ASSIGNMENT

of

Augmented Reality and Virtual Reality Laboratory
CS 435

Bachelor of Technology (CSE)

By

Vedant Shukla (21124042)

Fourth Year, Semester 7

Course In-charge: Prof. Darshan Parmar

Department of Computer Science and Engineering

School Engineering and Technology

Navrachana University, Vadodara

Spring

GitHub Repository Link: Inferno-Toss-VR
Demo Video Link: LINK

Objective:

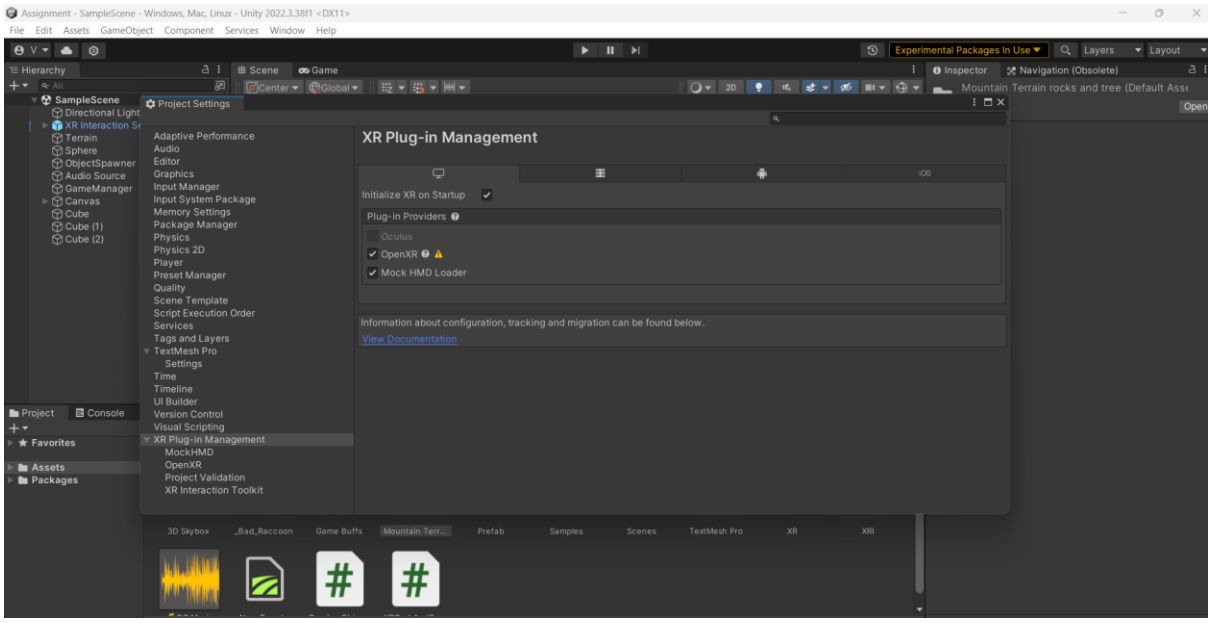
This project aims to create a VR environment using Unity, where players can interact with grabbable objects to score points. The game includes stationary and moving objects and a scoring system that awards points when an object is thrown or grabbed.

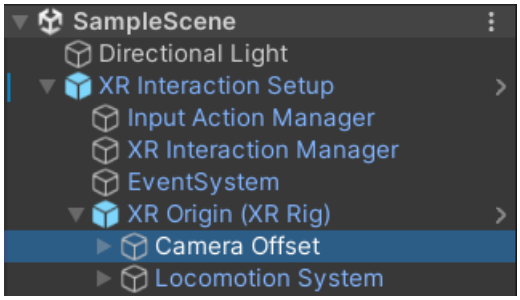
Project Workflow and Implementation

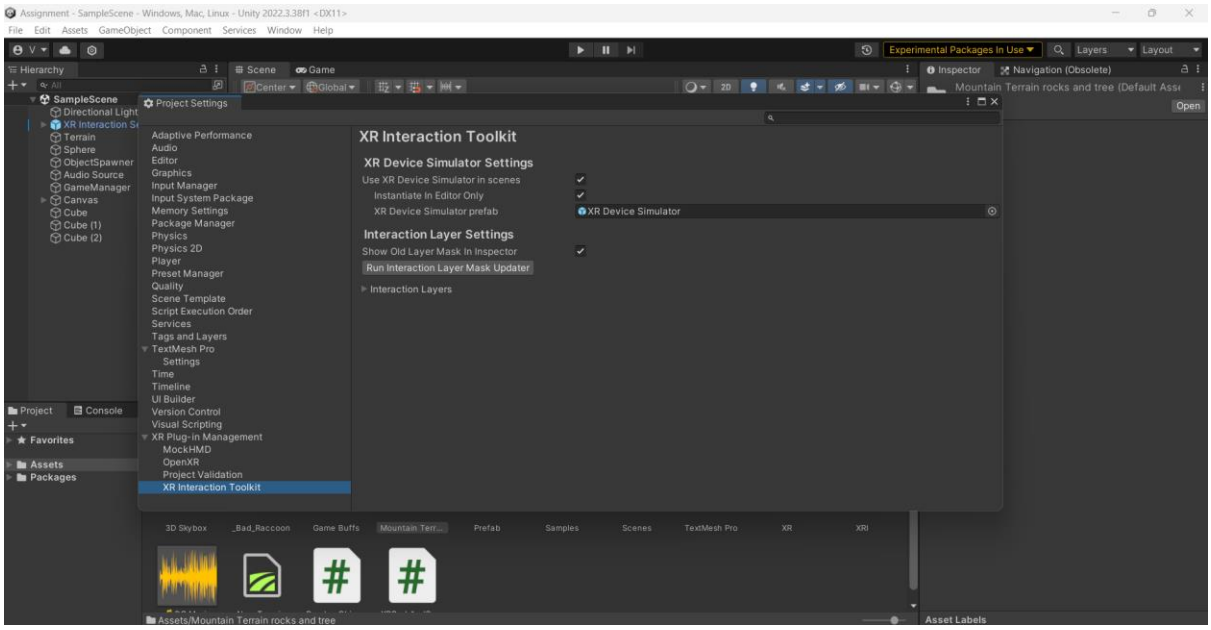
Task 1: Setting Up the Unity Project and VR Environment

1. Created a Unity project using the Universal Render Pipeline (URP).
2. Installed the XR Interaction Toolkit to enable VR interactions.
3. From the samples, imported the XR Device Simulator Toolkit.
4. Set up XR Plugin Management for compatibility with VR devices (Oculus, OpenXR).
5. Configured XR Origin to manage the VR environment.

Screenshots:



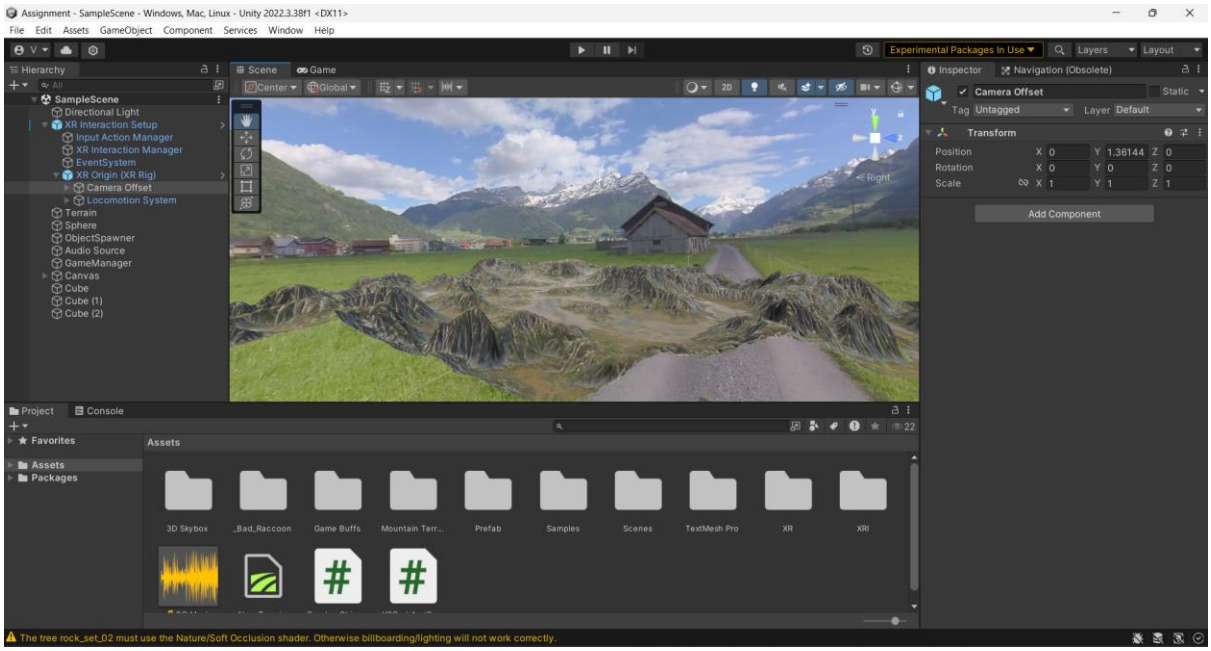


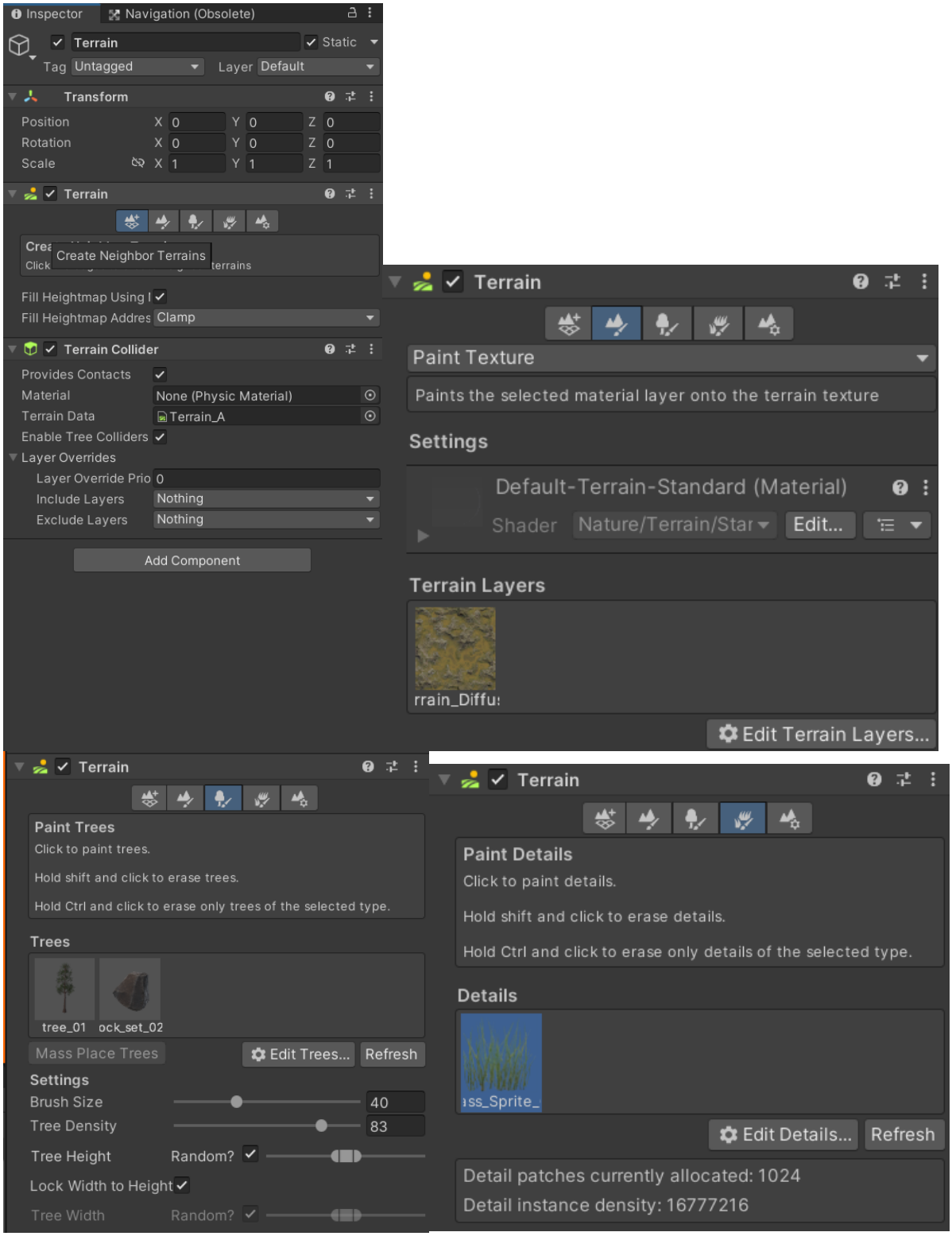


Task 2: Ground Plane and Skybox

1. Imported a free terrain asset from Unity's Asset Store to create the game environment.
2. Used a free skybox asset from Unity Asset Store to enhance the visual appeal.
3. Modelled terrain with Unity’s terrain tools to simulate a mountainous environment.

Screenshots:



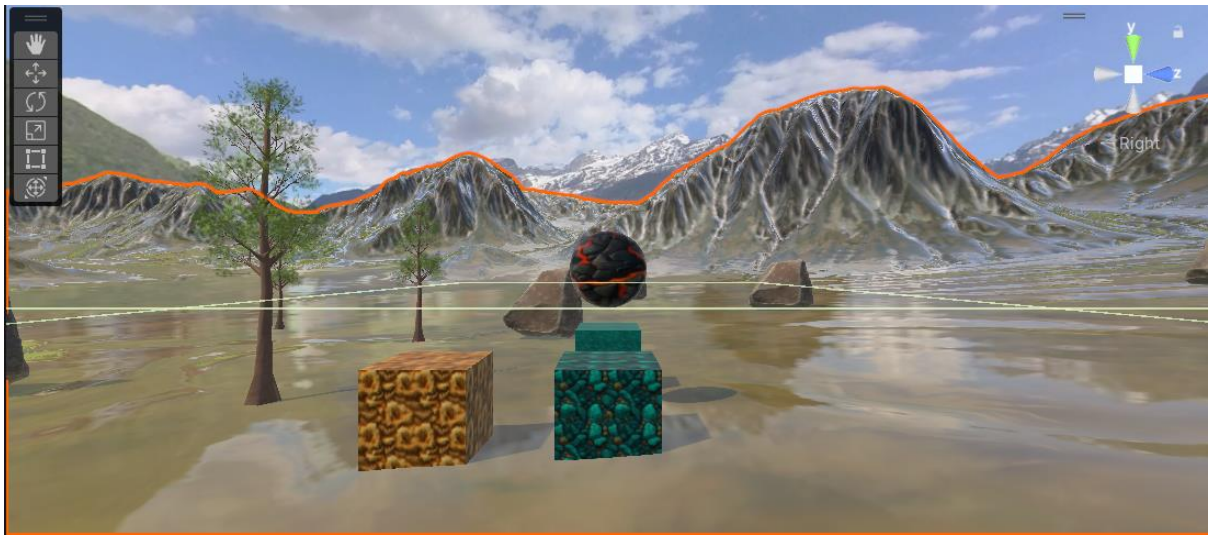


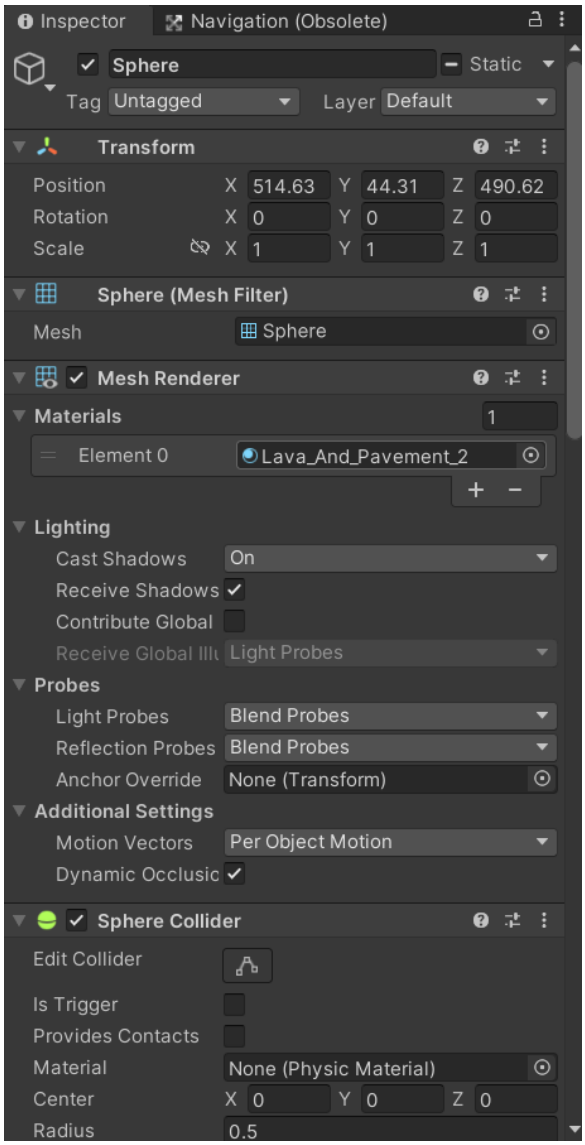
Task 3: Adding Environment Objects

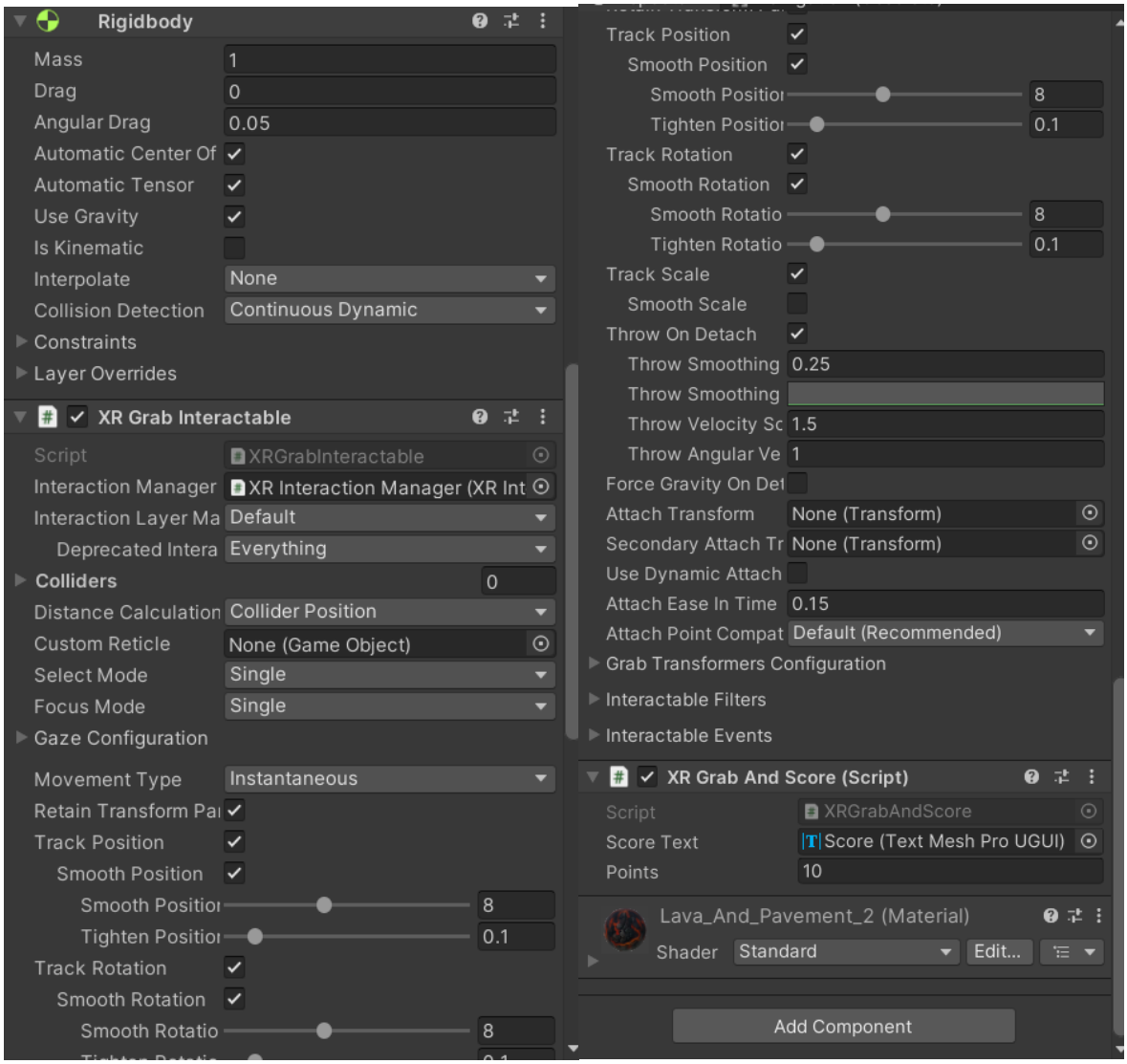
1. Expanded the terrain to provide enough space for movement and interaction.
2. Added environmental elements like trees and rocks for a more immersive experience.
3. Used lava assets with animation effects to form the central game element.

4. Also used different elements of water and wood for cubes and spheres.

Screenshots:







Task 4: Lighting and Shadows

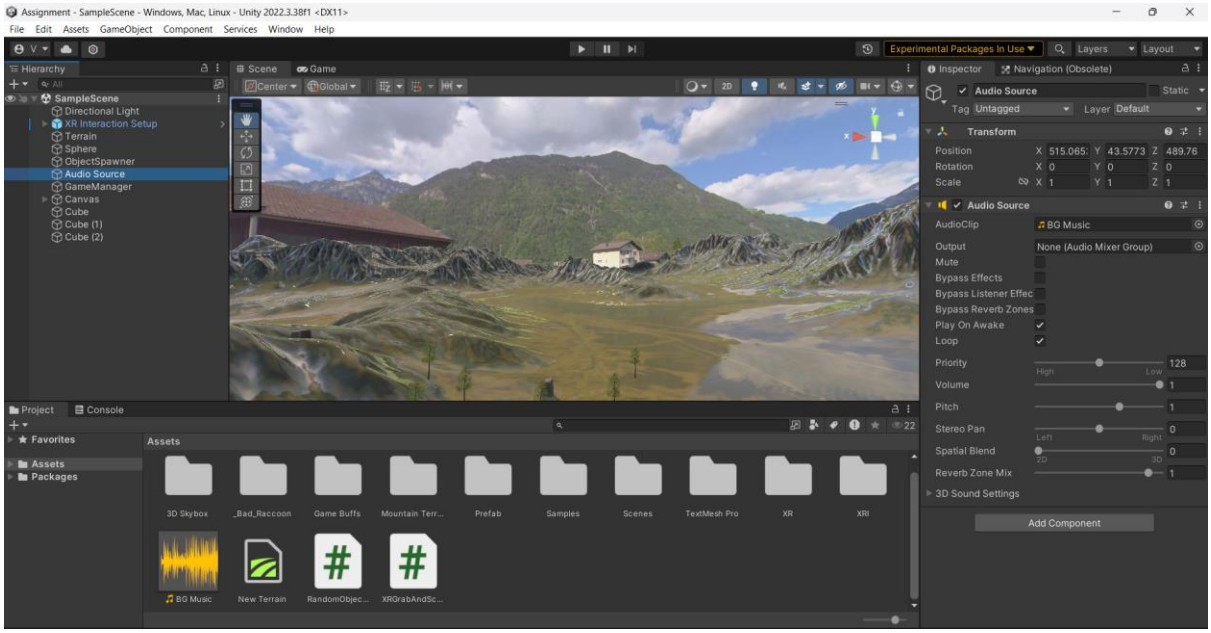
1. Added directional lighting to simulate sunlight and create a realistic effect.
2. Added Shadows and Reflections.
3. Configured lightmaps for real-time and baked lighting effects.

Screenshots:

Task 5: Audio Implementation

1. Created an object: Audio->Audio Source
2. Selected a small length audio clip for the background audio and put it on loop.
3. Configured an audio source in an empty game object to play the background music during gameplay.

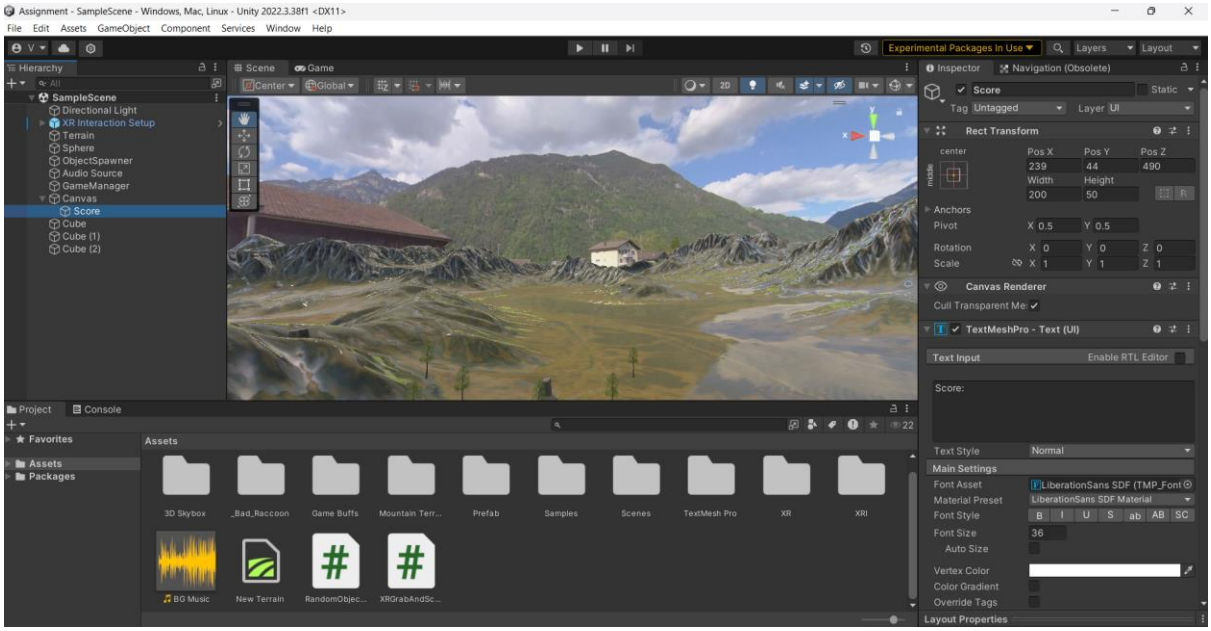
Screenshots:



Task 6: VR Interaction, Scripting and Scoring Mechanism

1. Object Interaction
   Utilized the XR Interaction Toolkit to configure objects that players can grab and throw.

- Added the XRGrabInteractable component to each object to enable interaction.
- Configured controllers to grab and release objects, allowing players to engage with the environment and score points.
- Created an empty object and named it" Object Spawner" and attached random object spawner script.
- Attached the XRGrabAndScore.cs script to sphere and cubes.
- Created a Canvas and TextMeshPro and named it score for displaying scores.



Script 1: XRGrabAndScore.cs

This script enables the scoring mechanism:

- Awards 10 points for each object.
- Updates the score on a UI element using TextMeshPro.

XRGrabAndScore.cs

Assets > XRGrabAndScore.cs

```cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.XR.Interaction.Toolkit;


public class XRGrabAndScore : MonoBehaviour
{
    // The player's score

    private int score = 0;

    // Reference to TextMeshPro UI element to display the score

    public TextMeshProUGUI scoreText;

    // Points awarded for grabbing and releasing the object

    public int points = 10;

    // Reference to XR Grab Interactable component

    private XRGrabInteractable grabInteractable;

    // Flag to check if the object is currently grabbed

    private bool isGrabbed = false;


    void Start()
    {
        // Initialize the score display at the start
        UpdateScoreText();

        // Get the XRGrabInteractable component on the object
        grabInteractable = GetComponent<XRGrabInteractable>();
```

```cs
        // Get the XRGrabInteractable component on the object
        grabInteractable = GetComponent<XRGrabInteractable>();

        // Add event listeners for grab and release actions
        grabInteractable.selectEntered.AddListener(OnGrab);
        grabInteractable.selectExited.AddListener(OnRelease);
    }

    // Called when the object is grabbed
    1 reference
    void OnGrab(SelectEnterEventArgs args)
    {
        if (!isGrabbed)
        {
            isGrabbed = true;
            Debug.Log("Object grabbed!");
        }
    }

    // Called when the object is released
    1 reference
    void OnRelease(SelectExitEventArgs args)
    {
        if (isGrabbed)
        {
            Debug.Log("Object released!");

            // Increase the score when the object is released
            IncreaseScore(points);

            // Reset the grabbed flag
            isGrabbed = false;
        }
    }

```

```
60      }
61
62      // Increase the player's score
        1 reference
63      void IncreaseScore(int points)
64      {
65          score += points;
66          UpdateScoreText();
67      }
68
69      // Update the TextMeshPro score display
        2 references
70      void UpdateScoreText()
71      {
72          if (scoreText != null)
73          {
74              scoreText.text = "Score: " + score.ToString();
75          }
76      }
77  }
78
```

2. Random Object Spawning

Developed a system to spawn two types of objects: static cubes and moving spheres.

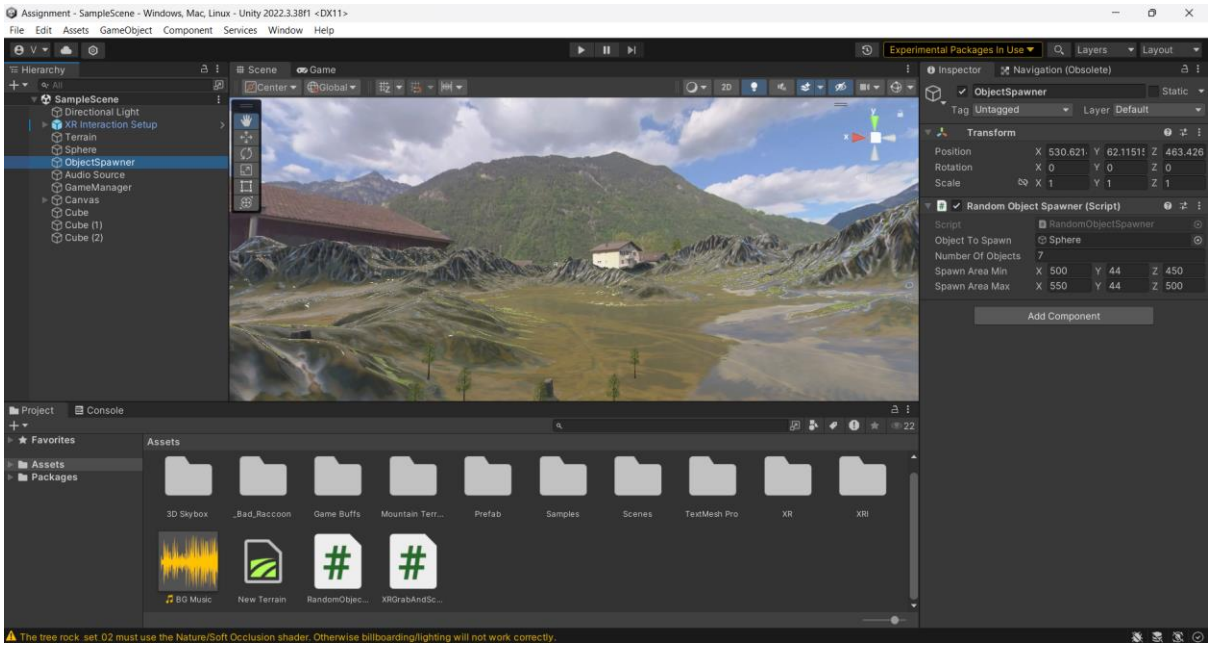
- Cubes are stationary and placed at fixed points for simple interaction.
- Spheres are spawned at random locations and move through the environment, adding a dynamic challenge.

Script 2: RandomObjectSpawner.cs

This script handles the random spawning and movement of objects:

- Configures random positions within defined boundaries.
- Spawns spheres that glide through the environment, encouraging player interaction.

Screenshots:

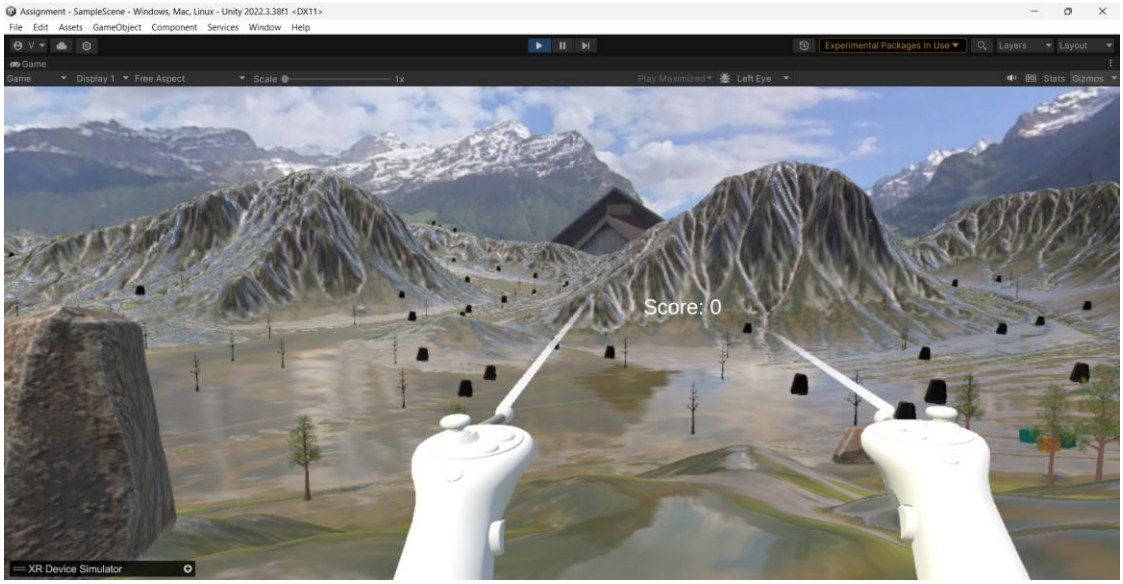


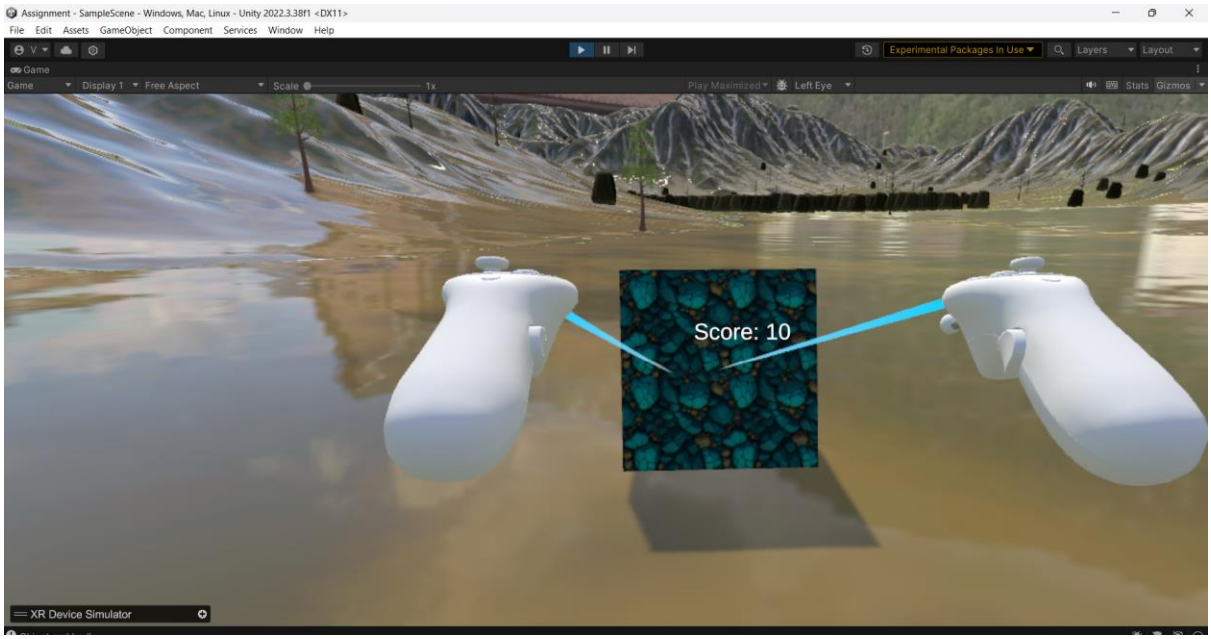
RandomObjectSpawner.cs

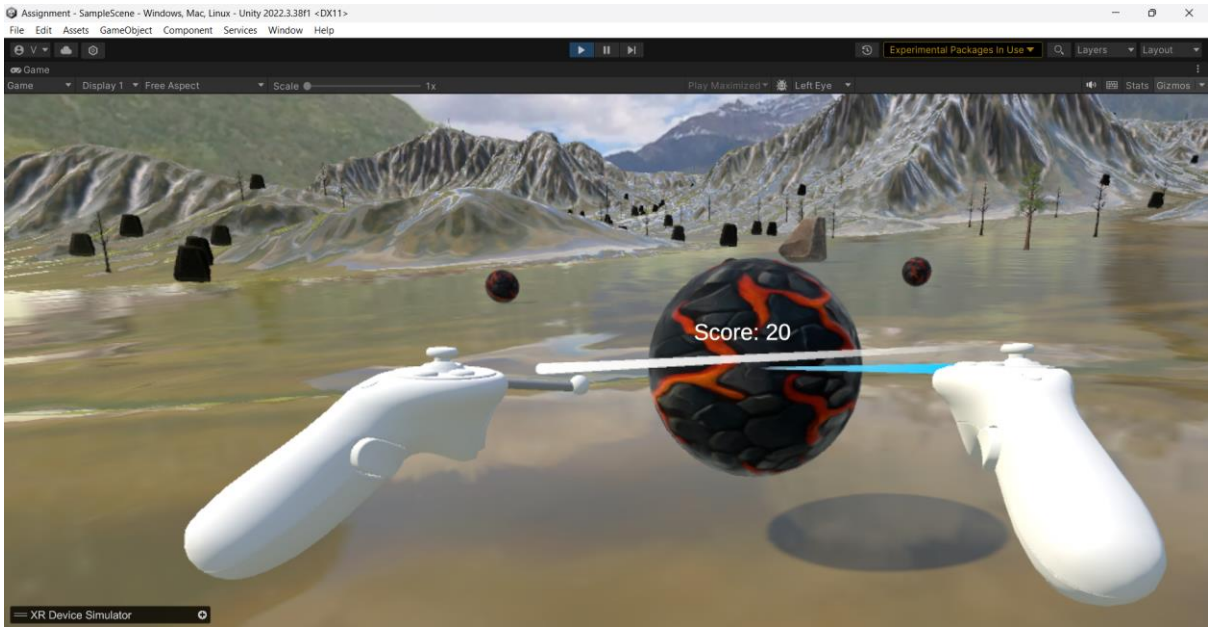
Assets > RandomObjectSpawner.cs > RandomObjectSpawner

```cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class RandomObjectSpawner : MonoBehaviour
{
    // Variables to assign the object you want to spawn and the spawn area limits
    1 reference
    public GameObject objectToSpawn;  // Assign your grabbable object prefab in the Inspector
    1 reference
    public int numberOfObjects = 2;  // The number of objects to spawn
    5 references
    public Vector3 spawnAreaMin;  // Minimum boundary of the spawn area
    5 references
    public Vector3 spawnAreaMax;  // Maximum boundary of the spawn area

    // Start is called before the first frame update
    0 references
    void Start()
    {
        // Spawn the objects when the game starts
        SpawnObjects();
    }

    // Method to spawn objects
    1 reference
    void SpawnObjects()
    {
        for (int i = 0; i < numberOfObjects; i++)
        {
            // Generate a random position within the defined area
            Vector3 randomPosition = new Vector3(
                Random.Range(spawnAreaMin.x, spawnAreaMax.x),
                Random.Range(spawnAreaMin.y, spawnAreaMax.y),
                Random.Range(spawnAreaMin.z, spawnAreaMax.z)
            );

```

```
            Vector3 randomPosition = new Vector3(
                Random.Range(spawnAreaMin.x, spawnAreaMax.x),
                Random.Range(spawnAreaMin.y, spawnAreaMax.y),
                Random.Range(spawnAreaMin.z, spawnAreaMax.z)
            );

            // Instantiate the object at the random position
            Instantiate(objectToSpawn, randomPosition, Quaternion.identity);
        }
    }

    // Optional: If you want to visualize the spawn area in the editor
    0 references
    private void OnDrawGizmosSelected()
    {
        // Draw a wireframe cube to show the spawn area
        Gizmos.color = Color.green;
        Gizmos.DrawWireCube((spawnAreaMin + spawnAreaMax) / 2, spawnAreaMax - spawnAreaMin);
    }
}
```







Conclusion

This VR project demonstrates fundamental VR interaction and scoring mechanisms in Unity. Players engage with a dynamic environment where they score points by grabbing and throwing objects, integrating both stationary and moving targets to enhance gameplay and difficulty. Points are awarded and displayed on the screen for the tasks.

Feedback and Challenges Faced:

I was once more familiar and preferred working in AR based games as I didn't have much knowledge about VR. But, through this project, I deepened my understanding of VR development in Unity, particularly with the XR Interaction Toolkit. Another main challenge for me was Implementing a scoring mechanism and random object spawning in C# along

with debugging it for different cases. This taught me valuable skills in Unity scripting and VR environment setup. Finally, with the help of my friends and my faculty I was able to complete this assignment.