Oyun Programlama Ödev 4

1)EnemyContainer

SpawnManager nesnesinin altına oluşturduğumuz boş nesnenin adına EnemyContainer adını veririz. EnemyContainer'ın görevi, sahnedeki düzeni ve hiyerarşiyi temiz tutmaktır. Yani, oyunda oluşturulan tüm düşmanların (enemy) sahne hiyerarşisinde tek bir GameObject altında toplanmasını sağlar.

```
enemy.transform.parent = enemyContainer.transform;  // Senin parentın aslında enemy container
```

Bu satırla yeni üretilen düşman (enemy) objesinin parent'ı enemyContainer olarak atanıyor. Böylece tüm düşmanlar bir klasör altında,EnemyContainer'ın altında, toplanmış olur.

2)stopSpawning

Görevi, yeni düşman üretmeyi (spawn etmeyi) durdurmaktır.

```
    [SerializeField]
    bool stopSpawning = false;
```

```
IEnumerator SpawnRoutine(){
    while(stopSpawning == false){
        Vector3 position = new Vector3(Random.Range(-9.5f, 9.5f),
                                       7.4f,
                                       0);
        GameObject enemy = Instantiate(enemyPrefab, position, Quaternion.identity);
        enemy.transform.parent = enemyContainer.transform;  // Senin parentın aslında enemy container
        yield return new WaitForSeconds(5.0f);  // 5 sn beklemek için
    }
}
```

Bu while döngüsü, stopSpawning false olduğu sürece çalışmaya devam eder. Yani oyun boyunca her 5 saniyede bir yeni düşman üretilir.

```
public void OnPlayerDeath(){

        stopSpawning = true;
    }
}
```

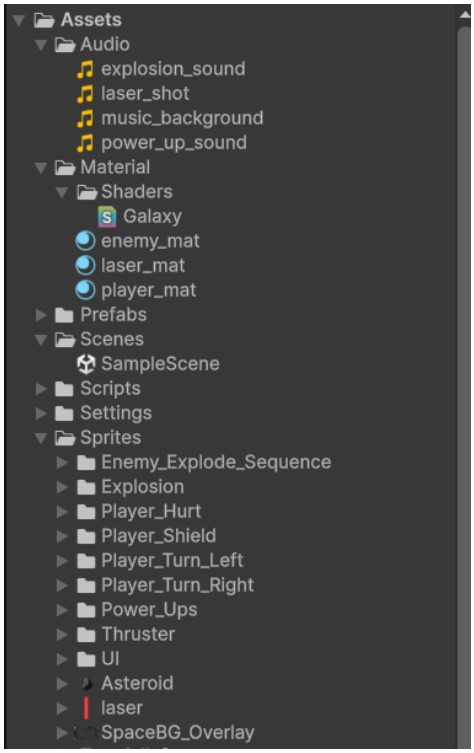
Ama bir noktada (örneğin oyuncu ölünce), başka bir fonksiyondan bu değeri resimdeki gibi yaparsak stopSpawning artık true olur.

while (stopSpawning == false) koşulu artık dönemez. Döngüden çıkar ve artık yeni düşman üretilmez.

3)Asset Ekleme

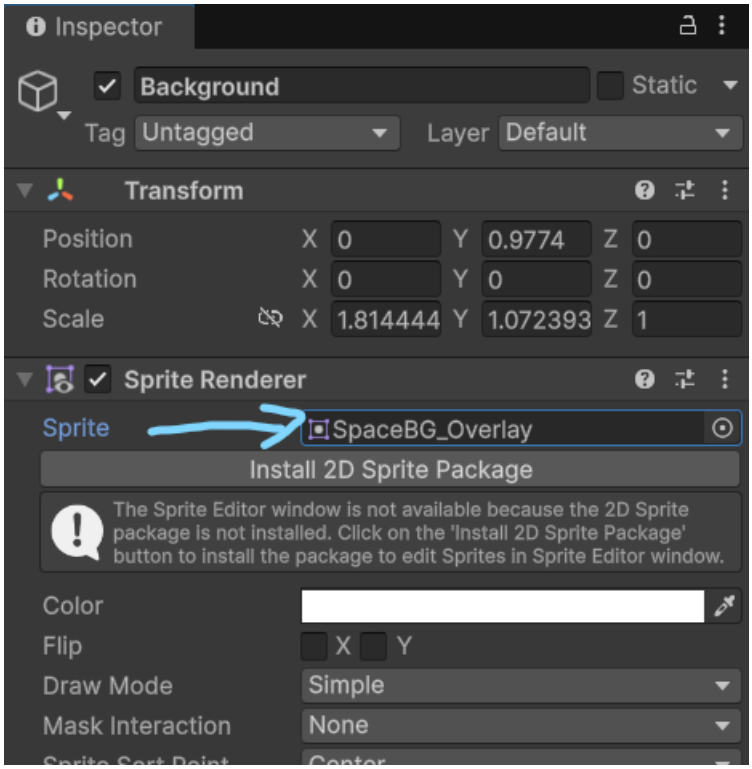
Asset'ler, oyunun görsel olarak tüm yapı taşı sayılabilir. Kod mantığıyla birleşerek, görsel ve ses ögelerini bir araya getirir.

Projemizde kullanacağımız asset dosyalarını (Audio, Sprites, Materials) proje dosyalarına ekleriz. Sprite, 2D oyunlarda karakterleri, arka planları, objeleri temsil eden görsellerdir. Projemizde şuana kadar sadece Sprite assetlerini ekledik.



4)Background Ekleme

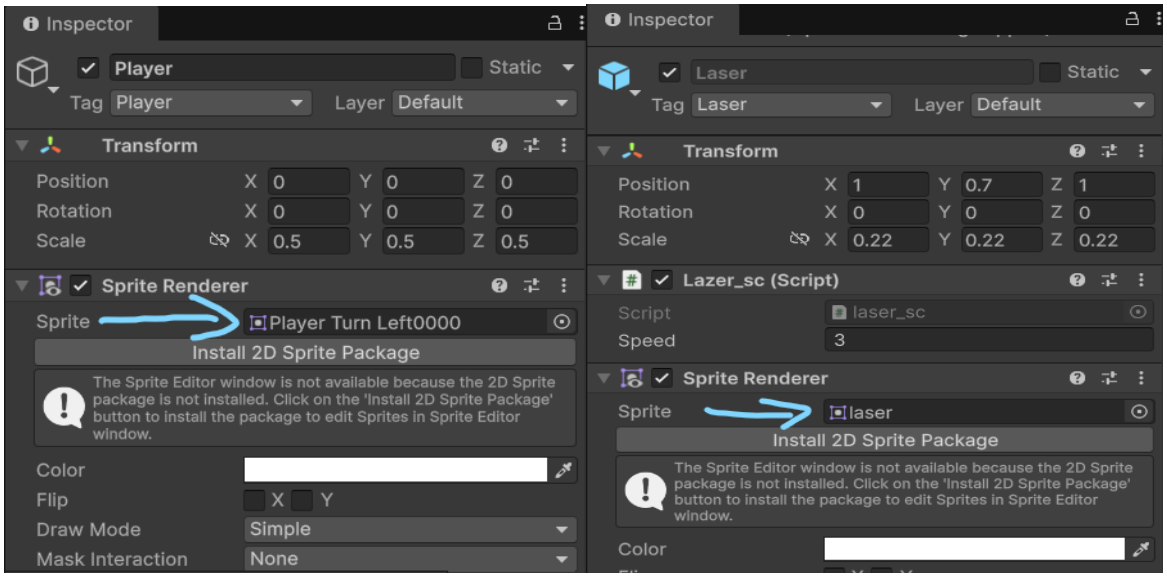
Hierarchy penceresinde sağ tık yapıp Create Empty ile oluşturduğumuz boş nesnemizin adını Background koyarız. Ardından Inspector pencesinde bulunan “Add Component” ile Sprite Renderer componenti ekleriz. Sonrasında proje dosyalarına eklediğimiz sprite assetlerinin içinde bulunan SpaceBG_Overlay dosyasını, eklediğimiz Spite Renderer componentinde bulunan “Sprite” böölümüne sürükleyip bırakırız.

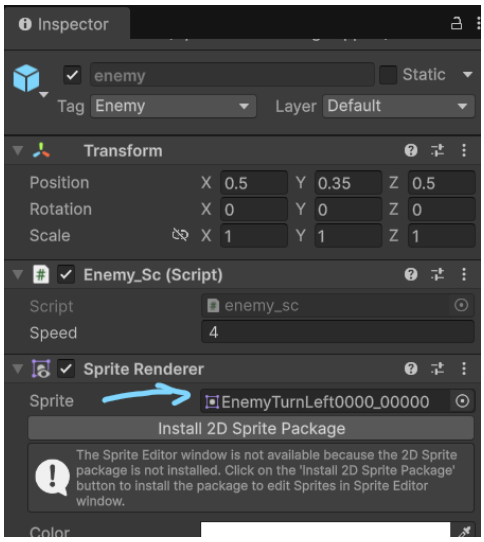


Inspector penceresinde Additional Settings kısmının içinde bulunan Sorting Layer seçeneğinde Add Sorting Layer seçilir ve ilki Background ikincisi Forground olmak üzere iki tane eklenir. Backgroundda olduğumuzdan Sorting layer Background seçilir. Bu, Player ve Enemy için Forground’dur. Böylelikle arkaplanın oyuncu ve düşmanın üstüne gelme olayı çözülür.

5)Player, Enemy ve Laser Asseti Ekleme

Background asseti eklerken yaptığımız süreç tekrarlanır. Aralarındaki tek fark sürüklenip bırakılan asset dosyasıdır





6)Projeyi 3D’den 2D’ye çevirme

Bugüne kadar yaptığımız bütün proje aşamalarında 3D Componentler kullandık ve ona uygun kodlar yazdık. Bu aşamalarla beraber eklediğimiz assetler ile proje 2Dye evrildi. Bu sebeple componentlerimizi ve kodlarımızı da buna uygun şekilde düzeltme ihtiyacı doğdu.

```
    void OnTriggerEnter2D(Collider2D other){

        if(other.tag == "Player"){

            //Yapılan: Player'ın canını bir eksilt
            Player_sc player_sc = other.transform.GetComponent<Player_sc>();
            player_sc.Damage();

            //Player_sc player = new Player_sc();   //Bu sahnedeki player değil klon

            Destroy(this.gameObject);
        }

        else if (other.tag == "Laser"){
            Destroy(other.gameObject);
            Destroy(this.gameObject);
        }
    }
}
```

Kodumuz önceki halinden bu haline güncellendi.

Şuana kadar kullandığımız componentler de 3D olduğundan onlar da kaldırılıp yerine 2D componentler eklenmeli. Bu componentler Box Collider ve Rigidbody componentleridir. Bunlar silinip yerine Box Collider 2D ve Rigidbody2D componentleri eklenmelidir ve buna uygun istenilen ayarlamalar yapılmalıdır. Bütün nesnelerimizde(Player, Enemy, Laser) bu uygulanmalıdır.

Script Dosyalarına Github Hesabımdan Ulaşabilirsiniz:

https://github.com/AsmBrk/Oyun-Programlama-Lab/tree/main/%C3%96dev4