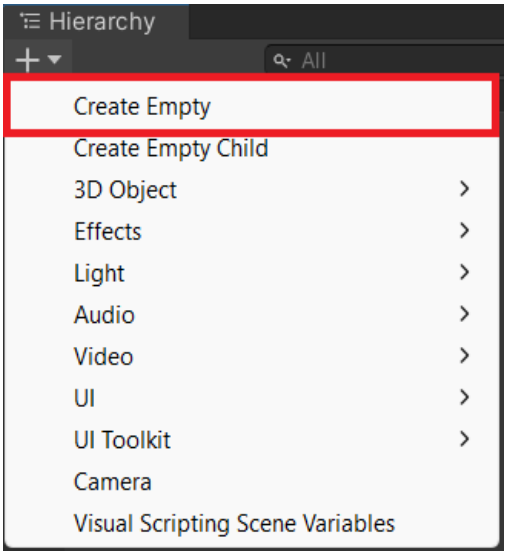
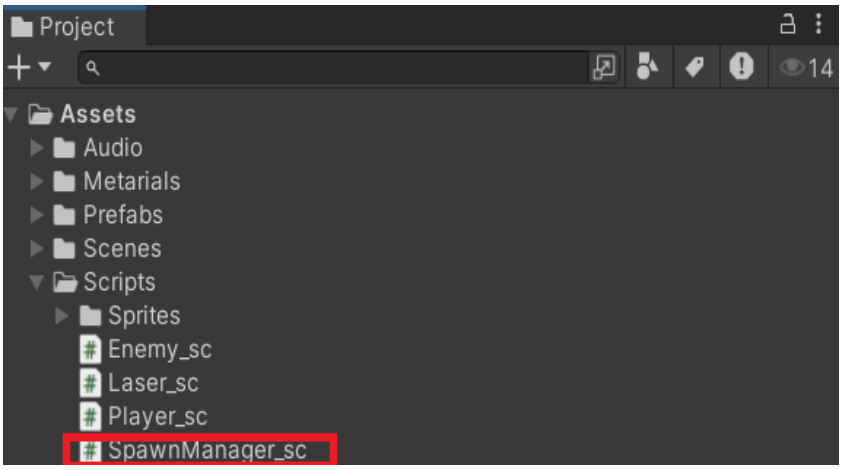
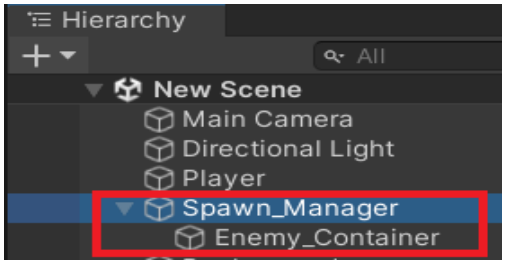
Spawn Manager Nesnesi, Düzenli Enemy Üretme ve CoRoutine Kullanımı

Öncelikle Hierarchy panelindeki “+” simgesine tıklayarak Create Empty ile boş bir nesne oluşturup Spawn_Manager ismini veriyoruz. Daha sonra script kalsörümüze bir adet SpawnManager_sc dosyası oluşturuyoruz.





Ardından aynı şekilde bir EnemyContainer nesnesi oluşturuyoruz ve bu nesnemizi SpawnManager nesnesinin içine sürükleyerek Enemy Container nesnensini SpawnManager nesnesinin bir çocuğu(child) yapıyoruz.




```
[SerializeField]
1 reference
GameObject enemyPrefab;
[SerializeField]
1 reference
GameObject enemyContainer;
```

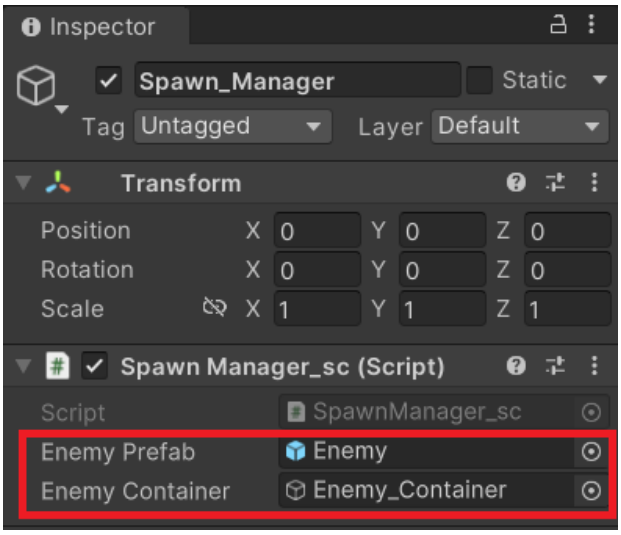
SpawnManager_sc dosyasının içine öncelikle yandaki şekildeki gibi enemyPrefab ve enemyContainer olacak şekilde 2 adet GameObject nesnesi oluşturuyoruz. Daha sonra IEnumerator tipinde SpawnRoutine adında bir fonksiyon oluşturuyoruz. Bu fonksiyonun içine ise bir while döngüsü açıyoruz. Döngünün içinde her tekrarda rastgele bir konum ve bu konumda oluşacak yeni bir enemy nesnesi üretiyoruz. Rastgele olan konumumuzu sahne üst sınırının üzerinde nesne oluştuğunda kullanıcı göremeyecek şekilde ama sınıra yakın bir kısımda aynı zamanda yatay eksende sahen max sınırları içerisinde olacak şekilde oluşturuyoruz. Daha sonra bu enemy nesnesini enemyContainer nesnesinin bir çocuğu(child) olacak şekilde bağlıyoruz. Ardından yield return new WatForSecond ile while döngüsünü istenilen süre miktarı kadar olacak şekilde donduruyoruz. Bu süre bitince tekrardan döngü işlemine devam edecek böylelikle enemy nesneleri istenilen süre aralıklarıyla üretilecek (burada 5 sn).

```
IEnumerator SpawnRoutine(){ //* Random Enemy Generator Function
    while(!stopSpawn){
        Vector3 position = new Vector3(Random.Range(-9.4f,9.4f),7.4f,0);
        GameObject new_enemy = Instantiate(enemyPrefab, position, Quaternion.
        identity);
        new_enemy.transform.parent = enemyContainer.transform;
        yield return new WaitForSeconds(5.0f);
    }
}
```

```
void Start()
{
    StartCoroutine("SpawnRoutine");
}
```

Daha sonra bu fonksiyonumuzu start fonksiyonu içinde çağırıyoruz. Zaten bir while döngüsü ile çalıştığından update fonksiyonu içerisinde çağırmıyoruz.

Son olarak unity uygulamasından SpawnManager_sc dosyasını Spawn_Manager nesnesine sürükleyerek bağlıyoruz sonra Spawn_Manager nesnesini seçip enemyPrefab nesnesini ve enemyContainer nesnesini ınspector panelindeki yerlerine sürüklüyoruz. Böylelikle gerekli bağlantılar kurulmuş olacak.



Tüm bu durumlar ile çok sayıda oluşacak enemy nesneleri bir ebeveyn(parent) nesnesinin içinde toplanmış ve kontrol edilebilir olacak.

Oyuncu Yok Olduğunda Spawn İşleminin Durdurulması

Oyuncu yok olduğunda spwan işleminin durması için while döngüsünün durması gerekmekte. Bunun için öncelikle SpawnManager_sc dosyasının içine bir adet boolean tipinde sopSpawn değişkeni oluşturuyor ve değerini false yapıyoruz. Ardından public olacak şekilde OnPlayerDeath fonksiyonu oluşturuyoruz. Bu fonksiyon stopSpawn değişkeninin değerini True yapacak şekilde olacak ve fonksiyonu çağırmak için player nesnesinin yok olması gerektiğinden bunu player yok olacağında çağırmalıyız. Bu durumda bu fonksiyon player nesnesi tarafıondan çağırılabilmeli. Bu sebeple public bir fonksiyon olacak.

```
bool stopSpawn = false;
```

```
public void OnPlayerDeath(){
    stopSpawn = true;
}
```

Son olarak stopSpawn değişkenini while döngüsünün koşulu olarak kullanıyoruz.

```
IEnumerator SpawnRoutine(){
    while(!stopSpawn){
```

Daha sonra player_sc dosyasının içine bir adet SpawnManager_sc değişkeni oluşturuyoruz.

```
SpawnManager_sc spawnManager_sc;
```

```
void FindSpawner(){
    spawnManager_sc =GameObject.Find("Spawn_Manager").
    GetComponent<SpawnManager_sc>();
    if(spawnManager_sc == null){
        Debug.Log("Spawn_Manager could not be find");
    }
}
```

Ardından player_sc dosyasının içerisinde FindSpawner adına bir fonksiyon oluşturuyoruz be içnde spwanManager_sc değişkenine, GameObject.Find() fonksiyonu ile sahnedeki Spwan Manager nesnesini sekildeki gibi ekliyoruz.

Daha sonra kontrol amaçlı bir if bloğu ile eğer işlem başarılı değil ise bir mesaj gönderiyoruz.

Bu fonksiyonu ise start fonksiyonu içerisinde çağırıyoruz.

```
public void Damage(){
    this.playerHealth--;
    if(playerHealth == 0){
        if(spawnManager_sc != null)
            spawnManager_sc.OnPlayerDeath();
        Destroy(this.gameObject);
    }
}
```

Ardından daha önceden oluşturduğumuz Damage() fonksiyonunun içerisine yandaki gibi eleme yapıyoruz. Burada player yok olmadan hemen önce spawnManager_sc değişkeni ile OnPlayerDeath fonksiyonuna ulaşarak while döngüsünü kırarak ve enemy üretimini durduruyoruz.