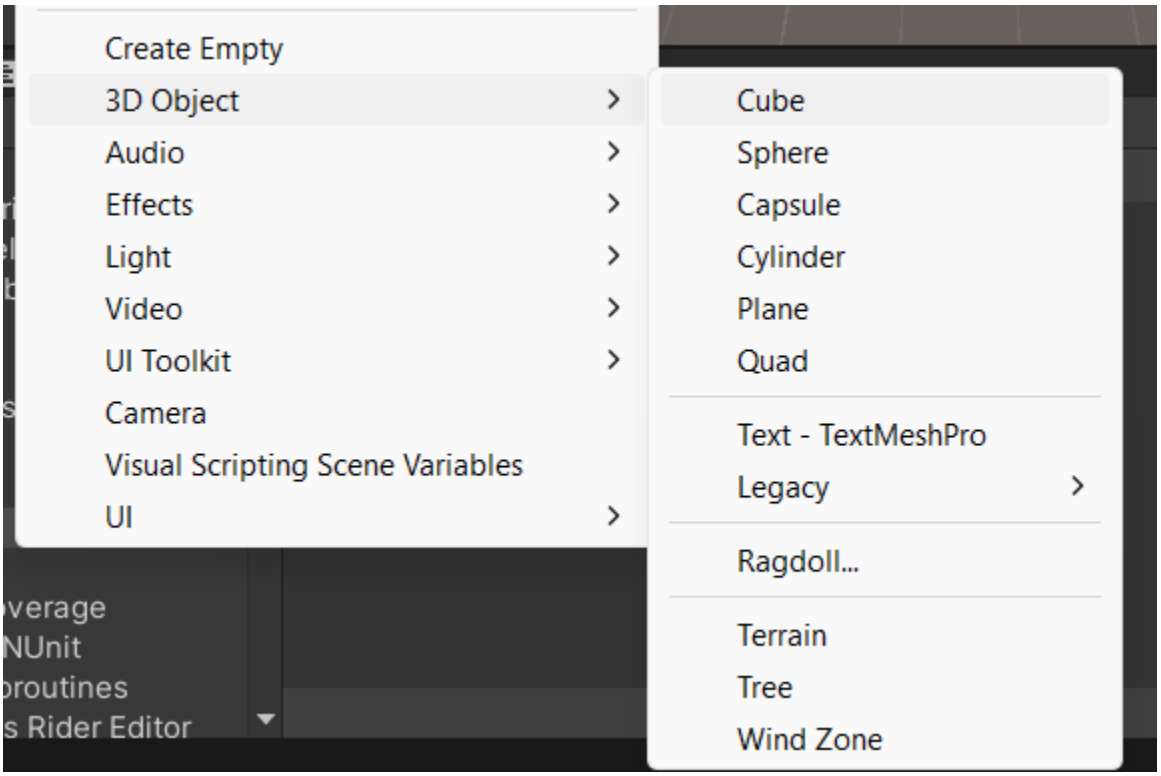
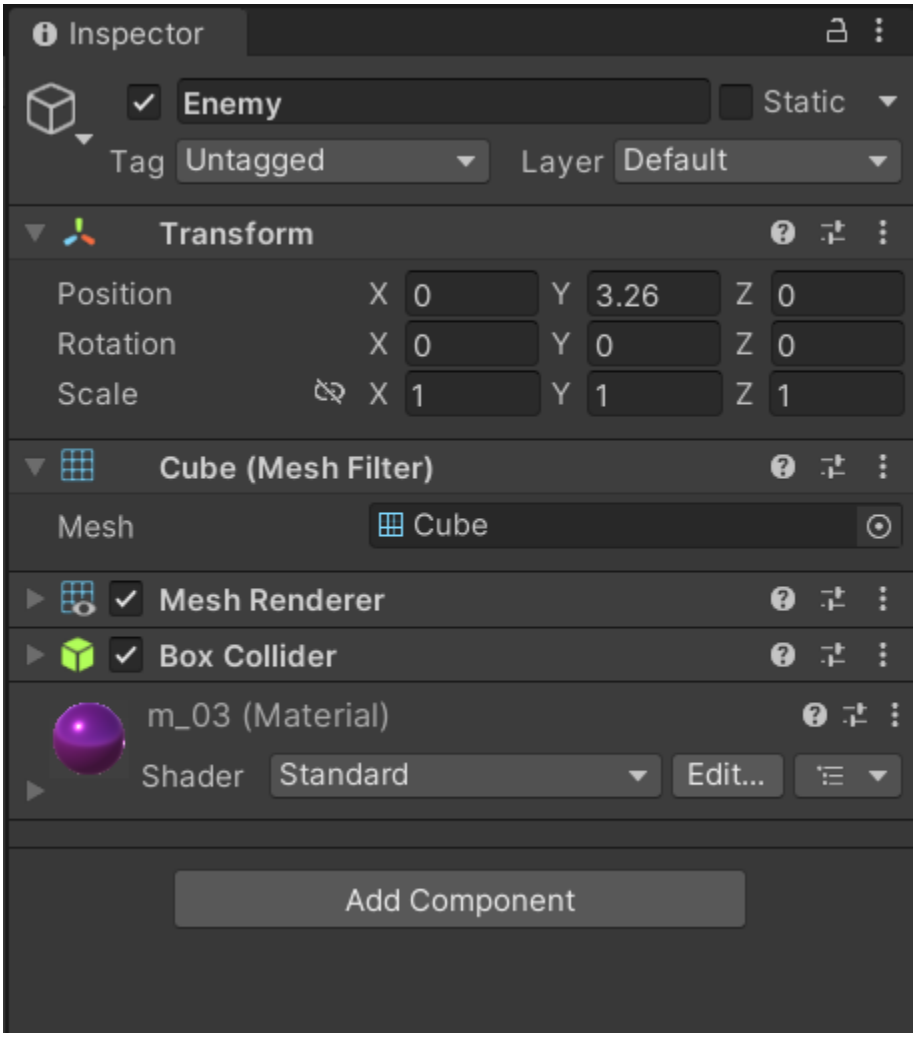
3.Hafta Ders Raporu – Batuhan Şengül

Prefab Oluşturma – “Enemy”

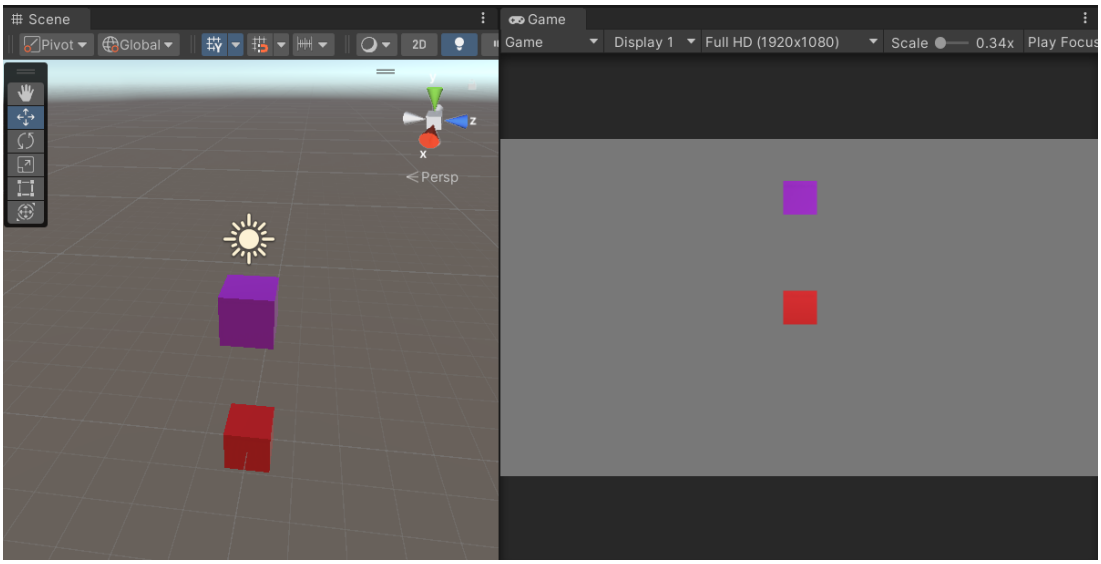
Hierchy sağ tıklama -> 3D object -> Cube seçilir. Bu örnekte yapacağımız enemy modelini oluşturmuş oluruz.

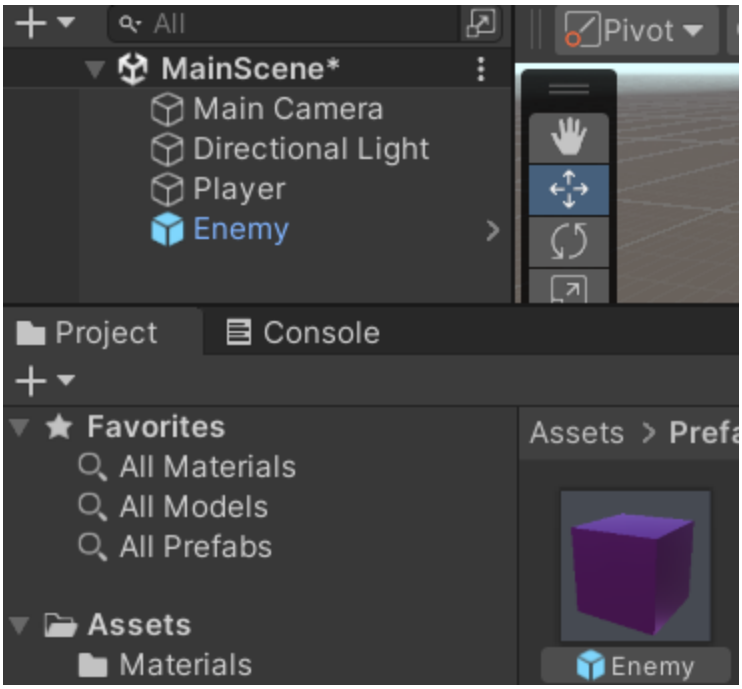


Oluşturulan küp oyun nesnesini Inspectordan ismi ayrıca önceki haftada gösterildiği gibi bir materyal oluşturulup-düzenlenip bu oyun objesine atanır.



Oluşturulan nesne game ve scene pencerelerinde aşağıdaki gibi gözükmektedir.





Oluşturulan oyun nesnesi Hierchy den Project penceresinde önceden oluşturulan "Prefabs" klasörüne sürüklenerek prefab haline getirilir. Eğer Hierchy de oyun nesnesi mavi renkle yazmaya başladıysa başarılı bir şekilde prefab oluşturulmuştur demektir.

Sahne üstündeki lazer oyun nesnesinde yapacağımız değişiklikler prefabde istenirse güncellenebilir ya da geri prefabdaki gibi eski haline getirilebilir.

Enemy

Rigidbody Bileşeni(Önceki Haftadan)

Rigidbody bileşeni, bir nesnenin fizik motoruyla etkileşime geçmesini sağlayarak, onu gerçek dünya fizik kurallarına (kütle, yerçekimi, kuvvetler, sürtünme vb.) tabi tutar. Bu bileşen, 3D ve 2D oyunlarda fiziksel etkileşimlerin yönetilmesi için çok önemli bir rol oynar. Eğer bir oyun nesnesine **Rigidbody** eklenmişse, bu nesne fiziksel hareketleri, çarpışmaları ve kuvvetleri simüle eder.

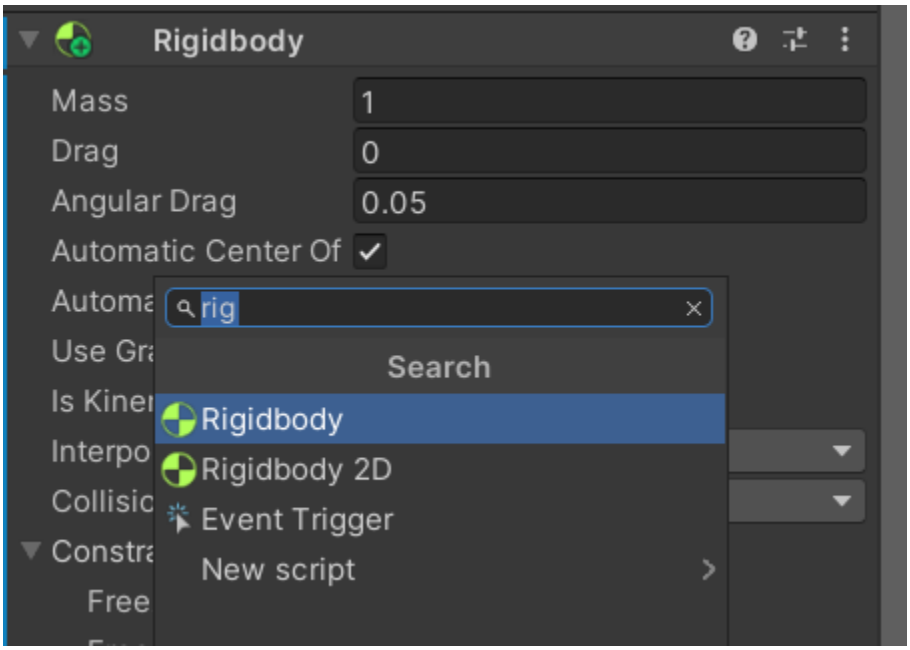
- **Rigidbody İçindeki Ayarlar:**

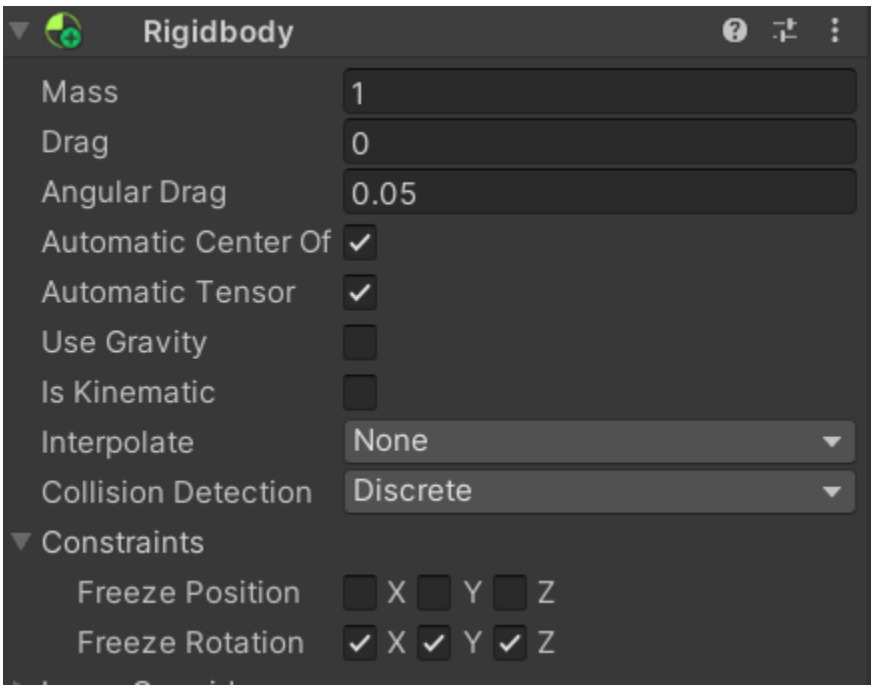
1. **Mass (Kütle):**
    - Nesnenin ağırlığını belirler. Daha büyük kütleli nesneler daha zor hızlanır ve kuvvetlerden daha az etkilenir.
2. **Drag (Sürüklenme):**
    - Nesnenin hızının azalmasını sağlar. Hava direnci veya sürtünmeye benzer. Drag ne kadar yüksekse, nesne o kadar hızlı durur.
3. **Angular Drag (Açısal Sürüklenme):**
    - Nesnenin dönme hızının azalmasını sağlar. Yüksek değerler, nesnenin dönmesinin daha hızlı durmasını sağlar.

4. **Use Gravity (Yerçekimi Kullan):**
    - Bu ayar etkinleştirildiğinde, nesne yerçekimi kuvvetine maruz kalır ve aşağı doğru düşer.
5. **Is Kinematic (Kinematik):**
    - Kinematik hale getirildiğinde nesne, fizik motoru tarafından etkilenmez. Yani çarpışmalar ve kuvvetler ona etki etmez, ancak kod ile kontrol edilebilir.
6. **Interpolate (Enterpolasyon):**
    - Hareketin yumuşaklığını artırır. Nesneler düşük kare hızlarında bile düzgün hareket eder.
        - **None**: Enterpolasyon yok.
        - **Interpolate**: Geçmiş karelerden enterpolasyon yaparak hareketi yumuşatır.
        - **Extrapolate**: Gelecek kareyi tahmin eder ve buna göre hareket eder.
7. **Collision Detection (Çarpışma Algılama):**
    - Nesnelerin çarpışmalarını nasıl algılayacağını ayarlar. Yüksek hızda hareket eden nesnelerin çarpışmaları kaçırmaması için önemlidir.
        - **Discrete**: Varsayılan ayar, sadece çarpışmalar fiziksel olarak temas ettiğinde algılanır.
        - **Continuous**: Yüksek hızlı nesneler için sürekli çarpışma algılaması yapılır. Daha fazla işlem gücü gerektirir.
        - **Continuous Dynamic**: Daha hassas sürekli çarpışma algılaması sağlar, özellikle hızlı hareket eden nesneler için kullanılır.
        - **Continuous Speculative**: Daha az işlemci gücü tüketir, ancak sürekli algılama sunar.
8. **Constraints (Kısıtlamalar):**
    - Nesnenin hareketini veya dönmesini belirli eksenlerde sınırlandırmanı sağlar.
        - **Freeze Position**: X, Y veya Z ekseninde hareketi durdurur.
        - **Freeze Rotation**: X, Y veya Z ekseninde dönmeyi durdurur

Enemy’nin Hareketi/Fiziği

Örneğe geri dönecek olursak enemy prefabine rigidbody bileşeni eklenir. (Add Component : Rigidbody) Bu bileşen lazere hareket kazandırmak için kullanılacaktır.





Enemy yerçekiminden etkilenmemesi için “Use Gravity” kapatılır. Enemy düz bir şekilde kalması için Rotation (x/y/z) değerleri kısıtlanır.

Script-“Enemy”

- **Kullanılan Değişkenler**

```
public class Enemy : MonoBehaviour
{
    [SerializeField]
    private float speed = 1f,speedRandomMultiplier=1f;
    private float xPos;
    private PlayerMovement pm;
```

Private float speed: Düşmanın hız çarpanı.

Private float speedRandomMultiplier: Düşmanların farklı hızlara sahip olması için kullanılan çarpan.

Private float xPos: Düşmanın x eksenindeki pozisyonunu tutacak değişken.

Private PlayerMovement pm: Player içindeki PlayerMovement scrpitinden gerekli verileri çekmek için kullanılacak değişken.

- **Start Fonksiyonu**

```
Unity Message | 0 references
void Start()
{
    pm = FindObjectOfType<PlayerMovement>();
    xPos = Random.Range(-pm.xBorderValue, pm.xBorderValue);
    transform.position = new Vector3(xPos, 6, 0);
    speed = speed+ Random.Range(0, speedRandomMultiplier);

}
```

1.Hierchy içinden PlayerMovement beleşeni bulunur, pm ye atanır.

2.pm içinden çekilen ekran sınırlarına göre rastgele bir x pozisyonu xPos a atanır.

3.Oyun içindeki enemy oyun objesinin pozisyonu uygun bir şekilde ayarlanır.

4.Düşmanın hızına inspectordan ayarlanan 0 ile hız çarpanı kadar bir aralıkta rastgele hız eklenir. Bu şekilde oyun içerisinde düşmanların farklı hızlarda olmasına sebep olunmuştur).

- **Update Fonksiyonu**

```
void Update()
{
    transform.position = transform.position + new Vector3(0, -speed * Time.deltaTime, 0);
    ResetPosition();
}
```

1.Her karede zamana bağlı olarak düşman hız çarpanı kadar aşağı doğru hareket ettirilir.

2.ResetPosition() fonksiyonu çağırılır.

- **ResetPosition() Fonksiyonu**

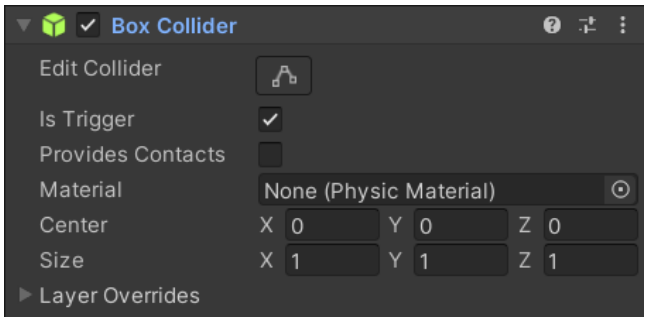
```
1 reference
private void ResetPosition()
{
    if (transform.position.y < -(pm.yBorderValue+1.20f))
    {
        xPos = Random.Range(-pm.xBorderValue, pm.xBorderValue);
        transform.position = new Vector3(xPos, 6, 0);
    }

}
```

Bu fonksiyon ilgili düşmanın transformunun y değerinin belirli bir değerin altına ulaşıp ulaşmadığını kontrol eder. Eğer bu değer belirlenin altına inerse düşman yeni bir x pozisyonuyla ekranın yukarısına gönderilir.

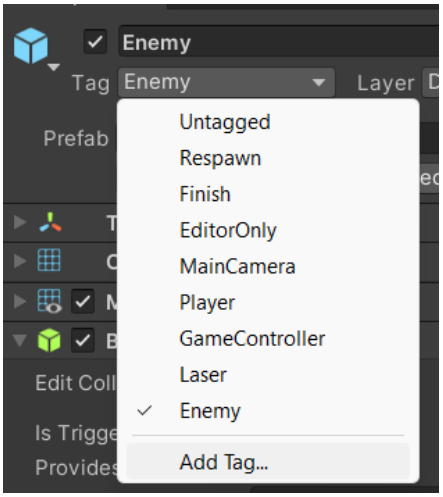
- **OnTriggerEnter() Fonksiyonu**

Gerekli Ayarlamalar

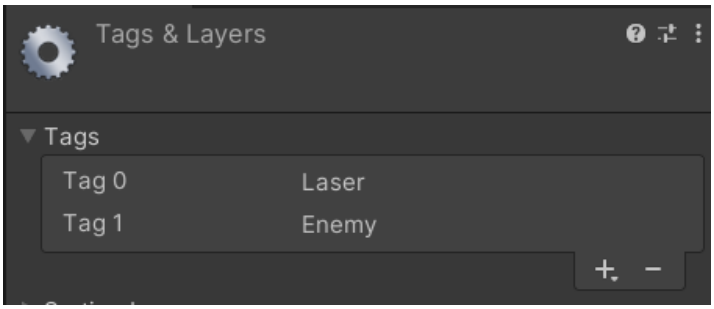
OnTriggerEnter() fonksiyonun çalışması için bir kaça ayarlama yapılması gerekmektedir. Bu ayarlamalar:



1.Düşman oyun nesnesinin Box Collider bileşeninin Is Trigger özelliği aktive edilmelidir. Bu değişiklik Düşman nesnesinin fiziksel olarak yer kaplamamasını sağlayacaktır yani diğer objeler düşman nesnesinin içinden geçecektir.



2.Düşman nesnesinin temasa girdiği objenin hangi nesne olduğunu anlamak için düşmanın ve lazer prefabi için 2 yeni tag oluşturulup, ilgili nesnelerin tagleri değiştirilmiştir.



Kod İçeriği

OnTriggerEnter temasın ilk gerçekleştiği anda çalışacaktır ve bu kodun içinde sırasıyla:

```
Unity Message | 0 references
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Player"))
    {
        pm.LowerLives();
        Destroy(gameObject);
    }
    else if (other.CompareTag("Laser"))
    {
        Destroy(other.gameObject);
        Destroy(gameObject);
    }
}
```

<u>Eğer Oyuncuysa:</u>

1. Oyuncunun PlayerMovement bileşeninden(pm) canını azaltacak fonksiyon(LoverLives() Fonksiyonu) çağırılır.

2. Oyuncuyla temas gerçekleştirilen düşman oyun ortamından yok edilir.

<u>Eğer Lazerse:</u>

1. Etkileşime giren lazer oyun ortamından yok edilir.

2. Lazerle temasa giren düşman oyun ortamından yok edilir.

Script-“Player”

- **Kod Güncellemeler**

1.Diğer bileşenlerden erişimin kolaylaşması için ekran sınır değerlerin private tan public olarak değiştirilmiştir.

```
public float xBorderValue, yBorderValue;
```

2.Oyuncunun canını tutacak bir integer oluşturulmuştur.

```
[SerializeField]
int lives = 3;
```

3.Canı azaltacak LowerLives() fonksiyonu eklenmiştir.

LoverLives() Fonksiyonu

```
public void LowerLives()
{
    lives--;
    if (lives <= 0)
    {
        Destroy(gameObject);
        Debug.Log("No Lives Left GAME OVER!!");
    }

}
```

Bu fonksiyon çağırıldığı zaman oyuncunun canı 1 azaltılır ve 0 değerinin altına düşüp düşmediği kontrol edilir. Eğer oyunucunun canı 0’ın altına düşerse oyuncu oyun nesnesi oyun evreninden yok edilir ve konsola “No Lives Left GAME OVER!!” yazdırılır.

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/3.Hafta

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

Hazırlayan

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