Ad: Yusuf

Soyad: Güney

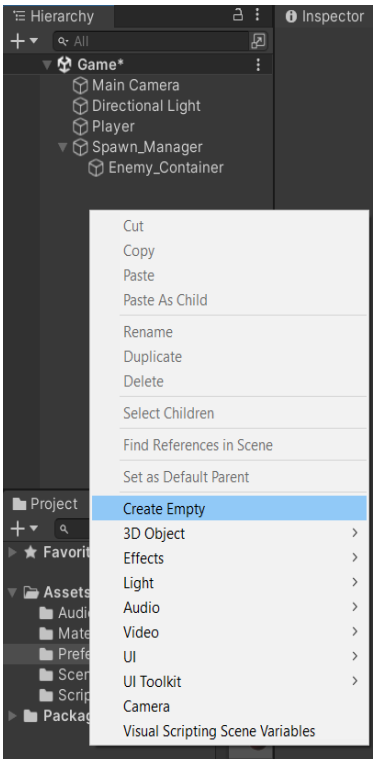
Numara: 22360859041

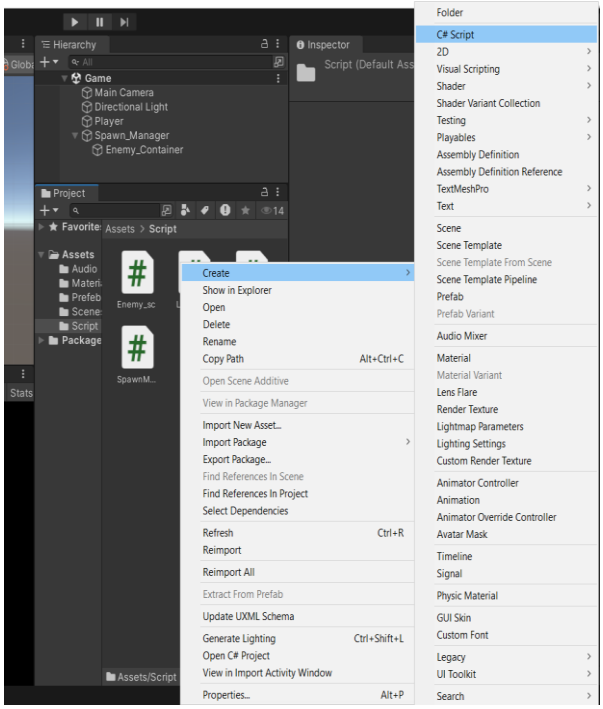
Ders: Oyun Programlama Hafta 3 Rapor

1-Spawn Manager boş oyun nesnesi ve script oluşturma

Hierarchy sekmesine sağ tık yapıp **CreatEmpty** seçerek boş oyun nesnesini oluşturuyoruz daha sonra ismini **Spawn_Manager** olarak değiştiriyoruz.

Scripti ise **Assets** klasörünün altındaki **script** dosyasının içerisinde sağ tık yaparak script dosyamızı oluşturabiliriz. Dosyanın adını **SpawnManager_sc** olarak belirliyoruz**.**





2-Spawn Manager script'i ile her 5 saniyede bir enemy üretme (spawn)

Gerekli tanımlamalar aşağıda gözükmektedir.

```
[SerializeField]
1 reference
GameObject enemyPrefab;

[SerializeField]
1 reference
GameObject enemyContainer;

2 references
bool stopSpawning = false;
```

```
IEnumerator SpawnRoutine(){

    while(stopSpawning == false){
        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);
    }
}
```

Bu kod düşmanları belirli aralıklarla sahnede rastgele pozisyonlarda üretmek için kullanılıyor. Kodunuzun nasıl çalıştığını adım adım açıklayalım:

Coroutine Tanımlaması: IEnumerator, Unity'de bir **coroutine**'i tanımlamak için kullanılan veri tipidir. Coroutine'ler, belirli bir işlemi zamana yayarak çalıştırmaya yarar bu örnekte ise düşmanları 5 saniyede bir üretmek için kullanılıyor.

stopSpawning değişkeni false olduğu sürece dönen bir while döngüsü oluşturuyoruz. Eğer stopSpawning true yapılırsa döngü duracak ve düşman üretimi sona erecek.

Vector3 position = new Vector3(Random.Range(-9.4f, 9.4f), 7.4f, 0) Bu kod satırı oluşacak enemy nesnesinin X ekseninde rastgele bir pozisyonda oluşmasını sağlıyor. Y ekseni değeri enemy nesnesinin oluşacağı yüksekliği belirtiyor z değeri 0 çünkü 2D bir sahne kullanıyoruz.

GameObject new_enemy = Instantiate(enemyPrefab, position, Quaternion.identity) **Instantiate** fonksiyonu ile enemyPrefab adı verilen, önceden belirlenmiş bir düşman prefab'inden yeni bir nesne üretiliyor. **Quaternion.identity** kullanılarak düşmanın rotasyonu varsayılan şekilde kalıyor.

new_enemy.transform.parent = enemyContainer.transform Yeni oluşturulan düşman nesnesi enemyContainer adlı bir nesnenin alt nesnesi yapılıyor. Bu, düşmanları organize etmek veya sahnedeki yerini belirlemek için yapılıyor.

yield return new WaitForSeconds(5.0f) her düşman üretildikten sonra döngü 5 saniye bekliyor ve sonra yeni düşman üretiliyor.

Bu fonksiyonun çalışmasını diğer soruda gösteriyoruz.

3-CoRoutine kullanımı

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

Döngü bir kere çalışacağı için start fonksiyonunu içerisinde **SpawnRoutine** fonksiyonu çalıştırılıyor.

4-Çok sayıda Enemy oluşacağı için bunların bir parent container oyun nesnesi için toplanması

```
new_enemy.transform.parent = enemyContainer.transform;
```

Daha önceki açıklanan ikinci soruda bu sorunun açıklamasını yapmıştık. Yeni oluşturulan düşman nesnesi enemyContainer adlı bir nesnenin alt nesnesi yapılıyor. Bu, düşmanları organize etmek veya sahnedeki yerini belirlemek için yapılıyor.

5-Oyuncu yok olduğunda spawn işleminin durdurulması için fonksiyon yazılması

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

Bu fonksiyon döngüyü kırmak için yazılan bir fonksiyondur. Oyuncu yok olduğunda **player_sc** içerisindeki **Damage** fonksiyonunda çalışacak ve düşman gelmesi durdurulacaktır.

6-Bu fonksiyonun Player_sc sınıfından oyuncu yok olduğunda çağırılması

```
0 references
public void Damage(){

    lives--;

    if(lives < 1){
        if(spawnManager_sc != null)
            spawnManager_sc.OnPlayerDeath();
        Destroy(this.gameObject);
    }
}
```

Önceki haftalarda yazdığımız **Damage** adlı fonksiyonun içerisinde ikinci bir if bloğu oluşturuyoruz. Bu if bloğu **spawnManager_sc**'nin null olup olmadığı kontrol edilir. Eğer **spawnManager_sc** null değilse, yani düzgün şekilde atanmışsa, o zaman **OnPlayerDeath()** fonksiyonu çağrılır. Bu kontrol, **spawnManager_sc** değişkeninin atanıp atanmadığını anlamak içindir, böylece kodun çökmesini engeller. **OnPlayerDeath()** fonksiyonu oyuncu yok olmadan hemen önce çağırılır bu da düşmanların gelmesinin durmasını sağlar.

Kodun tamamına Github sayfası üzerinden ulaşabilirsiniz

Github Linki: