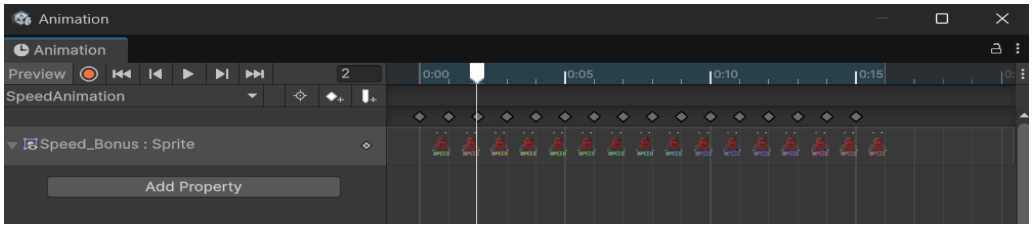
Oyun Programlama Ödev 6

1) Speed_Bonus Prefabı Oluşturma

Ödevde, Triple_Shot_Bonus prefabına benzer şekilde, oyuncu ile etkileşime girdiğinde aktifleşen farklı bonusların da prefab olarak hazırlanması gerekiyor. Bu bonuslardan biri de hız artırma bonusudur.

Hız bonusu prefabını oluşturmak için önce Project penceresindeki Sprites klasörü altında Power_Ups klasörüne girilir. Buradaki ilk sprite, Hierarchy ekranına sürüklenir ve adı Speed_Bonus olarak değiştirilir. Ardından Collider ve Rigidbody 2D componentleri eklenir. IsTrigger ekinleştirilir. Prefabın boyutları ayarlanır. Düzenlemeler tamamlandıktan sonra bu nesne, Prefabs klasörüne sürüklenerek prefab haline getirilir.

Bonusun oyunda animasyonlu görünmesi için tekrar Power_Ups klasörüne girilir. Speed bölümündeki ilk sprite'a tıklanır, sonra Shift'e basılı tutarak son sprite seçilir ve böylece aradaki tüm sprite'lar işaretlenir. Son olarak Hierarchy'deki henüz sahneye eklenmiş Speed_Bonus nesnesine tıklanır ve seçilen sprite'lar Animation penceresine eklenir.

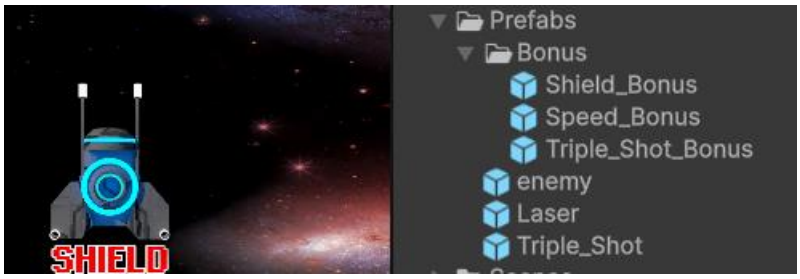


Animasyon oluşturulduktan sonra Speed_Bonus prefabı tamamen hazır hale gelir. Gerekli prefab atmaları SpawnMaanger'a ait Inspector penceresinde yapılır

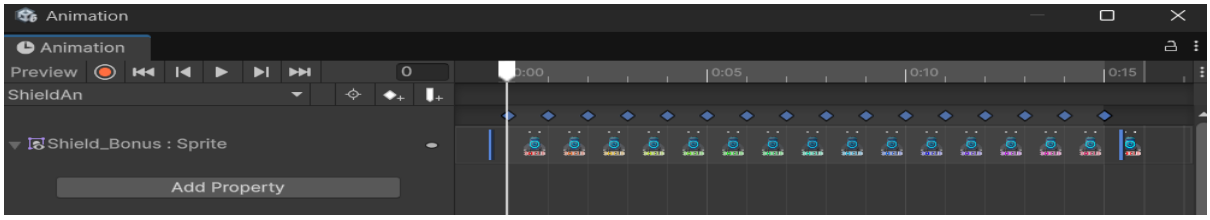
2) Shield_Bonus Prefabı Oluşturma

Speed_Bonus ve Triple_Shot_Bonus prefablarıında olduğu gibi, oyuncu ile temas ettiğinde etkinleşen farklı bonuslardan biri de kalkan bonusudur.

Speed_Bonus prefabında yaptığımız gibi Kalkan bonusu için de önce Project penceresindeki Sprites klasörü altında Power_Ups klasörüne girilir. Buradaki ilk sprite, Hierarchy bölümüne sürüklenir ve adı Shield_Bonus olarak değiştirilir. Ardından nesnenin boyutu ve component ayarları yapılır. Tüm düzenlemeler tamamlandığında, bu nesne Prefabs klasörüne sürüklenerek prefab haline getirilir.



Kalkan bonusunun da Speed ve Üçlü atış bonusu gibi oyunda animasyonlu görünmesi için bu bonuslarda yaptıkalrımız gibi tekrar Power_Ups klasörüne gidilir. Shield kısmındaki ilk sprite seçilir, ardından Shift tuşuna basılı tutularak son sprite'a tıklanır ve aradaki tüm kareler işaretlenir. Daha sonra Hierarchy'deki sahneye eklenmiş Shield_Bonus nesnesi seçilir ve işaretlenen sprite'lar Animation penceresine sürüklenir.



Animasyon oluşturulduktan sonra Shield_Bonus prefabı kullanıma hazır olur. Gerekli prefab atmaları SpawnMaanger'a ait Inspector penceresinde yapılır

3) Bonus_Id

Bonusların oluşturulmasını kontrol edebilmek için bonusId adında bir değişken tanımlanır. Bu değişkenin aldığı değere göre ilgili case çalışır ve kod akışı devam eder. Bonus_sc dosyasında yer alan switch–case yapısı sayesinde hangi fonksiyonun (TripleShotActive(), SpeedBonusActive(), ShieldBonusActive()) çalışacağı belirlenmiş olur.

```
void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Player")
        {

            Player_sc player_sc = other.transform.GetComponent<Player_sc>();
            if (player_sc != null)
            {
                switch (bonusId)
                {
                    case 0:
                    player_sc.TripleShotActive();
                    break;

                    case 1:
                    Debug.Log("Hız bonusu");
                    break;

                    case 2:
                    Debug.Log("Kalkan");
                    break;

                    default:
                    Debug.Log("Hata Durumu!");
                    break;
                }

            }

            Destroy(this.gameObject);
        }
    }
```

4) SpawnBonusRoutine Güncelleme

Eklenen bonusların darastgele bir şekilde ortaya çıkması için Spawnmanager_sc içinde bazı değişiklikler yapmamız gerekir.

Bunun için daha önce tanımladığımız

```
[SerializeField]
 private GameObject tripleShotBonusPrefab;
```

kodunu da değiştirmemiz gerekir. Yerinde bonusları tutmak amacıyla bonusPrefab[] adında bir dizi oluştururuz.

```
[SerializeField]
 GameObject[] bonusPrefabs;
```

Daha sonra SpawnBonusRoutine içinde bu diziye erişebilmek için randomBonus isminde bir değişken tanımlanır. Bu değişkenin alabileceği değer aralığı 0 ile 2 arasında olacak şekilde ayarlanır. Bunun nedeni, dizide toplam üç bonus bulunması ve bu bonusların dizideki indexlerine rastgele ulaşılmasını sağlamaktır.

randomBonus değeri her üretildiğinde, dizinin rastgele bir index'i seçilmiş olur ve böylece oyunda da bonuslar rastgele şekilde spawnlanır

```
47      IEnumerator SpawnBonusRoutine()
48      {
49          while (stopSpawning == false)
50          {
51
52              int waitTime = Random.Range(5, 10);
53              Debug.Log("Üçlü atış bekleme süresi:" + waitTime);
54              yield return new WaitForSeconds(waitTime);
55
56              Vector3 position = new Vector3(Random.Range(-9.18f, 9.18f),
57                                                                  7.4f, 0);
58
59              int randomBonus=Random.Range(0,2);
60              GameObject tripleShotBonus = Instantiate(bonusPrefab[randomBonus], position, Quaternion.identity);
61
62          }
63
64      }
```

Kodu bu şekilde güncelleriz.

5) isSpeedBonusActive ve isShieldBonusActive

Player_sc dosyasında oluşturduğumuz isSpeedBonusActive ve isShieldBonusActive değişkenleri, bu bonusların o anda aktif olup olmadığını kontrol etmek için kullanılır. Bu değişkenler, ilgili bonusların çalıştırıldığı fonksiyonlarda devreye girer.

```
[SerializeField]
private bool isTripleShotActive = false;

[SerializeField]
private bool isSpeedBonusActive=false;

[SerializeField]
private bool isShieldBonusActive=false;
```

Oyun başladığında bu değişkenlerin değeri false olarak ayarlanır, çünkü başlangıçta oyuncunun üzerinde herhangi bir bonus etkisi bulunmaz. Oyuncu oyun sırasında bir bonusla temas ettiğinde, o bonusa ait fonksiyonlar (TripleShotActive(), SpeedBonusActive(), ShieldBonusActive()) çalıştırılır ve ilgili değişken true yapılarak bonusun aktif hâle gelmesi sağlanır.

```
public void SpeedBonusActive()
  {
    isSpeedBonusActive= true;
    hareketHizi *= speedMultiplier;
    StartCoroutine(SpeedBonusCancelRoutine());
  }
IEnumerator SpeedBonusCancelRoutine()
  {
    yield return new WaitForSeconds(5.0f);
    isSpeedBonusActive = false;
    hareketHizi /= speedMultiplier;
  }
```

```
if (isShieldBonusActive)
    {
      isShieldBonusActive = false;
      return;
    }
```

Bu kod bloğu ile de çarpışma olana kadar kalkan bonusumuz aktif kalır.

6) Scriptlerdeki Fonkisyon Güncellemeleri

Oyuna eklenen yeni bonusların çalışması için en önemli kısım, bu bonusları etkinleştiren fonksiyonlardır. Bu fonksiyonlar, isSpeedBonusActive ve isShieldBonusActive değişkenleri üzerinden kontrol edilir. Üçlü atış bonusunda olduğu gibi, hız bonusu için de ayrı bir Coroutine tanımlanır. Bu coroutine içinde oyuncu karakterinin hızını artırmak için Player nesnesine ait speed değişkeni geçici olarak değiştirilir.

Bunu yapabilmek için bir hız çarpanı belirlenir. SpeedBonusActive() fonksiyonunda oyuncunun hızı bu çarpanla çarpılır ve coroutine çalıştığı sürece hız artmış şekilde kalır. Bonus süresi bittiğinde, artırılmış hız aynı çarpana bölünerek eski hâline geri getirilir. Böylece oyuncu hızı başlangıç değerine döner.

Kalkan bonusu için ShieldBonusActive fonksiyonu içinde isShieldBonusActive true yapılır. Bunun yanında, kalkan efektinin görünmesi için shieldVisualizer adıyla bir değişken tanımlanır ve SetActive(true) ile görünür hâle getirilir.

```
[SerializeField]
  GameObject shieldVisualizer;
```

Kalkan aktifken oyuncunun zarar almaması için Damage() fonksiyonunun güncellenmesi gerekir. Bu noktada tekrar isShieldBonusActive değişkeni kullanılır. Oyuncu hasar almadan önce kalkanın aktif olup olmadığı kontrol edilir. Eğer lazer oyuncuya çarptığında kalkan açıksa, kalkan kapanır ve isShieldBonusActive false yapılarak devre dışı bırakılır.

Script Dosyalarına Github Hesabımdan Ulaşabilirsiniz:

https://github.com/AsmBrk/Oyun-Programlama-Lab/tree/main/%C3%96dev%206