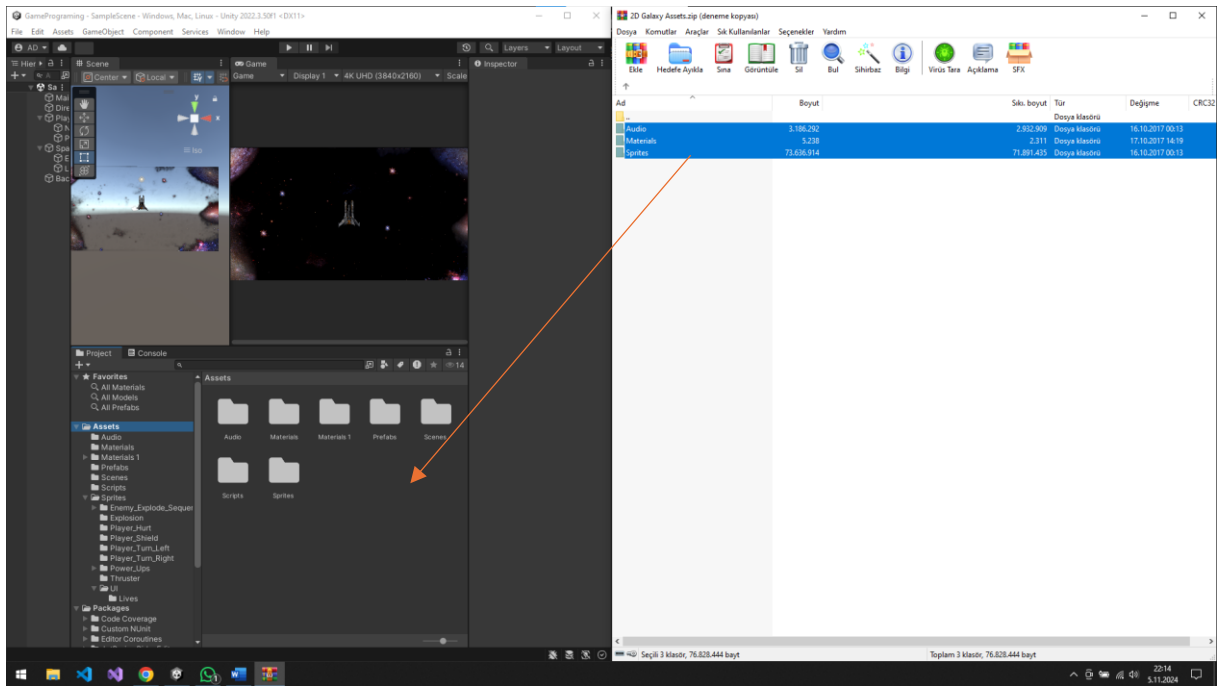


5. Hafta Raporu – Aleks Dulda 21360859025

Materials, Audio ve Sprites asset'lerinin projeye import edilmesi:

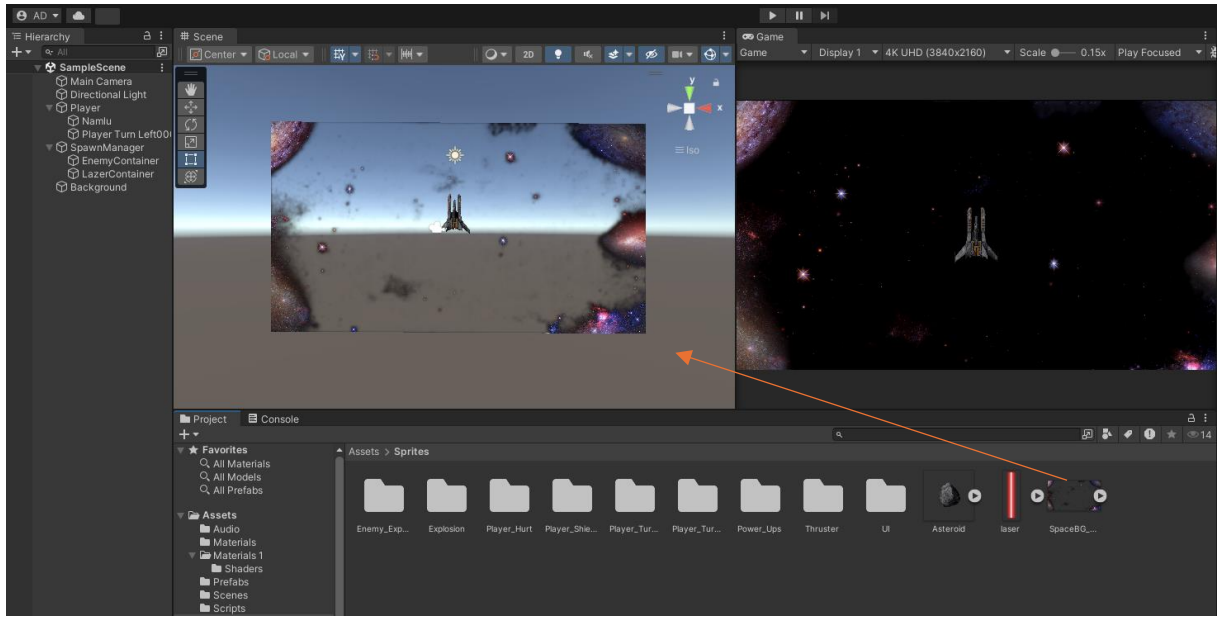
Bu kısımda e-Kampüste bulunan asset dosyalarını bilgisayarımıza indiriyoruz ve rar dosyasına sıkıştırılmış klasörleri Unity'de bulunan "Project" penceresindeki "Assets" kısmına sürükleyerek projemize bu hazır assetleri import etmiş oluyoruz.



Bu şekilde bizimle paylaşılan hazır assetleri projemize başarılı bir şekilde eklemiş oluyoruz.

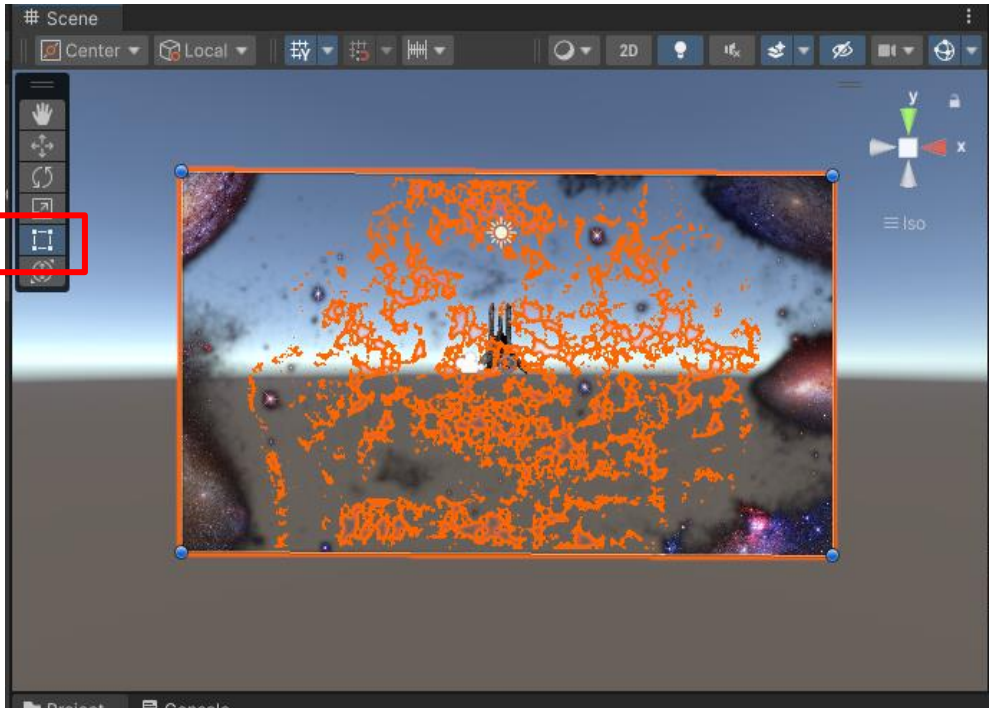
Arkaplan için sprite kullanımı:

Yüklediğimiz assetlerde “Sprites” klasörünün altında bir çok iki boyutlu çizimler bulunmakta. Bunlardan bir tanesi ise oyunumuz için hazırlanan arkaplan çizimi. Bu arkaplanı kullanmak için “Scene” kısmına bu çizimimizi sürükleyip bırak yöntemiyle oyunumuza ekliyoruz.



Ardından gerekli boyutlandırmayı yapabilmek için “Scene” kısmından “Rect Tool” aracını kullanarak sprite’ın kenarlarından istediğimiz boyutlandırmayı gerçekleştiriyoruz.

Rect Tool



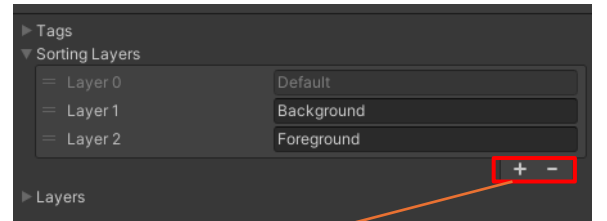
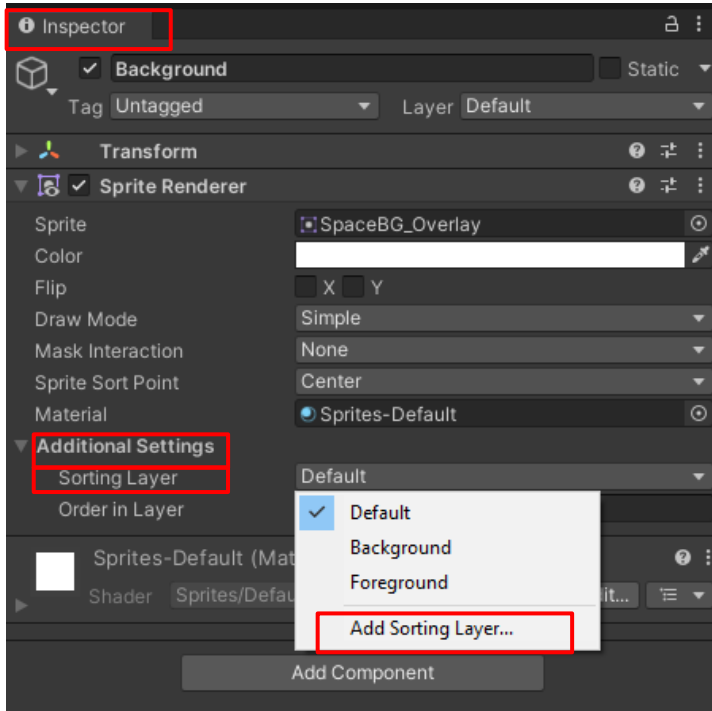
Sorting layer kullanımı:

Eklediğimiz spriteler'in belirli bir sırada olması gerekmektedir. Örneğin, oyunu oynarken bir player objesi arkaplanın arkasında olmaması gerekmektedir. Bu durumun oluşmaması için hiyerarşik bir düzen gerekir. Bunun için Sorting layer kullanıyoruz.

Örneğin arkaplan spirte için bu sorting layer uygulamasını gerçekleştirecek olursak şu aşamaları yapmamız gerekir.

Unityde bulunan "Inspector" kısmında "Additional Options" kısmında bulunan "Sorting Layer" kısmında "Add Sorting Layer..." kısmını kullanarak oluşturacağımız oyun projesinde gerekli olan veya olacak katmanlar oluşturabiliriz.

Bu oluşturduğumuz katmanların sırası aslında bize şu bilgiyi de verir. Küçük numaralı olan layer daha altta büyük numaralı olan layer daha üstte gözükecek şekilde konumlandırılır. Örneğin bu oyunda eğer arkaplana bir "Background" layeri vereceksek bu layer en küçük numaralı katman olmalı. Diğer oyun objeleri (Örneğin: Player, enemy, vb.) bu "Background" layerından daha büyük numaralı bir layerdan etiket almalı.



+ tuşuna basılarak yeni layerler eklenebilir.

Bu işlem projeye eklenen veya eklenecek diğer objeler için uygulanabilir. Bu adımları uygulayarak projemizdeki objelerin hiyerarşik olarak konumlandırılmasını sağlamış oluruz.

Enemy ve Laser'den tekrar prefab oluşturma:

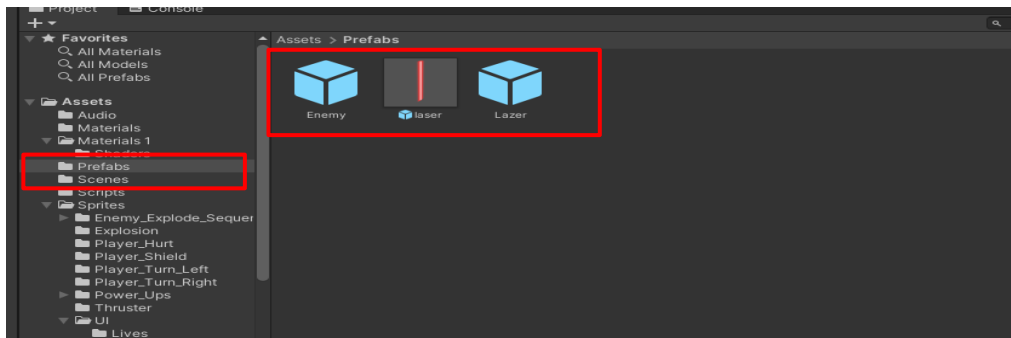
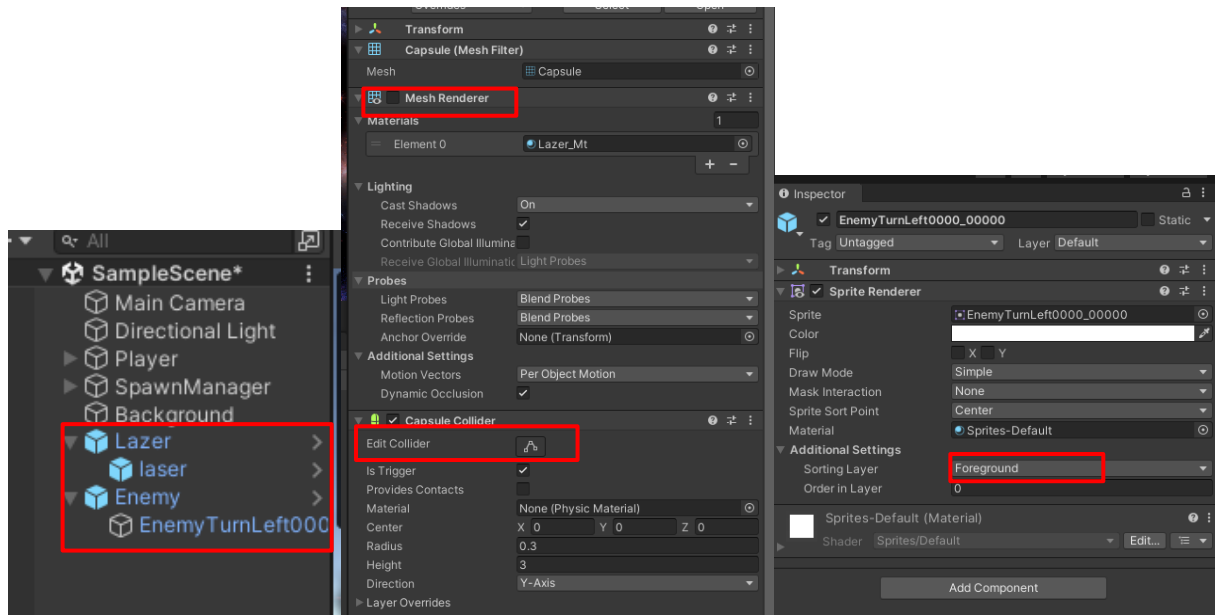
Enemy ve laser prefablarını tekrar oluşturmak yerine var olan prefabların üstüne bizlerle e-kampüste paylaşılan assetleri bu prefablara uygun olacak şekilde optimize ettim.

Var olan prefablara child olarak verilen assetleri ekledim. Pozisyonlarını var olan objenin orta noktasına aldım. Collider kısmından assette olan çizimin ortalama genişliğine göre diğer objeler ile etkileşime geçecekleri boyutları ayarladım. Ardından “Mesh Renderer” kısmını kapatarak sadece asseti gösterecek ve var olan obje gözükmeyecek şekilde ayarlanmış oldu.

Uzun lafın kisası zaten hali hazırda bulunan kutu objesi (enemy) ve capsule objesi (lazer) üzerine sadece asset eklemiş olduk ve colliderları ayarladık.

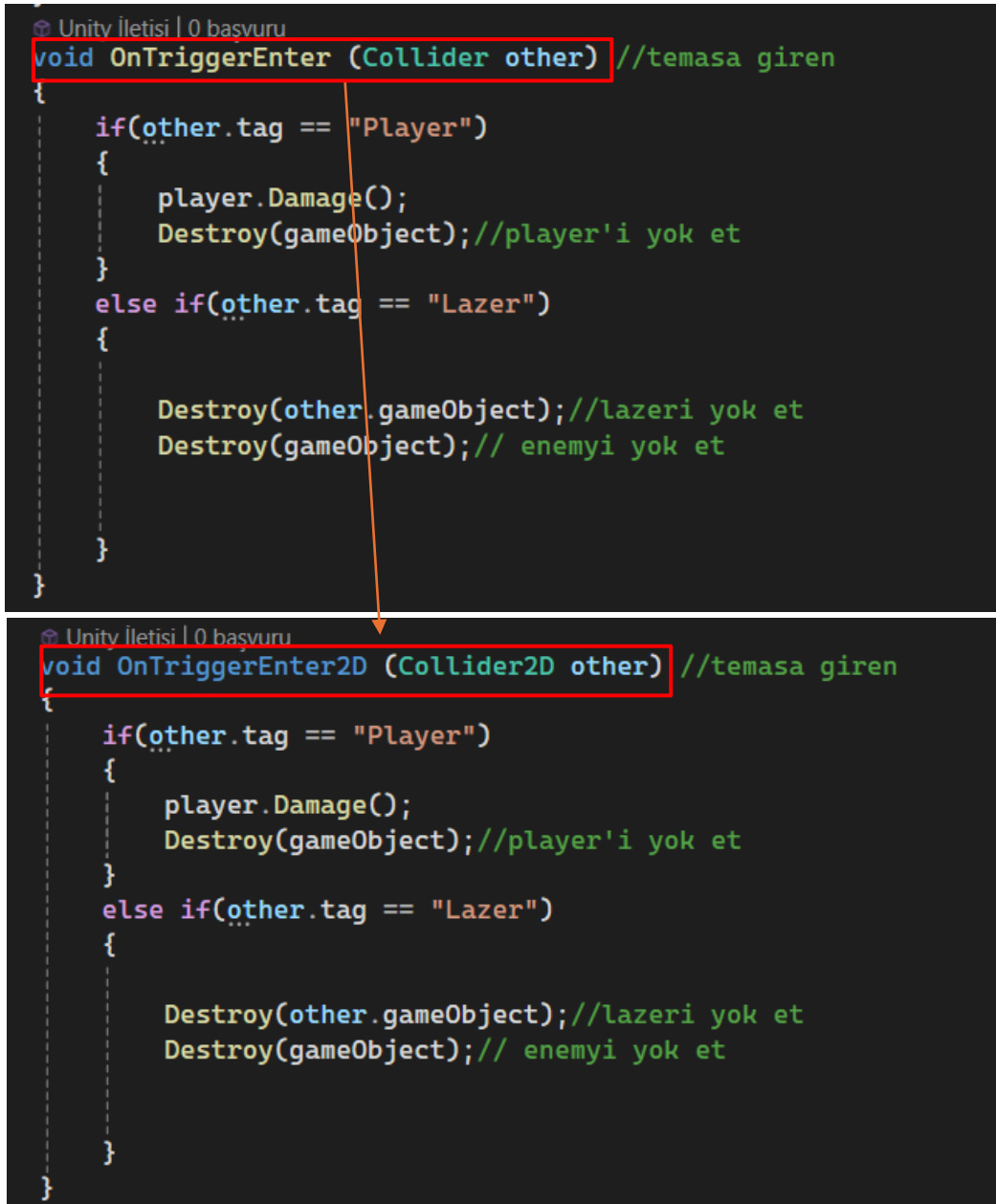
Colliderları değiştirmek istediğimiz objenin colliderında bulunan “Edit collider” kısmından istediğimiz boyutlarda colliderı güncelleyebiliriz.

Arkaplanı ayarlarken kullandığımız layerları bu Enemy ve Laser assetleri için de uyguladık ve bunlara da “Foreground” etiketiyle önde gözükecek şekilde ayarladık.



Bu şekilde Enemy ve Laser'den tekrar prefab oluşturmuş olduk.

Kodda 2D için gerekli değişikliklerin yapılması:



```
Unity İletisi | 0 başvuru
void OnTriggerEnter (Collider other) //temasa giren
{
    if(other.tag == "Player")
    {
        player.Damage();
        Destroy(gameObject); //player'i yok et
    }
    else if(other.tag == "Lazer")
    {
        Destroy(other.gameObject); //lazeri yok et
        Destroy(gameObject); // enemyi yok et
    }
}

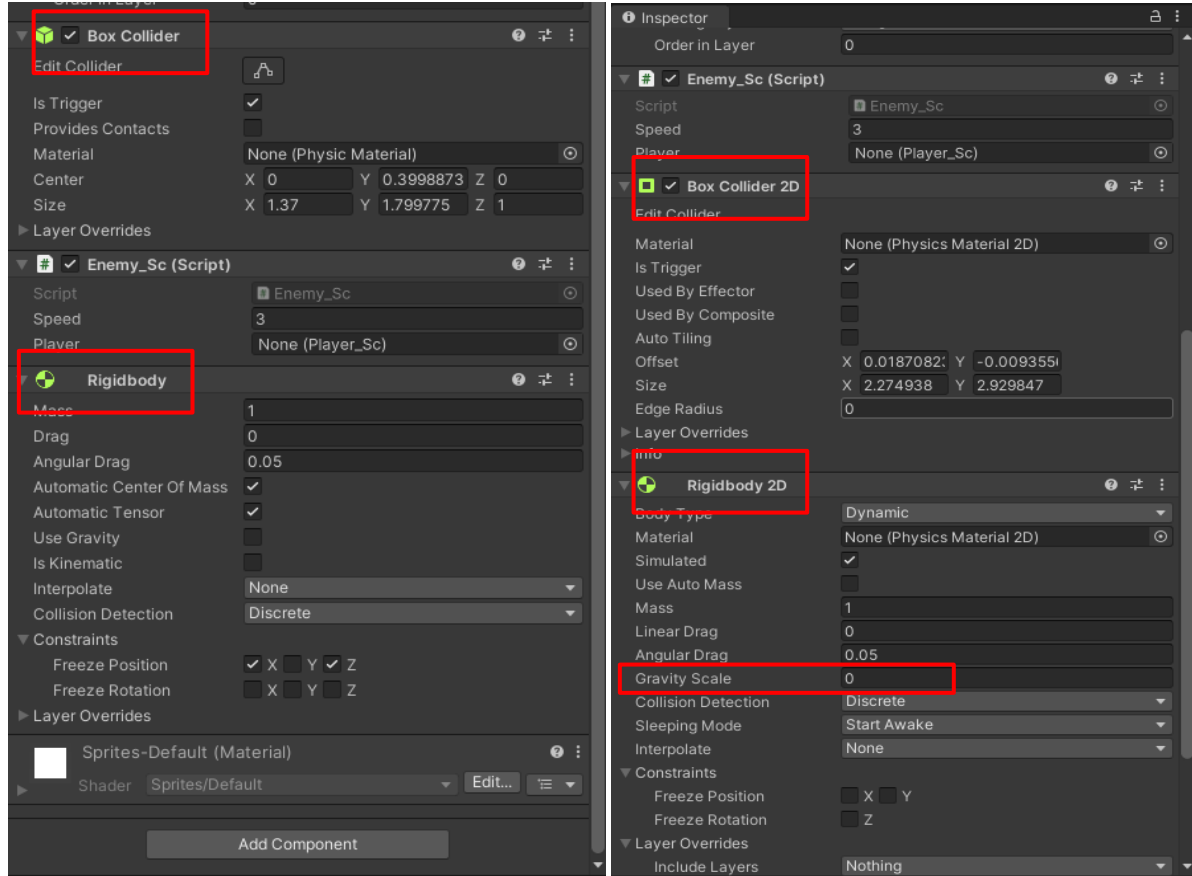
Unity İletisi | 0 başvuru
void OnTriggerEnter2D (Collider2D other) //temasa giren
{
    if(other.tag == "Player")
    {
        player.Damage();
        Destroy(gameObject); //player'i yok et
    }
    else if(other.tag == "Lazer")
    {
        Destroy(other.gameObject); //lazeri yok et
        Destroy(gameObject); // enemyi yok et
    }
}
```

Bu değişiklik, Unity'de 2D ve 3D fizik motorlarının ayrı olmasından kaynaklanır; OnTriggerEnter ve Collider, 3D nesnelerle etkileşimi sağlarken, OnTriggerEnter2D ve Collider2D 2D nesnelerle çalışır. Bu ayrım, 2D projelerde gereksiz 3D hesaplamalardan kaçınıp performansı artırmak için yapılmıştır.

Player, Enemy ve Laser nesnelerinin iki boyutlu olarak tekrar oluşturulması:

Bu kısımda elimizde var olan bu Player, Enemy ve Laser objelerinin componentlerinde bazı değişikliğe gideceğiz.

Bu objelerde bulunan Collider ve Rigidbody gibi componentlerini 2D versiyonları ile değiştireceğiz örnek olarak birtanesini aşağıdaki fotorafta görebilirsiniz.



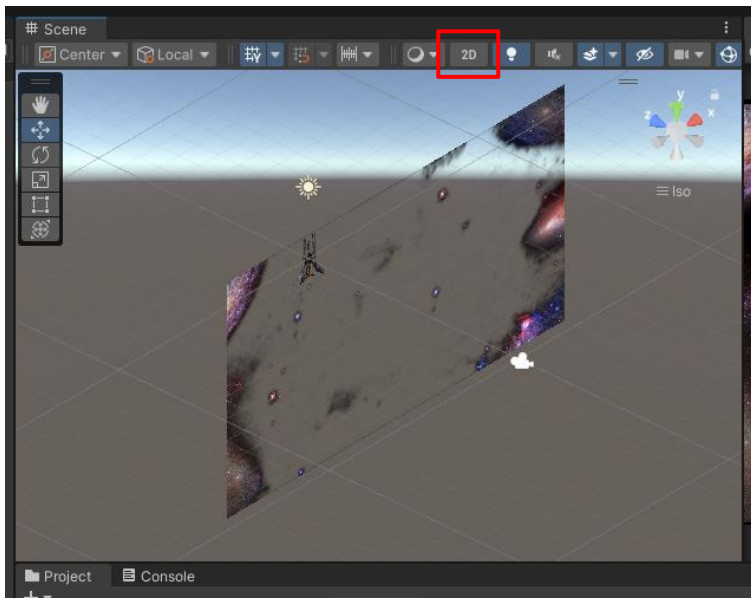
Bu değişiklikler ile var olan Player, Enemy ve Laser objelerini 2D olarak oluşturmuş olduk.

Rigidbody 2D componentinde “Gravity Scale” kısmını 0 yapmayı unutmayın.

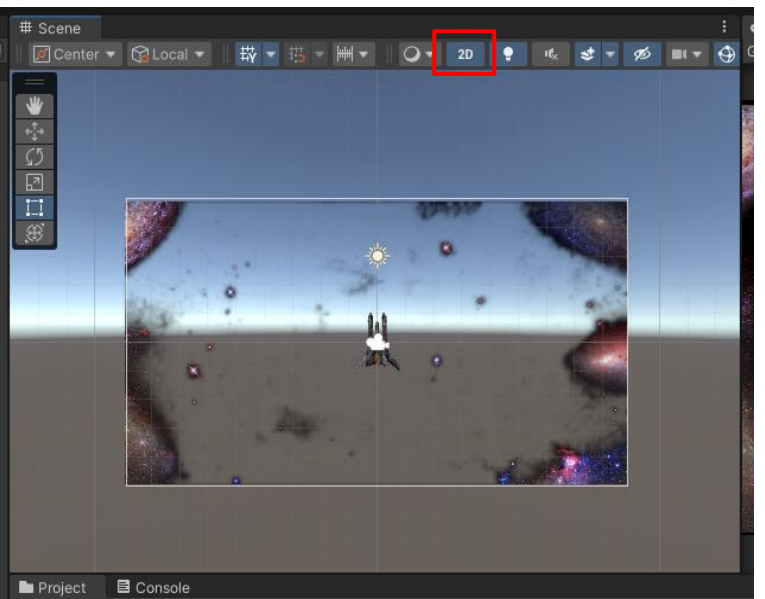
Perspektif görünümü, 2D-3D görünüm arasında geçiş yapma:

Unity’de 2D-3D görünüm arasında geçiş yapmak oldukça basittir. 2D ile 3D görünüm arasında geçiş yapabilmek için “Scene” kımına tıklanmış şekilde bilgisayarımızın “2” tuşuna basarsak 2D görünüm, “3” Tuşuna basarsak 3D bir görünüm almış oluruz.

Diğer bir yöntem ise hali hazırda bulunan 2D tuşuna basabilirsiniz.



3D



2D

Bu şekilde perspektif görünümü, 2D-3D görünüm arasında geçiş yapmış olduk.

Kaynakça

[https://en.wikipedia.org/wiki/Unity_\(game_engine\)](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_5