INFR 3110 Game Engine Design and Implementation

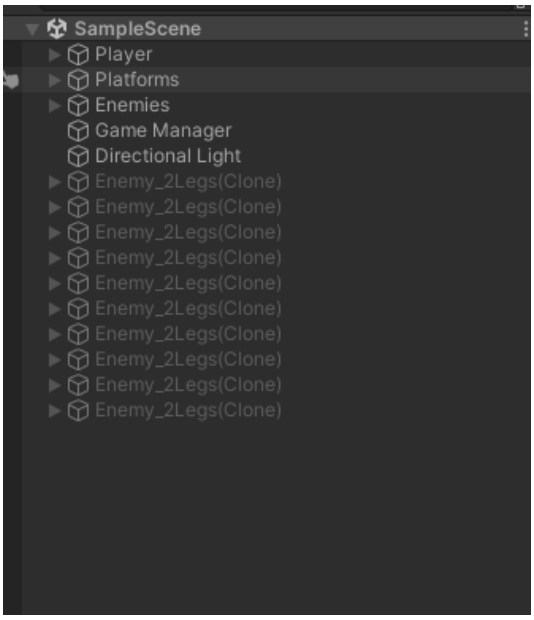
Final Exam

Practical Portion

Object Pooling:

Firstly, a class called ObjectPooler was created, along with a Pool class. The Pool class holds data like the tag (in this case we use “Enemy” because the goal is to spawn more “ghosts”), the prefab (again, we use an Enemy prefab), and size (the size of the pool, can be any number but I use 10). In the ObjectPooler class, we make a list called pools, a queue of game objects called objectPool, and lastly a Dictionary.

Essentially, we create a list of Enemy objects that instantiate in the beginning, and places them in a queue. The goal of this is to increase performance, as we create all these objects at the start instead of during runtime, which is an expensive process.




I chose to activate the enemies in the PlayerController script, specifically in the Attack function, so that when the player uses left-click, an enemy prefab is spawned at the camera position.

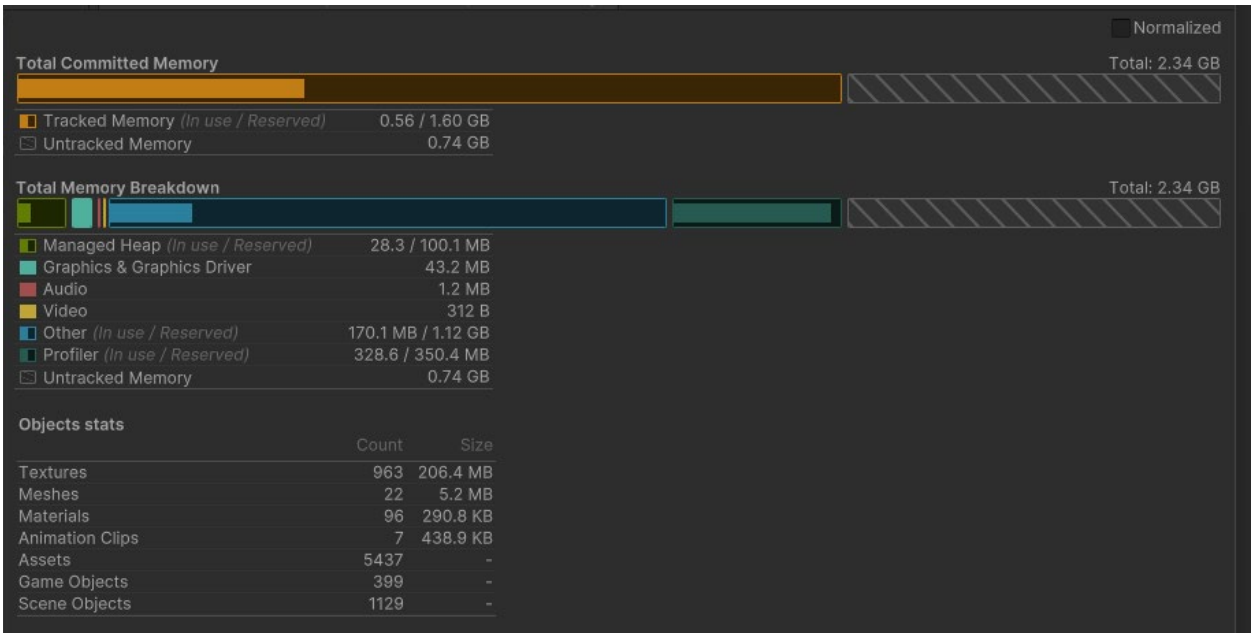
```
1 reference
private void Attack()
{
    // If player is not attacking, player can initialize the attack
    if(!isAttacking)
    {
        animator.SetTrigger("isAttacking");
        StartCoroutine(InitialiseAttack());

        //for object pooler
        Rigidbody enemyRb = ObjectPooler.instance.SpawnFromPool("Enemy", playerCamera.transform.position, Quaternion.identity).GetComponent<Rigidbody>();
    }
}
```

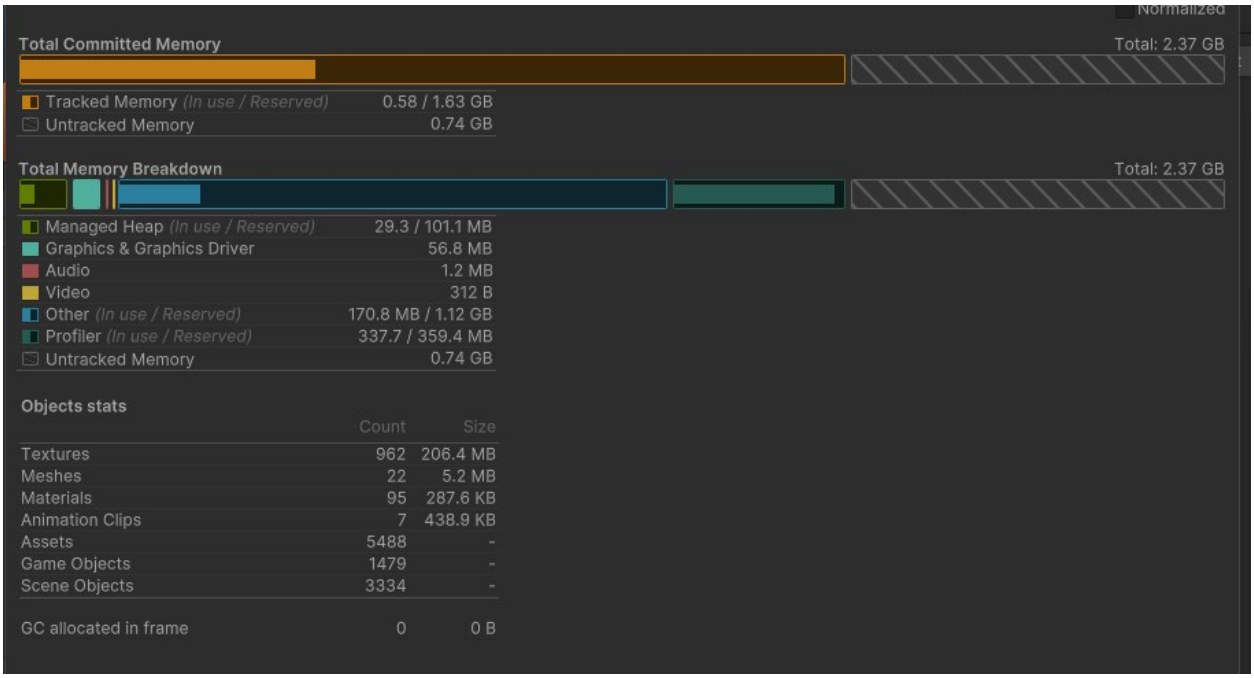
Now, each time the player left-clicks, an enemy is spawned. When the list runs out, which in our case, is after 10 enemies, the first active object is moved, and then the second, and so on.

This is what the memory looks like in the profiler:

List size of 10




List size of 50



Question 7

Singleton for ScoreManager.

The reasoning for the ScoreManager is simply to keep track of how many enemies the player kills. Score managers are very common in games and this game is no exception.

A singleton is used because only one instance is needed, and it provides a global access point.

The ScoreManager class is very simple, with one function ChangeScore that increase the score value by 1 and then prints it to debug .