Building a game

Project Plan

Prithvi Sathyamoorthy Veeran

CS3821 – Final Year Project

Supervised by: Julien Lange

Department of Computer Science

Royal Holloway, University of London

2024

ABSTRACT

With the game industry expanding rapidly over the last few years, the idea of games has drastically changed and adapted. With so many genres and styles that have been created, starting from the first pong game to the huge immersive world of rockstars Red Dead Redemption 2, building a game can take many different forms. Many aim to innovate and change the industry for good; some build on pre-existing formulas that are proven to be a staple within the audience; however, the idea of thinking outside box and pushing the limits of creativity and the current technology do not keep games bound to set ideologies, innovation is the main driving factor behind this all, and such an approach in this is exactly what I am aiming to achieve. This combination of ideas is exactly what I intend to take, with aims to design a 3-d top-down dungeon crawler game, a genre that is characterised by its exploration element of a procedurally generated map, with real-time combat and loot collection. Drawing inspiration from existing games such as the Diablo series and the Binding of Isaac. The game is set in a fantasy world where you, as the player, are tasked with navigating the dungeon, facing enemies, solving puzzles, and collecting items. This has the potential to demonstrate strong AI design, level generation, and player feedback.

The Godot engine has been selected for this game due to its robust open-source nature and powerful development capabilities. Godots node-based architecture and GDScript (similar to the Python language) provide an accessible but powerful environment for rapid prototyping and feature development. Also, it has great support for procedural generation, and AI behaviours make it a great engine to use. Procedural generation has been an extensively studied topic in game design as a method for creating the element of replay ability effectively (Smith et al., 2010), and this project will be building upon established algorithms such as cellular automata and random walks (Shaker et al., 2016). Furthermore, Godot's lightweight framework allows for the game to run smoothly and efficiently, even on lower-end hardware, which will broaden the potential audience. The documentation and community tutorials (Godot Engine Documentation, 2024) will be vital for this project, as they provide extensive resources that will help to resolve potential challenges encountered during development.

Some of the key functionalities of the game will include procedurally generated content. This will allow for endless combinations of map layouts and enemy encounters. This feature has become very popular in the modern game development scene, offering the ability to extend the gameplay without having to manually design and add each level (Smith et al., 2010). Combat and exploration are main features to be seen as well. Combat will take place in real-time, with the player having to control their unique character through the dungeon, fighting enemies, using different abilities to do so. Drawing from the Diablo series, the combat system will involve basic attacks, a way to dodge/block, and powerups, creating an engaging and strategic combat experience for the player. The health system, item collection, and the possibility of traps add layers of difficulty to the game, along with the added puzzles along the way to create further engagement and critical thinking of the player.

The AI for enemies will be another critical feature, requiring them to behave intelligently in coordination with the player. These things include simple enemy patrol paths, detecting player, and engaging in combat with them, while more complex AI types could feature teamwork, ranged attacks and ambush attacks.

While the game concept can be seen as ambitious, there will be several technical challenges that I will face, such as the procedural generation and the AI behaviour. Procedural generation requires that I carefully balance the random elements, such as enemy placement and room layouts - things that can be error-prone. This will be a good learning curve for myself to ensure each dungeon feels cohesive and fair to the player. Additionally, implementing an AI that reacts intelligently to the player is a significant challenge too. Enemies should pose a threat while also being fun to fight.

The main goal of this project is to create a game, that is repayable and successfully combines elements such as procedural content generation with real time combat and an intelligent AI system, with a smart code base that is complex in many ways. The idea behind games is to let the player feel like they are part of the world they are experiencing, and delivering that in a polished and engaging way is important.

TIMELINE (MILESTONES)

The timeline for the development of this game spans over the academic year, split into two terms. The project will be split into several phases to ensure that both the design and the implementation of the game are progressing well. Each phase will include specific milestones, ensuring that development stays on track and allows for flexibility for any unforeseen changes. As seen below:

1. **Pre-Production Phase (week 1-3)**

- **Objective:** Establish a clear foundation for the project, such as conceptualising the game mechanics, setting up the development environment, and preparing the initial assets. Conduction of research will still be going on, as well as research into the game engine.
    - **Week 1-2:**
        - Initial project planning and setup.
        - Study Godot engine and mechanics along with papers relevant to the mechanics of the game.
        - Create the initial project design document.
        - Meet with the supervisor and finalise ideas.
        - Start the draft of the project plan.
    - **Week 3:**
        - Carry out learning on the Godot engine.
        - Complete tutorials of basic core functions.
        - Complete the project plan.

2. **Core system development (week 4-10)**

- **Objective:** Focus on developing the games core gameplay mechanics, including the menu screens for the player to navigate, player controls, enemies (and their AI), dungeon map creation, basic combat mechanics, and a character class and stats section. The goal here is to have a playable prototype by the end of this phase. Alongside all of this, it is important to make sure that the report is being written up while the development process is being carried out.
    - **Week 4-5:**
        - Create the menus desired for the game, along with fluid connectivity between them.
        - Implement player movement and a basic combat system.
        - Test to see if these mechanics are suitably implemented.
    - **Week 6-7:**
        - Set up the player HUD.
        - Begin development of enemies and their factories.
    - **Week 8-9:**

        - Set up collision with player, enemies, and the placeholder map.
        - Test simple player, enemy, and map interaction.
    - **Week 9-10:**
        - Develop the procedural generation of the dungeon.
        - Integrate player and enemy mechanics into procedural levels.
    - **Week 11-12:**
        - Add any other features if there is time, such as power-ups, and start the development of the puzzles and mini games.
        - Refine some of the mechanics and refactor.
        - Consider all test cases for everything developed so far.
        - Prepare for the interim review, the 5000-word report and the demo code.

3. **Milestone: first playable prototype (end of term 1)**

- By the end of term one, there will be a working prototype, featuring the basic functions of the game such as exploration, combat, and rudimentary dungeon generation. In-house playtesting will begin at this stage, and the feedback from this will be collected to guide the development into the next term.

4. **Advanced features and refinement (Week 11-17)**

- **Objective:** In this phase I will refine core systems, add more complex features, such as advanced enemy AI, further intricate level design, and a more fleshed-out combat design. This phase will also contain debugging and playtesting.
    - **Week 11-13:**
        - Consider the feedback from the playtesting and adjust the timeline to fit in these comments.
        - Enhance the enemy AI (pathfinding, group behaviours).
        - Add more enemy types with potential unique abilities (bosses (optional)).
    - **Week 14-15:**
        - Implement the power-up and pickups system (inventory system).
        - Refine the procedural generation (environment obstacles, traps).
    - **Week 16-17:**
        - Test all the mechanics in the game so far.
        - Bug fixing and early optimisation is to be conducted.

5. **Milestone: Alpha version (mid-term 2)**

- At this stage, the game should be feature complete in terms of the core elements, with all the major mechanics functional. The focus of the project will now shift to finalisation, fixing bugs, polishing the game, and if there is extra time, any additional optional features.

6. **Polish and final testing (weeks 17-23)**

- **Objective:** Focus on the refinement of the game experience, including the visuals, user interface, and any other extra features. Extensive playtesting will be done to make sure the game is polished.
    - **Week 17-19:**
        - Finalise all the assets requires
        - Polish procedural map generation.

- **Week 20-22:**
  - Conduct final playtesting.
  - Resolve any remaining bugs and make final optimisations.
- **Week 23:**
  - Prepare the final submission.
  - Prepare for the final report and demo.
  - Publish game.

7. Milestone: Final submission (End of term 2)

- The game should now be complete, along with the final report and the demo for it. Any remaining work will be concluded.

RISKS AND MITIGATIONS

With the development of a game and any project for that matter, there are bound to be risks that will be involved and appear during the process. Identifying these risks early allows for the appropriate planning and mitigations, ensuring that the project stays on track and doesn't severely deviate from the planned outcomes. Below are the risks that are anticipated in the project, alongside proposed mitigation procedures.

Risk	Description	Impact	Probability	Mitigation	
				Prevention	Reaction
Unfamiliarity with the Godot engine	Albeit a powerful engine, it is my first time using it, posing a potential learning curve along with its custom scripting of GDScript.	Medium - May lead to delays in the early development stage as learning is still being conducted, leading to slower progress overall.	High	Allocate time early on to do focused learning through the tutorials provided and read through the documentation. Prototype early critical mechanisms required within the project to minimise risks later on.	Consult community forums, or if really needed switch to simpler mechanics and scale back complexity (last resort).
Integration of complex AI enemy behaviour	Designing AI that can react intelligently in real-time and navigate	High – this is a core element of the gameplay	Medium	Start with basic AI that use simple movement	Simplify the enemy behaviour if issues

	procedurally generated maps is a challenging task and could result in delays and will most likely require multiple iterations to refine.	and project, therefore without functional AI enemies, combat and game interaction will suffer, reducing the player experience. This will also delay further mechanics which will require a lot of time in the future, if this get pushed back.		and pathfinding and keep building on this gradually. Test the AI early and iteratively.	arise, focus on basics. Scale down if needs be, prioritise gameplay mechanics over enemy complexity.
Procedural dungeon generation	This is my first time trying to implement procedural generation into one of my games so there will be a significant amount of time needed for this feature, furthermore random generation may lead to unplayable or unbalanced layouts, requiring significant debugging.	High – if the procedural generation takes too long or doesn't work then other plans with have to be considered such as premade maps etc, reducing the complexity of this project. Broken layout may make the game unplayable	High	Begin with simple maps, then add complexity. Prototype early and playtest often to identify any generation issues early on.	Revert to predefined map layouts until problems are resolved. If problems persist potentially move to a hybrid generation system (partially procedural, partially hand-designed).
Overcomplicating game mechanics	Having too many game mechanics can result in poor code and half working mechanics, extending development time and then the debugging and refinement phase.	Medium – the player experience may be negatively impacted leading to confusion. It will lead to many code smells within the directory, and a 'half-baked' game.	Low	Use the "minimum viable product" (MVP) approach; focus on the essentials first. Expand mechanics later on based on time and complexity. Test frequently.	Remove non-essential mechanics or move them to a time where they could be implemented as optional features.

Visual and audio design overload	There is a risk of spending too much time on visual and audio design. Overly ambitious asset creation can lead to scope creep.	Medium - This could delay the implementation of core gameplay mechanics as not being able to find correct assets can be frustrating, leading to more unnecessary time being spent on it.	Low	Use placeholders for graphics early on in development. Focus will remain on functionality. Free and open-source assets will be utilized where possible to expedite development.	Decide on graphics that fit the purposes of the game in later stages of development to save time.
Time management	Balancing the work done on the project and other university modules/commitments may pose a slight problem, due to the nature of final year university there is going to be a substantial workload. I also have a part-time job, which I attend 3 times a week, 5pm-12pm.	High – lack of time due to these may lead to rushed, unfinished, or lower quality features, affecting potential grade, code and tidiness of everything.	Medium	Create a detailed weekly schedule, prioritise tasks using a task management system (Trello/ Microsoft to do). Track my progress against my milestone regularly. Adjust the scope early if any restraints arise.	Relocate non-essential tasks to a later date if other deadlines interfere. Drop them if time becomes an issue.
Underestimating project scope	Due to the nature of things being implemented into this game, the complexity and time required may be underestimated, leading to a rushed and disorganised development process.	High – missed deadlines or incomplete features heavily impact the project as well as the grade.	Low-medium	Make sure the project timeline is accurately developed early on.	As above, reduce or eliminate non-essential mechanics. If major issues that may jeopardize the project seek help from the supervisor.
External dependencies	Issues may arise from with any external resources or assets which	Low – missing or incorrect assets could	Low	Use placeholders until the final versions are	Continue development with placeholde

	could slow down the development process. Especially if external libraries are used.	stall development process.		available and test to see if they are compatible.	rs if needs be. Reduce reliance on external tools where possible.
Late discovery of bugs	With any game, bugs discovered in the late stages could delay the final product or compromise the games functionality.	High – any bugs discovered in the core functions of the game could make the game unplayable or lead to a product that is unpolished.	Medium	Plan and implement a regular testing schedule after each feature is implemented. Try to address or find bugs in core elements early on.	Use version control to revert back to a previous build.

BIBLIOGRAPHY

Adams, E., & Rollings, A. (2010). *Fundamentals of game design*. New Riders.

- This book offers foundation level knowledge of game design principles, such as the player engagement, creating immersive worlds and balancing difficulty. It will help in structuring the game mechanics and guide the overall game approach.

Blizzard Entertainment. (1996). *Diablo* (Version 1.0) [Video game]. PC. Blizzard Entertainment.

Blizzard Entertainment. (2000). *Diablo II* (Version 1.0) [Video game]. PC. Blizzard Entertainment.

- Both games are pioneering titles in the action RPG and dungeon crawling genre, with their core function driving the inspiration for my own game. analysis of these games will help in the development overall.

Dahlskog, S., Björk, S., & Togelius, J. (2015). Patterns, dungeons and generators. In *Foundations of Digital Games Conference, FDG, Pacific Grove, USA (2015)*. Foundations of Digital Games.

- This paper explores patters in procedural generation. It offers a theoretical foundations to help design and improve the way the algorithm is implemented into my game.

Godot Engine Documentation. (2024). *Godot Engine Documentation*. Retrieved from https://docs.godotengine.org/.

- The official documentation for Godot which will be essential for developing the game technically., as it provides the insights into the tools, functions and optimisations within the engine. This resource will be used throughout the whole project.

McMillen, E. (2011). *The Binding of Isaac* (Version 1.666) [Video game]. PC. Edmund McMillen.

- Another highly successful game in the genre of dungeon crawling. This game incorporates rouge-like mechanics, which will influence the system which will be touched upon in the project.

Millington, I. (2019). *AI for Games*. CRC Press.

- This text offers an in-depth analysis of AI techniques, which will guide the development of intelligent enemy behaviour and adaptive AI systems in the game. The book provides some algorithms and methodologies for creating an engaging enemy AI.

Nystrom, R. (2014). *Game programming patterns*. Genever Benning.

- Nystroms book will assist me in structuring my code efficiently, making it scalable and easier to manage, which is crucial to the projects long term maintenance, as it presents essential design patterns for game development.

Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer International Publishing.

- This book provides an overview of procedural content generation, which is key to my game. it will help guide my knowledge of procedural generation, contributing to diverse levels and replay ability.

Smith, G. (2014, April). Understanding procedural content generation: a design-centric analysis of the role of PCG in games. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

- This paper gives me a design-centric view of procedural content generation, focusing on how it enhances game design. This is relevant to me as I want the levels to be meaningful and not random.

Valtchanov, V., & Brown, J. A. (2012, June). Evolving dungeon crawler levels with relative placement. In *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*.

- This paper provides techniques on generating dungeon levels, especially focusing on the relative placement of objects and challenges. This helps me on my design for levels, such as the puzzles, obstacles and traps which is another key aspect of my game.