Experiment No.5
Develop a scene in Unity that includes a sphere and plane . Apply Rigid body component, material and Box collider to the game Objects. Write a C# program to grab and throw the sphere using the vr controller.
Date of Performance: 07/09/23
Date of Submission: 05/10/23

AIM: -

Develop a scene in Unity that includes a sphere and plane . Apply Rigid body component, material and Box collider to the game Objects. Write a C# program to grab and throw the sphere using the vr controller.

OBJECTIVES: -

This Unity project aims to create an interactive scene with a sphere and a plane, incorporating physics-based realism. It includes applying Rigidbody components to the objects for natural movement, assigning distinct materials for visual differentiation, and implementing Box Colliders for accurate collision handling. The primary objective is to enable interaction with a VR controller through a C# program, allowing users to grab and throw the sphere realistically. Extensive testing ensures a responsive and immersive VR experience, while documentation ensures the project's reproducibility and future development. This project provides practical experience in VR controller integration, physics simulation, and scripting, enriching the skill set for immersive VR development.

THEORY: -

The theory behind this Unity project involves several key aspects of game development and VR interaction:

1. Scene Creation:
    a. Unity Scene: Unity provides a platform for creating 3D environments and simulations. In this project, a scene is created to host the interactive sphere and plane GameObjects.
2. Physics Simulation:
    a. Rigidbody Component: Rigidbody is a Unity component that adds physics simulation to GameObjects. It enables realistic behaviors like gravity, collision detection, and dynamic movement. Applying Rigidbody to the sphere and plane ensures they respond to physics forces accurately.
3. Material and Visuals:
    a. Materials: Materials define the visual properties of objects, including their color, texture, and transparency. Applying materials to GameObjects allows for customization of their appearance.
4. Collision Detection:
    a. Box Collider: Box Collider is a type of Collider component in Unity used for defining the shape and boundaries of 3D objects. Applying Box Colliders to the sphere and plane ensures that they can interact with each other and other objects in the scene.

5. VR Controller Interaction:
    a. C# Scripting: Unity allows developers to use C# scripts to create interactive behaviors. In this project, a C# script is written to interface with a VR controller.
    b. VR Controller Input: VR controllers have input sensors and buttons that can detect user interactions. The script interprets these inputs to enable the user to grab and release the sphere.
    c. Physics-Based Throwing: The C# script applies forces to the sphere based on the controller's movement, simulating the action of grabbing and throwing it within the VR environment.

In summary, this project combines Unity's physics simulation, material and collider components, C# scripting, and VR controller integration to create an engaging VR experience where users can interact with and throw a sphere within a virtual environment. These skills are fundamental for developers interested in VR game development and interactive simulations.

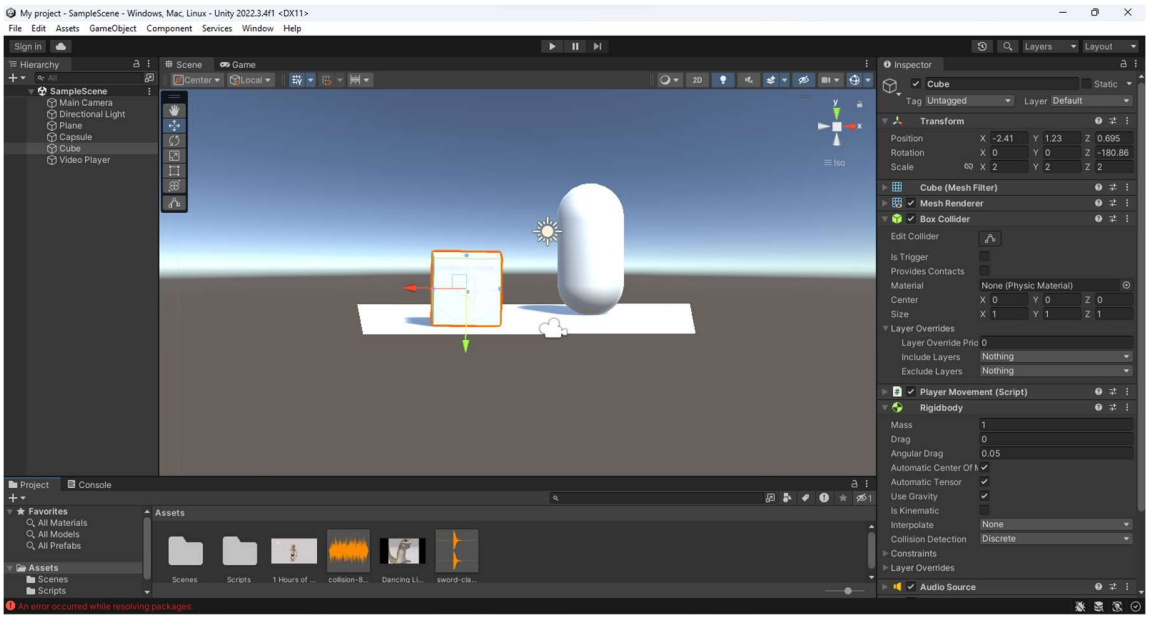
CODE: -

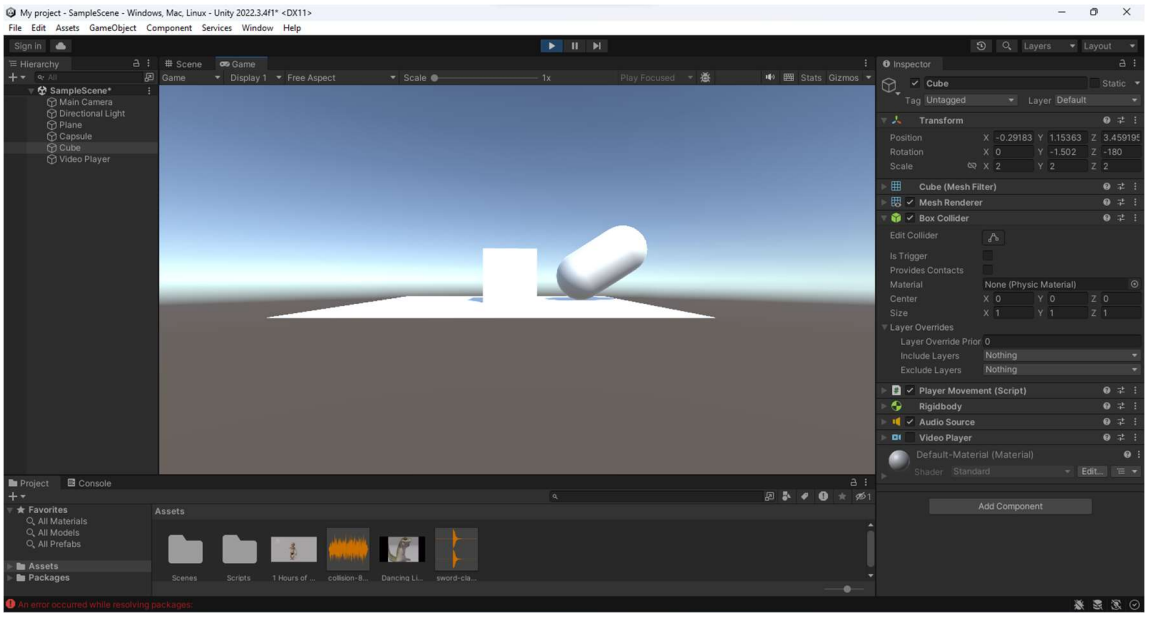
PlayerMovement.cs

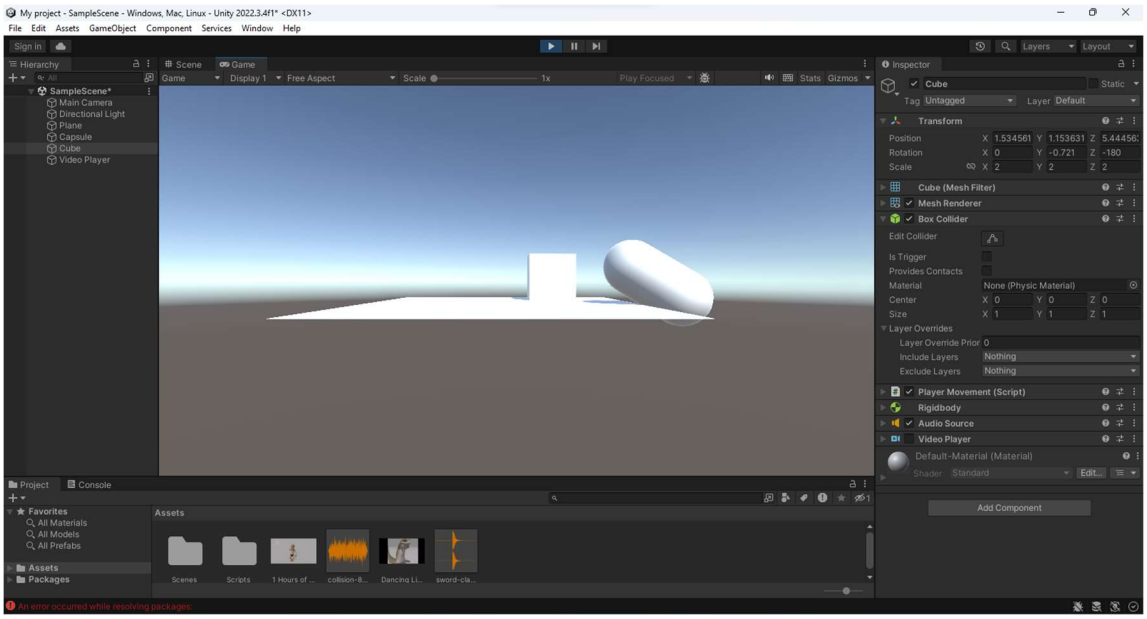
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class PlayerMovement : MonoBehaviour
{
   public float speed = 5.0f;
   public float jumpForce = 5.0f;
   public bool isOnGround = true;
   private float horizontalInput;
   private float forwardInput;
   private Rigidbody playerRb;
   public AudioSource tickSource;
   // Start is called before the first frame update
   void Start()
   {
      playerRb= GetComponent<Rigidbody>();
      tickSource=GetComponent<AudioSource>();
   }
```

```
// Update is called once per frame
  void Update()
  {
     //get player input
     horizontalInput = Input.GetAxis("Horizontal");
     forwardInput = Input. GetAxis("Vertical");

     // Move the player forward
     transform.Translate(Vector3.forward *Time.deltaTime * speed * forwardInput);
     transform.Translate(Vector3.right *  Time.deltaTime * speed * horizontalInput);

     //Let the player jump
     if (Input.GetKeyDown(KeyCode.Space)&& isOnGround)
     {
        playerRb.AddForce(Vector3.up*jumpForce, ForceMode.Impulse);
        isOnGround=false;
     }
  }
  private void OnCollisionEnter (Collision collision)
     {
        tickSource.Play ();
        if (collision.gameObject.CompareTag("Ground"))
        {
           isOnGround = true;
        }
     }
}
```

OUTPUT: -


My project - SampleScene - Windows, Mac, Linux - Unity 2022.3.4f1 <DX11>
File Edit Assets GameObject Component Services Window Help
Hierarchy
SampleScene
Main Camera
Directional Light
Plane
Capsule
Cube
Video Player
Scene
Game
Inspector
Cube
Transform
Cube (Mesh Filter)
Mesh Renderer
Box Collider
Player Movement (Script)
Rigidbody
Audio Source
Project
Console
Favorites
Assets
Scenes
Scripts


My project - SampleScene - Windows, Mac, Linux - Unity 2022.3.4f1* <DX11>
File Edit Assets GameObject Component Services Window Help
Hierarchy
SampleScene*
Main Camera
Directional Light
Plane
Capsule
Cube
Video Player
Scene
Game
Inspector
Cube
Transform
Cube (Mesh Filter)
Mesh Renderer
Box Collider
Player Movement (Script)
Rigidbody
Audio Source
Video Player
Default-Material (Material)
Add Component
Project
Console
Favorites
Assets
Packages



CONCLUSION: -

This Unity project has successfully integrated crucial elements of game development and virtual reality (VR) interaction. By creating a scene featuring a sphere and plane, applying Rigid body components for lifelike physics, implementing materials and Box Colliders for visual and collision properties, and scripting in C# to enable VR controller interaction, a dynamic and immersive VR experience has been achieved. The capacity to grasp and throw the sphere in the VR environment demonstrates Unity's potential for interactive simulations. This project enhances proficiency in physics simulation, scripting, and VR game development, emphasizing the importance of performance optimization for VR applications. It provides a solid foundation for creating captivating VR experiences.