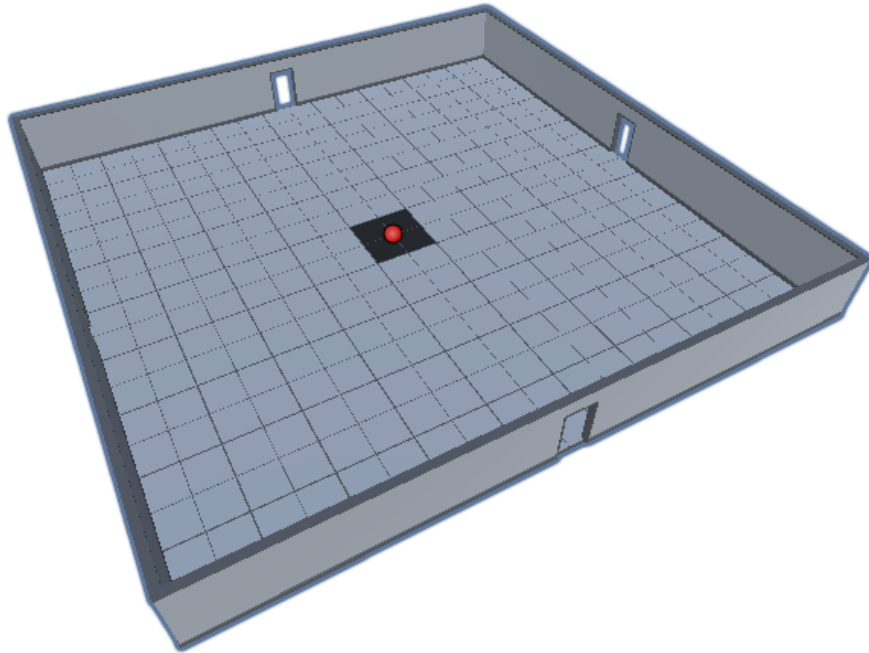


Figure 8: A screenshot of a room tile, showing the walls and exits of a tile (with the ceiling hidden)

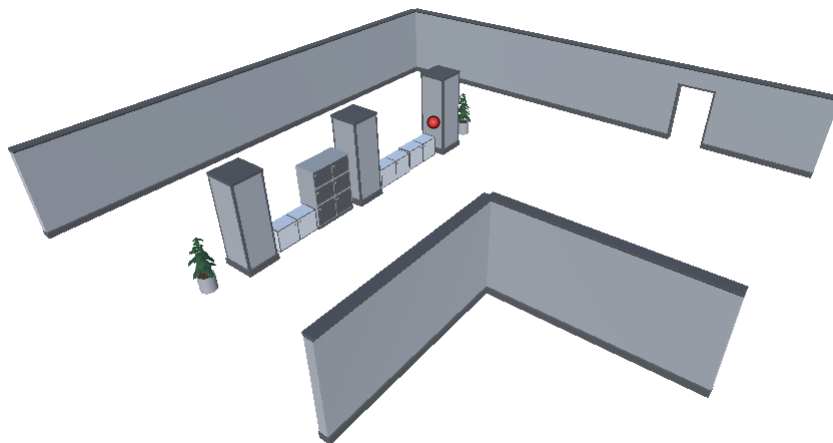


Figure 9: A screenshot of a room interior, designed to fit into any tile with any exit

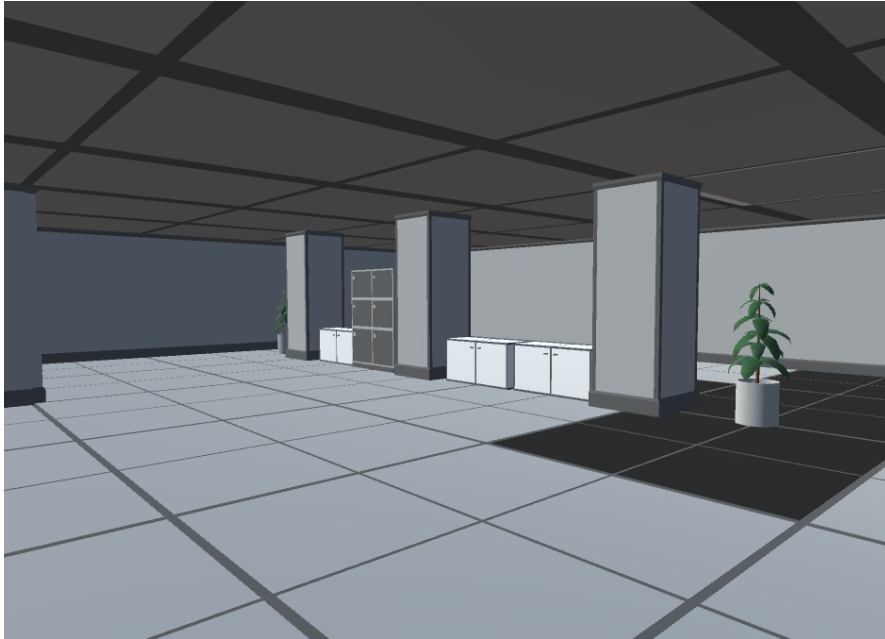


Figure 10: A screenshot of a procedurally generated room, with the tile and interior composited in a real-time environment

D. Tables

Epic	Stories
Build base "first person" 3D game features	Provides a basic level design Proves the concepts of vision and movement
Prototype level generation	Prove a method of generating a guaranteed path through a maze Prove a method of filling in non-guaranteed paths through the maze
Prototype room generation	Prove a method of generating room interiors Room generation must not interfere with guaranteed path
Prototype level re-generation	Design method to procedurally re-generate level sections
Add gameplay elements	Decide on how level is finished by the player Provide some interest and threat when playing Provide means of assisting navigation
Implement final designs	Use non-prototype textures Ensure performance meets goals

Table 1: Agile-style epics and associated user stories

Requirement type	Requirement	Priority	Status
Epic	Write Project Proposal	Should Have	Done
- Requirement	Research PCG	Should Have	Done
- Requirement	Write proposal	Should Have	Done
Epic	Write Literature Review	Should Have	Done
- Requirement	Research/find literature	Should Have	Done
- Requirement	Get feedback	Should Have	Done
- Requirement	Write review	Should Have	Done
Epic	Build base FPS features	Must Have	Done
- Requirement	Create title/game over screens	Should Have	Done
- Requirement	Create end goal, transitions to end screens	Should Have	Done
- Requirement	Implement first person controller	Must Have	Done
Epic	Prototype level generation	Must Have	Done
- Requirement	Design room tile prefabs	Must Have	Done
- Requirement	Implement "random walk" generation	Must Have	Done
- Requirement	Implement maze filling algorithm	Must Have	Done
- Requirement	Implement different sized tiles	Could Have	
Epic	Prototype room decoration	Must Have	Done
- Requirement	Design room decoration prefabs	Must Have	Done
- Requirement	Implement room decoration filling algorithm	Must Have	Done
- Requirement	Design random sub-decorator agents	Could Have	Done
Epic	Progress report	Must Have	Done
- Requirement	Design plan	Should Have	Done
- Requirement	Write first draft	Should Have	Done
- Requirement	Write final draft and submit	Must Have	
Epic	Prototype re-generation	Must Have	
- Requirement	Design approach for re-generating parts of level in real-time	Must Have	
- Requirement	Implement re-generation process	Must Have	
- Requirement	Implement re-generation trigger process	Must Have	
Epic	Level generator	Must Have	
Epic	Room generator	Must Have	
Epic	Level re-generator	Must Have	
Epic	Basic gameplay	Must Have	
Epic	Enhanced gameplay	Should Have	
Epic	Final report	Must Have	

Table 2: A table showing epics and requirements, their priorities and statuses

Left	Right	Down
40%	40%	20%

Table 3: A table showing probability of direction random walk agent will take

Level size	Generation time
4 x 6	6.34 seconds
6 x 10	6.89 seconds
15 x 30	7.28 seconds
30 x 50	9.61 seconds
70 x 100	13.20 seconds

Table 4: A table showing the project's level sizes in tiles and time it takes the procedural algorithms to generate the levels