|  |  |
| --- | --- |
|  | Building a game |
|  | Project Plan |
|  | Prithvi Sathyamoorthy Veeran  CS3821  Supervised by: Julien Lange |
|  | Department of Computer Science |

# 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 of pre-existing formulas which are proven to a staple within the audience; however the idea to keep thinking out the box and pushing the limits of creativity and the current technology do not keep games bound to set ideology’s, 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. The game is set in a fantasy world where you as the player is tasked with navigating the dungeon, facing enemies, solving puzzles and collection 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.

# 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 the development is on track and allows 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 prepare 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 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 project plan.**

1. **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.**
  + **Week 4-5:**
    - **Create the menus desired for the game, along with fluid connectivity between them.**
    - **Implement player movement and 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 powerups 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.**

1. **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 guid the development into the next term.**

1. **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 optimization to be conducted.**

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

* **At this stage, the game should be feature complete in the 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.**

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

* **Objective: Focus on the refinement t 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 the 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**

1. **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, 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 minimize risks later on. | Consult community forums or if really needs be 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 procedurally generated maps could result in delays and will most likely require multiple iterations to refine. | High – this is a core element of the gameplay 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. | Medium | Start with basic AI that use simple movement and pathfinding and keep building on this gradually. Test the AI early and iteratively. | Simplify the enemy behaviour if issues arise, focus on basics. Scale down if needs be, prioritize 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, 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. 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, prioritize 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 could slow down the development process. Especially if external libraries are used. | Low – missing or incorrect assets could stall development process. | Low | Use placeholders until the final versions are available and test to see if they are compatible. | Continue development with placeholders 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

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