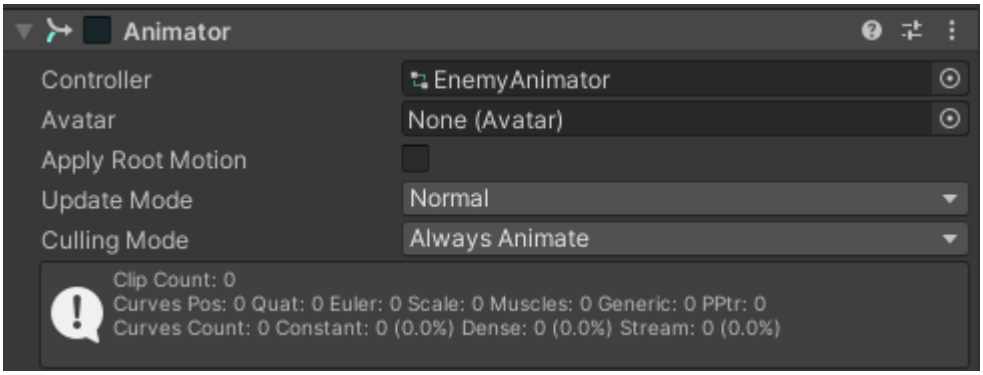
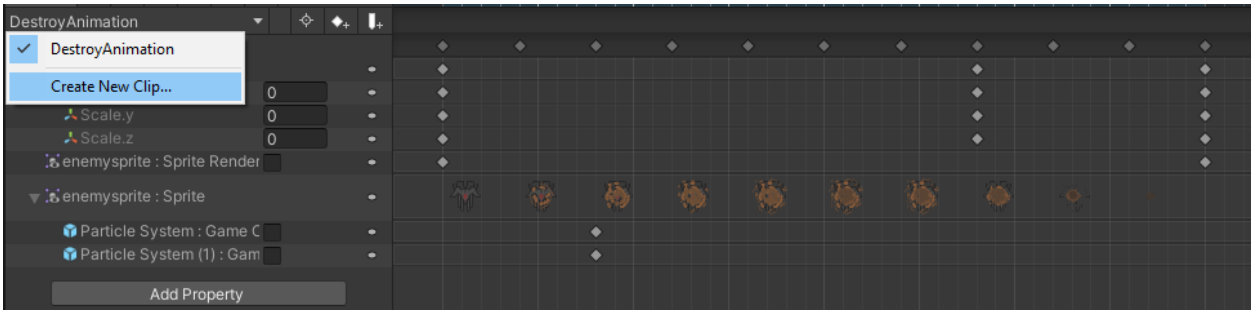
# 9.Hafta Ders Raporu – Batuhan Şengül

## Animasyonlar

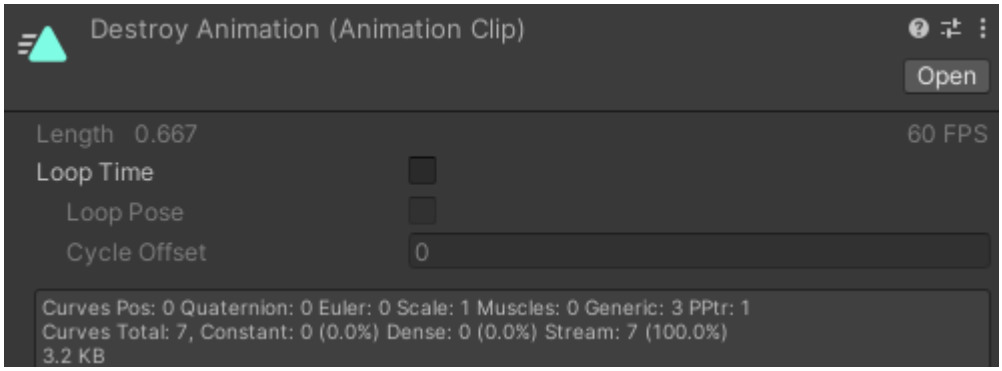
### Düşman



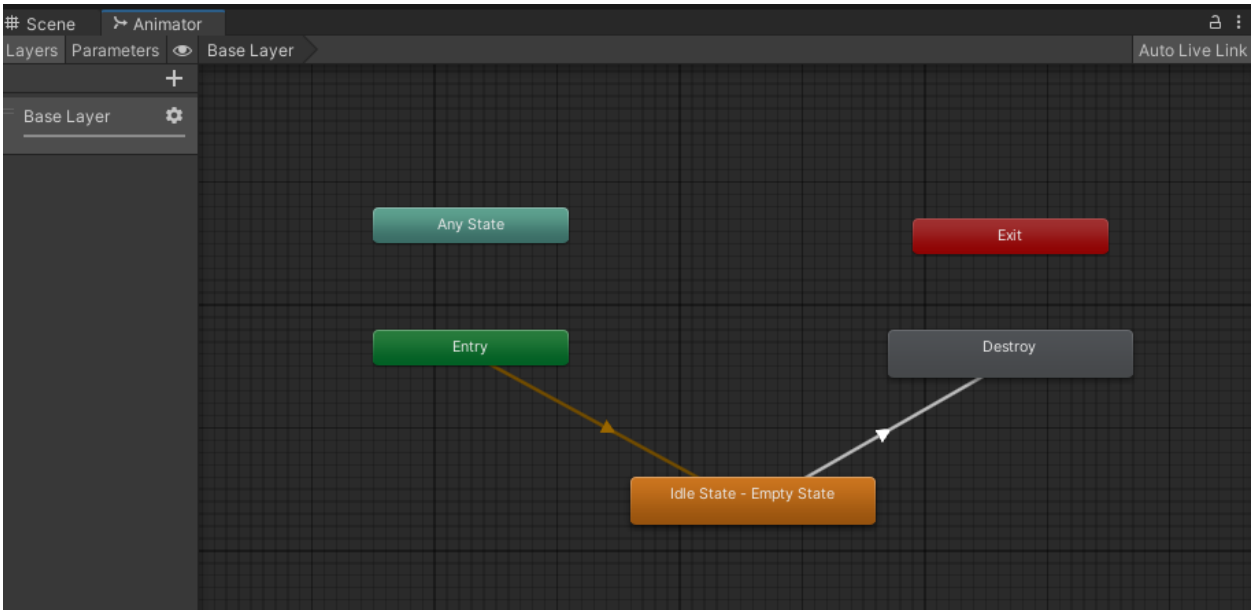

- Düşmana animator eklenmiştir. Animatorda durum geçişleri ileride anlatılmıştır.



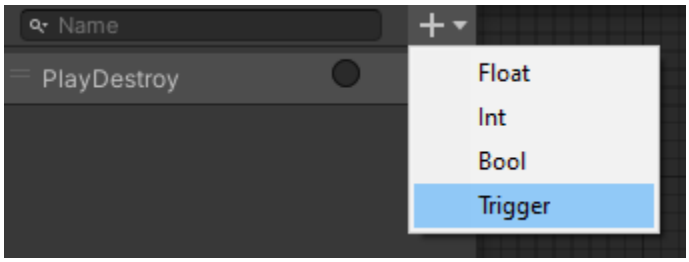

- Düşman vurulduğunda oynatılacak animasyon ayarlanmıştır. Bu animasyon basitçe patlama animasyonunu oynatır ve önceki haftalardan eklenen thruster particle effectini kapatır ve prüzsüz bir geçiş(yok oluş) için sprite'ın ebatlarını sona doğru küçültür.



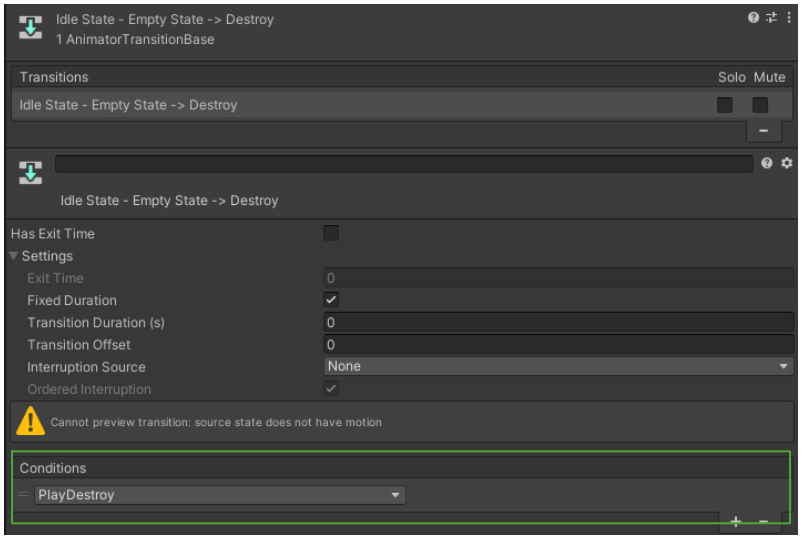

- Oluşturulan animasyon bir kez oynatılacağı için "Loop Time" özelliği kapatılır.




- Boş bir state oluşturulmuştur. Boş stateten yok edlime animasyonunun oynatıldığı state geçiş yapılmıştır.




- Trıgger tipinde “PlayDestroy” adlı bir parametre oluşturulmuştur.




- Oluşturulan geçişin ayarları yapılmıştır geçişin gerçekleşme durumu “PlayDestroy” parametresine bağlanmıştır, bu şekilde kod içerisinde çağırmamız mümkün olacaktır.

- **Enemy.cs Kod Değişiklikleri Güncellemeleri**

```
private Animator animator;
Unity Message | 0 references
void Start()
{


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

Animatoru tutacak değişken oluşturulmuş ve script başlatıldığında otomatik olarak oyun nesneinin animatör bileşeni bu değişkene atanmasını sağlayan kod kısımları eklenmiştir.

```
2 references
private void HandleDestruction()
{

    if (!isAsteroid)
    {
        animator.SetTrigger("PlayDestroy");
        speed = 0;
        col.enabled = false;
        Destroy(gameObject, 1f);
    }
    else
    {
        speed = speed / 2f;

        Destroy(gameObject,0.01f);
    }
}
```

Astroid olmayan durumda yani uzay gemisi düşman ise OnTriggerEnter'da çağırılacak ve animasyonun içerisindeki triggerı etkileyen ve animasyonun oynaması için oyun nesnesinin yok oluşunu geciktiren, vurulan düşmandan hasar alınmaması için gerekli collider ve hız ayarlarını değiştiren HandleDestruction fonksiyonu yazılmıştır.

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player")&& PlayerMovement.Instance.canTakeDamage)
    {
        //Debug.Log("Oyuncu temasi");
        PlayerMovement.Instance. SpawnExp(transform.position + new Vector3(0, 0, 0.05f));
        PlayerMovement.Instance.LowerLives();

        PlayerMovement.Instance.StartImmunity(2f);
        if (isAsteroid)
        {
            DuplicateAsteroid();
        }
        HandleDestruction();

    }
    else if (other.CompareTag("Laser"))
    {
        //Debug.Log("Lazer temasi");


        Destroy(other.gameObject);

        if (isAsteroid)
        {

            TextManager.Instance.AddScore(100 * Mathf.CeilToInt(1f/transform.localScale.x));
            DuplicateAsteroid();
        }
        else
        {
            TextManager.Instance.AddScore(100);
        }
        PlayerMovement.Instance.SpawnExp(transform.position + new Vector3(0, 0, 0.05f));
        HandleDestruction();
    }
}
```

OnTriggerEnter'da yukarıda yazılan HandleDestructıon fonksiyon çağırılması eklenmiştir ve animasyonlar başarılı bir şekilde oyunu eklenmiş olmuştur.

Oyun içerisinde sırasıyla düşman patlama animasyonunun kareleri bu şekildedir. Ayrıca önceki haftalarda eklenmiş olan duman efektlerinin varlığı da bu animasyonun bütününü güçlendirilmiştir.

## Asteroid

Öncelikle aynı işlevi olan 2 düşman tipi eklemek oyun tasarımı kısmında mantıklı değil ve ayrıca aynı hareket kodunu içerecek yeni bir script oluşturmaya da gerek yok ,derste anlatılanın aksine, o yüzden, astriod için Enemy.cs scripti tekrar düzenlenmeye karar verilmiştir.

NOT:Bu yüzden istenlenin dışına çıkacak bir tasarım yaptım. Basitçe Astroidler her vuruşta 2 ye bölünüyor. Bölündükçe daha hızlı oluyor ve ne kadar küçük bir astroid vurulursa o kadar fazla skor kazanılıyor. Astroidi'in küçülmesi de belirli bir noktaya kadar kod içerisinde kontrol ediliyor ve astroid noraml bir düşmandan farklı çapraz hareket ediyor ettiği yön doğrultusunda da kendi etrafında dönüyo. Bu şekilde yapılmasından ötürü astroid de sadece duman efektleri tercih ettim yani kendine özel bir yok edilme animasyonu tanımlamadım. Şimdi kod detaylarını anlatılacaktır.

- **Değişkenler**

```
[Header("Asteroid Settings")]
public bool isAsteroid = false;
public GameObject asteroidPrefab;
private Vector3 moveDirection;
private float spinSpeed;

private GameObject spriteGO;
int[] possibleXValues= {-2,-1,1,2 };
```

public bool isAsteroid = false: Astroid mi değil mi tutan değişken.

public GameObject asteroidPrefab: Astroid prefabini tutan değişken.

private Vector3 moveDirection: Astroid hareket doğrultusunu tutan değişken.

private float spinSpeed: Astroid kendi etrafında dönme hızını tutan değişken.

private GameObject spriteGO: Dönecek olan spriteı tutan oyun nesnesi değişkeni.

int[] possibleXValues= {-2,-1,1,2 }: Rastgele seçilecek hareket doğrultularında x i tutan dizi.

- **Start() Fonksiyonu**

```
if (isAsteroid && moveDirection == Vector3.zero)
{
    moveDirection = new Vector3(possibleXValues[Random.Range(0, possibleXValues.Length)], Random.Range(-speed, -1.5f), 0f).normalized;
}

xPos = Random.Range(-PlayerMovement.Instance.xBorderValue, PlayerMovement.Instance.xBorderValue);
if (!isAsteroid)
{
    transform.position = new Vector3(xPos, 6, 0);
    speed += Random.Range(0, speedRandomMultiplier);
}
else
{
    RotateAsteroid();
}


if (isAsteroid && spinSpeed == 0)
{
    spinSpeed = Random.Range(50f, 100f);
}
else
{
    spinSpeed *= 1.1f;
}
```

Astroid oluştuğunda harekt doğrultusu yoksa o ayarlanır.

Hareket edeceği doğrultuya doğru bir kez döndürülür.(RotateAstroid())

Kendi etrafında döndürme hızı ayarlı değilse o ayarlanır.

- **Update() Fonksiyonu**

```
if (isAsteroid)
{
    transform.position += moveDirection * speed * Time.deltaTime;
    SpinAsteroid();

}
else
{
    transform.position += new Vector3(0, -speed * Time.deltaTime, 0);
}

if (isAsteroid)
{
    WrapAsteroidPosition();
}

ResetPosition();
```

Astroid için hareket doğrultusunda ilerletilir.

Ekranın sağından ya da solundan çıkarsa ona göre ekrana geri getirilir(WrapAstroidPosition()).

Ekranın en aşağısına ulaşınca geri yukarı ışınlanır(ResetPosition()).

- **SpinAsteroid() Fonksiyonu**

```
1 reference
private void SpinAsteroid()
{

    if (moveDirection.x > 0)
    {
        spriteGO.transform.Rotate(0, 0, -spinSpeed * Time.deltaTime);
    }
    else
    {
        spriteGO.transform.Rotate(0, 0, spinSpeed * Time.deltaTime);
    }

}
```

Update içinde bu fonksiyon çağırılarak astroidin gittiği yöne doğru(sağ ya da sol) kendi etrafında döndürülür.

- **WrapAsteroidPosition() Fonksiyonu**

```
1 reference
private void WrapAsteroidPosition()
{

    if (transform.position.x > xBorder + 2f)
    {

        transform.position = new Vector3(-xBorder-1.2f, transform.position.y, transform.position.z);

    }
    else if (transform.position.x < -xBorder - 2f)
    {

        transform.position = new Vector3(xBorder + 1.2f, transform.position.y, transform.position.z);

    }

}
```

Ekran sınırlarına göre astroin bu sınırlara dışına çıkmasına karışın konumu tekrar ayarlayan fonksiyondur.

- **DuplicateAsteroid() Fonksiyonu**

```
private void DuplicateAsteroid()
{

    if (transform.localScale.x > 0.4f)
    {
        Vector3 currentScale = transform.localScale;

        GameObject duplicate1 = Instantiate(asteroidPrefab, transform.position, Quaternion.identity);
        duplicate1.transform.localScale = currentScale * 0.70f;
        Enemy asteroid1 = duplicate1.GetComponent<Enemy>();
        asteroid1.isAsteroid = true;
        asteroid1.moveDirection = moveDirection.normalized;
        asteroid1.transform.parent=Referances.Instance.transform;


        GameObject duplicate2 = Instantiate(asteroidPrefab, transform.position, Quaternion.identity);
        duplicate2.transform.localScale = currentScale * 0.70f;
        Enemy asteroid2 = duplicate2.GetComponent<Enemy>();
        asteroid2.isAsteroid = true;
        asteroid2.moveDirection = new Vector3(-moveDirection.x, moveDirection.y, moveDirection.z).normalized;
        asteroid2.transform.parent = Referances.Instance.transform;

        asteroid1.speed += duplicate1.transform.localScale.y*1.5f;
        asteroid2.speed = asteroid1.speed;


    }


}
```

Astroid oyuncu tarafından yok edildiği zaman oluşturulan çocuk astroidlerin ayarlamalarının yapıldı fonksiyondur.

- **SpawnProtect(float duration) Fonksiyonu**

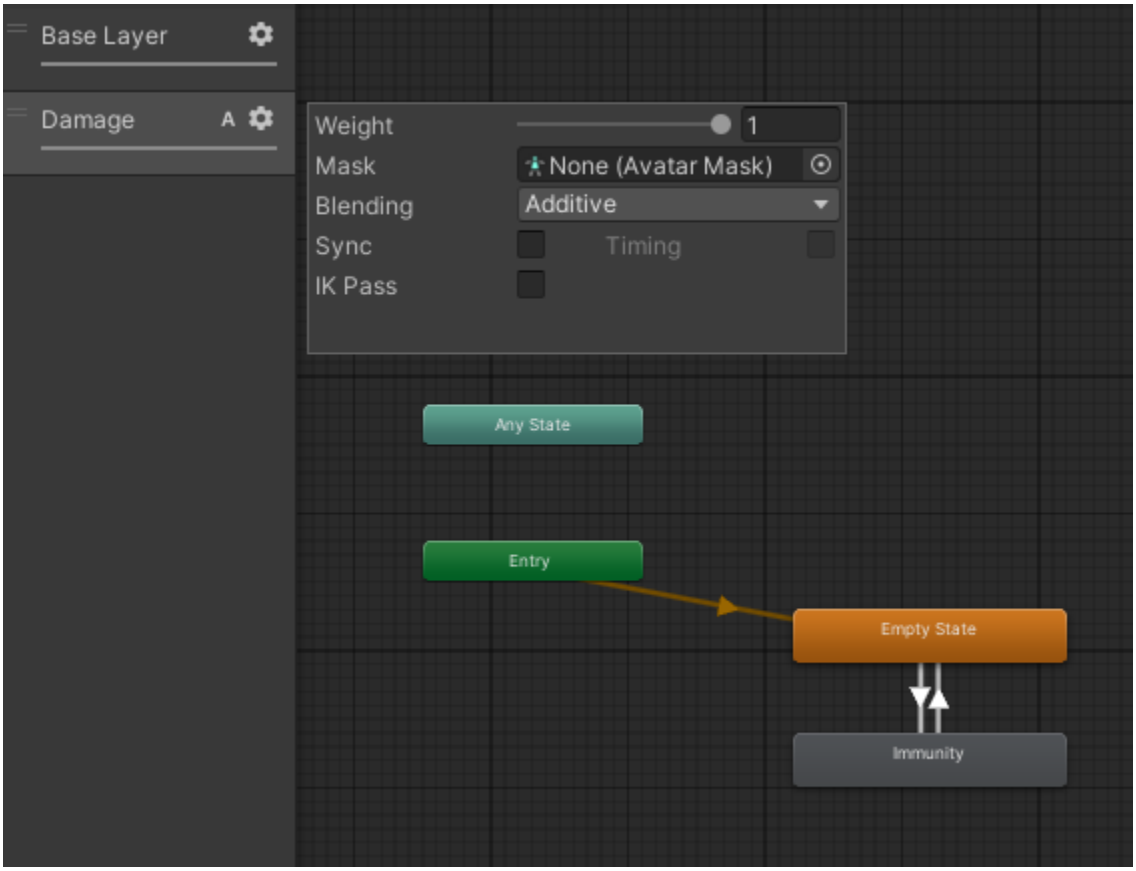
```
IEnumerator SpawnProtect(float duration)
{
    col.enabled = false;
    yield return new WaitForSeconds(duration);
    col.enabled = true;
}
```

Oyun uzayında oluşturulan nesnelerin bir süre hasar almasını engellemeye yarayan bir fonksiyondur.

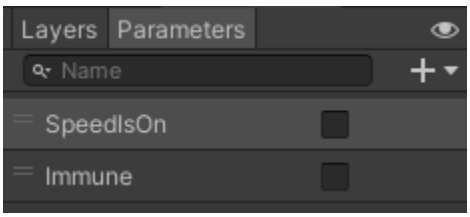
## Player

Oyun içeresine thruster tasarım olarak beğenilmediğinden eklenmemiştir. Hasar alındığını belli etmek adına oyunucunun sprite'ı kırmızı yanıp sönmesi sağlanmıştır.

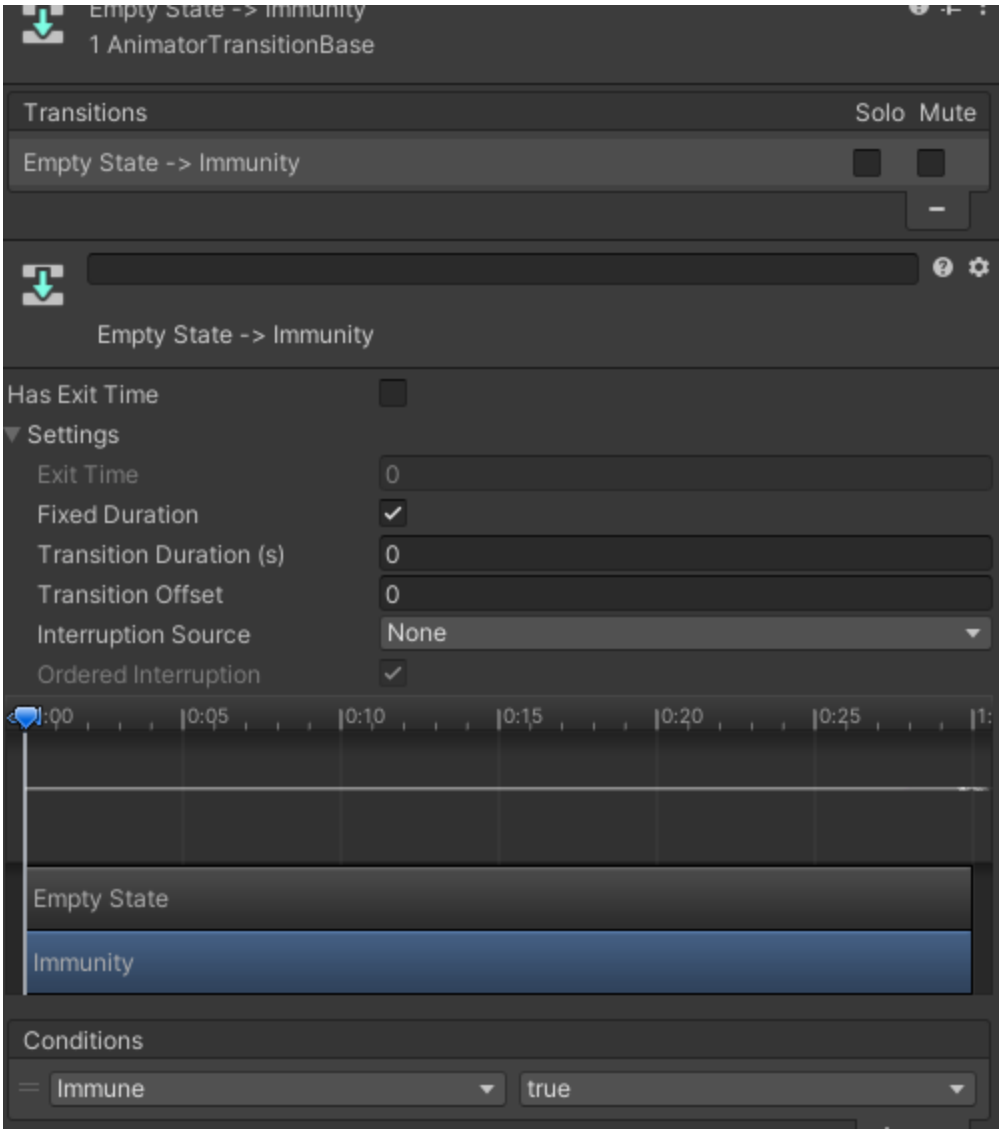
- **Animator**



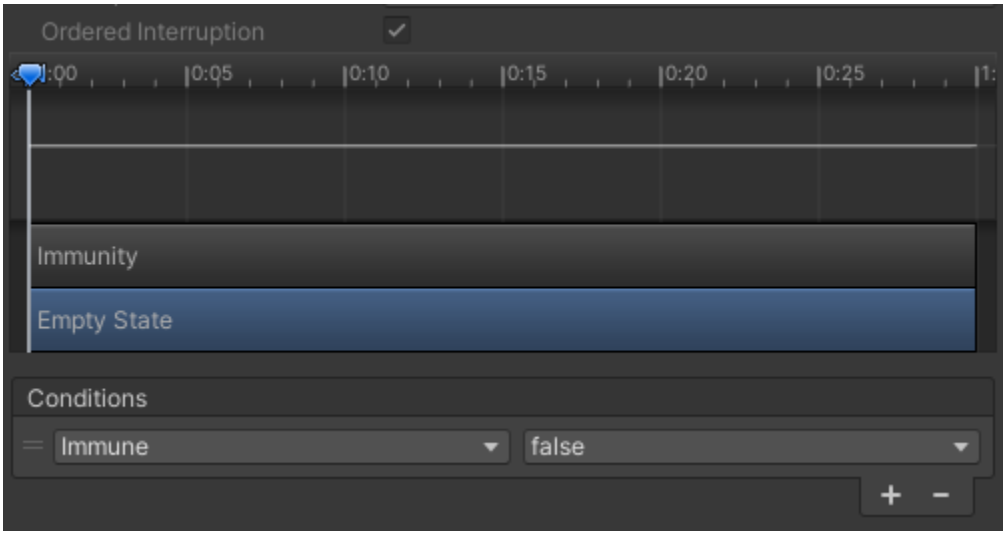

Yeni bi layerda oluşturulmuş blending mode "Additve" , weight "1" olarak ayarlanarak animasyon ağacı yapılmış, geçişler eklenmiştir.



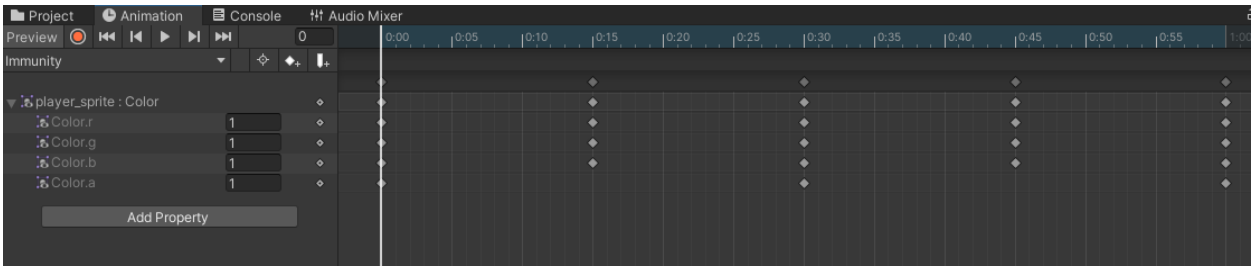

"Immune" adı verilen yeni bir bool tipli değişken eklenmiştir.




Boş stateten geçiş kondisyonu “Immune==true” olduğu animasyon stateinden geri dönüşte “Immune==false” olduğu durumda çalışması için ayarlar yapılmıştır.



- **Animation-Immunity(Hasar Alma)**




  Bu iki renk arasında yanıp sönen looplanan basit bir animasyon ayarlanmıştır.

- **Kod**

```
2 references
IEnumerator Immunity(float time)
{
    canTakeDamage = false;
    animator.SetBool("Immune", true);
    yield return new WaitForSeconds(time);
    animator.SetBool("Immune", false);
    canTakeDamage =true;
}
```

Hasar alındığı zaman bu animasyonu oynatacak ve belirli bir süre sonra durduracak coroutine yazılmıştır.

```
1 reference
public void StartImmunity(float time)
{
    if (immunityCoroutine == null)
    {
        immunityCoroutine = StartCoroutine(Immunity(time));
    }
    else
    {

        StopCoroutine(immunityCoroutine);

        immunityCoroutine = StartCoroutine(Immunity(time));
    }
}
```

Çok kze üst üste çağırılmaması için bunu da kontrolünü yapan fonskiyon yazılmıştır.

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player")&& PlayerMovement.Instance.canTakeDamage)
    {
        //Debug.Log("Oyuncu temasi");
        PlayerMovement.Instance. SpawnExp(transform.position + new Vector3(0, 0, 0.05f));
        PlayerMovement.Instance.LowerLives();

        PlayerMovement.Instance.StartImmunity(2f);
        if (isAsteroid)
        {
            DuplicateAsteroid();
        }
        HandleDestruction();

    }
```

Düşman scripti içerisinde de oyuncuyla temasa geçildiğinde bu animasyonu oynatacak fonskiyon çağırılmıştır.

# SpawnManager

## Gücellemeler ve Değişiklikler

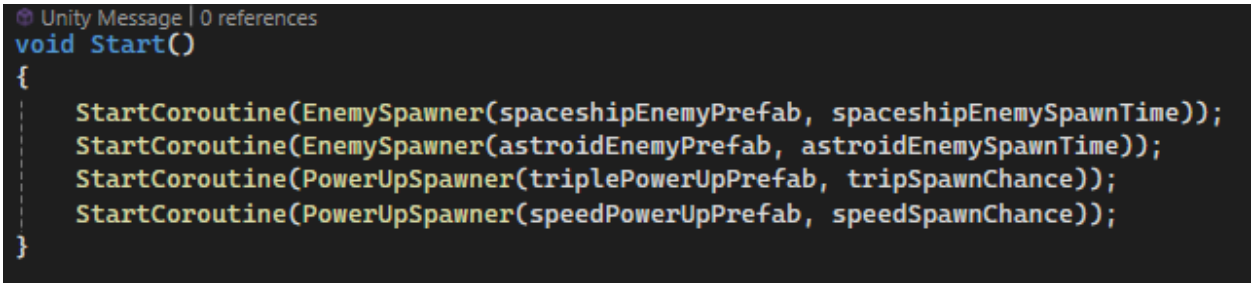
- **Değişkenler**

```
[Header("Spaceship Enemy Spawn Settings")]
[SerializeField] private GameObject spaceshipEnemyPrefab;
[SerializeField]private float spaceshipEnemySpawnTime = 2f;

[Header("Astroid Enemy Spawn Settings")]
[SerializeField] private GameObject astroidEnemyPrefab;
[SerializeField] private float astroidEnemySpawnTime = 8f;
```

- **IEnumerator EnemySpawner(GameObject enemy,float spawnTime)**

```
2 references
IEnumerator EnemySpawner(GameObject enemy,float spawnTime)
{
    while (PlayerMovement.Instance.lives > 0)
    {
        yield return new WaitForSeconds(spawnTime);
        lastSpawnedEnemy =Instantiate(enemy,
            (new Vector3(Random.Range(-PlayerMovement.Instance.xBorderValue,
            PlayerMovement.Instance.xBorderValue),PlayerMovement.Instance.yBorderValue+1.2f,0)),
            Quaternion.identity);
        lastSpawnedEnemy.transform.parent = Referances.Instance.enemyContainer.transform;

    }
}
```

Her düşman tipine uygun olacak şekilde çalışması için güncellenmiştir.




Oyun başladığında gerekli değişkenler atanarak sürekli olarak düşmanların canlanması sağlanmıştır.

## Kaynakça

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

https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UIBasicLayout.html

https://chatgpt.com/

## Proje Kodu ve Github Repo

Kod: https://github.com/bathuchan/btu-gameprogramming-BatuhanSengul/tree/main/Reports/9.Hafta

Proje Repo: https://github.com/bathuchan/btu-gameprogramming-BatuhanSengul

## Hazırlayan

Batuhan Şengül – 20360859008- bathu.sengul@gmail.com