My Portfolio

- Siddharth Shankar

Game Designer

I am game designer and video editor with proficient experience on Unreal engine [5] and Davinci Resolve respectively. I am extremely comfortable using Blender have made assets and animations for my games.

I like making games for the same reason I love playing them, it just comes naturally to me. I have been making games in some sense of the word since childhood and the first program I ever wrote was a number guessing game that I wrote in C++ back in 2020. Then I started making small games on game making websites and eventually unreal engine 4 and 5.

In this document is all of the work I have done on games and 3d software that Is not lost to time and space.

1. Dragon Duel: A Combat Vertical Slice

[2026 – Unity Vertical Slice Project]

Play Here: https://sidjoey.itch.io/dragonduel
Source Code: https://github.com/SidJoey/DragonGameAssesment_Code
All Builds: https://github.com/SidJoey/Dragon_Duel_Builds

1.1 Introduction

Dragon Duel began as a technical assessment for a Unity Developer role.

The original requirement was to build a small arena battle between two dragons, one player-controlled and one AI-controlled, with three abilities and a functional UI.

After submitting the assessment, I chose to expand the project into a polished vertical slice. My goal was simple:

Ship a small, complete Unity game quickly, learn deeply through execution, and build momentum by finishing something properly.

The core gameplay loop stayed consistent from the first version to the final build:

Fight → Try to win clean → Chase a perfect score → Run it back.

Rather than adding unnecessary systems, I focused on reinforcing that loop and polishing everything around it.

1.2 Tools

- Unity 6 (URP)
- C#
- Shader Graph, custom fireball effect
- Unity UI, Canvas system
- NavMesh for AI movement
- Cross-platform builds, Windows, Android APK, WebGL

1.3 Programming & Game Systems

This project was built entirely in C#, with a focus on clean system separation and animation-driven gameplay.

Core Combat System

Implemented three player abilities:

- Fire Breath, projectile-based ranged attack
- Tail Attack, melee hitbox driven by animation events
- Fly Attack, takeoff, airborne fire, land sequence

Combat uses:

- Animation Events to trigger hitboxes and effects
- Locking system to prevent attack spamming
- Cooldowns tied to UI feedback
- Clear attack commitment for better game feel

Enemy AI

- NavMesh-based movement
- Distance-based attack selection
- Cooldown-controlled ability usage
- Uses the same combat framework as the player

This kept combat consistent and modular.

Health and Game Flow

- Health system with UnityEvents
- Win, Lose, Draw states
- Combat disabling on death
- Scene-based restart and menu flow
- Clear end-state UI

The project evolved from functional to fully structured game flow.

Score System, Perfect Loop Focus

To reinforce replayability, I added a scoring system:

- Base win score
- Penalty for damage taken
- Time-based deductions
- Perfect bonus if the player wins without taking damage

If the run is perfect, the UI highlights it instead of just showing a number.

This directly reinforces the core loop, win clean, chase mastery.

Cross-Platform Architecture

Expanded the project beyond the original assessment:

- Separate PC and Mobile input handlers
- Platform-based UI activation
- Virtual joystick integration
- 60 FPS frame lock on mobile
- WebGL build support

The game now runs on:

- Windows
- Android
- Web

This required restructuring input architecture to separate:

- Movement controller
- Combat controller
- Platform-specific input layers

Visual and Audio Polish

- Shader Graph fireball effect
- Combat SFX, fire, tail, death
- Win, Lose, Draw audio feedback
- Background music management
- Device-based UI layout adjustments

Focus was on clarity and responsiveness over visual complexity.

1.4 Design Philosophy

Dragon Duel is intentionally small.

Instead of building a bloated feature list, I focused on:

- Tight combat feel
- Clear feedback
- Replayable core loop
- Cross-platform functionality
- Shippable scope

The goal was not to build a large game. It was to prove I could design, implement, polish, optimize, and ship a complete Unity vertical slice efficiently.

This project represents:

- Rapid learning in Unity
- Clean system structuring
- Cross-platform thinking
- Shipping discipline


Dragon Duel
FPS: 686
Play
Quit
Made By :
Siddharth Shankar
"Sidjoey"


Player Dragon
AI Dragon
Tail Attack
FPS: 780

Fig 1.1 - 1.4 : Screenshots from the game

2. The Biscuit Makes You Risk it: [2025 – Brackey's GameJam 2025.2]

Play Here: https://shorturl.at/gXc18

2.1 Introduction

My friend Prakarsh and I took part in **Brackey's Game Jam 2025.2**, where the theme was "*Risk It For The Biscuit*" . We wanted to experiment with something chaotic, fun, and different — so we came up with the idea of a **first-person action survival game** where every match is altered by a random risk.

The concept was simple but challenging: the player presses a button, and from a pool of curses, one is assigned at random. Each curse changes the way the game plays out, making every attempt unique and forcing the player to adapt. After that, the player has to survive **five escalating waves of enemies**, all while dealing with their handicap.

We divided tasks based on our strengths. I focused on the **core gameplay systems, AI enemy wave design, and the curse logic**, while Prakarsh worked on **3D models, animations, and world assets**. Even under the tight two-day deadline, we were able to deliver a complete game loop.

2.2 Tools

The game was built in **Unreal Engine 5.4.4**, with Blender used for asset adjustments and animation tweaks.

2.2.1 Programming / Game Logic (My Role)

The game logic was fully implemented using **Blueprints** in Unreal Engine, Unreal's visual scripting system built on C++. My knowledge of C++ helped me structure the gameplay logic cleanly within Blueprints, making the system both efficient and modular.

Key implementations included:

- A **randomized curse system** using enumerators, assigning one of five risks dynamically to the player:
    1. **Inverted Controls** – flips movement and camera input.
    2. **Low Gravity** – reduces player gravity, making movement floaty and harder to control.
    3. **Slippery Movement** – decreases friction, adding momentum drift to movement.
    4. **Slow Fire Rate** – lowers the weapon's firing speed, forcing careful aim.
    5. **Reduced Health** – cuts player health in half for the whole run.
- **Enemy AI** built with wave progression and scaling difficulty.

- **FPS combat mechanics** with responsive aiming and movement.
- A clean design philosophy that avoided unnecessary HUD clutter, relying instead on clear gameplay feedback.

This project gave me practical experience with Unreal's AI tools and enumerators, and pushed me to prototype fast while keeping systems flexible.

2.2.2 3D Work / Assets (Prakarsh's Role)

Prakarsh created and optimized the **enemy models, weapons, and environmental props** to be game-ready. He made the animations for the player and the enemy character, refining them in Blender for smoother gameplay integration. His contribution gave the game a cohesive visual identity despite the jam constraints.

2.2.3 Map Design (Joint Work)

The map design was a shared responsibility. We used base assets from Fab (formerly Unreal Marketplace) and adjusted them for our scenario. Prakarsh focused on layout and lighting, while I worked on **optimization**, leveraging my experience with Unreal.

The result was a compact arena that showcased both the **chaotic survival gameplay** and the **random risk mechanic** effectively.

Fig 2.1 : Game Starts by pressing a button to choose your curse

Fig 2.2 : One of the 5 curses that are in the game

Fig 2.3 : our Cannibal enemy chasing the player

3.Untitled Puzzle Game [The Riddle Facility]

3.1 Introduction

This project is a mystery puzzle game created as part of my ongoing design experiments. The core idea is to combine **classic riddle-solving** with **environmental puzzles** in a strange test-chamber-like environment, inspired by *Portal 2*. Unlike typical puzzle games, riddles here directly **set up the puzzles**, blurring the line between language, logic, and interaction.

The goal is to create an atmosphere of mystery and unease, where every puzzle feels like a piece of a larger system testing the player's mind.

Also since this is a First Person Game, I have made all of the animations and movement for the first person characters myself and that work is here.

3.2 Gameplay Loop

1. Explore the environment and discover **Notes** containing riddles.
2. Use riddles to solve **direction, logic, or environmental puzzles**.
3. Unlock new areas and face **enemy encounters** that introduce danger.
4. Survive and progress to uncover the deeper mystery of the facility.

3.3 Environment Design

- **Ground Floor Layout**:
    - **Tree Room (Direction Puzzle)** – Centerpiece puzzle where the player aligns symbols at N, S, E, W based on a riddle.
    - **Notes** – Scattered lore and riddle clues (Note 1: Medium puzzle, Note 2: Key puzzle, Note 3: Medium puzzle 2).
    - **AI Room Puzzle ("Lunch")** – A thematic room where the AI introduces itself or sets a challenge.
    - **Enemy Encounter** – The player finds a hostile AI-driven enemy, breaking the pure puzzle pacing with tension.
    - **Stair Access to First Floor** – Reward for solving ground floor puzzles.

3.4 Puzzle Examples

- **Riddles**
    - *Long Riddle*: Philosophical, about the mind as a prison.
    - *Short Riddle*: Classic object riddle (e.g., "river").
- **Puzzle Mechanics**

- **Direction Puzzle** (Tree Room) – Aligning nature's elements with N/S/E/W.
- **Order Puzzle** – Step on elemental buttons in the right sequence.
- **Laser Reflection Puzzle** – Use mirrors to direct a beam.

3.5 Smart AI Enemies

To add tension and variety, I am implementing **AI systems inspired by Resident Evil games**.

- These enemies are **not simple chasers** – they learn and react.
- **Behaviors include**:
  - Patrolling until disturbed.
  - Hearing noises and investigating.
  - Searching intelligently instead of following fixed paths.
  - Forcing the player to **strategize stealth or timing** between puzzle-solving.
- This makes every encounter unpredictable, preventing the player from feeling safe even in puzzle chambers.

3.6 Vision

The final game will merge **the elegance of riddles** with **the intensity of survival encounters**, creating a unique rhythm of thinking and tension. Players will feel like they are part of an elaborate experiment – a world where language, puzzles, and AI are all weapons of control.

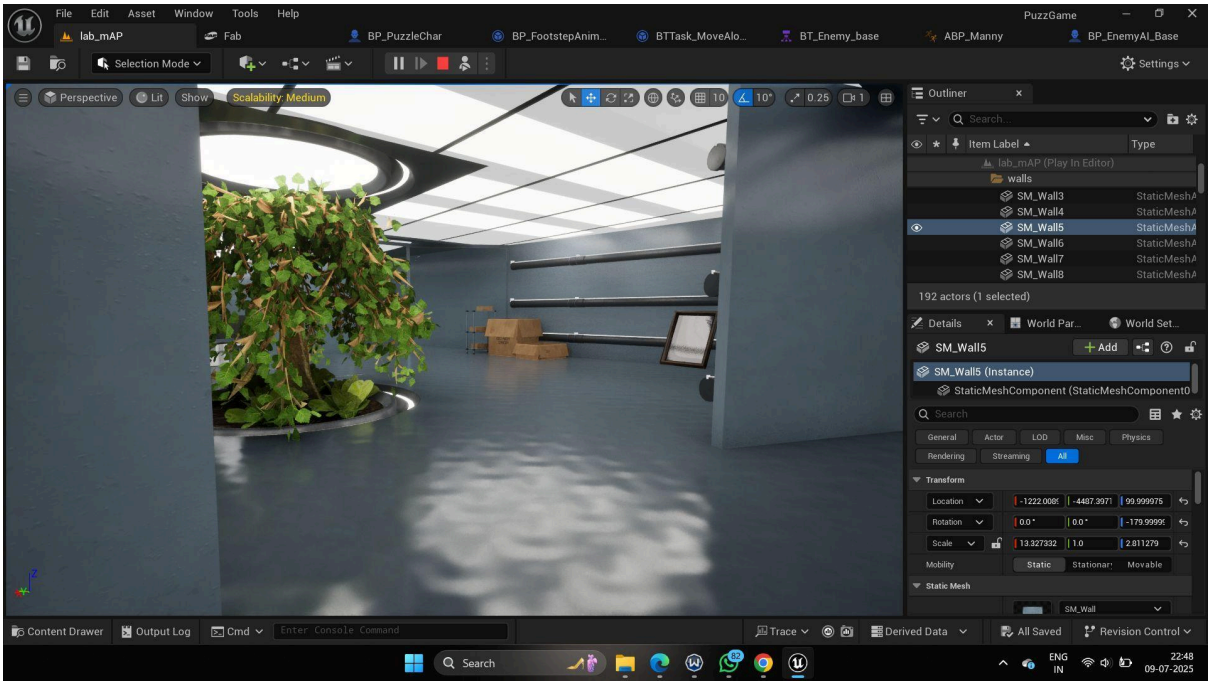


Fig 3.1 : The Main tree room (Central Location)

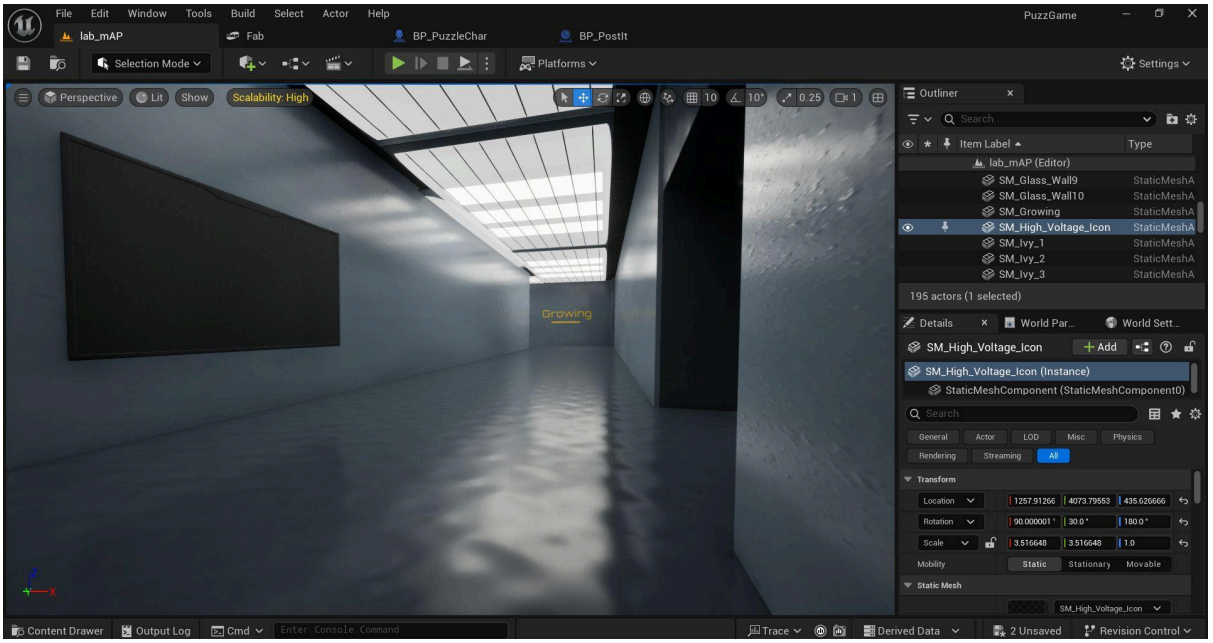


Fig 3.2 : The Long Hallway

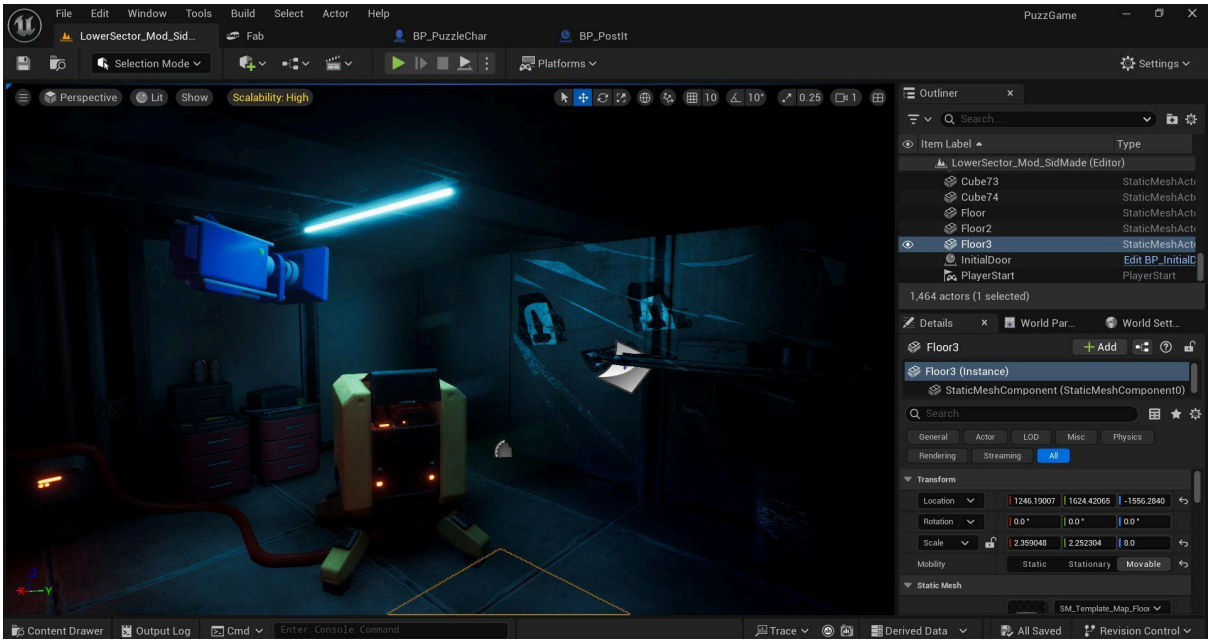


Fig 3.3 : First Puzzle / Map Entrance

4.RetroPunk [a static rail shooter inspired by House Of The Dead 2 and virtua cop] :

4.1 Introduction :

My friend Prakarsh and I took part in a game jam organized by our college and the theme was reverse. We came up with the idea of a static rail shooter because we wanted to challenge ourselves to make something we never had before.

We got the map assets from Fab and get to work, we designed the map and then we started working on our strengths, he started the work on the gun and character models and I started on building the gameplay. The game jam came and went but we are continuing development on this game because we like working with each other and we think we have a fresh perspective on the static rail shooter genre.

The game centres around quick decision making. We have 2 different bullet types and correspondingly 2 different enemy types ; Fire and Ice. The player will have to discern the enemy type and then shoot them with the opposing bullet type.

We plan to have other enemy types as well inside of the enemy umbrella but for now all we have is a bruiser-brawler type of mutant enemy.

4.2 Tools:

The game is made using Unreal Engine 5.4.4 and Blender 4.

4.2.1 Programming / Game Logic:

The game Logic [my part] was done completely in blueprints which is Unreal Engine's own visual scripting language that was built on C++. I know how to program in C++ but Unreal has a lot more resources both online and in its engine documentation that make Blueprints a much easier option to use, while still being powerful C++ object oriented programming.

The logic of the program is that same as if you were to code it and knowing how to use C++ is what allowed me to make the game within 36 hours.

Inheritance is a very fundamental and important part of game design as I have come to learn and I have used it extensively throughout this game to save time and resources and also because it makes everyone's life that much easier.

The enemy ai has one parent class which then has the child classes of the fire and ice enemy respectively, this allows me to have the base functionality of an enemy which is to attack the player and also allows for individual enemy customization.

The gun switches its inner material depending on what bullet type is selected which makes it clear to the player that they're about to fire the bullet type related to the colour without unnecessary HUD elements allowing for maximum immersion.

I used Unreal Engine tools such as the enumerators and AI tools that I hadn't used before and if I had, I hadn't had hands on experience with them and now I can confidently say that I know more about the engine and how I like to make games than I did previously and that is a win.

4.2.2 Character Models / Gun Model and Other 3d Work:

My friend Prakarsh handled the 3D aspect of the game. He made the gun model taking inspiration from our own ideas and the R99 from Apex Legends. He also found and adjusted the enemy character to our game, making sure the models are optimized and game ready.

He prepared the animations from Mixamo and then tweaked them to fit our use case on Blender and made some of the assets that make the map feel more alive.

4.2.3 Map Design:

The Map design is something both of us had little experience with but I feels as though we have created the perfect map for our scenario.

We got the base assets from Fab (formerly Unreal Marketplace) and got to work tweaking them.
Prak worked on the base layout and the polycounts and other optimization that had to be done while I was working on the lighting. My professional experience with colour grading software such as Davinci Resolve helped me make the lights feel ominous but inviting.

We had fun and the map looks great because of it, this isn't the updated map because that is on Prak's machine as we had to separate for the night because of hostel rules and then we had midterms , but this map reflects our skills in the area of world creation.

Screenshots:

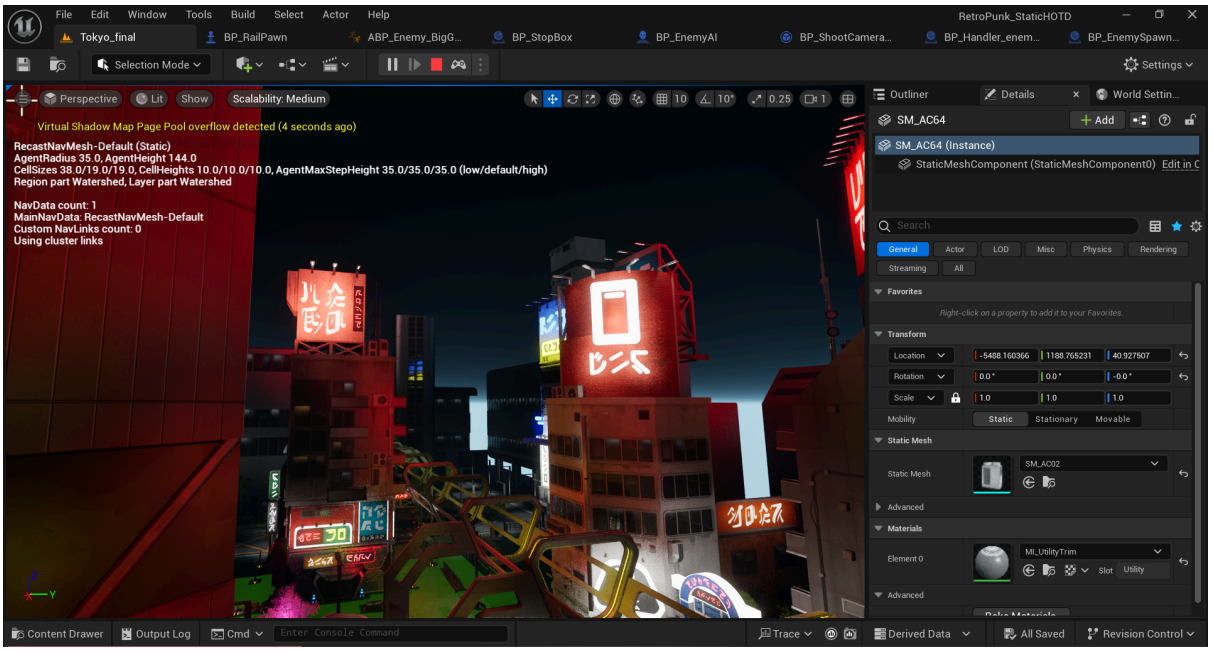


Fig 4.1 In Engine View Of The map

Fig 4.2 In Game View

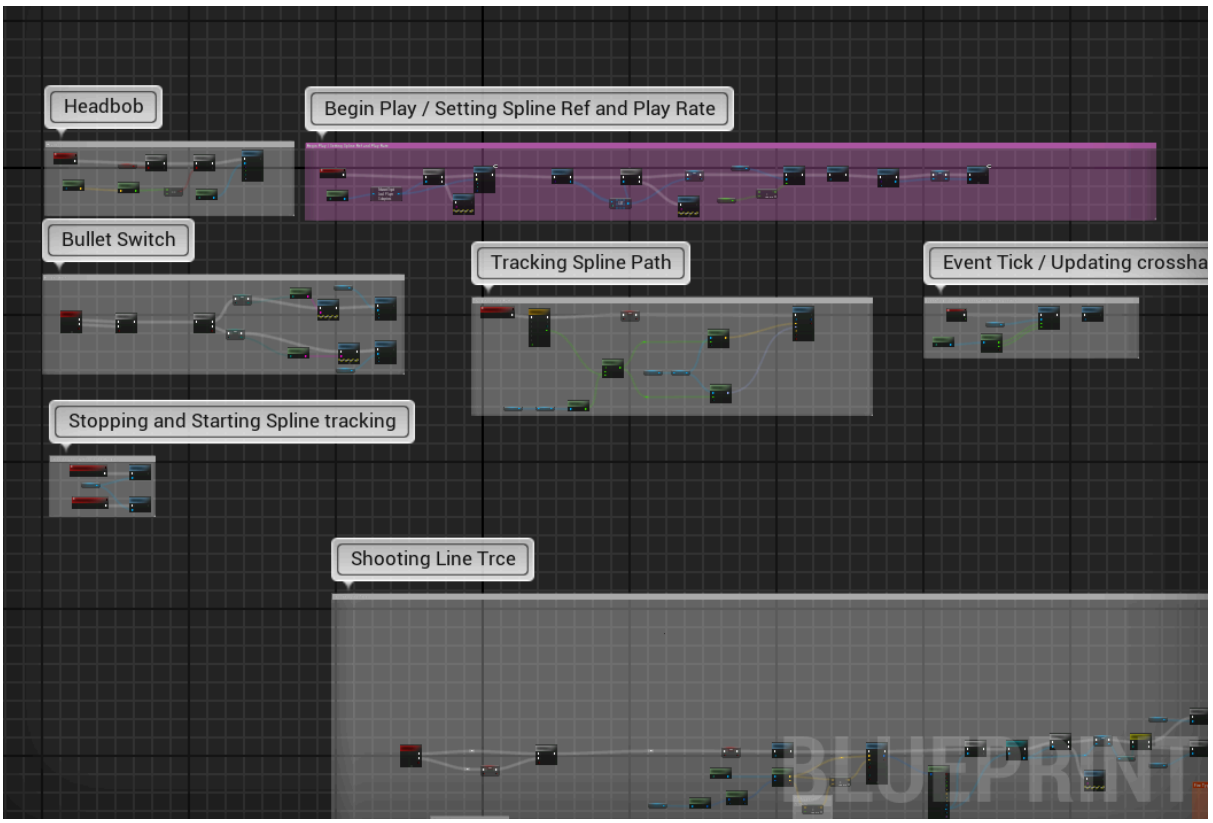


Fig 4.3 Player Character Logic

Fig 4.4 Enemy Ai Model in Engine

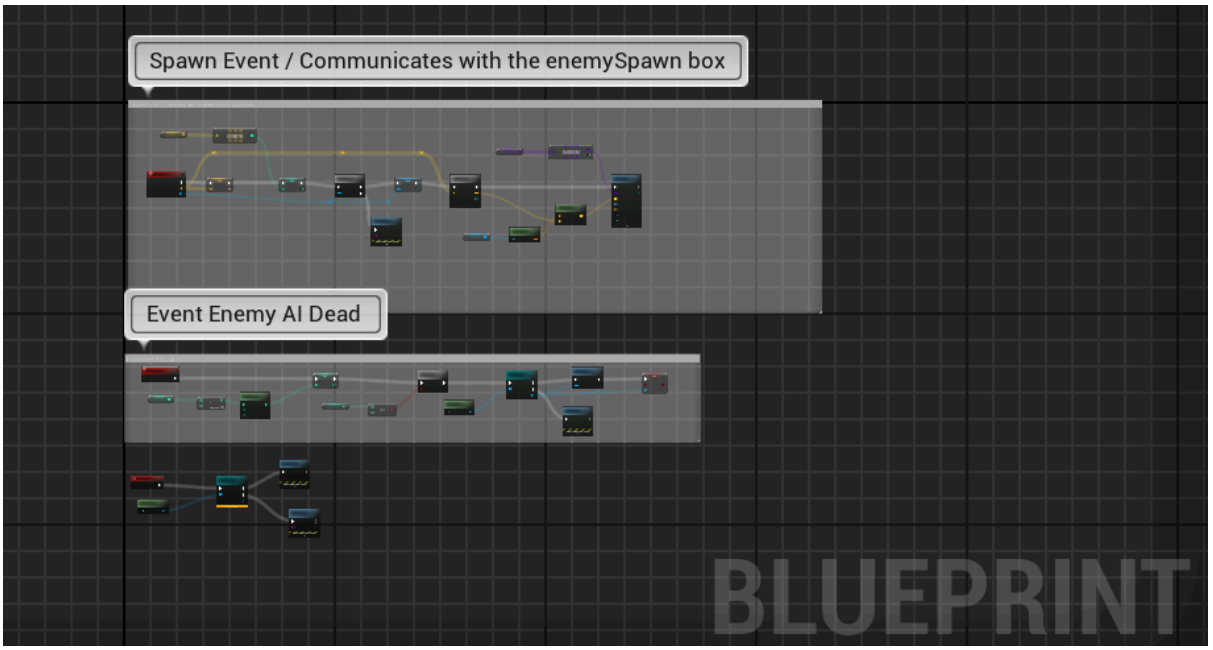


Fig 4.5 enemy handler (controls the game flow)

5. Top Down Zombie Shooter: [2023 – made as a college project]

5.1 Introduction:

This project was a solo venture I undertook as part of my college coursework. The Top Down Zombie Shooter was my first attempt at creating a complete game experience on my own, pushing me to handle all aspects of game development independently.

I wanted to create something that would challenge me technically while delivering a fun, action-packed experience. The classic zombie survival theme gave me a perfect playground to experiment with enemy AI, player movement mechanics, and combat systems.

The game centres around survival against increasingly difficult waves of zombies. The player must manage ammunition, health, and positioning while fighting off hordes of the undead. I implemented a progression system where zombies become stronger and more numerous as the player advances, creating an escalating challenge.

This project taught me valuable lessons about scope management and the importance of focusing on core gameplay elements before adding additional features.

5.2 Tools:

The game was developed using Unreal Engine 5 and Blender for minor asset modifications.

5.2.1 Programming / Game Logic:

Working solo on this project gave me complete ownership of the game logic, which I implemented using Unreal Engine's Blueprint system. This project was instrumental in developing my understanding of player movement systems and basic enemy AI behavior.

For player movement, I implemented a responsive top-down control scheme that allowed for fluid movement in all directions while aiming

independently with the mouse. This dual-input system created engaging gameplay that required players to think about positioning while fighting off enemies.

The zombie AI was my first serious attempt at creating enemy behaviour systems. I developed a simple but effective pathfinding system that allowed zombies to navigate around obstacles while pursuing the player. The zombies would detect the player through line-of-sight and sound triggers, creating tension as players needed to be mindful of the noise they generate creating an escalating challenge.

This project taught me valuable lessons about scope management and the importance of focusing on core gameplay elements before adding additional features.

I implemented a wave-based spawning system that gradually increased difficulty by spawning more zombies with enhanced attributes. This system taught me about game pacing and difficulty curves, ensuring players would face appropriate challenges as they progressed.

5.2.2 Character Models / Assets:

I utilized character and weapon assets from an online course I had taken. While I didn't create these from scratch, I gained valuable experience modifying and optimizing them using Blender to fit my game's aesthetic and technical requirements.

I made adjustments to textures, rigs, and animations to ensure they worked seamlessly within my game's systems. This experience gave me a practical understanding of asset integration and optimization in game development.

5.2.3 Map Design:

The game environment was built using a premade asset pack called Assetville that I acquired from Fab (formerly Unreal Marketplace). This provided a solid foundation for creating a believable urban environment for the zombie apocalypse scenario.

I customized the layout to create interesting gameplay spaces with strategic chokepoints, open areas, and various forms of cover. I paid particular attention to creating a balanced play space that would give players options for different combat strategies while still maintaining tension.

The lighting was designed to create an appropriate atmosphere for a zombie game, with dark corners, flickering lights, and an overall sense

of abandonment. I used post-processing effects to enhance the mood and create a more immersive experience.

While using premade assets, I learned valuable lessons about level design principles, pacing, and creating environments that support the core gameplay mechanics.

Screenshots:

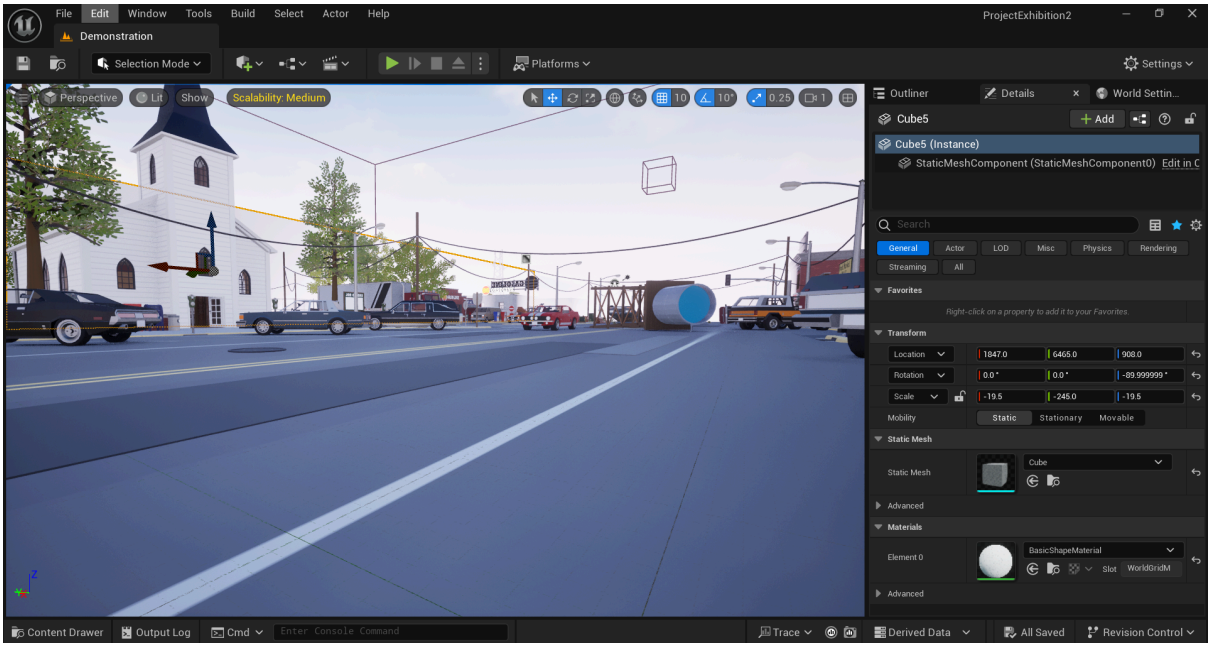


Fig 5.1 In engine map view

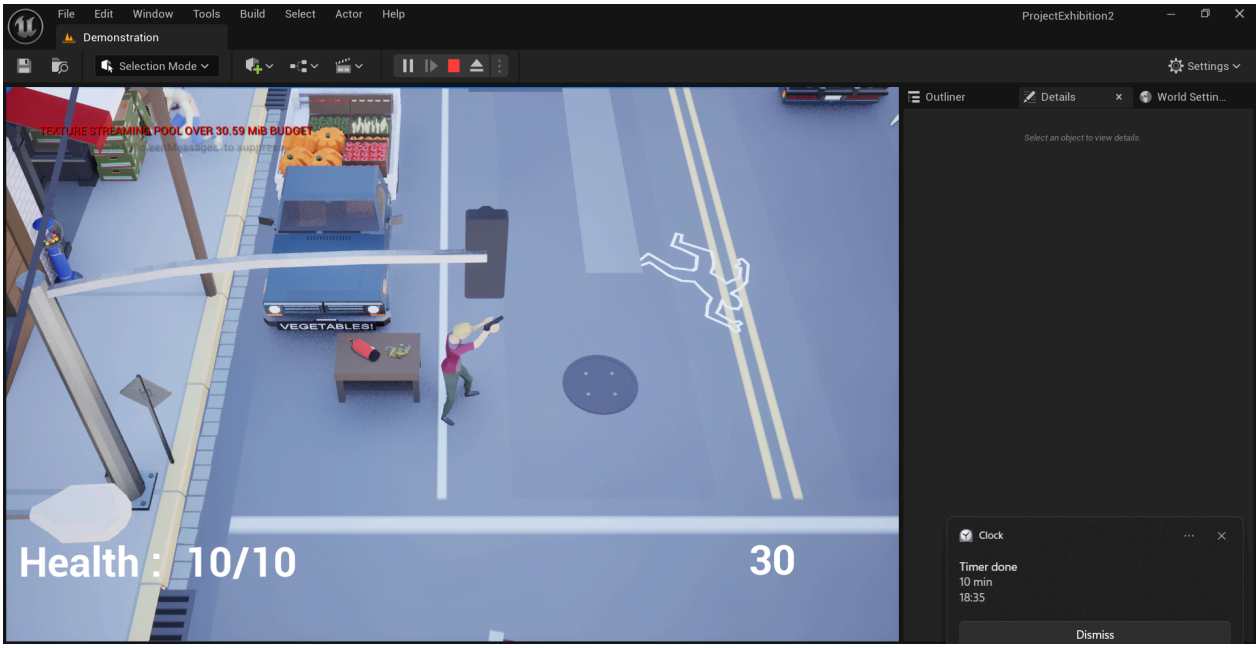


Fig 5.2 In game view of the map and player character

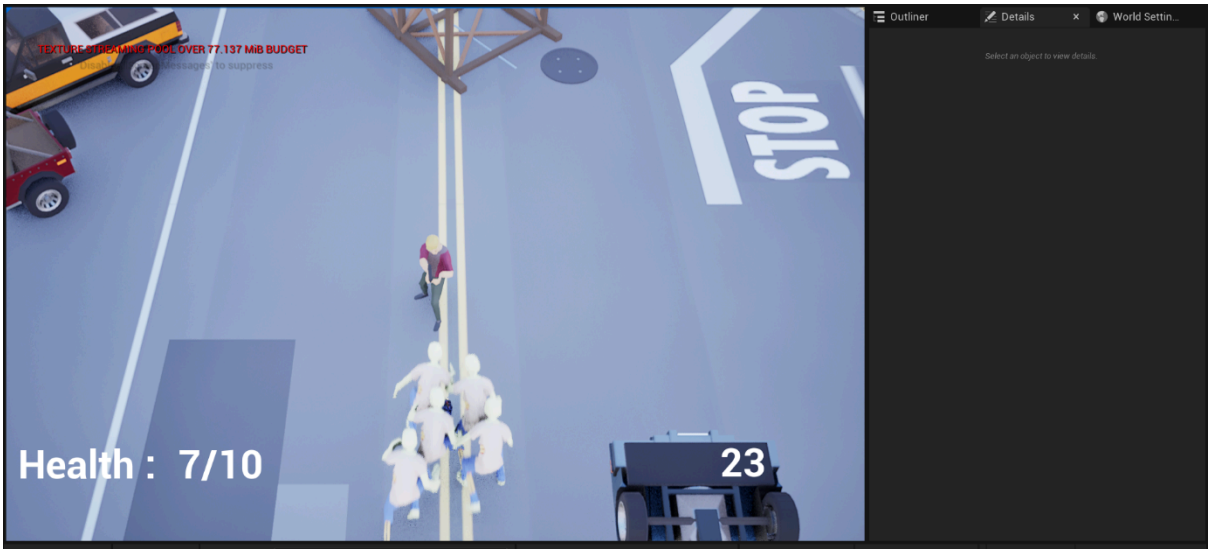


Fig 5.3 In game view of player character being chased by zombies

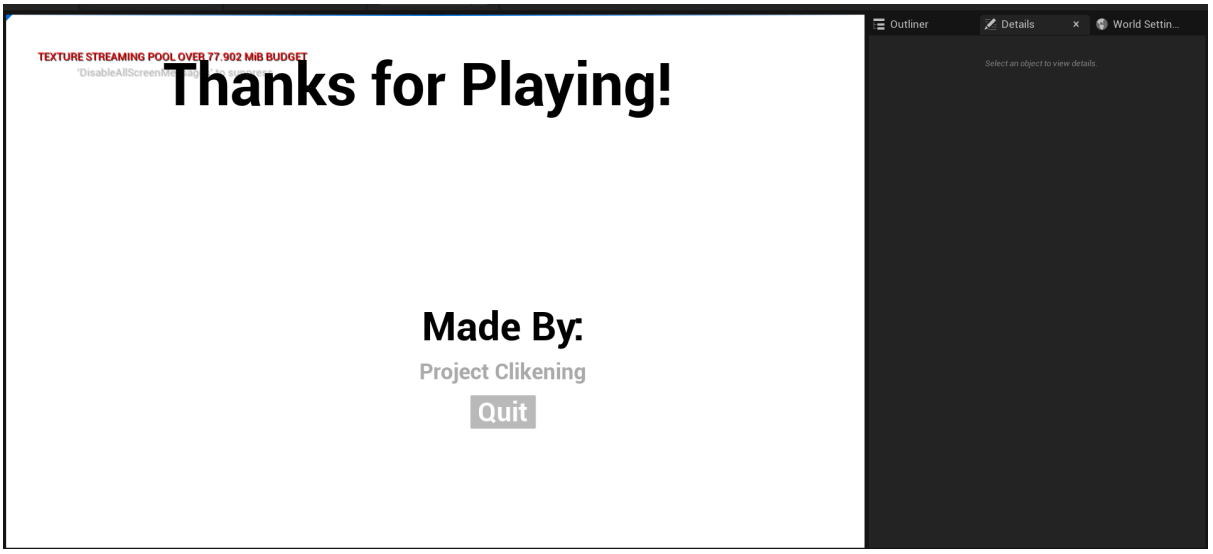

TEXTURE STREAMING POOL OVER 77.902 MiB BUDGET
Thanks for Playing!
Made By:
Project Clikening
Quit
Outliner
Details
World Settin...
Select an object to view details.

Fig 5.4 – 5.5 UI

6. Confronting my fears and creating a game environment:

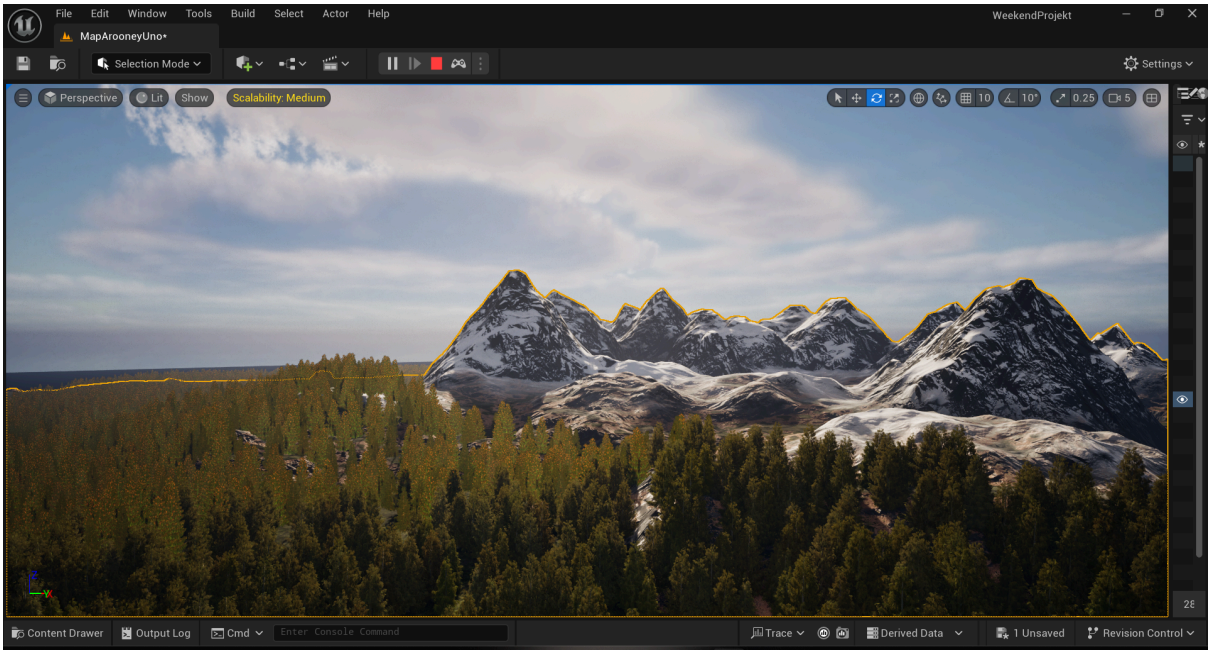
My first project of the year represented a significant personal challenge as I confronted my long-standing fears of terrain creation and map design. Having previously relied on pre-made assets and environments, I decided to push my creative boundaries by building a landscape from scratch.

I decided to make a small island in Unreal Engine 5.4.4, populating it with trees and other natural elements while focusing on optimization. For this project, I utilized premade tree assets from Fab (formerly Unreal Marketplace) which allowed me to focus on placement and composition rather than 3D modelling.

The terrain development process involved configuring landscape materials, also sourced from Fab, to create realistic ground textures. I learned how to set up height-based material blending, which automatically applied snow to higher elevations and transitioned to grass, dirt, and rock at lower levels. This technique created natural-looking transitions without requiring manual texture painting for every area.

The process was uncomfortable but ultimately rewarding, as I discovered that creating convincing terrain was less about technical perfection and more about understanding natural patterns and player psychology. This experience expanded my capabilities as a developer and gave me confidence to tackle other aspects of game creation I had been reluctant to explore.

Screenshots:



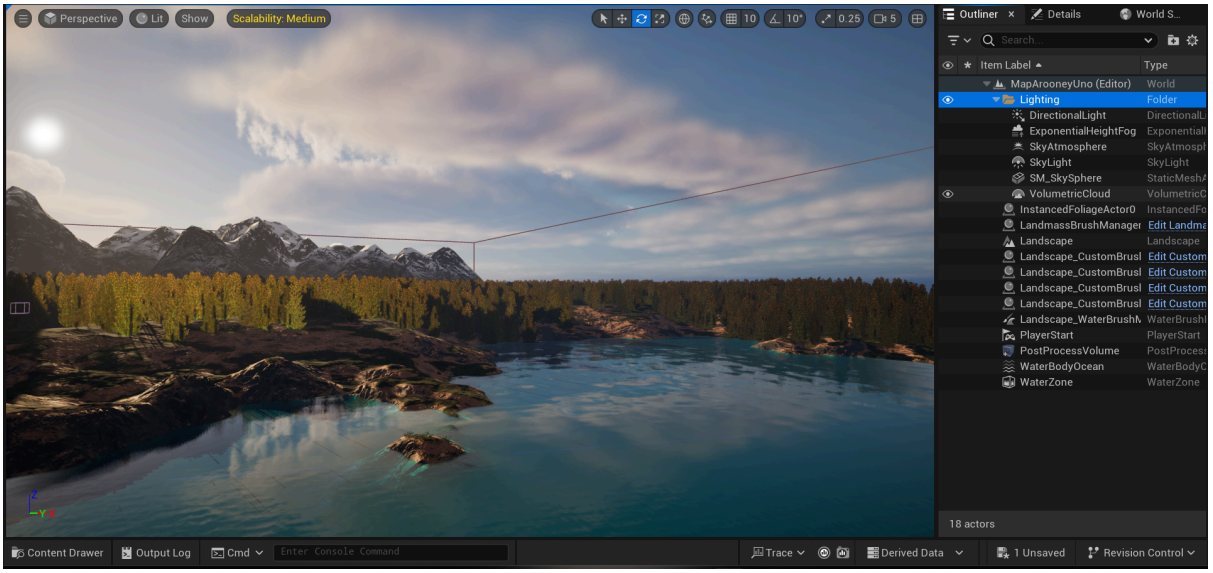


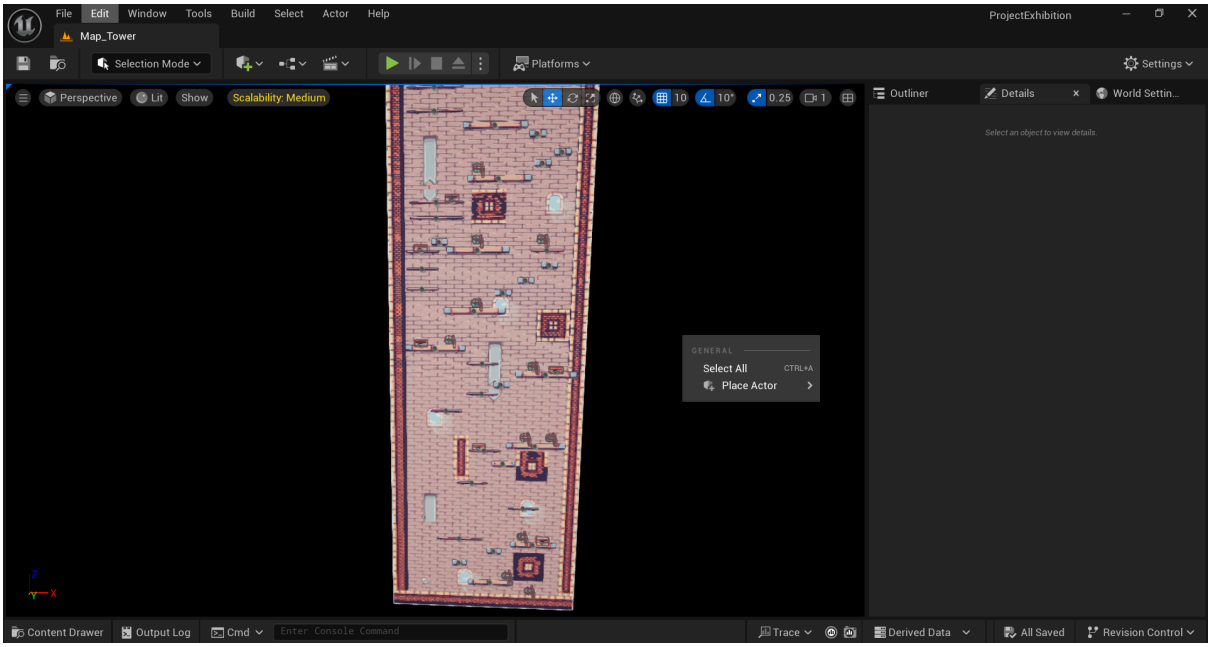
Fig 6.1 – 6.3 Different angles and Lighting on my island

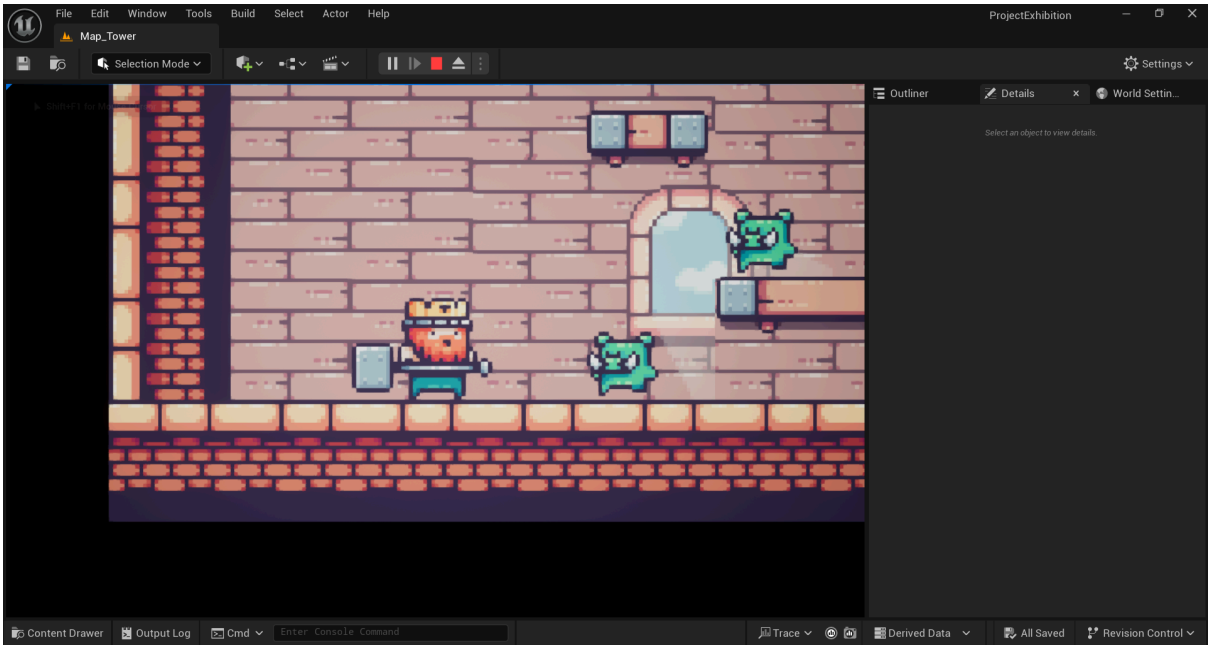
7. 2D Games that I made when starting out

My game development journey began with an online course By Cobra Code focused on Unreal Engine basics. This course taught me fundamental concepts specifically for 2D game creation, covering essential mechanics like player movement, collision detection, and game loops.

The hands-on approach allowed me to immediately apply new techniques to simple projects, building my confidence to tackle more complex challenges independently. This foundation gave me both the technical skills and problem-solving mindset needed to eventually transition into 3D development and pursue my own creative projects.

Screenshots:





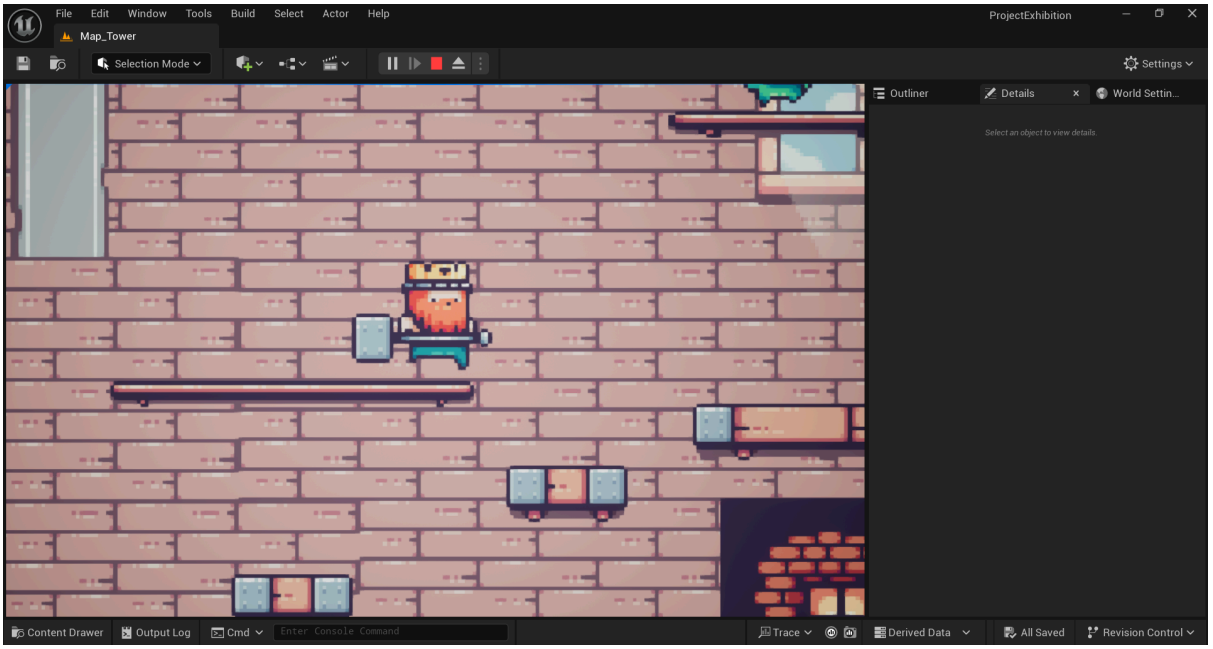
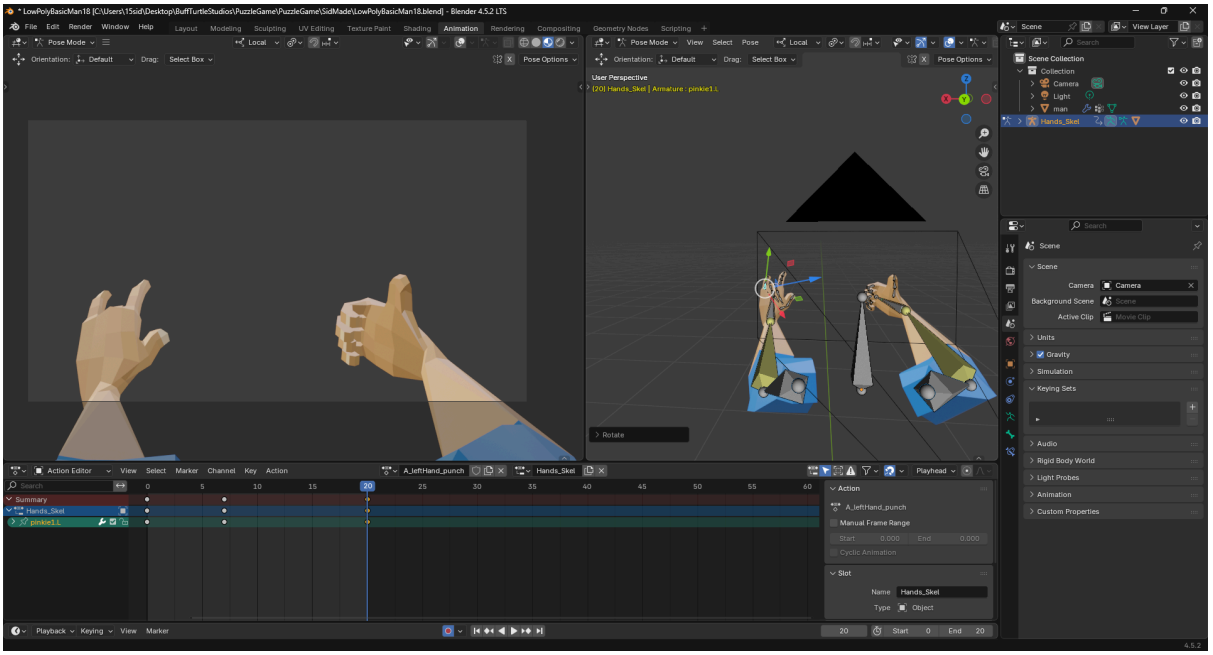


Fig 7.1 – 7.3 in-engine and in-game view of the game

Blender Work:

As I stated previously I am comfortable and familiar with blender as well as I know it is important to be familiar with all of your tools. That is what makes a good builder.

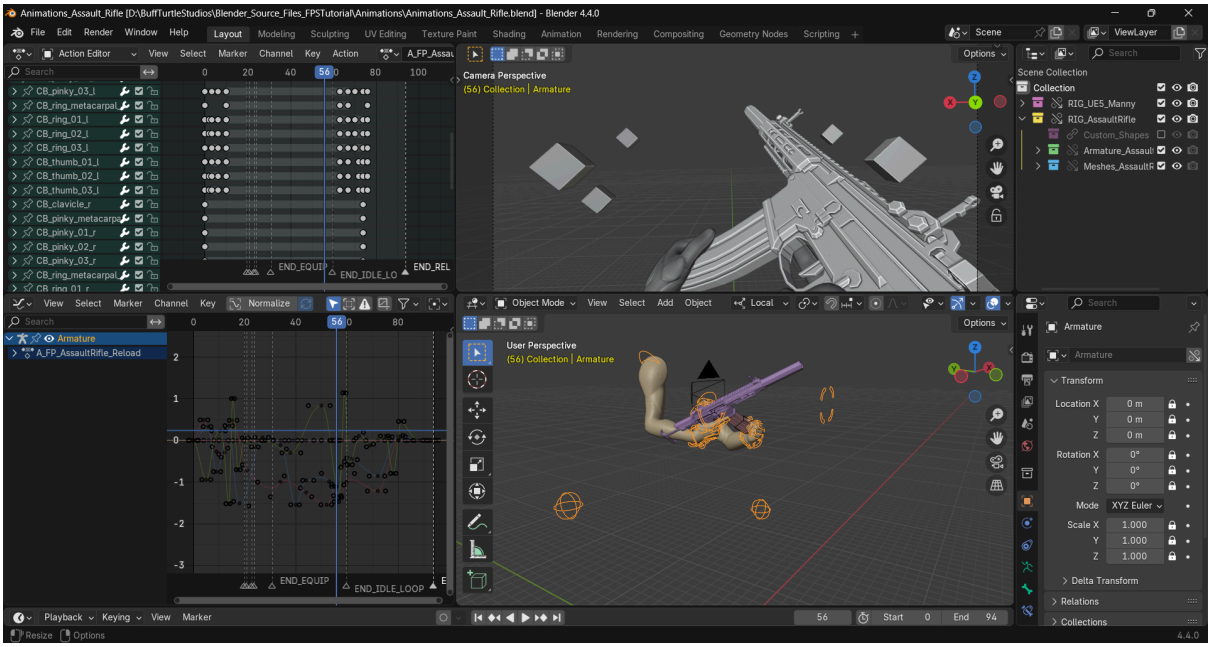
Here are a few screenshots and gifs of my work:



Hands I made for my puzzle game with all the animations

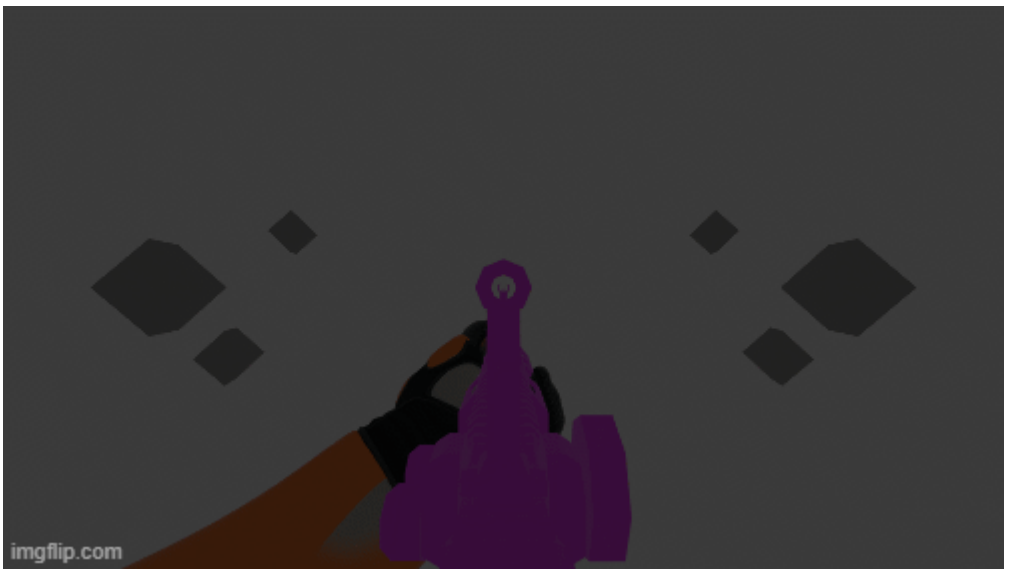
A list of most of the animations

This is how they look in blender



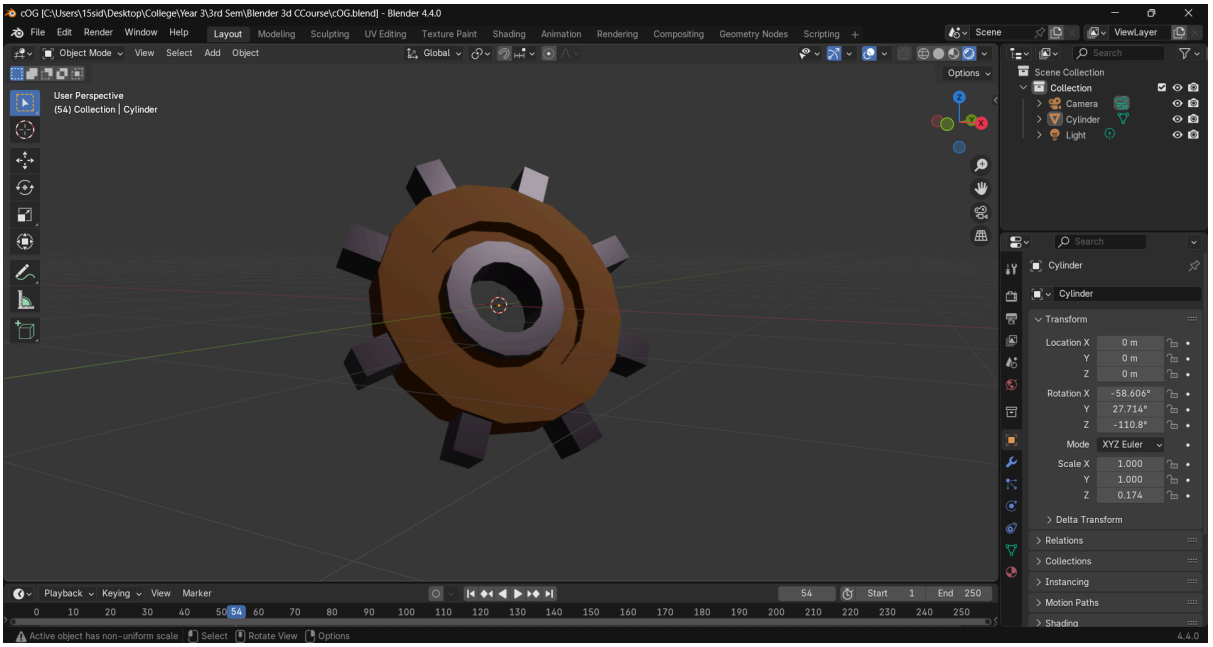
Layout of my Blender Fps Animation Creation File

The Reload animation that I made.

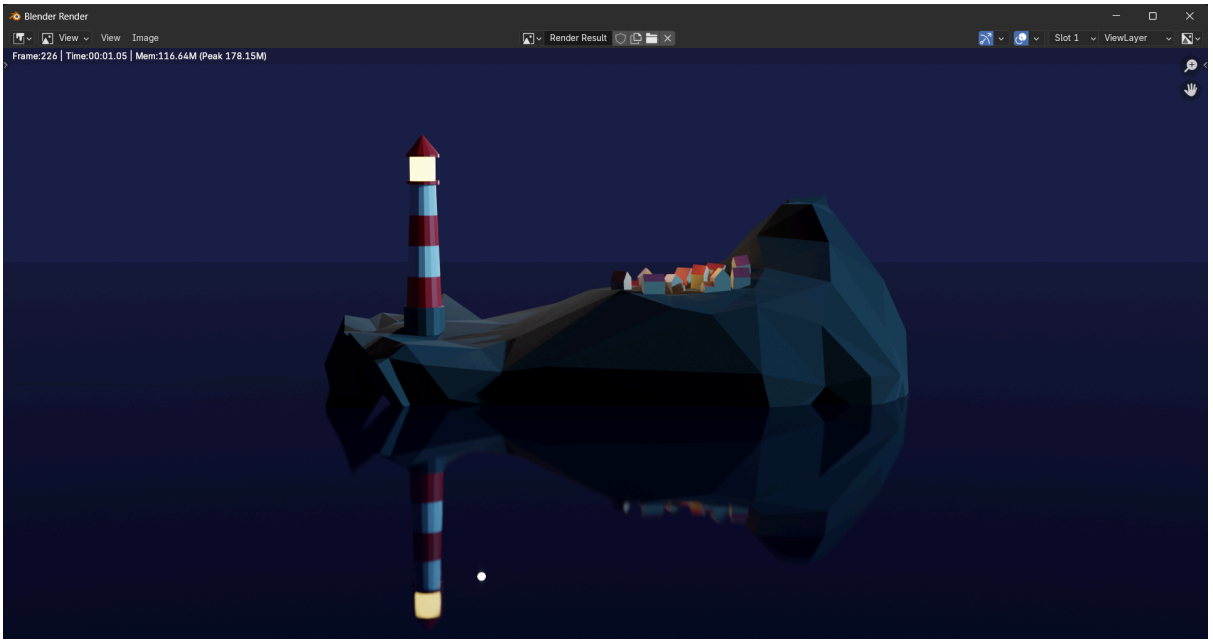


The firing animation I made (looks better in blender and game)

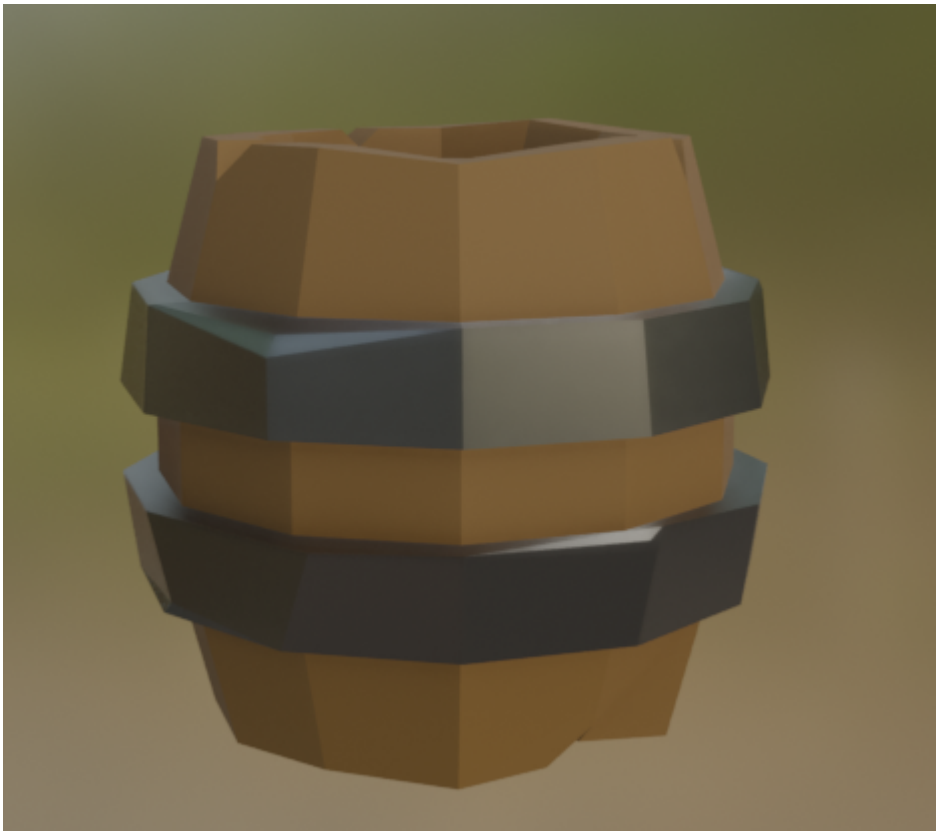
Low Poly Buff Character for my next project



Blender class assignment to create a cog in under 2 mins



Lonesome island : made this while learning blender from a course



Low Poly Environmental Objects Inspired by Sea Of Thieves