# A screenshot of a computer AI-generated content may be incorrect.

# 7 – Implementation

## 7.1 Project Setup and Design - Sprint 1 (8.11.2024 – 22.11.2024)

Overview

This was the first sprint of the project, after attending the required seminar, I chose the type of software project I wanted to develop – that being a game, and selected an assistant supervisor to support the project. The choice of creating a game came easy to me as I aspire to work in the games development industry after graduating, with this project complete it would stand as a solid portfolio piece I can show to employers and display on my portfolio website (morganhodge.co.uk). This project would also strengthen my skills as a game developer, through project management , good use of version control, and 3D Unity skills.

During this first sprint I focused on outlining the initial concept, goals, and setting up my workspace. Following guidance from the seminar, I began drafting the Game Design Document(GDD), which would act as a foundational reference throughout the development process.

Sprint Tasks

* Set Up Work Environment
  + Create Github Repository
  + Set up Github Desktop
  + Create new Trello board and give access to supervisor
  + Gather all required information such as briefs and required documentation and add them to the repository
* Level Design (Paper Based)
  + Level 1
  + Level 2
* Basic Player Movement
* First Person Camera Movement (Mouse)
* First Person Walking Movement (Keyboard)
* Test Scene Created

Summary Of Actions

The player movement created for this first sprint was required as I wouldn’t be able to proceed with the development or test anything I would be implementing if I could not move around the scene.

The code implementation shown in Figure XYZ demonstrates how player movement was set up. One of the strengths of this system is its flexibility, key parameters such as movement speed, jump force, and jump cooldown are easily adjustable. This allowed for efficient playtesting and balancing throughout development, without the need to rewrite core logic.

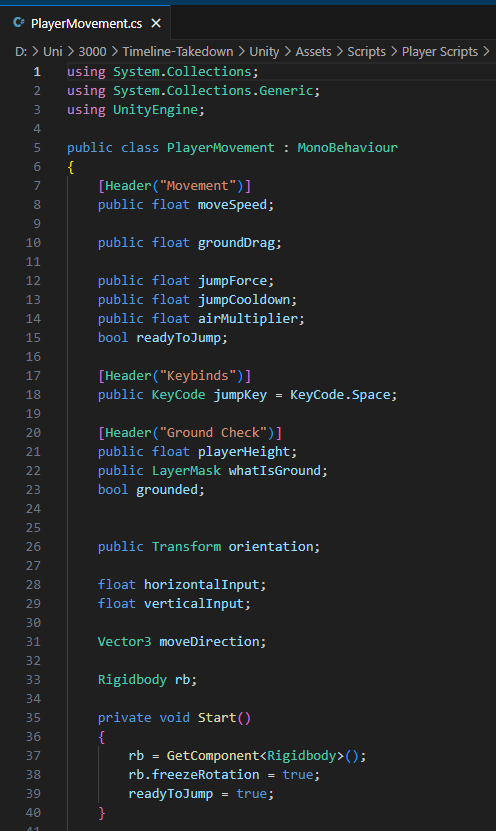
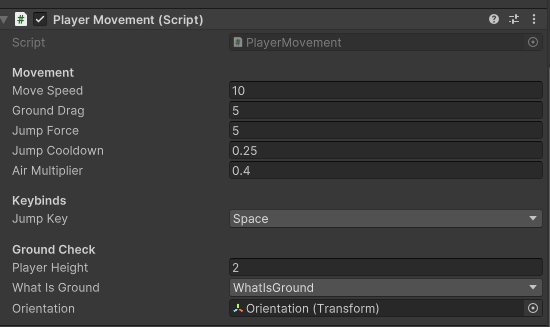
 

Figure XYZ FIGURE XYZ

Additionally, the paper-based level designs were created in this sprint but were not displayed until the Devlog 1.

## 7.2 Research and Game states - Sprint 2 (22.11.2024 – 6.12.2024)

Overview

This second sprint mostly consisted of setting up game states such as a Start Screen, Pause menu and settings tab.

Research was also conducted in this sprint, exploring art styles I could use as well as potential assets for the enemy type – this was documented within the Devlog 1.

During my research into art styles and visual design within video games, I was recommended several books that would be relevant to game development. As a fan of the franchise they are based on, I was eager to purchase these books and learn from them. The books I purchased during this sprint and studied included:

* The Art of Fallout 4 – A detailed look at the environment and character design choices made in fallout 4 , this was useful reference material for post-apocalyptic and sci-fi aesthetics.
* Elden Ring Art Book Vol1 – This book offered insight to the world building , creature design and atmospheric art direction that was used in Elden Ring, this helped me understand how to setup an atmosphere in a level using certain colours and audio.
* Blood Sweat and Pixels by Jason Schreier – This is not an art book but was highly recommended within the game’s development community. This was an insightful read as it provided a look on how games are really constructed and the struggles and triumphs of game development across multiple different studios. This was a strong motivational resource during my own development.

Sprint Tasks

* Devlog 1 created – <https://www.youtube.com/watch?v=Su4Demj-MFw>
* Paper Based Concept Art
* Created basic UI on Inkscape
* Game states created Menu-Start-Pause
* Movement tweaking

Summary Of Actions

Concept art was sketched up, this was the result:

A paper with drawings and words

AI-generated content may be incorrect.

*Figure X, Concept Art*

The UI elements were initially created using Inkscape. I approached this stage knowing that these assets would just act as placeholder visuals, rather than the final design. The goal at this stage was to have functional UI in place for testing gameplay mechanics and user interactions. An example of this early placeholder Art can be found below in figure XYZ.

A screenshot of a video game

AI-generated content may be incorrect.

*Figure X , Basic UI created in inkscape*

When researching into how to create the different game states (such as main menu, pause, game over) I thought I would have to create separate Unity scene for each state and manage them using scene indexes in the build settings. However, through further research and advice from online resources, I learnt that using separate canvas objects with a single scene was a much more efficient approach. This method allowed for better performance, 0 loading time, easier UI management and smoother transitions between game states.

## 7.3 Assets and Online Resources - Sprint 3

Overview

This sprint primarily focused on researching suitable online assets and exploring AI tools to generate temporary placeholder art. I aimed to find assets that matched the intended visual style and tone of the game while also making sure they were suitable for implementation into Unity.

During this stage, I created a test scene where I imported and arranged some of the selected assets as this allowed me to visually evaluate how well they fit and the overall aesthetic.

Sprint Tasks

* Devlog 2 created – https://www.youtube.com/watch?v=RXlF7QE14os
* Created Document of assets
* Created a test level and imported certain assets
* Used tools to generate UI and Loading Screens
* Started to create a block out level for game mechanics to be tested

Summary Of Actions

When creating the document of assets I would be potentially using, I focused on finding free, copy safe assets that could be used within the game. To keep track of the resources I found I created a word document compiling the names and download links of each asset. A copy of this assets list can be found in Appendix X.

During a recent supervisor meeting I was recommended an AI tool called DeepAI, which is capable of generating images for free. Following this advice, I used DeepAI to produce placeholder art assets for the game. Since developing fine visuals was not a top priority at this stage, using AI generated assets allowed me to maintain visual consistency while focusing on gameplay functionality.

Lastly, I downloaded a selection of the most appropriate models from the asset list I had compiled. These assets were then imported into a test scene in Unity, allowing me to evaluate their visual compatibility with the games intended style and theme. This hands-on assessment provided immediate visual feedback, helping me decide whether each asset was suitable for continued use or needed to be replaced or modified.

A screenshot of a computer game

AI-generated content may be incorrect.

*Figure xyz : A screenshot from my devlog showin the models imported*

## 7.4 Enemy & Main Game Loop - Sprint 4

Overview

At this stage of development I was working on the games functionality you could now end the game by picking up “end game” part. Enemies are now implemented, they have the ability to locate and follow the player, and adjustable parameters like speed. Additionally, the player now has a health system, allowing them to take damage, die, and regenerate health.

Sprint Tasks

* Devlog 3 – https://www.youtube.com/watch?v=vxEHmGPZ-y4
* Health Functionality
* Enemy Controller
* Level can be completed
* UI
* Map Design on Dungeon Scrawl

Summary Of Actions

To enable enemy pathfinding and movement I had to first bake a NavMesh onto the ground surface within the scene. This allowed the enemies to navigate the environment using Unity built in NavMesh system.

I then create a script that uses Unity’s NavMeshAgent component to control the enemy’s movement. The script constantly updates the enemy’s destination to follow the player’s position. To avoid potential runtime errors, I added a check that checks to see if the enemy is still alive, that the NavMeshAgent is still active, and that it remains on a valid section of the NavMesh before attempting to move.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure XYZ: Enemy Controller

During this sprint, the players health system was implemented, allowing the player to take damage and heal. If the players health bar reaches zero, the game ends. However, enemies are not yet capable of dealing damage. To test this functionality of the health system, I created a test object that when interacted with would damage and heal the player. This was used as a temporary method to test the health mechanics until enemy interactions could be fully integrated.

Lastly, I transferred the paper-based level designs to digital format using a tool called Dungeon Scrawl, this was recommended by my supervisor. This was beneficial as it provided a clearer visual representation of the levels, and this was crucial when trying to translate designs into Unity.

A white paper with writing on it

AI-generated content may be incorrect.A screenshot of a game

AI-generated content may be incorrect.

Figure XYZ: TOP PAPER BTM DIGITAL

## 7.5 Minimum Viable Product - Sprint 5

Overview

At this stage, the game had reached its Minimum Viable Product (MVP) phase, meaning it had the core functionality required to be considered a working project. The main game loop was fully implemented, including the wave manager, which handles enemy spawning and progression. Projectile mechanics were set up, allowing the player to attack enemies. Enemies were now able to both take and deal damage and were set to despawn upon death as animations had not been implemented yet. Additionally, the wave system was functioning correctly, with new waves only starting after the appropriate number of enemies had been defeated in the current round.

Sprint Tasks

* Early Stage of Wave manager
* Projectiles
* Enemy Spawn Points
* Enemies can take damage
* Start of Power Up implementation

Summary Of Actions

The wave manager script is a central part of the games round based survival system, it controls the flow of enemy waves, tracking the progress of the round through the use of Unity Text Mesh Pro UI, and manages the boss spawning.

The script begins by checking the current wave number and uses a predefined array called enemiesPerWave to determine how many enemies should be spawned for each wave. If the wave is marked as a boss wave, a boss enemy is also spawned. To prevent duplicate spawns, a bossSpawned flag ensures the boss is only spawned once per wave.

Each time an enemy is defeated, the method EnemyDefeated() is called which increments the kill count, but this may be removed by the final release as this is mostly used as a debugging tool and not a final game feature. Also, the number of enemies remaining in the current wave is decreased and the UI is updated to reflect this.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure XYZ Enemy Defeated Method*

When all enemies in a wave are defeated, the game waits a few seconds using WaitForNextWave() before transitioning to the next wave using the StartNewWave() func. Once the final wave is complete, the script plays a victory audio que and stops any more waves being spawned.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure XYZ*

The enemy spawner is responsible for spawning enemies and bosses during gameplay, based on wave progression that is integrated within the wave manager. It allows me to set spawn points and enemy prefabs through the Unity inspector, allowing me to have full control and flexibility over how and where the enemies appear.

When a new wave starts the StartWave() method is called, which begins a coroutine SpawnWave() that instantiates a set number of enemies at randomised spawn points. A delay between spawns is needed or multiple enemies can spawn at the same spawn point , causing issues such as overlapping or leading to the enemies damaging each other.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figute XTZ*

Whenever an enemy is defeated, EnemyDefeated() is called, which passes that information back to the wave manager to update the wave progression and necessary UI elements.

Projectiles

## 7.6 Minimum Viable Product - Sprint 6

Overview

Sprint Tasks

* Devlog 4
* Public Play Test
* Level 1 Pretty much completed
* Ambience sound script
* A lot of doing L1 Asthetics

Summary Of Actions

## 7.7 Poster & Level 2 - Sprint 7

Overview

Sprint Tasks

* Poster
* Level 2
* Gamestates
* Level 2 Design

Summary Of Actions

## 7.8- Sprint 8

Overview

Sprint Tasks

* Devlog 5
* Level 2
* Playtest

Summary Of Actions

## 7.9- Sprint 9

Overview

Sprint Tasks

* Devlog 6
* Refinements
* Fixing stuff

Summary Of Actions

## 7.10- Sprint 10

Overview

Sprint Tasks

* Devlog 7
* Polishing everything

Summary Of Actions

## 7.11 Uploading the Game – Sprint 11

Minimal Viable Product

Minimum Awesome Product