Design Goals

Our goal was to create an action roguelike "Survivors-like" game where the player controls a Lich who can manipulate and use his own soul in combat. Our game was heavily inspired by Vampire Survivors and Death Must Die.

The most important design goal of our game was how the controls felt. We strove to have a game that felt nice to play from the get-go, where the player character moved and attacked in a satisfying way by eliminating as much jank that went against that satisfaction as possible.

Player

We had a specific "Body and Soul" type of fighting that we wanted the player to be able to utilize, namely:

- The player can use the Lich's normal form (his body, in a sense) as a means to fight enemies in melee range.

- The player can utilize his soul as an alternate way to fight enemies from a ranged distance.

- The player can "dash" to his soul, This also damages enemies if travel distance is great enough.

Multiple different iterations were tested, improved, or discarded depending on how good it felt to play and if it brought out the fun of the game. For example, the soul was originally supposed to take damage, and/or have a separate health bar. We realized that it went against the fun of being able to utilize the soul's dash offensively and racking up waves to crash into, and so we discarded the idea and made the soul invulnerable instead.

Enemies

We had specific design goals for the enemies in the game, each enemy had to:

- Have telegraphed attacks that clearly show where the player can expect to take damage.

- Be easy to defeat one on one, but harder when the enemies are in larger numbers.

- Every enemy is driven by a state machine that transitions based on basic inputs.

Minibosses

Minibosses were an iteration of basic enemies with enlarged sprites, different coloring and more health. They also drop better rewards, in the form of chests, that increase your level.

Bosses

Bosses had similar considerations, but we went more in depth when designing them. They had to have better decision making and multiple abilities with cooldowns. The Necromancer boss has two abilities, which he uses when they are off cooldown:

- Lightning attack
    - Cooldown: 6 seconds
    - Damage: 1 per frame (70 damage max)
    - Follows the player, slows down over time using a smoothstep function

- Wave attack
    - Cooldown: 8 seconds
    - Damage: 30
    - Large area of effect (AOE) around the enemy

On top of having to use both abilities, the Necromancer had better decision making than a standard enemy. He had to:

- Check and perform attack or spell if within range and cooldown has passed
- Enforce minimum distance without overriding attack/spell actions
- Approach player if not close enough to attack/cast and not currently performing an action
- Approach player if they had recently used an ability

Enemy Wave Spawner

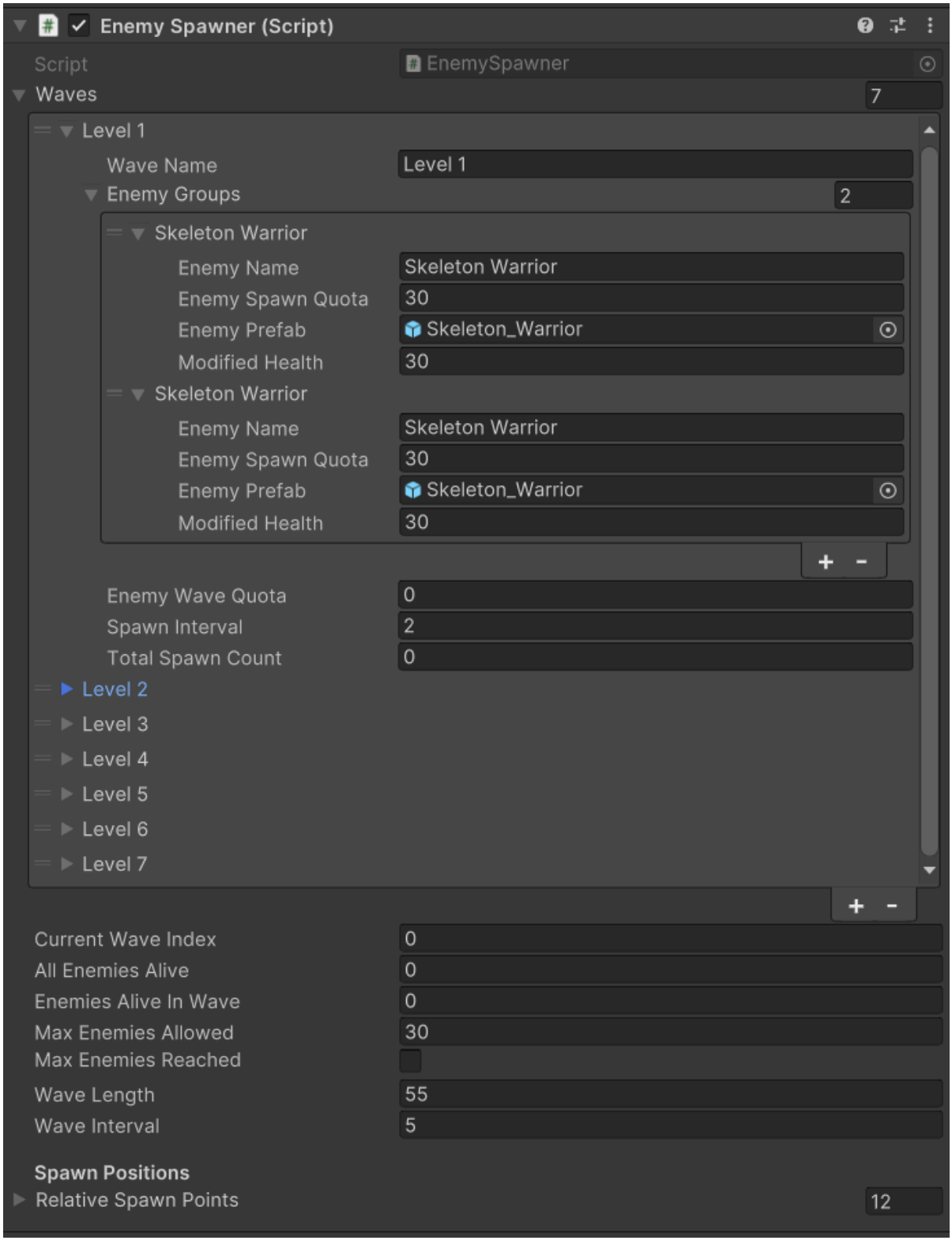
We wanted a system that could create waves of these enemies, customizable depending on how many could spawn, in what interval, and other such things. We settled on basing our spawner system on the intended spawn system of Vampire Survivors, with a few changes of our own. Taken from the wiki:

> Upon entering a stage, an initial amount of enemies depending on the stage modifiers spawns. Thereafter the game periodically attempts to spawn enemies. Enemies generally spawn just outside the screen and can despawn if the player moves far enough from them.
>
> Enemies normally arrive in waves - one wave every minute. Each wave specifies a minimum amount and a spawn interval for the enemies. If the minimum amount is not met when the game attempts to spawn enemies, enemies will be spawned until the quota is filled. If more enemies than the minimum amount is present, one of each type of enemies in the wave will be spawned. When 300 or more enemies are alive the game will not spawn more enemies periodically, and only bosses and enemies from map events can spawn.
> (https://vampire-survivors.fandom.com/wiki/Enemies#Waves)

Our enemy spawner fulfilled these prospects:

- Waves
    - Could create new waves.
    - Could choose which type of enemy groups (prefabs) would spawn in a corresponding wave, as well as their spawn interval.
    - How long a wave would last, and the interval between waves. Default was 55 seconds, followed by a 5 second interval.
- Enemies:
    - Could choose how many enemies of an enemy group would be spawned for a wave.
    - Could modify the health of an enemy group, in order to beef up the enemies in later waves.

- Spawning
    - How many enemies were allowed to be alive at once in the game.
        - If a new wave was about to begin and there were still enemies that needed to be spawned from the previous wave, the game would spawn every remaining enemy in bulk during the wave interval.
        - Enemies that were from a previous wave would not be counted towards the max cap of enemies allowed at once. Essentially allowing for enemies from the previous wave to exist, whilst also allowing more enemies of the current wave to spawn. This was done to prevent stalling and to prevent players from stopping the current wave from spawning enemies if they were at the cap.
    - Enemies had 12 relative spawn points to choose from that were scattered just outside of the player's camera. When spawning enemies, it would take the player's position, combined with one of the spawn position's coordinates to spawn enemies outside of the player's perspective.
        - If the player ran away from enemies, they would be re-positioned to one of those relative spawn points in order to combat stalling and despawning of enemies.


Enemy Spawner (Script)
Script
EnemySpawner
Waves
7
Level 1
Wave Name
Level 1
Enemy Groups
2
Skeleton Warrior
Enemy Name
Skeleton Warrior
Enemy Spawn Quota
30
Enemy Prefab
Skeleton_Warrior
Modified Health
30
Skeleton Warrior
Enemy Name
Skeleton Warrior
Enemy Spawn Quota
30
Enemy Prefab
Skeleton_Warrior
Modified Health
30
Enemy Wave Quota
0
Spawn Interval
2
Total Spawn Count
0
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7
Current Wave Index
0
All Enemies Alive
0
Enemies Alive In Wave
0
Max Enemies Allowed
30
Max Enemies Reached
Wave Length
55
Wave Interval
5
Spawn Positions
Relative Spawn Points
12

Level Up System

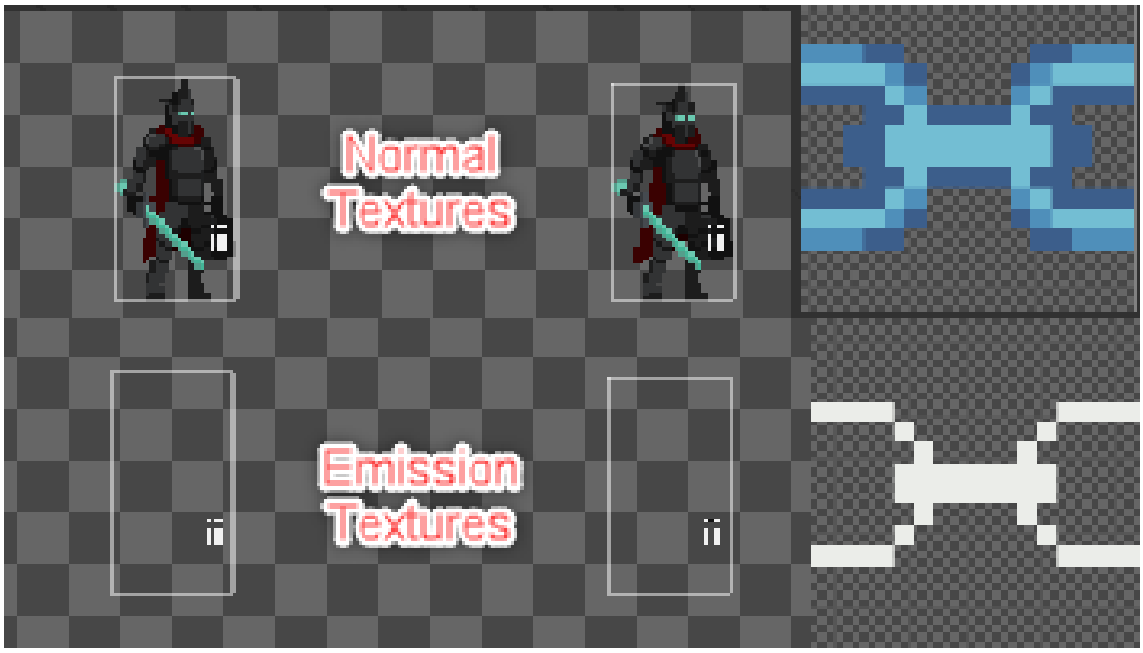
We wanted a system similar to the one in Vampire Survivors, where upon killing an enemy they drop an experience gem that the player collects to gain experience for leveling. The main difference between our system and the one in Vampire Survivors is that ours was more simplified. In our game, the player can select from a list of upgrades that upgrade a single stat for one weapon, either the Main weapon attack of the Lich or the ranged attack of the Soul. Each upgrade also has rarities which are weighted differently.The higher the rarity the better (and rarer) it is.

This system rewards players that have played the game before, since they learn what is best to upgrade and what to skip. This also makes the game feel different if you focus on different aspects.

Experimental features:

This project allowed us to try a few things that we have never attempted before.

We tried to increase the visual feel of the connection between the Lich and the soul by using emission sprites as secondary textures and shaders that light them up, giving them a bloom/glow feeling.



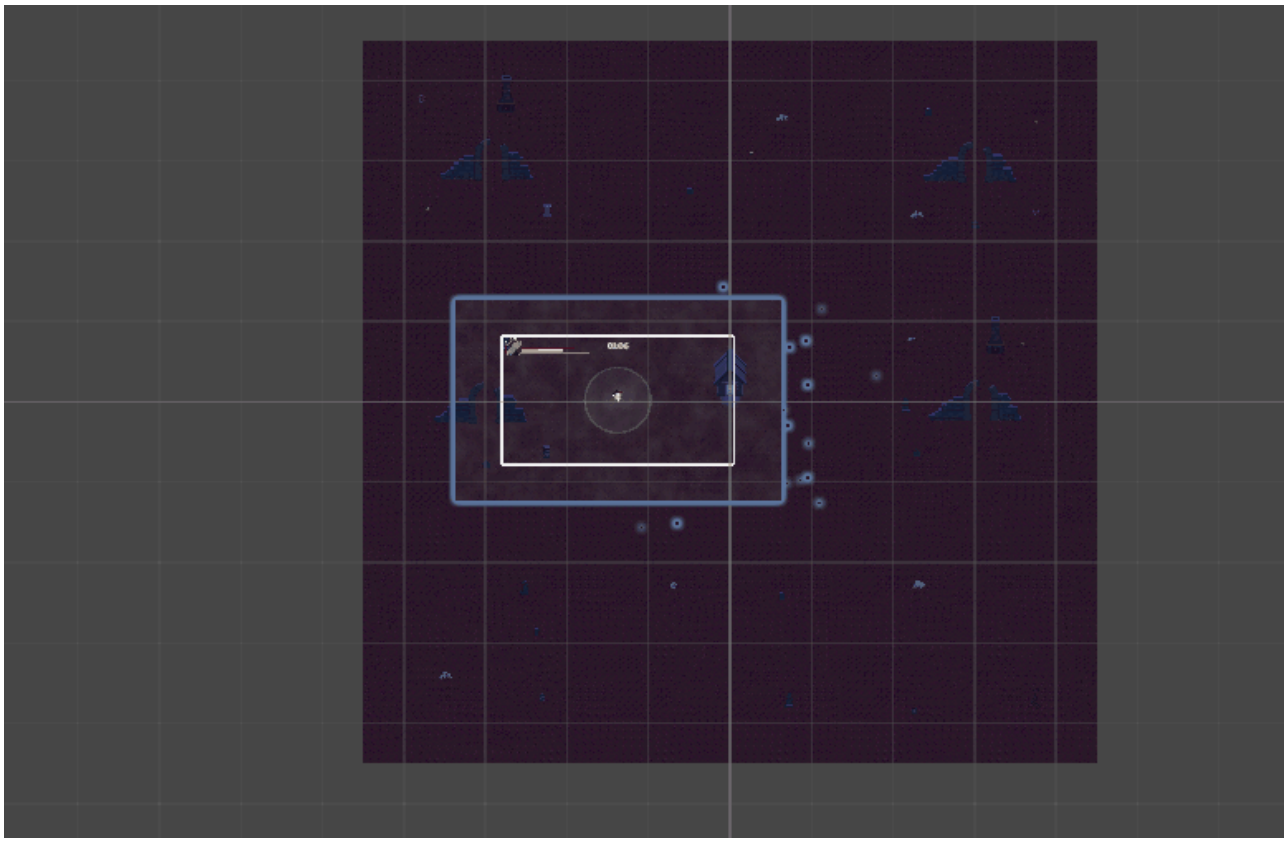
The screenshot on the left showcases the normal and emission textures, featuring frames of the player's idle animation on the top, and their corresponding emission textures below.
The screenshot on the right, likewise showcases the chain texture and the chain emission texture.

The emission textures are then utilized by the shader which increases the intensity of the light and gives it the desired effect.

In addition to the emission textures, we also had two layers of shaders that worked as our fog system, they had different speeds to make the fog seem more natural. We also had a particle system that spawned small wisp-like light particles to enhance the atmospheric feel.

Map generation:

Early on in the project we swapped from a static player map, to a randomly generated one that chooses random chunks to spawn around the player. Each chunk is then given a weight to choose which chunks should spawn more often, and which ones less so. These chunks are then deactivated when the player moves away from them, to be reused later. The chunks also have prop spawn locations that have a similar functionality to the chunks. Props are chosen at random and each one has a weight. The prop with the most weight was an empty prop, as the world felt too cluttered with objects allowing them to spawn in every location.



These props locations would have housed interactables as well, such as shrines (battle-shrines, blessings), and breakables (would contain a health potion, rare gem, etc.). Unfortunately, those could not be implemented in time.