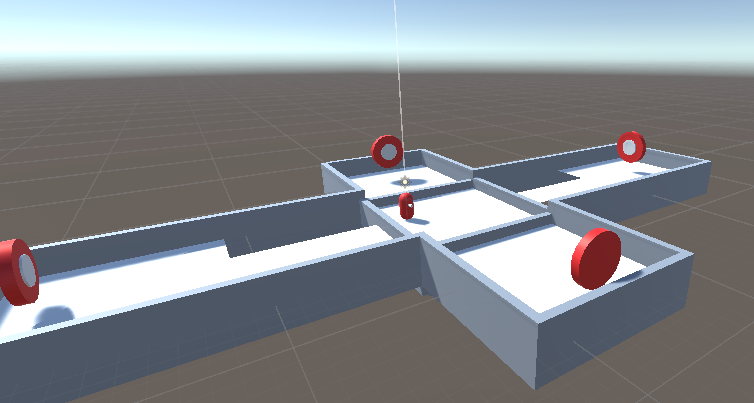
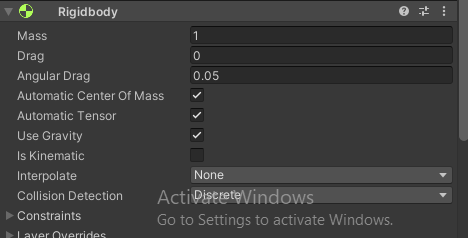
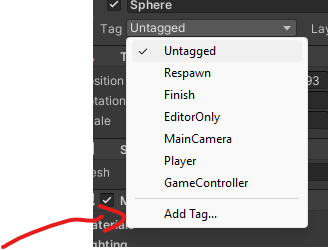
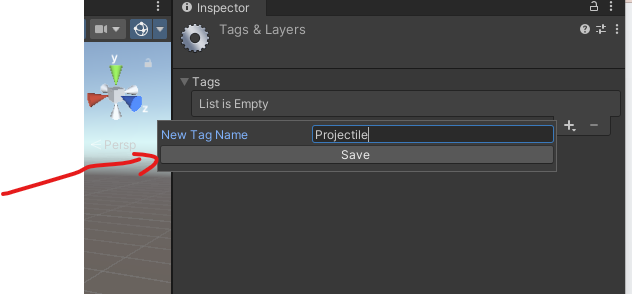
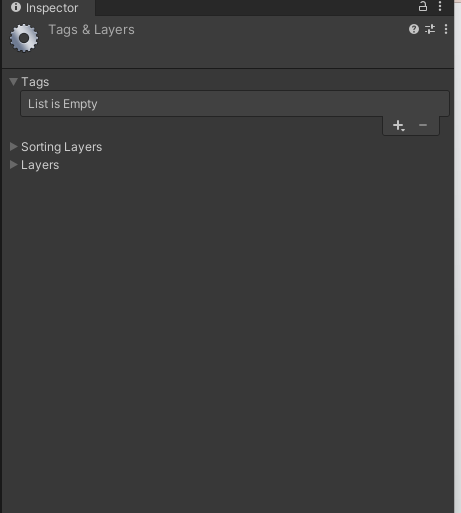
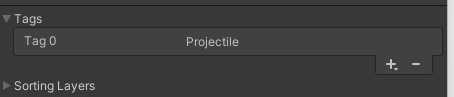
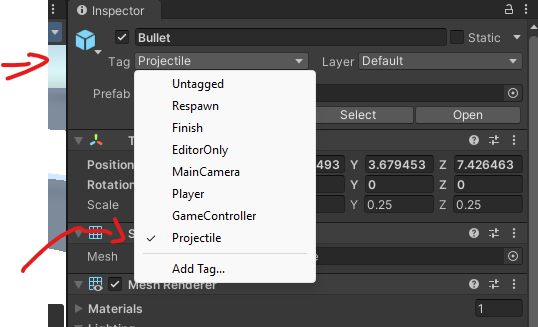
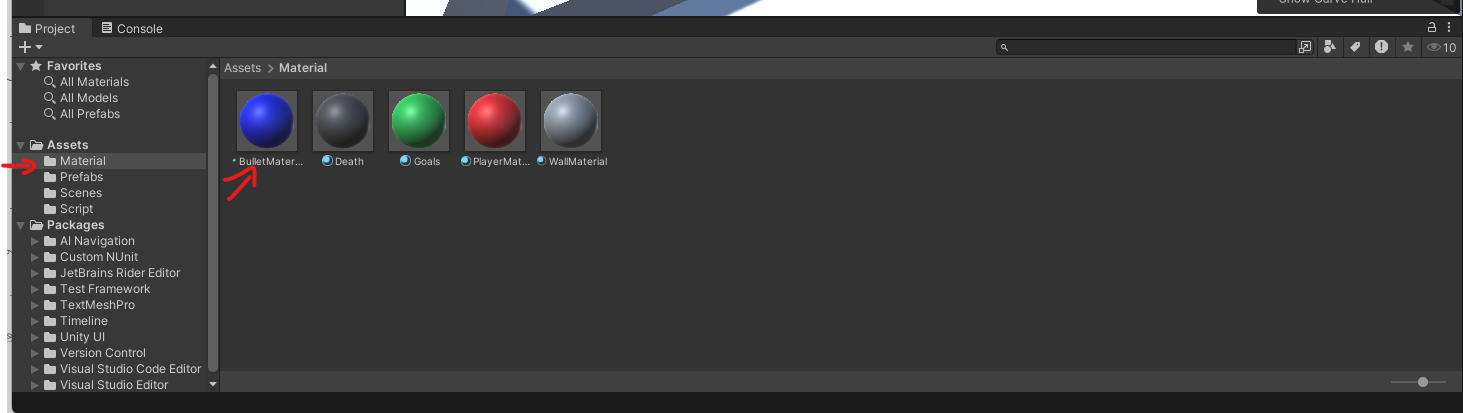
# Lab 3: Instantiate, Invoke, and Object Lifetime

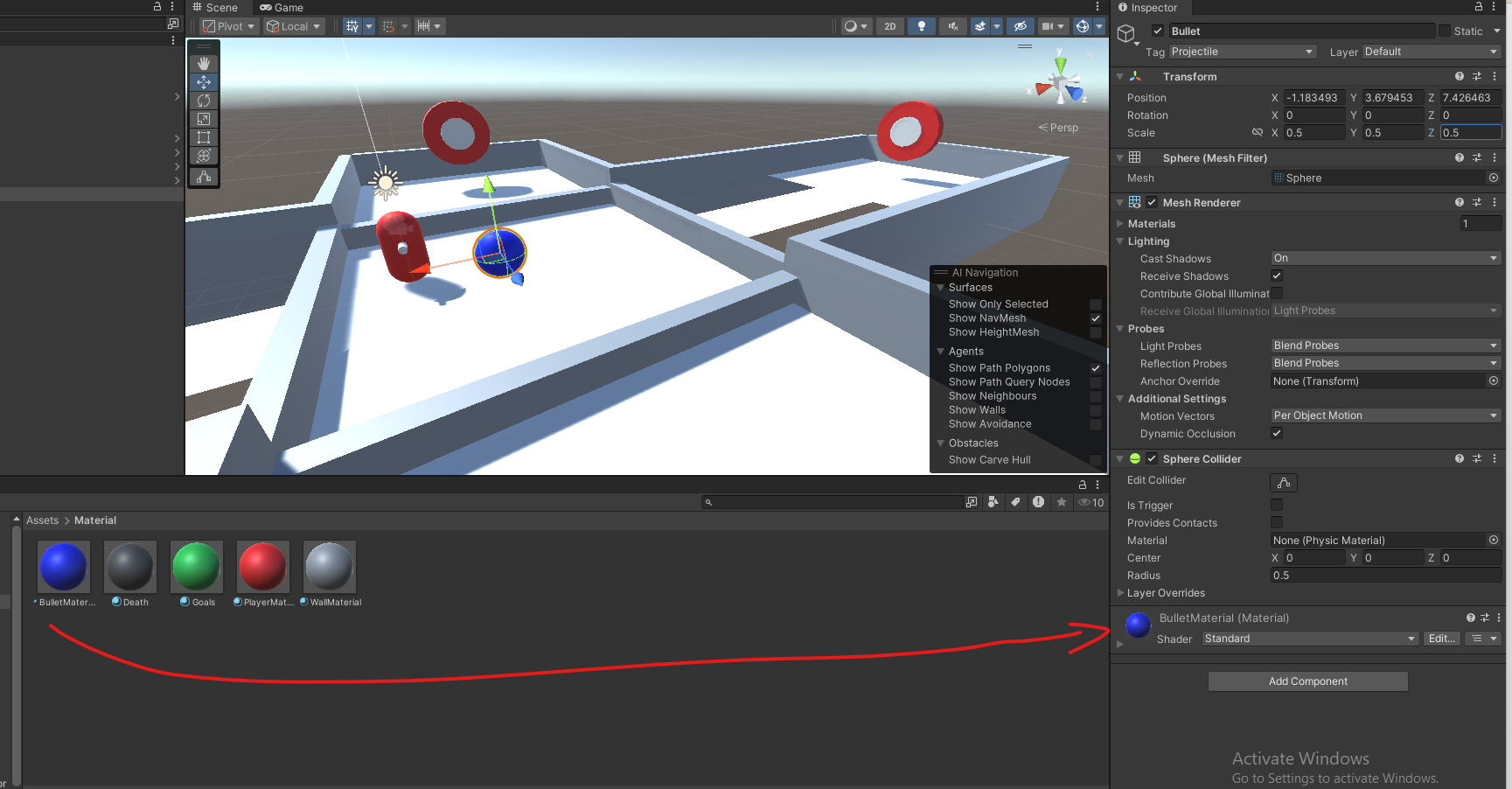
#### **Setup:**

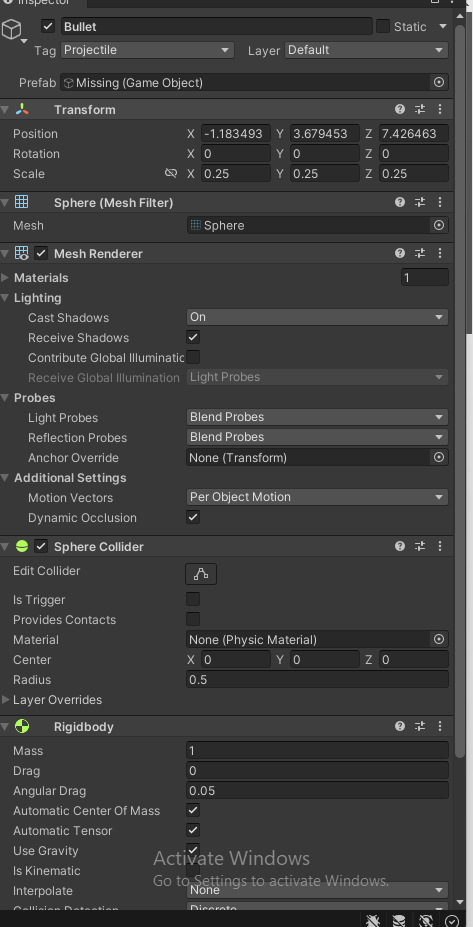
1. Clone the repository:
   * Open this link : <https://classroom.github.com/a/xsIFwI4x>
   * Follow the instruction to clone the repo with Github Desktop
2. Open the sample scene in scene folder   
   

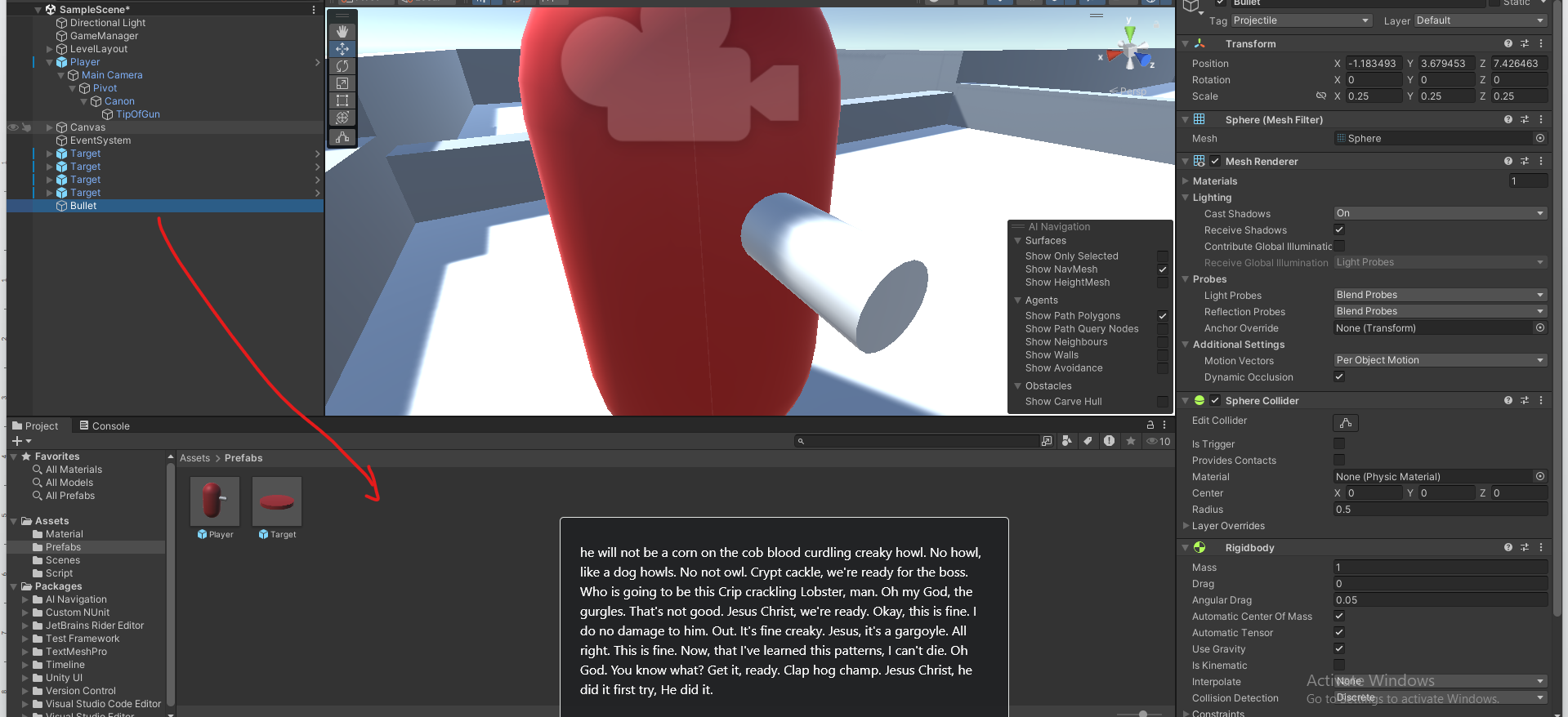
### **Task 1: Create a Bullet/Canonball Prefab**

1. **Create a new GameObject**:
   * Right-click in the *Hierarchy* and select Create > 3D Object > Sphere.
2. **Add Rigidbody Component**:
   * Select the sphere in the *Hierarchy*.
   * In the *Inspector* panel, click Add Component and search for Rigidbody.
   * Make sure the Rigidbody's Use Gravity checkbox is ticked to allow gravity to affect the bullets and make sure it is not a Kinematic object by making sure that the IsKinematic Checkbox is NOT ticked.
   * 
3. **Add Tag**:
   * Click the sphere again, go to the *Inspector*, and click on the Tag dropdown.
   * Select or add a new tag called "Projectile" by clicking Add Tag, then choose Projectile from the list.
   * 
   * 
   * Make sure you choose projectile for you bullet after creating the tag.
4. 
5. Apply the bullet material





* + Your bullet should look something like this :
  + 

1. **Save it as a Prefab**:
   * Drag the sphere from the *Hierarchy* into the *Project* panel to save it as a prefab.
   * 

### **Task 2: Create a BulletComponent Script**

1. **Create a Script**:
   * In the *Project* panel, right-click and select Create > C# Script.
   * Name it BulletComponent.
2. **Edit the Script**:
   * Open BulletComponent.cs in the code editor.
   * The bullet component will contain the code for destroying the ball

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BulletComponent : MonoBehaviour

{

void Start()

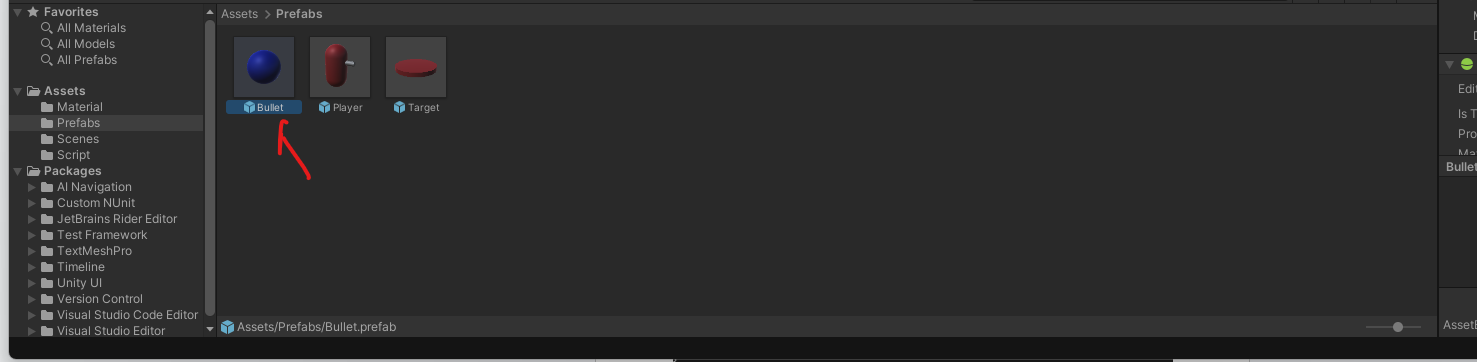
{

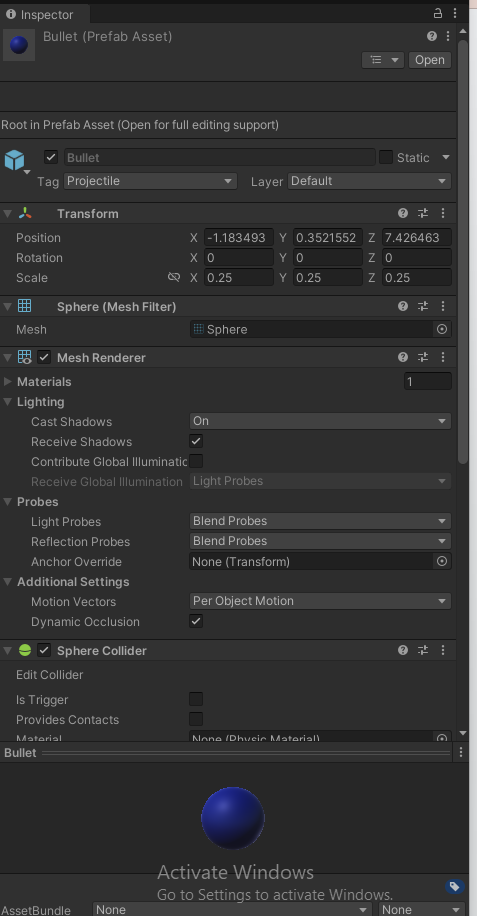
// Destroy object after a few seconds

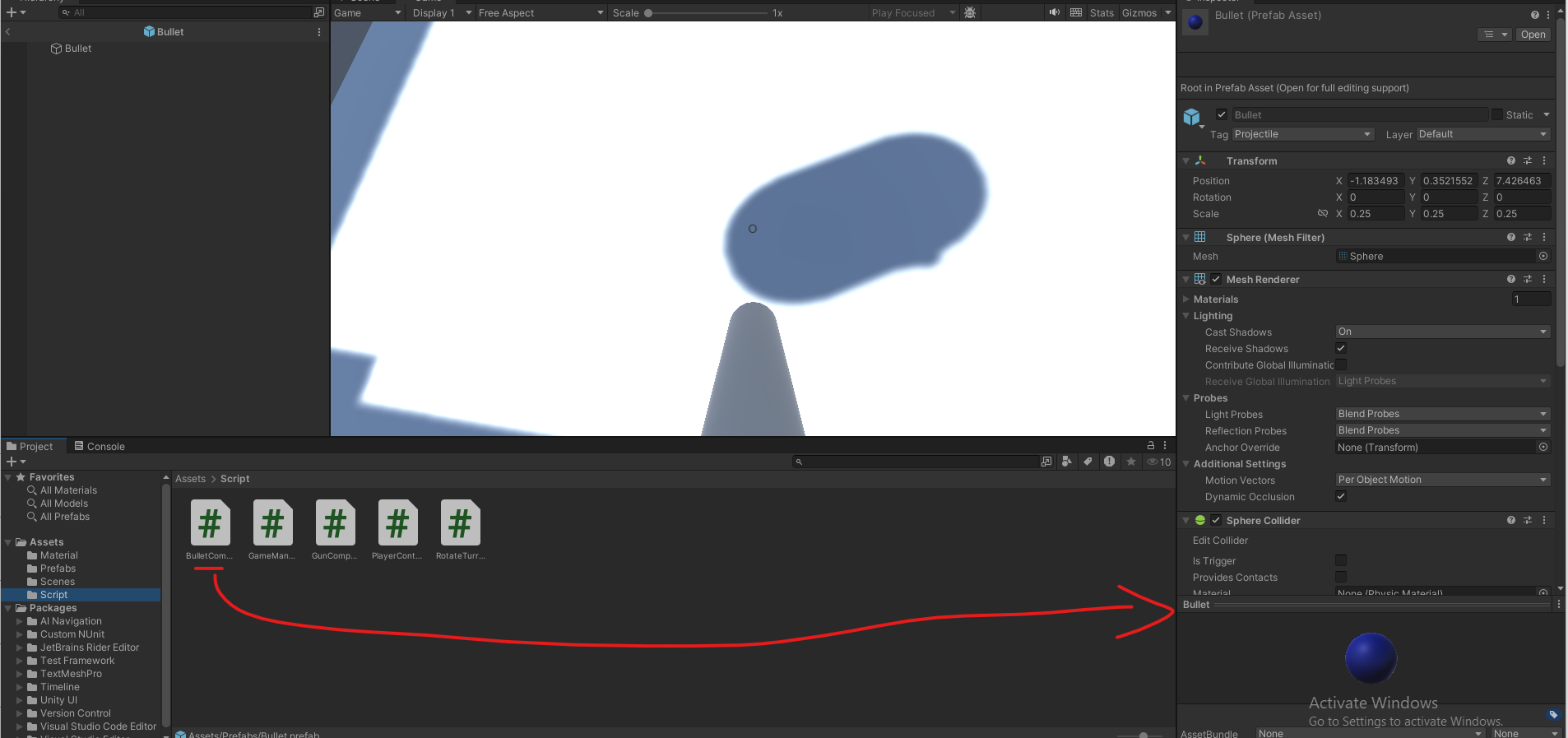
Destroy(gameObject, 5f);

}

}

1. **Attach the Script**:
   * Attach the BulletComponent script to your bullet prefab by dragging it from the *Project* panel to the prefab's *Inspector*.
   * 





### **Task 3: Modify the Gun component to Spawn the bullets**

1. **Edit the Script**:
   * Open GunComponent.cs and implement the following:

using UnityEngine;

public class GunComponent : MonoBehaviour

{

public GameObject bulletPrefab;

public Transform bulletSpawnPoint;

public float bulletMaxImpulse = 100.0f;

public float maxChargeTime = 3.0f;

private float chargeTime = 0.0f;

private bool isCharging = false;

void Update()

{

// TODO add the logic to track player keeping the input down.

if (Input.GetButtonUp("Fire1"))

{

ShootBullet();

}

}

void ShootBullet()

{

GameObject bullet = Instantiate(bulletPrefab, bulletSpawnPoint.position, bulletSpawnPoint.rotation);

Rigidbody rb = bullet.GetComponent<Rigidbody>();

// TODO change that equation so that it adds an impulse that follows charge time

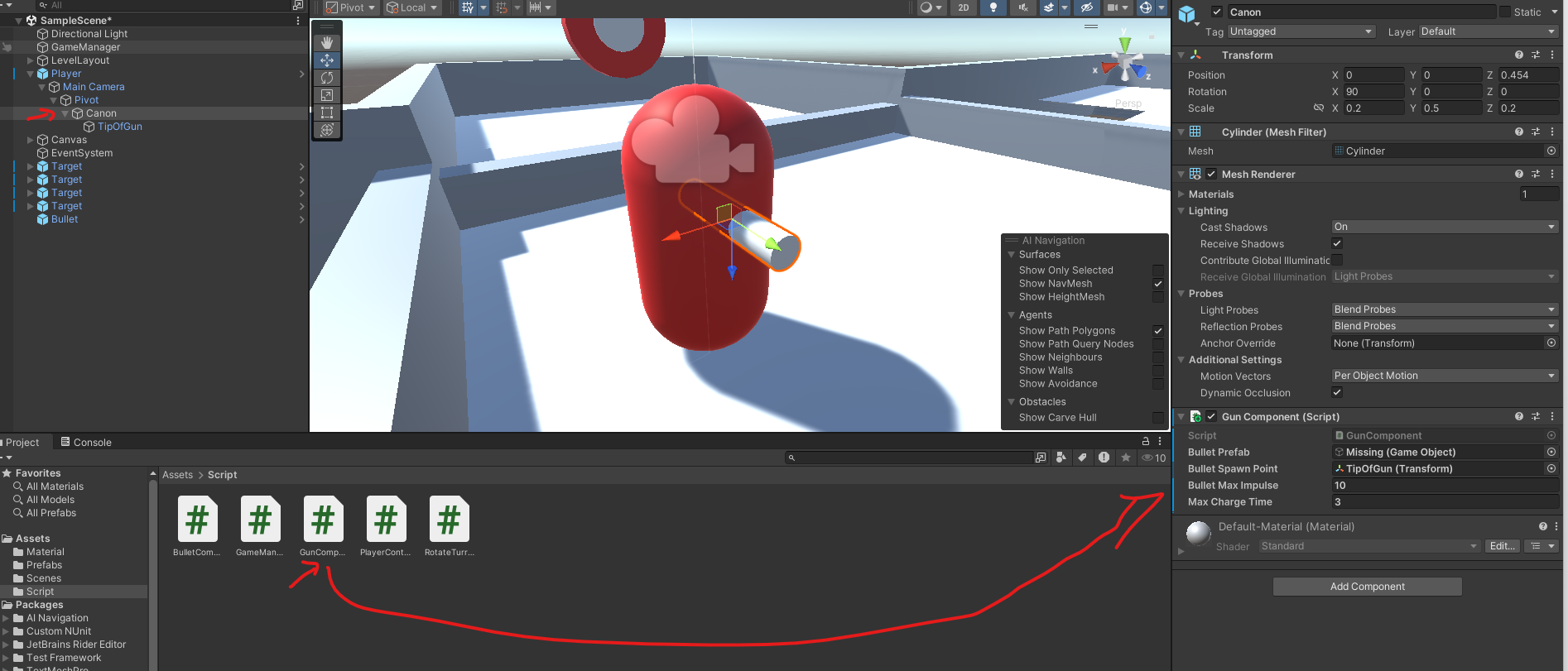
float bulletImpulse = bulletMaxImpulse;

// An impulse is a force you apply on a object in a single instant.

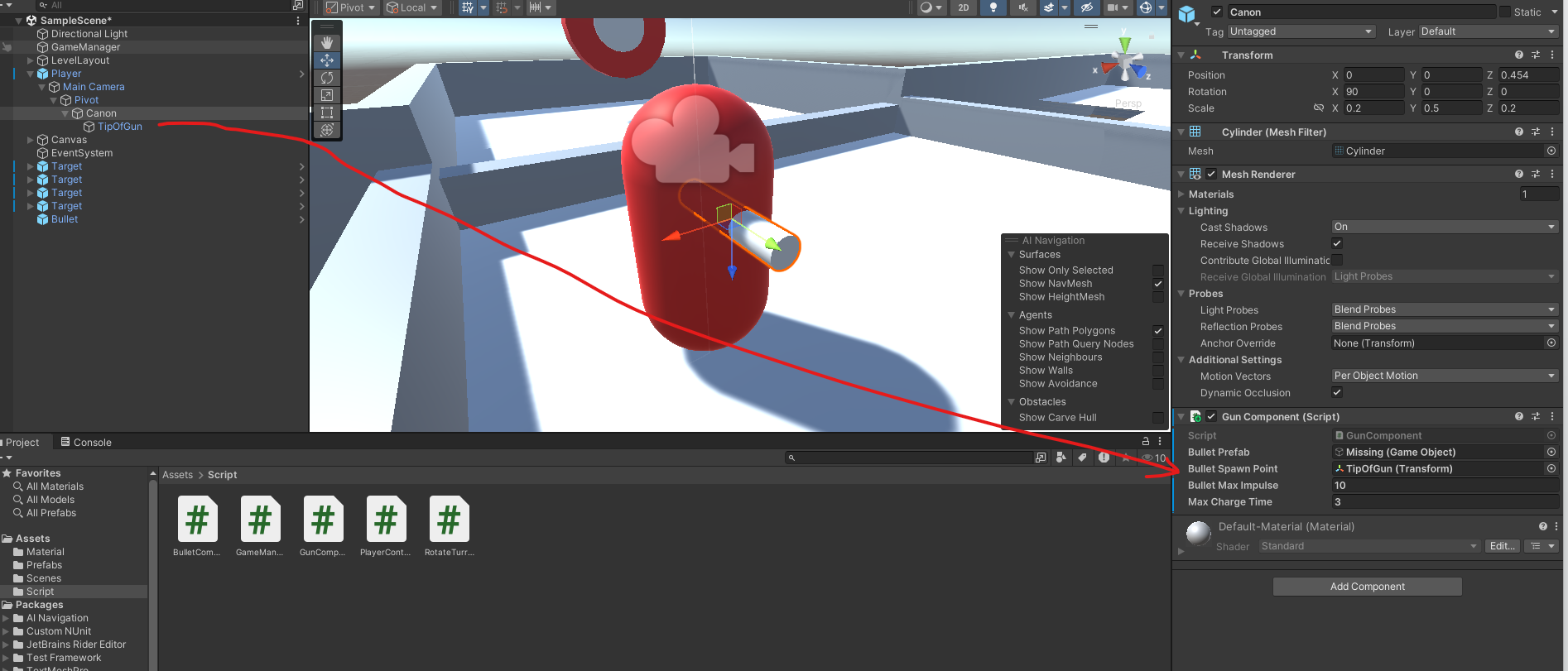
rb.AddForce(bulletSpawnPoint.forward \* bulletImpulse, ForceMode.Impulse);

}

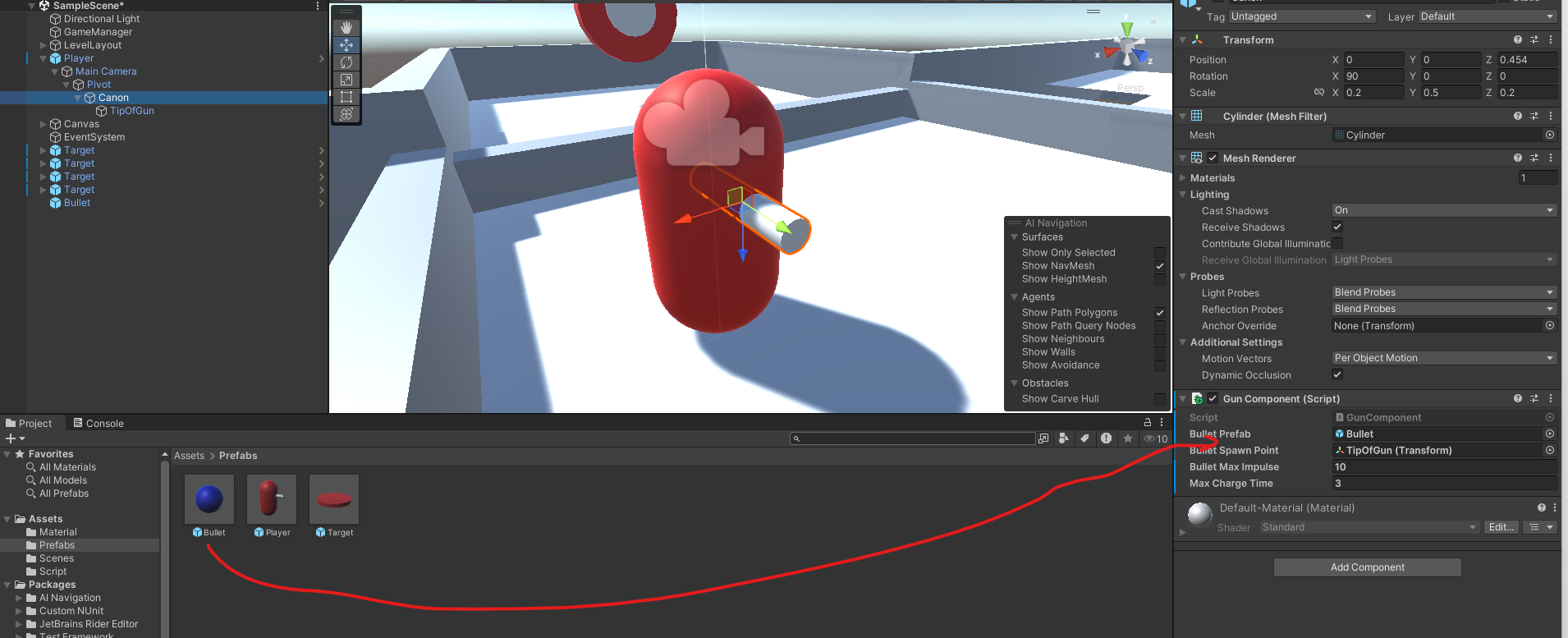
}

1. **Configure the Gun**:
   * The gun component should already be assigned to the Gun gameObject (A child of the player) You will need to assign the bullet prefab and the bullet spawn point in the *Inspector*.
   * 

Assign **the tip of gun transform** as the bullet spawn point.



And the bullet prefab as the Bullet Prefab Proprety



### **Task 5: Modify the Gun Component**

You will need now to modify the gun component Update function to add a charging mechanic for bullet speed. ( IE Charge longer, Bullet goes further )  
**Here are few hints :**

* This will be done within the Update() function.
* You need to detect when the player starts holding the fire button. Use an if condition with the boolean Input.GetButtonDown("Fire1") to detect when the button is initially pressed.
  + When a player first press down the button the chargetime should reset to 0.
* Using the boolean Input.GetButton("Fire1") increments the charge time every frame the button is held down. By adding the Time.deltaTime to the charge time.
  + You should increase the chargeTime value over time using Time.deltaTime, which gives you the time passed between each frame.
* To avoid excessively fast bullets, you can apply a maximum limit to the chargeTime.
  + Use Mathf.Clamp to keep the chargeTime between 0 and a certain max value (like 3 seconds).
    - chargeTime = Mathf.Clamp(chargeTime, 0, maxChargeTime);

**Copy the code you came up with here:   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

void Update()

    {

        // if the player starts pressing the button

        if (Input.GetButtonDown("Fire1"))

        {

            chargeTime = 0.0f;

            isCharging = true;

        }

        // if the player holds the button

        if (Input.GetButton("Fire1"))

        {

            chargeTime += Time.deltaTime;

            chargeTime = Mathf.Clamp(chargeTime, 0, maxChargeTime);

        }

        // if the player releases the button

        if (Input.GetButtonUp("Fire1"))

        {

            ShootBullet();

            isCharging = false;

        }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Now change the ShootBullet function to

void ShootBullet()

{

GameObject bullet = Instantiate(bulletPrefab, bulletSpawnPoint.position, bulletSpawnPoint.rotation);

Rigidbody rb = bullet.GetComponent<Rigidbody>();

// Scale bullet force based on charge time

float bulletImpulse = (chargeTime / maxChargeTime) \* bulletMaxImpulse;

rb.AddForce(bulletSpawnPoint.forward \* bulletImpulse, ForceMode.Impulse);

}

In the end your GunComponent should look something like this **( NOTE : the next code snippet is a picture)**

### **Task 4: Target Interaction and Scoring**

1. **Create TargetComponent Script**:
   * In the *Project* panel, create a new C# script called TargetComponent.
2. **Edit the Script**:
   * Open TargetComponent.cs and implement collision handling:

using UnityEngine;

public class TargetComponent : MonoBehaviour

{

private Renderer targetRenderer;

private Color originalColor;

public Color hitColor = Color.green; // Change to any color you want

private void Start()

{

targetRenderer = GetComponent<Renderer>();

if (targetRenderer != null)

{

originalColor = targetRenderer.material.color;

}

}

private void OnCollisionEnter(Collision collision)

{

if (collision.gameObject.CompareTag("Projectile"))

{

GameManager.Instance.IncrementScore();

// Change color

if (targetRenderer != null)

{

targetRenderer.material.color = hitColor;

}

// Restore color and hide target after 5 seconds

Invoke("ResetColor", 5f);

}

}

private void ResetColor()

{

if (targetRenderer != null)

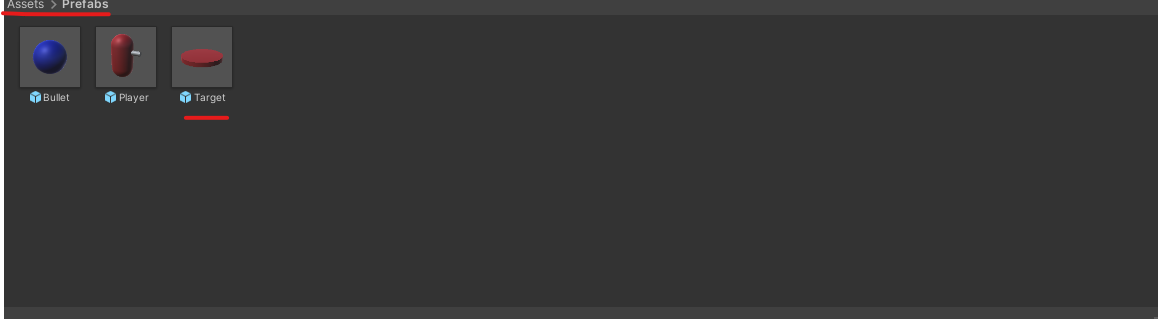
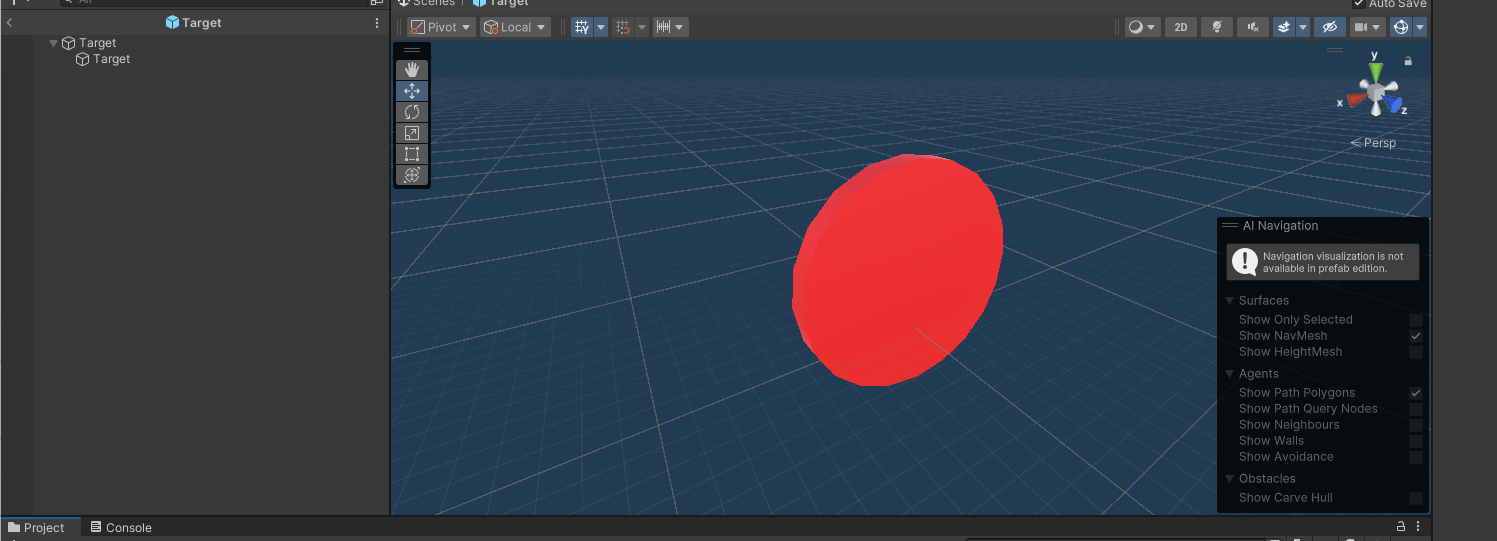
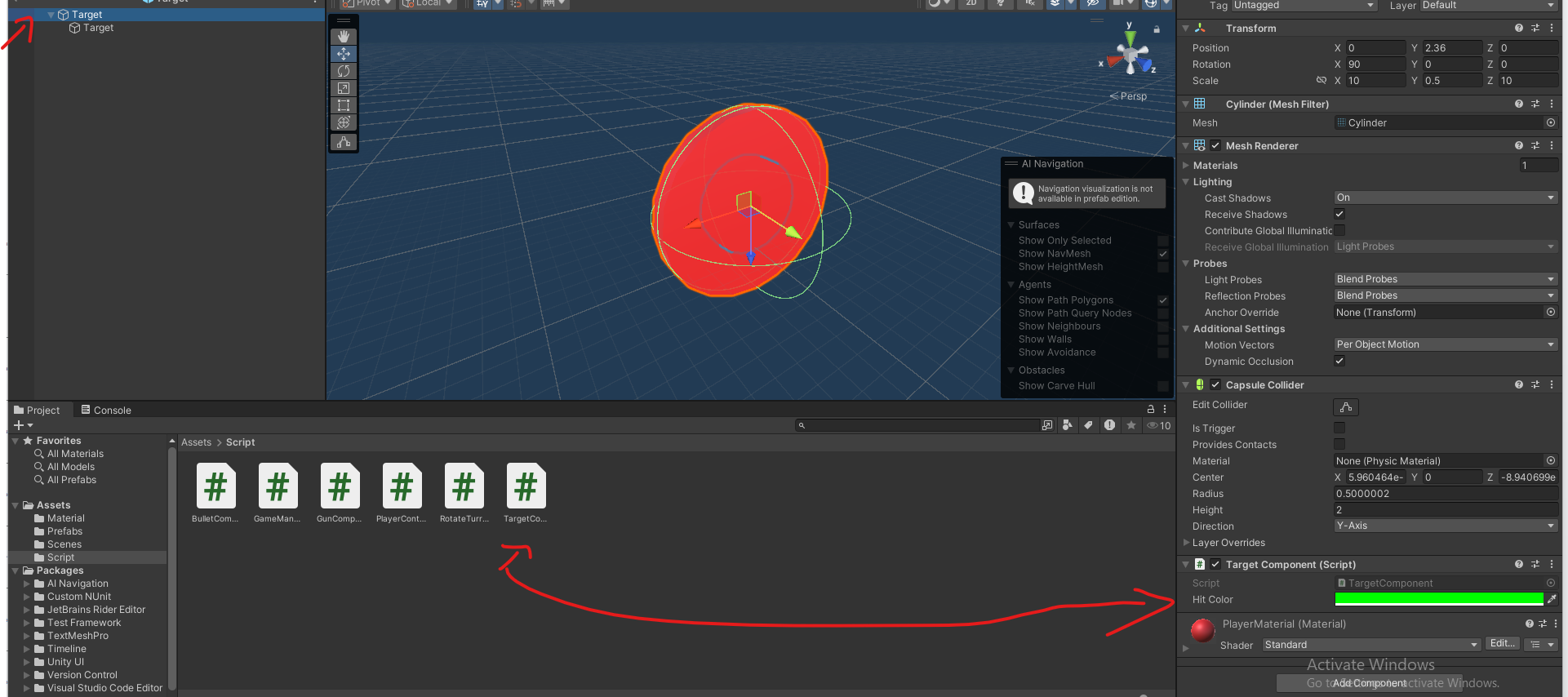
{

targetRenderer.material.color = originalColor;

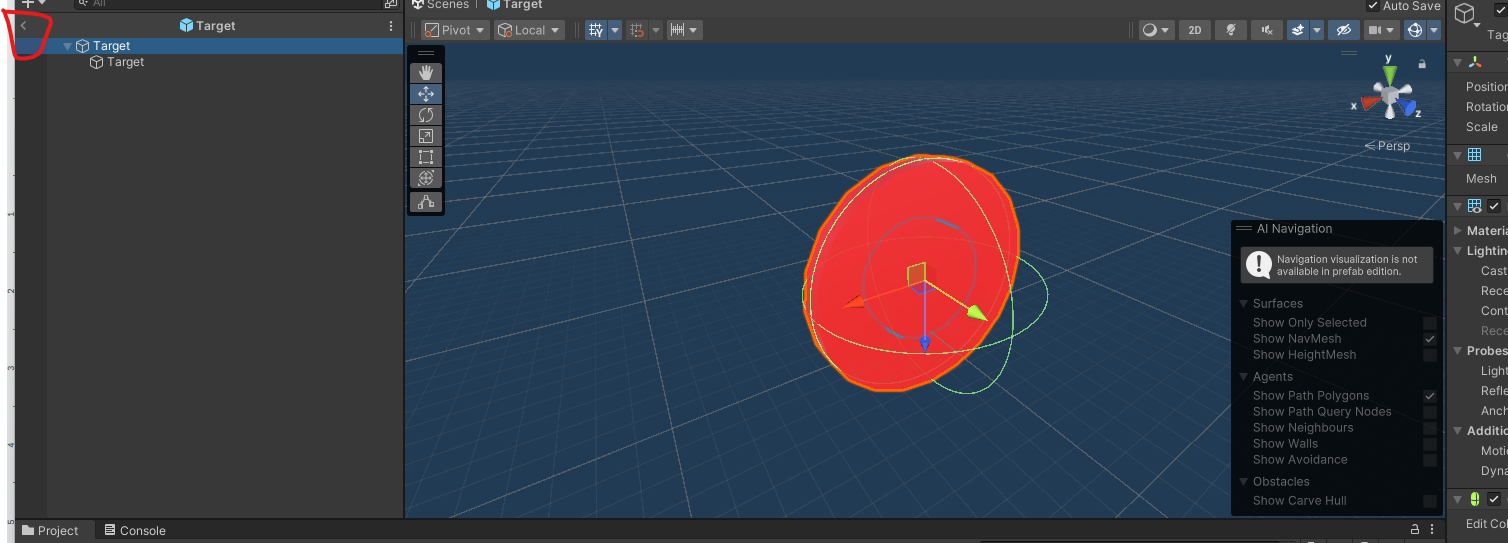
}

}

}

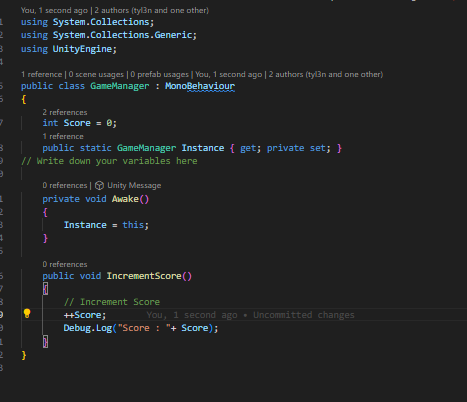
1. **Attach the Script**:
   * Select the target prefab in your Prefab folder and double click it.  
     
   * It will change the viewport to a blue background with only your prefab where the scene would be
   * 
   * Now click on the parent object and add the TargetComponent to it
   * 
   * This will add the target component on all the Target object in your scene.

You can go back to your scene by pressing the arrow



1. **Connect to GameManager**:
   * Ensure your GameManager script has an IncrementScore() method. This will update the score when a bullet hits the target.

Make sure the GameManager IncrementScore Is properly implemented



### **Submission**

1. **Add this file to your Asset folder and commit to your repo.**