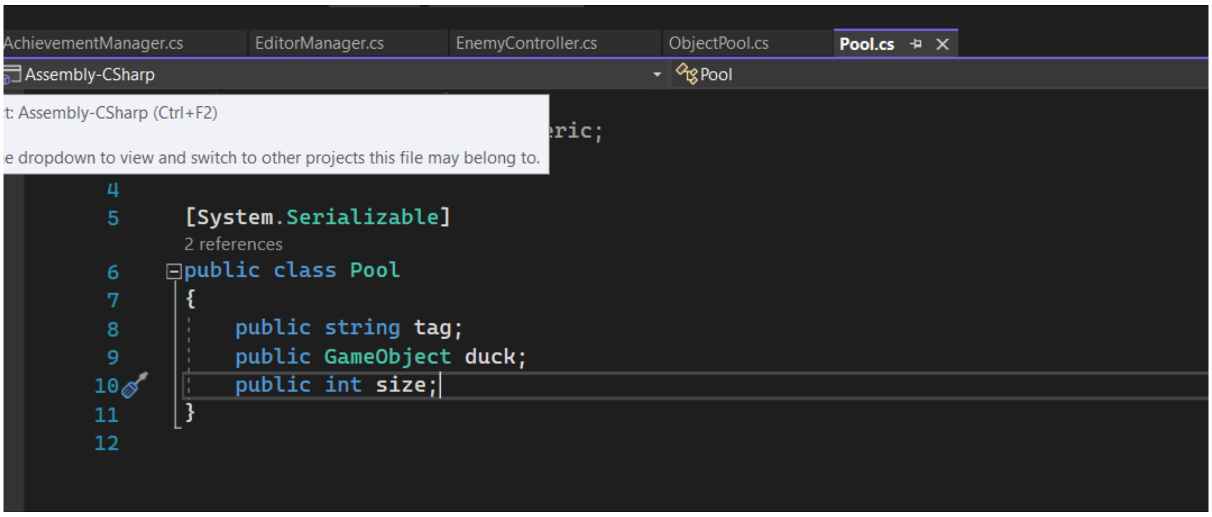
DUCK HUNT QUESTIONS

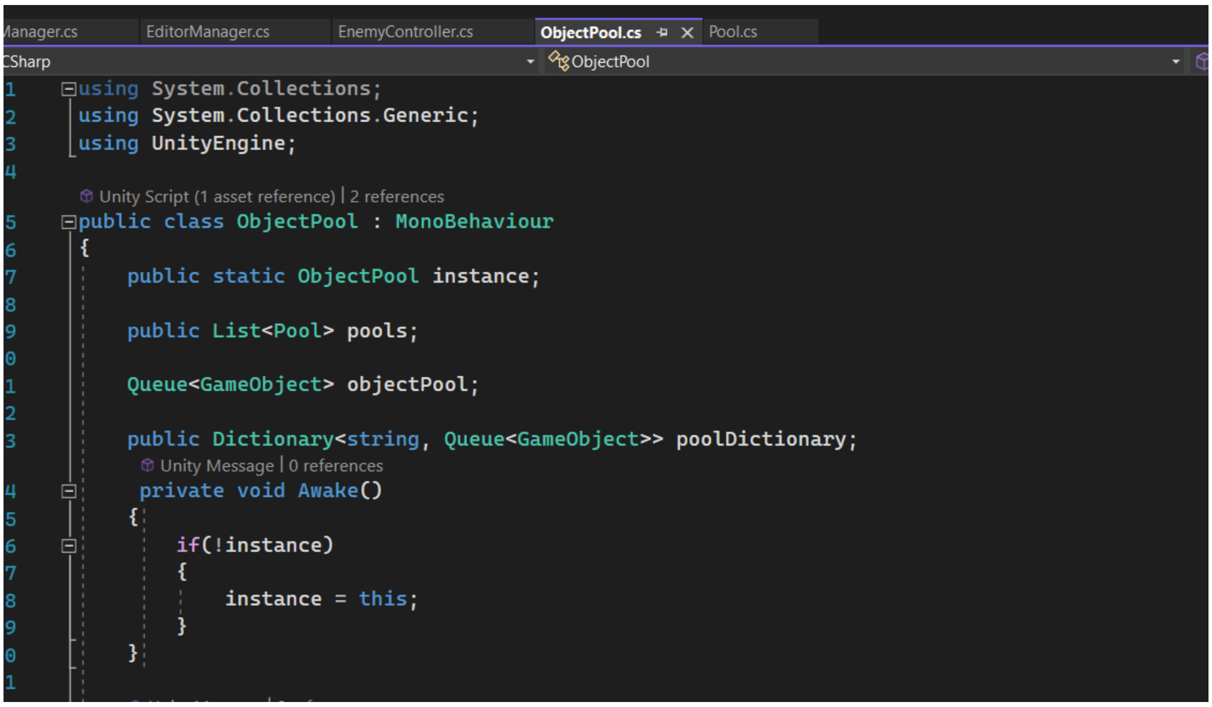
Object Pooling

I implemented object pooling by first creating a script called pool which saves our tags, GameObject and size of our ducks. This is where we will be giving all the information in our Game Manager Script.



```
eric;

[System.Serializable]
public class Pool
{
    public string tag;
    public GameObject duck;
    public int size;
}
```

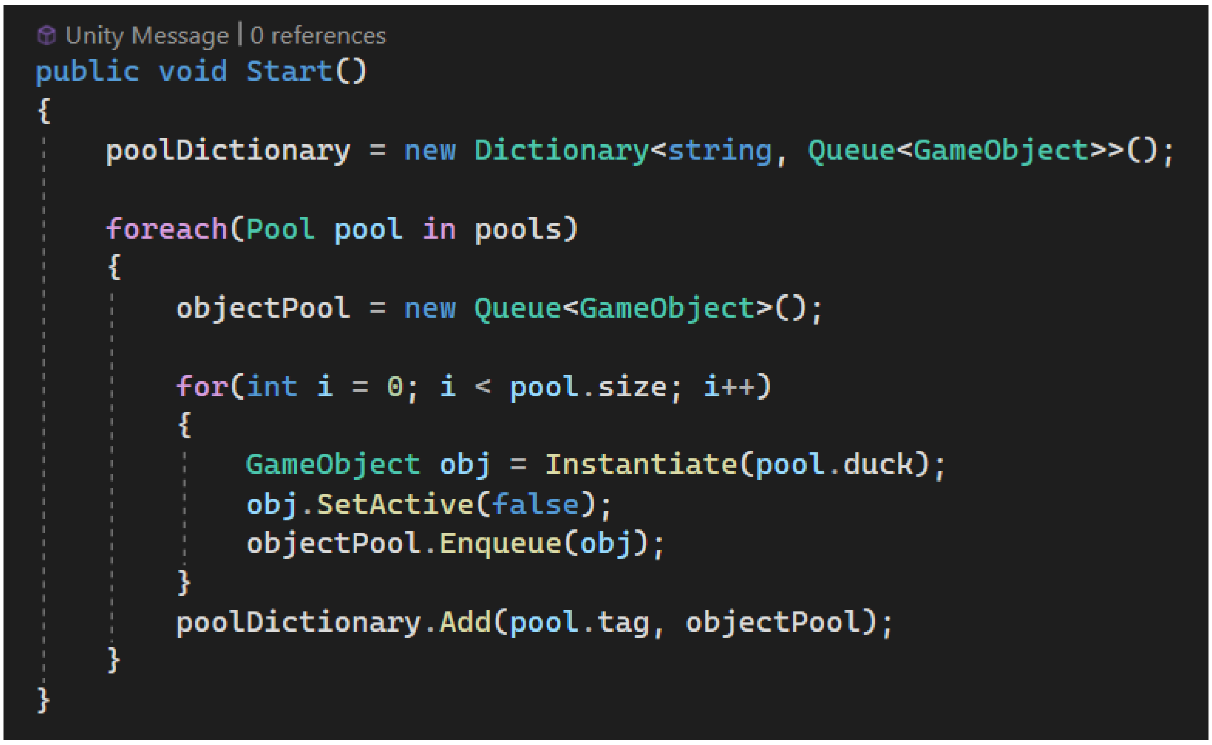
One creating our Pool Script We actually created our ObjectPool script where we will be creating our objects



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ObjectPool : MonoBehaviour
{
    public static ObjectPool instance;

    public List<Pool> pools;

    Queue<GameObject> objectPool;

    public Dictionary<string, Queue<GameObject>> poolDictionary;
     private void Awake()
    {
        if(!instance)
        {
            instance = this;
        }
    }
```

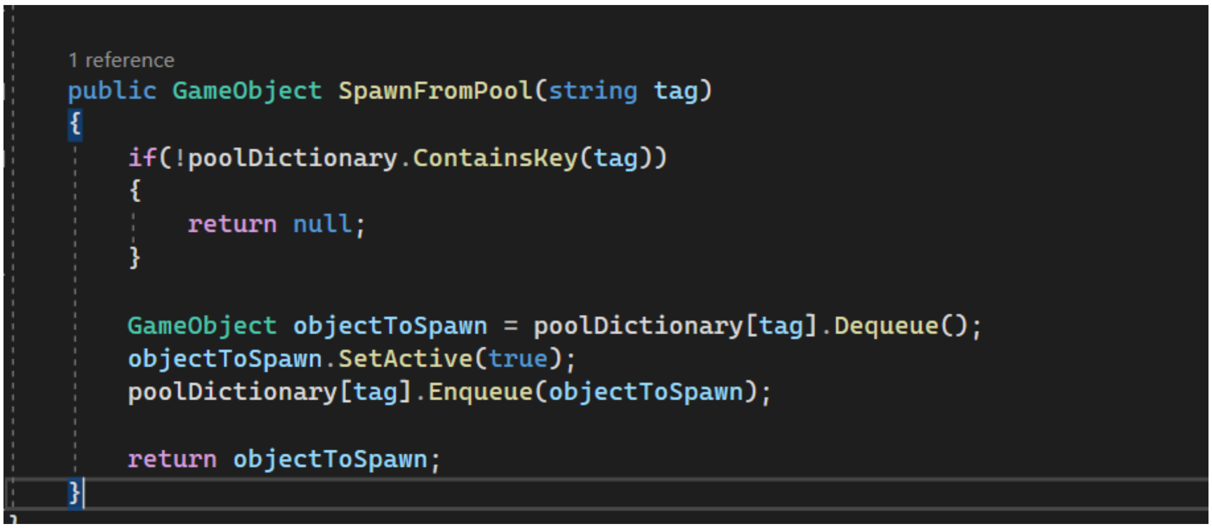
First we create a list of our Ducks then create an instance of our game object using a singleton.

In our Start Function we created a dictionary of our Gameobject in the Que we made and used a Foreach to see how many objects we have in our pool



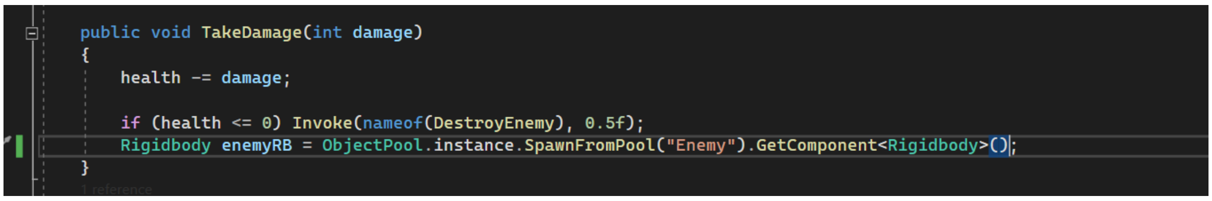
```
Unity Message | 0 references
public void Start()
{
    poolDictionary = new Dictionary<string, Queue<GameObject>>();

    foreach(Pool pool in pools)
    {
        objectPool = new Queue<GameObject>();

        for(int i = 0; i < pool.size; i++)
        {
            GameObject obj = Instantiate(pool.duck);
            obj.SetActive(false);
            objectPool.Enqueue(obj);
        }
        poolDictionary.Add(pool.tag, objectPool);
    }
}
```

Then We created a Function called Object to spawn which takes a string of our Ducks Tag. Once we Deque our Object we want to create a new object right after and spawn in our enemy



```
1 reference
public GameObject SpawnFromPool(string tag)
{
    if(!poolDictionary.ContainsKey(tag))
    {
        return null;
    }

    GameObject objectToSpawn = poolDictionary[tag].Dequeue();
    objectToSpawn.SetActive(true);
    poolDictionary[tag].Enqueue(objectToSpawn);

    return objectToSpawn;
}
```

Then in our Enemy Controller script once our enemy dies we spawn another enemy from our SpawnFromPoo function in our ObjectPool Script



```
public void TakeDamage(int damage)
{
    health -= damage;

    if (health <= 0) Invoke(nameof(DestroyEnemy), 0.5f);
    Rigidbody enemyRB = ObjectPool.instance.SpawnFromPool("Enemy").GetComponent<Rigidbody>();
}
```

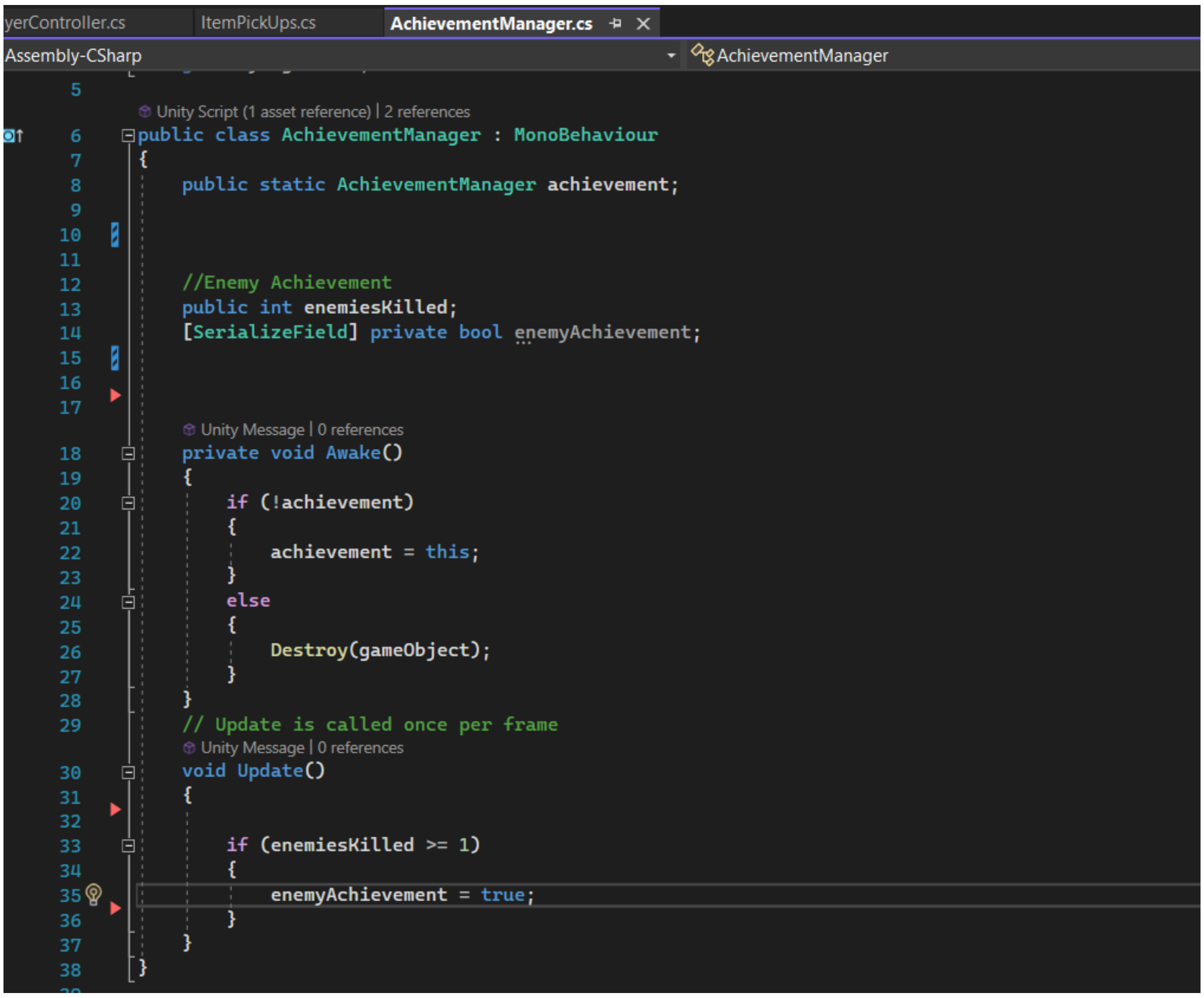
This Optimises our scene because instead of just creating new instances of game objects every single time we just create one and load and reload them when needed. This method will increase our performance for our game.

Command Pattern

I Couldn't Create A Command Pattern for this part. But If I were to implement it I would use a command script which would invert player movement from My Player Controller Whenever the player would miss their two shots. This would be done by just counting how many times the player has shot their gun and using a conditional statement we can apply this effect to the player. This Benefits the designated Game because by making the game more challenging alongside with creating a more fun experience.

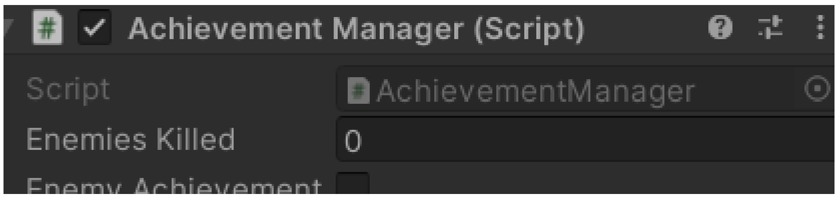
Question 7.

In My game I implemented Management system that tracks achievements for killing a certain amount of enemies GameManger Contains a Script called Achievement Manager which is a singleton that keeps track of how many enemies have been killed

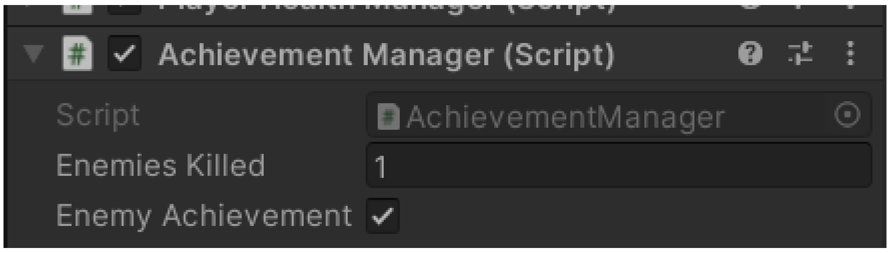

yerController.cs
ItemPickUps.cs
AchievementManager.cs
Assembly-CSharp
AchievementManager
Unity Script (1 asset reference) | 2 references
Unity Message | 0 references
Unity Message | 0 references

```
public class AchievementManager : MonoBehaviour
{
    public static AchievementManager achievement;



    //Enemy Achievement
    public int enemiesKilled;
    [SerializeField] private bool enemyAchievement;



    private void Awake()
    {
        if (!achievement)
        {
            achievement = this;
        }
        else
        {
            Destroy(gameObject);
        }
    }
    // Update is called once per frame
    void Update()
    {

        if (enemiesKilled >= 1)
        {
            enemyAchievement = true;
        }
    }
}
```

And if Enemy has been killed then the achievement will be Awarded



```
Script              AchievementManager
Enemies Killed      0
Enemy Achievement   
```



```
Script              AchievementManager
Enemies Killed      1
Enemy Achievement   ✓
```