Ad: Yusuf

Soyad: Güney

Numara: 22360859041

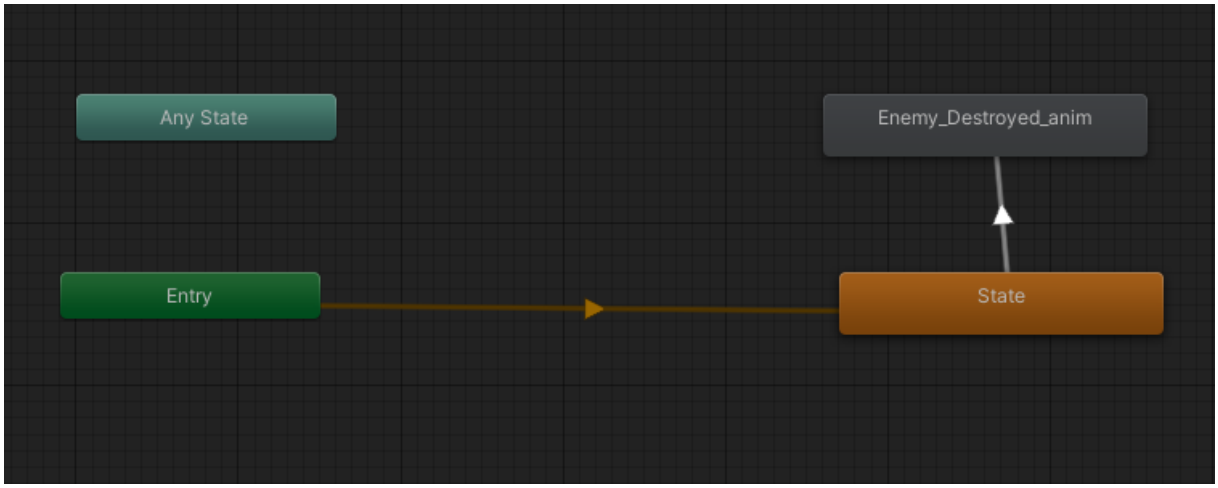
Ders: Oyun Programlama Hafta 9 Rapor

1-Düşman gemileri yok edildiğinde düşman patlama animasyonunu gösterin (loop time ayarı, has exit time ayarı, transition duration ayarı, animation controller yeni state ekleme, default state değiştirme, transition ekleme, trigger ekleme, transition'a condition ekleme, trigger'ın script içinde set edilmesi)

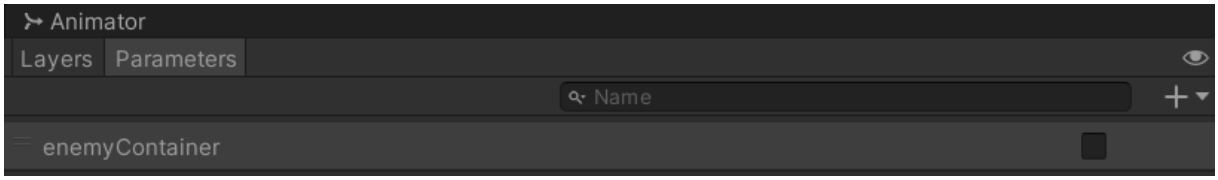
Düşman patlama animasyonunu eklemek için Sprites klasörü içerisindeki **Enemy_Explode_Sequence** klasöründeki patlama nesnelerini kullanarak animasyonu oluşturuyoruz. Animasyonu oluştururken **window penceresinen animation** seçeneğine tıklayarak **animation** penceresini açıyoruz daha sonra **Animations** dosyasının içerisine **Enemy_Destroyed_anim dosyasınını** oluşturuyoruz.

Animasyon ayarlarını yapmak için **Enemy_Destroyed** animasyon dosyasına çift tıklayarak **animator** penceresini açıyoruz.

Sayfaya sağ tıklayarak yeni bir state ekliyoruz ve gerekli bağlantıları aşağıdaki şekildeki gibi yapıyoruz.



Burada Parametres sekmesine gelerek bool veri tipinde enemy container adlı yeni bir değişken oluşturuyoruz bu değişkeni daha sonra patlama aniamsyonu için script dosyasında kullanacağız ekliyoruz.



2-Nesne yok etmede gecikme eklenmesi, hız sabitleme ile yok olan düşman gemisinin zarar vermesinin önüne geçilmesi

Enemy_sc dosyasındaki **OnTriggerEnter2D** fonksiyonunda, daha önce düşman gemilerini **Destroy(this.gameObject)** komutuyla anında yok ediyorduk. Ancak, bu komuta 2.5f gecikme ekleyerek ve geminin hızını 0'a ayarlayarak, düşman gemisinin ateş edildikten sonra olduğu yerde durmasını ve bu sırada patlamasını sağladık. Böylece, düşman gemisi patlarken oyuncumuzun hasar almasının önüne geçiyoruz.

```
0 references
void OnTriggerEnter2D(Collider2D other){
    if(other.tag == "Player"){
        Player_sc playersc = other.GetComponent<Player_sc>();
        playersc.Damage();

        anim.SetTrigger("OnEnemyDeath");
        Destroy(this.gameObject, 2.5f);
        speed = 0;
    }
    else if(other.tag == "Laser"){
        Destroy(other.gameObject);
        if (player_sc != null){
            player_sc.UpdateScore(10);
        }
        anim.SetTrigger("OnEnemyDeath");
        Destroy(this.gameObject, 2.5f);
        speed = 0;
    }
}
```

3-Asteroid ekleme (collider, rigid body, vs.), asteroid'in z ekseninde sürekli hareket etmesi

Sprites klasöründen asteroid nesnesini ekrana sürüklüyoruz. Daha sonra **Sprite Renderer** bölümünden **Sorting Layer** ayarını **Foreground** olarak değiştiriyoruz. Yeni oluşturduğumuz **Asteroid_sc** scriptini sürükleyerek asteroid nesnesinin üzerine bırakıyoruz. Ardından **Add Component** seçeneğiyle **Circle Collider 2D** ekliyor, **Is Trigger** seçeneğini aktif hale getiriyor ve **Edit Collider** seçeneğiyle nesnemizin etki alanını düzenliyoruz. Son olarak, nesneye **Rigidbody 2D** bileşenini ekliyoruz.

Asteroid_sc scriptinin içerisinde cismin z ekseninde işlemini **update()** fonksiyonunu içerisinde gerçekleştiriyoruz.

```
void Update()
{
    transform.Rotate(Vector3.forward * rotateSpeed * Time.deltaTime);
}

0 references
```

4-asteroid patlama animasyonu (prefab oluşturarak script ile dinamik eklenme ve yok edİlme)

Sprites klasöründeki **Explosion** dosyasının içerisindeki ilk nesneyi sahneye sürüklüyoruz. Bu nesnenin **Sorting Layer** ayarını **Foreground**, **Order in Layer** değerini ise 1 olarak ayarlıyoruz. Daha sonra **Explosion** dosyasındaki diğer nesneleri kullanarak patlama animasyonunu oluşturuyoruz. Oluşturduğumuz **Explosion** nesnesini **Prefabs** klasörüne sürükleyerek bir prefab haline getiriyoruz.

Asteroidin patlama işlemini **Asteroid_sc** dosyasında gerçekleştiriyoruz.

```
0 references
void OnTriggerEnter2D(Collider2D other){


    if(other.tag == "Laser"){
        Instantiate(explosionPrefab, transform.position, Quaternion.identity);
        Destroy(other.gameObject);
        spawnManager_sc.StartSpawning();
        Destroy(this.gameObject);

    }
}
```

Burada eğer laser nesnesiyle çarpışma gerçekleştiyse önce laser daha sonra nesne yok edilir ve animasyon aktif hâle getirilir.

5-Spawn routine'lerinin asteroid yok edildikten 3 saniye sonra başlaması

explosion_sc adında yeni bir script oluşturuyoruz start fonksiyonunu içerisinde Destroy(this.gameObject, 3.0f); bu komutu çalıştırarak asteroid nesnesi yok edildiktene sonra patlama efekti de bittikten sonra 3 saniye bekleyerek oyunu başlatma işlemini gerçekleştiriyoruz.

```
void Start()
{
    Destroy(this.gameObject, 3.0f);
}
```

6-Thruster ekleme

Sprites klasöründeki **Thruster** dosyasının içerisindeki ilk nesneyi sahneye sürüklüyoruz. Diğer nesneleri de kullanarak thruster animasyonunu oluşturuyor ve konumunu geminin arkasında olacak şekilde ayarlıyoruz.

Daha sonra bu nesneyi de Player nesnesinin alt nesnesi olacak şekilde üzerine sürüklüyoruz

7-Sağ ve sol motor için hasar animasyonlarının eklenmesi

Sprites klasöründeki **Player_hurt** dosyasının içerisindeki ilk nesneyi sahneye sürüklüyoruz ve diğer nesneleri kullanarak animasyonu oluşturuyoruz. Daha sonra bu nesnenin bir kopyasını oluşturarak sırasıyla **Right_Engine** ve **Left_Engine** isimlerini veriyoruz. Konumlarını ayarladıktan sonra, bu nesneleri Player nesnesinin alt nesnesi olacak şekilde Player nesnesinin içerisine sürüklüyoruz.

Kodların tamamına Github sayfası üzerinden ulaşabilirsiniz:

Github Linki: