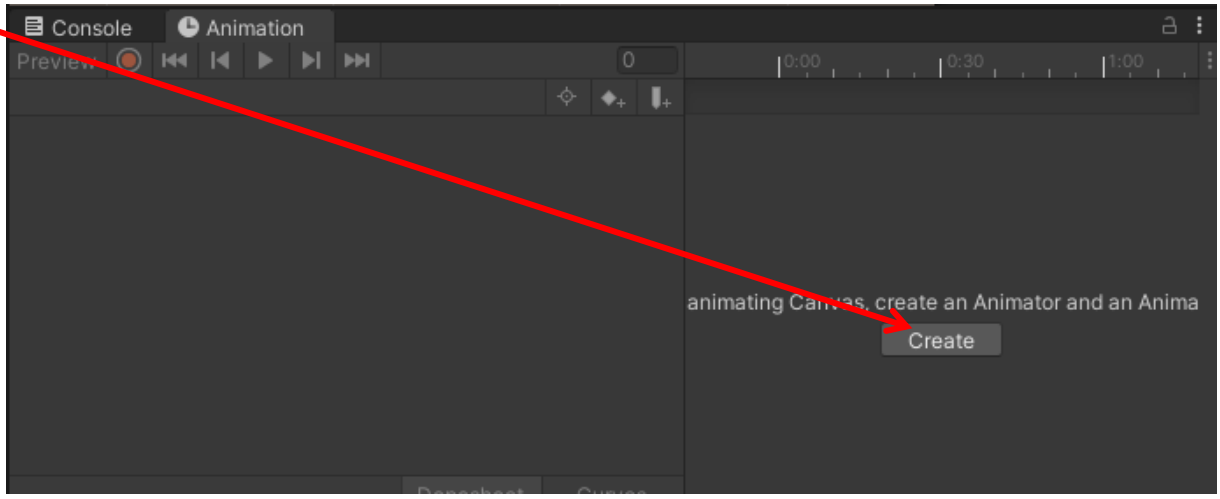
10. Hafta Raporu – Aleks Dulda 21360859025

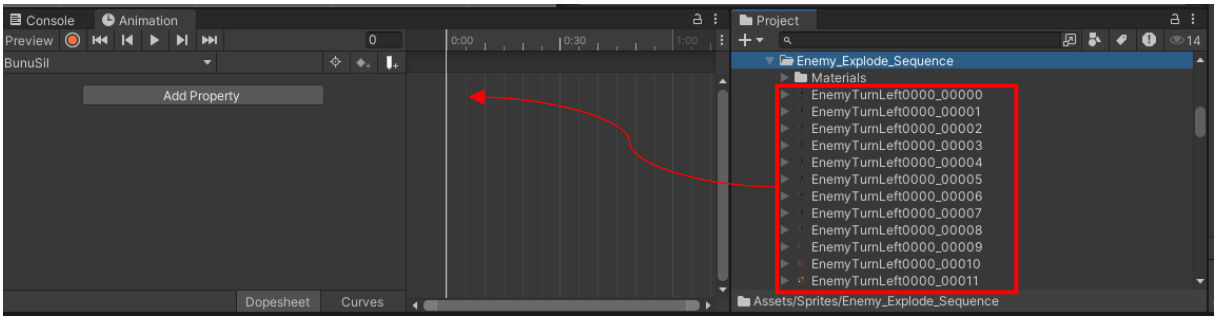
Düşman gemileri yok edildiğinde düşman patlama animasyonunu gösterin.

Bir enemy prefabı'nı sahnemize sürükle bırak metoduyla ekleyelim. Ardından "Animation" sekmesinden "Create" butonuna tıklayarak bir animasyon motoru ekliyoruz.



Açılan pencerede bu animasyon motorumuzu nereye kaydetmek istiyorsak (Tercihen Animation adlı bir dosya açmanızı ve oraya kaydetmenizi öneririm.) oraya kaydediyoruz.

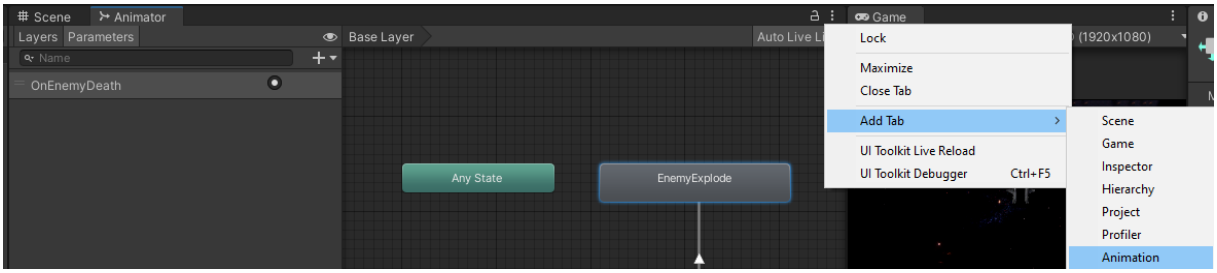
Dosyayı oluşturduktan sonra Spriteları yerleştireceğimiz kısım açılmakta. Bu kısma istediğiniz spriteları yerleştirerek "Enemy" öğesini animasyonumuzu eklemiş bulunuyoruz.



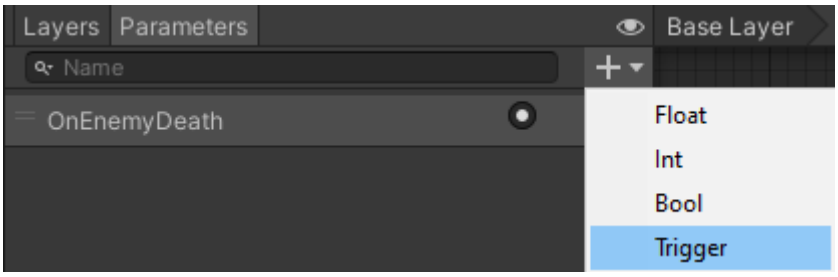
Sürükleyip bırakarak spriteları yerine yerleştirmiş olursunuz.

Bu kısımdan sonra oyunu başlattığınızda "Enemy" öğesi sürekli yok olma animasyonuyla karşımıza gelecektir. Bunu düzeltmek için animasyona koşul koyacağız. Bu koşulumuz ise düşman lazer öğesiyle veya player öğesiyle temasta bulunduğunda çalışması gerek olduğunu belirtecez.

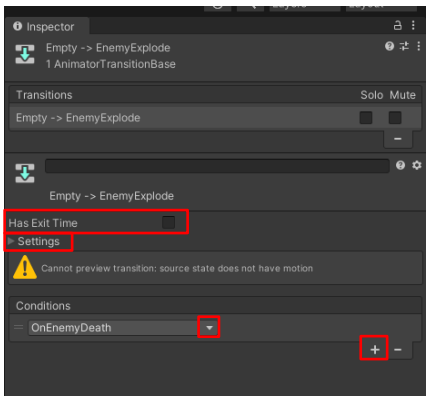
Bunun için bir pencere açmamız gerekiyor. Bu pencerimizi “Scane” penceresinin yanında bulunan üç noktadan “Animator” penceresi açabilir.



Base Layer kısmında boş bir durum oluşturuyoruz. Bu boş duruma girişten sonraki durum olarak sağ tıklayıp “set as a default state” seçeneğine tıklıyoruz. Bu sayede giriş durumunda animasyonumuz direkt çalışmayacak. Bu statenten animasyon statemize , animasyon statemize sağ tıklayıp “make transition” seçerek ilgili animasyona yönlendirme yapmaktayız. Bu kısımda bir koşul eklememiz gerekiyor ki enemy ile etkileşime geçildiğinde animasyonumuz gerçekleşsin. Bunu Animator sekmesinde bulunan “Parameters” seçeneği ile yapıyoruz.



Burdan bir Trigger değişkeni ekliyoruz. Base layer’da animasyona giden oka tıkladığımızda inspector penceresinde bazı koşulları ekleyebileceğimiz kısımları görmekteyiz. Ordan “Conditions” kısmında “+” işaretine basarak bir koşul ekliyoruz. Bu koşul tahmin edebileceğiniz üzere bizim animasyonun aktifleşmesi için yapacağımız kontrol değişkeni.



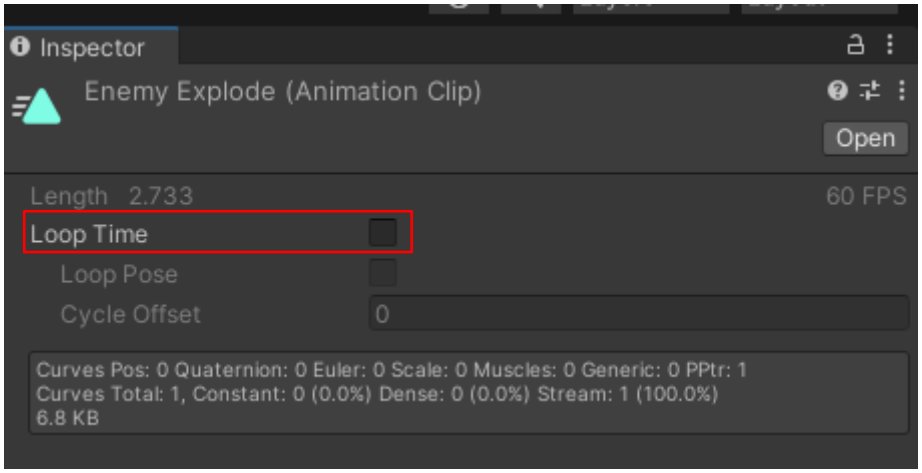
Oluşturduğunuz trigger değişkeni direkt gelmediyse ekranda bulunan ok kısmında bakabilirsiniz. Ayrıca animasyonun güzel gözükmesi için “Has Exit Time” seçeneğini kapalı olarak ayarlıyoruz. Çünkü spritelar arası gecikmeler yaşanmasın diye.

Ayrıca aynı bu kısımda “Has Exit Time” altında “Settings” kısmı mevcut bu kısımdan “transition duration” değerini 0 yapmalıyız. Çünkü Transition Duration animasyonlar veya durum geçişleri sırasında iki durum (state) arasında geçiş yapılırken geçen süreyi belirler. Bu değeri 0 yaparak arada bekleme olmadan yumuşak geçişle bir animasyon yapmamızı sağlar.

Loop Time, Animasyon klibinde Loop Time etkin değilse, Animator'da oynatıldığında animasyon bir kez oynar ve durur. Bizde bunu istediğimiz için bu ayarı kapalı tutmamız gerekiyor ki diğer oluşan enemy prefabları bu animasyondan etkilenmesin. Bu ayar için izlenecek adımlar:

Bu ayarı yapmak için şu adımları izleyin:

1. Project panelinde, ilgili animasyon klibini bulun.
2. Animasyon klibini seçin ve “**Inspector”** panelinde açın.
3. **“Loop Time”** seçeneğini işaretini kaldırın.
4. Ayarları kaydetmek için “**Apply”** düğmesine tıklayın.



Tüm değişkenleri ayarladıktan sonra yapmamız gerekn Enemy_SC scriptinde bazı kodlar yazmak olacaktır.

Animator anim; şeklinde bir değişken oluşturuyoruz bu değişken animatör kısmında oluşturduğumuz “OnEnemyDeath” değişkenini kontrol edebilmek için yapılır. anim = GetComponent<Animator>(); Komutuyla oluşturduğumuz değişkenlere erişim için bir yol oluşturuyoruz.

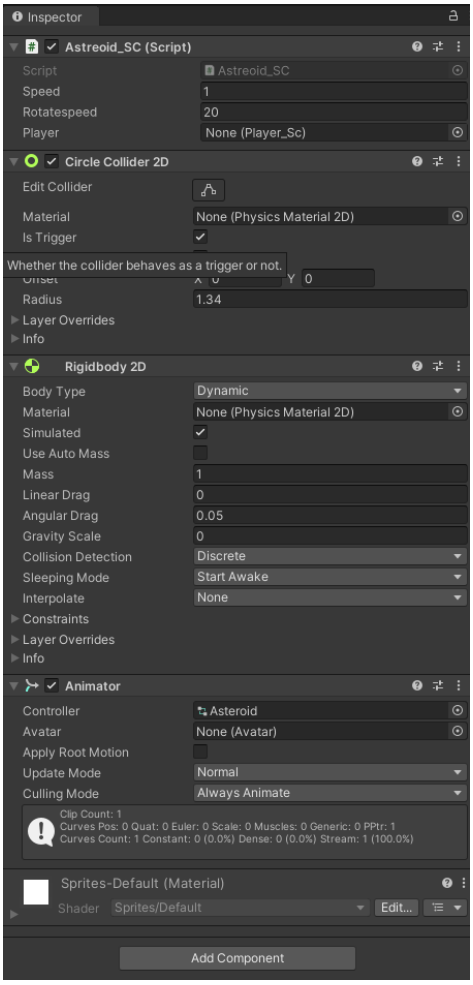
Zaten hali hazırda var olan çarpışma kontrollerine anim.SetTrigger("OnEnemyDeath"); komutunu ekliyoruz. Bu sayede çarpışma gerçekleştiğinde bu animasyon çalışacak ve yok olma efekti gerçekleşecek. Ayrıca oyun nesnemizi yok etmeden önce hızını 0’lamak gerekir çünkü animasyon sürecince bize doğru patlayan bir geminin bize hasar vermesini istemeyiz.

Animasyon düzgünce gerçekleşebilmesi için nesnemizi yok etmeden önce belirli bir süre beklememiz gerekir o da Destroy komutunun bir başka aldığı değişkeni eklemekten geçer" Destroy(gameObject, **1.0f**);// enemyi yok et " yazılarak gerekli süreyi animasyona eklemiş oluruz.

```
Animator anim;


Unity İletisi | 0 başvuru
void Start()
{
    player=GameObject.FindObjectOfType<Player_Sc>();
    if (player == null)
    {
        Debug.LogError("Player not found!");
    }
    anim = GetComponent<Animator>();
    if (anim == null)
    {
        Debug.LogError("Animator not found!");
    }
}
```

```
void OnTriggerEnter2D (Collider2D other) //temasa giren
{
    if(other.tag == "Player")
    {
        if (player.isShieldActive)
        {
            player.isShieldActive = false;

            anim.SetTrigger("OnEnemyDeath");
            speed = 0;
            Destroy(gameObject,1.0f);// enemyi yok et
        }
        else{
            player.Damage();
            anim.SetTrigger("OnEnemyDeath");
            speed = 0;
            Destroy(gameObject,1.0f);// enemyi yok et
        }
    }
    else if(other.tag == "Lazer")
    {

        Destroy(other.gameObject);//lazeri yok et
        anim.SetTrigger("OnEnemyDeath");
        speed = 0;

        Destroy(gameObject, 1.0f);// enemyi yok et
        player.score += 10;

    }
```

Asteroid ekleme



Hazır olan sprite'ı sürükle bırak yöntemiyle sahnemize bırakıyoruz. Eklemek istediğimiz componentleri ekliyoruz(collider, rigid body, vs.). Bu komponentleri inspector kısmından "Add Component" tuşundan kolaylıkla aratarak yapabiliriz.

Unutulmaması gereken şey circle colliderda "is trigger" işaretli olması ve Rigidbody kısmında "gravity scale" 0 olmasıdır.

Görselde gözüktüğü şekilde ayarlar kullanılabilir.

Bu astreoidimize "Astreoid_SC" adında bir script dosyasına gerekli kodlarımızı yazacağız.

Animator olarak gördüğünüz component ise bizim astreoidimizin yok olduğunda oluşacak animasyondur.

Astreoid_SC

```
void Update()
{
    transform.Rotate(Vector3.forward * rotatespeed * Time.deltaTime);
    transform.Translate(Vector3.down * speed * Time.deltaTime, Space.World);

    if (transform.position.y < -5.4f)
    {
        float randomX = Random.Range(-player.xVal, player.xVal);
        transform.position = new Vector3(randomX, player.yVal + 1, 0);
    }
}
```

Bu kısımda astreoidimizin kendi etrafında dönerek sahnenin altına düşecek şekilde gerekli transformlar gerçekleştirilmiştir. Önemli olan kısım transform.translate fonksiyonuna verdiğimiz Spacae.World parametresidir. Bu parametre nesnemizin -y yerine sahnenin altını hedeflemekte bu sayede aşağıya doğru dönerek ilerlemekte.

Çarpışma Kontrolü

```
Unity İletisi | 0 başvuru
void OnTriggerEnter2D(Collider2D other) // temasa giren
{
    if (other.tag == "Player")
    {
        if (player.isShieldActive)
        {
            player.isShieldActive = false;
        }
        else
        {
            player.Damage();
        }

        anim.SetTrigger("IsTouch");
        speed = 0;
        Destroy(gameObject, 1.0f); // enemy'i yok et
    }
    else if (other.tag == "Lazer")
    {
        Destroy(other.gameObject); // lazeri yok et
        anim.SetTrigger("IsTouch");
        speed = 0;
        Destroy(gameObject, 1.0f); // enemy'i yok et
        player.score += 10;
    }
}
```

Bu kısımda çarpışma kontrolü yapılmakta. Eğer astreoid player ile karşılaşırsa neler olacağını lazer ile karşılaşırsa neler olacağını göstermekte.

Anim.SetTrigger animasyon etkinleştirmesi içindir. Yani eğer gerekli etkileşimler gerçekleşirse astreoid'e tanımlanan patlama animasyonu aktif hale gelerek yok olacaktır. Ayrıca speed değişkenini de 0'larız ki animasyon bitene kadar yok olacak nesne ilerleyip player'a hasar vermesin.

Bu değişikleri yapabilmek için yan tarafta görüle değişkenleri tanımlamamız gerekir.

Animasyonlar için Animator anim;

player_SC scriptindeki değişkenlere ve fonksiyonlara erişebilmek için

private Player_Sc player; // Kalıtım alındı

player = GameObject.FindObjectOfType<Player_Sc>();

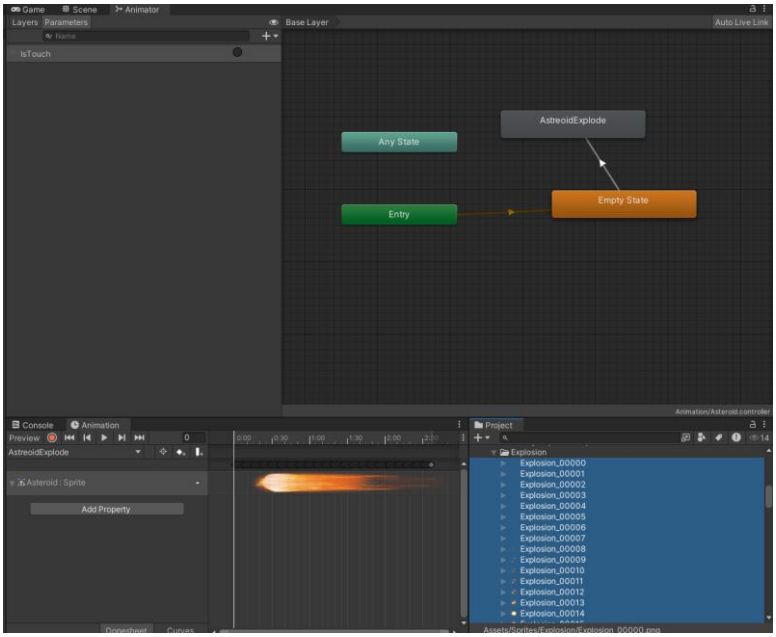
kodlarını bu scriptimize de eklemek gerekir.

```
[SerializeField]
private float speed = 1.0f;

[SerializeField]
private float rotatespeed = 20.0f;

[SerializeField]
private Player_Sc player; // Kalıtım alındı

private Animator anim;

// Start is called before the first frame update
Unity İletisi | 0 başvuru
void Start()
{
    player = GameObject.FindObjectOfType<Player_Sc>();

    if (player == null)
    {
        Debug.LogError("Player not found!");
    }

    anim = GetComponent<Animator>();
    if (anim == null)
    {
        Debug.LogError("Animator not found!");
    }
}
```

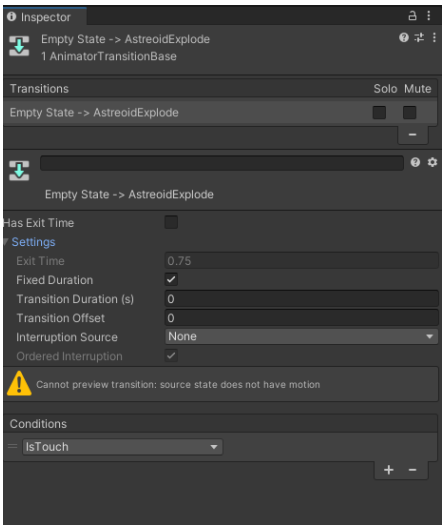
Astreoid Animasyonu

Diğer animasyonları yaptığımız gibi öncelikle nesnemize bir animasyon motoru oluşturuyoruz. Bunu animation sekmesinden create tuşuna tıklayarak yapabiliriz. Oluşan motorumuza hazırda olan spritelarımızı ekliyoruz.

Önemli olan kısım bu animasyonun ne zaman aktifleşeceğidir. Bunun için animatör sekmesinden yeni bir state oluşturup başlangıçta default state olarak atamak gerekir ardından gerçekleşmesini istediğimiz animasyona default state'ten bir transition oluşturmak. Bu sayede bir koşul ekleyerek istediğimiz zaman bu animasyonu çalıştırabileceğiz. Bunun için Animator>Parameters kısmından trigger türünde bir değişken tanımlıyoruz. Bu değişken aktifleştirildiği zaman animasyonumuz çalışacak. Bunun kontrolünü astreoid_SC başlıklı yerde gözüken şekilde aktif edebiliriz.



Dip Not: Trigger değişkenimizi isTouch adında tanımladık bu ad ile kodlarımızda çağırmamız gerekiyor



Yan tarafta görüldüğü üzere "has exit time" kapalı ve "transition duration" 0 olarak tanımlanmıştır. Bu sayede daha yumuşak geçişler sağlanacaktır.

Conditions kısmında ise IsTouch koşulumuzu eklemiş bulunmaktayız bu sayede koşullu olarak animasyonumuzu aktifleştirebileceğiz.

Asteroid spawn routine

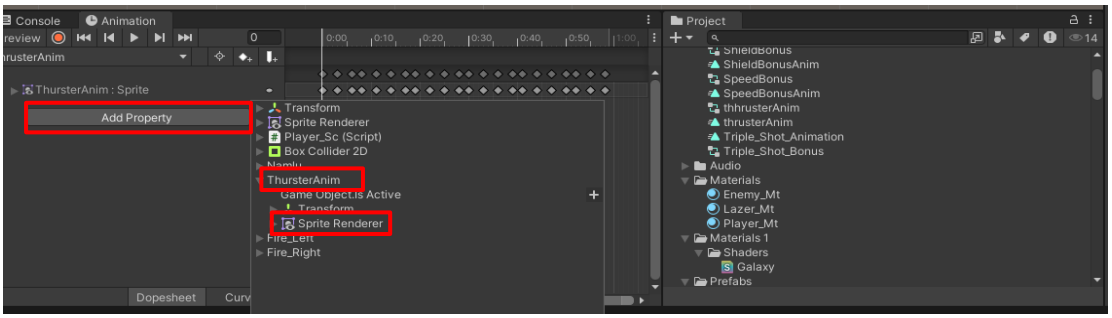
Bu aşamada bir spawnManager_Sc dosyamızda ufak bir değişiklik yapmamız yeterlidir. Bu değişiklik enemy için oluşturduğumuz coroutine'ni hem astreoid hem de enemy için kullanmak olacaktır. Bunun için coroutine aldığı değişkenlere zaman ve gameobject eklemek. Bu sayede gerektiği zaman istediğimiz başka düşmanları veya yok edilmesi gereken nesneleri de eklemiş olacağız.

```
[SerializeField]
GameObject Astreoid;
[SerializeField]
GameObject enemyPrefab;


Unity İletisi | 0 başvuru
void Start()
{
    player = GameObject.FindObjectOfType<Player_Sc>();
    StartCoroutine(SpawnEnemyRoutine(enemyPrefab,5.0f));
    StartCoroutine(SpawnEnemyRoutine(Astreoid, 10.0f));
    StartCoroutine(SpawnBonusRoutine());
}
2 başvuru
IEnumerator SpawnEnemyRoutine(GameObject prefab,float spawntime)
{
    while (player.Health > 0)
    {

        Vector3 position = new Vector3(Random.Range(-player.xVal, player.xVal), player.yVal + 2, 0);
        GameObject new_enemy = Instantiate(prefab, position, Quaternion.identity);
        new_enemy.transform.parent = enemyContainer.transform;
        yield return new WaitForSeconds(spawntime);
    }
}
1 başvuru
```

Bu değişiklikler sayesinde astreodimiz 15snde bir yeniden oluşup oyuncumuza zorluk oluşturacaktır.

Thruster ekleme, sağ ve sol motor için hasar animasyonlarının eklenmesi

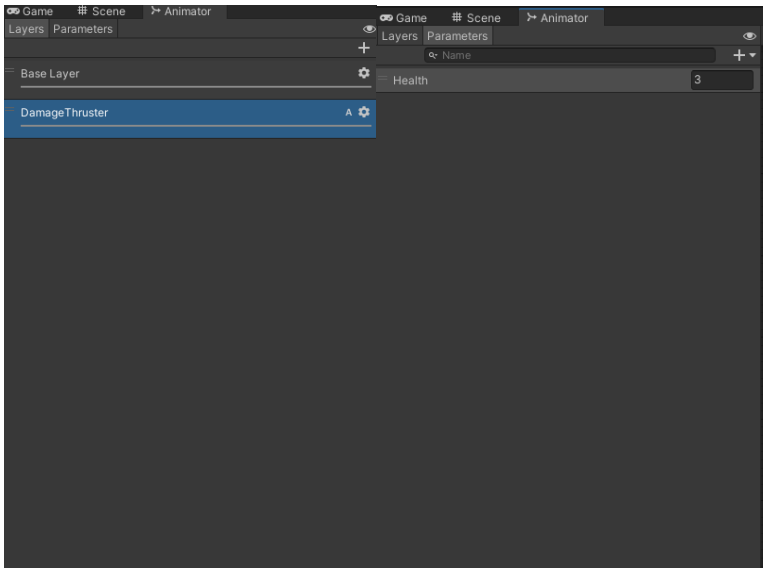
Öncelikle Thruster sprite'ını sahnemize süürkle bırak yapıp istediğimiz ölçüleri ayarlıyoruz. İstediğimiz konuma koyuyoruz. Ardından bu spriteları player nesnesinin altına hiearchy ekliyoruz ki beraber hareket etsin. Ardından her animasyonda yaptığımız gibi bu animasyonumuz için bir animation motoru oluşturuyoruz. Ve spritelarımızı sürükle bırak yöntemiyle animasyon motorumuza ekliyoruz. Burada önemli olan kısım hiearchy oluştuduğumuz için bu sürükleyip bıraktığımız animasyon spritelarını hiearchy kısmından thruster sprite'ını bulup ekliyoruz.

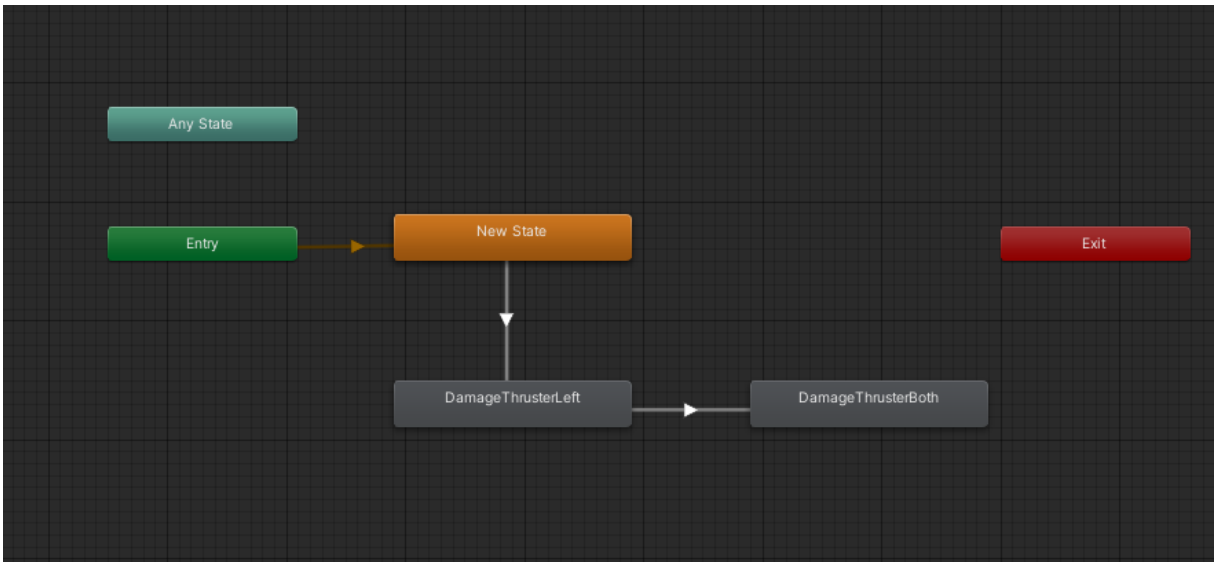


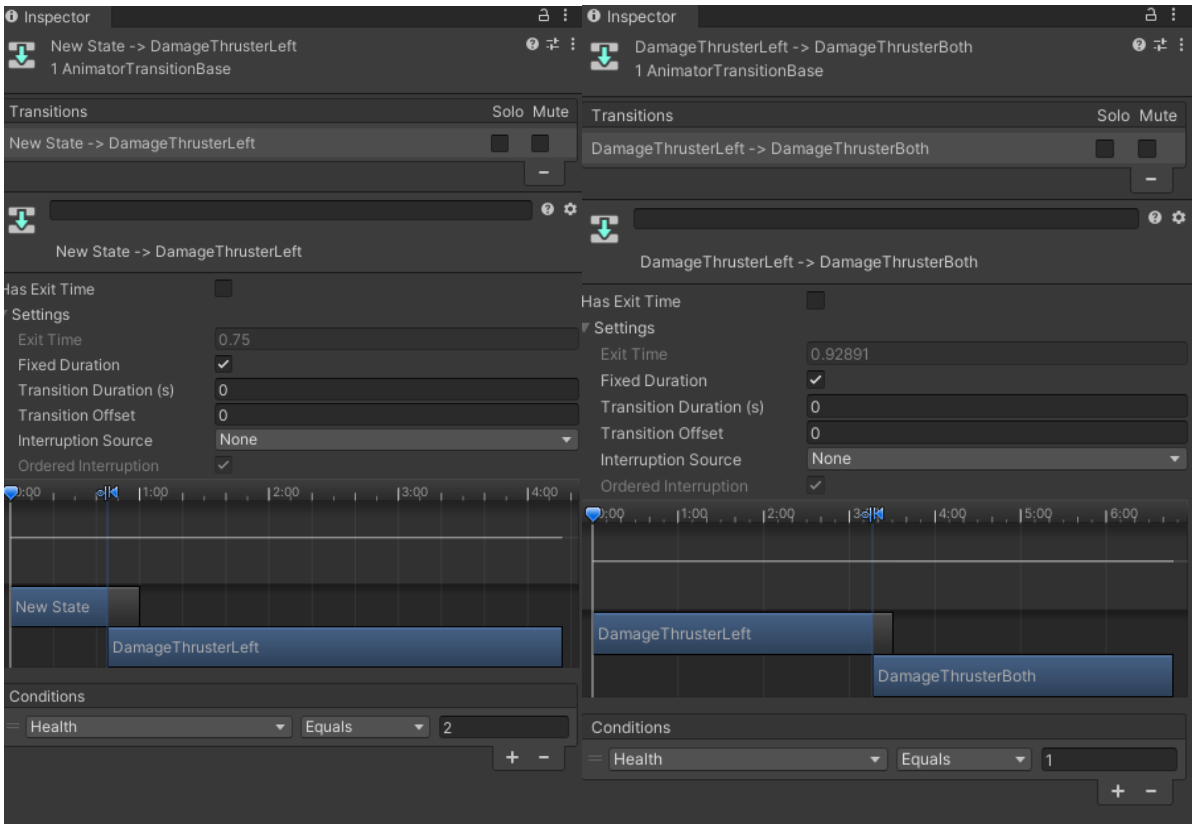
Bu şekilde sprite renderer içinde sprite'ı bulup ekliyoruz ve istediğiniz animasyon spriteları bu property'nin karşısına ekleyebilirsiniz.

Default olarak açık kalması gereken bir animasyon olduğu için herhangi bir koşul eklemiyoruz ve direk bu şekilde bırakabiliriz.

Thruster damage yediği zaman oluşacak hasar animasyonlarını bu thurster için yaptığımız adımları aynen yapıyoruz. Farklı olan kısım ise bir koşu bağlı olarak çalışacak olmasıdır. Animator>Layer kısmından başka bir katman oluşturuyoruz ki diğer thurster animasyonundan başka bir zaman ve bir koşula bağlı olarak çalışabilsin. İki motor için farklı animasyonlar ekledikten sonra ilk koşulumuz can değişkenimizin 2ye eşit olduğunda bir taraf 1e eşit olduğunda diğer taraf animasyonu çalışacak şekilde koşul ekliyoruz.







Transitionlardaki koşullar ve gerekli ayarlar bu şekilde ayarlanarak bu kısımdaki işimizi bitirmiş oluyoruz. Bu ayarları bir önceki animasyonda anlatılmıştır.

Player_sc dosyasındaki damage fonksiyonuna bu int değişkenini güncellemek ve animasyonu çalıştırabilmek için şu şekilde bir ekleme yapıyoruz.

```
public void Damage()
{
    Health -= 1;
    anim.SetInteger("Health", Health);
    if (Health <= 0)
    {
        Debug.Log("GAME OVER");
        UI_Manager_Sc.UpdateLives(Health);
        Destroy(gameObject);

    }
    UI_Manager_Sc.UpdateLives(Health);

}
```

Bu sayede hasar aldıkça uzay aracımızın hasar aldıkça oluşacak hasar animasyonları bu şekilde oluşmuş olacak.

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_10