BLM0364 OYUN PROGRAMLAMA – HAFTA 9 RAPORU

Yasin Ekici

21360859029

Rapor İçeriği

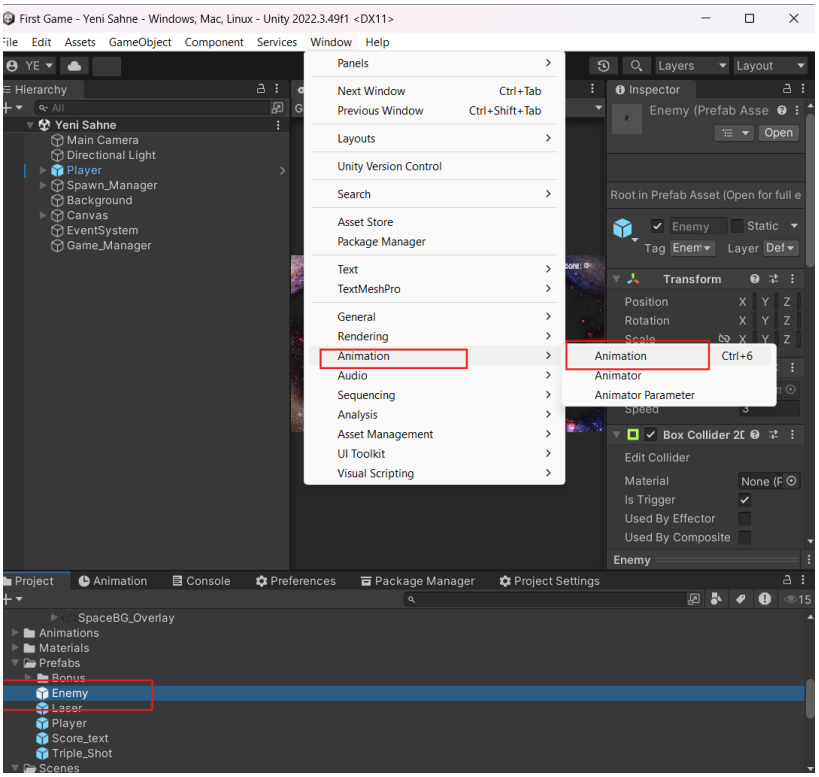
Düşman gemileri yok edildiğinde düşman patlama animasyonunu gösterin (loop time ayarı, has exit time ayarı, transition duration ayarı, animation controller yeni state ekleme, default state değiştirme, transition ekleme, trigger ekleme, transition'a condition ekleme, trigger'ın script içinde set edilmesi), nesne yok etmede gecikme eklenmesi, hız sabitleme ile yok olan düşman gemisinin zarar vermesinin önüne geçilmesi, asteroid ekleme (collider, rigid body, vs.), asteroid'in z ekseninde sürekli hareket etmesi, asteroid patlama animasyonu (prefab oluşturarak script ile dinamik eklenme ve yok edilme), spawn routine'lerinin asteroid yok edildikten 3 saniye sonra başlaması, thruster ekleme, sağ ve sol motor için hasar animasyonlarının eklenmesi .

Bu raporda anlatılan kodlar için: https://github.com/YasinEkici/game-programming

Asset'ler için:
https://e.pcloud.link/publink/show?code=XZhFVJZuzLw5aWEuizh3kDHE43iNVyak6l7

1. Üçlü Atış Bonusu

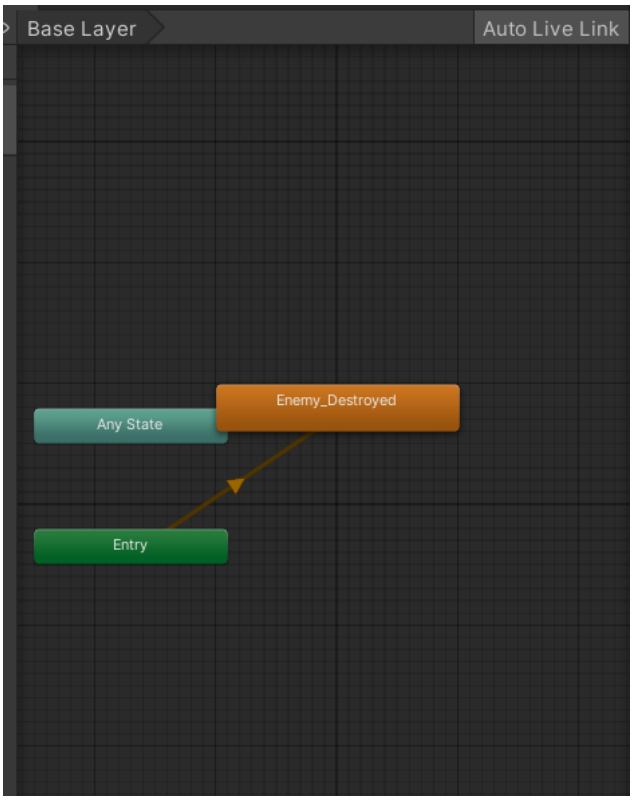
- **Düşman yok edildiğinde düşman patlama animasyonunu oynatın.**
- **Önce animasyonu oluşturun.**
  - **Enemy prefabını açın.**
  - **Enemy seçiliyken animasyon penceresine gidin (açık değilse Window -> Animation -> Animation).**



- **Create'e tıklayın ve Animations klasörü altına Enemy_Destroyed_anim olarak kaydedin.**
- **Kayıt modunu etkinleştirin.**
- **Enemy_Explodes_Sequence altındaki tüm sprite'ları seçin ve animasyon penceresine sürükleyin.**
- **Kayıt modunu devre dışı bırakın ve test için animasyonu oynatın.**
- **Sahneye geri dönün.**
- **Enemy_Destroyed_anim'i seçin ve Loop Time'ı devre dışı bırakın.**
-

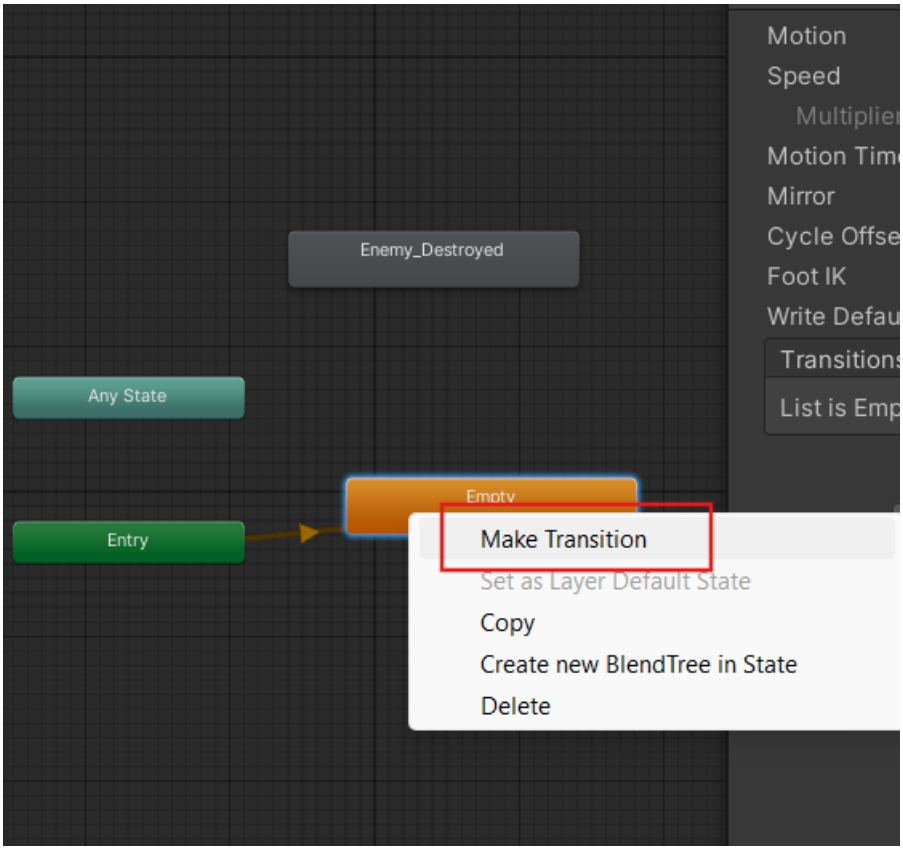
➢ **Enemy_Destroyed_anim animasyon denetleyicisini açın.**

- **Adı Enemy olan denetleyiciyi Enemy_Destroyed olarak yeniden adlandırın.**
- **Enemy_Destroyed'ı açın.**



- **Turuncu renk varsayılan durumu ifade eder ve otomatik olarak oynatılır.**
- **Unity'e yok etme animasyonunu zamanı gelene kadar oynatmamasını söyleyin.**

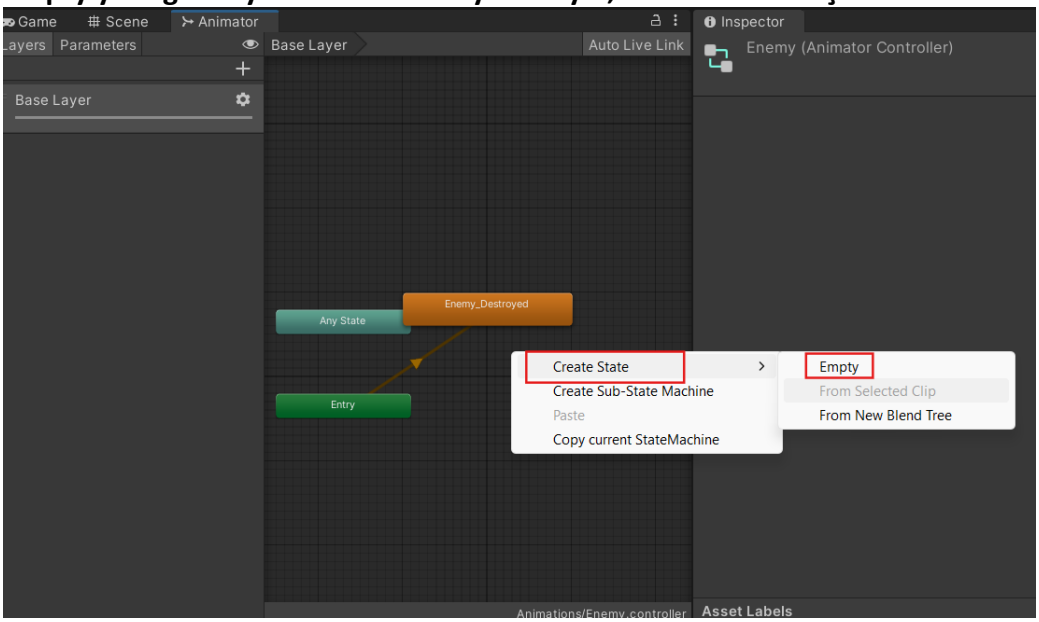
➢ **Sağ tıklayın ve boş bir durum oluşturun.**



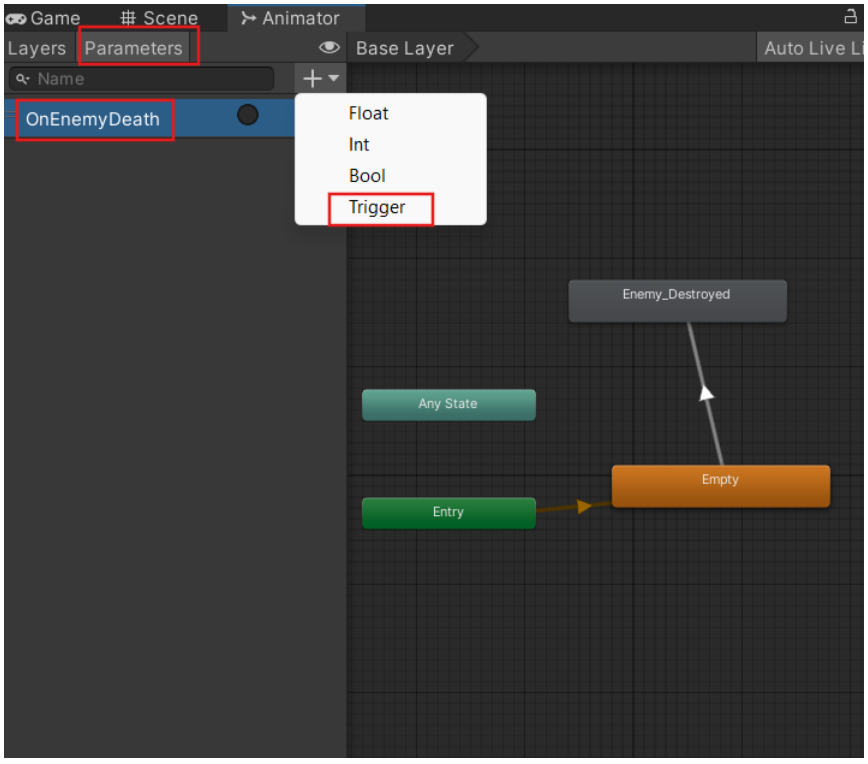
- **Yeni durumu seçin ve Empty olarak yeniden adlandırın.**
- **Empty durumuna sağ tıklayın ve varsayılan durum yapın.**

➢ **Empty ile Enemy_Destroyed_anim arasında bir geçiş oluşturun.**

- **Empty'ye sağ tıklayın ve Transition'ya tıklayın, sonra hedefi seçin.**

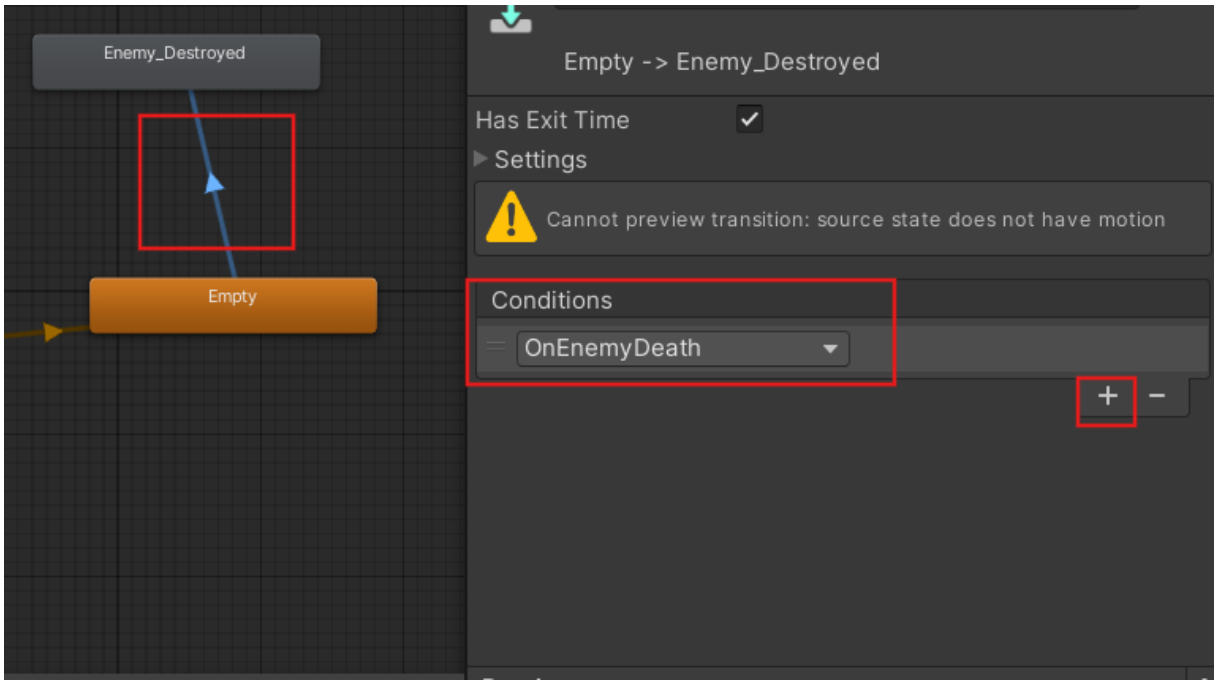


- **Soldaki Parameters sekmesini açın, Trigger ekleyin ve adını OnEnemyDeath olarak değiştirin.**



- **Transition'ı seçin ve ayarlara gidin, yeni bir Condition ekleyin.**

➢ **Trigger etkinleştirilirse, Empty'den Destroy animasyonuna geçiş yapar.**



- **Enemy scriptini açın.**
- **Animator bileşenine bir handler oluşturun.**

```
private Animator anim;
```

- **Start fonksiyonunda bileşenini atayın.**

```
anim = GetComponent<Animator>();
```

- **Enemy yok edildiğinde setTrigger metodunu çağırın. Enemy scriptinde;**

```
void OnTriggerEnter2D(Collider2D other){
    if(other.tag == "Player"){
        Player_sc player = other.transform.GetComponent<Player_sc>();
        if(player != null){
            player.Damage();
        }
        anim.SetTrigger("OnEnemyDeath");
        Destroy(this.gameObject);
    }
    else if (other.tag == "Laser"){
        Destroy(other.gameObject);
        if(player != null){
            player.UpdateScore(10);
        }
        anim.SetTrigger("OnEnemyDeath");

        Destroy(this.gameObject);
    }
}
```

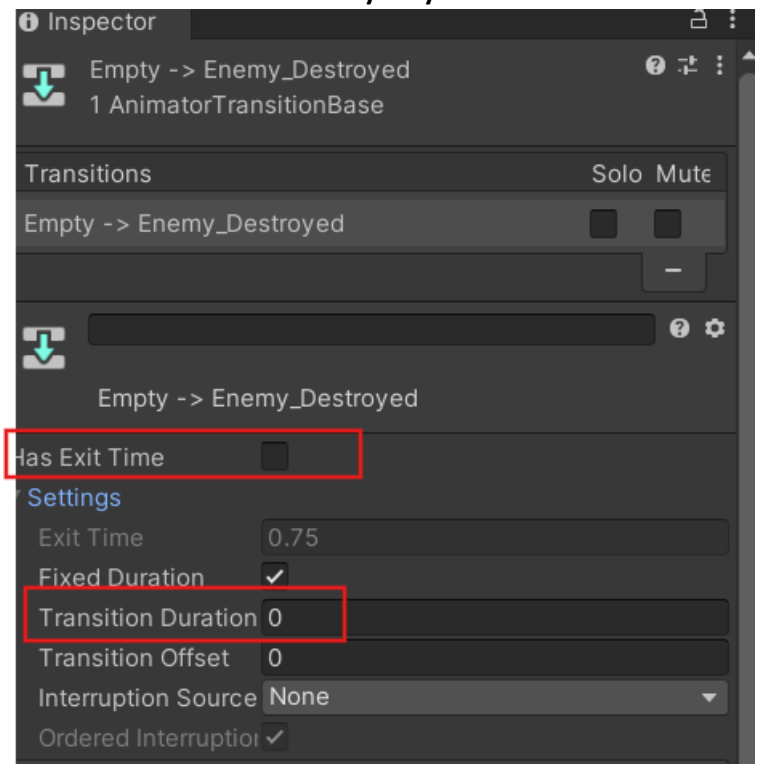
- **Şu anki halinde animasyon oynamıyor çünkü objeyi yok etme işlemini animasyonu çalıştırmadan önce yapıyoruz.**
- **Enemy objesini yok etmeden önce bir gecikme ekleyin.**
- **Animasyon uzunluğuna bakın (~ 2.8 saniye).**
- **Destroy metoduna yeterli bir gecikme ekleyin;**

```
Destroy(this.gameObject, 2.8f);
```

- **Şu an oyunumuzda bir hata var Enemy yok edildikten sonra 2.8 saniye boyunca oyuncuya zarar verebiliyor, bunu düzeltmemiz gerekir.**
- **Eğer Enemy'nin hızını öldükten sonra 0'larsak hareket edemez.**
- **Oyuncu patlamaya doğru hareket ederse, zarar anlamlı olur.**

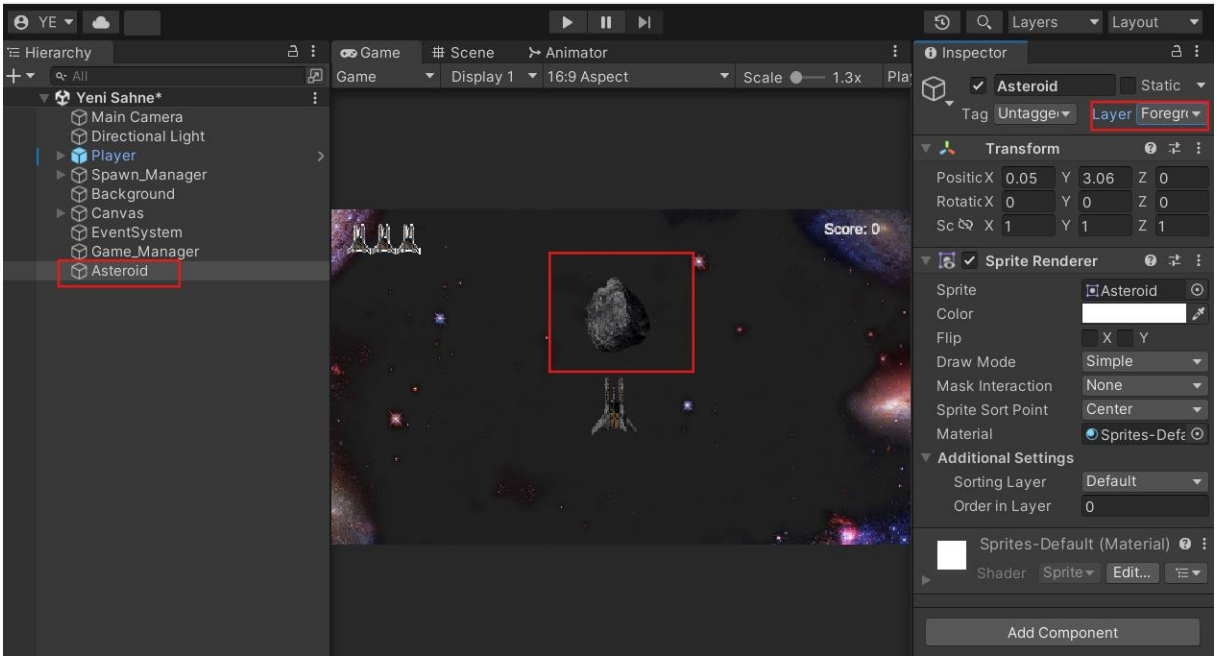
```
anim.SetTrigger("OnEnemyDeath");

speed = 0f;

Destroy(this.gameObject, 2.8f);
```

- **Enemy'ye ateş ettiğimiz an ile animasyonun başlaması arasında gecikme var.**
- **Empty ile Destroy durumları arasındaki geçişte Has Exit Time'ı kapatın.**
- **Transition Duration 0 olarak ayarlayın.**



Asteroid Oluşturma

- **Asteroid sprite'ını Hiyerarşi Penceresine sürükleyin.**
- **Oyuncunun üzerine konumlandırın.**
- **Sıralama katmanını Foreground olarak ayarlayın.**
- **CircleCollider2D ekleyin ve boyutunu ayarlayın, isTrigger'ı true yapın.**
- **RigidBody2D ekleyin ve GravityScale'i 0 yapın.**



- **Asteroid_sc adında yeni bir C# scripti oluşturun.**

- **Yeni scripti Asteroid oyun nesnesine ekleyin.**
- **Scriptte, oyun nesnesini Z ekseninde sürekli döndürün;**

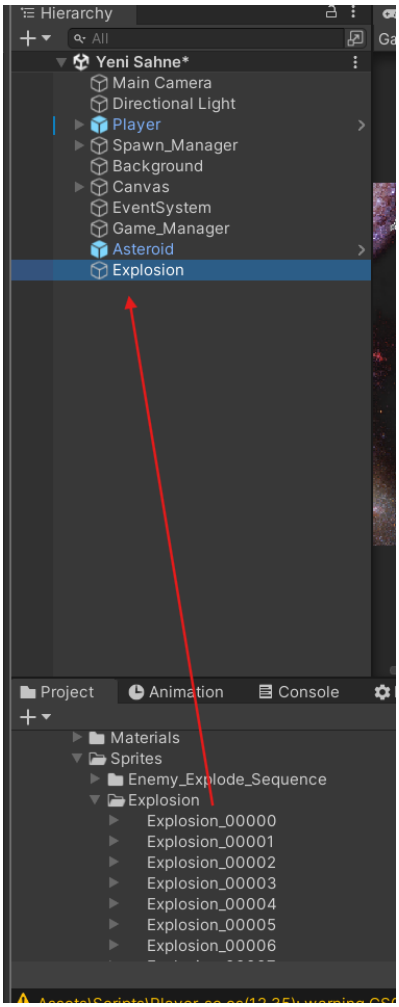
  [SerializeField] private float rotateSpeed= 20.0f;

  Update fonksiyonuna;

  transform.Rotate(Vector3.forward * rotateSpeed * Time.deltaTime);

- **Asteroid oyun nesnesini bir prefab yapın.**
- **Asteroid için bir patlama oluşturacağız;**

- **Sprites altındaki Explosion klasöründeki sprite'ları kullanacağız.**
- **Explosion klasöründeki ilk sprite'ı Hiyerarşi Penceresine sürükleyin.**
- **Adını Explosion olarak değiştirin.**



- **Explosion nesnesini seçin ve Animasyon Penceresini açın.**
- **Create düğmesine tıklayın ve Animations klasörü altına Explosion_anim olarak kaydedin.**
- **Kayıt modunu etkinleştirin, sprite'ları Animasyon Penceresine sürükleyin ve kayıt modunu devre dışı bırakın.**
- **Animasyon için Loop Time'ı false olarak ayarlayın.**

- **Explosion'ı bir prefab yapın.**
- **Asteroid scripti içerisinde Lazer ile çarpışmayı kontrol edin.**
- **Patlama nesnesini prefabdan asteroidin pozisyonunda oluşturun.**
- **Önce lazer nesnesini ardından asteroidi yok edin.**

- **Patlama prefabını instantiate etmek için özel bir değişken tanımlayın.**

```
[SerializeField] GameObject explosionPrefab;
```

- **Prefab değerini Inspector penceresinden ayarlayın.**

- **Eğer asteroid lazerle çarpıştıysa**
- **Patlama nesnesini oluşturun.**
- **Lazer ve asteroid nesnelerini yok edin.**

```
void OnTriggerEnter2D(Collider2D other){

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

    Instantiate(explosionPrefab, transform.position, Quaternion.identity);

    Destroy(other.gameObject);

    Destroy(this.gameObject);

    }
```

```
    }
```

- Asteroid patladıktan sonra patlama (clone) nesnesi Hiyerarşi Penceresinde kalıyor.
- Explosion_sc adında yeni bir C# scripti oluşturun.
- Yeni scripti Explosion prefabına ekleyin.
- Scriptin ana amacı, oyun nesnesi oluşturulduktan belli bir süre sonra nesneyi yok etmektir.
- Start metodunda 3 saniye bekleyin ve ardından nesneyi yok edin.

```
    void Start()
      {
        Destroy(this.gameObject, 3f);
      }
```

- Oyunun asteroid yok edildikten sonra başlayacağız,
- Bunun için asteroid scriptinin SpawnManager ile iletişim kurması gerekir
- Bunun için Spawn Manager scriptinde bir fonksiyon oluşturun;

```
public void StartSpawning(){
    StartCoroutine(SpawnEnemyRoutine());
    StartCoroutine(SpawnBonusRoutine());
  }
```

Düşmanlar ve bonuslar spawnlanmadan önce delay koyun;

```
IEnumerator SpawnEnemyRoutine(){
    yield return new WaitForSeconds(3.0f);
    while(stopSpawning == false){
      Vector3 position = new Vector3(Random.Range(-9.4f, 9.4f), 7, 0);
      GameObject newEnemy = Instantiate(enemyPrefab, position, Quaternion.identity);
      newEnemy.transform.parent = enemyContainer.transform;
      yield return new WaitForSeconds(5.0f);
    }
  }
```

```
IEnumerator SpawnBonusRoutine(){
    yield return new WaitForSeconds(3.0f);
    while(stopSpawning == false){
        Vector3 position = new Vector3(Random.Range(-9.8f, 9.8f), 7.4f, 0);
        int randomBonus = Random.Range(0,3);
        Instantiate(bonusPrefabs[randomBonus], position, Quaternion.identity);
        yield return new WaitForSeconds(Random.Range(5,10));
    }
}
```

- **Asteroid scripti içerisinde SpawnManager için bir özel değişken tanımlayın.**
- **Start metodunda SpawnManager bileşenini bulun.**

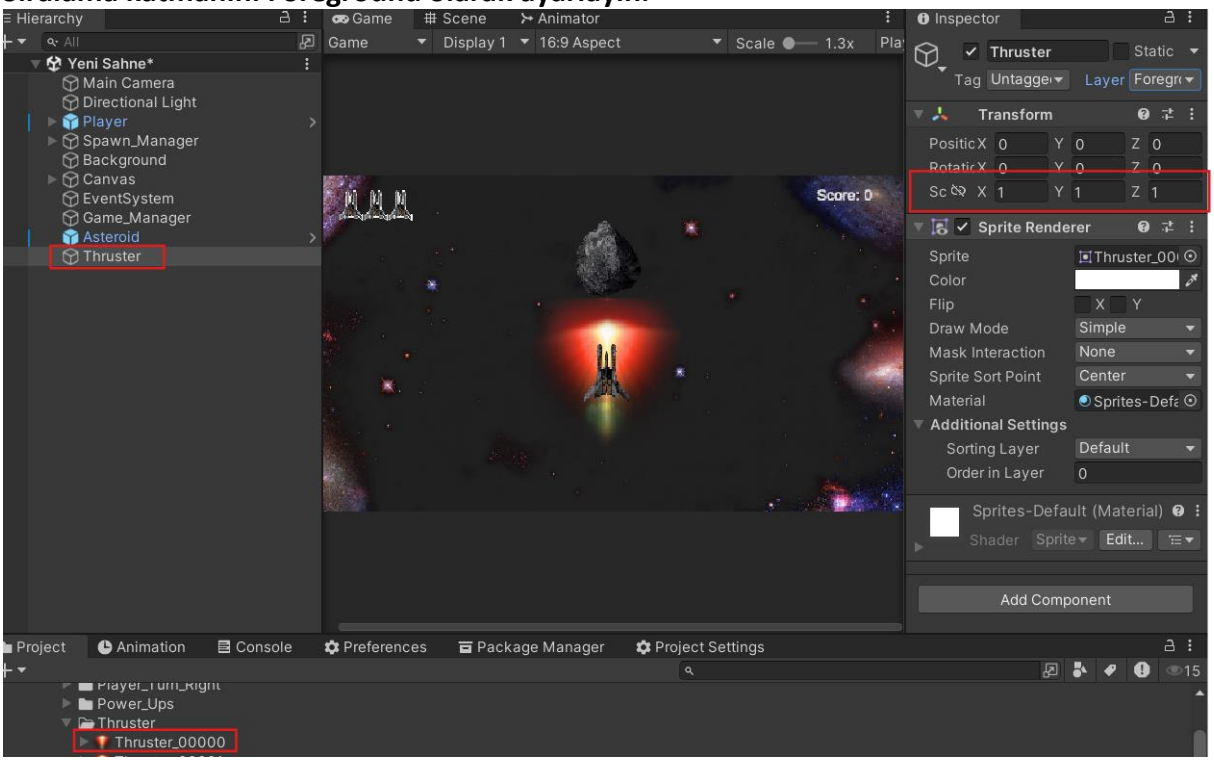
```
void Start()
    {
        spawnManager =
GameObject.Find("Spawn_Manager").GetComponent<SpawnManager_sc>();
        if(spawnManager == null){
            Debug.LogError("Spawn Manager is NULL");
        }
    }
```

- **OnTriggerEnter2D metodunu güncelleyin ve spawn fonksiyonunu çağırın.**

```
void OnTriggerEnter2D(Collider2D other){
        if(other.tag == "Laser"){
        Instantiate(explosionPrefab, transform.position, Quaternion.identity);
        Destroy(other.gameObject);
```

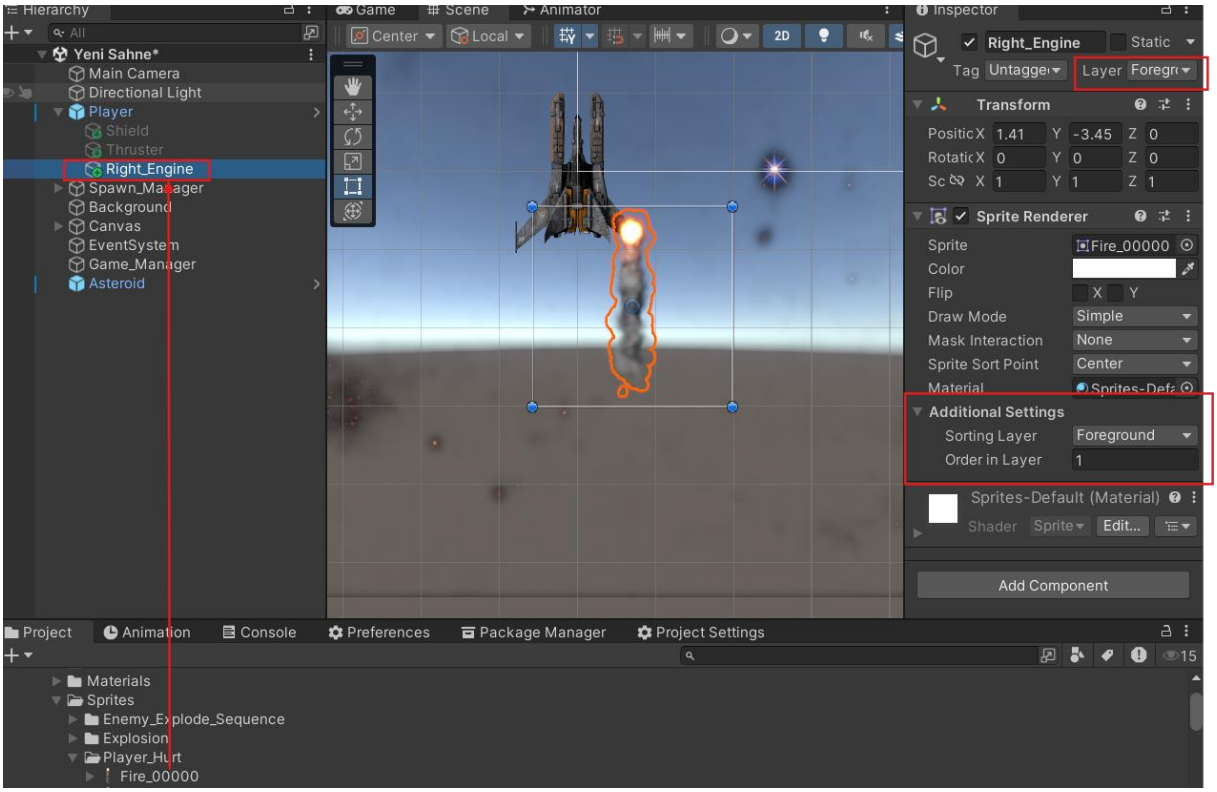
```
        spawnManager.StartSpawning();

        Destroy(this.gameObject);

        }

}
```

- Player için thruster;
- Oyuncu nesnesine bir thruster ekleyin.
- Thruster klasöründeki ilk thruster sprite'ını bulun ve Hiyerarşi Penceresine sürükleyin.
- Adını Thruster olarak değiştirin.
- Ölçeği 1,1,1 olarak ayarlayın.
- Sıralama katmanını Foreground olarak ayarlayın.

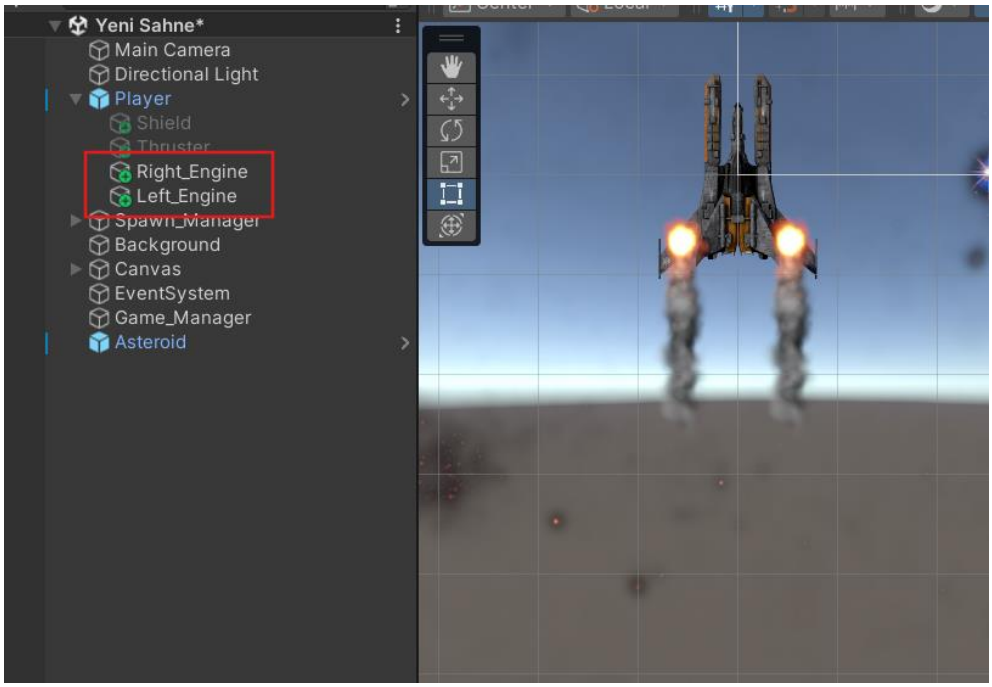


- Thruster nesnesi oyuncu nesnesiyle birlikte hareket etmelidir.
- Thruster'ı oyuncunun child'ı yapın.
- Thruster'ı oyuncunun arkasına uygun şekilde konumlandırın.
- Thruster'ı animasyon haline getirin.
- Thruster'ı seçin ve Animasyon Penceresini açın.
- Create'e tıklayın ve Animations klasörü altına Thruster_anim olarak kaydedin.
- Kayıt modunu etkinleştirin, thruster sprite'larını Animasyon Penceresine sürükleyin ve kayıt modunu devre dışı bırakın.

- **Oyuncu Hasar Görselleştirmesi;**
- **Player Hurt klasöründeki ilk fire sprite'ını seçin ve Hiyerarşi Penceresine sürükleyin.**
- **Adını Right_Engine olarak değiştirin.**
- **Oyuncunun child'ı yapın.**
- **Sıralama katmanını Foreground olarak ayarlayın.**
- **Katman sırasını 1 olarak ayarlayın.**
- **Sağ kanada uygun şekilde konumlandırın.**
- **Ölçeği 1,1,1 olarak ayarlayın**



- **Nesneyi çoğaltın.**
- **Adını Left_Engine olarak değiştirin.**
- **Sol kanada uygun şekilde konumlandırın.**



- **Her iki nesneyi başlangıçta kapatın.**
- **Oyuncu hasar aldığında bunları birer birer etkinleştirin.**
- **Player scriptinde;**
- **Sağ ve sol motor ateşleri için iki özel değişken oluşturun.**

```
[SerializeField] private GameObject rightEngine;

[SerializeField] private GameObject leftEngine;
```

- **Damage fonksiyonunu güncelleyin, kalan canlara göre engine arızalarını gösterin;**

```
 if(lives == 2){

   leftEngine.SetActive(true);

 }

 else if(lives == 1){

   rightEngine.SetActive(true);

 }
```

- **Sağ motoru seçin ve Animasyon Penceresini açın.**
- **Create'e tıklayın ve Animations klasörü altına Engine_Failure_anim olarak kaydedin.**
- **Kayıt modunu etkinleştirin, Player Hurt klasöründeki sprite'ları Animasyon Penceresine sürükleyin ve kayıt modunu devre dışı bırakın.**
- **Sol motoru seçin ve Animator bileşeni ekleyin.**

- **Sol motorun Animator Kontrolcüsünü Engine_Failure olarak ayarlayın.**