Final Year Project Plan

Full Unit – Project Plan

Dungeon Crawler

Salma Bocus

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Julien Lange

Department of Computer Science
Royal Holloway, University of London

October 11, 2024

Declaration

This **project plan** has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: Salma Bocus

Date of Submission: 11th October 2024

Signature:

Table of Contents

Chapter 1: Abstract

The gaming industry is an influential and rapidly growing sector in technology. Games continue to evolve offering immersive experiences and complex mechanics and play an important role in digital culture. My motivation behind this project stems from my passion for gaming and game development. I am inspired to create a unique project to contribute to this field whilst combining my love for storytelling, art and technology.

Rogue-like dungeon crawlers have long been a popular genre in game development, recognised for their replayability driven by procedurally generated levels, permadeath mechanics, and gradually increasing difficulty. This project aims to design and implement a sophisticated rogue-like game that utilises advanced game development technologies, alongside well-established design patterns, to produce a fully deployable product. Core features of the game will include procedurally generated dungeons, a real-time combat system, a loot and inventory system, and player character progression.

I plan to develop the game using Godot, an open-source engine with support for both 2D and 3D game creation. Godot uses a native scripting language GDScript and has built-in support for exporting to multiple platforms. As part of the development process, I conducted research into several game development platforms and engines, including Unity3D, Unreal Engine, and Godot. Unity, though widely used in the industry, has a steeper learning curve and licensing costs for larger commercial projects. Godot, being open-source and lightweight, offers a simpler workflow, particularly with its GDScript language. The scene system in Godot provides flexibility and makes it well-suited for procedural generation tasks [1].

The primary technical challenge of this project lies in the implementation of procedural dungeon generation, which is key to creating varied and engaging gameplay environments. Algorithms such as cellular automata and random walk methods [2] will be implemented to ensure each level provides a fresh experience while maintaining balance and challenge for players. This procedural content generation will be essential in enhancing the game's core loop, offering players endless combinations of dungeon layouts [3]. The game will be developed with a focus on modularity and scalability, adhering to industry-standard software engineering practices. Design patterns such as Factory, Observer, and State [4] will be employed to structure the game's architecture and ensure it is extensible and maintainable. The Game Loop pattern will be at the core of the game's real-time mechanics, ensuring smooth and responsive gameplay.

In addition to procedural generation, the project will explore the development of an AI system for enemy behaviours [5]. Enemies will display varied and adaptive strategies, including pathfinding, attack strategies, and adaptive difficulty based on player progression. The AI will adjust to the player's progress, introducing new challenges as the player advances. This will be achieved using Godot's AI navigation features and custom AI algorithms like A* for pathfinding [6] and finite state machines [7] for managing enemy states during gameplay.

The game will also include a dynamic combat system, where players will be able to utilise different abilities, weapons, and power-ups. This system will be implemented to allow fluid interaction between the player and enemies, with real-time updates to game states. The Command pattern will be used to manage player input and actions, for flexibility and easy adaptation of different controls and interactions. Godot's input system can help simplify this process while supporting multiple input types for a wider reach across devices.

Godot's scene tree architecture will make it easier to create a modular system for user interfaces (UI) and game states. The game will feature an intuitive UI that displays essential information such as health, inventory, and map progression. Threading will be explored to ensure that UI elements remain responsive, even during intense in-game action. This will allow for multitasking and efficient execution of parallel tasks [8]. Godot's animation and particle systems will enable smooth character and environmental animations and implementing multithreading techniques can help ensure UI responsiveness improving overall performance and player experience.

The culmination of this project will demonstrate the application of software engineering principles in game development and will result in a fully functional game, showcasing technical proficiency in Godot and a strong understanding of modern game development frameworks and design patterns.

Chapter 2: Timeline

My timeline for this project outlines key milestones across two terms, with the milestones bold and underlined below. The timeline begins with research and familiarisation with Godot, followed by the development of core mechanics such as procedural dungeon generation, player movement and AI for enemies. The second term focuses on refining the combat system and adding animations, before concluding with testing, optimisation and deployment on itch.io. If time permits, I plan to extend the project by incorporating a multiplayer co-op mode, allowing players to collaboratively explore the dungeon and interact with one another.

2.1 Term 1

- Week 1:
  - Familiarise with Godot's interface, scene tree structure and GDScript.
- Week 2:
  - Research procedural generation algorithms with focus on cellular automata and random walk techniques; and how to implement in Godot.
  - Review Factory, Observer and State design patterns.
- Week 3-4:
  - Implement basic random walk algorithm in Godot using GDScript.
  - Use Godot's tilemap system to build grid-based system that generates rooms and corridors dynamically.
- Week 5-6:
  - Implement player movement with Godot's input system.
  - Integrate basic interaction mechanics – moving between rooms, interacting with doors etc.
- Week 7-8:
  - Start working on basic enemy AI using Godot navigation.
  - Implement simple combat system – allowing player to attack enemies and take damage. **Should now have player movement, enemies and combat.**
- Week 9-10:
  - Refine procedural generation algorithm – add variations in room types and enemy placements
  - Prepare for interim review – showcase progress in dungeon generation, player movement and combat mechanics.
- Week 11:
  - **Submit a working dungeon generation prototype with basic gameplay**.
  - Finalise the first term report.

2.2 Term 2

- Week 1-2:
  - Expand on enemy AI behaviours using finite state machines.
  - Develop advanced player abilities and power-ups.
- Week 3-4:
  - Add character and enemy animations.
  - Add particle effects for abilities, spells, environment interactions.
- Week 5-6:
  - Finalise UI elements – advanced inventory management, dungeon maps and end-game screens.
  - Integrate sound effects and background music.
- Week 7-8:
  - **Conduct playtesting sessions and collect feedback.**
  - Identify bugs and usability issues. Evaluate gameplay balance, UI, performance.
  - Optimise game for performance to ensure smooth gameplay on target platform.
- Week 9-10:
  - Prepare final game release for deployment on itch.io.
  - **Complete final report and prepare for viva.**

Chapter 3: Risks and Mitigations

3.1 Familiarisation with Godot

Godot is an engine that is new to me and therefore has a learning curve to master the scene system, GDScript and 2D/3D mechanics. By setting aside time early in the project, I can familiarise myself with the engine, its documentation and community resources. I can also focus on small experiments with Godot's systems before moving into full development.

3.2 Procedural Generation Complexity

Implementing procedural generation in Godot might present challenges such as performance issues with large dungeons and difficulty in creating balanced layouts. I can mitigate this challenge by starting with a simpler random walk algorithm and progressively adding complexity, testing performance as the dungeon size increases.

3.3 AI Development

Building an AI system in Godot will be time-consuming especially when handling multiple enemy types with different behaviours. I can begin with basic AI using Godot's pathfinding system and progressively enhance this with finite state machines. Moreover, I can test different AI behaviours in small, controlled environments before integrating them into my game.

3.4 Time Management

Learning the engine and experimenting with untested features may slow down development. By focusing on core mechanics first (such as procedural generation, movement and AI), sticking to the timeline and prioritising important features, I can manage my time effectively.

3.5 Exporting and Deployment

Exporting the game to multiple platforms, particularly for itch.io, may present unforeseen issues. The mitigation for this risk is to test early exports to HTML5 and desktop builds during the development process. I can also use Godot's extensive export options to ensure smooth deployment across platforms.

3.6 Feedback and Iteration Limitations

A limited access to play testers or feedback during development may hinder my ability to iterate on gameplay mechanics and identify issues early on. By planning regular playtesting sessions and allowing feedback through a survey for example, I can establish a clear feedback-loop enabling me to make informed adjustments to gameplay based on user experience.

Bibliography

[1] P. Hatton, “Godot Engine vs Unity: which is right for you?,” Creative Bloq, 16 July 2024. [Online]. Available: https://www.creativebloq.com/3d/video-game-design/godot-engine-vs-unity-which-is-right-for-you.

Web article discussing the differences between Godot and Unity which aided my decision on which platform to use.

[2] M. J. Nelson, J. Togelius and N. Shaker, “Constructive generation methods for dungeons and levels,” in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research,* N. Shaker, J. Togelius and M. J. Nelson, Eds., Springer, 2016, pp. 31-55.

Book discussing procedural content generation techniques, essential for dungeon generation. Also discusses cellular automata and random walk algorithms.

[3] J. Togelius, G. N. Yannakakis, K. O. Stanley and B. Cameron, “Search-Based Procedural Content Generation: A Taxonomy and Survey,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 3, pp. 172-186, 2011.

Research paper presenting a taxonomy and survey of procedural content generation techniques with a focus on search-based approaches that are particularly relevant for dungeon generation.

[4] R. Nystrom, Game Programming Patterns, 1st ed., Genever Benning, 2014.

Book providing an in-depth look at design patterns used in game development. This resource helped my understanding in structuring game code and introduced me to Game Loop, Factory, Observer and State patterns.

[5] S. Rabin, Game AI Pro 2: Collected Wisdom of Game AI Professionals, 1st ed., S. Rabin, Ed., A K Peters/CRC Press, 2015.

This book covers a range of topics on game AI, focusing on practical techniques used in real-world games. Useful for learning more about pathfinding, enemy behaviours and adaptive difficulty.

[6] I. Millington and J. Funge, “Pathfinding A*,” in *Artificial Intelligence For Games – 2nd ed.,* Morgan Kaufmann, 2009, pp. 215-237.

Section in a book detailing the pathfinding algorithm A in a clear and understandable way which allowed me to briefly discuss this in the abstract.*

[7] I. Millington and J. Funge, “Finite State Machines,” in *Artificial Intelligence For Games – 2nd ed.*, Morgan Kaufmann, 2009, pp. 310-315.

Different section in the same book previously that explains finite state machines for enemy behaviours.

[8] J. Gregory, Game Engine Architecture, Third Edition, CRC Press, 2018.

Book with detailed sections on multithreading in game engines which is relevant to my project's focus on smooth performance and responsive user interfaces.

[9] J. Linietsky and A. Manzur, “Godot Docs,” Godot Engine, 2024. [Online]. Available: https://docs.godotengine.org/en/stable/index.html.

Official documentation for Godot with tutorials. Essential resource for my project.

[10] GDQuest, “Learn GDScript,” GDQuest, [Online]. Available: https://gdquest.github.io/learn-gdscript/.

Online resource with video-based tutorials on how to develop games in Godot and building mechanics from scratch in GDScript. These tutorials will be useful in expanding my skills.

[11] J. Johnson, Godot 4 Game Development Cookbook, Packt Publishing, 2023.

Book detailing the features of Godot 4 with tutorials on advanced techniques. A useful future resource for my project.