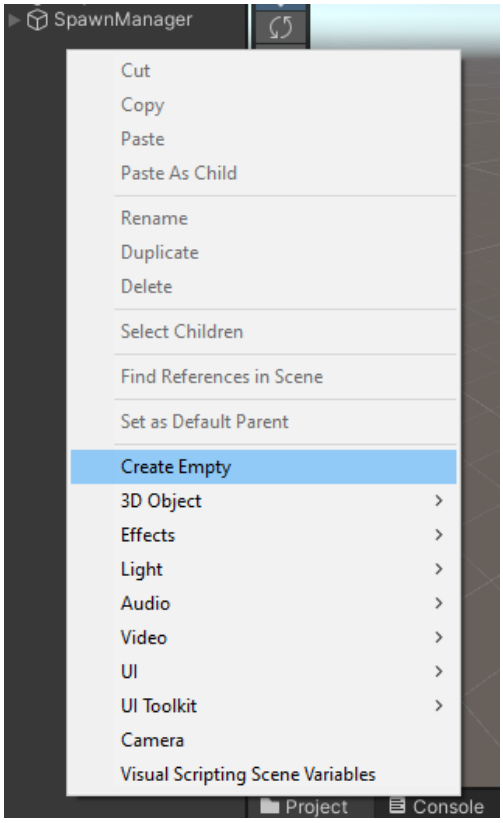
4. Hafta Raporu – Aleks Dulda 21360859025

Spawn Manager boş oyun nesnesi ve script'i oluşturma:

Öncelikle “Hierarchy” kısmında yeni bir boş obje oluşturuyoruz yani “Create Empty” oluşturuyoruz (Bu işlem için “SpawnManager” olarak adlandırılmıştır.)



Bu oluşturduğumuz boş obje bizim yeni oluşacak enemy’ler için bir görünmez obje olacak.

Bu şekilde boş oyun nesnemizi oluşturmuş olduk.

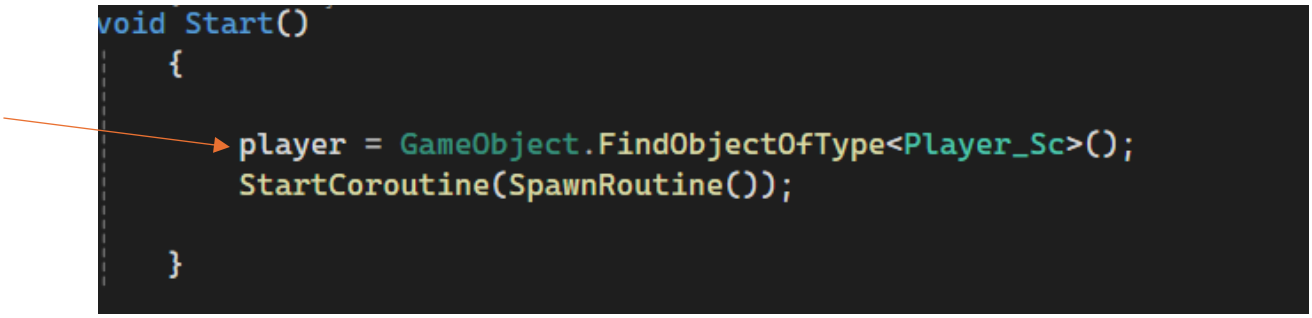
Bu boş oyun nesnesi için SpawnManager_Sc adında bir script dosyası açıyorum ve içinde kodlarımı yazmaya başlıyorum.

```
1 başvuru
IEnumerator SpawnRoutine()
{
    while (player.Health>0)
    {

        Vector3 position = new Vector3(Random.Range(-player.xVal, player.xVal), player.yVal + 2, 0);
        GameObject new_enemy = Instantiate(enemyPrefab, position, Quaternion.identity);
        new_enemy.transform.parent = enemyContainer.transform;
        yield return new WaitForSeconds(5.0f);



    }
}
```

Bu kısımda yaptığım şey düşmanların belirli aralıklarla yaratılmasını sağlayan bir IEnumerator tabanlı "Coroutine" fonksiyonudur.

Fonksiyonun adı SpawnRoutine, ve temel olarak oyuncunun sağlık değeri (player.Health) 0'dan büyük olduğu sürece belirli bir zaman aralığında düşman (enemyPrefab) oluşturur.

Player.Health bilgisini çekebiliyoruz çünkü bu yazdığımız scriptin içine player_Sc dosyasından kalıtım aldım. Bu şekilde player_Sc içinde public tanımladığım bir değişkeni başka scriptlerde kullanabilme olanağı sağladı.



IEnumerator: Coroutine'ler Unity'de uzun süren işlemleri (örn. belirli zaman aralıklarında işlemler yapma) gerçekleştirmek için kullanılır. Bu Coroutine, yield return ifadesi ile belirli bir süre bekleyip döngüyü devam ettirebilir.

Oyuncu yok olduğunda spawn işleminin durdurulması için fonksiyon yazılması:

while (player.Health > 0) Oyuncunun sağlık değeri 0'dan büyük olduğu sürece döngü devam eder. Yani oyuncu **ölmediği** sürece bu döngü çalışır. Ölmediği kısmı önemli çünkü oyuncu ölürse yeni düşman üretilmeyecek.

Tam anlamıyla bir fonksiyon yazmak yerine böyle bir çözüm yolu kullandım.

Vector3 position = new Vector3(Random.Range(-player.xVal, player.xVal), player.yVal + 2, 0);

Bu satır, yeni düşmanın nerede ortaya çıkacağını belirler. Düşman, oyuncunun xVal değerine bağlı olarak rastgele bir yatay pozisyonda ve yVal'nin 2 birim üstünde bir dikey pozisyonda yaratılır. Random.Range(-player.xVal, player.xVal) ifadesi, düşmanın oyuncunun çevresinde rastgele bir x koordinatında yaratılmasını sağlar.

Döngüyü oluşturduğum health verisini çektiğim gibi yeni oluşacak düşmanlarında nerede oluşacağını belirlemek için oyuncunun konumunu kullandım.

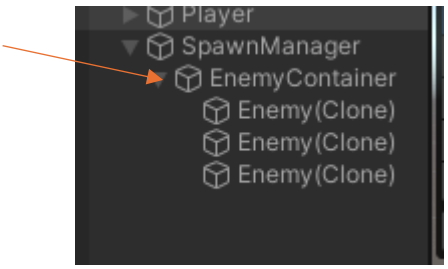
GameObject new_enemy = Instantiate(enemyPrefab, position, Quaternion.identity);

Bu satırda, enemyPrefab isimli prefab (hazır şablon obje) belirlenen position pozisyonunda ve varsayılan rotasyonla (Quaternion.identity) sahneye instantiate (yani kopyalanır) edilir.

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

new_enemy.transform.parent = enemyContainer.transform;

Yeni yaratılan düşman objesi, enemyContainer adlı bir objenin alt objesi (child) olarak atanır. Bu, düşmanları bir arada düzenlemek ve yönetmek için kullanılabilir.



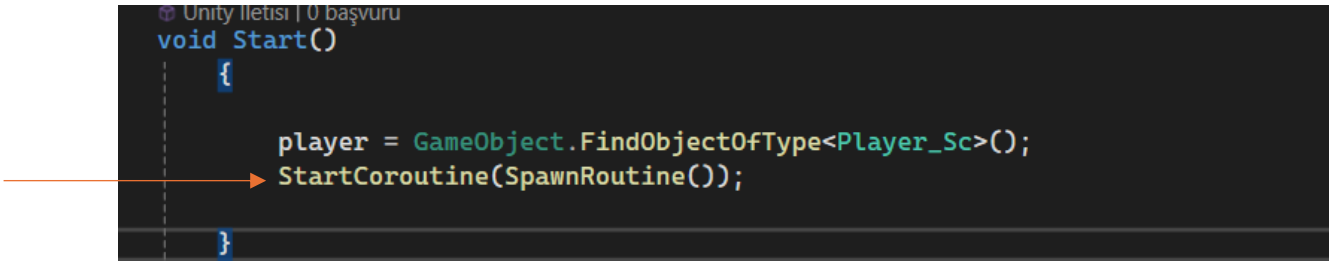
Bu kısımda yine “Create Empty” oluşturuyoruz ve bir parent child ilişkisiyle daha derli bir şekilde çalışmış oluyoruz. Hierarchy kısmını temiz kullanmış oluyoruz.

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

yield return new WaitForSeconds(5.0f);

Her düşman yaratıldıktan sonra Coroutine 5 saniye bekler ve ardından döngü tekrar eder. Böylece düşmanlar 5 saniyede bir yaratılır.

CoRoutine kullanımı



StartCoroutine(SpawnRoutine()); ifadesi, Unity'de coroutine başlatmak için kullanılan bir yöntemdir. Coroutine'ler, belirli işlemleri oyun döngüsünde zamanla gerçekleştirmek için kullanılır. Özellikle, gecikme, animasyon veya bekleme süreleri içeren işlemler yapmak istediğinizde coroutine'ler çok yararlıdır.

StartCoroutine(SpawnRoutine()) ifadesi, SpawnRoutine adlı coroutine'i çalıştırır.

SpawnRoutine bir coroutine yöntemi olarak tanımlanır. Bu yöntemin başında IEnumerator dönüş türü bulunur ve yield return ifadeleri içerir. Bu yield ifadeleri, belirli bir süre beklemek veya bir koşulun gerçekleşmesini beklemek gibi işlemler yapmanıza olanak tanır.

```
IEnumerator SpawnRoutine()
{
    while (player.Health>0)
    {

        Vector3 position = new Vector3(Random.Range(-player.xVal, player.xVal), player.yVal + 2, 0);
        GameObject new_enemy = Instantiate(enemyPrefab, position, Quaternion.identity);
        new_enemy.transform.parent = enemyContainer.transform;
        yield return new WaitForSeconds(5.0f);



    }
}
```

Bu şekilde de oluşacak yeni enemy’lerin pozisyonları ne sıklıkla çıkacakları ayarlanmış oldu.

Kaynakça

https://en.wikipedia.org/wiki/Unity_(game_engine)

https://chatgpt.com/

https://docs.unity3d.com/Manual/index.html

Github link

https://github.com/AleksDulda/GamePrograming

https://github.com/AleksDulda/GamePrograming/tree/main/Rapor/Week_4