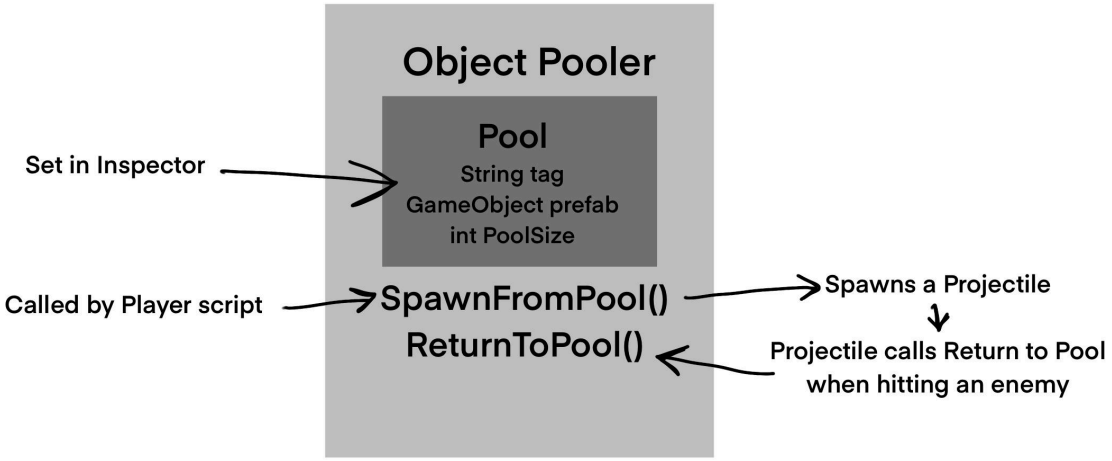
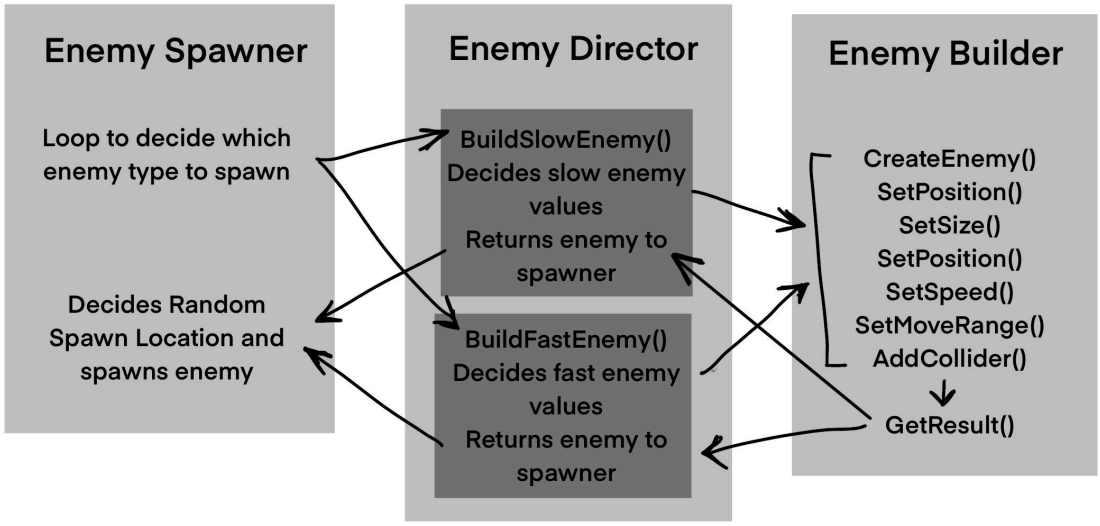
Object Pooler Pattern

The ObjectPooler class has a public subclass for a Pool, which allows the Main class to create a list to manage multiple pools with different objects in each. The list is exposed to the inspector, where the prefab to spawn, the tag of the prefab, and the size of the pool can all be set. In the Start method, The object pooler instantiates the prefab according to the size of the pool, then setting them inactive and adding them to a queue. The object pooler class also defines a SpawnFromPool function as well as a ReturnToPool function, which can be called from any other objects that wish to perform either of those actions. The SpawnFromPool function allows its caller to define the location and rotation of the object when spawning, then sets the object to be active, and dequeues the object. The ReturnToPool function does the opposite, setting the object to be inactive and requeuing the object, rather than destroying the object. Currently the script lives in an empty GameObject called ObjectPooler, and only handles the spawning of projectiles.



Builder Pattern

The Builder pattern consists of 3 scripts, the EnemyBuilder, the EnemyDirector, and the EnemySpawner. The EnemyBuilder script defines a function for creating a new Enemy object, adding the appropriate components to the enemy. It also defines functions for modifying various aspects of the enemy it creates, such as the movement speed, the movement range, the size, or the sprite. The final function in this script returns the enemy after it has been built using the other functions. The EnemyDirector script acts as the caller for all of the scripts in the EnemyBuilder script, and defines two functions for creating two types of enemies, one fast and small enemy, and one slow and large enemy. It decides how to modify each type of enemy's variables, and then returns that result from the EnemyBuilder class to the caller of the EnemyDirector's functions. The EnemySpawner class acts as the caller for the EnemyBuilder's functions, which has a simple for loop within the Start function to decide whether to spawn a fast or slow enemy based on if the index of the loop is even or odd.



Observer Pattern

In the script EnemyMovement, I added a public delegate void OnEnemyDeath() and a public static event OnEnemyDeath onEnemyDeath. I then would invoke onEnemyDeath in void OnDestroy. In the script ScoreKeeper, I have an int score and a method AddToScore, which adds ‘one’ to the score. OnEnable, I subscribe AddToScore to onEnemyDeath so that when onEnemyDeath is invoked, it calls AddToScore. OnDisable, I unsubscribe AddToScore from onEnemyDeath to avoid memory leaks.