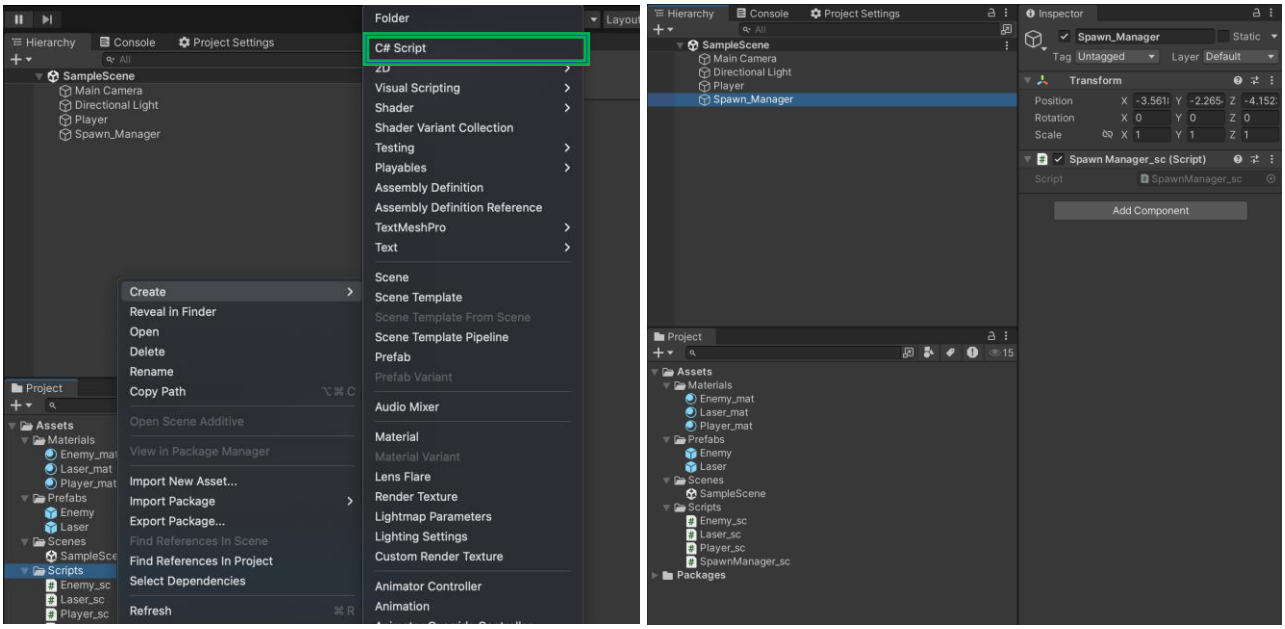
Oyun Programlama Ödevi – Hafta 4

Ad: Eren
Soyad: Köse
Numara: 22360859075
GitHub Kod Linki: https://github.com/erennkose/btu-oyun-programlama/tree/main/Hafta4

- ## Spawn Manager Boş Oyun Nesnesi ve Script'i Oluşturma

  Spawn Manager boş oyun nesnesi oluşturmak için öncelikle **Hierarchy** bölümünde boşluğa sağ tık atıyoruz ve **Create Empty** seçeneğini seçiyoruz ve adını **Spawn_Manager** olarak ayarlıyoruz. Bu şekilde Spawn Manager boş oyun nesnesi oluşturmuş oluyoruz. Ardından Assets klasörü altındaki Scripts klasörüne sağ tıklayıp **Create > C# Script** seçeneklerini izliyoruz ve scriptimize **SpawnManager_sc** adını veriyoruz. Bu ürettiğimiz scripti **sürükle-bırakla** nesnemizin üzerine bırakıyoruz ve nesnemize scriptimizi entegre etmiş oluyoruz.



- ## Spawn Manager Script'i ile Her 5 Saniyede Bir Enemy Üretme (Spawn)

  Scriptimizi oluşturup Spawn_Manager nesnemize entegre ettikten sonra scriptimize çift tıklıyoruz ve scripti kod editörümüzde açıyoruz. Öncelikle spawn kontrolünü sağlamak için fonksiyonların dışında, classın içinde bir bool değişken oluşturuyoruz ve default olarak **false** değerini atıyoruz. Ayrıca enemy prefabimize de erişmemiz gerekeceğinden **enemyPrefab** adında bir GameObject oluşturuyoruz. Aynı zamanda oluşturulan bu enemyleri toplamak için kullanacağımız **enemyContainer** adında başka bir GameObject daha oluşturuyoruz. enemyPrefab ve enemyContainer değişkenlerine ileride Unity uygulamasından erişmemiz gerekeceğinden önlerine **[SerializeField]** ekliyoruz. Ardından Unity uygulamamıza geri dönüp **Spawn_Manager** nesnemize tıkladıktan sonra **Inspector** kısmında gözüken scriptimizin altında seçmemiz gereken seçenekler olacak. Enemy Prefab yazan yerin sağındaki noktaya tıklayıp **Enemy prefabimizi** seçiyoruz. Aynı şekilde **Enemy Container'ımızı** da seçiyoruz. Bu tanımlamaların ardından kodumuza dönüp bir fonksiyon oluşturacağız. Fonksiyonumuzu coroutine şeklinde kullanacağımızdan dolayı geri dönüş türümüz **IEnumerator** olacak. **stopSpawning** değişkenimiz **false** olduğu müddetçe bu fonksiyon çalışacak şekilde **while** döngüsü ekliyoruz. Bu while içinde de rastgele x pozisyonunda, map üst sınırından yeni enemy nesnesi spawnlanması durumunu sağlıyoruz. Bu yeni enemy nesnesini oluşturmak için **Instantiate** fonksiyonu kullanıyoruz. Bu oluşturmadan sonra oluşturulan her enemy nesnesine nesneleri düzende tutmak amacıyla **enemyContaineri parent** olarak atıyoruz. Fonksiyonun 5 saniyede bir çalışmasını istediğimiz için ise return yaparken **WaitForSeconds** fonksiyonunu kullanarak fonksiyonun 5 saniye beklemesini sağlamış oluyoruz.

```
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);
    }
}
```

```
[SerializeField]
1 başvuru
GameObject enemyPrefab;
[SerializeField]
1 başvuru
GameObject enemyContainer;
2 başvuru
bool stopSpawning = false;
```

- ## CoRoutine Kullanımı

  Coroutine bir işlemi zamana yayarak yapmamızı sağlayan işleme denir. SpawnManager_sc içinde yazdığımız **SpawnRoutine** fonksiyonumuz bir coroutine fonksiyonudur. Bu fonksiyondaki yazdığımız **yield return new WaitForSeconds (5.0f)** kod parçası bu fonksiyonumuzu coroutine yapmamızı sağlar. Bu kod parçası sayesinde çağırıldığında bu fonksiyon 5 saniye aralıkla çalışır. Bu fonksiyonu **Start** kısmında çağıracağız. Bunun sebebi ise SpawnRoutine fonksiyonumuzun her framede değil, 5 saniyede bir çalışmasını istememizdir. Start içerisinde bu coroutine fonksiyonunu direkt çağıramıyoruz. Bu fonksiyonu çağırmak için **StartCoroutine** fonksiyonuna parametre olarak **SpawnRoutine** fonksiyonunu girmemiz gerekiyor. Bu şekilde 5 saniyede bir çalışacak olan SpawnRoutine coroutine fonksiyonumuzu projemizde çağırmış oluyoruz.

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

- ## Çok Sayıda Enemy Oluşacağı İçin Bunların Bir Parent Container Oyun Nesnesi İçin Toplanması

  Her 5 saniyede bir yeni enemy spawnlanacağı yukarıda anlatılmıştı. Teorik olarak düşündüğümüz zaman hiçbir enemy nesnesini öldürmediğimiz sürece çok fazla sayıda enemy nesnesi oluşabilir. Bu da nesneleri kontrol ettiğimiz **Hierarchy** kısmında karışıklık ortaya çıkarır. Bunu engellemek için ise **Enemy_Container** adında bir **boş oyun nesnesi** oluşturuyoruz. Oluşturmak için **Hierarchy** kısmının altındaki boşluğa sağ tık atıp **Create Empty** seçeneğini seçiyoruz. Ardından adını **Enemy_Container** şeklinde ayarlıyoruz. Ardından bu yeni nesnemizi sürükle-bırak yöntemiyle Spawn_Manager nesnemizin üzerine bırakıyoruz. Bu sayede enemy nesnelerimizi **Hierarchy** bölümünde kontrol altında tutabileceğiz ve karışıklığı engelleyeceğiz. Oluşturulan enemyleri bu container içerisine doldurma işlemini ise kod kısmında yapıyoruz. **SpawnManager_sc** scriptindeki **SpawnRoutine** fonksiyonu içerisinde yazdığımız kodlarda oluşturduğumuz yeni enemy nesnelerini yeni bir **GameObjecte** atıyoruz. Bu **GameObjecte .transform.parent** kod parçasını ekleyerek **parent ataması** yapıyoruz. Bu sayede oluşturulan her yeni enemy nesnesi bu container altında oluşturulmuş oluyor.

```
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);
    }
}
```

- ## Oyuncu Yok Olduğunda Spawn İşleminin Durdurulması İçin Fonksiyon Yazılması

  Şu andaki kodumuzda oyuncumuz yok olduğu zaman enemyler oluşturulmaya devam ediyor. Bunu engellemek için oluşturmuş olduğumuz **stopSpawning** değişkenini kullanacağız. Bu durumu sağlamak için öncelikle SpawnManager_sc scriptinde **OnPlayerDeath** adında bir **public** fonksiyon tanımlıyoruz. Bu fonksiyonun tek amacı stopSpawnin bool değişkenini default hali falsedan trueya çevirmektir. Bu değişken **true** olduğunda da SpawnRoutine fonksiyonundaki while döngüsünün şartı artık sağlanmayacak ve spawning duracaktır.

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

  Bu fonksiyon tanımlandıktan sonra çağırmak için **Player_sc** scirptine dönmemiz gerekiyor. Cana göre spawningi Player_sc içerisinde bu fonksiyonu çağırarak durduracağız.

- ## Bu Fonksiyonun Player_sc Sınıfından Oyuncu Yok Olduğunda Çağırılması

Player_sc içerisinden SpawnManager_sc içerisindeki OnPlayerDeath fonksiyonuna ulaşmak için bir SpawnManager_sc nesnesi oluşturuyoruz.

```
SpawnManager_sc spawnManager_Sc;
```

Bu nesneyi oluşturduktan sonra Start içerisinde bu spawnManager_Sc nesnesine **GameObject.Find("Spawn_Manager").GetComponent<SpawnManager_sc>()** kod parçasını atayarak nesnenin SpawnManager_sc içerisindeki fonksiyonları Player_scde kullanabilmemizi sağlamış oluyoruz. Bunun bir kez yapılmasını istediğimiz için bunları **Start** içerisinde yaptık. Ardından bu nesneye ulaşıp ulaşamadığımızı anlamak için bir if koşuluyla bu durumu kontrol ettiriyoruz. Bulamazsak konsoldan bir mesaj yazdırıyoruz.

```
void Start()
{
    spawnManager_Sc = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager_sc>();
    if(spawnManager_Sc == null){
        Debug.Log("Spawn_Manager oyun nesnesi bulunamadı.");
    }
}
```

Eğer **spawnManager_Sc** değişkenimiz null değilse, canımıza göre spawningi durdurmamız gerekir. Can durumunu zaten **Damage** fonksiyonumuzda kontrol ettiğimiz için Spawn_Manager_sc scriptinde bulunan **OnPlayerDeath** fonksiyonunu da **Damage** fonksiyonu içerisinde çağırabiliriz. Canımızın 1'in altında olup olmadığını kontrol ettiğimiz koşul altında **spawnManager_Sc** nesnemiz null değil ise **OnPlayerDeath** fonksiyonumuzu **spawnManager_Sc nesnemiz üzerinden** çağırıyoruz. Bu sayede Player nesnemiz öldüğü zaman enemy nesnesi üretmeyi de durdurmuş oluyoruz.

```
public void Damage(){
    lives--;
    if(lives < 1){
        if(spawnManager_Sc != null){
            spawnManager_Sc.OnPlayerDeath();
        }
        Destroy(this.gameObject);
    }
}
```