Unity API (Input Axis, Raycast and Instantiation Methods)

LAB # 6

Fall 2024

CSE-411L Intro to Game Development Lab

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Abdullah Hamid

Date:

21st December 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Objective:

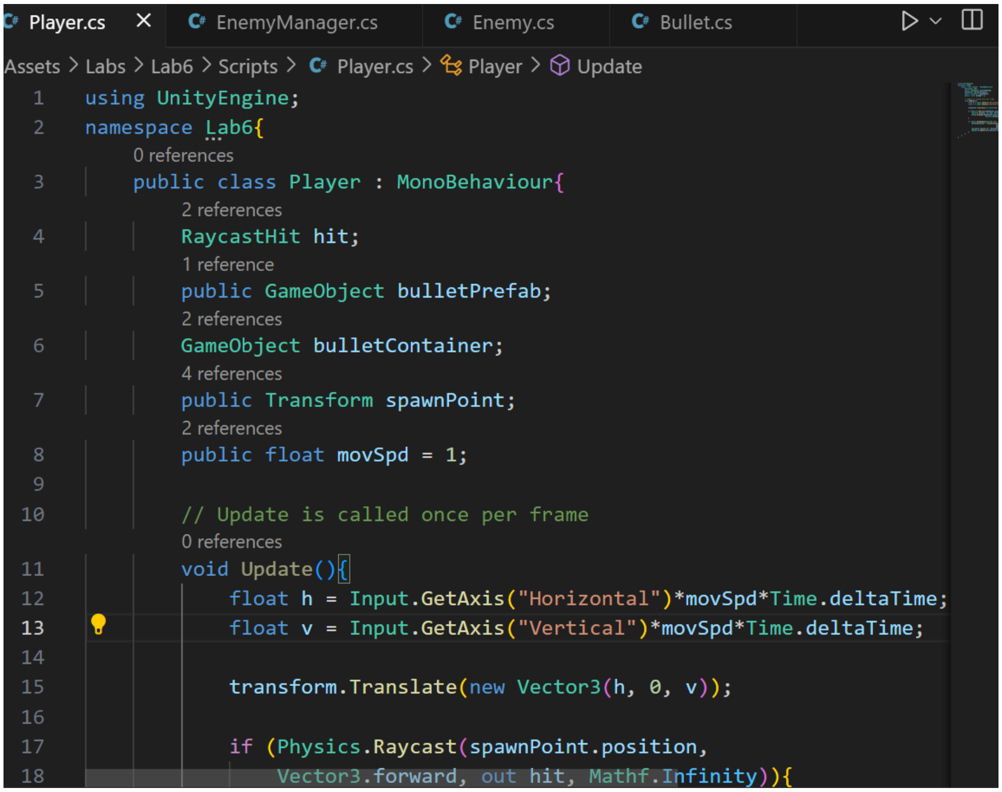
In this lab we further explored the Unity API.

Tasks:

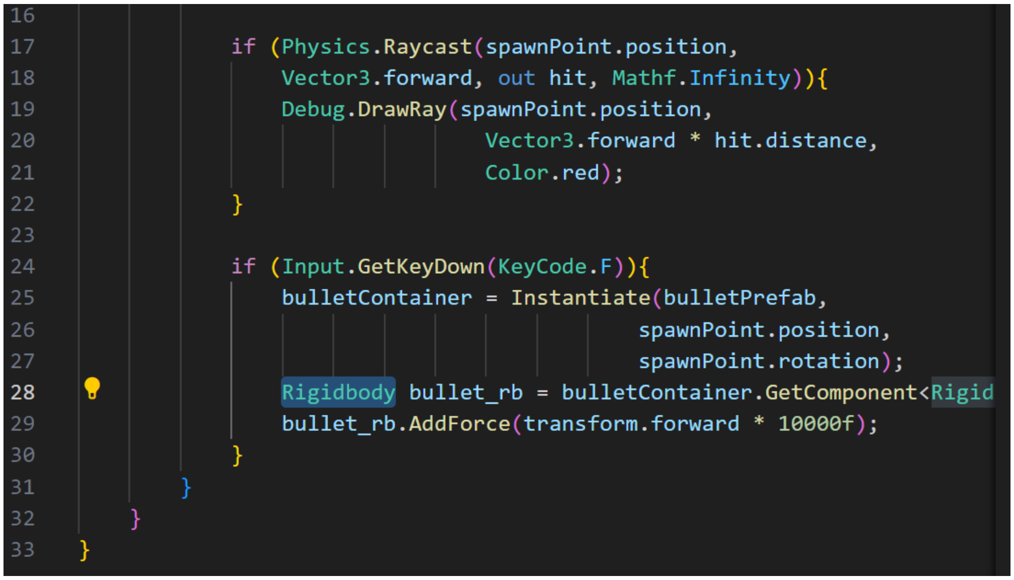
- Open/create a Unity scene.
- Create a player cube that moves forward, backward, left, and right using axis.
- The camera in the scene should be set to a top-down view.
- The player should be able to shoot bullets the bullets must destroy themselves after 2 seconds.
- Create 3 enemies on the plane.
- When the bullets hit the enemies, the enemies must turn first yellow and then red and the then destroy themselves with 1 second interval between each behavior .
- Create an array of these strings (Dead!, Killed!. Defeated!) when the enemy dies any of these messages should print on console “Randomly”

Code:

Player class

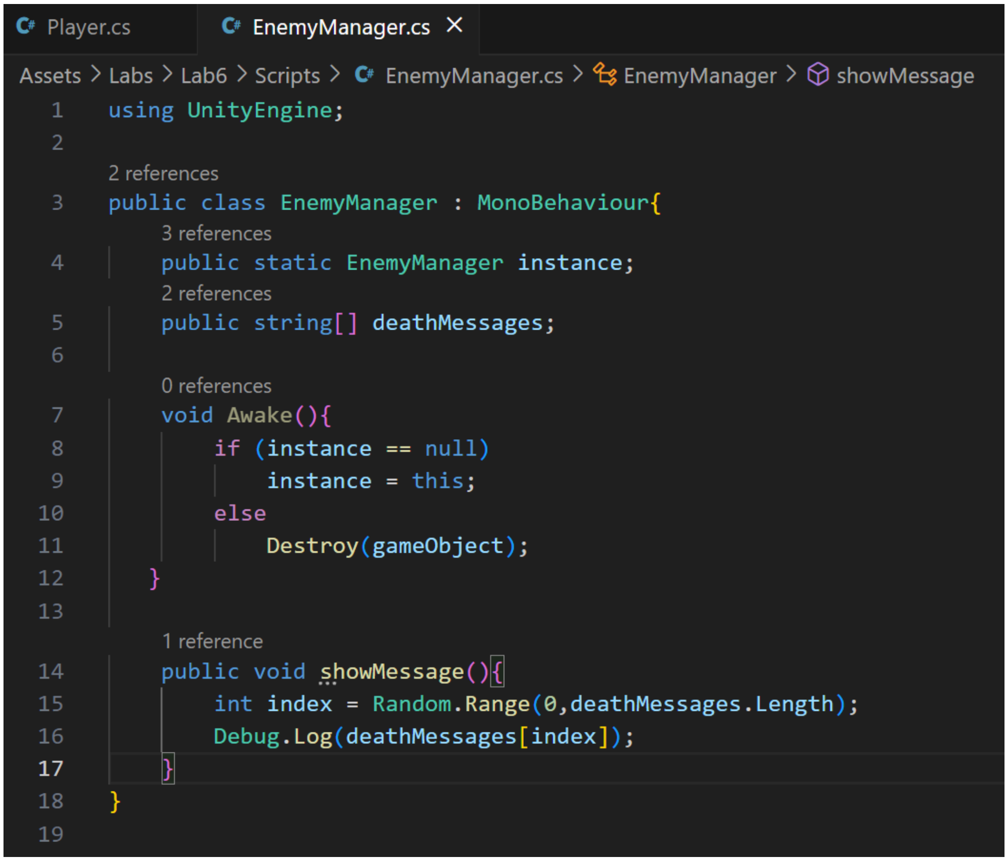

Player.cs
EnemyManager.cs
Enemy.cs
Bullet.cs
Assets > Labs > Lab6 > Scripts > Player.cs > Player > Update

```
using UnityEngine;
namespace Lab6{
    public class Player : MonoBehaviour{
        RaycastHit hit;
        public GameObject bulletPrefab;
        GameObject bulletContainer;
        public Transform spawnPoint;
        public float movSpd = 1;

        // Update is called once per frame
        void Update(){
            float h = Input.GetAxis("Horizontal")*movSpd*Time.deltaTime;
            float v = Input.GetAxis("Vertical")*movSpd*Time.deltaTime;

            transform.Translate(new Vector3(h, 0, v));

            if (Physics.Raycast(spawnPoint.position,
                Vector3.forward, out hit, Mathf.Infinity)){
```



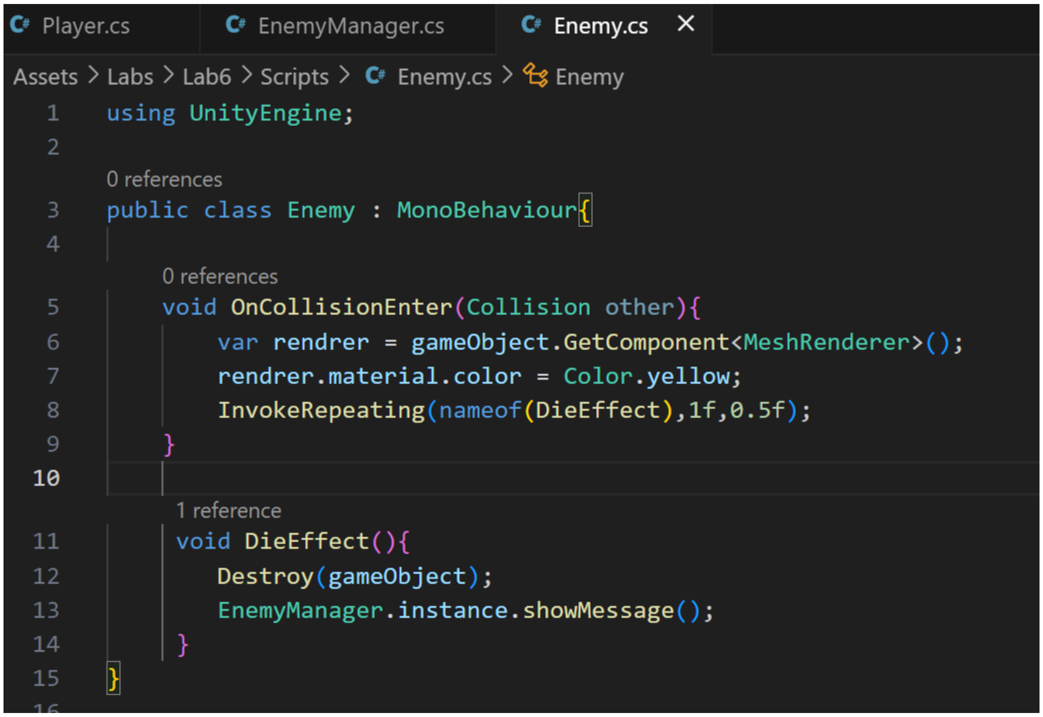
```
            if (Physics.Raycast(spawnPoint.position,
                Vector3.forward, out hit, Mathf.Infinity)){
                Debug.DrawRay(spawnPoint.position,
                                Vector3.forward * hit.distance,
                                Color.red);
            }

            if (Input.GetKeyDown(KeyCode.F)){
                bulletContainer = Instantiate(bulletPrefab,
                                            spawnPoint.position,
                                            spawnPoint.rotation);
                Rigidbody bullet_rb = bulletContainer.GetComponent<Rigid
                bullet_rb.AddForce(transform.forward * 10000f);
            }
        }
    }
}
```

EnemyManager class


Player.cs
EnemyManager.cs ×
Assets > Labs > Lab6 > Scripts > EnemyManager.cs > EnemyManager > showMessage

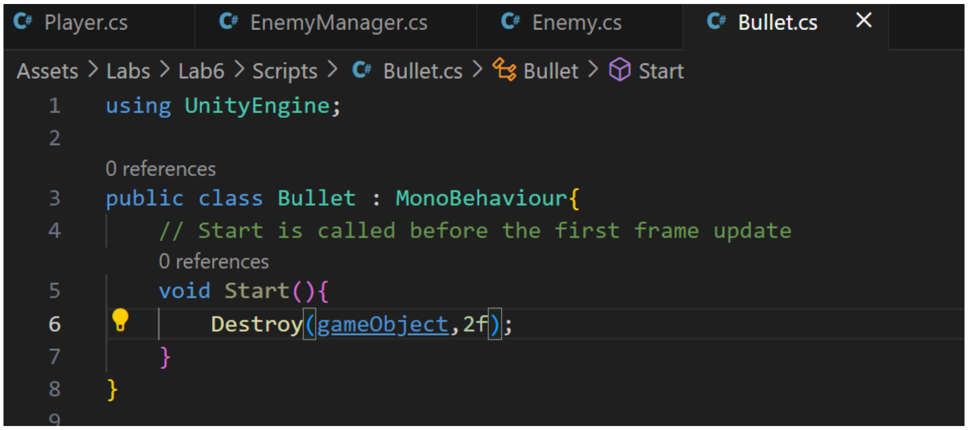
```
using UnityEngine;

2 references
public class EnemyManager : MonoBehaviour{
    3 references
    public static EnemyManager instance;
    2 references
    public string[] deathMessages;

    0 references
    void Awake(){
        if (instance == null)
            instance = this;
        else
            Destroy(gameObject);
   }

    1 reference
    public void showMessage(){
        int index = Random.Range(0,deathMessages.Length);
        Debug.Log(deathMessages[index]);
    }
}

```

Enemy class



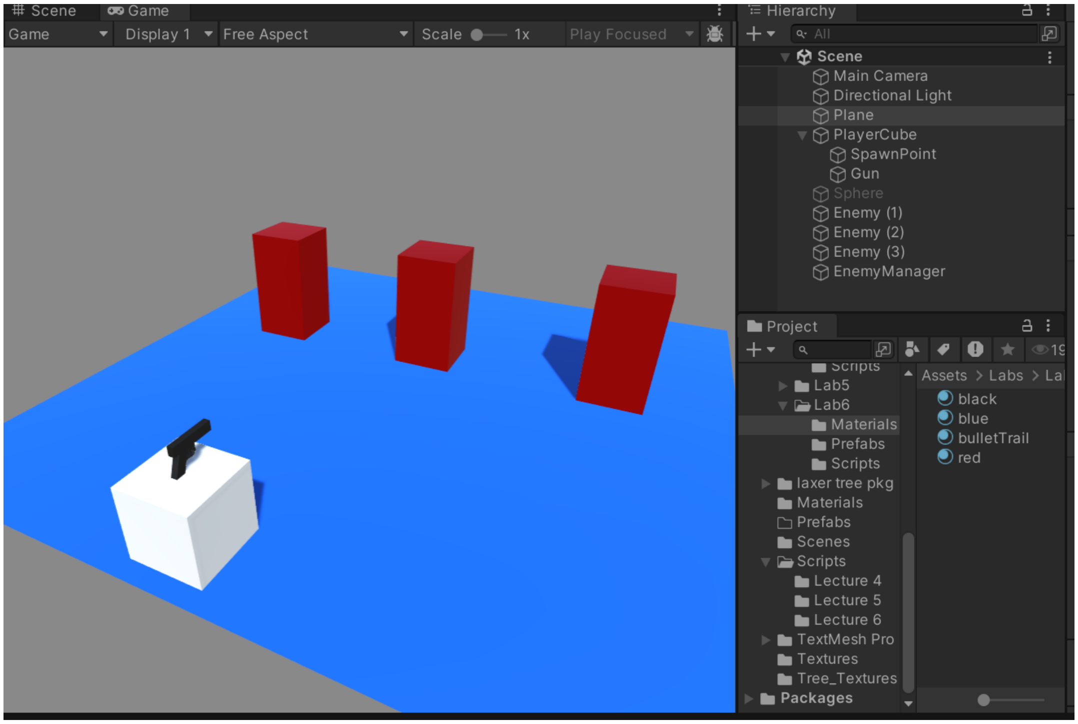
```
using UnityEngine;

public class Enemy : MonoBehaviour{

    void OnCollisionEnter(Collision other){
        var rendrer = gameObject.GetComponent<MeshRenderer>();
        rendrer.material.color = Color.yellow;
        InvokeRepeating(nameof(DieEffect),1f,0.5f);
    }

     void DieEffect(){
        Destroy(gameObject);
        EnemyManager.instance.showMessage();
     }
}
```

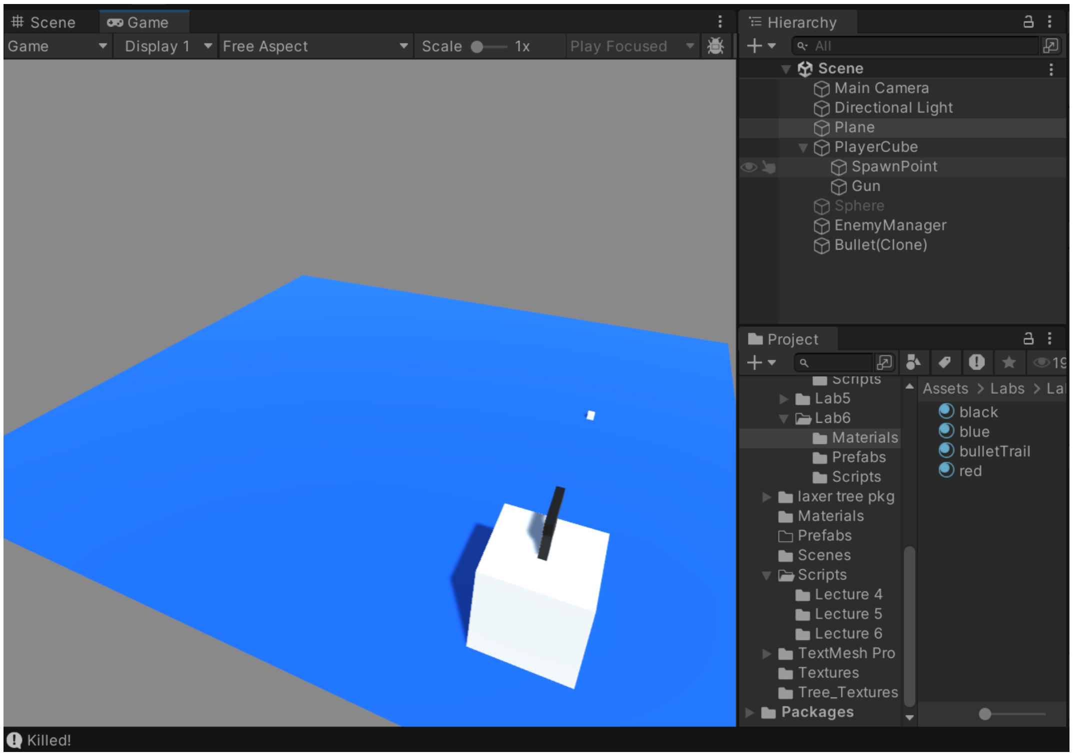
Enemy class



```
using UnityEngine;

public class Bullet : MonoBehaviour{
    // Start is called before the first frame update
    void Start(){
        Destroy(gameObject,2f);
    }
}
```

Output:





```
Killed!
```