Assignment 1: Exploring a Game Engine

Jm98757 – Juan Diego Mendez

Functionality:

The Scripts define types wall and ball, and for this assignment the simulation basically generates 3 ball type objects inside a box made out of 6 wall type objects. Each one has their one collision and mesh child nodes. The balls due to their coded behavior move around with a set speed (I put different speeds for each of the 3) And they also have an omni-light node which ends up making the simulation look interesting. The main thing ball type objects do is detect when area_entered is called and when there is a collision they basically bounce in a completely elastic way. For collisions with other balls the program calculates the distance between the balls and divides it by 2. That is how it gets a vector of the normal which is used with the R = I - 2N(N·I) formula to get the resulting vector. And for the walls the type has an implemented method which can be called to receive the normal for the wall (which can be set in the editor).

Issues:

I faced many issues, at first mainly with binding the methods, but later also by thinking about collision detection in the wrong way. I finally was not able to use a raycast to make an interaction for the extra credit.

How to Play the simulation:

Open in a terminal the directory named gdextension_cpp_example. Then execute the command scons platform=(your platform). Once that is done, open the godot binary and inside godot open the project and play the scene named main

Code Layout:

Script 1 - Ball:

The Ball script defines the behavior of a 3D ball that moves within the game environment.

It provides methods to get and set the speed of the ball.

The ball's position is updated based on its speed and direction in the _process method.

It handles collisions with other objects by emitting signals ball_hit and wall_hit.

The _on_ball_collision method calculates the new direction of the ball after collisions with walls or other balls.

```
void Ball::_on_ball_collision(const Area3D* area) {

    if (area->get_class() == "Wall") {
        UtilityFunctions::print("Collision Wall");
        emit_signal("wall_hit");
        Wall* wall = const_cast<Wall*>(reinterpret_cast <const Wall *>(area));
        direction = calculate(wall->get_normal());
        direction.normalize();

    } else if (area->get_class() == "Ball") {
        UtilityFunctions::print("Collision Ball");
        emit_signal("ball_hit");
        Vector3 ball_normal = area->get_position() - get_position();
        direction = calculate(ball_normal);
        direction.normalize();

    } else {
        UtilityFunctions::print("HUUUUH");
    }
```

This is the main method that makes collisions possible.

Script 2 - Wall:

The Wall script defines the behavior of 3D walls in the game environment.

It provides methods to get and set the normal vectors of the walls in the x, y, and z directions.

The normal vectors can be manipulated to change the orientation of the walls.

This script doesn't have a significant role in the gameplay but serves as a part of the game environment.

```
ClassDB::add_property("Wall", PropertyInfo(Variant::FLOAT, "normal_x",
PROPERTY_HINT_RANGE, "-1,1,0.01"), "set_normal_x", "get_normal_x");
    ClassDB::add_property("Wall", PropertyInfo(Variant::FLOAT, "normal_y",
PROPERTY_HINT_RANGE, "-1,1,0.01"), "set_normal_y", "get_normal_y");
    ClassDB::add_property("Wall", PropertyInfo(Variant::FLOAT, "normal_z",
PROPERTY_HINT_RANGE, "-1,1,0.01"), "set_normal_z", "get_normal_z");
```

These are the lines that add the property to modify x y and z values for the normal of each wall, these have to be put manually through the editor for this class, but they are already set for the demo.

Script 3 - Audio:

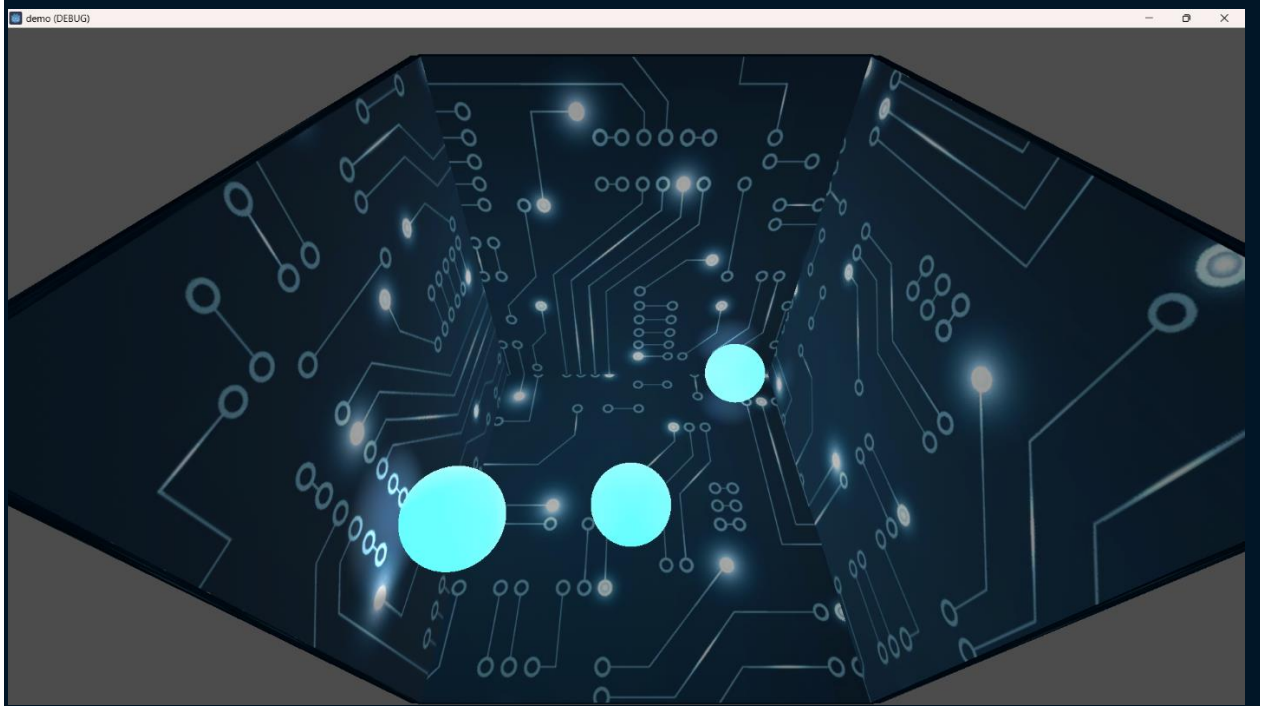
The Audio script is written in C# and inherits from AudioStreamPlayer3D, which is a Godot class for 3D audio playback.

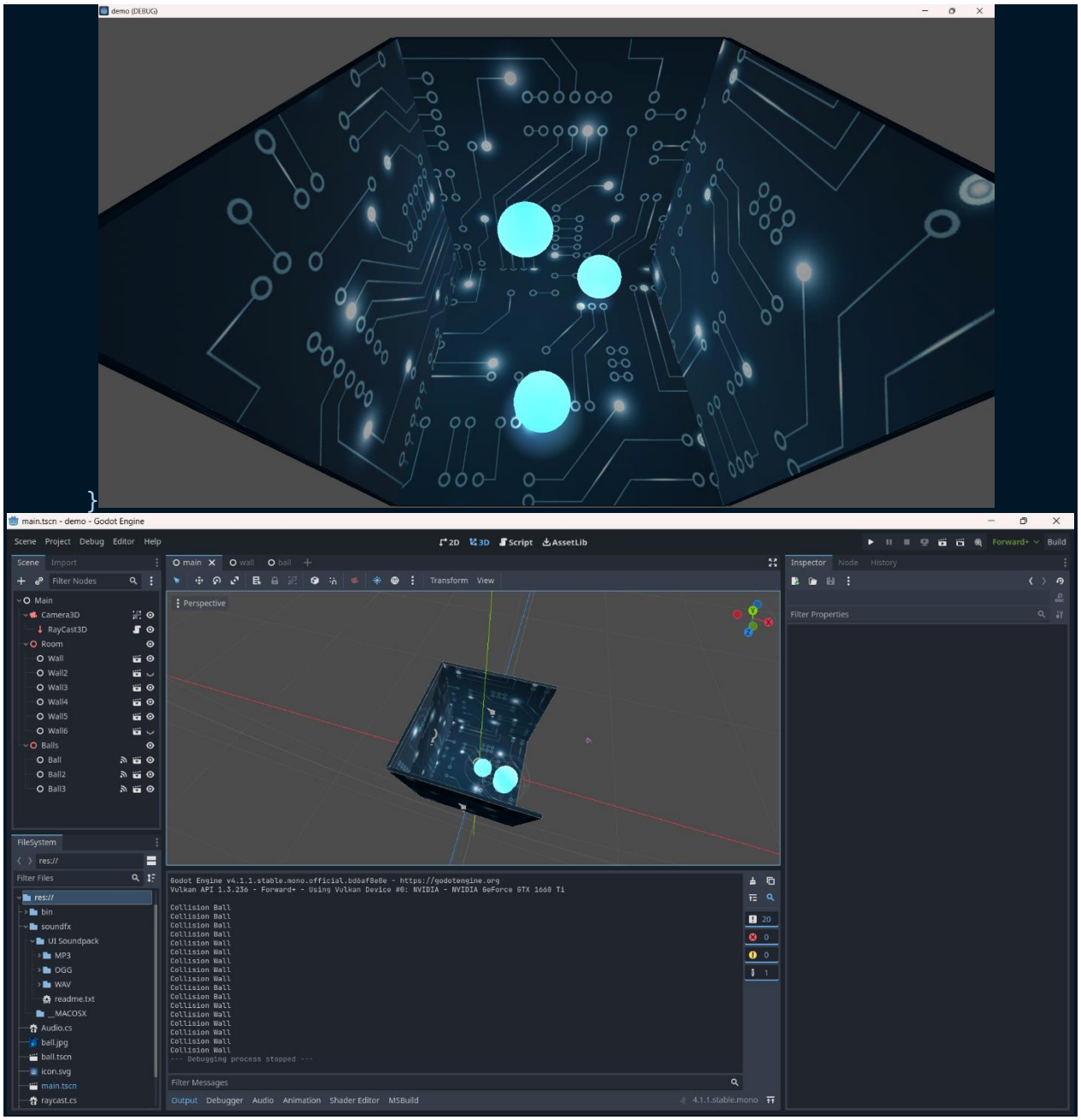
It loads and plays audio files when specific signals are emitted from other game objects (e.g., wall_wall_hit and ball_ball_hit).

The choice of which audio file to play is random, adding variability to the sound effects.

```
public void _on_wall_wall_hit() {

        if (GD.Randf() >= 0.5) {
            this.Stream = ResourceLoader.Load<AudioStream>("res://soundfx/UI
Soundpack/MP3/Minimalist11.mp3");
        } else {
            this.Stream = ResourceLoader.Load<AudioStream>("res://soundfx/UI
Soundpack/MP3/Minimalist3.mp3");
        }
        this.Play();
```




demo (DEBUG)
main.tscn - demo - Godot Engine
Scene Project Debug Editor Help
2D 3D Script AssetLib
Forward+ Build
Scene Import
Filter Nodes
Main
Camera3D
RayCast3D
Room
Wall
Wall2
Wall3
Wall4
Wall5
Wall6
Balls
Ball
Ball2
Ball3
main wall ball
Transform View
Perspective
Inspector Node History
Filter Properties
FileSystem
res://
Filter Files
bin
soundfx
UI Soundpack
MP3
OGG
WAV
readme.txt
__MACOSX
Audio.cs
ball.jpg
ball.tscn
icon.svg
main.tscn
raycast.cs
Godot Engine v4.1.1.stable.mono.official.bd6af8e8e - https://godotengine.org
Vulkan API 1.3.236 - Forward+ - Using Vulkan Device #0: NVIDIA - NVIDIA GeForce GTX 1660 Ti
Collision Ball
Collision Wall
--- Debugging process stopped ---
Filter Messages
Output Debugger Audio Animation Shader Editor MSBuild
4.1.1.stable.mono