**1. Düşman Prefab Oluşturma**

Düşman karakterini oyun sahnesine eklemek için bir prefab (önceden hazırlanmış nesne) oluşturacağız.

**Adımlar:**

1. **Yeni bir 3D nesne ekleyin**:
    - Unity'de **Hierarchy** penceresine sağ tıklayın ve 3D Object > Cube seçeneğini seçin (veya başka bir model ekleyebilirsiniz).
    - Bu nesne, düşmanımız olacak.
2. **Düşman Nesnesini Hazırlama**:
    - **Inspector** penceresinde, nesneye bir isim verin (örneğin, "Enemy").
    - Nesnenin pozisyon, ölçek ve diğer özelliklerini ayarlayın.
3. **Prefab Oluşturma**:
    - **Project** penceresinde boş bir klasör oluşturun (örneğin, "Prefabs").
    - Düşman nesnesini **Hierarchy** penceresinden sürükleyerek bu klasöre bırakın. Böylece bir **Prefab** oluşturmuş olursunuz.

**2. Düşman Hareketini Sağlayan Kodun Yazılması**

Düşmanın sürekli veya belirli bir düzende hareket etmesini sağlamak için basit bir hareket scripti yazacağız.

**Adımlar:**

1. **Hareketi Oluşturma**:
    - Oluşturduğumuz **EnemySC** script dosyasında **Movement** isimli fonksiyon oluşturuyoruz.

```
0 references
void Movement(){
    transform.Translate(Vector3.down * speed * Time.deltaTime);
    if (transform.position.y <= -11)
    {
        Respawn();
    }
```

```
void Respawn()
{
    transform.position = new Vector3(Random.Range(-10, 10), 7, 0);
}
```

    - Böylelikle düşmanın hareket ve ekran sınırları dışında yeniden belirme fonksiyonu oluşmuş oluyor.

**3. Rigidbody ve Collider Bileşenleri Eklenmesi**

Düşmanın fiziksel çarpışmalarını ve hareketini yönetmek için Rigidbody ve Collider bileşenleri eklemeliyiz.

**Adımlar:**

1. **Rigidbody Ekleme**:
    - **Hierarchy** penceresinde düşman nesnesini seçin.
    - **Inspector** penceresinde Add Component butonuna tıklayın ve Rigidbody bileşenini ekleyin.
    - Use Gravity seçeneğini kapatın, çünkü düşman sürekli hareket edecek ve yer çekimi nedeniyle düşmesini istemiyoruz.
2. **Collider Ekleme**:
    - Eğer düşman nesnenizde zaten bir Collider yoksa Add Component menüsünden uygun bir Collider ekleyin.
    - Is Trigger seçeneğini aktif hale getirin, çünkü çarpışmaları tespit edip tetikleme işlemleri gerçekleştireceğiz.

**4. OnTriggerEnter Fonksiyonu ile Çarpışma Tespiti**

Düşman ile oyuncunun çarpışmasını tespit etmek ve çarpışma anında gerekli aksiyonları almak için OnTriggerEnter fonksiyonunu kullanacağız.

**Adımlar:**

1. **Çarpışmayı Yönetme Script'i**:
    - Düşman nesnesi için yeni bir fonksiyon yazacağız (**OnTriggerEvent**).
    - Aşağıdaki kodu ekleyin:

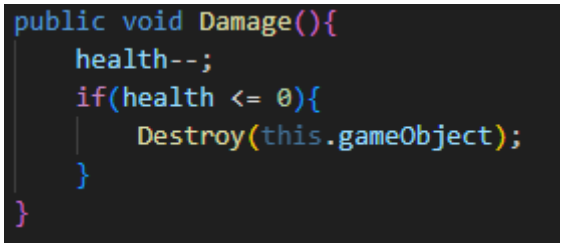
```
void OnTriggerEnter(Collider other){
    Debug.Log(other.transform.name+" Çarpıştı");
    if(other.tag == "Player"){
        PlayerSC player = other.transform.GetComponent<PlayerSC>();
        player.Damage();
        Debug.Log("Player Health"+ player.health);
        Destroy(this.gameObject);

    }
    if(other.tag == "Bullet"){
        Destroy(other.gameObject);
        Destroy(this.gameObject);
    }
}
```

- Bu kod player ve düşmana hasar veren kodu çalıştırmaktadır. Player'dan can value'sunu alarak can değerini eksiltir. Eğer **Enemy, Bullet** Nesnesi ile çarpıştıysa ikisi de yok olur.

5. **Oyuncu Hasar Alma İşlemleri**

Burası **PlayerSC** isimli scripte aşağıdaki kod ile sağlanmaktadır.



```
public void Damage(){
    health--;
    if(health <= 0){
        Destroy(this.gameObject);
    }
}
```

Son olarak bu yaptığımız değişikleri **EnemyPrefab**'a uygulamayı unutmuyoruz.