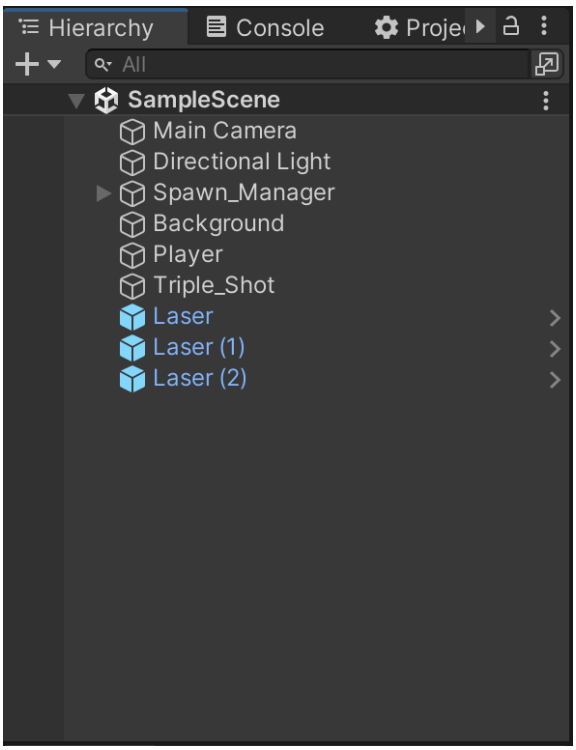
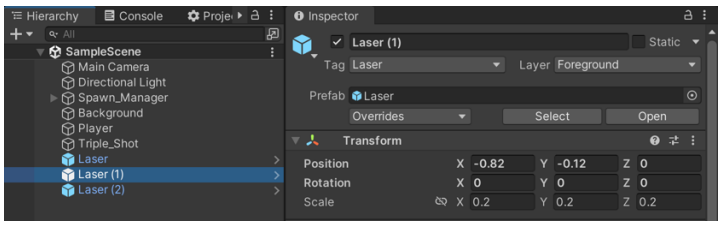
Oyun Programlama Ödevi – Hafta 6

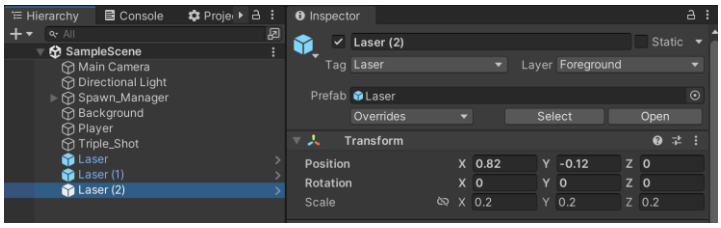
Ad: Eren
Soyad: Köse
Numara: 22360859075
GitHub Kod Linki: https://github.com/erennkose/btu-oyun-programlama/tree/main/Hafta6

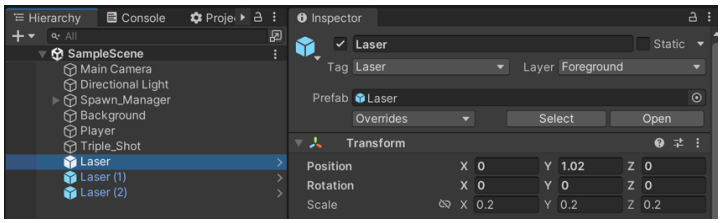
- ## Üçlü Atış Bonusu: Laser'den İki Kopya Daha Oluşturma ve Bu Üç Laser'in Konumlarını Ayarlama

Laser prefabimizi üç kere sürükle-bırak yöntemiyle **Hierarchy** kısmına bırakıyoruz. Bu işlemden sonra ise Laserlerimizin konumlarını tek tek **Inspector** kısmındaki **Transform** alında bulunan **Position** değerleriyle düzenliyoruz.



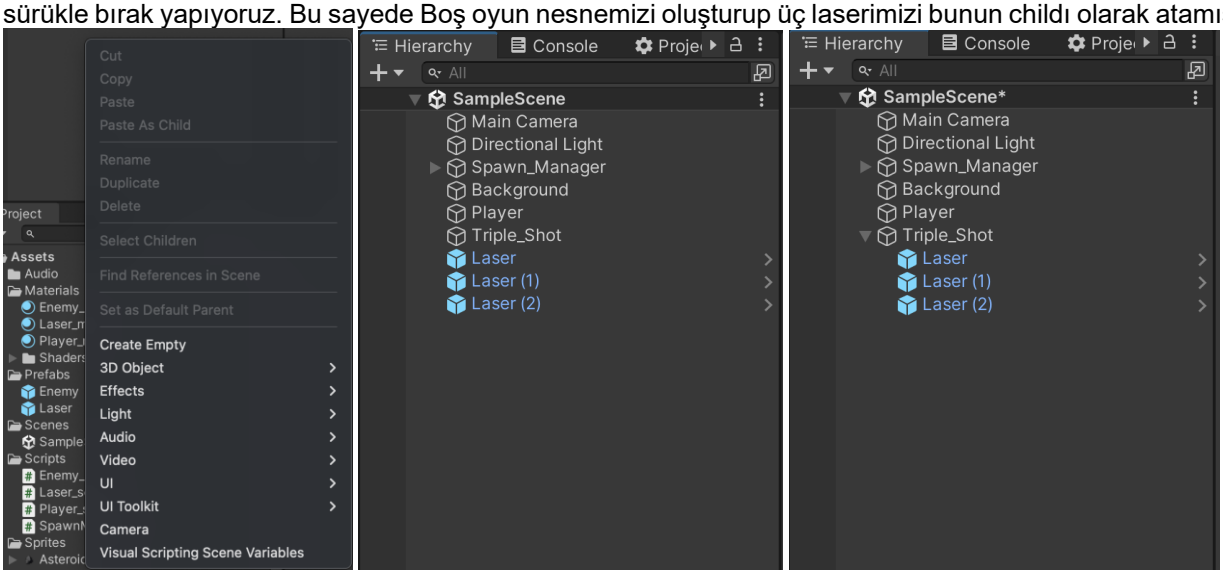






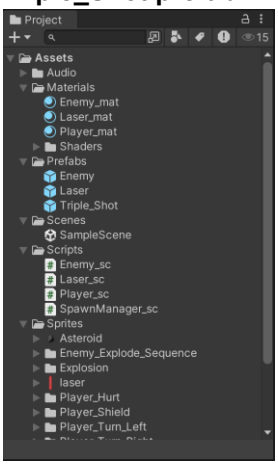
- ## Boş Bir Oyun Nesnesi Oluşturun ve Üç Laser'i Bunun Child'ı Olarak Atayın

Öncelikle **Hierarchy** kısmında **Create Empty** diyerek boş bir nesne oluşturuyoruz ve **Triple_Shot** olarak isimlendiriyoruz. Ardından bir önceki maddede oluşturduğumuz üç Laser nesnesini **Triple_Shot** nesnemizin üstüne sürükle bırak yapıyoruz. Bu sayede Boş oyun nesnemizi oluşturup üç laserimizi bunun childı olarak atamış oluyoruz.



- ## Bu Oyun Nesnesinden Bir Prefab Oluşturun

İçerisine Laser nesnelerimizi eklemiş olduğumuz **Triple_Shot** nesnemizi sürükle-bırak yöntemiyle **Prefabs** klasörümüz içine bırakıyoruz. Ardından bu nesnemizi **Hierarchy** kısmından silebiliriz. Bu işlemler sonucunda **Triple_Shot prefabimiz** oluşmuş oluyor.



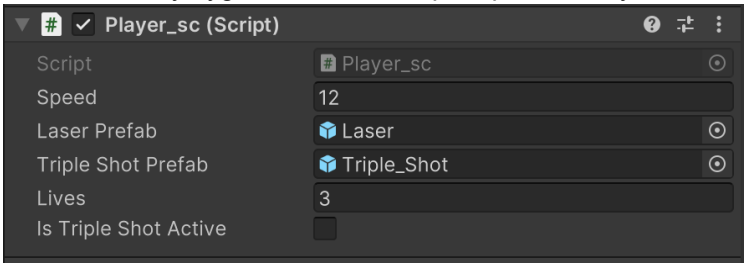
- ## Üçlü bonusun hangi durumlarda aktive edileceğinin mantığını geliştirin ve kodlayın

Üçlü bonusu aktive etmek için öncelikle bir **bool** değişken oluşturmalıyız. Bu değişken sayesinde gerektiğinde üçlü bonusu aktifleştirip gerektiğinde deavktive edebiliriz. Öncelikle **Player_sc** scriptimiz içerisinde **isTripleShotActive** adında bir **bool** değişken oluşturup default olarak **false** değeri atıyoruz. Sonrasında ise **public GameObject** tipinde **TripleShotPrefab** adında bir değişken daha oluşturuyoruz.

```
public bool isTripleShotActive = false;
```

```
public GameObject tripleShotPrefab;
```

Ardından Unity uygulamamıza dönüp bu prefabi Playerımızın **Inspector** kısmından seçiyoruz.



Oluşturduğumuz değişkenlerden **isTripleShotActive** değişkeninin **true** olması durumunda bizim üçlü bonusumuzu aktive etmemiz ve Player nesnemizi üçlü laser ateş etmesini sağlamamız gerekmektedir. Bunun için ise kodumuza bazı fonksiyonlar eklememiz gerekmektedir. Bu fonksiyonlar sayesinde aşağıda oluşturacak olduğumuz üçlü bonus oyun nesnemizle temas halinde üçlü bonusu aktifleştireceğiz. Bunu da bir sonraki maddelerde gerçekleştireceğiz.

- ## Üçlü bonus sprite'ı toplandığında belirli bir süre için üçlü bonus'u etkinleştirin

Player_sc scriptimizde fonksiyonların dışında, classımızın içinde **ActivateTripleShot** adında bir fonksiyon oluşturuyoruz. Bu fonksiyon bizim **isTripleShotActive** değişkenimizi **true** yapmamızı sağlayacak. Oyuncuya sınırsız üçlü bonus vermek istemeyeceğimiz için belli bir süre sonrasında **isTripleShotActive** değişkenimizi tekrar **false** yapmamız gerekmekte. Bunun için ise **Coroutine** kullanacağız. Yine fonksiyonlarımızın dışına, classımızın içine bir fonksiyon tanımlıyoruz. **IEnumerator** tipinde **TripleShotBonusDisableRoutine** adında bir fonksiyon oluyor. Bu fonksiyon içinde ise 5 saniye sonra **isTripleShotActive** değişkenimizi **false** yapmasını sağlıyoruz. Bu sayede oyuncu aldığı bonusu sınırsız kullanmamış oluyor. En son oluşturduğumuz **TripleShotBonusDisableRoutine** coroutineinini **ActivateTripleShot** fonksiyonumuzda çağırıyoruz. Bu sayede de coroutine aktifleşmiş oluyor.

```
public void ActivateTripleShot(){
    isTripleShotActive = true;
    StartCoroutine(TripleShotBonusDisableRoutine());
}
1 başvuru
IEnumerator TripleShotBonusDisableRoutine(){
    yield return new WaitForSeconds(5.0f);
    isTripleShotActive = false;
}
```

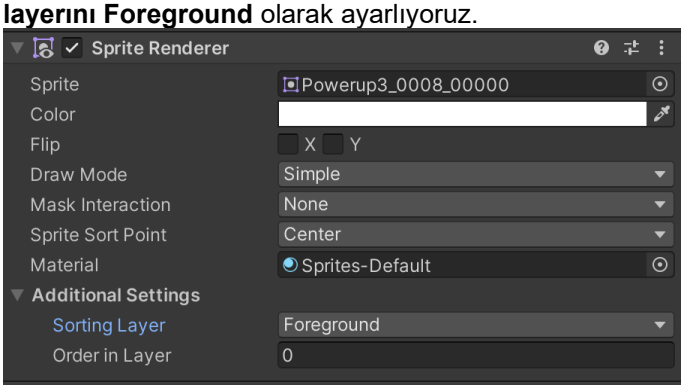
- ## FireLaser fonksiyonunda gerekli güncellemeleri yapın

FireLaser fonksiyonumuz içerisinde space butonuna basılma ve zaman kontrolü yapılan if-else'in içerisine başka bir if-else açıyoruz. **isTripleShotActive** değişkenimiz **false** ise normal laser, **true** ise triple laser oluşturulmasını sağlıyoruz. Üçlü laserimizin oluşacağı konumu test ederek kodda ona göre giriyoruz. Bu sayede üçlü bonusumuzu da kontrol ederek **FireLaser** fonksiyonumuzu düzenlemiş oluyoruz.

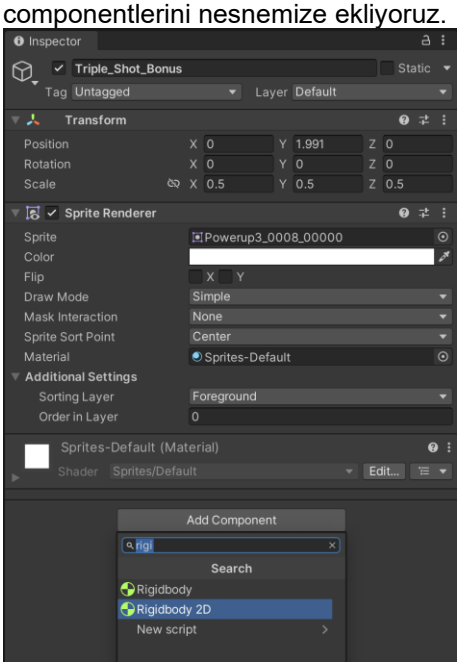
```
void FireLaser(){
    if ((Input.GetKeyDown(KeyCode.Space)) && (Time.time > nextFire)){
        if(!isTripleShotActive)
        {
            Instantiate(laserPrefab, transform.position + new Vector3(0, 1.05f, 0), Quaternion.identity);
            nextFire = Time.time + fireRate;
        }
        else if(isTripleShotActive)
        {
            Instantiate(tripleShotPrefab, transform.position + new Vector3(0, 0.5f, 0), Quaternion.identity);
            nextFire = Time.time + fireRate;
        }
    }
}
```

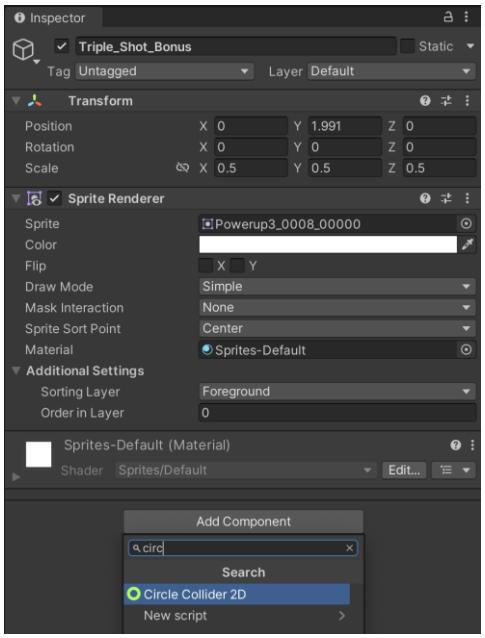
- ## Üçlü bonus sprite'ından bir nesne oluşturun, ölçeğini ayarlayın, circle collider ve rigidbody ekleyin, sorting layer'ı düzenleyin

Üçlü bonusu aktive etmek için öncelikle Üçlü bonusumuzu belirten bir nesnemiz olmalı. Bunun için **Sprites** klasörümüzün altındaki **Power_Ups** klasöründe bulunan **Triple_Shot** klasörünün ilk görselini sürükle-bırak yöntemiyle **Hierarchy** kısmına bırakıyoruz ve **Inspector** bölümündeki **Transform'un** altında bulunan **Scale** değişkenleriyle nesnemizin boyutlarını isteğe göre düzenliyoruz. Nesnemize de **Triple_Shot_Bonus** adını veriyoruz. İsimlendirmeden sonra ise **Inspector** kısmındaki **Sprite Renderer** kısmından nesnemizin **sorting layerını Foreground** olarak ayarlıyoruz.

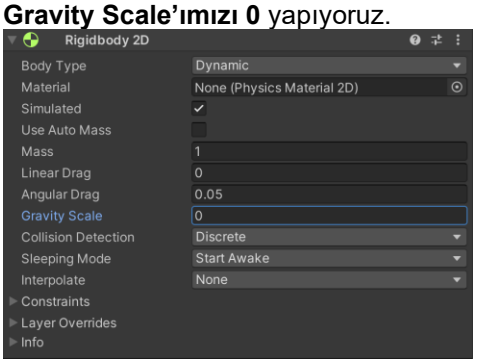


Player nesnemiz ve yeni oluşturduğumuz bu Triple_Shot_Bonus nesnemiz temas etmesi durumunda Playerımızın üçlü laser ateş etmesini istiyoruz. Bu temas durumunu algılayabilmemiz için nesnemize **collider** ve **rigidbody** eklememiz gerekiyor. Nesnemiz 2D olacağından dolayı colliderımızı ve rigidbodymizi de 2D eklemeliyiz. Bu componentleri eklemek için öncelikle **Triple_Shot_Bonus** nesnemize tıklıyoruz. Ardından Inspector kısmının en altında bulunan Add Component butonuna tıklıyoruz ve sırasıyla **Rigidbody 2D** ve **Circle Collider 2D** componentlerini nesnemize ekliyoruz.

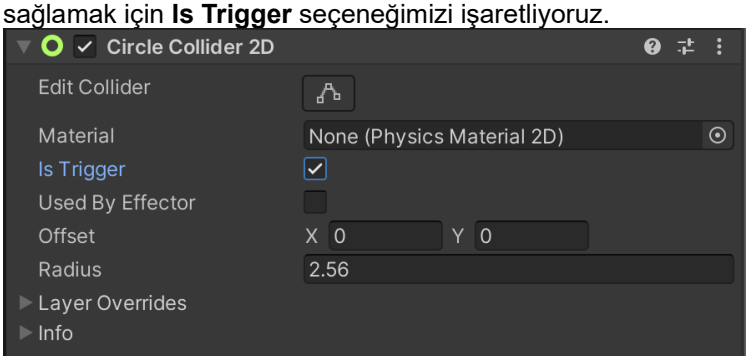




Rigidbody 2D adlı componentimizi ekledikten sonra bu nesnemize yerçekimi uygulamak istemediğimizden dolayı **Gravity Scale'ımızı 0** yapıyoruz.



Circle Collider 2D adlı componentimizi ekledikten sonra ise nesnemizin Player ile olan temasını algılamamızı sağlamak için **Is Trigger** seçeneğimizi işaretliyoruz.



- **<u>Script oluşturarak üçlü bonus nesnesi ile ilişkilendirin, script ile nesnenin belirli bir hızla hareket etmesini sağlayın. Ekran dışına çıkıldığında nesneyi yok edin. Çarpışma kontrolü ekleyin.</u>**

Şimdi ise üçlü bonusumuz için oluşturduğumuz **Triple_Shot_Bonus** nesnemiz için bir script oluşturacağız. Bunun için **Scripts** klasörümüze gidiyoruz ve **Bonus_sc** adında bir script oluşturuyoruz. Gelecekte oluşturacağımız diğer bonuslar için de gerekecek kodları buraya ekleyeceğiz. Oluşturduktan sonra bu scripti sürükle-bırak ile **Triple_Shot_Bonus** nesnemizin üzerine bırakıyoruz. Ardından scripti kod editörümüzde açıyoruz. Bu nesnemizin hareketini kontrol etmek için Update fonksiyonu içerisinde bir **if-else** kontrolü kullanıyoruz. Nesnemizin sahneden çıkması durumunda yok edilmesini sağlamak amacıyla eğer sınırı geçersek nesneyi yok et kodunu yazıyoruz.

```
if(transform.position.y < -5.38){
    Destroy(this.GameObject());
}
```

Nesnemizin sahnede durduğu durumlarda ise nesnemizin bir hızla y ekseninde aşağı doğru düşüşünü sağlamak istiyoruz. Bu durumu sağlamak için fonksiyonların dışında, classımızın içinde **int speed** değişkeni tanımlıyoruz ve bu değişkene bir değer atıyoruz. Ardından ise else koşulumuza geri dönüp y ekseninde -3 yönünü gösteren bir Vector3 vektörü tanımlıyoruz. Tanımlamamızın ardından ise **transform.Translate** fonksiyonuna parametre olarak bu **vektörümüzle speed** değişkenimizin ve **Time.deltaTime** değerlerinin çarpımını veriyoruz. Bu sayede nesnemizin hareketini de sağlamış oluyoruz.

```
else{
    Vector3 direction = new Vector3(0,-1,0);
    transform.Translate(direction * Time.deltaTime * speed);
}
```

Update fonksiyonumuzun içi şu şekilde görünmelidir;

```
void Update()
{
    if(transform.position.y < -5.38){
        Destroy(this.GameObject());
    }
    else{
        Vector3 direction = new Vector3(0,-1,0);
        transform.Translate(direction * Time.deltaTime * speed);
    }
}
```

Bu bonusumuzun Player nesnemizle çarpışma durumunu da burada kontrol etmemiz gerekiyor. Bunun için ise fonksiyonlarımızın dışında, classımızın içinde **OnTriggerEnter2D** adındaki fonksiyonu tanımlıyoruz ve parametre olarak **Collider2D other** girilmesini istiyoruz. Fonksiyonun içinde ise if koşulu ile temas edilen Collider2D'nin tagının Player olup olmadığının kontrolünü yapıyoruz. Eğer temas edilen Collider2D Player ise Player_sc scriptimizdeki **ActivateTripleShot** fonksiyonumuzu çağırmamız gerekmekte. Bunu da bir sonraki maddede gerçekleştireceğiz.

- ## Player kodu içinden üçlü bonus koduna erişin ve üçlü bonus'u etkinleştirin

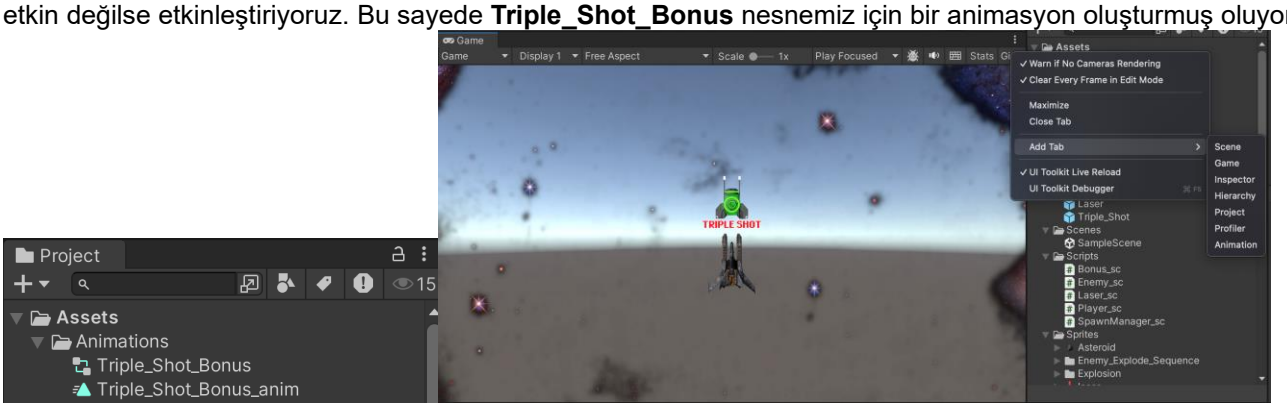
Bonus_sc scriptimizden Player_scdeki **ActivateTripleShot** fonksiyonuna erişmek için if koşulumuz içinde bir **Player_sc nesnesi** oluşturuyoruz ve bu nesneye **other.transform.GetComponents<Player_sc>()** diyerek Bonus_sc içerisinde Player_scde bulunan fonksiyonların kullanımına olanak sağlıyoruz. Ardından küçük bir if koşulu ile bu Player_sc nesnesini doğru şekilde atayıp atayamadığımızı kontrol ediyoruz. Bir sorun yoksa **ActivateTripleShots()** fonksiyonumuzu nesnemiz üzerinden çağırıyoruz. Bu sayede üçlü bonusu etkinleştirmiş oluyoruz. Ardından ise bu bonus nesnemizi optimizasyon amacıyla yok ediyoruz.

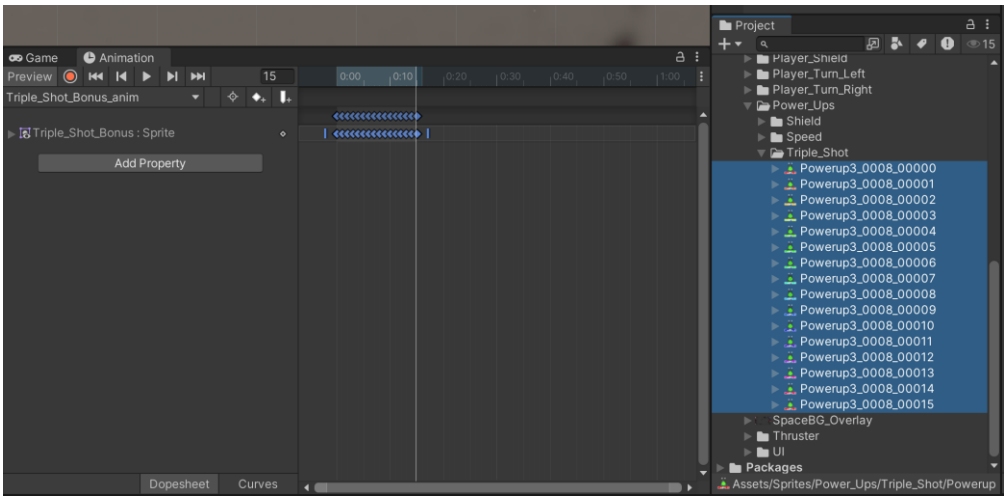
```
void OnTriggerEnter2D(Collider2D other){

    if(other.tag == "Player"){
        Player_sc playersc = other.transform.GetComponent<Player_sc>();
        if(playersc != null){
            playersc.ActivateTripleShot();
        }
        Destroy(this.gameObject);
    }
}
```

- ## Üçlü atış bonusu nesnesi için animasyon ekleyin

Öncelikle **Project>Assets** klasörü altında **Animations** adında bir klasör oluşturuyoruz. Sonrasında ise klasörümüze sağ tık atıp **Create>Animation** seçeneklerini izleyerekve **Triple_Shot_Bonus_anim** isimlendirmesini yaparak bir animasyon oluşturuyoruz. Oluşturma işlemi bittikten sonra ise sürükle-bırak ile Triple_Shot_Bonus nesnemizin üzerine bırakıyoruz. Ardından **Triple_Shot_Bonus** nesnemize tıklanıkken Game ekranımızın açık olduğu kısmın sağ üstünde bulunan üç noktaya tıklayıp **Add Tab>Animation** seçeneklerini izleyerek Animation sekmesini açıyoruz. Ardından Sprites>Power Ups klasörü altında PowerUp3 ile başlayan tüm görselleri shift ile tutup **Animation** tabına bırakıyoruz. Ardından Triple_Shot_Bonus_anim animasyonumuza tıklayıp Loop Time seçeneği etkin değilse etkinleştiriyoruz. Bu sayede **Triple_Shot_Bonus** nesnemiz için bir animasyon oluşturmuş oluyoruz.





- **SpawnManager'ı üçlü atış bonusunu da dikkate alacak şekilde düzenleyin**

Normalde **SpawnManager_sc** scriptimizde sadece Enemy spawnlamak amacıyla bir **SpawnRoutine** coroutine fonksiyonu bulunmaktaydı. Fakat artık burada bonus spawnlama işlemi de yapacağımızdan dolayı bazı değişkilikler yapmamız gerekecek. Öncelikle SpawnRoutine adlı coroutine fonksiyonumuz enemy spawnladığı için adını SpawnEnemyRoutine yapıyoruz. Start fonksiyonu içerisinde de çağırdığımız için orada da **SpawnRoutine** yazan yeri **SpawnEnemyRoutine** yapıyoruz.

```
void Start()
{
    StartCoroutine(SpawnEnemyRoutine());
}
```

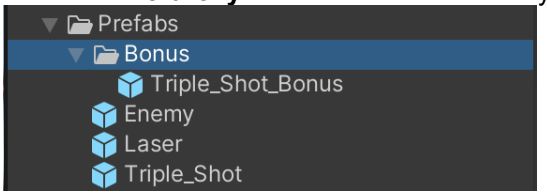
```
IEnumerator SpawnEnemyRoutine()
{
    while(stopSpawning == false)
    {
        Vector3 position = new Vector3(Random.Range(9.4f,-9.4f), 7.4f, 0);
        GameObject new_enemy = Instantiate(enemyPrefab, position, Quaternion.identity);
        new_enemy.transform.parent = enemyContainer.transform;
        yield return new WaitForSeconds(5.0f);
    }
}
```

Şimdi ise enemy spawnlama mantığımıza benzer şekilde bonus nesnemizin spawnlanmasını da sağlayacağız. Öncelikle classımızın içinde, fonksiyonlarımızın dışında stopSpawning adında bir bool değişken oluşturup false default değerini atıyoruz. Ardından **SpawnEnemyRoutine** fonksiyonumuzda yaptığımız gibi **SpawnBonusRoutine** adında bir coroutine fonksiyonu tanımlıyoruz. İçerisine ise SpawnEnemyRoutine kodunun aynısını, **stopSpawning** değişkeninin **false** olduğu sürece çalışmasını sağlayan bir **while** döngüsü içine atıyoruz. Ardından SpawnEnemyRoutine içinden aldığımız kodda enemyPrefab olan kısmı **tripleShotBonusPrefab** yapıyoruz, süreyi ise 5.0f'den **3.0f**'e alıyoruz. Bu sayede bonus nesnemizin spawnlanması durumunu da ayarlamış oluyoruz. SpawnBonusRoutine coroutine fonksiyonumuzu da yazdığımıza göre şimdi bu fonksiyonu da Start fonksiyonumuzdan çağıracağız. Bu bir coroutine fonksiyon olduğundan dolayı SpawnEnemyRoutine ile aynı şekilde StartCoroutine içerisinde çağırıyoruz. Bu sayede SpawnManager_sc scriptimiz de düzenlenmiş oluyor.

```
void Start()
{
    StartCoroutine(SpawnEnemyRoutine());
    StartCoroutine(SpawnBonusRoutine());
}
```

```
IEnumerator SpawnBonusRoutine(){
    while(stopSpawning == false)
    {
        Vector3 position = new Vector3(Random.Range(9.4f,-9.4f), 7.4f, 0);
        Instantiate(tripleShotBonusPrefab, position, Quaternion.identity);
        yield return new WaitForSeconds(3.0f);
    }
}
```

Script düzenlendikten sonra ise Unity uygulamamıza geri dönüyoruz ve **Triple_Shot_Bonus** nesnemizi **prefab** haline getirmemiz gerekiyor. Gelecekte gelecek olan bonuslarımızı da düşündüğümüzden dolayı **Prefabs** klasörümüzün altında **Bonus** adında bir klasör açıp nesnemizi de sürükle-bırak ile bu klasörün içine atıyoruz. Ardından **Hierarchy** kısmından nesnemizi siliyoruz.



Prefabimizi oluşturduktan **Spawn_Manager** nesnemize tıklıyoruz. **Inspector** kısmında görünen scriptten **Triple Shot Bonus Prefabimizi** seçiyoruz. Bu sayede bonusumuzun oyunda spawnlanmasını sağlamış oluyoruz.