Lab 2: Working with Physics and Rotation

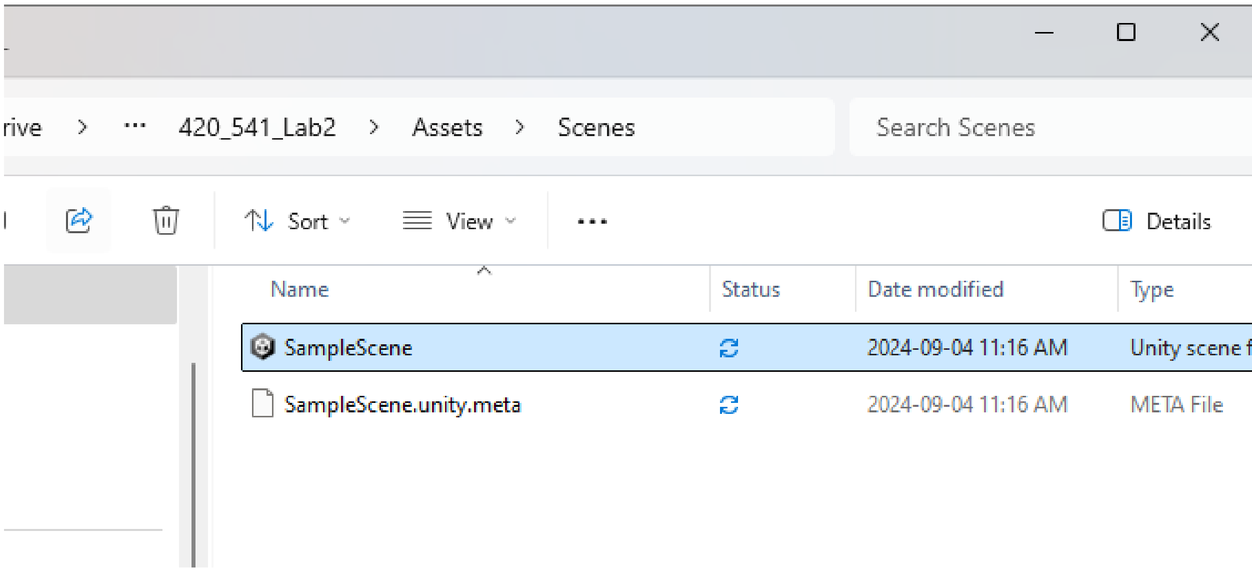
This is the lab consists of fixing an already existing scene. In order to download the project you will have to clone this git repository :

https://github.com/amcnabbbaltar/420_541_Lab2

Setup

- Clone the Git Repository:
    - Clone the project repository to your machine by running in a cmd line :

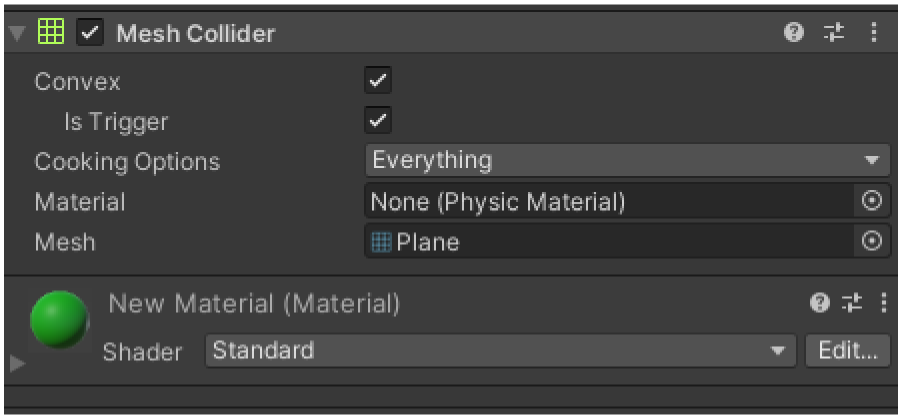
      ```
      git clone https://github.com/amcnabbbaltar/420_541_Lab2
      ```

- Open the project in Unity.
    - In the scene folder double click on the sample scene
    - Click convert in Place once the project is converted.
    - Open the Scene by double clicking on sample Scene.
    - 

Name	Status	Date modified	Type
SampleScene		2024-09-04 11:16 AM	Unity scene f
SampleScene.unity.meta		2024-09-04 11:16 AM	META File

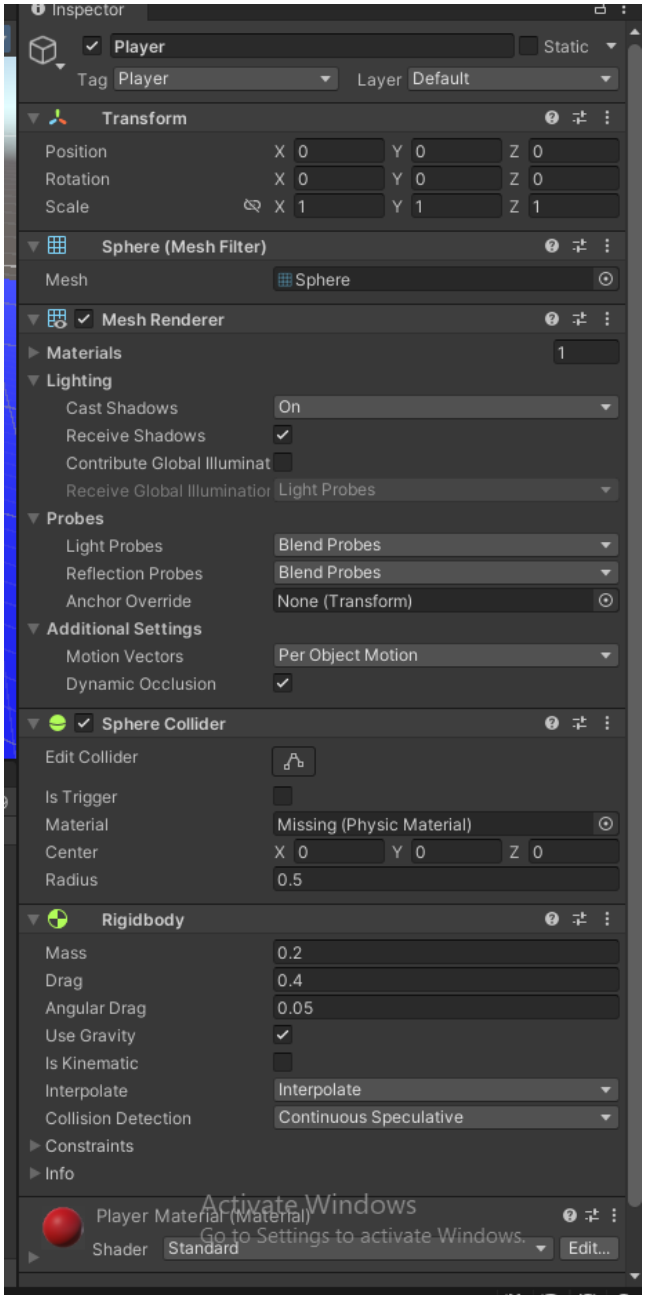
Task 1: Adjust Object Physics Settings (1 pt)

The project should open to a level that looks like this :

1. **Green Goal Object**:
   - Select the green goal object called EndGoal in the scene Hierachy.
   - Make sure that it is set as a **static** and a **Trigger** object:
     - In the Inspector, check that the object is **not** using Rigidbody.
     - Add a **Collider** (if not already present) and check the box for **Is Trigger**.
     - 

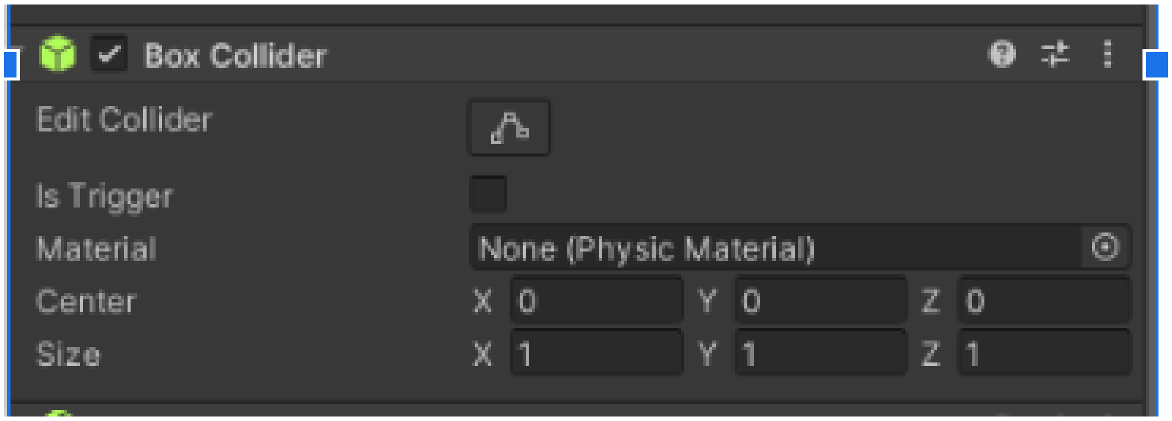
       Mesh Collider
       Convex ✓
       Is Trigger ✓
       Cooking Options: Everything
       Material: None (Physic Material)
       Mesh: Plane
       New Material (Material)
       Shader: Standard

2. **Red Player Object**:
   - Select the player object (the red ball) .
   - Ensure it is a **dynamic** object:
     - Add a **Rigidbody** component (if not already present).
     - Make sure the **Use Gravity** option is checked.
     - Set the **Is Kinematic** property to **false**.

3. 

```
Inspector
✓ Player                                  Static
Tag Player                 Layer Default

Transform
  Position          X 0    Y 0    Z 0
  Rotation          X 0    Y 0    Z 0
  Scale             X 1    Y 1    Z 1

Sphere (Mesh Filter)
  Mesh              Sphere

✓ Mesh Renderer
  Materials                                 1
  Lighting
    Cast Shadows                 On
    Receive Shadows              ✓
    Contribute Global Illuminat  ☐
    Receive Global Illuminatior  Light Probes
  Probes
    Light Probes                 Blend Probes
    Reflection Probes            Blend Probes
    Anchor Override              None (Transform)
  Additional Settings
    Motion Vectors               Per Object Motion
    Dynamic Occlusion            ✓

✓ Sphere Collider
  Edit Collider
  Is Trigger        ☐
  Material          Missing (Physic Material)
  Center            X 0    Y 0    Z 0
  Radius            0.5

Rigidbody
  Mass              0.2
  Drag              0.4
  Angular Drag      0.05
  Use Gravity       ✓
  Is Kinematic      ☐
  Interpolate       Interpolate
  Collision Detection  Continuous Speculative
  Constraints
  Info

Player Material (Material)
  Shader   Standard                         Edit...
```

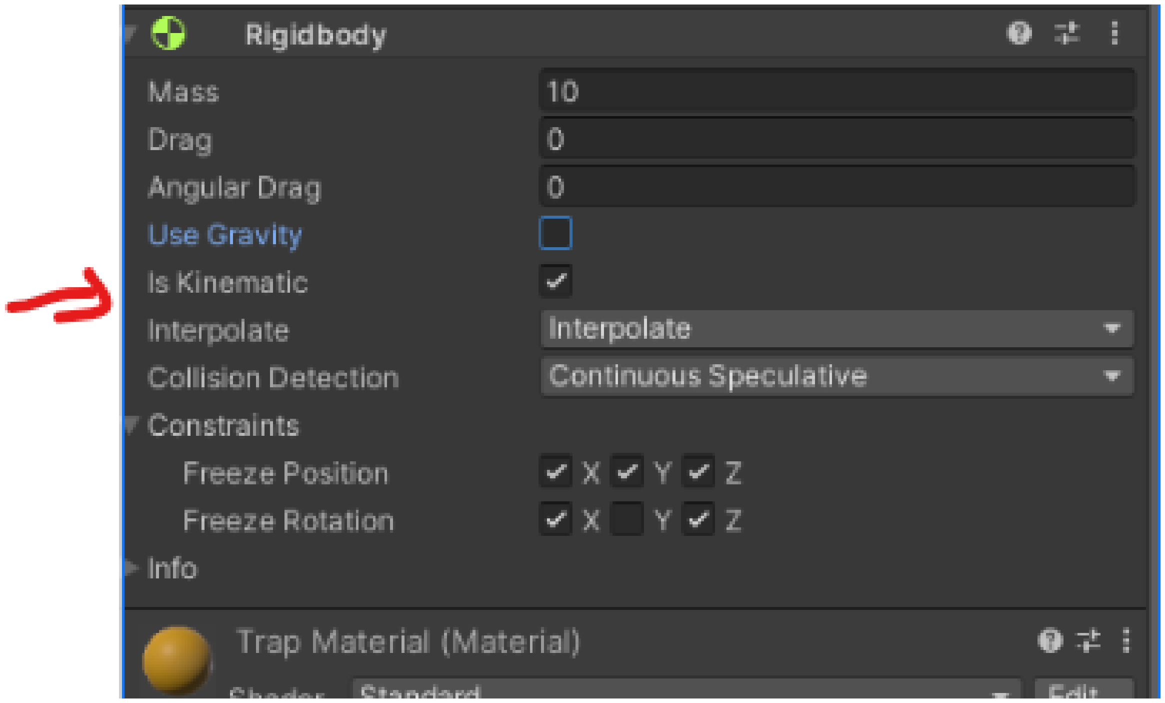
4. **Yellow Trap Object**:
   - Select the yellow trap object.
   - Ensure it is **NOT** a **Trigger** by making sure the IsTrigger Option is **NOT** checked.



Box Collider

Edit Collider	
Is Trigger	☐
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Size	X 1 Y 1 Z 1

- Ensure it is a **kinematic** object:
  - Add a **Rigidbody** component.
  - Set **Is Kinematic** to **true** and **Use Gravity** to **false**.



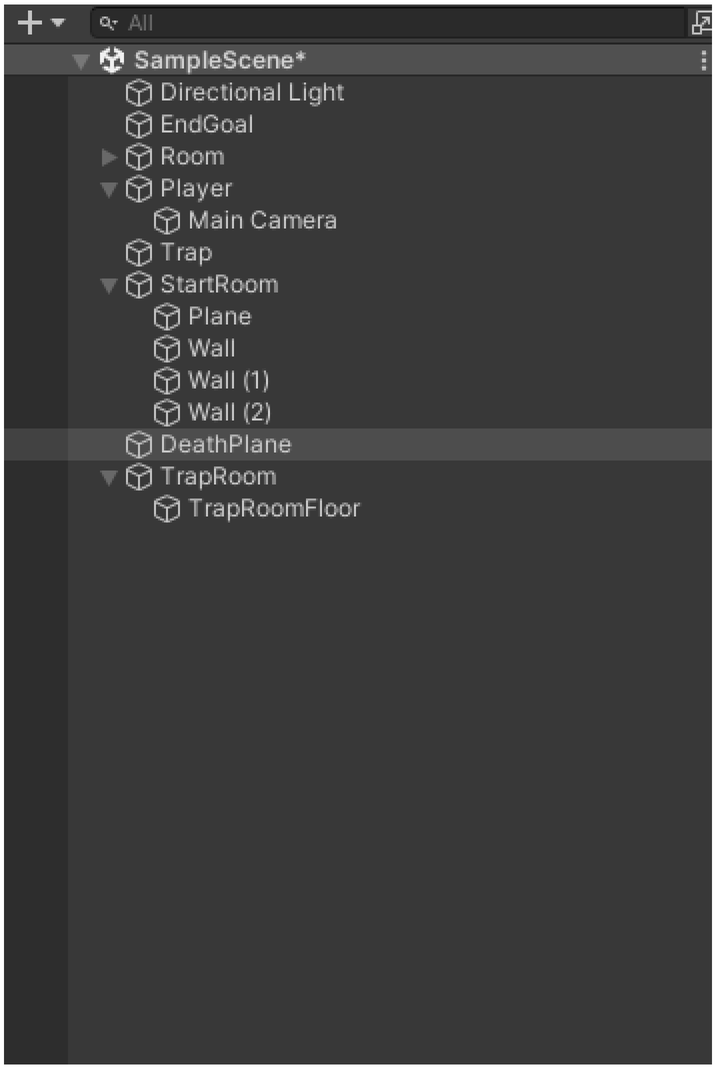
Rigidbody

Mass	10
Drag	0
Angular Drag	0
Use Gravity	☐
Is Kinematic	✓
Interpolate	Interpolate
Collision Detection	Continuous Speculative
Constraints	
Freeze Position	✓ X ✓ Y ✓ Z
Freeze Rotation	✓ X ☐ Y ✓ Z
Info	

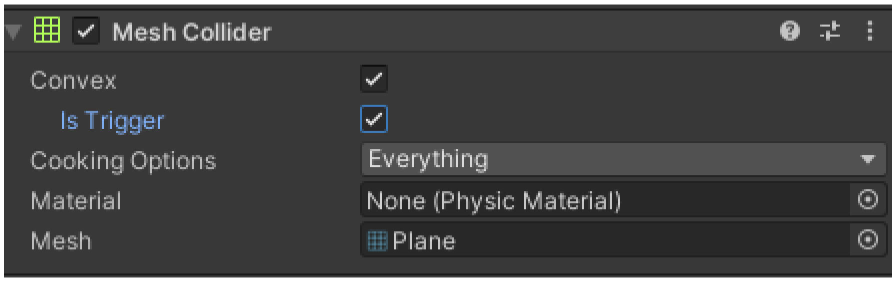
Trap Material (Material)

5.Setup the death Plane

- Select the Blue object called Death plane in the scene Hierachy.



- Make sure that it is set as a **static** and a **Trigger** object:
  - In the Inspector, check that the object is **not** using Rigidbody.
  - Add a **Collider** (if not already present) and check the box for **Is Trigger**.


Mesh Collider
Convex ✓
Is Trigger ✓
Cooking Options: Everything
Material: None (Physic Material)
Mesh: Plane

Task 2: Create Player Movement Script (1 pt)

1. Create a new C# script named PlayerController.
2. Attach this script to the red player object (the red ball).
3. Open the script and use the following code to move the player using the Rigidbody based on user input:

```
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    public float moveSpeed = 10f;
    private Rigidbody rb;

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

    void FixedUpdate() // Physics-related operations should go
here
    {
        // Get input for horizontal and vertical axes
        float moveX = Input.GetAxis("Horizontal");
        float moveZ = Input.GetAxis("Vertical");

        // Calculate the movement direction based on the player's
forward and right vectors
        Vector3 movement = (transform.right * moveX) +
(transform.forward * moveZ);

        // Apply movement to the rigidbody
        rb.AddForce(movement * moveSpeed, ForceMode.Force);
    }
}
```

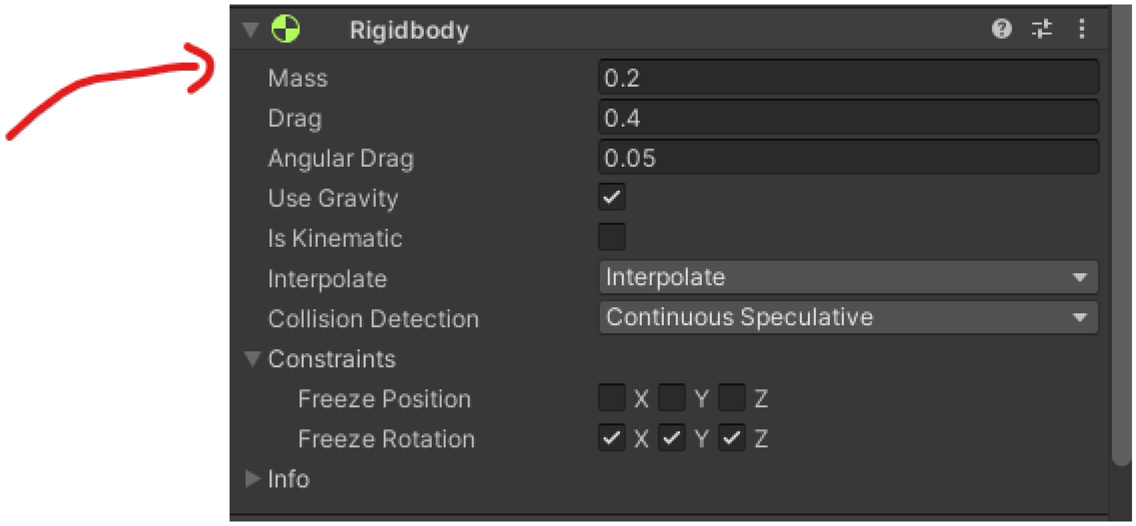
4. Save the script and test the movement by playing the scene.

5. What happens when you try to move the ball ? :

With the use of the WASD keys or the arrow keys, the ball is able to navigate around the map, on a horizontal and vertical axis.

6. Lets fix the ball movement. Right now the weight of the ball is too small for the force we apply on it. So the acceleration that the physics engine impart to the ball is very high so lets make our Player weight bigger

- Change the mass parameter in the RigidBody Component



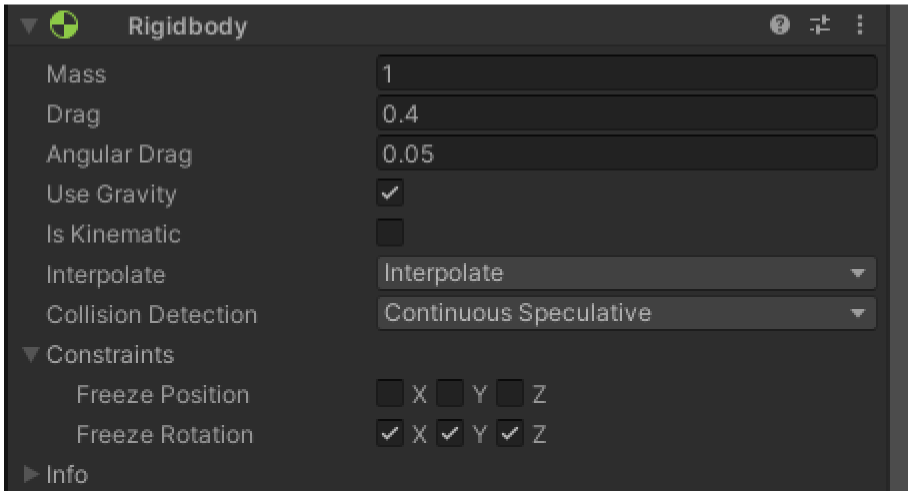
Rigidbody

Mass	0.2
Drag	0.4
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	☐
Interpolate	Interpolate
Collision Detection	Continuous Speculative
Constraints	
Freeze Position	☐ X ☐ Y ☐ Z
Freeze Rotation	✓ X ✓ Y ✓ Z
Info	

-

Let set it to 1 kg

Rigidbody.MoveRotation



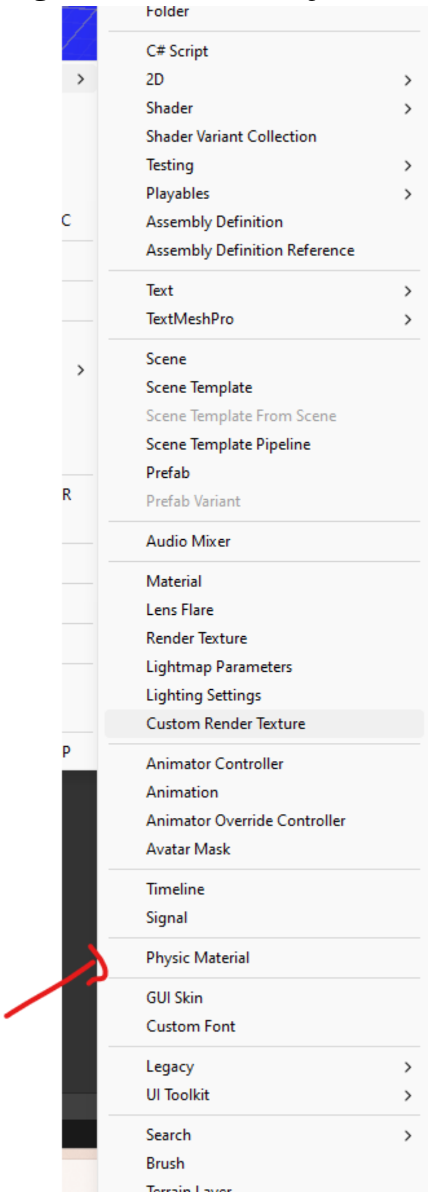
Rigidbody

Mass	1
Drag	0.4
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	☐
Interpolate	Interpolate
Collision Detection	Continuous Speculative
Constraints	
Freeze Position	☐ X ☐ Y ☐ Z
Freeze Rotation	✓ X ✓ Y ✓ Z
Info	

In the inspector you can change the movespeed if you want to move faster

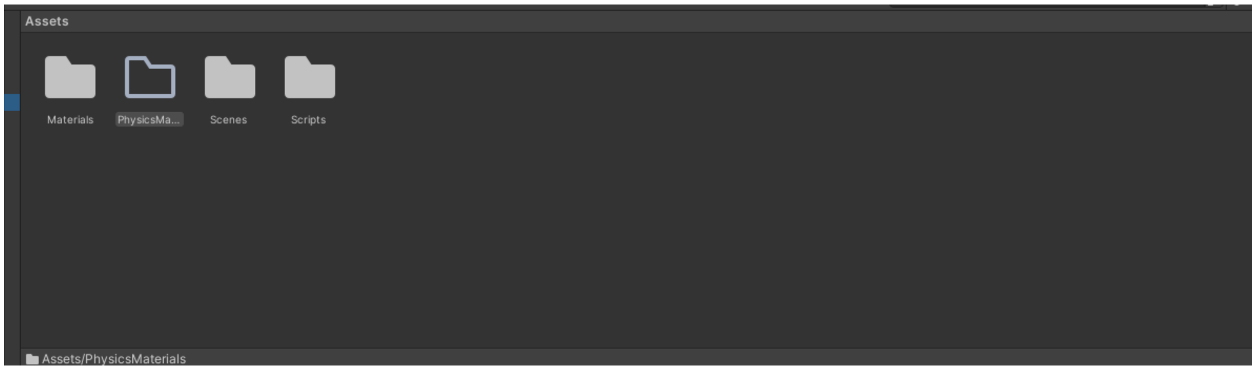
Task 3: Add a Physics Material to the Ball (1 pt)

1. Create a **Physics Material**:
    - Right-click in the **Project** window and select Create > Physics Material.

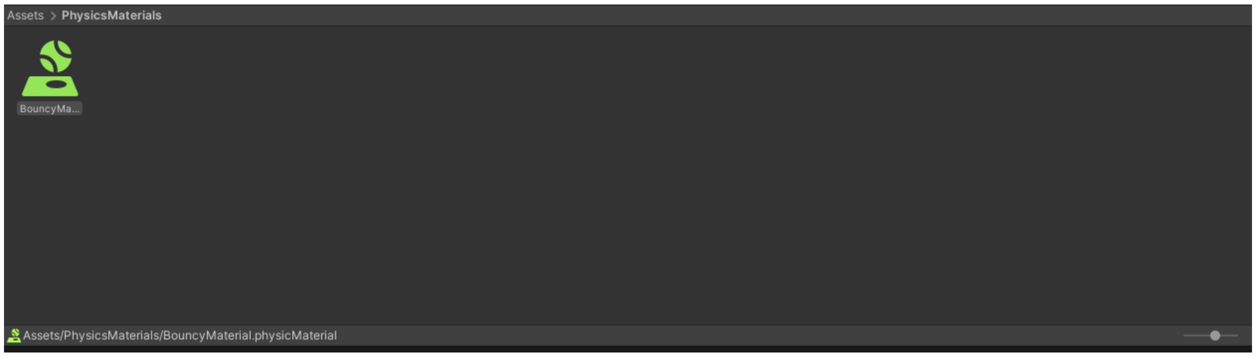


    -
    - Name it BouncyMaterial.

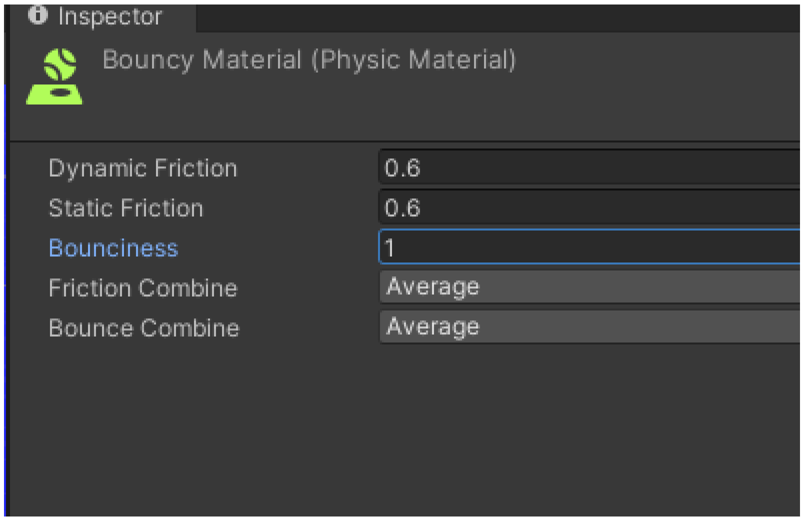
- Add it to a new folder called PhysicsMaterial



2. Double click on BouncyMaterial and

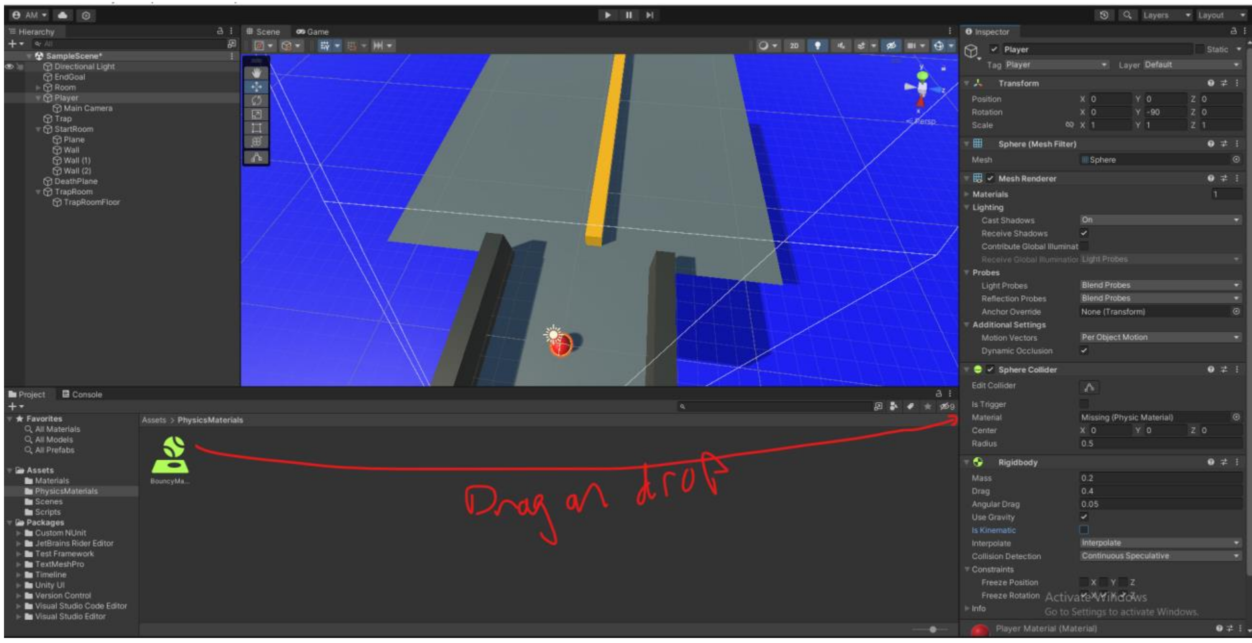

Assets > PhysicsMaterials
BouncyMa...
Assets/PhysicsMaterials/BouncyMaterial.physicMaterial

3. Set the **Bounciness** value to 1 to make the ball extra bouncy.



Dynamic Friction	0.6
Static Friction	0.6
Bounciness	1
Friction Combine	Average
Bounce Combine	Average

4.
5. Assign the BouncyMaterial to the red ball’s **Collider** component.

- 

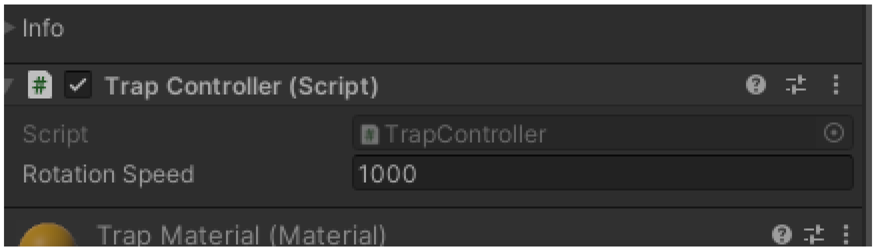
Player
Tag Player Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y -90 Z 0
Scale X 1 Y 1 Z 1
Sphere (Mesh Filter)
Mesh Sphere
Sphere Collider
Is Trigger
Material Missing (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5
Rigidbody
Mass 0.2
Drag 0.4
Angular Drag 0.05
Use Gravity ✓
Is Kinematic
Interpolate Interpolate
Collision Detection Continuous Speculative

Task 4: Rotate the Yellow Trap (1 pt)

1. **Initial Attempt (with Transform.Rotate)**:
   - Create a new C# script called TrapController.
   - Attach the script to the yellow trap object.
   - Use the following code to try rotating the trap with Transform.Rotate:

```
using UnityEngine;

public class TrapController : MonoBehaviour
{
    public float rotationSpeed = 100f;

    void Update()
    {
        transform.Rotate(0, rotationSpeed * Time.deltaTime, 0);
    }
}
```

In the inspector set the rotation speed to 1000



Script: TrapController
Rotation Speed: 1000

Now move your player to the middle of the trap what do you notice ?

As the trap moves very rapidly, when bringing the player using the movement keys to the middle of the trap, It gets stuck and is unable to get out of the rotating trap, skipping the collision that would kick the ball around instead.

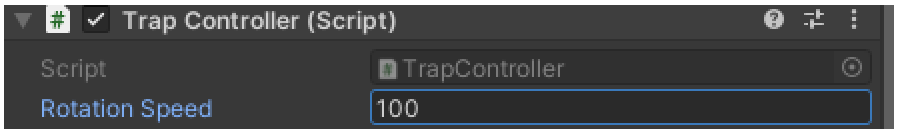
This issue is due to the fact that changing the rotation an object through it's transform is a discrete movement . It essentially teleport it to your new angle so if the changes is big enough it will not collide with an object in the way.

One way to correct for that is to use Continuous Collision Detection but it is very heavy for the CPU so we will use another method that allows collision to be properly updated

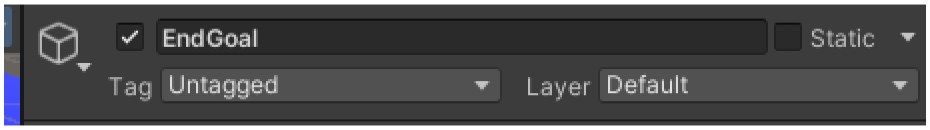
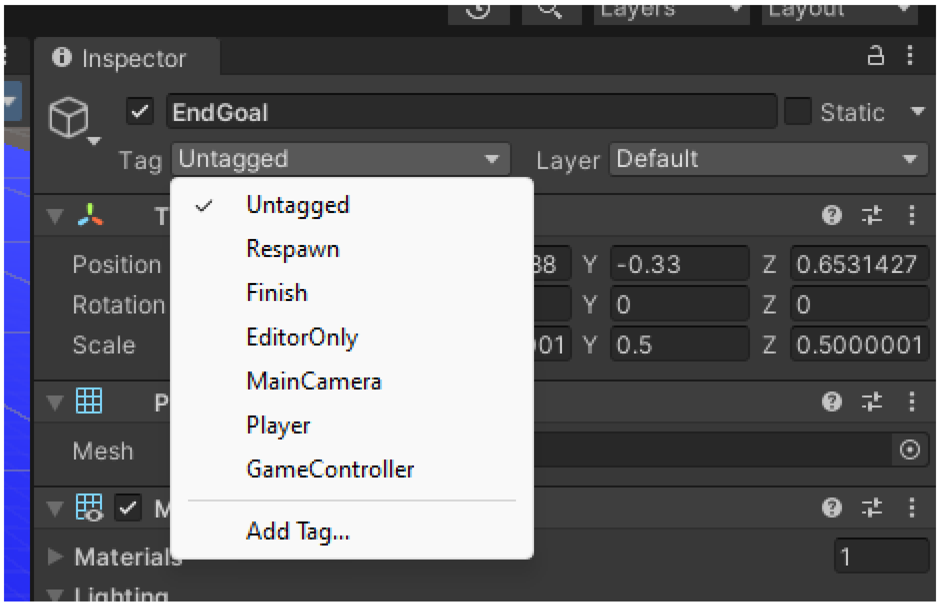
2. **Fix Rotation (Rigidbody.MoveRotation)**:
    - Notice clipping issues with the initial attempt.
    - In order to make sure that during a rotation the physics calculation are done properly we will use the RigidBody.MoveRotation
    - Modify the code to rotate the object using **Rigidbody.MoveRotation**:

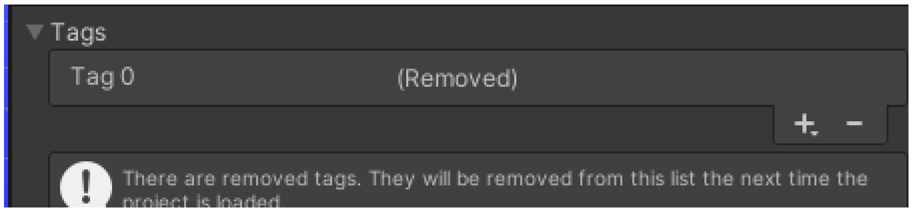
```
using UnityEngine;

public class TrapController : MonoBehaviour
{
    public float rotationSpeed = 100f;
    private Rigidbody rb;

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

    void FixedUpdate()
```

```
        {
            Quaternion deltaRotation = Quaternion.Euler(0,
    rotationSpeed * Time.fixedDeltaTime, 0);
            rb.MoveRotation(rb.rotation * deltaRotation);
        }
    }
```

- **Set the rotation speed back down to 100 now.**
- 
  Trap Controller (Script)
  Script: TrapController
  Rotation Speed: 100

Task 5: Detect Player Reaching the Goal (1 pt)

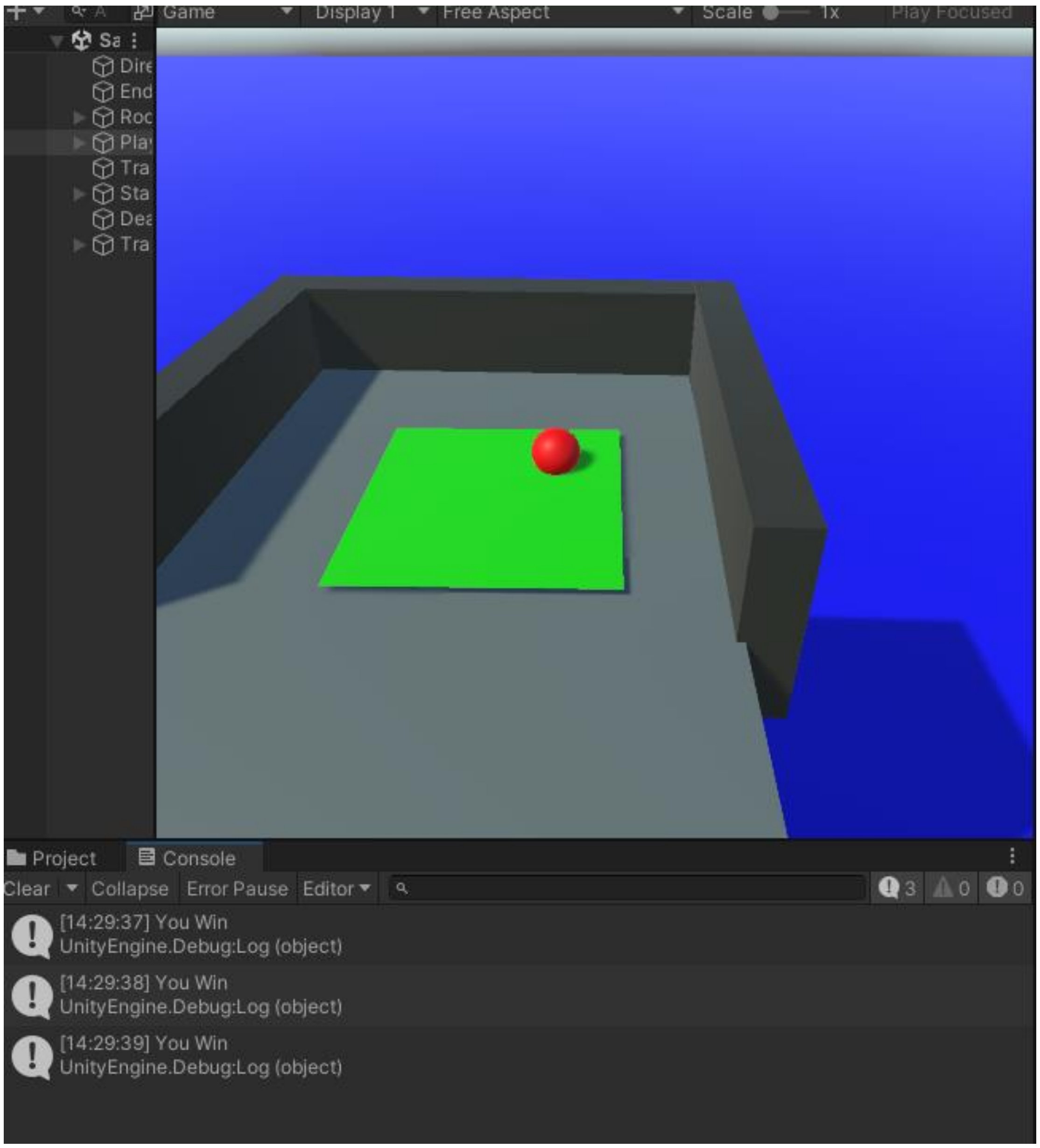
1. Add a tag named Goal to the green goal object:
   - Select the green goal object, go to the **Inspector**, and assign the Goal tag.
   - 
     EndGoal
     Static
     Tag: Untagged
     Layer: Default
   - If one doesnt exist create one. With the AddTag option
   - 

     Position: Y -0.33, Z 0.6531427
     Rotation: Y 0, Z 0
     Scale: Y 0.5, Z 0.5000001
   - 
     Tags
     Tag 0 (Removed)
     There are removed tags. They will be removed from this list the next time the project is loaded.
   - Press the + button to add a tag and Name it Goal

2. Modify the PlayerController script to detect collisions by adding this method to the Player Controller Script:

```
void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Goal"))
    {
        Debug.Log("You Win");
    }
}
```

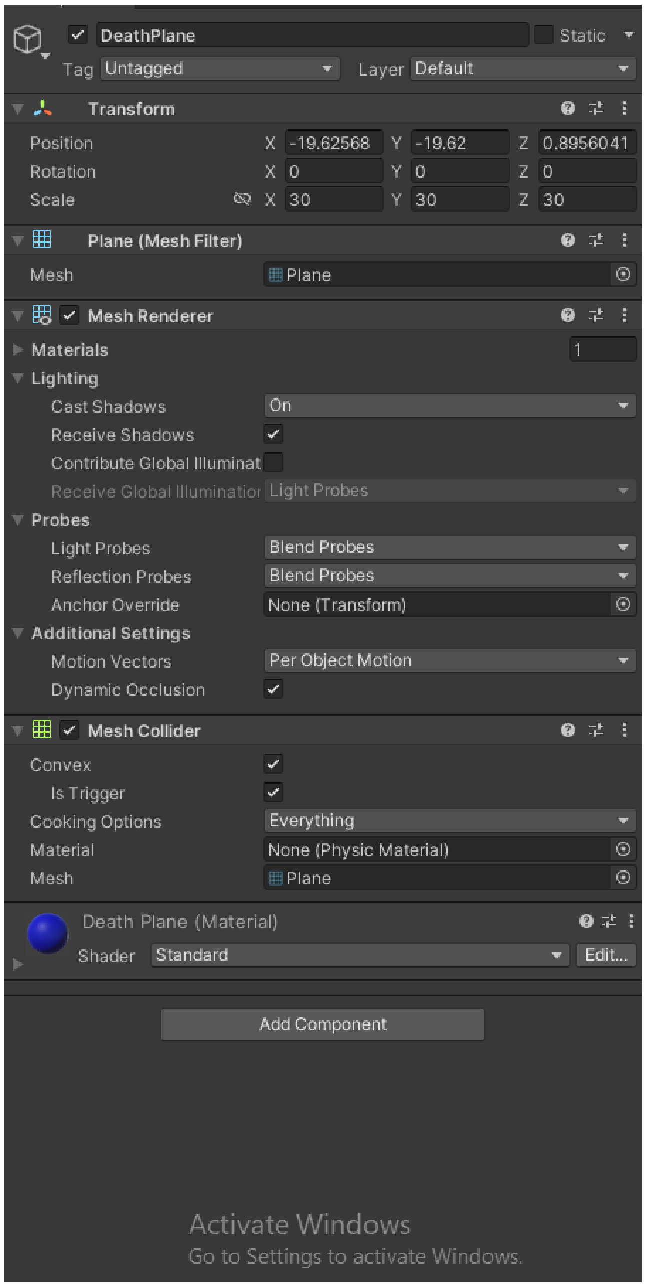
3. Test to ensure the message "You Win" appears in the console when the player touches the green goal.
4. **Add a screenshot here of the debug :**



```
[14:29:37] You Win
UnityEngine.Debug:Log (object)
[14:29:38] You Win
UnityEngine.Debug:Log (object)
[14:29:39] You Win
UnityEngine.Debug:Log (object)
```

Task 6: Restart Game on Death (1 pt)

1. Add a tag named DeathPlane to the blue death plane object.



DeathPlane ✓ Static
Tag Untagged Layer Default

Transform

Position	X -19.62568	Y -19.62	Z 0.8956041
Rotation	X 0	Y 0	Z 0
Scale	X 30	Y 30	Z 30

Plane (Mesh Filter)

Mesh: Plane

Mesh Renderer ✓

Materials	1
Lighting	
Cast Shadows	On
Receive Shadows	✓
Contribute Global Illuminat	
Receive Global Illuminatior	Light Probes
Probes	
Light Probes	Blend Probes
Reflection Probes	Blend Probes
Anchor Override	None (Transform)
Additional Settings	
Motion Vectors	Per Object Motion
Dynamic Occlusion	✓

Mesh Collider ✓

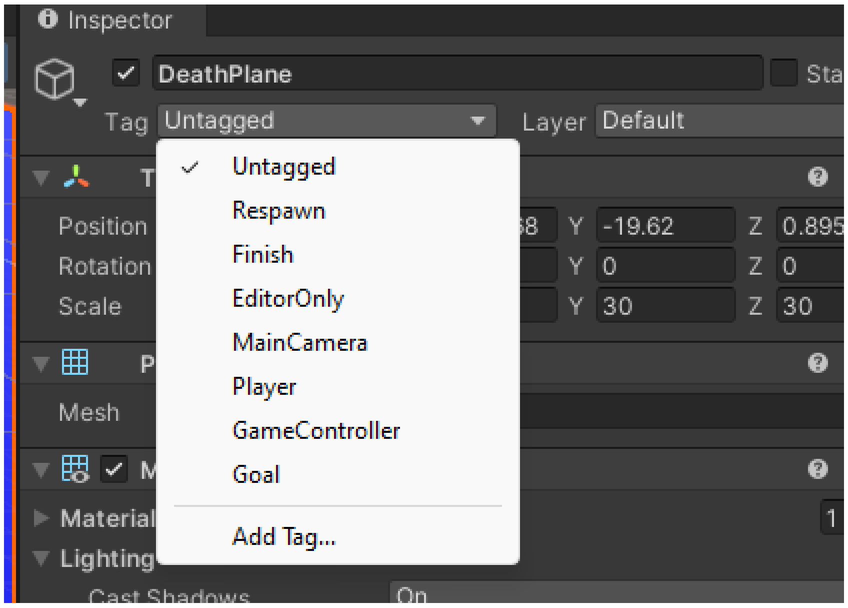
Convex	✓
Is Trigger	✓
Cooking Options	Everything
Material	None (Physic Material)
Mesh	Plane

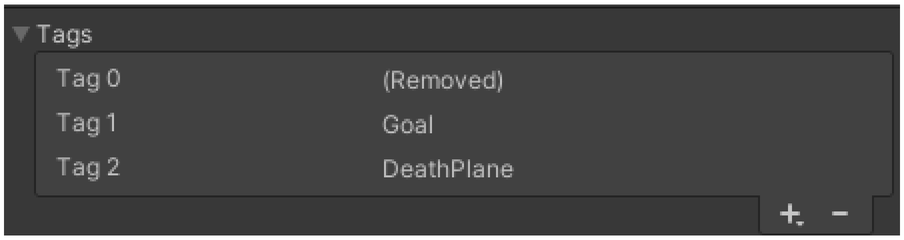
Death Plane (Material)
Shader Standard Edit...

Add Component

Activate Windows
Go to Settings to activate Windows.

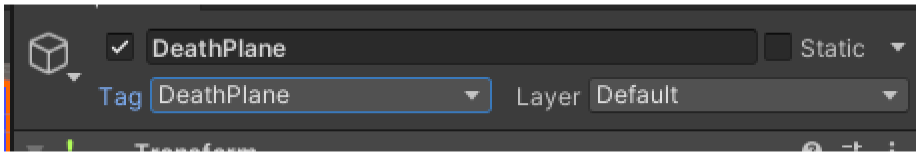
2.

3. 

4. 

Tags

Tag 0	(Removed)
Tag 1	Goal
Tag 2	DeathPlane

5. 

DeathPlane

Static

Tag: DeathPlane

Layer: Default

6. Then Modify the PlayerController script to teleport the player back to its original position when it touches the death plane by adding these lines of code before the FixedUpdate loop:

```
private Vector3 startPosition;

void Start()
{
    rb = GetComponent<Rigidbody>();
    startPosition = transform.position;
}

void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Goal"))
    {
            Debug.Log("You Win");
        }
        else if (other.CompareTag("DeathPlane"))
        {
            rb.position = startPosition;
        }
    }
```

3. Test to ensure the player is reset to the original position when touching the death plane.

How to submit your project :
1) Add the URL to your github repo here :
____

2) Make sure your project is public on github.
3) Write your answer in this document and send this document through LÉA.