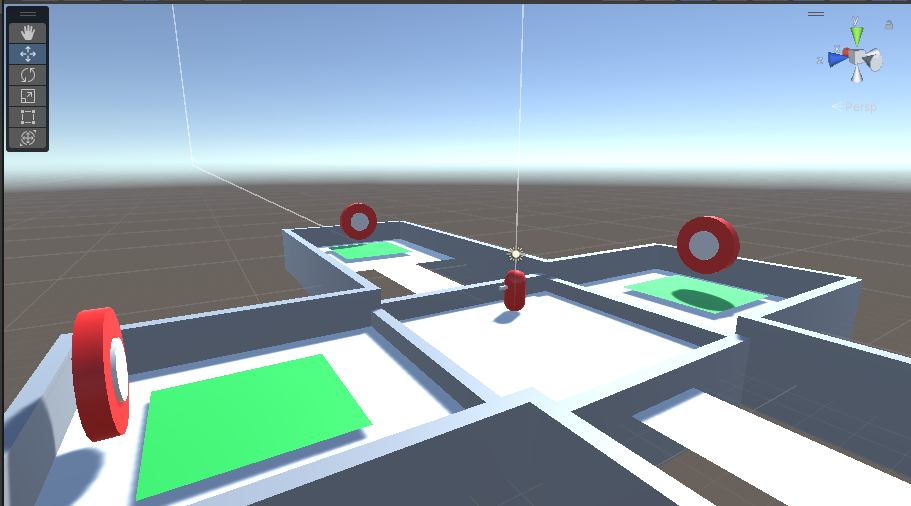
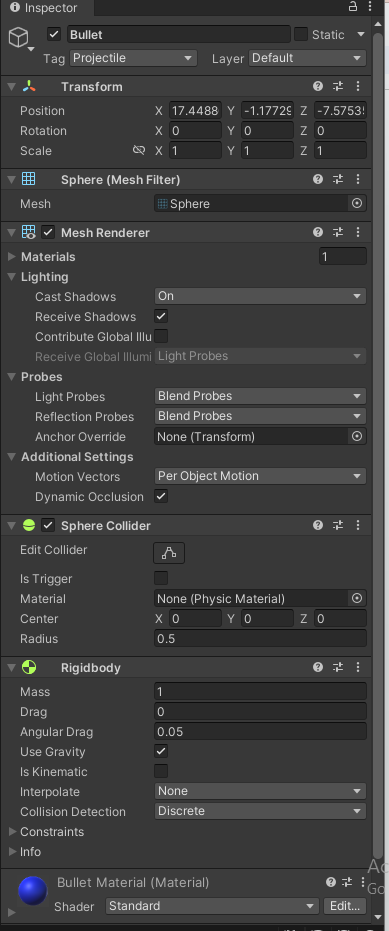
# Lab 3: Instantiate, Invoke, and Object Lifetime

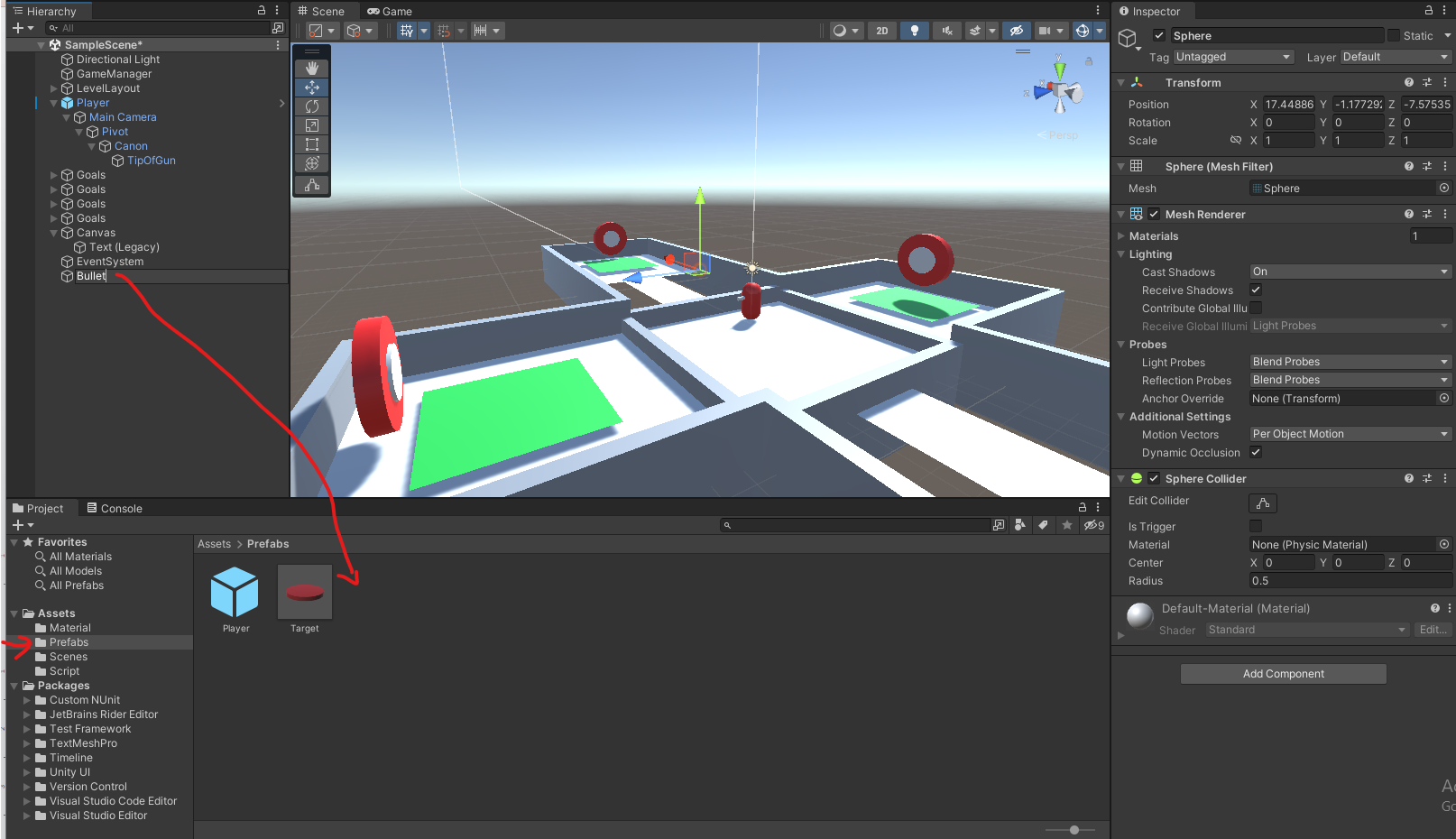
#### **Setup:**

1. Clone the repository:
   * Open your terminal or Git Bash.
   * Run the command: git clone https://github.com/amcnabbbaltar/420\_541\_Lab3
   * This will download the project files to your local machine.
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.
   * Ensure the Rigidbody's Use Gravity checkbox is ticked to allow physics simulation.
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.



1. **Save as 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.
   * Add logic to apply force to the bullet:

using UnityEngine;

public class BulletComponent : MonoBehaviour

{

public float bulletSpeed = 10f; // Speed of the bullet

void Start()

{

// Add an impulse force in the forward direction

Rigidbody rb = GetComponent<Rigidbody>();

rb.AddForce(transform.forward \* bulletSpeed, ForceMode.Impulse);

}

}

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: Create a Bullet Spawning Script**

1. **Create a New Script**:
   * In the *Project* panel, right-click and create a new C# script called GunComponent.
2. **Edit the Script**:
   * Open GunComponent.cs and implement the following:

using UnityEngine;

public class GunComponent : MonoBehaviour

{

public GameObject bulletPrefab;

public Transform bulletSpawnPoint;

public float chargeSpeed = 10f;

private float chargeTime = 0f;

void Update()

{

if (Input.GetButtonUp("Fire1"))

{

// Spawn bullet when Fire1 is released

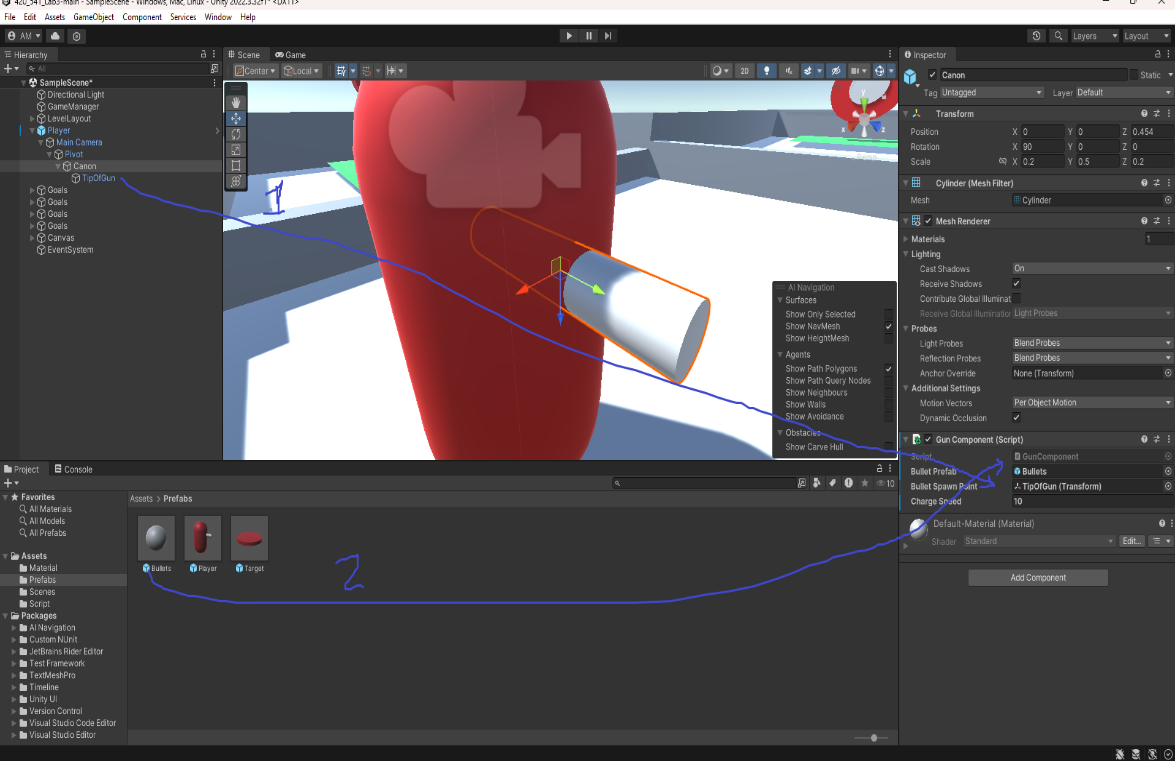
GameObject bullet = Instantiate(bulletPrefab, bulletSpawnPoint.position, bulletSpawnPoint.rotation);

}

}

}

1. **Configure the Gun**:
   * In Unity, It is in the player Gameobject :  
     
   * Attach the GunComponent to it and assign the bullet prefab and the bullet spawn point in the *Inspector*.
     1. Make sure to use the tip of the gun as the bullet spawn point.
     2. And to drag the bullet prefab from the Project Hiearchy



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

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

* You need to detect when the player starts holding the fire button. Use 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.
* While the fire button is held down, use Input.GetButton("Fire1") to continue increasing the charge time. This will be done within Update().
  + 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).
* Access the bullet component to change it’s speed according to your chargetime
  + You can access an object component with this line of code : BulletComponent bulletComp = bullet.GetComponent<BulletComponent>();

**Copy the code you came up with here :**using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GunComponent : MonoBehaviour

{

public GameObject bulletPrefab;

public Transform bulletSpawnPoint;

public float chargeSpeed = 10f;

private float chargeTime = 0f;

private float chargeMin = 0f, chargeMax = 300f;

void Update()

{

if (Input.GetButtonUp("Fire1"))

{

// Spawn bullet when Fire1 is released

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

BulletComponent bulletComp = bullet.GetComponent<BulletComponent>();

bulletComp.bulletSpeed = chargeSpeed \* chargeTime;

}

if (Input.GetButtonDown("Fire1"))

{

chargeTime = 0f;

}

if (Input.GetButton("Fire1"))

{

chargeTime += Time.deltaTime;

chargeTime = Mathf.Clamp(chargeTime, chargeMin, chargeMax);

}

}

}

### **Task 4: Bullet Lifetime Management**

1. **Modify BulletComponent**:
   * Open BulletComponent.cs again and add a delayed destruction method:

void Start()

{

Rigidbody rb = GetComponent<Rigidbody>();

rb.AddForce(transform.forward \* bulletSpeed, ForceMode.Impulse);

// Destroy the bullet after 5 seconds

Destroy(gameObject, 5f);

}

1. **Test the Bullet**:
   * Play the scene and ensure the bullet is destroyed 5 seconds after being spawned.

### **Task 5: 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

{

public void OnCollisionEnter(Collision collision)

{

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

{

GameManager.Instance.IncrementScore();

// Hide target after 5 seconds

Invoke("HideTarget", 5f);

}

}

void HideTarget()

{

gameObject.SetActive(false);

}

}

1. **Attach the Script**:
   * Select the target objects in your scene and attach the TargetComponent script to them.
2. **Connect to GameManager**:
   * Ensure your GameManager script has an IncrementScore() method. This will update the score when a bullet hits the target.

### **Bonus Task 6: Change the Target's Parent Object Color**

1. **Modify TargetComponent**:
   * Add the following code to the TargetComponent.cs script inside the collision check:

Renderer parentRenderer = transform.parent.GetComponent<Renderer>();

if (parentRenderer != null)

{

parentRenderer.material.color = Color.green;

Invoke("ChangeColorBack", 5f); // Change color back after 5 seconds

}

void ChangeColorBack()

{

Renderer parentRenderer = transform.parent.GetComponent<Renderer>();

if (parentRenderer != null)

{

parentRenderer.material.color = Color.red;

}

}

1. **Test the Color Change**:
   * When the target is hit by a bullet, the parent object should change to green and back to red after 5 seconds.

### **Submission**

1. **Zip Your Assets Folder**:
   * Save your Lab to your git hub and then send a link to your github repo:  
     https://github.com/Mujahc/Game-Programming-02.git
   * Make sure the Repo is set to public
2. **Submit**:
   * Submit this document filled with the proper info
   * If the repository is private, ensure the instructor has access by adding them as a collaborator.

