

Lab 2: Working with Physics and Rotation

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

<https://classroom.github.com/a/kmBDSydU>

Setup

- Accept the assignment

itations/62c8c941785c38f8c989b358a14cc248

GitHub Classroom

GitHub Education



amcnabbaltar-vanier-classroom-36a36b

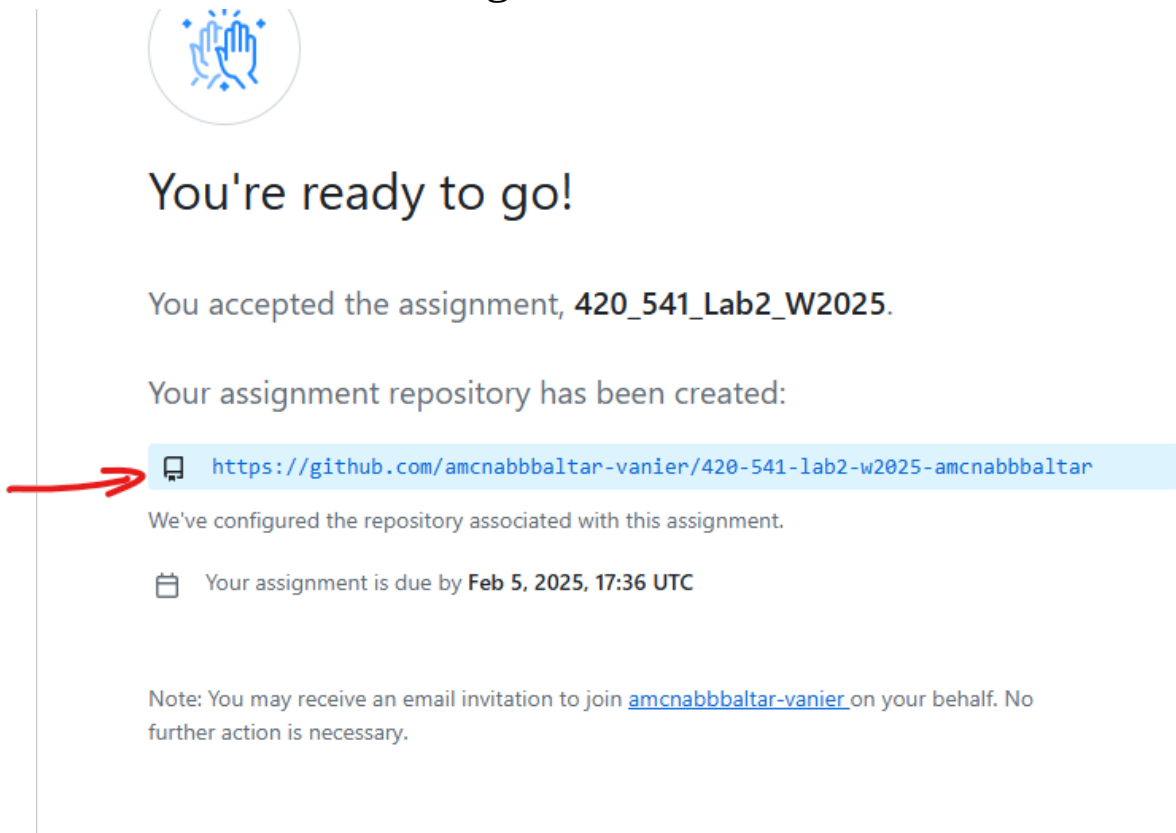
Accept the assignment —

420_541_Lab2_W2025

Once you accept this assignment, you will be granted access to the `420-541-lab2-w2025-amcnabbaltar` repository in the `amcnabbaltar-vanier` organization on GitHub.

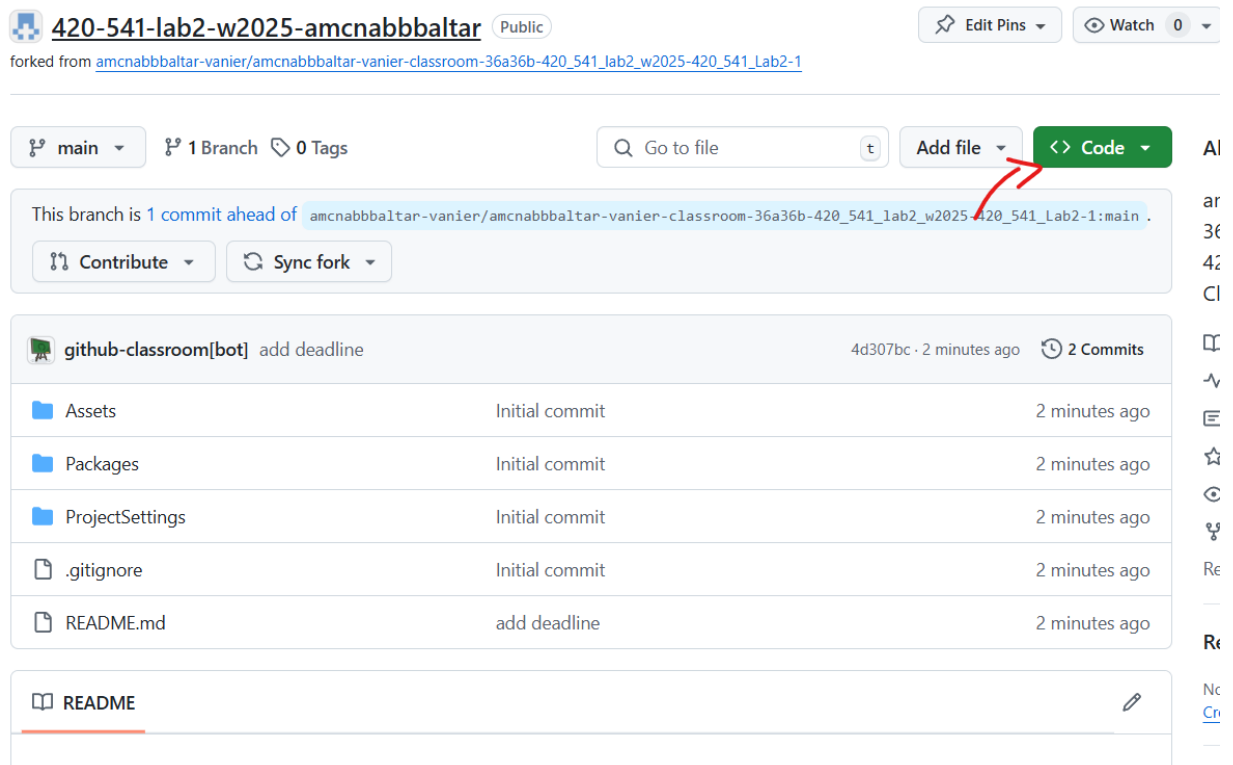
Accept this assignment

- Click on the created assignment link



The image shows a GitHub Classroom assignment confirmation page. At the top, there is a blue icon of two hands clapping. Below it, the text reads "You're ready to go!". This is followed by "You accepted the assignment, 420_541_Lab2_W2025." and "Your assignment repository has been created:". A red arrow points to a blue box containing the repository URL: <https://github.com/amcnabbbaltar-vanier/420-541-lab2-w2025-amcnabbbaltar>. Below this, it says "We've configured the repository associated with this assignment." and "Your assignment is due by Feb 5, 2025, 17:36 UTC". A note at the bottom states: "Note: You may receive an email invitation to join [amcnabbbaltar-vanier](#) on your behalf. No further action is necessary."

-
- Then clone the created repo with github desktop.



The image shows the GitHub repository page for "420-541-lab2-w2025-amcnabbbaltar". The repository is public and was forked from "amcnabbbaltar-vanier/amcnabbbaltar-vanier-classroom-36a36b-420_541_lab2_w2025-420_541_Lab2-1". The page shows the main branch with 1 commit ahead of the parent repository. A red arrow points to the "Code" button. Below the repository information, there is a table of commits. The latest commit is by "github-classroom[bot]" with the message "add deadline", committed 2 minutes ago. The commit includes files: Assets, Packages, ProjectSettings, .gitignore, and README.md. The README.md file is highlighted in red.

Commit Message	Commit Hash	Time Ago	Commits
github-classroom[bot] add deadline	4d307bc	2 minutes ago	2 Commits
Assets	Initial commit	2 minutes ago	
Packages	Initial commit	2 minutes ago	
ProjectSettings	Initial commit	2 minutes ago	
.gitignore	Initial commit	2 minutes ago	
README.md	add deadline	2 minutes ago	

Go to file

Add file

<> Code

Local

Codespaces

Clone

HTTPSSSHGitHub CLI

https://github.com/amcnabbbaltar-vanier/420-541

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

3 minutes ago

About

amcnabbbaltar-vanier-36a36b-420_541_lab2_420_541_Lab2-1 create Classroom

Readme

Activity

Custom properties

0 stars

0 watching

0 forks


Report repository

Releases


1_lab2_w2025-420_541_Lab2-1

Go to file t Add file <> Code


Local Codespaces


 Clone ?

HTTPS SSH GitHub CLI

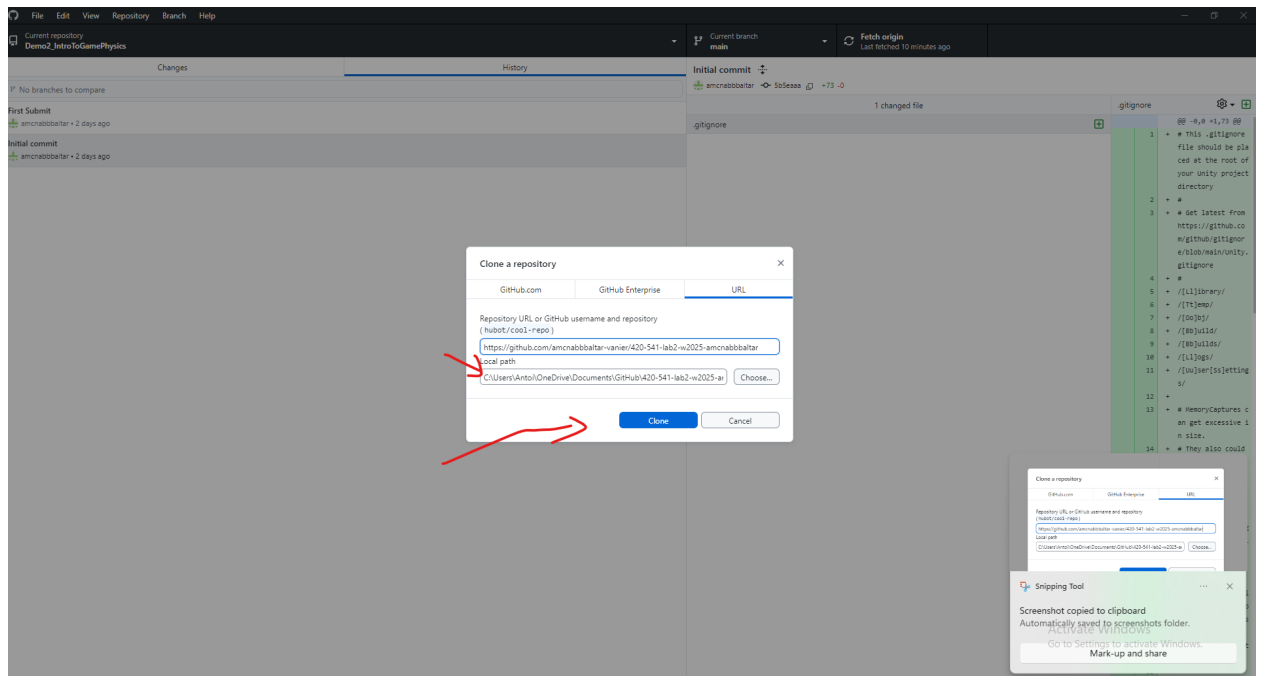
`https://github.com/amcnabbbaltar-vanier/420-541` 

Clone using the web URL.

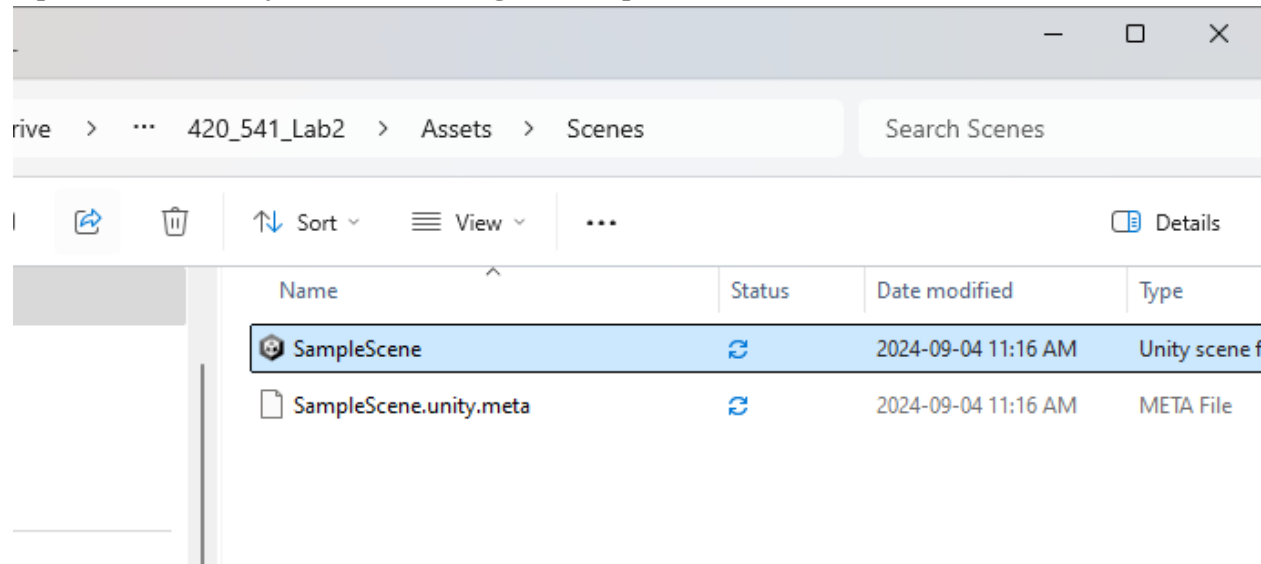
 Open with GitHub Desktop

 Download ZIP

-
- Choose a folder and clone the repo



-
- Once the folder is cloned open it in Unity by opening the scene
- 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.

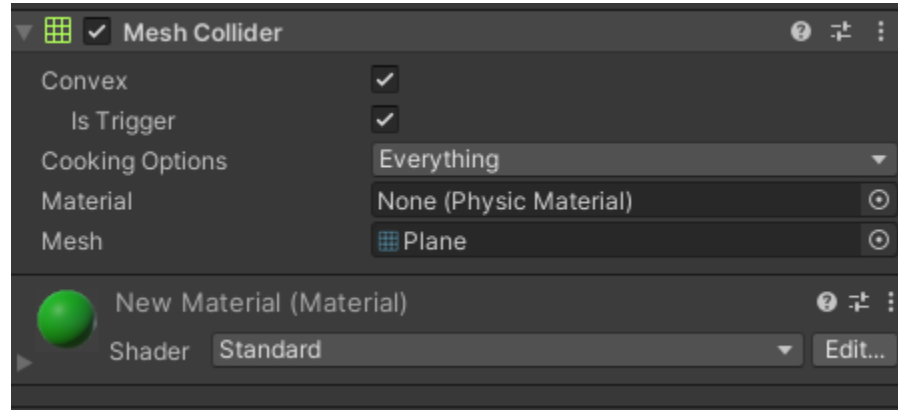


Task 1: Adjust Object Physics Settings (1 pt)

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

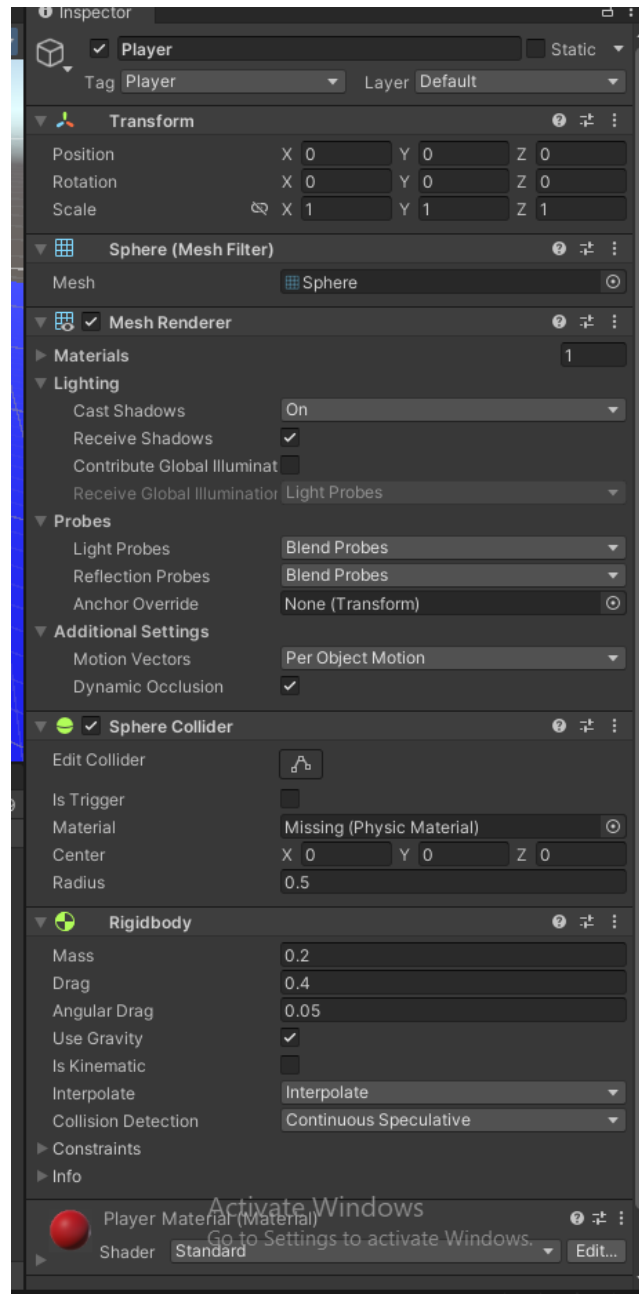
1. Green Goal Object:

- 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**.



2. Red Player Object:

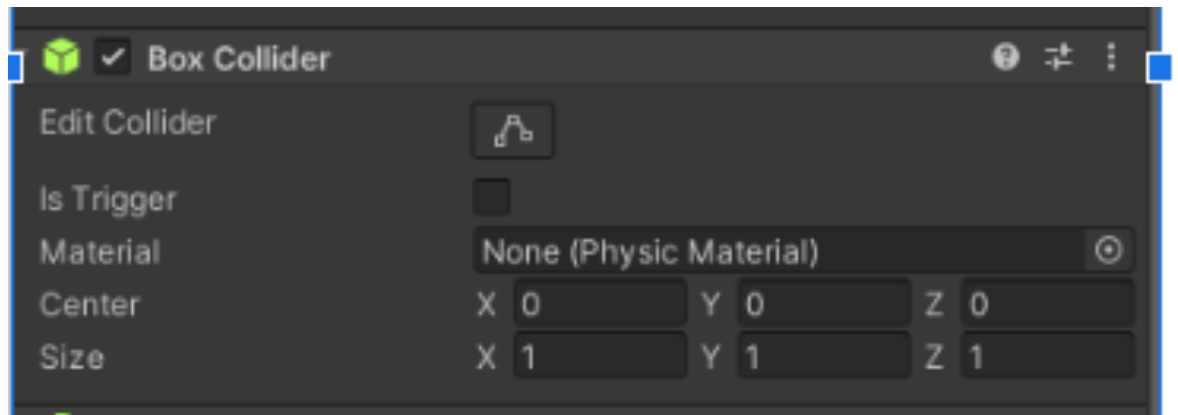
- Select the player object (the red ball) .
- Ensure it is a **dynamic** object:
 - Add a **Rigidbody** component (if not already present).
 - Make sure the **Use Gravity** option is checked.
 - Set the **Is Kinematic** property to **false**.



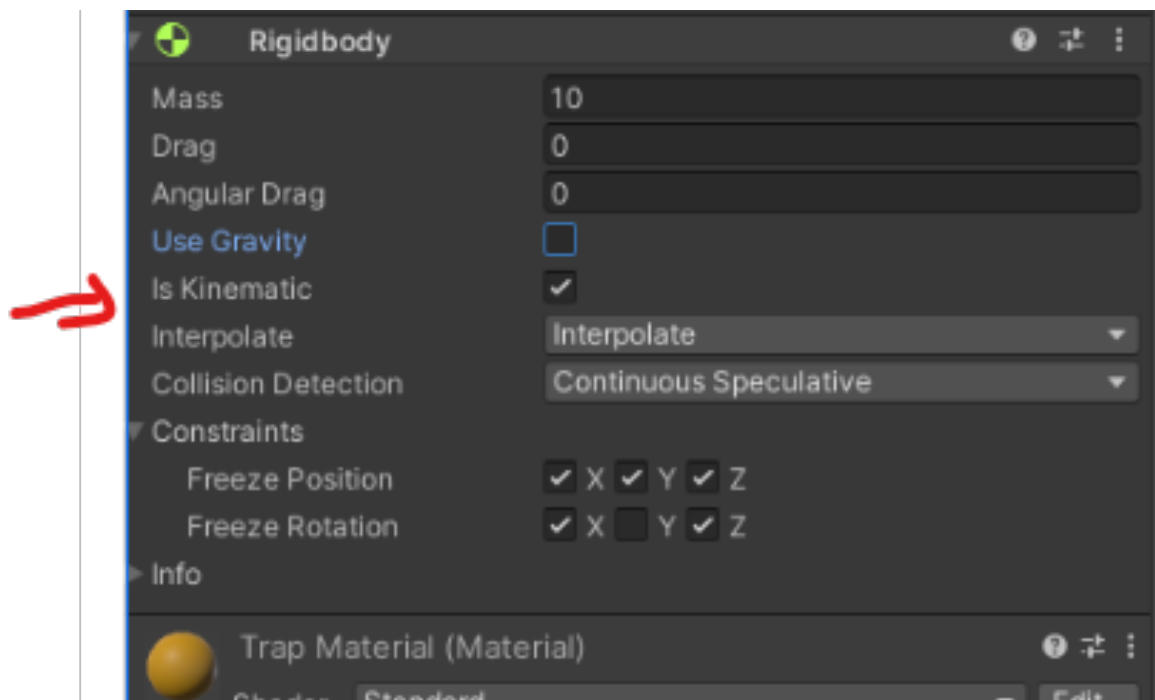
3.

4. **Yellow Trap Object:**

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

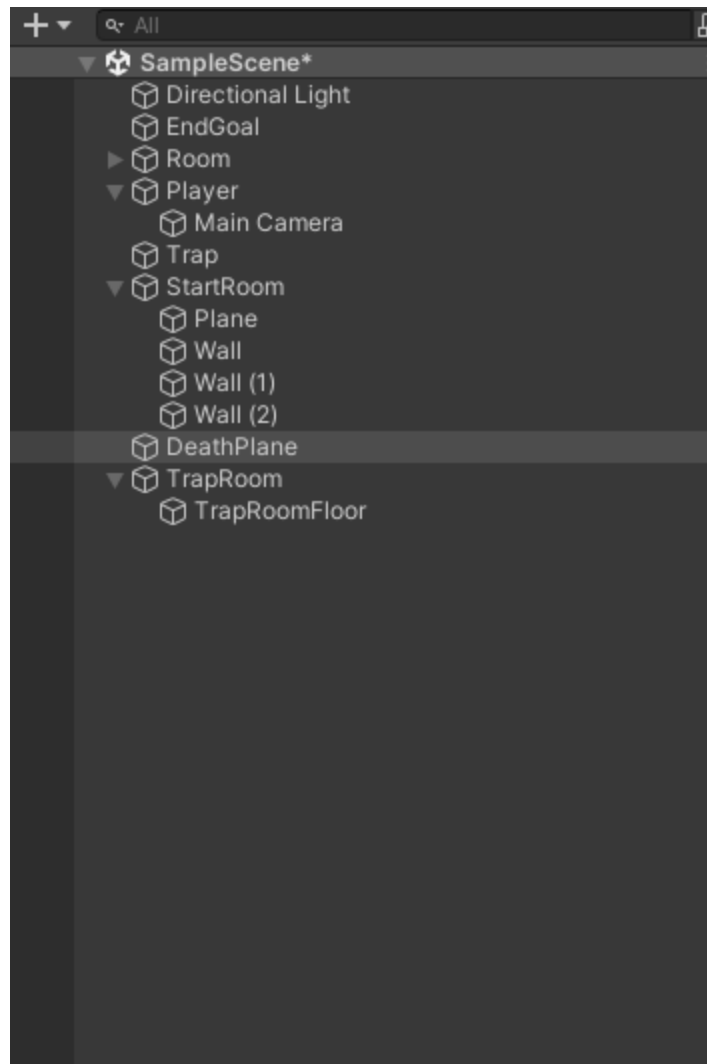


-
- Ensure it is a **kinematic** object:
 - Add a **Rigidbody** component.
 - Set **Is Kinematic** to **true** and **Use Gravity** to **false**.

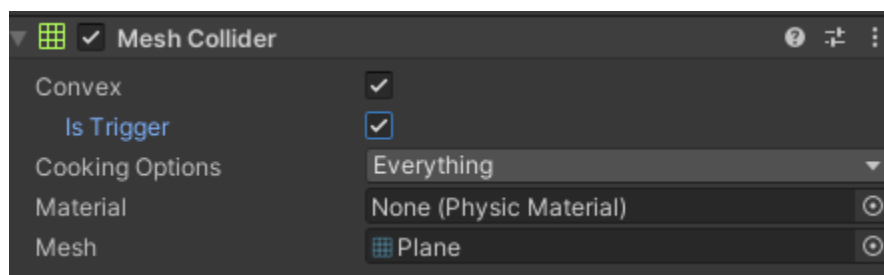


5.Setup the death Plane

- 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**.



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);

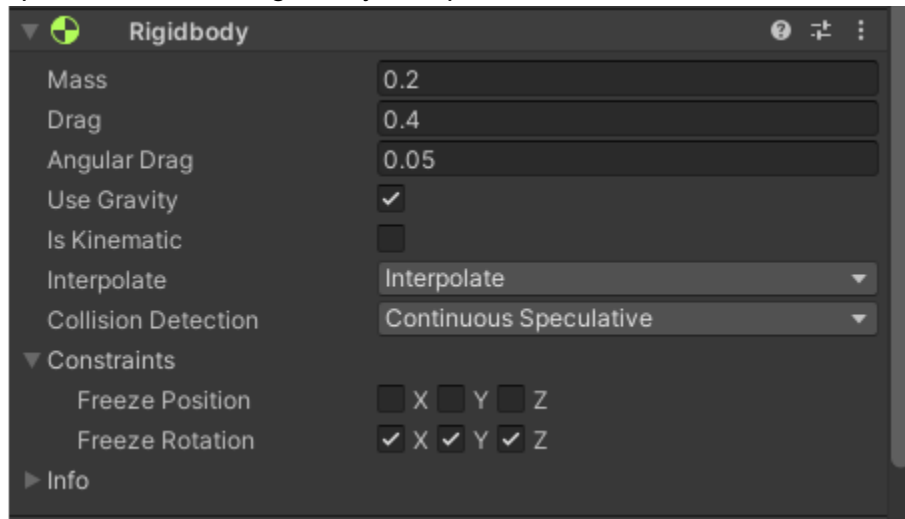
        // Apply movement to the rigidbody
        rb.AddForce(movement * moveSpeed, ForceMode.Force);
    }
}
```

4. Save the script and test the movement by playing the scene.
5. What happens when you try to move the ball ? :

___The code allows the object to move in 4 direction using the keys.

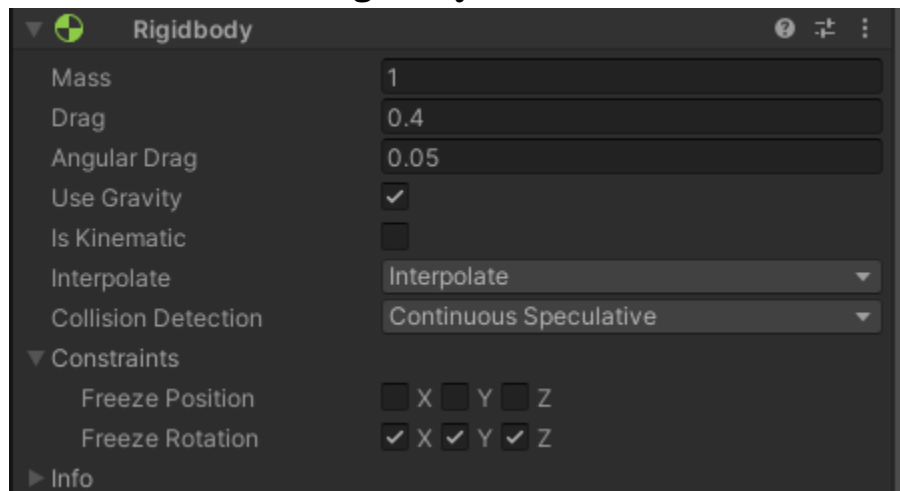
6. Lets fix the ball movement. Right now the weight of the ball is too small for the force we apply on it. So the acceleration that the physics engine impart to the ball is very high so lets make our Player weight bigger

- Change the mass parameter in the Rigidbody Component



Let set it to 1 kg

Rigidbody.MoveRotation

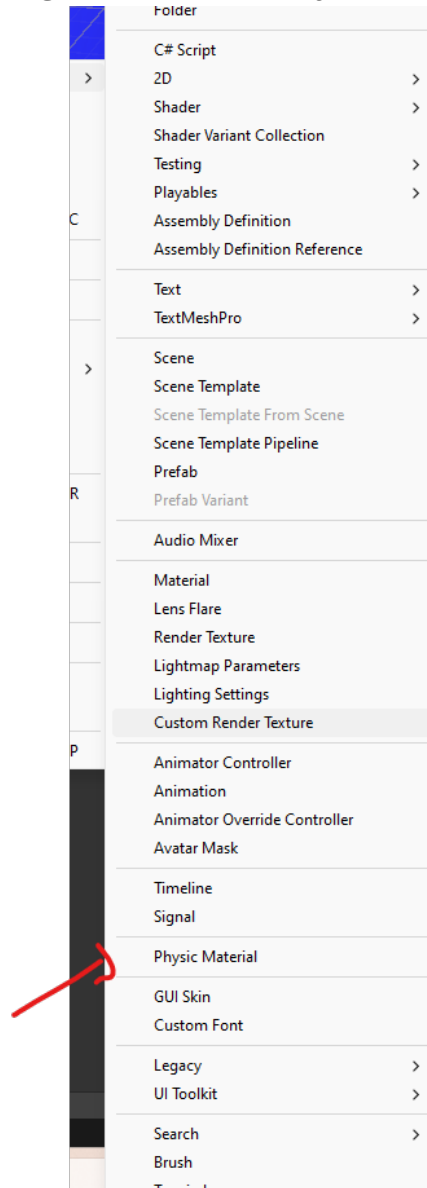


In the inspector you can change the movespeed if you want to move faster

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

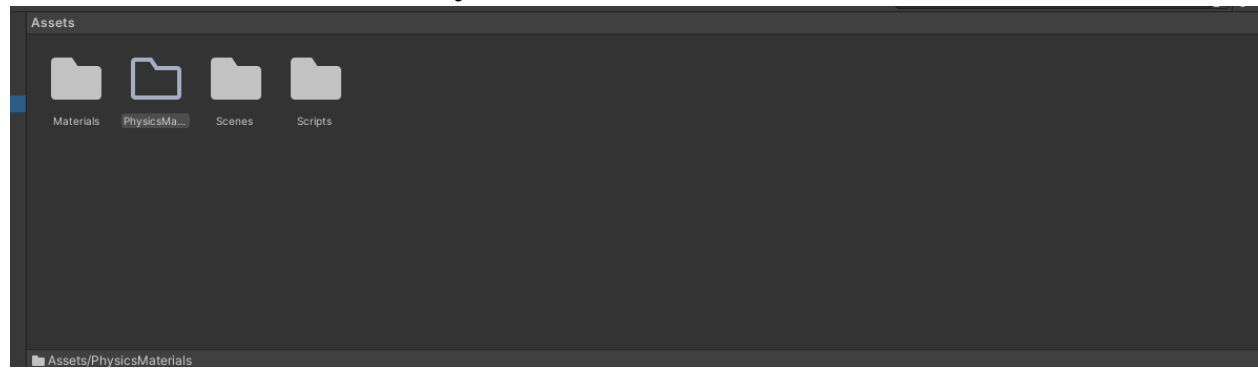
1. Create a **Physics Material**:

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

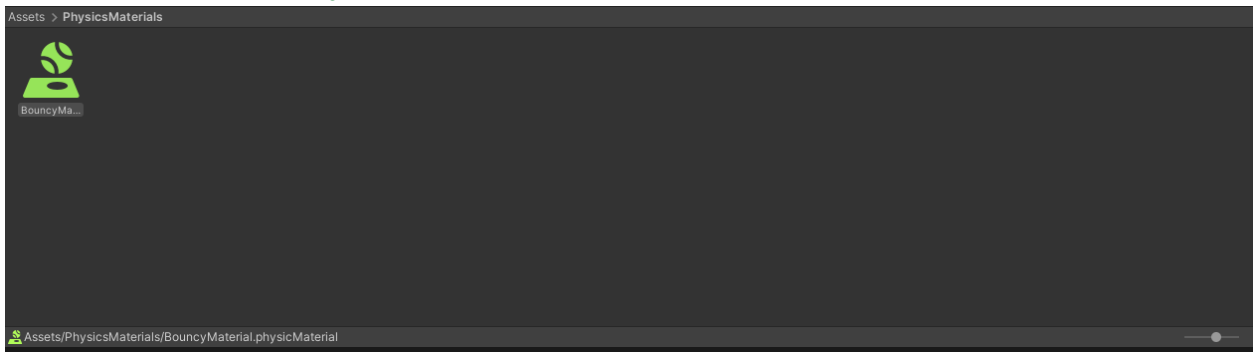


-
- Name it **BouncyMaterial**.

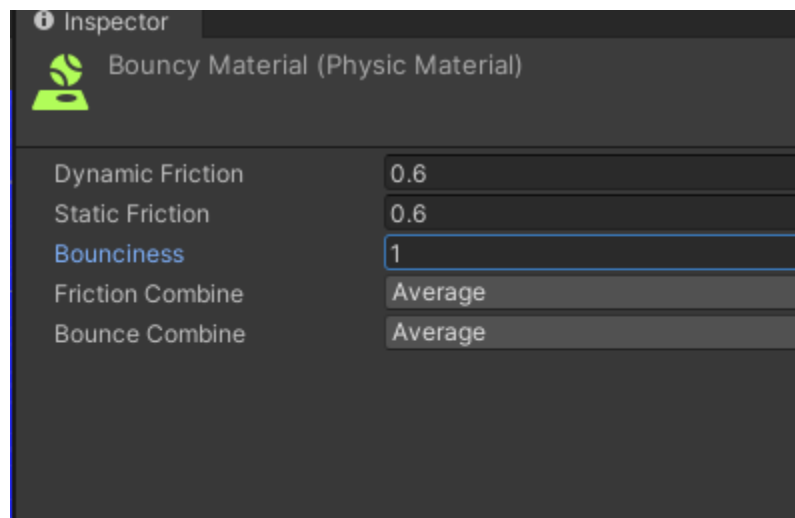
- Add it to a new folder called PhysicsMaterial



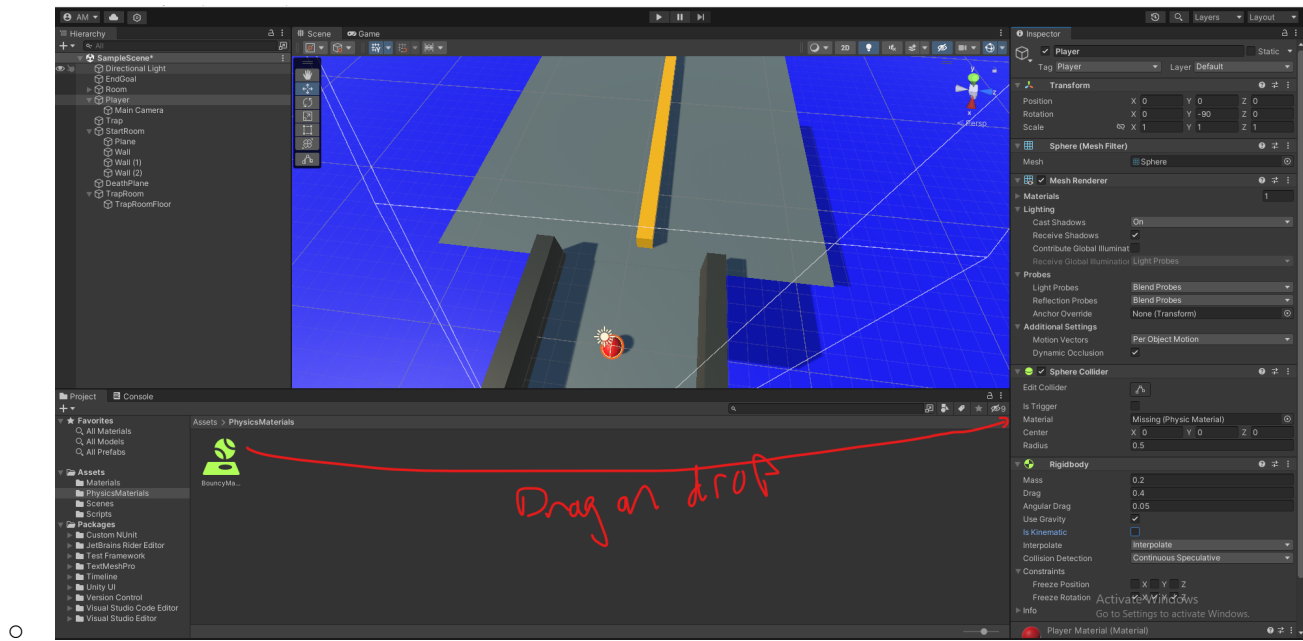
2. Double click on **BouncyMaterial** and



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



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



Task 4: Rotate the Yellow Trap (1 pt)

1. Initial Attempt (with Transform.Rotate):

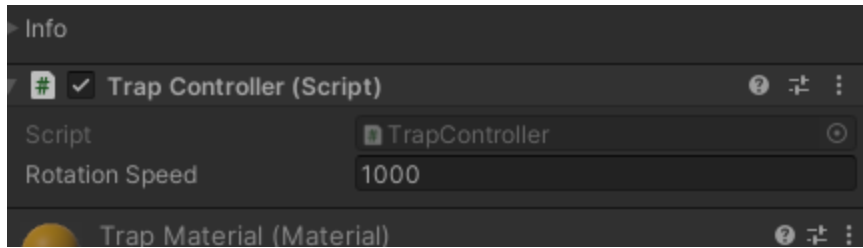
- Create a new C# script called **TrapController**.
- Attach the script to the yellow trap object.
- Use the following code to try rotating the trap with **Transform.Rotate**:

```
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



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

__It is hard for the circle to move in the middle of the trap because the speed and at the same time, there's a clipping issue where in the ball is not touched by the trap.

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;

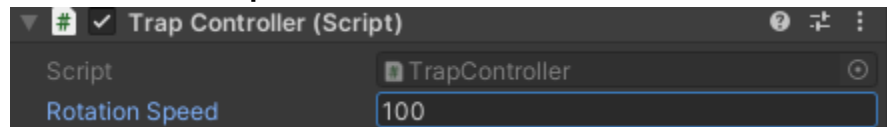
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }
}
```

```

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

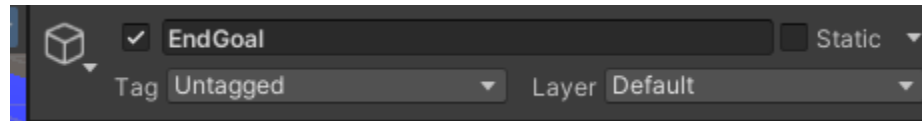
```

- Set the rotation speed back down to 100 now.

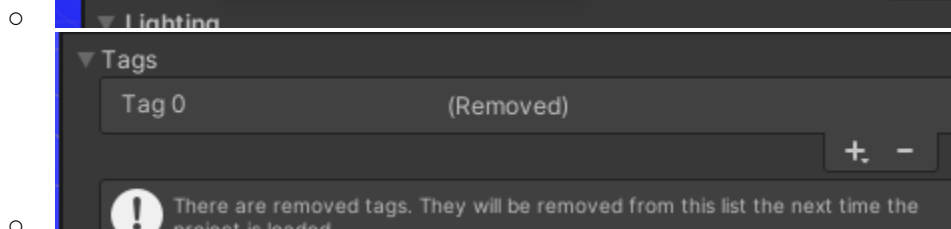
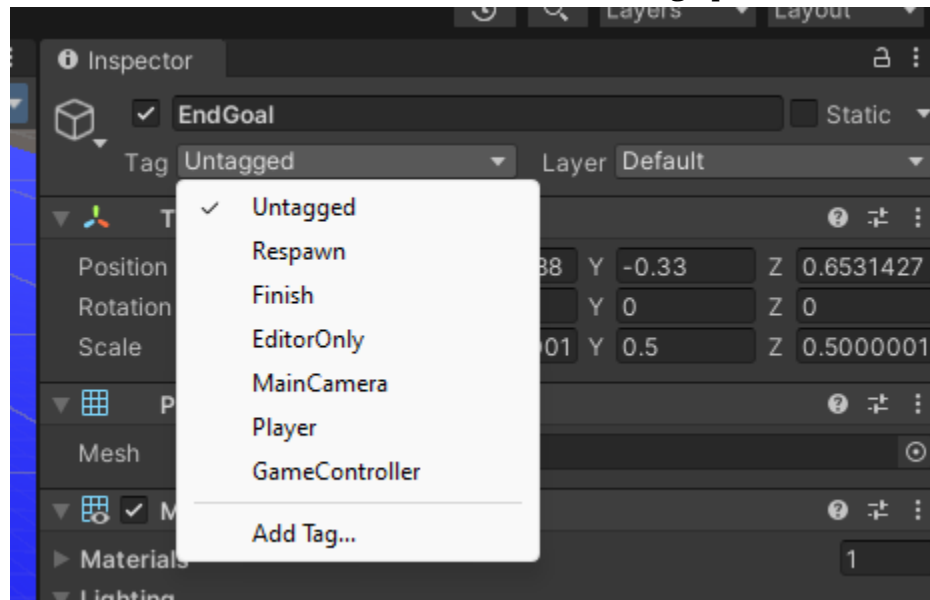


Task 5: Detect Player Reaching the Goal (1 pt)

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.



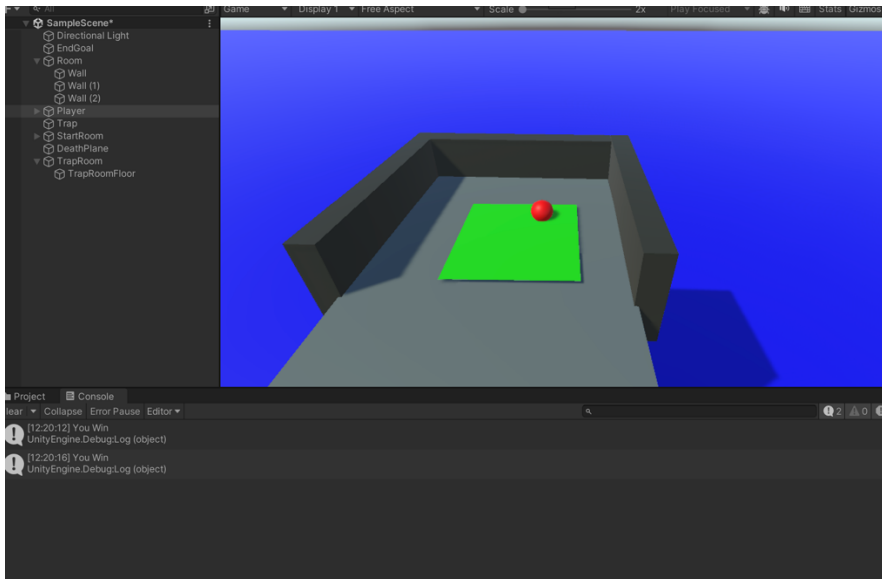
- If one doesn't exist create one. With the AddTag option



- 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:

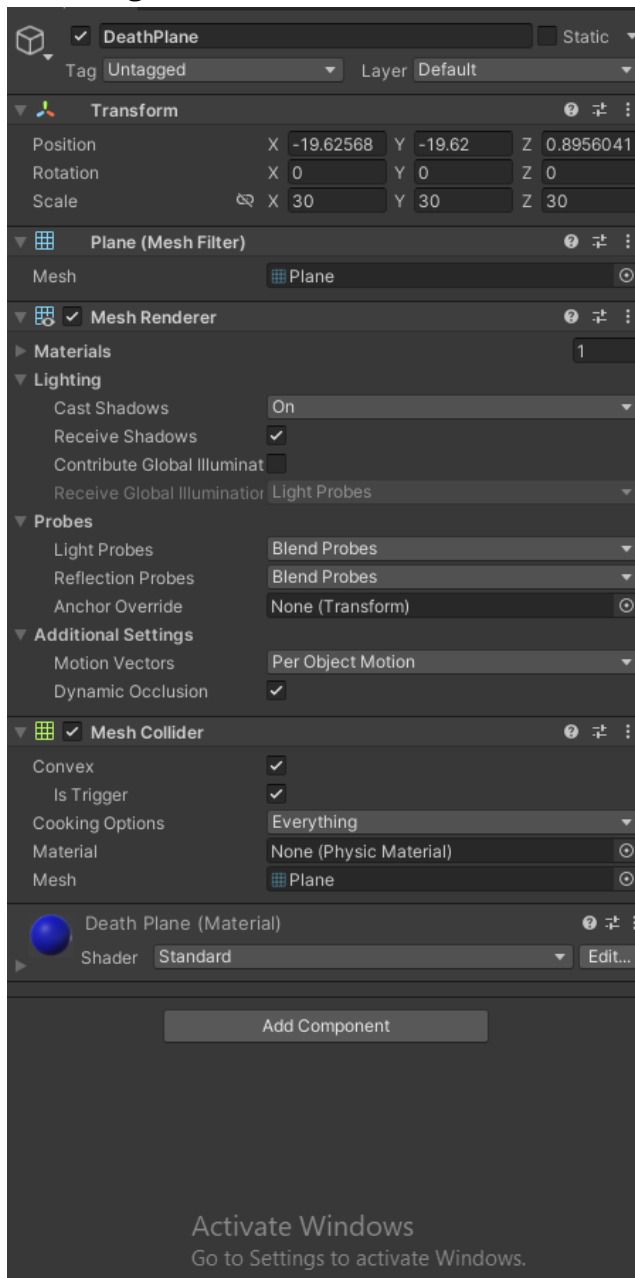
```
void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Goal"))
    {
        Debug.Log("You Win");
    }
}
```

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 :**

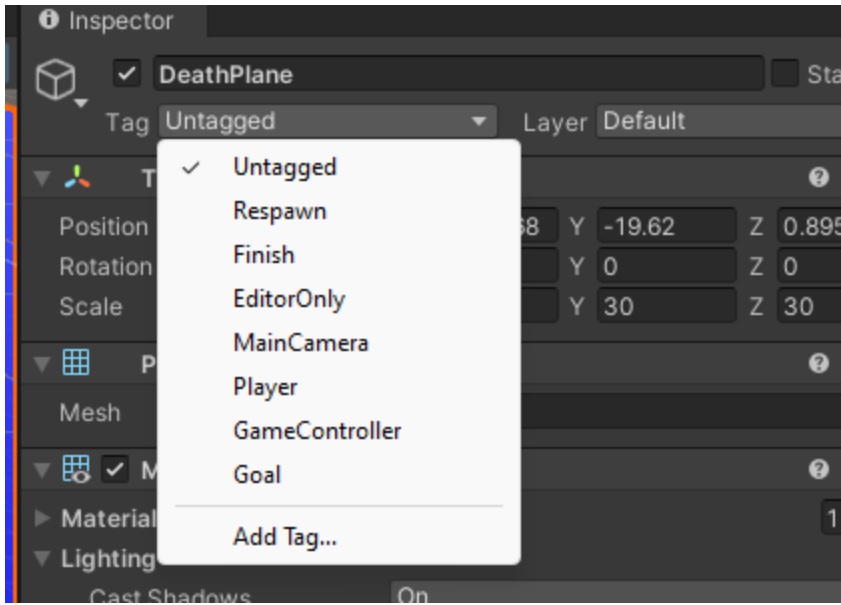


Task 6: Restart Game on Death (1 pt)

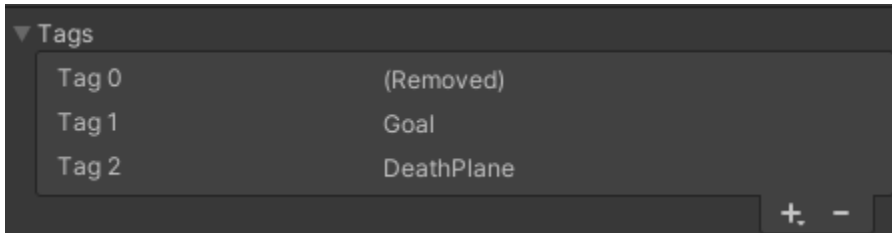
1. Add a tag named **DeathPlane** to the blue death plane object.



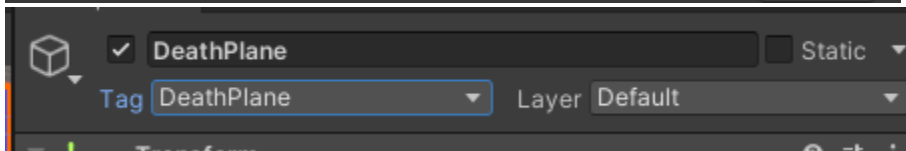
- 2.



3.



4.



5.

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 this filled in document to your project folder
- 2) Submit your code on github. I should see your repo on github classroom.