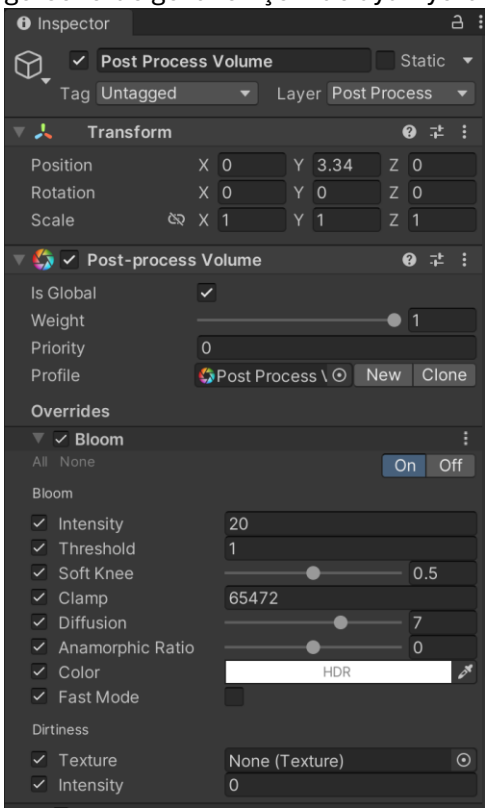
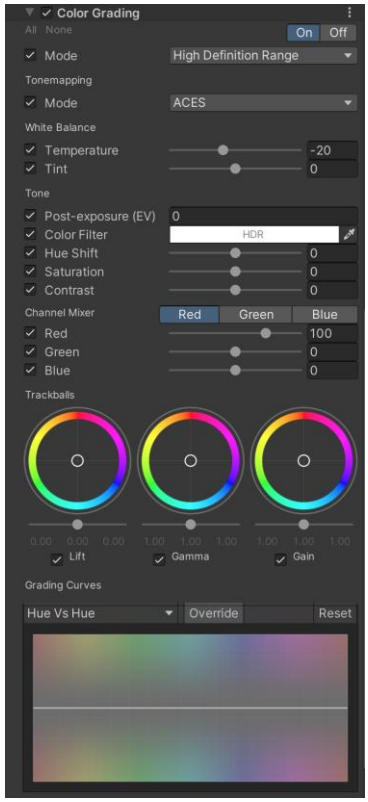
Oyun Programlama Ödevi – Hafta 10

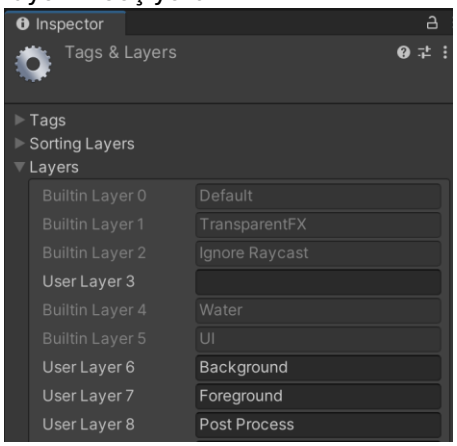
Ad: Eren
Soyad: Köse
Numara: 22360859075
GitHub Kod Linki: https://github.com/erennkose/btu-oyun-programlama/tree/main/Hafta10

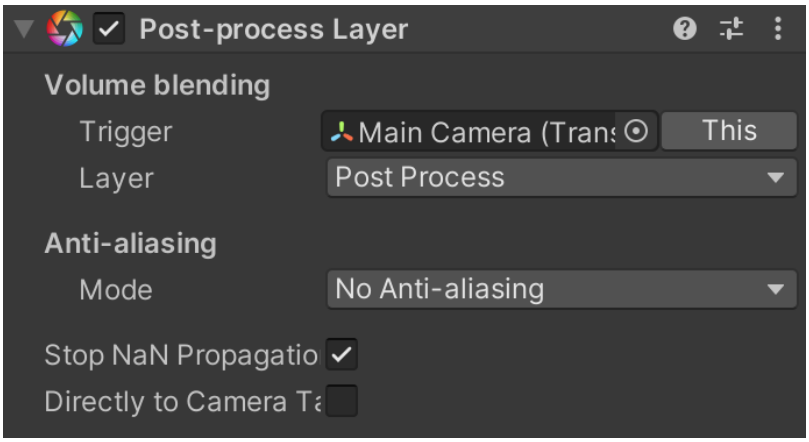
- **Post Processing Paketinin Kurulumu, Projeye Post Processing Uygulanması**
  Post processing paketini **Window > Package Manager** kısmından sağ üstteki arama çubuğuna **Post Processing** yazıp aratarak bulup sağda çıkan **Install** butonu ile yüklüyoruz. Ardından Hierarchy kısmından **Create Empty** seçeneğiyle bir boş oyun nesnesi oluşturup adını Post Process Volume yapıyoruz. Ardından bu nesnede **Add Component** diyerek **Post-process Volume** componentini ekliyoruz. **Is Global** seçeneğini aktif hale getiriyoruz. Ardından **Profile** kısmından **New** butonu ile yeni profil oluşturuyoruz. **Add Effect** butonuna tıklayıp sırasıyla **Bloom** ve **Color Grading** özelliklerini ekliyoruz. Altında gelen kısımlara **All** butonuna tıklayarak hepsini aktive ediyoruz. Ayarlarını ise alttaki görsellerde gözüken şekilde ayarlıyoruz.



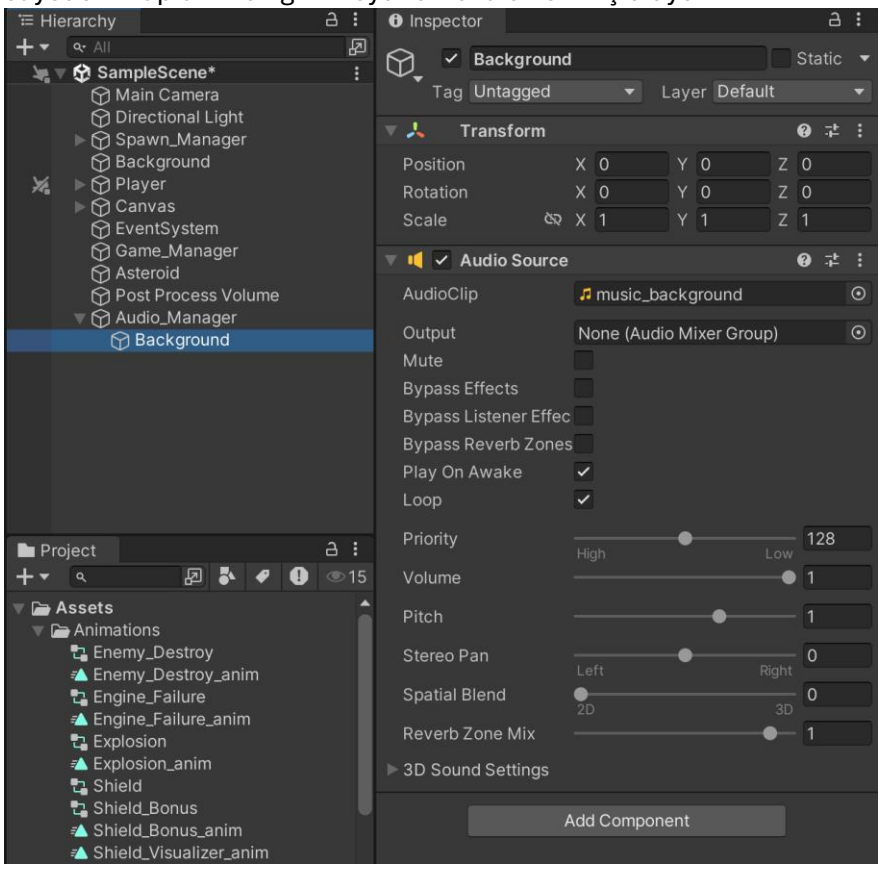


Nesnemizde bu ayarlamaları yaptıktan sonra **Hierarchy** kısmından **Main Camera** nesnesine tıklıyoruz. Burada **Add Component** diyerek **Post-process Layer** componentini ekliyoruz. Burada Layer eklememiz gerekiyor. Bunun için özel bir Layer oluşturmalıyız. Bunu yapmak için Inspector sekmesinde en üst sağda bulunan Layer kısmına tıklayıp **Add Layer** diyoruz ve **Post Process** adında bir layer ekliyoruz. Ardından Post-process Layer componentine geri dönüyoruz ve oradaki **Layer** kısmından **Post Process** layerını seçiyoruz.





- **Arkaplan Müziği Ekleme**
  Arka plan müziği eklemek için öncelikle **Hierarchy** kısmından **Create New** ile **Audio_Manager** adında bir nesne oluşturuyoruz. Altında başka bir boş nesne oluşturup adını **Background** koyuyoruz. Bu Background nesnesinde **Add Component** butonuna basıp **Audio Source** adındaki componenti ekliyoruz. Bu componentteki **AudioClip** kısmına, **Assets > Audio** klasöründeki **music_background** ses dosyasını sürükleyip bırakıyoruz. Ardınan ise **Play On Awake** ve **Loop** özelliklerini aktifleştiriyoruz. Bu sayede Arkaplan müziğimiz oyunumuza eklenmiş oluyor.



- **Lazer Müziği Ekleme**
  Lazer müziği eklemek için, öncelikle **Player** nesnemize **Audio Source** componentini ekliyoruz. Ardından ise **Loop** seçeneğini kapatıyoruz. Sonrasında **Player_sc** scriptimizi kod editörümüzde açıyoruz. Öncelikle **AudioSource** tipinde ve **AudioClip** tipinde iki değişken oluşturuyoruz.

```
AudioSource audioSource;

[SerializeField]
1 başvuru
AudioClip laserSoundClip;
```

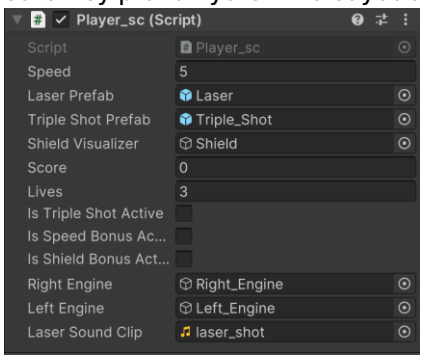
Ardından **Start** fonksiyonu içerisinde **audioSource** değişkenimize atama yapıyoruz.

```
audioSource = GetComponent<AudioSource>();
```

Atamamız null olması sonucunda hata mesajı ekrana yazdırıyoruz, eğer null değilse **audioSource.clip = laserSoundClip** atamasını gerçekleştiriyoruz.

```
if(audioSource == null){
    Debug.Log("Player_sc::Start audioSource nesnesi bulunamadı.");
}
else{
    audioSource.clip = laserSoundClip;
}
```

Bu işlem sonrasında Unity ekranımıza geri dönüyoruz ve **Player** nesnemizin **script** componentinin altındaki **Laser Sound Clip** kısmına **Assets > Audio** kısmında bulunan **laser_shot** ses dosyasını sürükleyip bırakıyoruz. Bu sayede lazerimize ses eklemiş oluyoruz.



- **Patlama Müziği Ekleme**

Patlama sesi eklemek için öncelikle Explosion prefabimize gidiyoruz. Burada **Add Component** diyerek **Audio Source** componentini ekliyoruz. Componentin altındaki **Audio Clip** kısmına **Assets > Audio** klasörü altındaki **explosion_sound** ses doysasını sürükleyip bırakıyoruz. **Loop** seçeneğinin **kapalı** olduğuna emin oluyoruz. Bu ayarlar **Asteroid** nesnemizdeki **patlama sesini** sağlayacaktır.

Ardından Enemy nesnelerimizin patlama sesinin eklenmesine geçiyoruz. Bunun için Enemy prefabimize **Add Component** diyerek **Audio Source** componentini ekliyoruz. Bu component altındaki **Audio Clip** kısmına yine explosion_sound sesini sürükleyip bırakıyoruz. **Play On Awake** ve **Loop** seçeneklerinin **kapalı** olduğundan emin oluyoruz. Bu işlemler sonrasında **Enemy_sc** scriptini açıyoruz. Burada Enemy nesnesinin Laser veya Player nesnesiyle teması sonrası bu sesin çıkmasını istiyoruz. Bunun için ise öncelikle gerekli değişken tanımlamasını yapıyoruz ve **Start** içerisinde Componenti atıyoruz.

```
AudioSource audioSource;
audioSource = GetComponent<AudioSource>();
```

Sonrasında **OnTriggerEnter2D** kısmında Player ve Laser nesneleriyle temasın tespit edildiği kısımda **audioSource.Play()** şeklinde sesin oynatılmasını sağlıyoruz.

```
void OnTriggerEnter2D(Collider2D other){
    if(other.tag == "Player"){
        // Canini azalt
        Player_sc playersc = other.GetComponent<Player_sc>();
        playersc.Damage();
        anim.SetTrigger("OnEnemyDeath");
        speed = 0;
        audioSource.Play();
        Destroy(this.gameObject, 2.5f);
    }
    else if(other.tag == "Laser"){
        Destroy(other.gameObject);
        if (player_sc != null){
            player_sc.UpdateScore(10);
        }
        anim.SetTrigger("OnEnemyDeath");
        speed = 0;
        audioSource.Play();
        Destroy(this.gameObject, 2.5f);
    }
}
```

- **Bonus Yakalama Müziği Ekleme**

  Bonus yakalama müziği için **Bonus_sc** scriptinde **AudioClip audioClip** şeklinde bir değişken oluşturuyoruz.

```
AudioClip audioClip;
```

  Ardından bu değişkeni kullanarak **OnTriggerEnter2D** içerisinde Player ile temas sonucunda bonus nesnesinin bulunduğu konumda bu sesin çalınması için gereken kod parçası olan **AudioSource.PlayClipAtPoint** fonksiyonunu ekliyoruz. İçerisine değişkenimizi ve nesnenin konumunu veriyoruz.

```
void OnTriggerEnter2D(Collider2D other){

    if(other.tag == "Player"){
        Player_sc playersc = other.transform.GetComponent<Player_sc>();
        AudioSource.PlayClipAtPoint(audioClip,transform.position);
        if(playersc != null){
            switch (bonusId){
                case 0:
                playersc.ActivateTripleShot();
                break;
                case 1:
                playersc.ActivateSpeedBonus();
                break;
                case 2:
                playersc.ActivateShieldBonus();
                break;
                default:
                Debug.Log("Default value in switch case");
                break;
            }
        }
        Destroy(this.gameObject);
    }
}
```

  Sonrasında ise Unity uygulamasından her bonus prefabimizin altındaki script componentinde bulunan **Audio Clip** kısmına **Assets > Audio** klasöründe bulunan **power_up_sound** ses dosyasını sürükleyip bırakıyoruz. Bu sayede bonus yakalama sesi de eklenmiş oluyor.