[2024] - Lost-Tombs

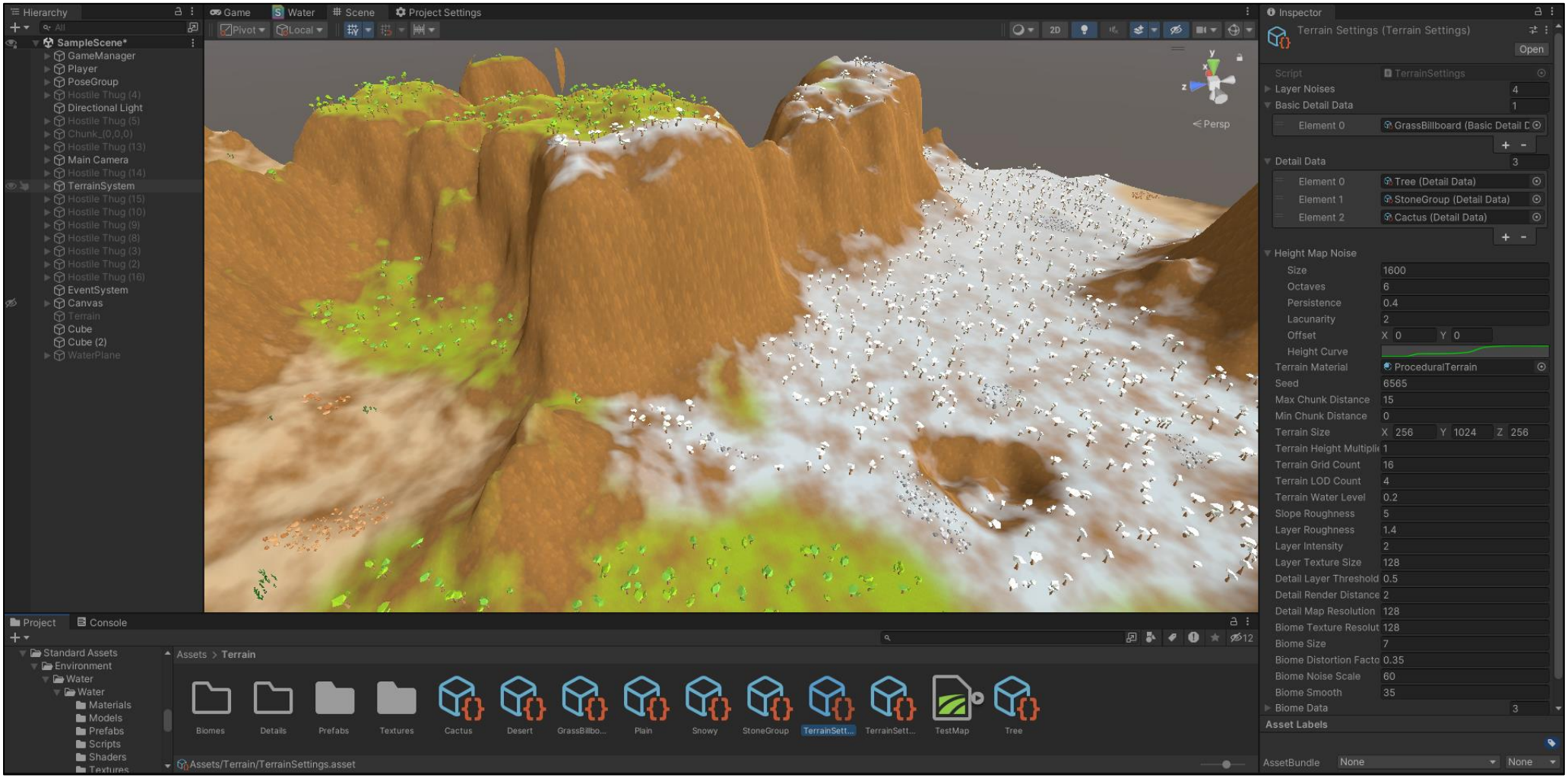
Lost-Tombs is a top-down RPG where you battle skeletons and undead within abandoned castles and crypts.

I created this game with mechanics inspired by Diablo 4, where players explore dungeons and collect ancient relics while battling skeleton enemies. The game follows a top-down perspective, offering a dynamic experience as players navigate through perilous environments, uncovering secrets and engaging in combat. The core gameplay revolves around gathering relics from the past while surviving encounters in hostile dungeons.

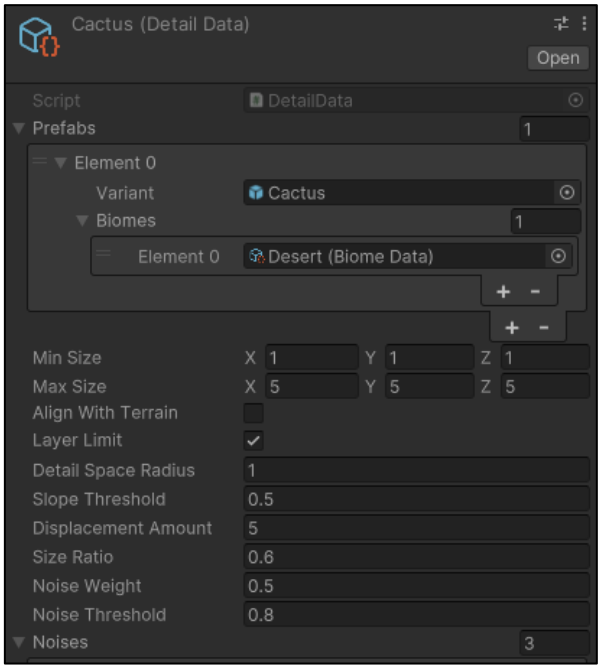
One of the standout features I developed is an optimized X-ray effect that enhances the visual clarity of the environment, allowing players to see through walls and objects without compromising performance. This was crucial in maintaining smooth gameplay for the top-down view. The enemy AI was custom-built, offering a challenging and adaptive combat system that keeps players engaged.

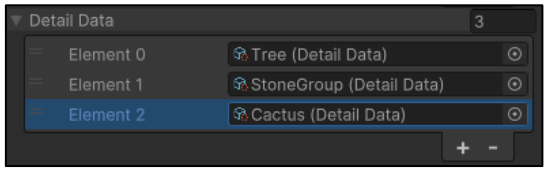
[2022] - Procedural Terrain with Biome

In this project, I developed a framework in Unity that allows for procedural terrain generation with various biome features, and I created a custom editor script to give designers fine control over how the terrain is generated. This system was primarily designed to create diverse environments in the RPG game I am working on, but it is flexible enough to be used in other projects.

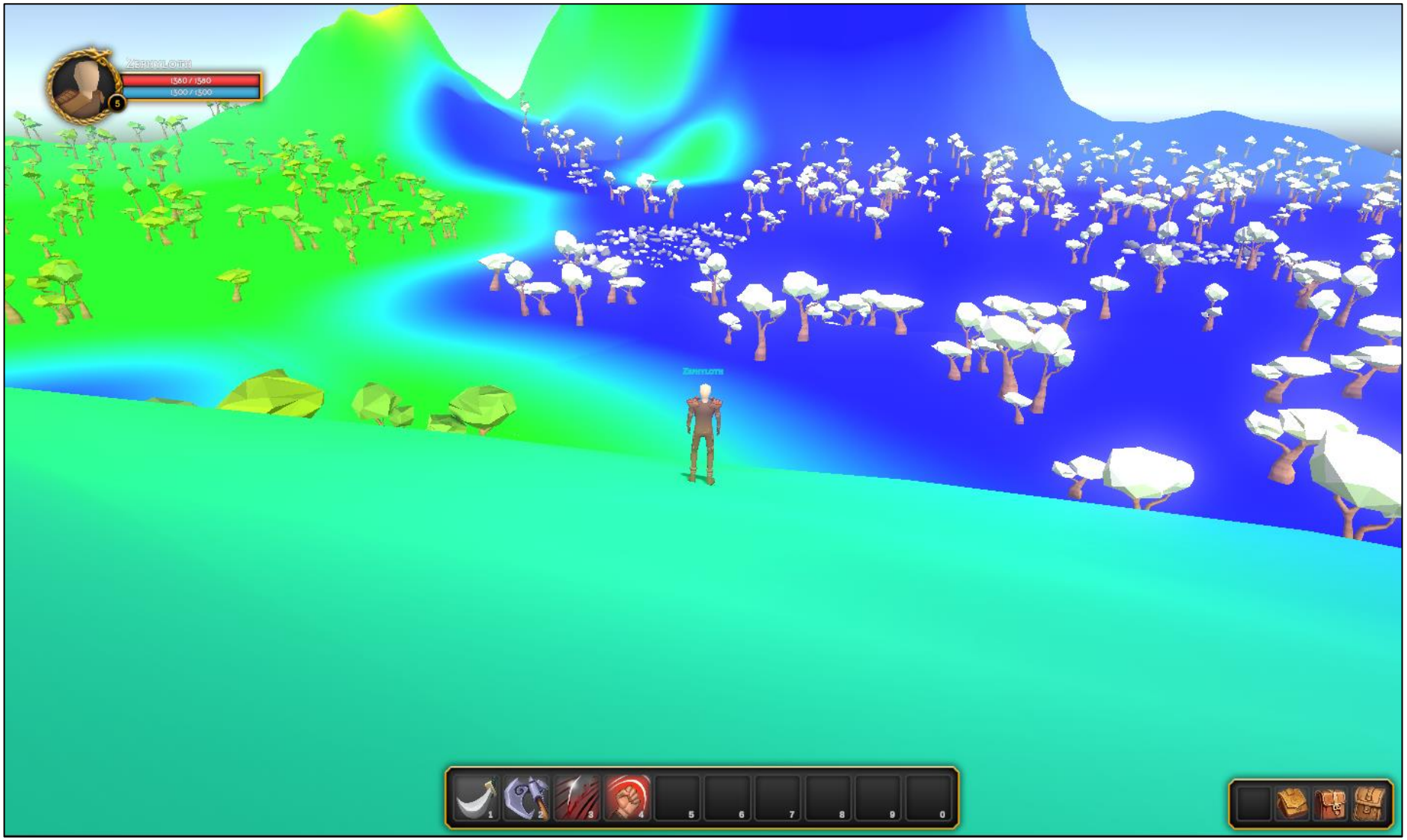


To enhance the environmental detail, I used ScriptableObjects to define settings for different biomes. For instance, I could easily add cactus plants to desert regions or adjust the density of vegetation in forest areas. Each detail, such as size, frequency, and its associated biome, is configurable through these settings. I also integrated native C commands for efficiently placing objects on the terrain.





To handle terrain textures dynamically and ensure high performance, I implemented compute shaders to procedurally generate the terrain's biome textures. These textures are basically distorted voronoi textures where each color channel is mapped to a specific biome texture. Consequently, this provided precise control over biome transitions and created highly optimized workflows for large-scale terrains.

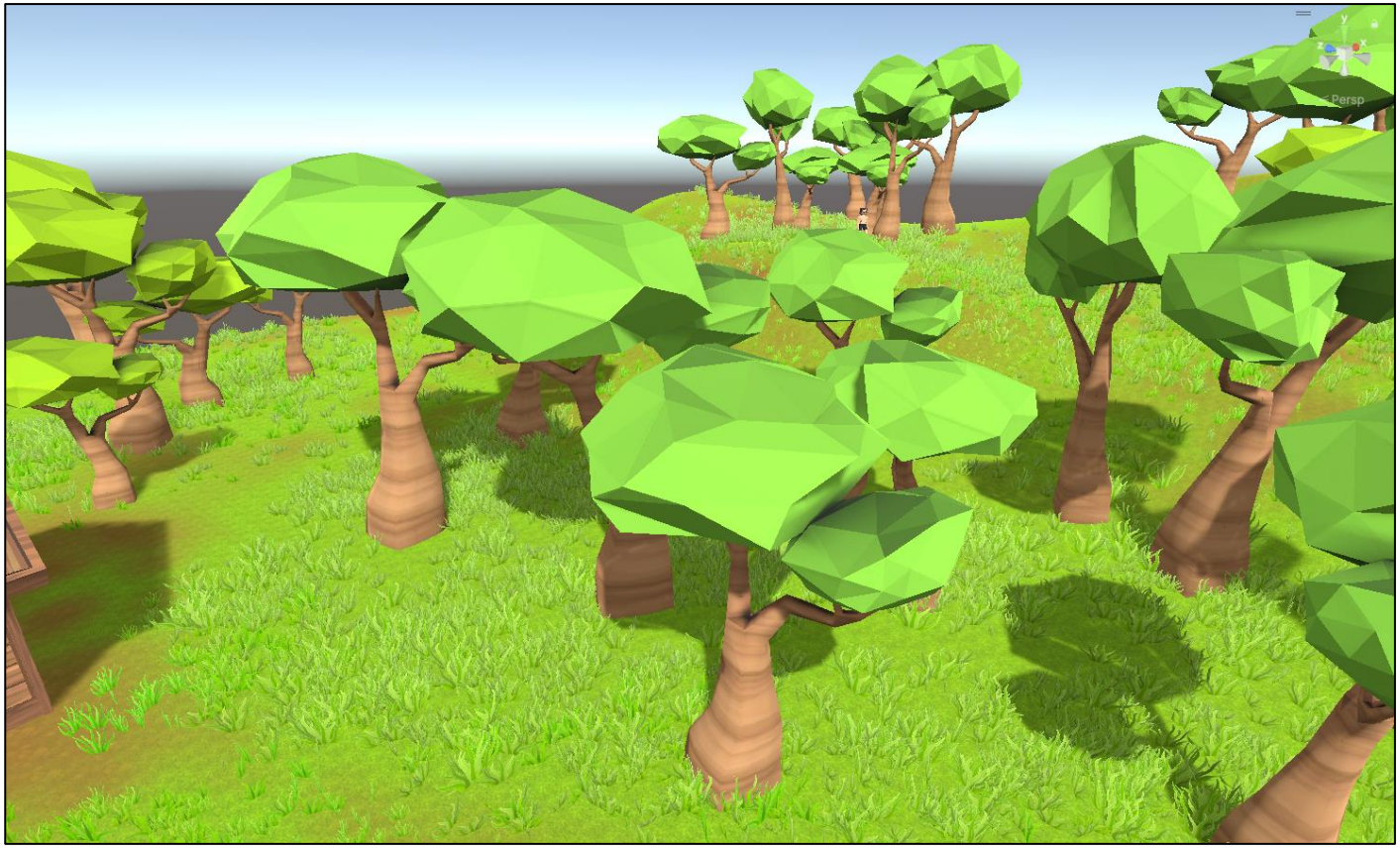


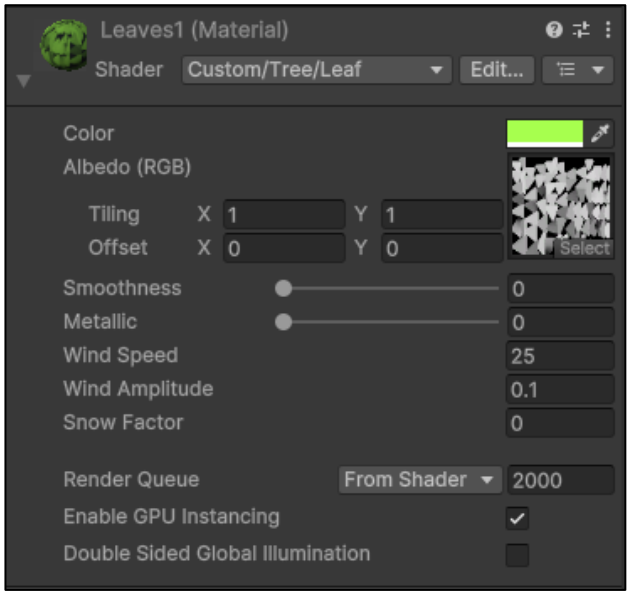
[2021] – Slethron RPG Game

I am currently working on an RPG-based game using the Unity engine, which integrates all the core elements of the RPG genre, including a skill system. This project is built entirely on a story that I created, and all the game's code and models are designed and implemented by me. The game features a fully functional character customization system where players can change their clothing, as well as an inventory system that is connected to the player's equipment and items. Additionally, I have designed a skill system allowing players to learn and upgrade abilities as they progress through the game. The NPCs are controlled using a custom-built AI system, and they interact with the player in various ways, such as chasing the player.

In terms of world design, I personally modeled the environments, including houses, characters, and other objects, creating a rich and detailed world. I also developed the minimap system, which helps players navigate through the game's world. Moreover, I added comprehensive loot system that is customizable for every NPC in the game.

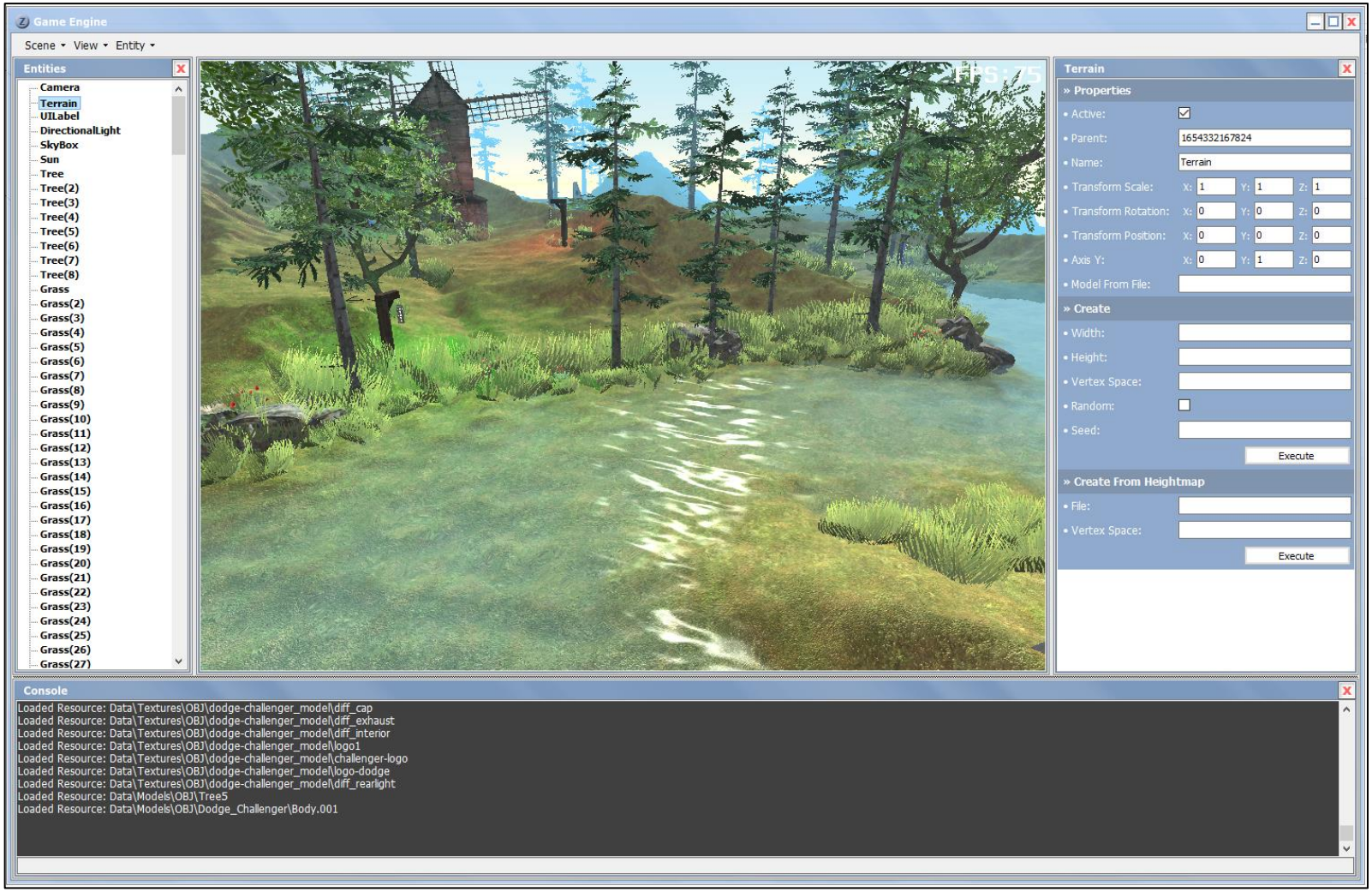
One of the unique technical aspects of this project is the shader I wrote for the trees, allowing them to sway realistically in the wind, adding a dynamic touch to the environment.



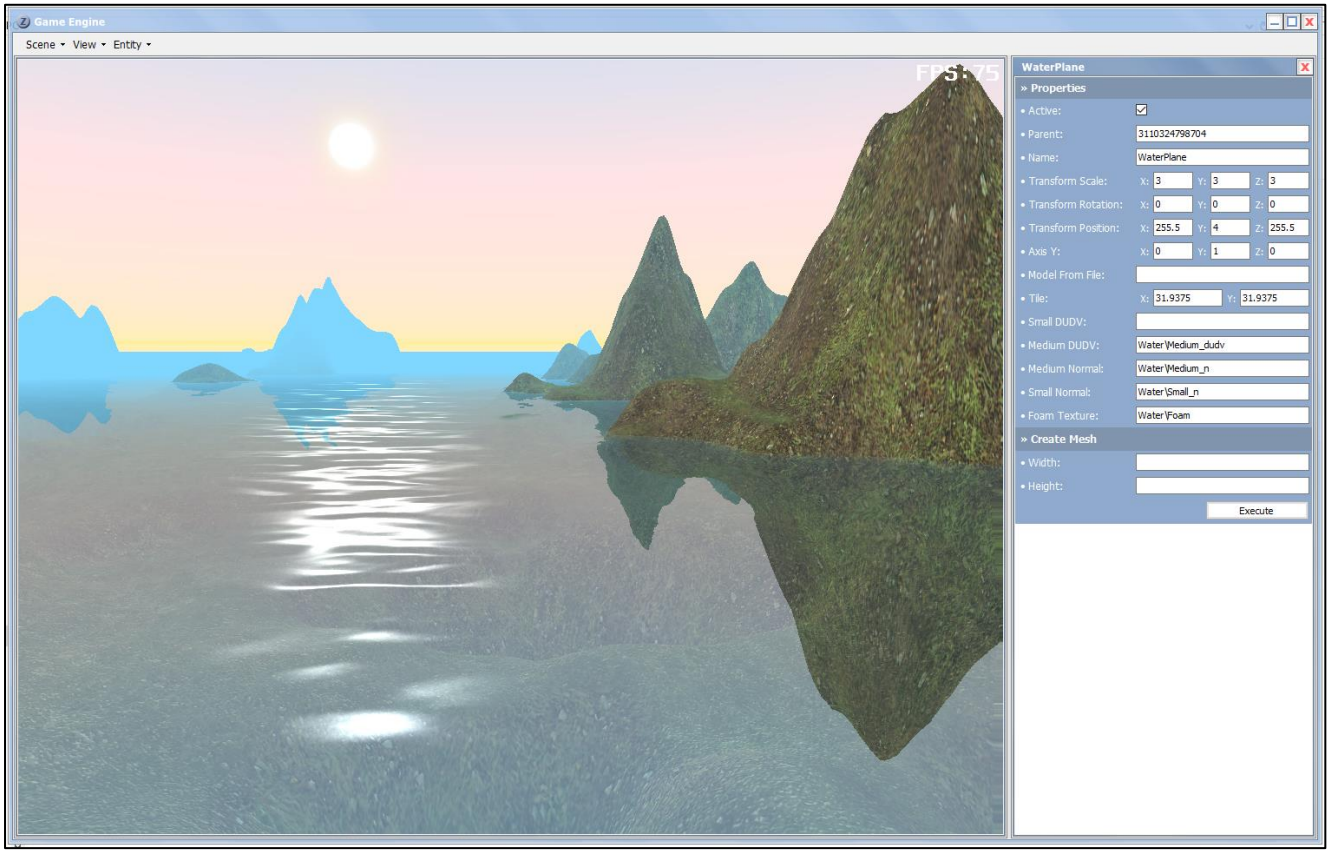


[2017] - Vortex Game Engine

I developed the Vortex Game Engine from scratch using C++ and OpenGL, which was a highly technical project that taught me a great deal about low-level graphics programming and ga me engine architecture. The engine uses an entity-component system (ECS) to manage game objects and their behaviors efficiently. I focused heavily on optimization, multi-threading, designing custom data structures such as hashmaps and linked lists to ensure fast data access.



One of the more challenging aspects of this project was creating my own event system, which allowed for decoupled communication between different parts of the engine. I also developed the user interface using C#, and to bridge the gap between the C++ engine core and the C# UI, I wrote a reflection system that automatically exports C++ functions to be accessible in the C# layer. This allowed for greater flexibility and ease of use when designing tools and interfaces for the engine.




```cpp
//[Reflectable]
class WaterPlane : public WorldObject
{
public:
    WaterPlane();
    ~WaterPlane();

    void OnStart() override;
    void OnUpdate(GameTime GameTime) override;
    void OnDraw(GameTime GameTime) override;
    void OnDrawDepth(GameTime GameTime)override;

    WaterPlane* Clone() override;

    virtual void LoadAttribute(ifstream& Stream, vector<string>& Attribute) override;

    //[Export]
    void CreateMesh(float Width, float Height);

    //[Export]
    string GetFoamTexture();
    //[Export]
    void SetFoamTexture(string File);

    //[Export]
    string GetSmallNormal();
    //[Export]
    void SetSmallNormal(string File);
```