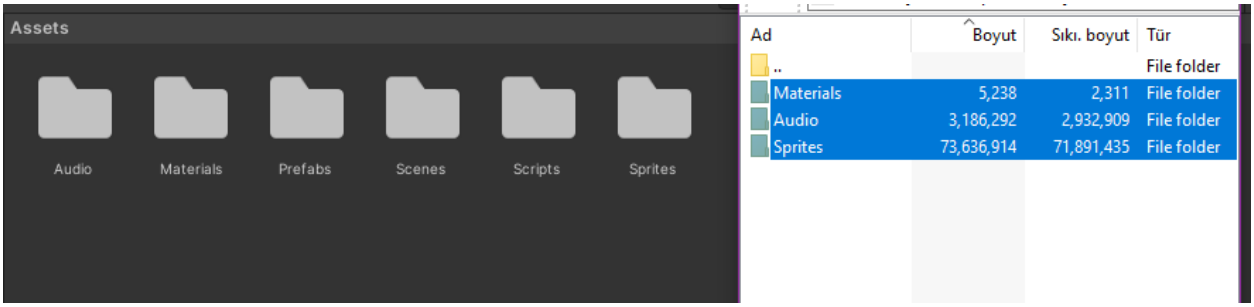
5.Hafta Ders Raporu – Batuhan Şengül

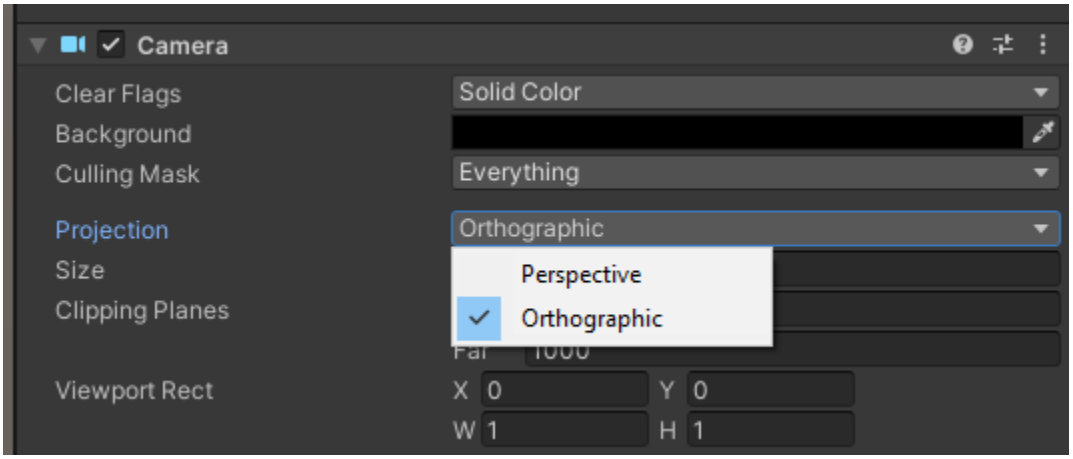
Dosyaları Unitye Aktarmak

İstenilen dosyalar dosya gezgininden sürüklenerek Unitye aktarılır. Bu oyun için bizimle paylaşılan dosyalar bu şekilde oyun dosyalarına eklenmiştir.



2 Boyut Ayarları

2 boyutlu bir görüntü istediğimiz için kameranın derinlik bilgisini işlemesine ihtiyacımız yoktur. “MainCamera” oyun nesnesindeki “Camera” bileşeninden görüntü işleme(Projection) ayarı Perspecitive -> Orthographic ayarlanır. Bu şekilde kamera görüntüsü çıktısı 2 boyutlu olacaktır.



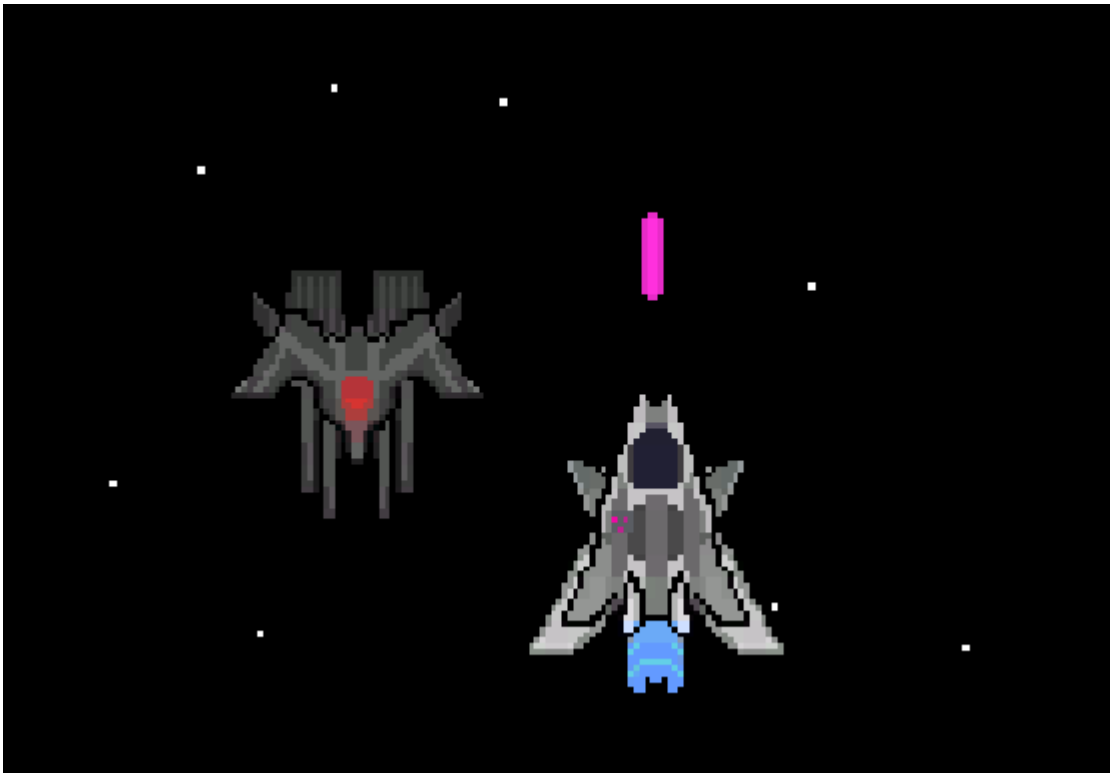
Sprites

Sprite, 2D oyunlarda kullanılan grafiklerin ve görsellerin temel yapı taşıdır. Genellikle bir karakter, nesne veya arka plan gibi oyun içinde gösterilmek istenen herhangi bir şey sprite olarak tanımlanabilir.

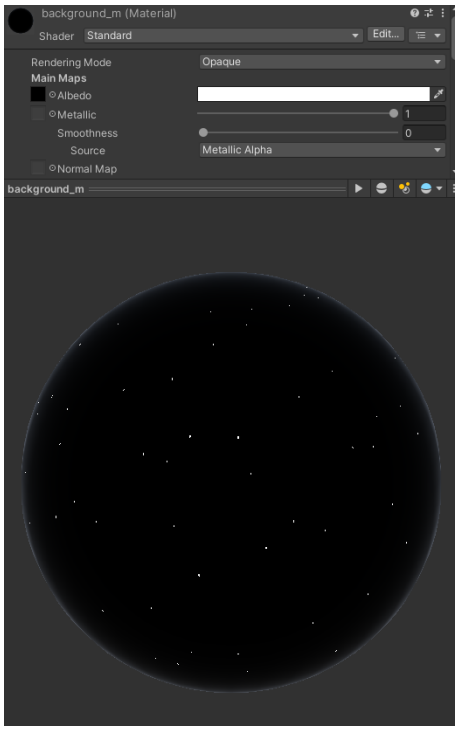
Unity içine eklenen bir sprite sahneye sürüklenerek oyuna eklenebilir. Bu oyunda:

- **Kullanılan Spritelar**

Bu oyun için bizimle paylaşılanlar dışına kendi tasarımlarım kullanılmıştır.



- **Arkaplan**



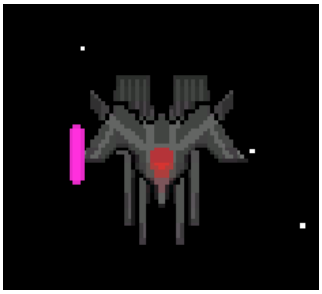
Aşaüıdaki görsel texture olarak kullanan bir materyal oluşturulmuştur. Oluşturulan materyal Sahnede dik olarak ayarlanan bir plane nesnesine atanmıştır. Kod ile okunan input değerlerine bağlı materyalin offseti değiştirilerek oyuncu sanki uzayda hareket ediyormuş gibi hissettirilmek istenmiştir.

Animasyon kodu:

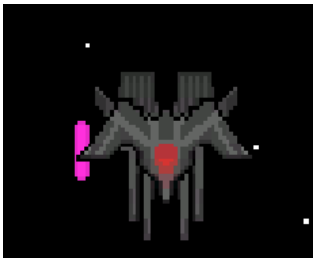
```
Unity Script (1 asset reference) | 0 references
public class BackgroundEffects : MonoBehaviour
{
    public float horizontalScrollSpeed = 0.1f;
    public float verticalScrollSpeed = 0.1f;
    float horValue;
    [SerializeField]private Renderer rend;
    private PlayerMovement pm;
    float offsetX, offsetY;
    Unity Message | 0 references
    void Start()
    {
        rend = GetComponent<Renderer>();
        pm = FindObjectOfType<PlayerMovement>();
    }

    Unity Message | 0 references
    void Update()
    {
        if (pm.lives>0)
        {
            horValue = Input.GetAxis("Horizontal");
            offsetY= Time.time * -horizontalScrollSpeed;
            if (horValue > 0)
            {
                offsetX = Time.deltaTime * -verticalScrollSpeed;
            }
            else if (horValue < 0)
            {
                offsetX = -Time.deltaTime * -verticalScrollSpeed;
            }
            else
            {
                offsetX = 0;
            }
        }
        else
        {
            offsetX = 0;
        }


        rend.material.mainTextureOffset = new Vector2(rend.material.mainTextureOffset.x+offsetX, offsetY);
    }
}
```

- **Sprite Layer**

Sprite layer'lar oyun içindeki öğelerin görsel sırasını kontrol ederek hangi sprite'ın önde, hangisinin arkada görüneceğini belirler. Örnek vermek gerekirse:



- Lazerin sprite layer'ı 10 , düşmanın ki -10 olduğu durumda lazer düşmanın üstünde gözükür.

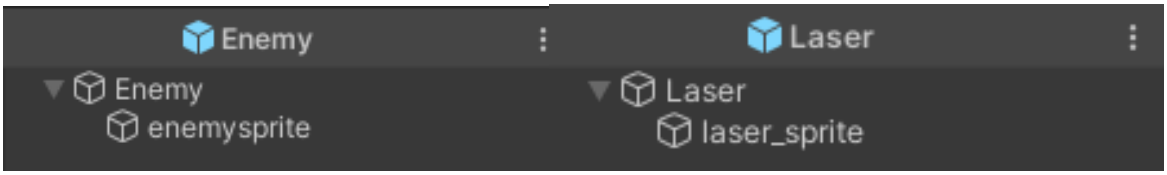


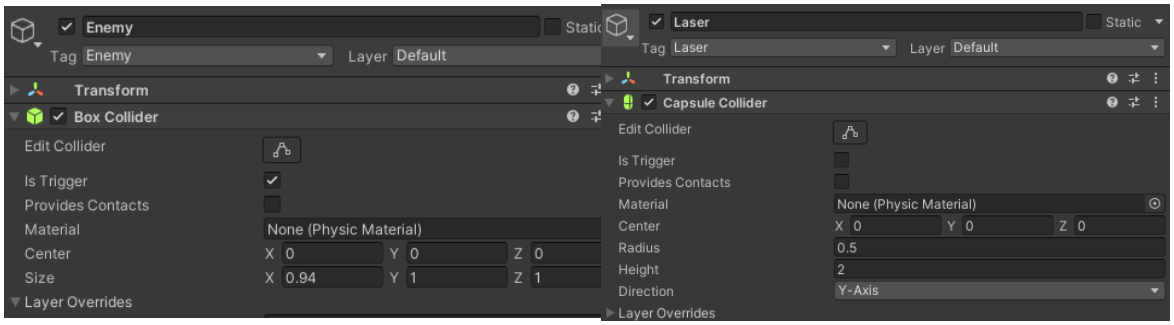
- Lazerin sprite layer'ı -10 , düşmanın ki 10 olduğu durumda lazer düşmanın altında gözükür.

Bu oyun için sprite layerlar en üsten en alta sırayla Player>Laser>Enemy>Background olarak ayarlanmıştır.

Prefab Güncellemeleri

Enemy ve Laser prefablarine spritelar eklenmiştir. İstenenden farklı olarak 2d colliderlara geçiş yapılmamıştır sadece boyutları düzenlenmiştir.





Ekstralar

- **PlayerMovement – HandleMovement Değişiklikleri**

Oyunu daha dinamik yapmak adına hareket için bir input algılandığında ona göre PlayerSprite 'ı sağa ya da sola döndürülmesi/yatırılması sağlanmıştır. Bu istenilen için eklenen değişkenler:

```
[SerializeField] private GameObject playerSprite;
public float tiltAmount = 5f;
public float tiltSpeed = 5f;
public float maxTiltAngle = 5f;
```

Eklenen kod kısmı (Yeşil Kutu):

```
1 reference
public void HandleMovement()
{

    horizontalVal = Input.GetAxis("Horizontal");
    verticalVal = Input.GetAxis("Vertical");


    //Oyun nesnesinin Sag veya Sola haraketi

    transform.Translate(new Vector3(horizontalVal * speed * Time.deltaTime, verticalVal * speed * Time.deltaTime, transform.position.z));

    //Ekran keanar sinirlari
    transform.position = new Vector3(Math.Clamp(transform.position.x, -xBorderValue, xBorderValue),
        Math.Clamp(transform.position.y, -yBorderValue, yBorderValue),
        transform.position.z);


    targetZRotation = -horizontalVal * tiltAmount;


    float currentZRotation = playerSprite.transform.localEulerAngles.z;

    if (currentZRotation > 180) currentZRotation -= 360;

    float zRotation = Mathf.Lerp(currentZRotation, targetZRotation, Time.deltaTime * tiltSpeed);


    zRotation = Mathf.Clamp(zRotation, -maxTiltAngle, maxTiltAngle);

    playerSprite.transform.localEulerAngles = new Vector3(
        playerSprite.transform.localEulerAngles.x,
        playerSprite.transform.localEulerAngles.y,
        zRotation
    );
}
```

Bu kod ile girdiye (yatay eksen) bağlı döndürme yapması sağlanmıştır. Oyun içinde şu şekilde gözükmektedir:

Sola ve Sağa yatık Spritelar

- **2 Boyutlu Collider Kullanımı**

2D colliderlar, 3D ortamda kullanıldığında çeşitli uyumsuzluklar ve performans sorunları çıkarabilmektedir. 2D colliderların 3D uzaydaki sınırlı etkileşim kabiliyetleri nedeniyle çarpışma hesaplamalarında doğruluk kaybı veya hatalı sonuçlar meydana gelebilir. Özellikle 3D ortamın fizik motoru, farklı boyutlardaki collider türleriyle çalışırken daha karmaşık hesaplamalara ihtiyaç duyabilir ve bu durum performansı olumsuz etkileyebilir. Ayrıca, 2D colliderlar z eksenini hesaba katmadığından beklenmeyen çakışmalar ve geçiş hataları oluşabilmektedir.

3D ortamda tasarladığım 2D başka projelerimde karşılaşılan sıkıntılar, ileri dönük ve geliştirilebilirliğe daha açık bir proje ortaya çıkarmak istendiğinden bu projede 3D colliderların tercih edilmesi uygun bulunmuştur. Örnek başka birkaç sebepte bu blog sayfasında gösterilmiştir.

https://discussions.unity.com/t/is-it-ok-to-use-2d-colliders-with-3d-models/107973

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/4.Hafta

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

Hazırlayan

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