**Mason Sanders - CSCI 43800 Final Project Documentation**

**About The Game:**

My final game is a first-person platformer that takes place in an alternate dimension. The goal is to collect the light sphere and bring it back to the obelisk that’s at the start of each level. There are 4 levels in total. In each level there are obstacles that the player must figure out how to overcome to progress through the level. The first mechanic introduced is basic platforming, in the first level the player must use double jumps to get to harder to reach platforms. In the second level, wall jumping is introduced. In the third and fourth levels, turrets and electric fields will stand in your way. When the player completes all 4 levels, they will receive a “The End” screen.

**Controls**

General Movement: WASD

Jump: space bar

Double jump: space bar while in the air

Wall jump: space bar while on a wall (note: after a wall jump is performed and the player is off a wall, the player will be allowed to double jump again)

Shoot: mouse left click (you can block turret bullets with your own bullets)

Interact(deposit light at the obelisk or press a button): e

Quit: Esc

**Scenes and Descriptions**

*Level, Level2, Level3, Level4:*

These are the scenes used for the actual levels they contain all the other scenes that are needed to make a whole level.

*Player:*

This is the scene that the player controls, it has no mesh, but contains a collision shape and a spatial node for the camera and the gun

*LightSphere:*

This scene represents the light that the player picks up and brings to the obelisk to complete a level. It’s made using particle effects

*button:*

This represents the button that the player has to press to disable electric fields. All buttons are added to a Buttons node group in each level

*ElectricField:*

This is the electric field made using particles, each electric field is added to an ElectricFields node group in each level

*Obelisk:*

This is the strange floating object at the beginning of each level, the player’s goal is to bring the light sphere to this object to be able to start the next level

*Platform:*

These are the blue platforms that the player needs to jump across to progress through the level

*DeathPlane:*

An invisible box that sits far beneath the level, if the player touches this, they die and the level resets.

*HUD:*

The canvas layer that displays the player’s current health.

*EndHUD:*

The canvas layer responsible for displaying the “THE END” message at the end of the game

*EndScreeen:*

The 3D scene displayed at the end of the game, it contains the EndHUD and an immovable camera.

*Explosion:*

The particle effects that are spawned once A PlayerBullet or TurretBullet collide with an object

*PlayerBullet:*

The energy bullet that the player is able to shoot, indicated by light coloring.

*TurretBullet:*

The energy bullet that the turret is able to shoot, indicated by darker coloring.

*Turret:*

The turret that shoots the player if they get too close. It’s a rigid body so the player can knock it off of platforms

*Portal:*

The dark portal that appears once the light is deposited at the obelisk, stepping through it starts the next level, or goes to the EndScreen if the player is on Level4

**Scripts and Descriptions:**

*Level.gd:*

This script is attached to all the level scenes. It sets the mouse mode, and controls the switching of levels by getting the current scene name and moving to the next scene appropriately.

*Player.gd:*

This script is attached to the player, The script’s physics process function handles death, motion and jumping, input, collisions, and interactivity with other objects. Some of these are factored out into different functions to make physics process easier to read. Unhandled input handles mouse movement.

The ready function sets up some sound effects, and preloads the player bullet. The electric field index is also set to 0 here so that the correct electric fields are disabled upon pressing buttons.

*Button.gd:*

This script is attached to buttons and determines whether or not the player can press the button if they’re close enough to it.

*HUD.gd:*

This script is attached to the HUD and its job is to make sure the label on the canvas layer displays the correct player health.

*Turret.gd:*

This script controls the turret. The turret will spawn TurretBullets when the player is close enough, and it spawns them traveling in the unit vector direction to the player. The fire rate of the turret is also controlled through a fire rate variable and a rate counter variable. Rate counter starts at 0 and counts up on each call to the process function. Once the rate counter is equal to the fire rate, the turret can spawn a bullet, and the rate counter is reset to 0.

*TurretBullet.gd and PlayerBullet.gd:*

The TurretBullet and PlayerBullet scripts act in mostly the same way with only a few differences. They both travel until they collide with an object, or are a certain distance from the player, then they are freed. Since the turret is a rigid body, the PlayerBullet must be able to get a Kinematic collision from colliding with the turret, so infinite inertia for move and collide must be false, and the bullet applies a central impulse to the turret upon collison. The TurretBullet has infinite inertia set to false, but no impulse is applied, this makes it so the turret doesn’t accidentally shoot itself off of a platform.

*Portal.gd*

This script is attached to the portal in each level that leads to the next. But since Level.gd handles changing levels, all this script does is make it so that the collision and visibility are disabled upon starting the level, so the player can’t just immediately move to the next

*ObeliskLight.gd:*

This script is only attached to the LightSphere that appears next to the Obelisk upon completing a Level. Like the portal, this script’s job is to disable visibility at the start of the level. The Player.gd will later make this and the portal visible upon interaction with the obelisk.

*Explosion.gd:*

This script is for the explosions that spawn when the bullets collide with objects. Since the explosion is a one shot particle effect, this script causes an emission upon ready being called, and once the emission is finished, the explosion is freed.

**Tools and External resources used:**

The only external resources used are the particle effect images provided by Kenney at:

<https://www.kenney.nl/assets/particle-pack>

Other than that, All models and materials were made using blender, with the exception of the procedural sky, which is in the player’s camera in godot. And of course the game was made using Godot. The 4 sound effects for pressing the button, collecting the light, getting shocked by the electric field, and opening the portal were made using FL Studio 20.