İSTANBUL GELİŞİM ÜNİVERSİTESİ

İSTANBUL GELİŞİM MESLEK YÜKSEKOKULU

BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ

BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

PIXEL PUMMEL 2D – MOBİL OYUN PROGRAMLAMA PROJESİ

FİNAL PROJE ÖDEVİ

Hazırlayan

Furkan Demir

Ödev Danışmanı

Öğ. Gör. Adnan Kürşat TEKE

İSTANBUL – 2024

ÖDEV TANITIM FORMU

YAZAR ADI SOYADI	**:** Furkan DEMİR
ÖDEVİN DİLİ	**:** Türkçe
ÖDEVİN ADI	**:** Pixel Pummel 2D – Mobil Oyun Programlama Projesi
BÖLÜM	**:** Bilgisayar Teknolojileri
PROGRAM	**:** Bilgisayar Programcılığı
ÖDEVİN TÜRÜ	**:** Final Proje Ödevi
ÖDEVİN TESLİM TARİHİ	**:** 29.05.2024
SAYFA SAYISI	**:** 21
ÖDEV DANIŞMANI	**:** Öğr. Gör. Adnan Kürşat TEKE

BEYAN

Bu ödevin/projenin hazırlanmasında bilimsel ahlak kurallarına uyulduğunu, başkalarının ederlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, ödevin/projenin herhangi bir kısmının bu üniversite veya başka bir üniversitedeki başka bir ödev/proje olarak sunulmadığını beyan eder, aksi durumda karşılaşacağım cezai ve/veya hukuki durumu kabul eder; ayrıca üniversitenin ilgili yasa, yönerge ve metinlerini okuduğumu beyan ederim.

29.05.2024

Furkan DEMİR

KABUL VE ONAY SAYFASI

220111597 numaralı **Furkan DEMİR'**in Unity – C# programı kullanılarak yapılan Pixel Pummel 2D adlı çalışması, benim tarafımdan Final Ödevi olarak kabul edilmiştir.

Adnan Kürşat TEKE

Öğretim Görevlisi

İÇİNDEKİLER

ÖZET

Bu raporda Unity ve Microsoft Visual Studio 2022 programı kullanılarak Pixel Pummel 2D adlı proje benim tarafımdan yapılmış olup projede adım adım ilerlediğim, Ana Menü yapımı, Harita 1 yapımı, Harita 2 yapımı ve Harita 3 yapımı detaylı bir şekilde anlatılmıştır.

ÖN SÖZ

Bu proje çalışmasında Unity ve Microsoft Visual Studio 2022 programı kullanılarak Pixel Pummel 2D adlı oyun projesinin yapımı anlatılmıştır. Ödevin hazırlanmasında, bu uygulamanın kullanımını eğitim ve öğretim hayatında bizlere anlattığı dersler sayesinde öğreten Adnan Kürşat TEKE hocama teşekkür ediyorum.

Furkan DEMİR

a. Genel Bilgiler

- **Öğrencinin Öğretim Durumu (NÖ/İÖ) :** 2. Sınıf Normal Öğretim
- **Öğrencinin Adı Soyadı :** Furkan DEMİR
- **Öğrencinin Numarası :** 220111597
- **Programı :** Bilgisayar Teknolojileri Bölümü Bilgisayar Programcılığı Programı
- **Projenin Adı :** C# Platform Oyunu Projesi – ZorİO
- **Projenizin Ana Formunun İsmi :** Form1

b. Projenizin amacı nedir ?

Bu projenin amacı aynı ekranda aynı anda 2 kişinin oynayabildiği ve aynı eğlenceye sahip olması amaçlanmıştır.

c. Projeniz nasıl bir problemi çözmüştür ?

Bu proje 2 kişinin aynı anda eğlenmeyi hedef alarak insanların birbiriyle sosyalleşmesi ve güzel vakit geçirmesi düşünülerek yapılmıştır.

d. Projeniz tamamen orijinal mi ?

Bu proje başka projeler ve oyunlardan esinlenerek yapılmıştır. Sonuç olarak orijinal değildir. Bu projeye benzer farklı platform oyunları mevcuttur.

e. Projeniz neden önemlidir ?

Oyun ve eğlence anlamında insanların sıkılmaması ve eğlenceli vakit geçirmesi için önemli olduğu düşünülmüştür.

f. Projenizde tamamladığınız hedefler nelerdir ?

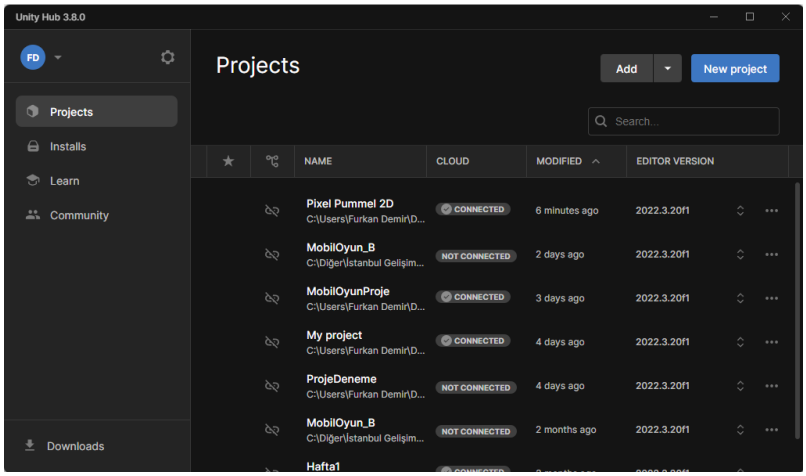
Projedeki hedefimiz öncelikle 2 kişinin aynı anda kontrol edebileceği karakterler yapmaktı. Ardından oyuncuların tek bir harita oyunu oynamak yerine birden fazla farklı haritalar tasarlayarak oyuncuların sıkılmaması ve eğlenerek oynamaları hedef alındı.

g. Projenizde eksik kalan hedefler nelerdir ?

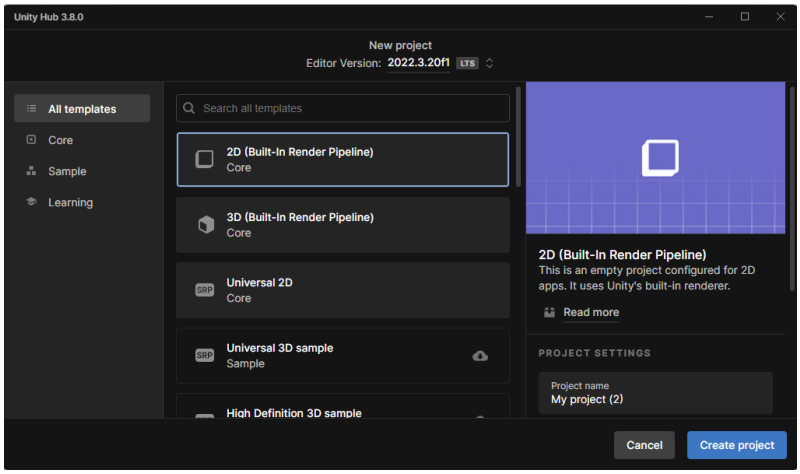
Projede eksik kalan hedef yok. Aslında tüm hedeflere ulaşıldı fakat karakterlerin animasyonları istenilen gibi olmadı. Bazı haritalarda bug mevcut.

1. UNITY'DE YENİ PROJE NASIL OLUŞTURULUR ?

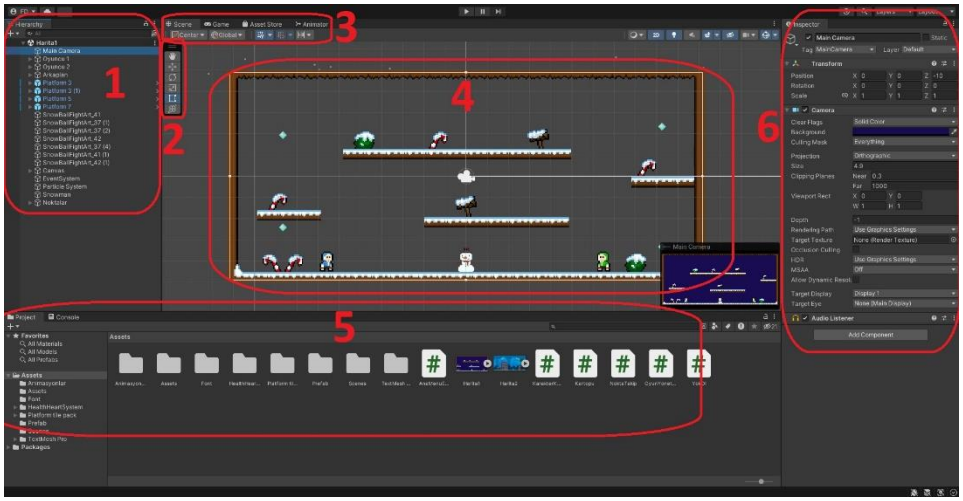
Unity Hub programı kullanılarak "New Project" kısmına girilir. Ardından yapılacak projenin şablonu seçilir. Bu projede 2D şablonu kullanılmıştır. "Project Name, Location ve Unity Organization" bilgileri girildikten sonra "Create Project" butonuna tıklanır ve yeni proje oluşturulur.



Resim - 1 Unity Hub Projects Ekranı



Resim - 2 Unity Hub Templates Ekranı



Resim - 3 Unity Sample Scene Ekranı

Yeni proje oluşturulduktan sonra çalışacağımız alan yani "Sample Scene" ekranı gelir.

1 numara ile işaretlenen yer "Hierarchy" kısmı. Oyunda bulunan bütün nesneleri içeren alandır.

2 numaralı alanda kamera ve oyuna eklenen nesnelerin yer, yön, boyut gibi özelliklerini değiştirebileceğimiz seçenekler yer almaktadır.

3 numaralı alanda "Scene" ekranı, "Game" önizleme alanı ve projede kullanılacak Assetleri indirmek için kullanılacak "Asset Store" ekranı yer almaktadır.

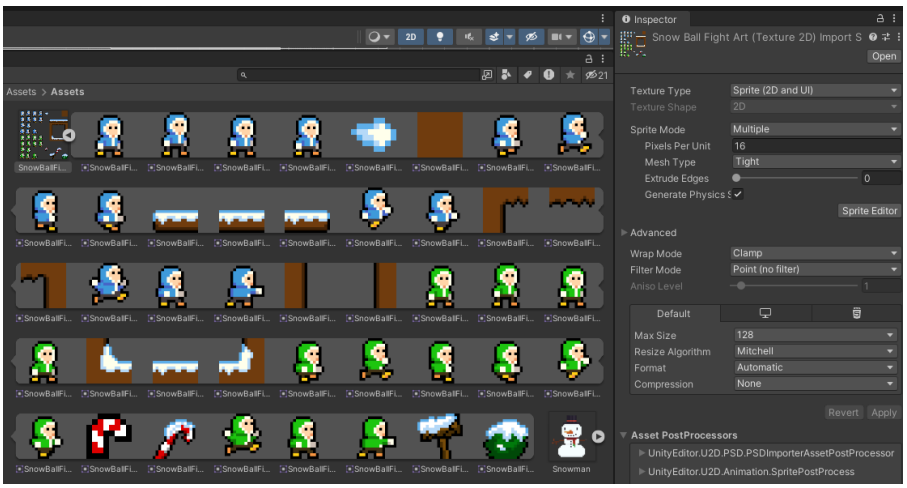
4 numaralı alan "Scene" ekranıdır. Projede yapılacak tüm işlemler bu ekran üzerinde görüntülenir.

5 numaralı alan projemizde bulunan tüm bileşenlerin bulunduğu, Asset Store'da indirilen bileşenlerin projeye aktarıldığı bölümdür.

6 numaralı alan projeye eklenen nesnenin tüm özelliklerini gösteren alandır. "Add Component" kısmında seçili nesneye farklı hazır özellikler ekleyebilir veya "New Script" tuşuna basarak kendimiz özellik yaratabiliriz.

2. PIXEL PUMMEL 2D YAPIM AŞAMASI

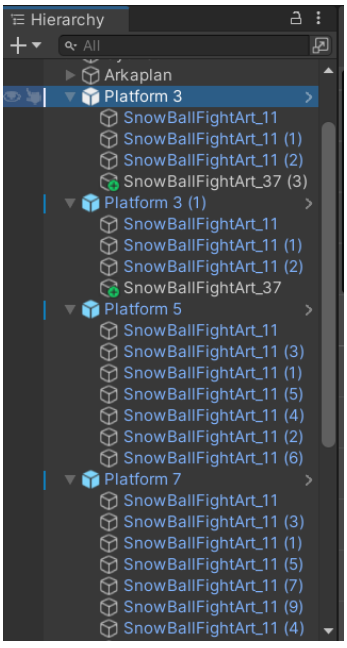
Yeni çalışma alanı oluşturduktan sonra Asset Store'dan projede kullanılacak nesneler indirilip projeye aktarıldı. Projeye eklenen Asset'e tıklanıp "Inspector" penceresinde bulunan "Pixels Per Unit" özellik değeri 16, "Max Size" özelliği 128 yapıldı ve "Sprite Editor" ekranına geçildi. "Sprite Editor" ekranında her parça 16x16'lık karelere bölündü.



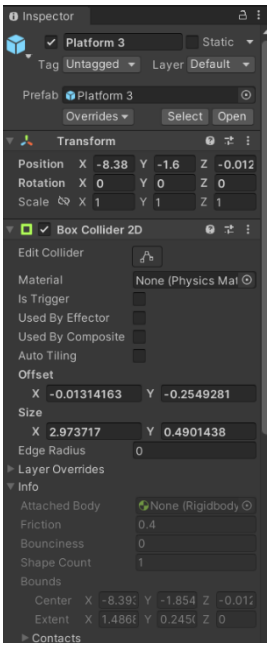
Resim - 4 Asset Sprite Editor Ekranı

2.1. Pixel Pummel 2D Harita Oluşturma

"Scene" ekranına "Assets" klasöründen haritamızı oluşturmaya başladık. "Main Camera" sınırlarına sığacak şekilde zeminlerimizi yerleştirdik. Karakterimizin haritanın altına düşmemesi için tüm zeminlere "Box Collider 2D" özelliği ekledik.

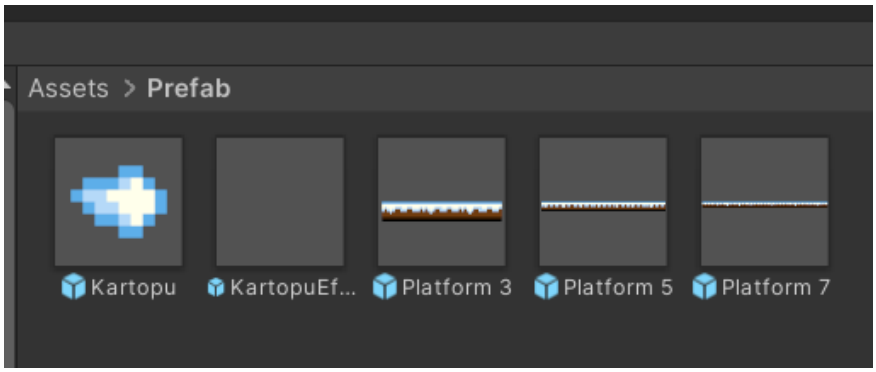


Resim - 6 Zeminler



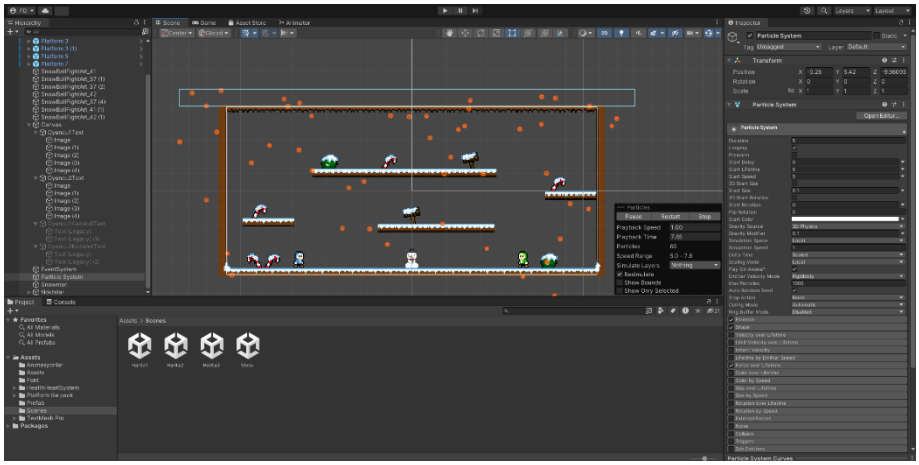
Resim - 5 Zemin Özellikleri

Haritaya eklediğimiz zeminleri “Platform” adında klasöre topladık ve onları tekrar kullanabilmek için “Prefab” klasörüne sürükledik.



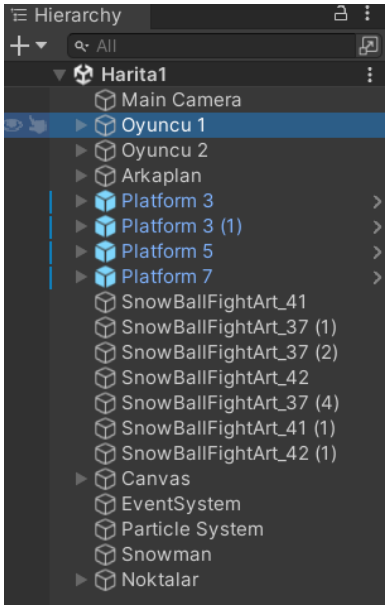
Resim - 7 Prefab Klasörü

Harita 1’de bulunan kar detayı için “GameObject > Effect > Particle Effect” ekledik ve gerekli ayarlamaları yaptık. Böylece kar yağıyormuş gibi animasyon verdik.

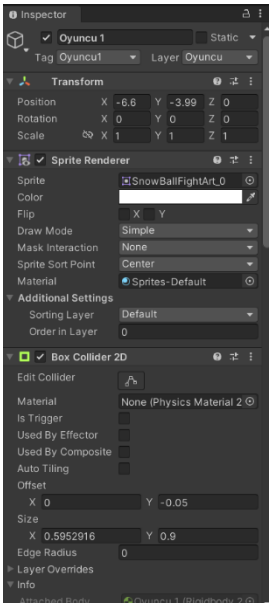


2.2. Pixel Pummel 2D Karakter Oluşturma (Oyuncu 1)

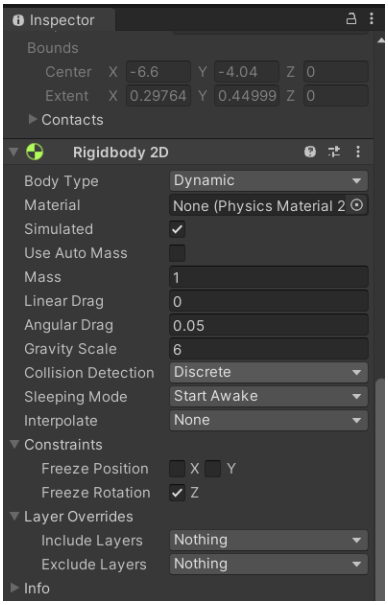
Oyunumuzda bulunan karakterimizn adını "Oyuncu 1" olarak değiştirdik ve "Inspector" penceresinde bulunan "Add Component" kısmından fizik motoru olan "Rigidbody 2D" özelliğini ekledik ve bu sayede karakterimize hareket özelliği kazandırdık. Ardından "Box Collider 2D" ekleyerek karakterimizin zeminde bulunan "Box Collider 2D" ile temas ederek harita altına düşmemesini sağladık. Karakterimizin koşarken veya zıplarken takılıp takla atmaması için "Z eksen özelliğini devre dışı bıraktık.



Resim - 10 Karakter Oluşturma

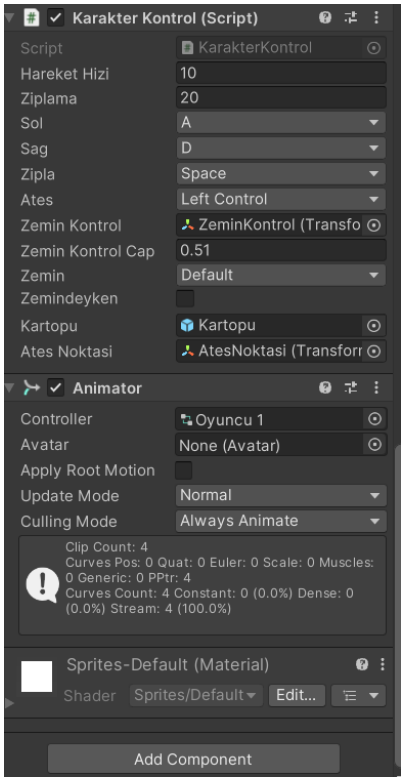


Resim - 9 Oyuncu 1 Box Collider 2D

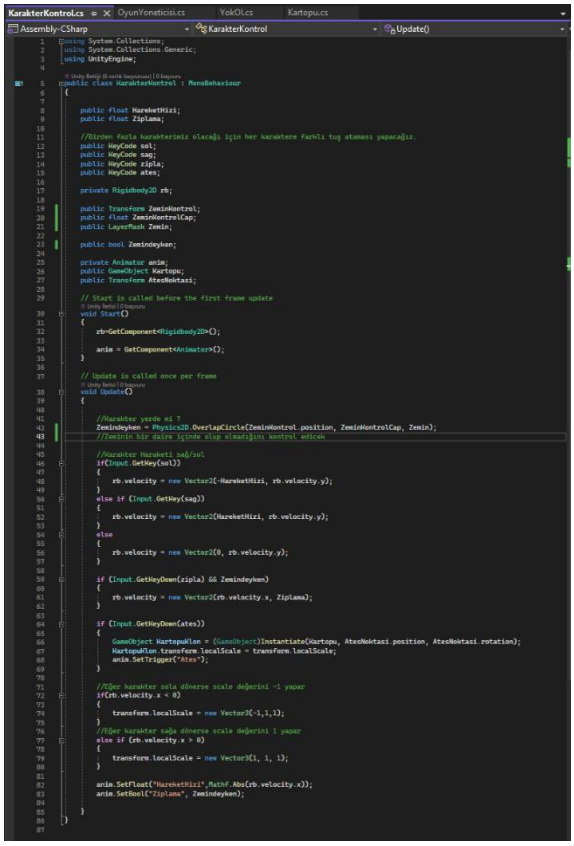


Resim - 8 Oyuncu 1 Rigidbody 2D

Karakterimize gerekli özellikleri atadıktan sonra artık karakterimizi kontrol etmek için kendimiz özellik ekleyeceğiz. “Add Component” kısmından “New Script” seçiyoruz ve “KarakterKontrol” adında script oluşturuyoruz. Oluşturduğumuz Script’e çift tıklayarak Microsoft Visual Studio programımızı açıyoruz ve buraya karakterimizde bulunmasını istediğimiz özellikleri yazarak ekliyoruz. Karakterimizin yürüme hızı, zıplama gücü, kontrol tuşları, karakterimizin zeminde olup olmadığının kontrolü ve bunun gibi bir çok özelliği düzenleyebiliyoruz.



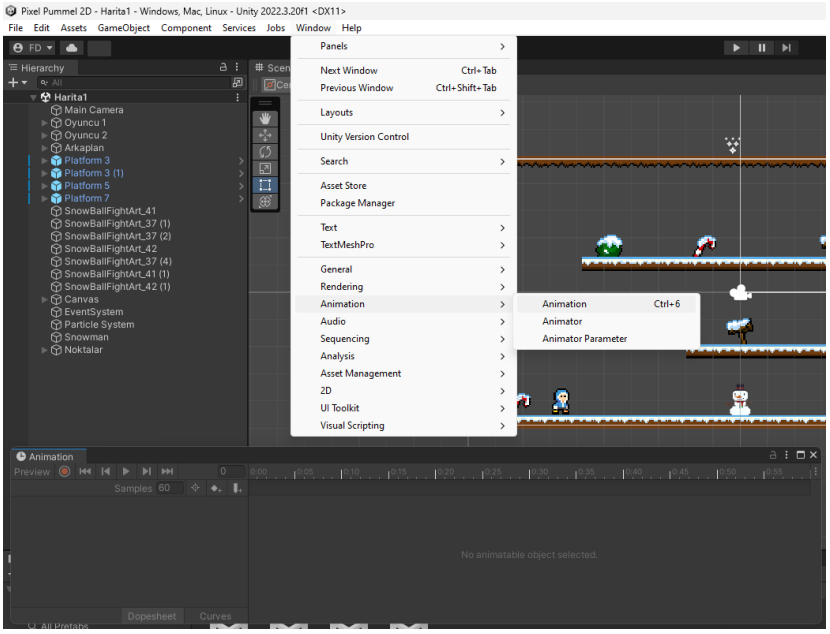
Resim - 12 Oyuncu 1 Kontrolleri Script



Resim - 11 Karakter Kontrolleri Kod Kısmı

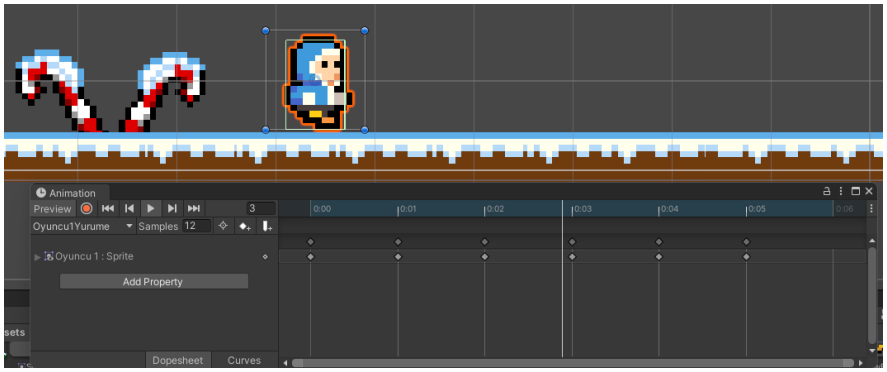
2.3. Karaktere Animasyon Ekleme (Animator)

Karakterimizin hareket halindeyken animasyonu olması için "Animation" penceresine erişmemiz gerekmektedir. Animator penceresine erişmek için "Window > Animation > Animation" seçeneklerinden veya "Ctrl+6" kısayolu ile erişilir.



Resim - 13 Animation Penceresi Açma İşlemi

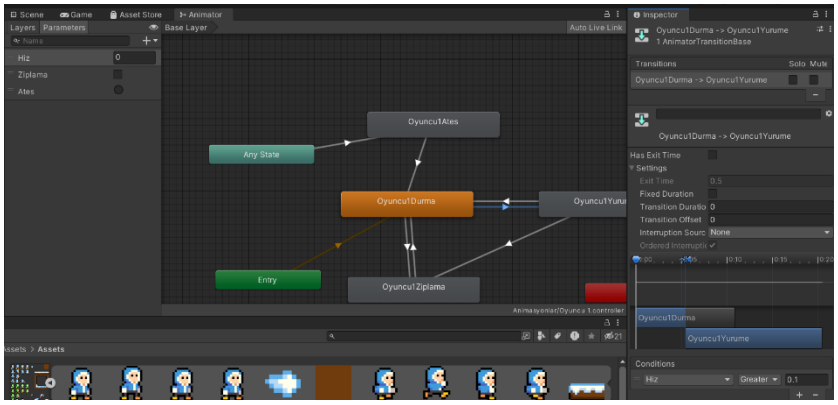
Animation penceresine eriştikten sonra Assets klasörümüzün içinde karakterin Idle, Jump, Fight vb. geçişleri seçilip "Animation" penceresine sürüklenir. "Samples" kısmında animasyonun saniyede kaç kere tekrarlanması isteniyorsa değerler girilir.



Resim - 14 Animation Penceresi

“Animation” eklendikten sonra “Window > Animation > Animator” penceresine gelinir ve karakterin animasyon geçişleri ayarlanır.

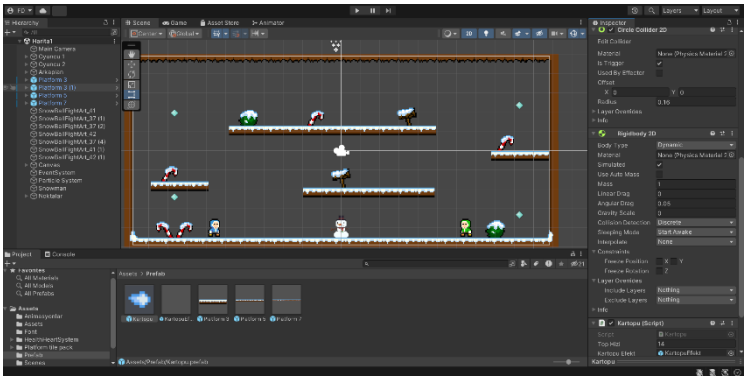
“Layers” kısmında karakterimizin animasyon geçiş hızlarını ayarlamak için katman ekliyoruz. Karakterin hızını ayarlamak için adını “Hiz” koyduk. Bunun gibi diğer özellikler için katmanlar oluşturduk. Karakterin durma animasyonundan yürüme animasyonuna geçmesi için “Oyuncu1Durma”ya sağ tık yapılır ve “Make Transition” kısmına tıklanır ve “Oyuncu1Yurume”ye tıklanır. Eklenen animasyonu özelleştirmek için “ok”a tıklanır. Ardından “Inspector” penceresinde, karakterin durma pozisyonundan yürüme pozisyonuna anında geçmesi için “Has Exit Time”ı devre dışı bırakıyoruz ve “Conditions” kısmına eklediğimiz katmanı girip “Greater” seçiyoruz ve “0.1” yapıyoruz. Diğer animasyonlara da aynı işlemler yapıldı.



Resim - 15 Animator Penceresi

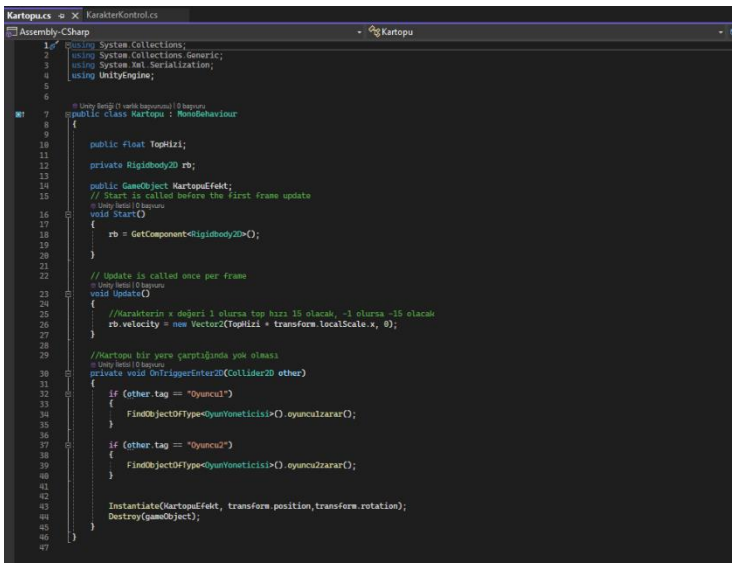
2.4. Ateş Etme Kod ve Özellikleri

Karakterlerin birbiriyle savaşması ve canlarını götürmesi için "Kartopu" adını verdiğimiz nesne ekledik. Bu nesneye "Circle Collider 2D" ve "Rigidbody 2D" özelliği atadık. Kartopu'nun ateş ettikten sonra düz bir çizgide gitmesi için Rigidbody 2D'de bulunan "Gravity Scale" özelliğini "0" yaptık. Kartopu nesnesini tekrar kullanmak için Prefab nesnesine dönüştürdük.



Resim - 16 Kartopu Nesnesi Özellikleri

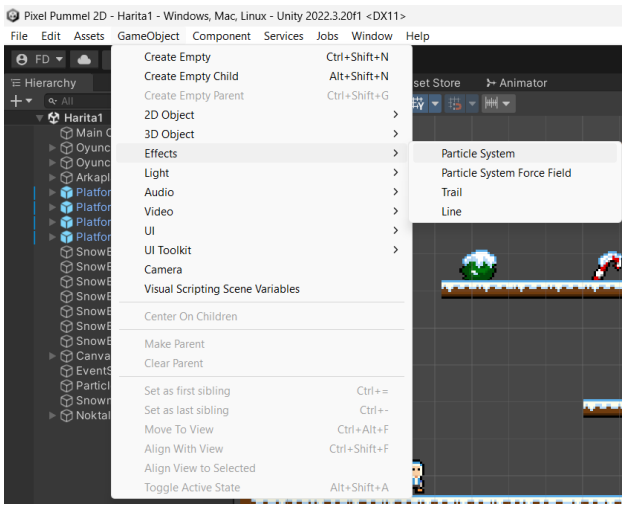
Ardından "Add Component > New Script" kısmından "Kartopu" adında Script oluşturduk. Bu scripte kartopunun hızını ve kartopunun herhangi bir Collider 2D ile temasa geçtiğinde yok olması için gerekli kodlar yazıldı.



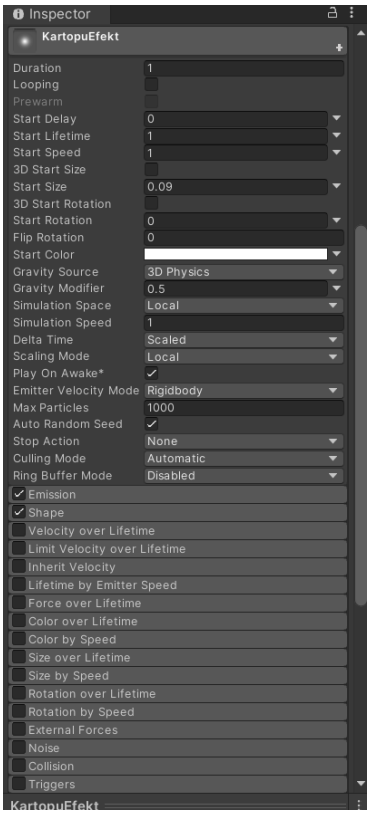
```
using System.Collections;
using System.Collections.Generic;
using System.Xml.Serialization;
using UnityEngine;


public class Kartopu : MonoBehaviour
{

    public float TopHizi;

    private Rigidbody2D rb;

    public GameObject KartopuEfekt;
    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();

    }

    // Update is called once per frame
    void Update()
    {
        //Karakterin x değeri 1 olursa top hızı 15 olacak, -1 olursa -15 olacak
        rb.velocity = new Vector2(TopHizi * transform.localScale.x, 0);
    }

    //Kartopu bir yere çarptığında yok olması
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Oyuncu1")
        {
            FindObjectOfType<OyunYoneticisi>().oyuncu1zarar();
        }

        if (other.tag == "Oyuncu2")
        {
            FindObjectOfType<OyunYoneticisi>().oyuncu2zarar();
        }


        Instantiate(KartopuEfekt, transform.position,transform.rotation);
        Destroy(gameObject);
    }
}
```

Resim - 17 Kartopu Script Kodları

Kartopumuzun herhangi bir Collider 2D'ye temas ettiğinde/çarptığında animasyon çıkması için "GameObject > Effects > Particle System" ekledik.

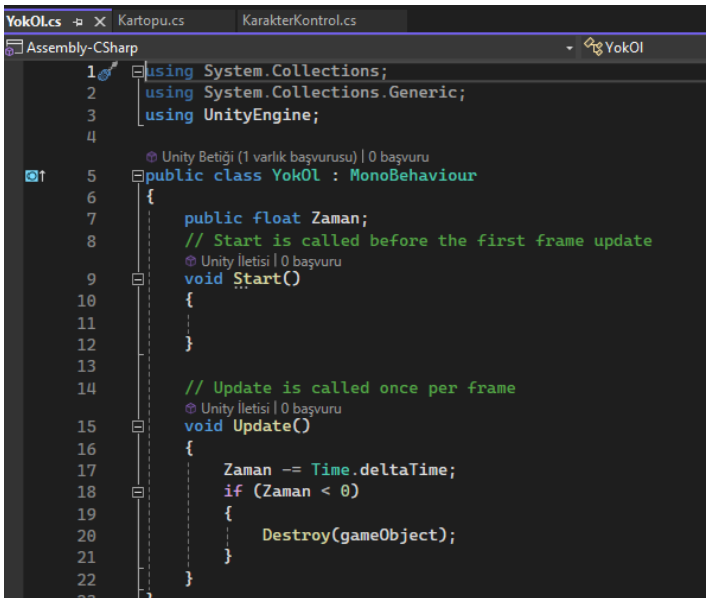


Resim - 18 Particle System Ekleme



Resim - 19 Particle System Özellikleri

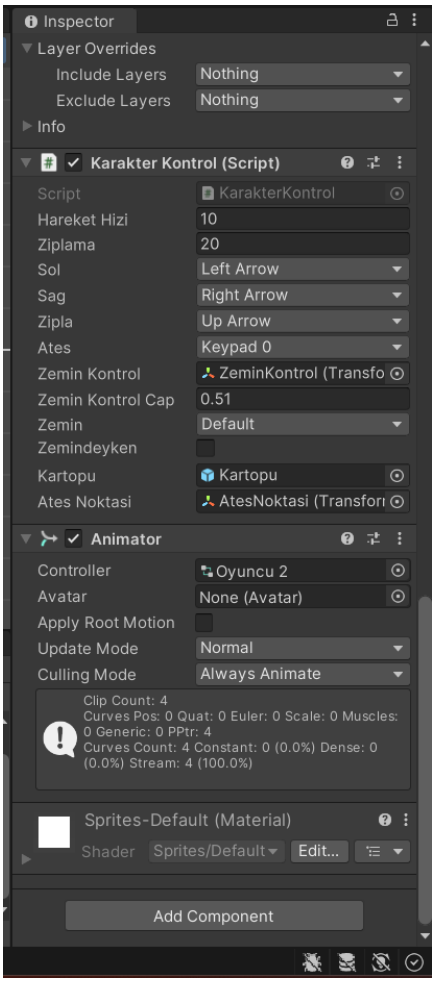
Animasyonun ardından Kartopu'nun kaybolması için ve arka planda yer kaplamaması için "Yok Ol" adında Script oluşturduk.



Resim - 20 Yok Ol Script Kodları

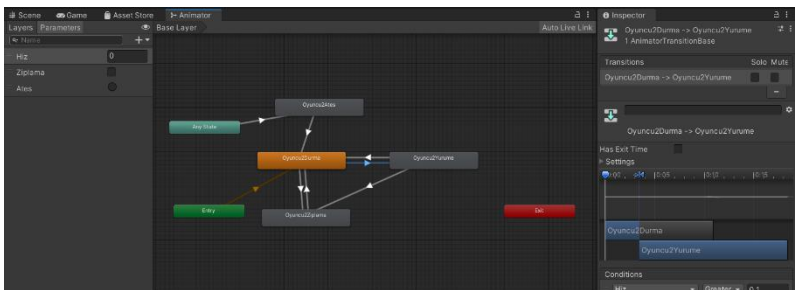
2.5. Oyuncu 2 Yapım Aşaması

Oyuncu 1 karakterimizi Ctrl+D yaparak çoğalttık ve yeni kopyanın adını "Oyuncu 2" olarak değiştirdik. Ardından Assets'de bulunan 2. karakterin görseliyle "Animator" penceresinde değiştirdik. Oyuncu 2'nin kontrol tuşlarını değiştirdik.



Resim - 21 Oyuncu 2 Özellikleri

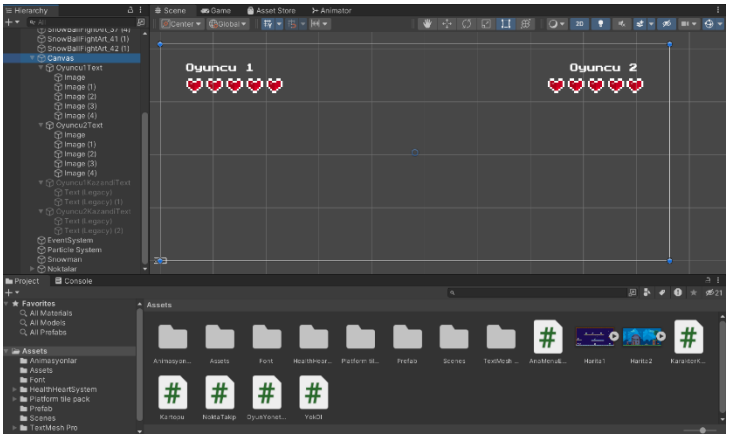
Oyuncu 1'i kopyalayıp çoğalttığımız ve Oyuncu 2 olarak değiştirdiğimiz için Oyuncu 2'nin animasyonlarını "Animator" penceresinden "Oyuncu 1" ile ilgili olan animasyonları "Oyuncu 2" olarak değiştirdik.



Resim - 22 Oyuncu 2 Animasyonları

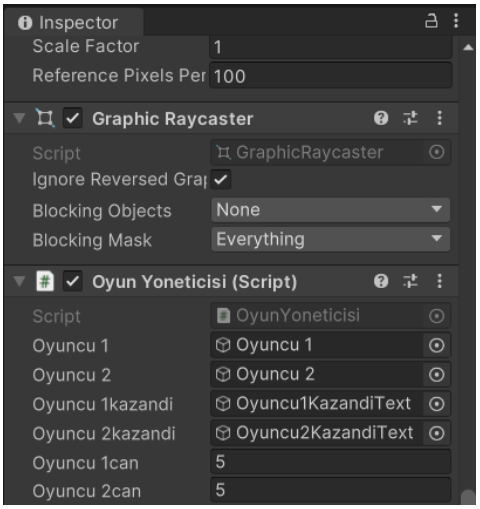
2.6. Sağlık Sistemi Yapım Aşaması

Artık oyunda 2 karakterimiz var, karakterlerimizin can barı olacak ve "Kartopu" nesnesine temas eden karakterin canı -1 azalacak. "Hierarchy" penceresinde boş bir yere sağ tık yaptıktan sonra "UI > Legacy > Text" seçtik ve "Oyuncu 1, Oyuncu 2" adlarını yazdık. Ardından "OyuncuText" içlerine sağ tık yapıp "UI > Image >" seçtik ve Assets kısmından "Kalp" nesnemizi sürükleyip bıraktık. Böylece "Scene" ekranında arayüzümüzü tasarladık.



Resim - 23 Sağlık Sistemi

Arayüzün ardından "Kalp" nesnelerine özellik eklemek için "Canvas" içine "Add Component" kısmından "Oyun Yoneticisi" adında Script oluşturduk. Bu scriptte Oyuncular, oyuncuların can sayısı, ve nesnelerin Image'leri tanımlandı.



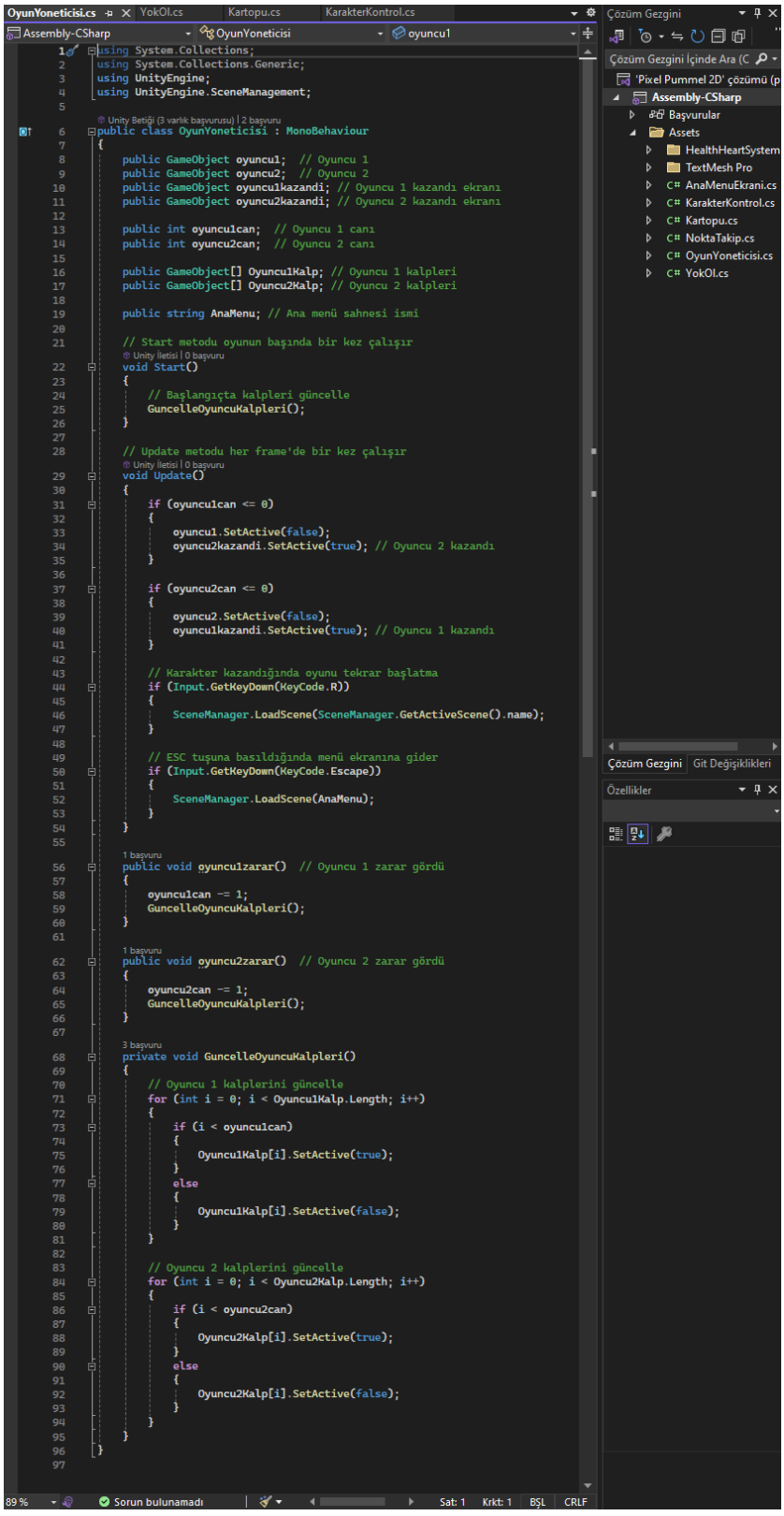
Resim - 25 Oyun Yoneticisi Script Özellikleri 1

Resim - 24 Oyun Yoneticisi Script Özellikleri 2

2.7. Oyun Bitti Ekranı Yapım Aşaması

Oyuncu 1 veya Oyuncu 2 kazandığında ekrana "Oyuncu 1 Kazandı !" mesajı gelir. Klavyeden "R" tuşuna basılırsa oyun tekrar başlar. "ESC" tuşuna basılırsa Ana Menüye geri döner. Bu işlemler "Oyun Yoneticisi" scriptinde yazılan kodda bulunmaktadır.

Resim - 26 Oyun Bitti Ekranı

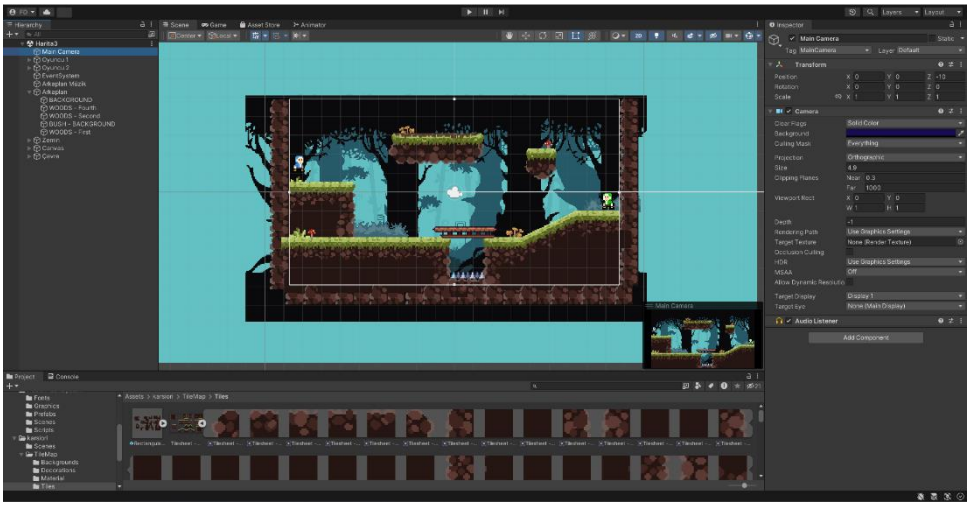


Resim - 27 Oyun Yoneticisi Script Kodları

2.8. Harita 2 ve 3 Yapım Aşaması

Project kısmında bulunan “Assets > Scenes” ekranından Harita1’e CTRL+D yapıp çoğalttık ve adını “Harita2” olarak değiştirdik. “Hierarchy” kısmında bulunan nesneleri kaldırdık ve Asset Store’dan farklı Asset import ettik. Eklediğimiz Assette bulunan nesneleri sürükle bırak yaparak harita tasarladık. Zemin olmasını istediğimiz yerlere “Box Collider 2D” özelliği ekledik.

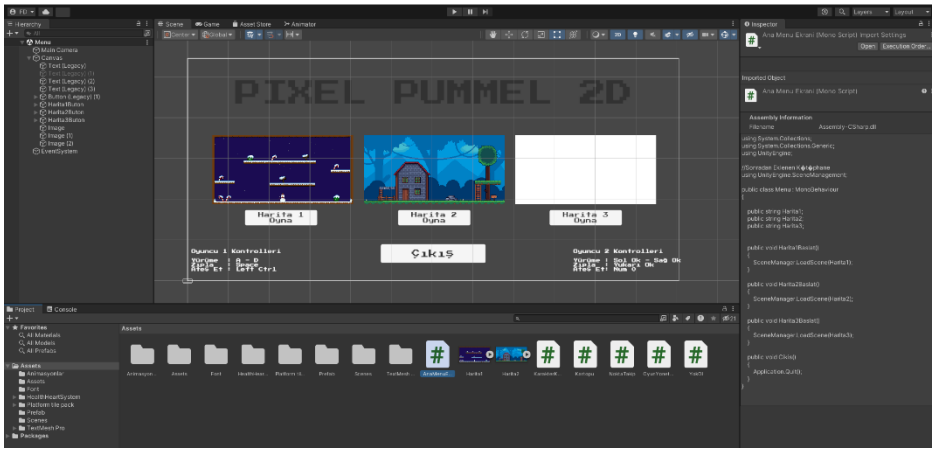
Resim - 28 Harita 2 Tasarımı



Resim - 29 Harita 3 Tasarımı

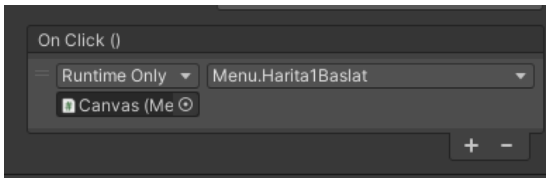
2.9. Ana Menü Yapım Aşaması

Ana Menü düzeni Text, Image ve Button ekleyerek yapıldı. Harita1, Harita2 ve Çıkış butonları eklendi. Bu butonların özellikleri script ile atandı.

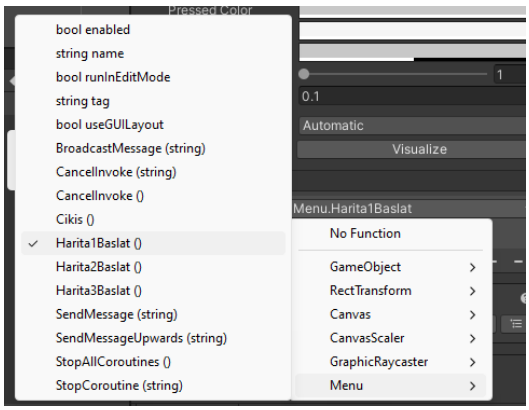


Resim - 30 Ana Menü Tasarımı

Button'a tıklandığında hangi sahneye gitmesi gerektiğini script sayesinde eklediğimiz özellikte seçiyoruz.

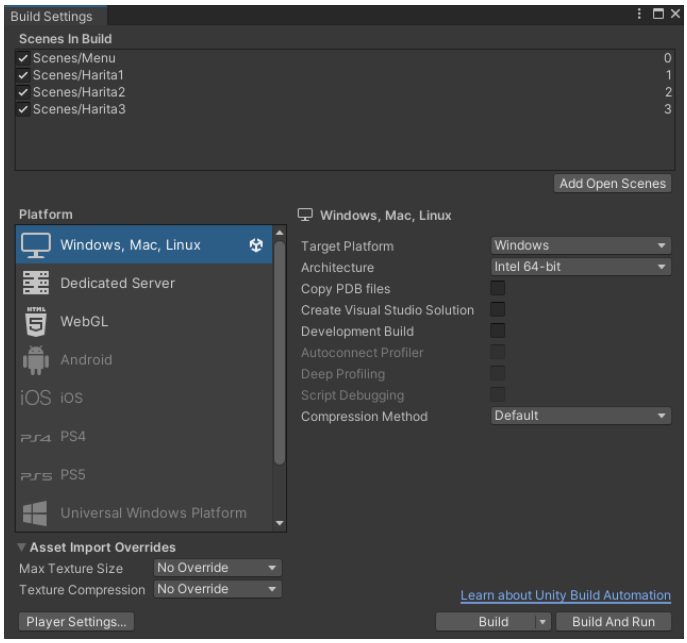


Resim - 32 Button Inspector



Resim - 31 Button Inspector 2

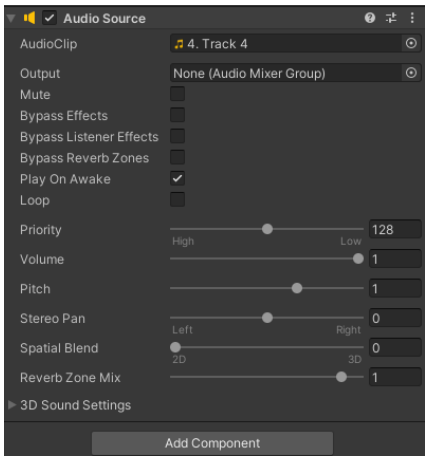
Sahneler oluşturulduktan sonra doğru bir şekilde çalışması için "Build Settings" penceresinde indis sayısına göre sahneler sıraya koyulur. Eğer sahneler gözükmüyorsa Project klasöründe Scenes klasöründen sürükle bırak yaparak veya "Add Open Scenes" butonuna basılır.



Resim - 33 Build Settings Ekranı

2.10. Arka Plan Müzik Ekleme

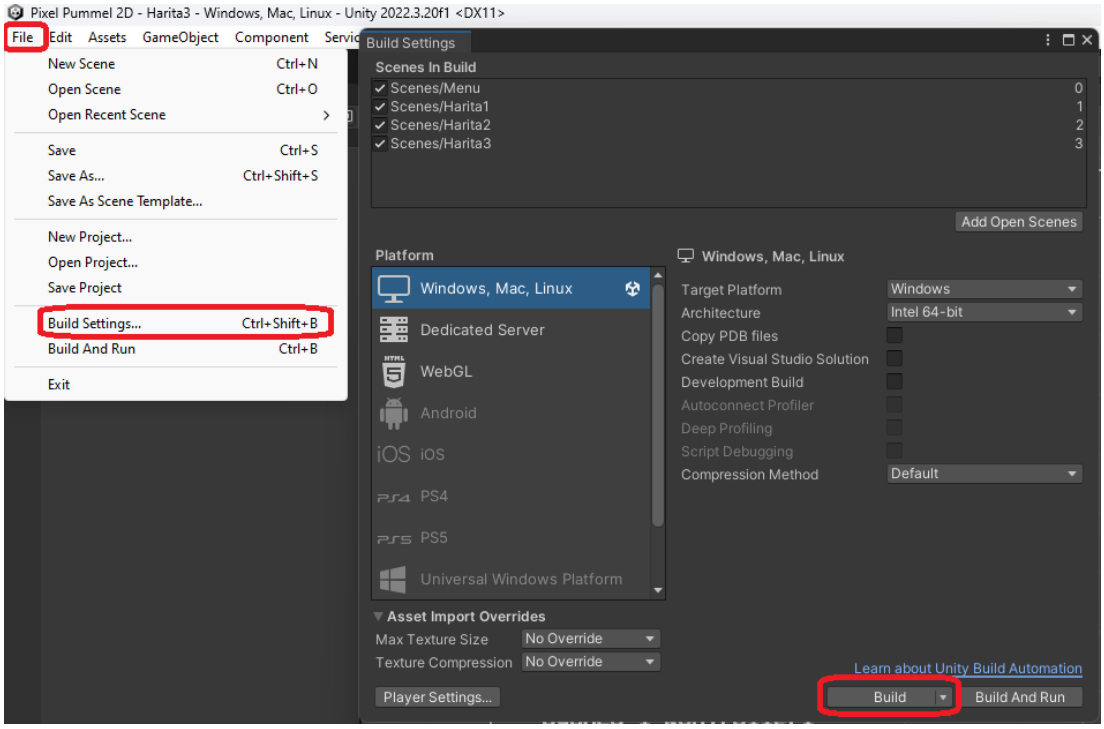
Arka plana müzik eklemek için "Hiyerarchy" penceresinde yeni boş klasör oluşturulur. Oluşturulan yeni klasöre isim verdikten sonra "Inspector" penceresinde "Add Component" kısmından "Audio Source" eklenir. "AudioClip" kısmından proje klasörümüzdeki ses seçilir ve sesin sürekli çalmasını istediğimiz için "Play on Awake" seçeneği işaretlenir.



Resim - 34 Arka Plan Müzik Ekleme

2.11. Oyunun Exe Dosyasına Çıkarılması

Oyun tamamlandıktan sonra .exe dosyasına çıkarmak için “File > Build Settings > Build” adımları uygulanır. “Build” tuşuna tıklandıktan sonra klasörün nereye kaydedileceği seçilir.



Resim - 35 Build Settings Ekranı

KAYNAKÇA

1- https://youtu.be/bMTrDpKU100?si=bCGKNzwC1-B2GLJ8
(Son Erişim : 22.05.2024)

2- https://youtu.be/nPigL-dIqgE?si=SQeqfFxIlnmd7Qc8
(Son Erişim : 22.05.2024)

3- https://youtu.be/UhZvGJ4SJZ8?si=IhGJv-Hyneho4ftO
(Son Erişim : 22.05.2024)

4- https://assetstore.unity.com/packages/2d/environments/platform-tile-pack-204101
(Son Erişim : 22.05.2024)

5- https://youtu.be/6xn0Sokihdc?si=AePpjt8w0IqC7dE8
(Son Erişim : 22.05.2024)

6- https://youtu.be/TcranVQUQ5U?si=pDCCCHq7_CjL6Dzh
(Son Erişim : 22.05.2024)

7- Snowball_Fight_Resources.zip (dropbox.com)
(Son Erişim : 22.05.2024)

8- 8Bit Music Album - 051321 | Audio Music | Unity Asset Store
(Son Erişim : 26.05.2024)