Report Lab4

Name: Poch Sreypov

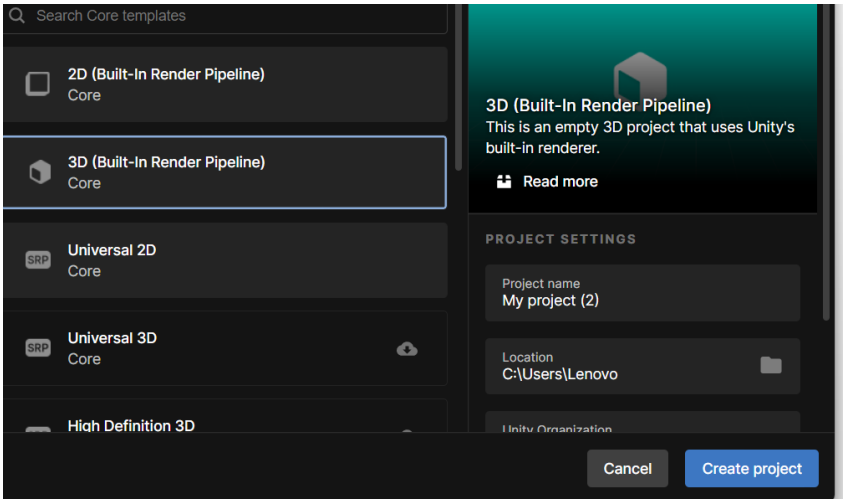
Sambor Sopheakline

I. Introduction

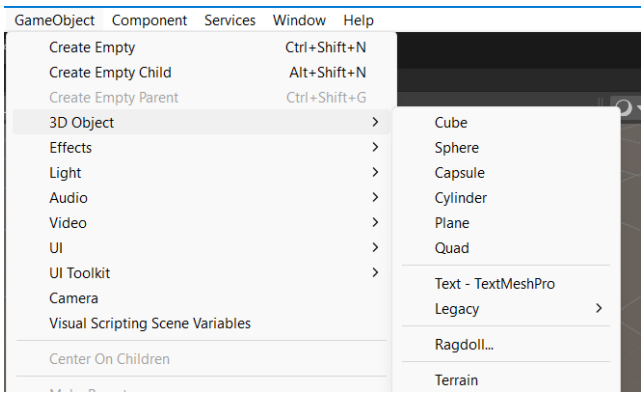
In this project, I will create a simple physics-based game in Unity where a ball rolls through a maze. The primary goal is to demonstrate the use of Unity's Rigidbody component and physics engine features, allowing for realistic movement and interaction with the environment.

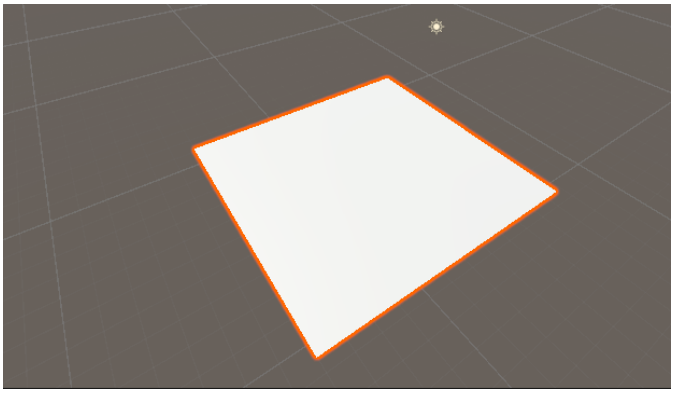
II. Creating the Game

- Step1: Create a new project

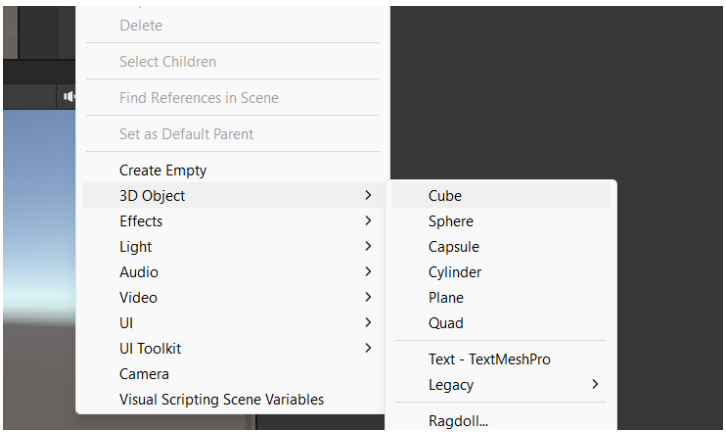


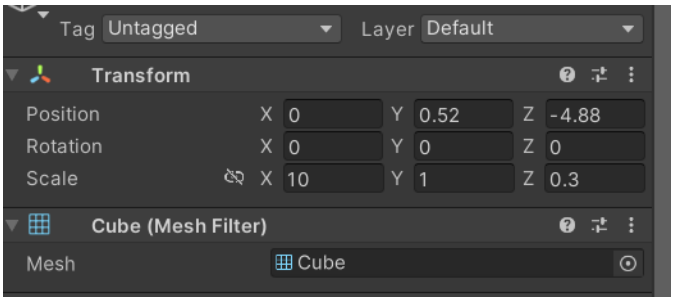
- Step2: In the Hierarchy, create a ground plane: GameObject > 3D Object > Plane to create a ground surface for the maze.

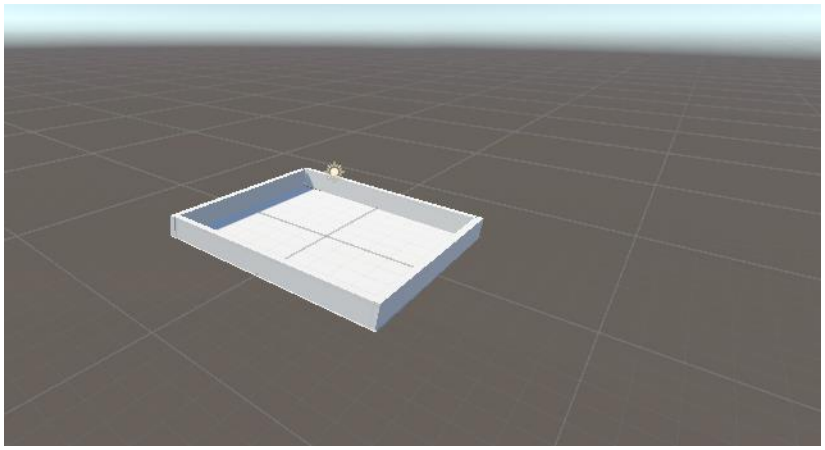


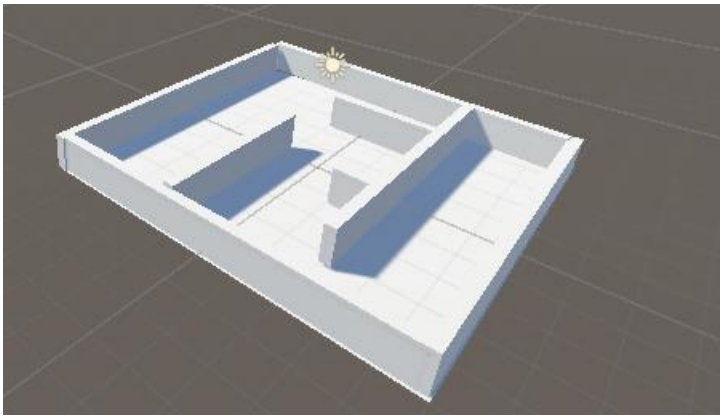


- Step3 : Add walls using cubes: GameObject > 3D Object > Cube. Scale and position them to form a maze layout. Constructs the maze layout, defining the boundaries that the player must navigate around.

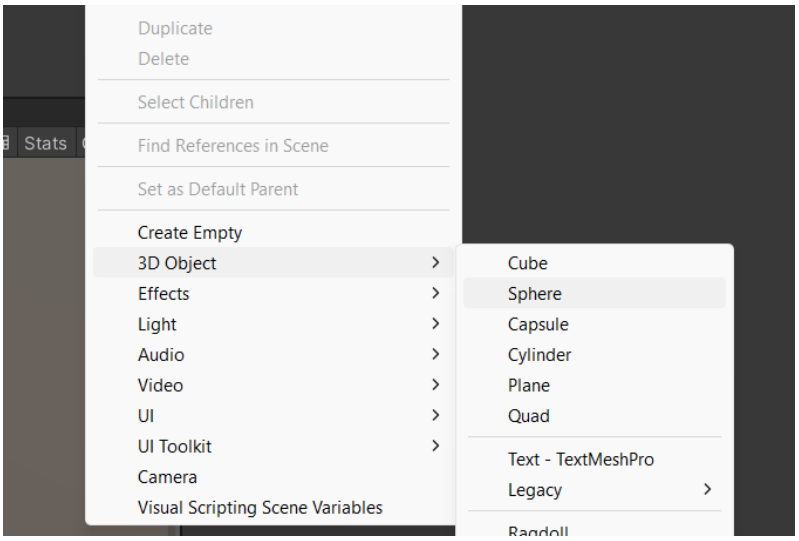


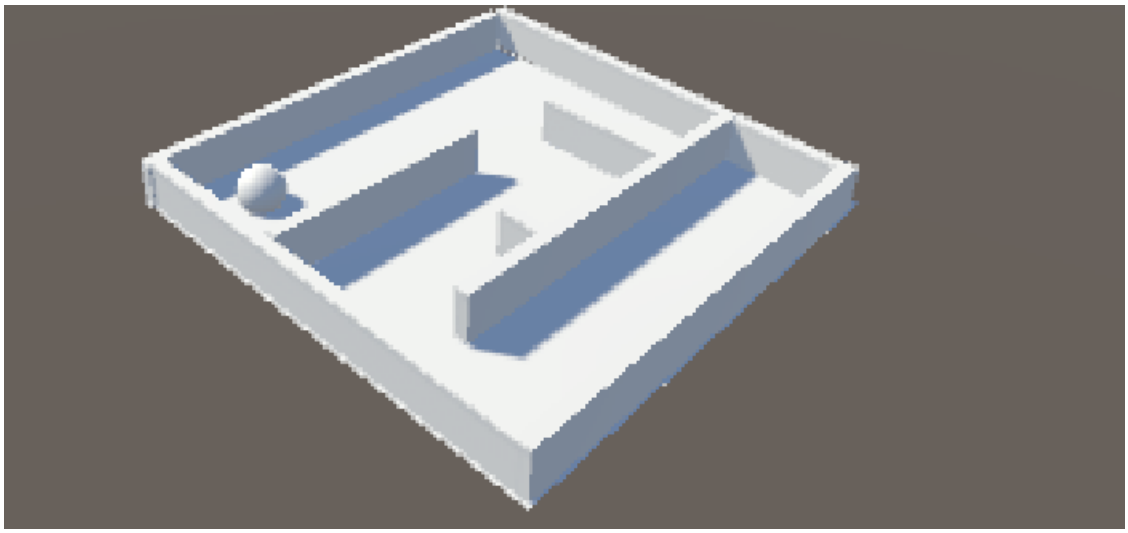




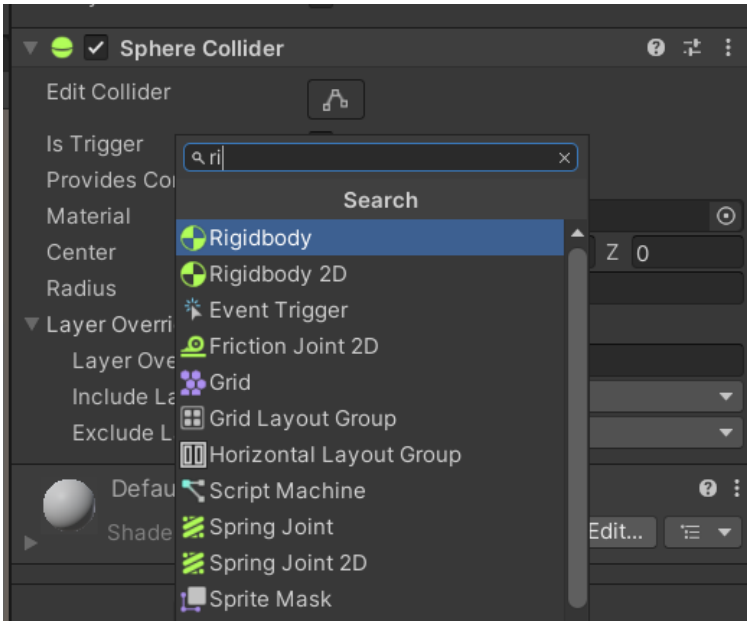


- Step 4: Create a sphere: GameObject > 3D Object > Sphere. Represents the player-controlled object (the ball) that will roll through the maze.

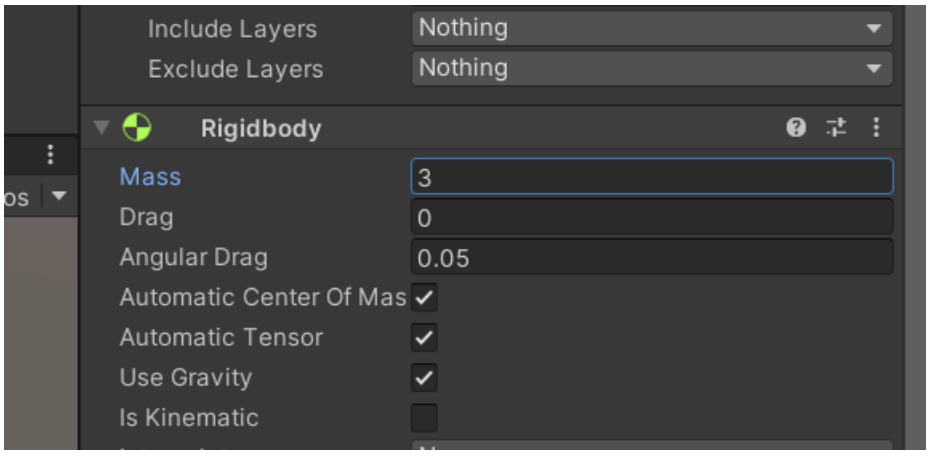




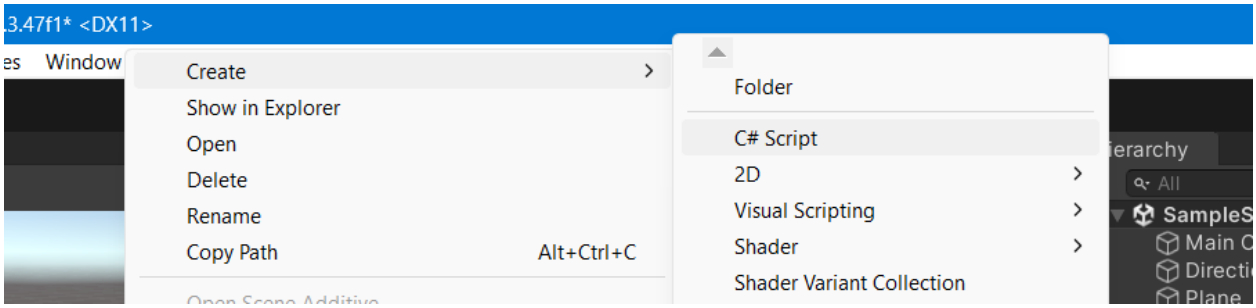
- Step 5: Select the sphere and add a Rigidbody: Inspector > Add Component > Rigidbody. Enables the ball to interact with Unity's physics engine, allowing it to respond to forces and collisions.

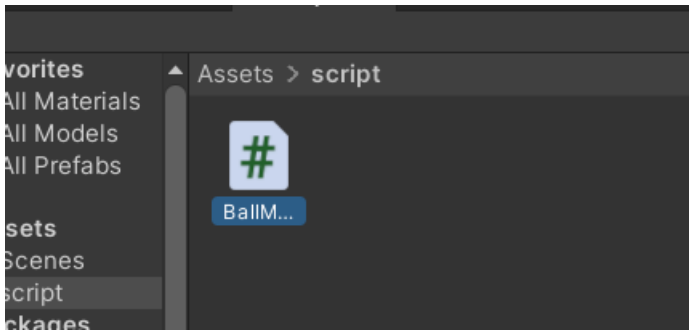


- Step 6: Adjust mass to 3. This Affects how the ball behaves under physics interactions, influencing its movement and how it collides with other objects.

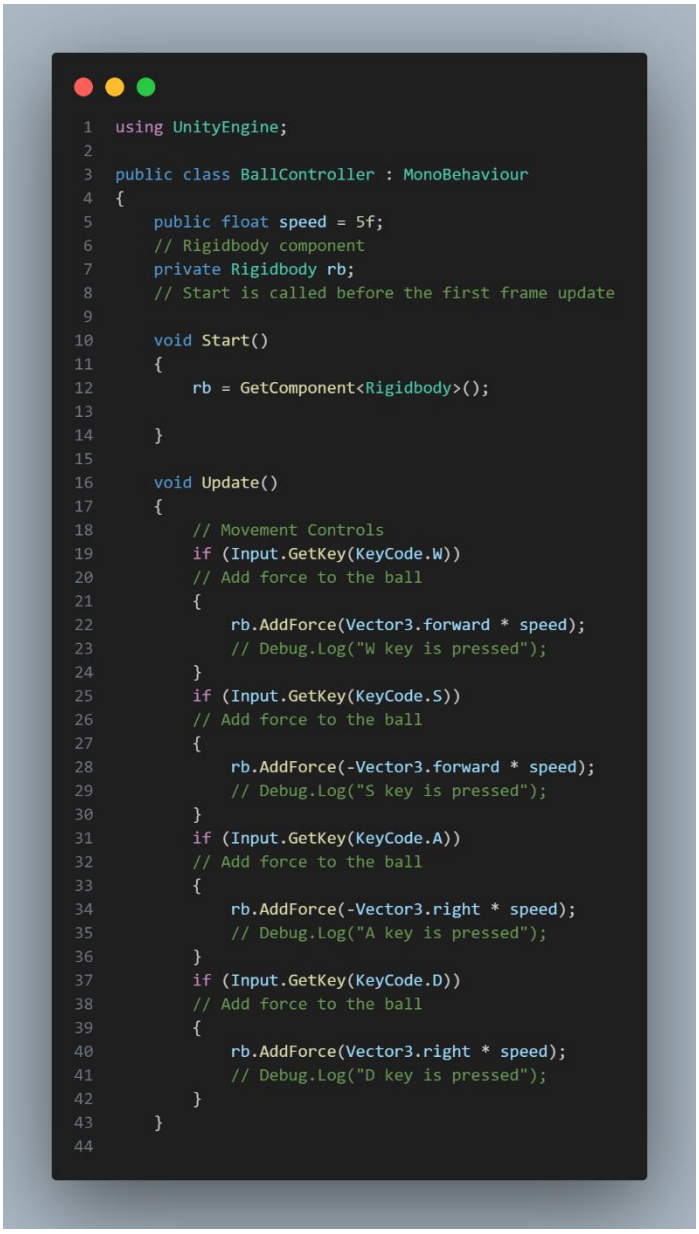


- Step 7: Create Script Right click in a script folder the create a script file.





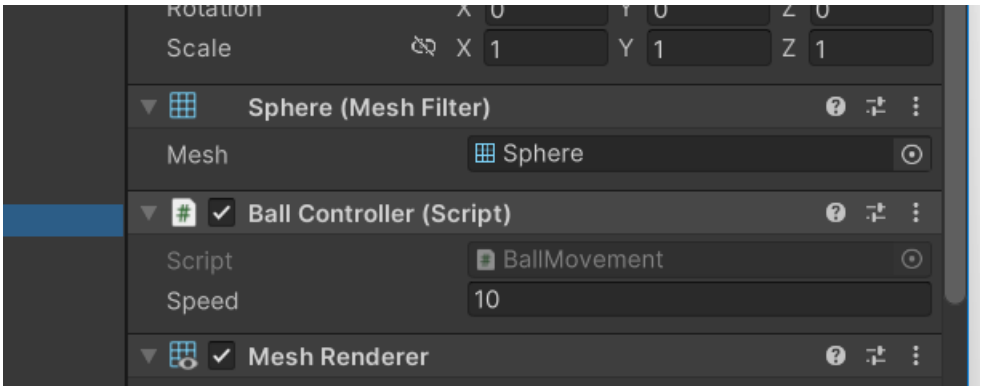
- Step 8 : Write Script by Implements the logic for moving the ball and detecting collisions, making the game functional and engaging.



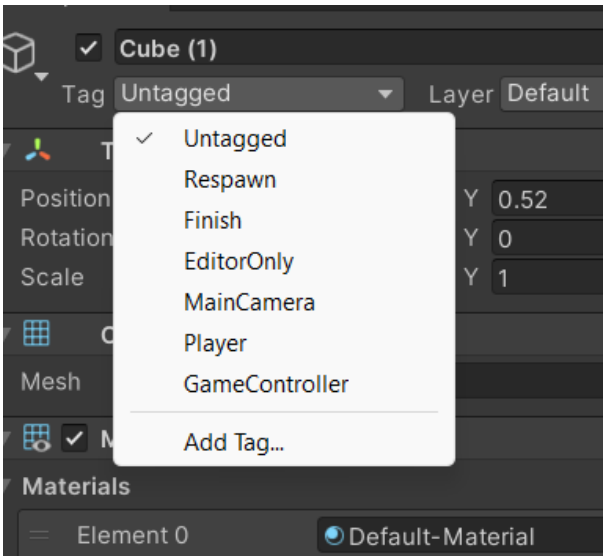
```
using UnityEngine;

public class BallController : MonoBehaviour
{
    public float speed = 5f;
    // Rigidbody component
    private Rigidbody rb;
    // Start is called before the first frame update

    void Start()
    {
        rb = GetComponent<Rigidbody>();

    }

    void Update()
    {
        // Movement Controls
        if (Input.GetKey(KeyCode.W))
        // Add force to the ball
        {
            rb.AddForce(Vector3.forward * speed);
            // Debug.Log("W key is pressed");
        }
        if (Input.GetKey(KeyCode.S))
        // Add force to the ball
        {
            rb.AddForce(-Vector3.forward * speed);
            // Debug.Log("S key is pressed");
        }
        if (Input.GetKey(KeyCode.A))
        // Add force to the ball
        {
            rb.AddForce(-Vector3.right * speed);
            // Debug.Log("A key is pressed");
        }
        if (Input.GetKey(KeyCode.D))
        // Add force to the ball
        {
            rb.AddForce(Vector3.right * speed);
            // Debug.Log("D key is pressed");
        }
    }

```

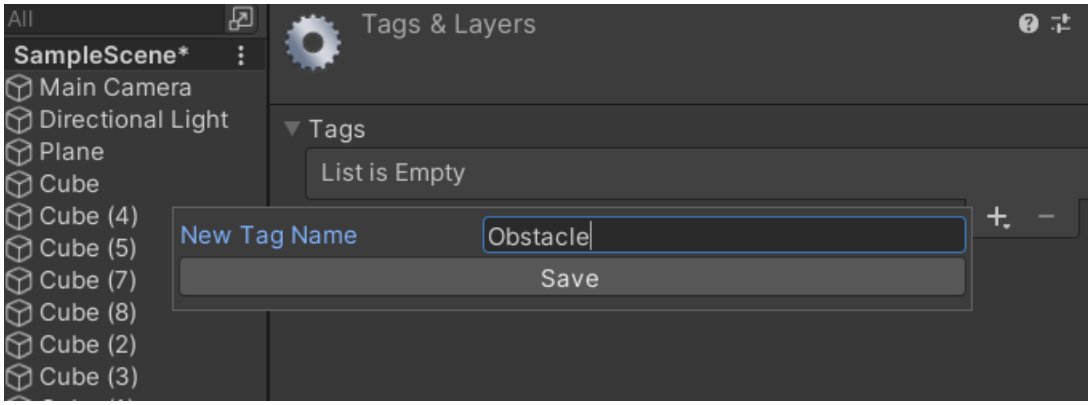
```
1
2       void OnCollisionEnter(Collision collision)
3       // Check for collision with obstacles
4       {
5           if (collision.gameObject.CompareTag("Obstacle"))
6           // Log a message to the console
7           {
8               Debug.Log("Hit an obstacle!");
9               // display hit an obstacle message
10          }
11      }
12  }
```

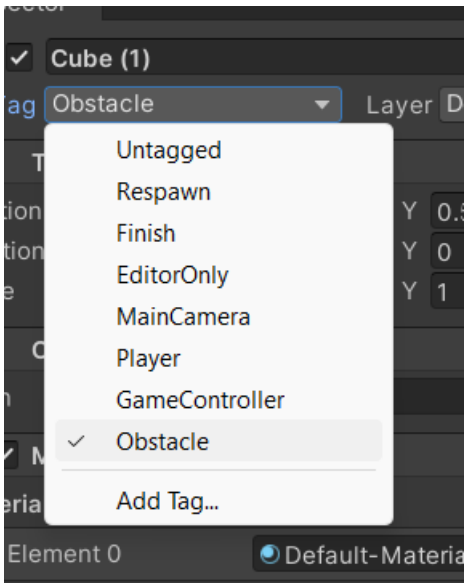
- Step 9: Move the script file to Sphere for attach the file.

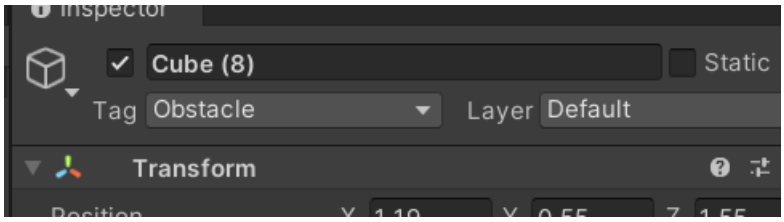


- Step 10: create a tag for the cubes (walls) by clicking on the Tag dropdown, then create a new tag and name it 'Obstacle.' After that, assign this tag to all the walls.




All
SampleScene*
Main Camera
Directional Light
Plane
Cube
Cube (4)
Cube (5)
Cube (7)
Cube (8)
Cube (2)
Cube (3)
Tags & Layers
Tags
List is Empty
New Tag Name
Obstacle
Save


Cube (1)
Obstacle
Layer
Untagged
Respawn
Finish
EditorOnly
MainCamera
Player
GameController
Obstacle
Add Tag...
Element 0


Cube (8)
Static
Tag
Obstacle
Layer
Default
Transform