Kütahya Dumlupınar Üniversitesi Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Oyun Programlama Dersi Final Sınavı

Proje Bilgileri:

Proje Adı: Çöldeki Şövalye

Ders Adı: Oyun Programlama Dersi

Ders Kodu: 131727509

Öğretim Üyesi:

Dr. Öğr. Üyesi Emre GÜNGÖR

Dumlupınar Üniversitesi, Kütahya

Hazırlayan Öğrenci:

Yasin CAN – 201613172907

Giriş

Bu rapor “Çöldeki Şövalye” isimli 2D olarak Unity üzerinde C# script dili kullanılarak geliştirilen oyun projesini açıklamaktadır. Oyunda, şövalye görünümüne sahip oyuncu ve “wraith” isimli düşman bulunmaktadır. Oyun çöl ortamında kurgulanmış, buna yönelik nesneler eklenmiştir. Oyunun kullanıcı ara yüzü tamamlanmış olup bu ekran başlangıç ekranı olarak belirlenmiştir.

Arkaplan, menü, ses, hareketli platform, kayıt işlevleri, ruh toplama verileri gibi eklenen özelliklerden ilerleyen kısımlarda bahsedilecektir.

Kullanılan Teknolojiler

Oyun Motoru: Unity 2020.3.19f1

Programlama Dili: C#

Kodlama Editörü: Visual Studio Code 1.62.1

Sınıflar ve Sorumlulukları

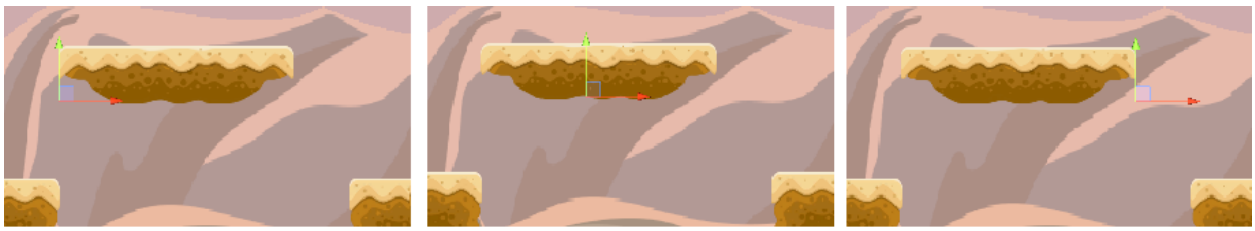
Numarası	Sınıf Adı	Sorumlulukları
1	BgParallaxScroll.cs	Oyunda arka plan hareketlerini yönetmek için kullanılmaktadır. Oyuncu hareketlerini baz alarak arka planda Material üzerinden offset değişiklikleri yoluyla ters yönlü hareket işlevi yürütmektedir.
2	GameManagerSc.cs	Kullanıcı tarafından gelen isteklerin arka planda yürütülmesini sağlamaktadır. Bu istekler oyunun başladığı sahneye gitmek, aktif sahneyi takip etmek, oyun içi duraklama menüsü gösterimi, ana menü buton işlevlerinden oluşmaktadır.
3	InGameCanvas.cs	Oyun içerisinde pause (“P” & “Esc”) tuşu ile gelen menü işlevlerini yürütür. Oluşturulan metotlar ile oyuna devam, kayıt, kayıttan

		yükleme, ayarlar, çıkış gibi buton işlevlerini yürütür.
4	Platform2Dmove.cs	Hareketli platform için belirlenen 3 hedef nokta arasında 1-2-3-2-1 sırası ile sürekli kayma hareketi yapmayı sağlar. Oyuncu üzerine geldiğinde parent'ı olarak ayarlayarak hareketine oyuncuyu da dahil etmektedir.
5	SaveLoad.cs	Oyunu kaydetme ve kayıtlı oyunu yükleyerek oyuncuya kaldığı yerden devam etme imkânı sağlamaktadır. Oyuncu verilerini alarak bu dosyayı oyunun klasörüne "GameSaveData.save" isimli dosya olarak kaydetmektedir.
6	Singleton.cs	Programlamada diğer sınıflar ile implementasyonda kullanılacak olan soyut tanımlı sınıftır.
7	SoundManager.cs	Oyun müziklerinin kontrolünü sağlamakta ve sahnelere göre değişiklik durumlarını yönetmektedir.

Hareketli Platform

Oyunda geçiş yolu üzerinde iki uzak zemin arasında ilerleyebilmek için üzerinden geçilmesi gereken hareketli platform oluşturulmuştur. Oyuncu bu platform üzerinde kalırsa platform oyuncuyu da alarak hareketine devam edecektir.

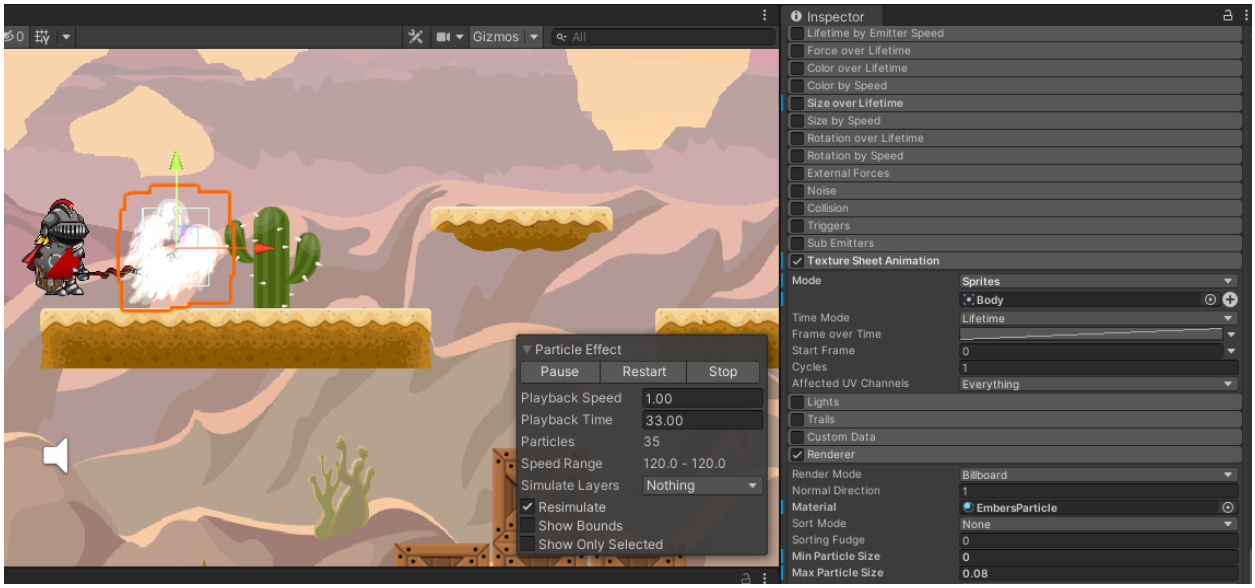
Platformun hareketi sırasıyla belirlenen 3 nokta üzerinden olmaktadır. Hedef noktalar, y ve z eksenleri sabit, x eksenleri artarak ilerleyen düzlem üzerinde bulunmaktadır. (Şekil 1)



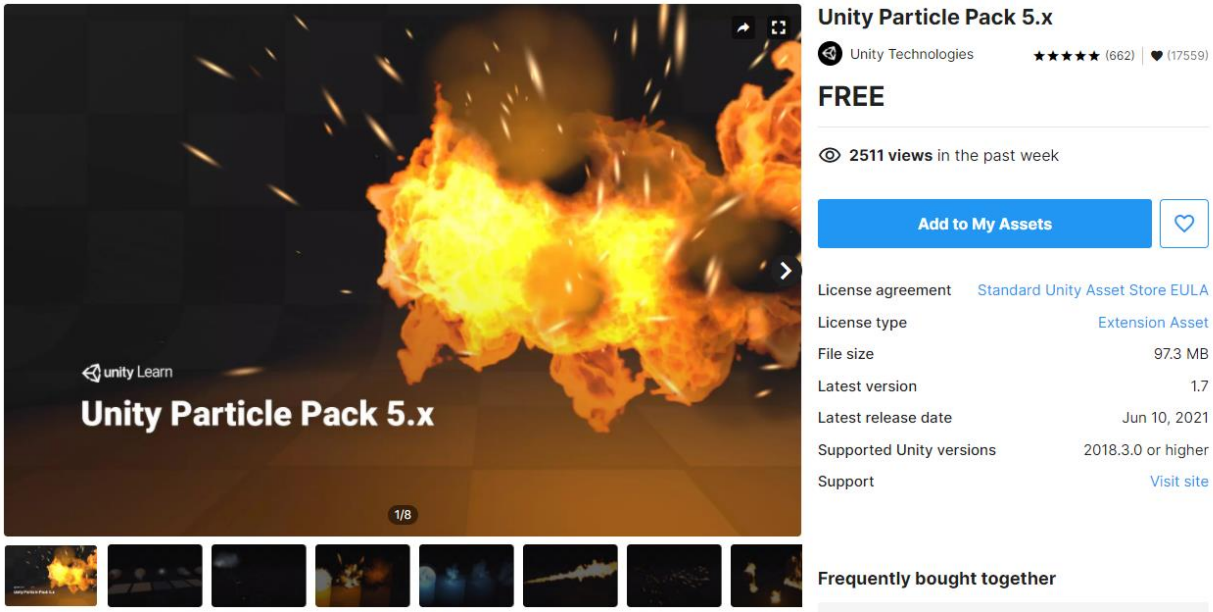
Şekil 1 Hareketli Platform

Düşman Ruhu Oluşturma (Particle System)

Düşman karakter, oyuncu tarafından öldürüldüğünde ortaya düşman ruhu çıkması için partikül sistemi kullanılmıştır. Kişisel tasarım katkısı olarak Şekil 2'de görülebileceği gibi öldürülen düşmanın pelerini partikül olarak belirlenmiştir. Unity Particle Pack 5.x (Şekil 3) desteği alınarak pelerinlere ruh görünümü verilmiştir.



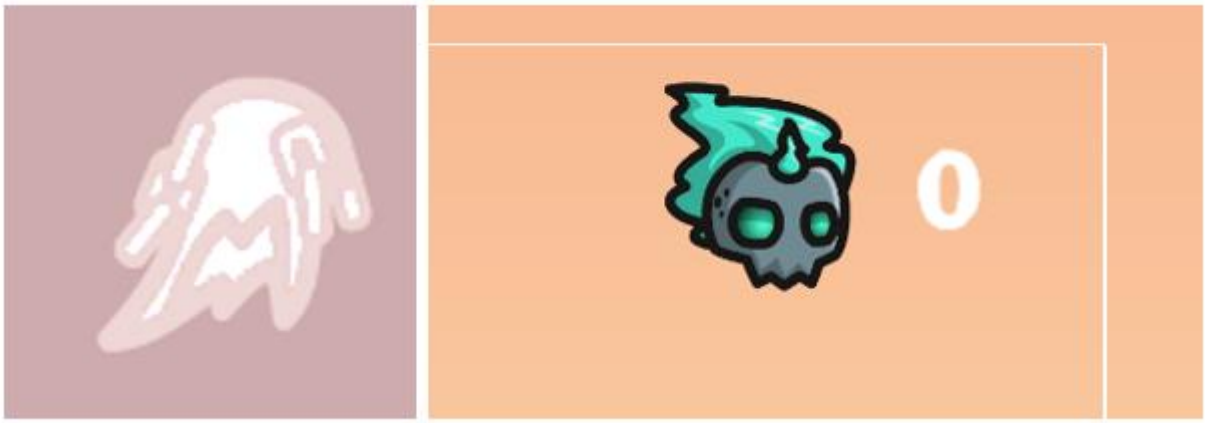
Şekil 2 Düşman Ruh Oluşumu



Şekil 3 Unity Particle Pack 5.x

Düşman Ruhu Alma

Oyunda öldürülen her düşman için ortaya çıkan ruh (Şekil 2) oyuncu tarafından alınabilmektedir. Oyundaki bir skor verisi olarak da değerlendirilebilecek olan bu veri Şekil 2'de görüldüğü gibi ekranın sağ üst bölümüne sabitlenmiştir. Her düşman öldürüldüğünde ortaya çıkan ruh toplanarak sayı artırılmaktadır.



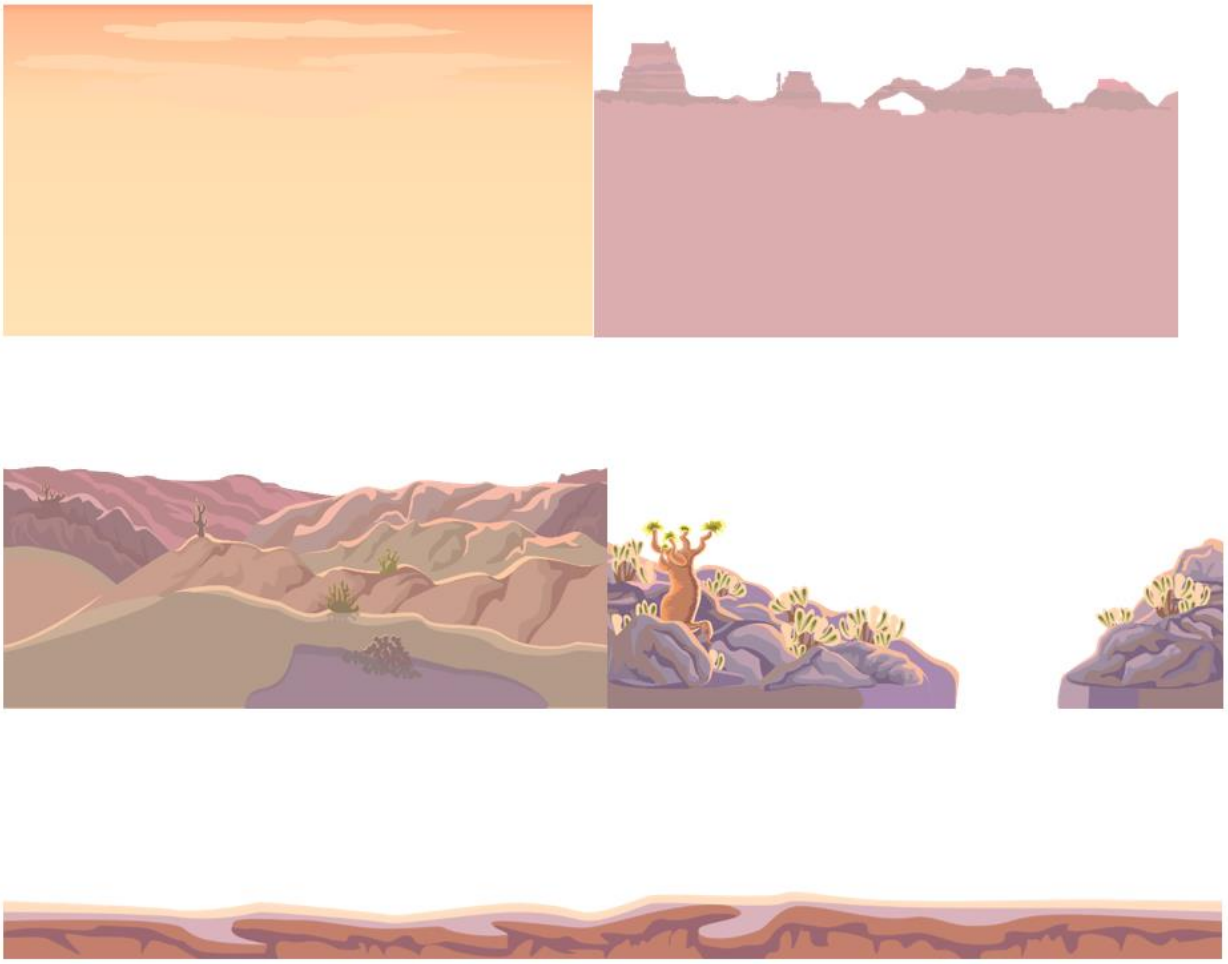
Şekil 4 Ruh Toplama Sistemi

Arkaplan Tasarımı

Arkaplan olarak çöl ortamı hissini derinleştiren https://craftpix.net/freebies/free-desert-scrolling-2d-game-backgrounds/ adresinde yer alan çöl görsellerinden yararlanılmıştır. Bu görseller 5 aşamalı olarak z ekseninde kaydırılmak üzere kullanılmıştır. Her aşamaya ait görsel Şekil 5'te görüldüğü gibidir.

Farklı görsellerin üst üste gelmesiyle oluşan ve hareket izlenimini veren görsel akışı "BgParallaxScroll.cs" isimli script dosyası ile kontrol edilmektedir.

Bu dosya oyuncunun hareketini izleyip arkaplan görselini ters yönde hareket etmesini sağlayacak şekilde kodlanmıştır.



Şekil 5 Arkaplan Görselleri

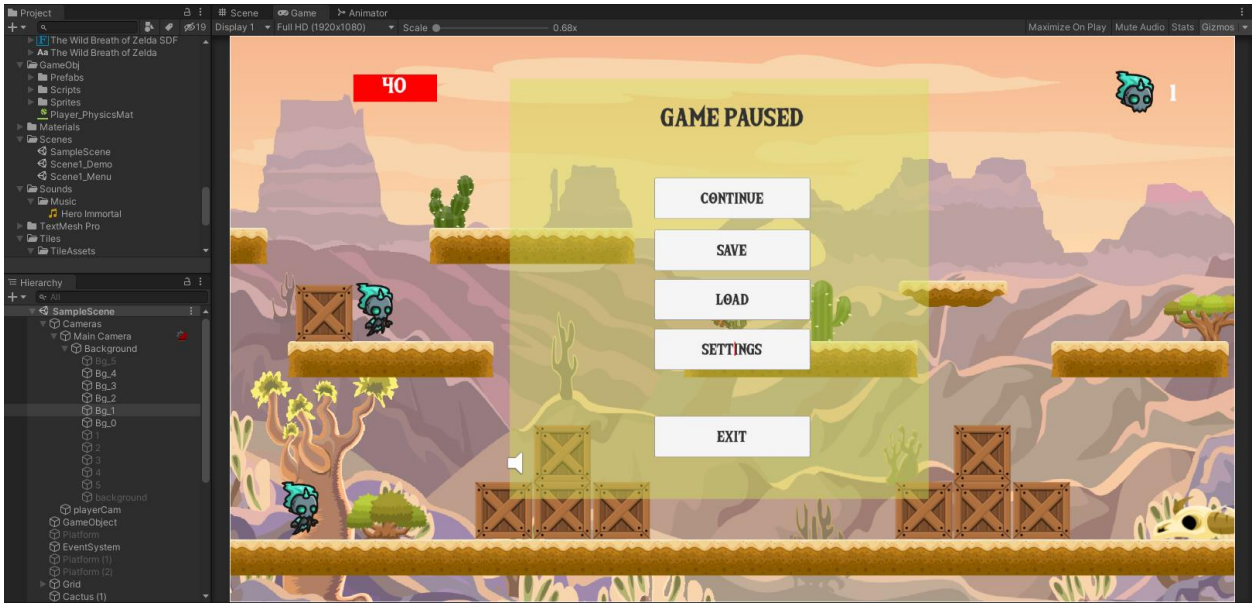
Menü – İç Menü

Oyunun girişinde oyuncuyu karşılayan ve isteğini alan ana menü tasarlanmıştır. Bunun yanında oyun içerisinde “P” veya “Esc” tuşlarıyla tetiklenen “InGameCanvas.cs” script dosyasındaki Update metodu ile sürekli kontrol edilen duraklama isteğine cevap verecek iç menü de bulunmaktadır.

Ana menü oyun ilk açıldığında kullanıcıyı karşılayan ve içerisinde Start, Load, Settings ve Exit seçenekleri bulunan butonlar karşılamaktadır. Şekil 6’da açılış sayfası görseli eklenmiştir.

Şekil 6 Ana Menü Ekranı

"P" veya "Esc" tuş kombinasyonu ile açılan iç menü görseli Şekil 7'de görülmektedir. Oyun içi menü açıldığında tüm hareketler durmaktadır. Oyun içi menü içeriğinde Continue, Save, Load, Settings ve Exit butonları bulunmaktadır.

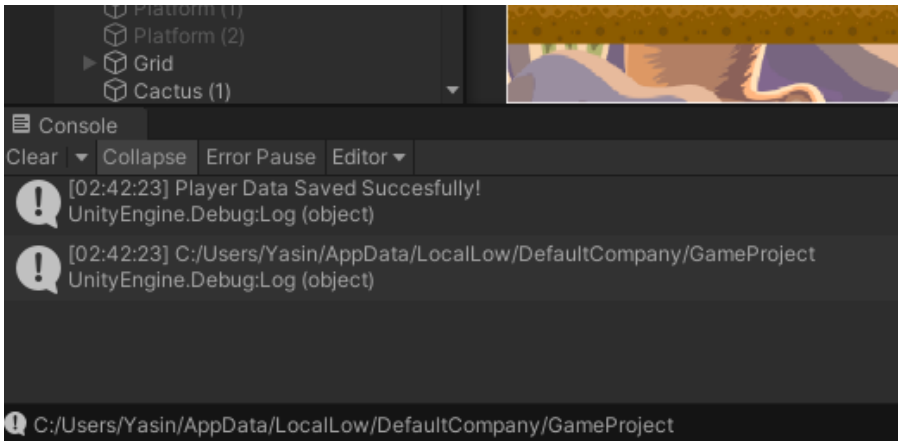


Şekil 7 Oyun İçi Menü Ekranı

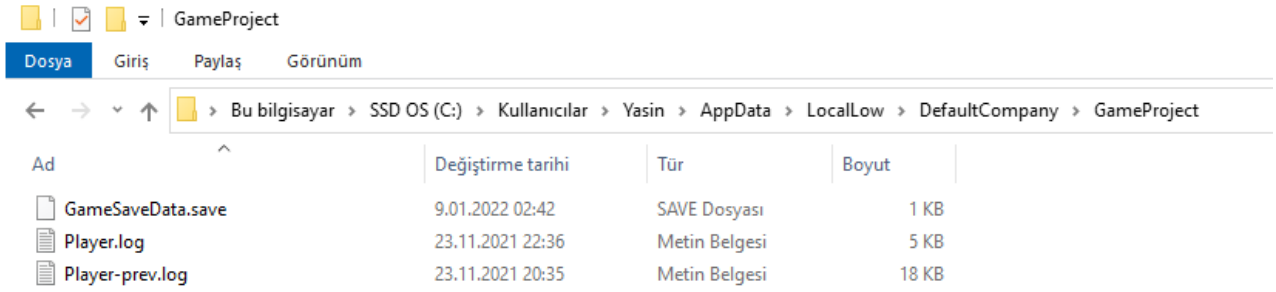
Kaydetme ve Kayıttan Geri Yükleme

Kayıt ve geri yükleme işlemleri için “SaveLoad.cs” dosyası ile kodlar yazılmıştır. Buna göre oyun klasöründe “/GameSaveData.save” ismiyle dosya kaydedilmiş olacaktır. “SaveLoad.cs” script dosyasının SaveData metoduyla oyuna ait veriler kaydedilirken, LoadData metodu ile kaydedilmiş olan oyun açılmaktadır.

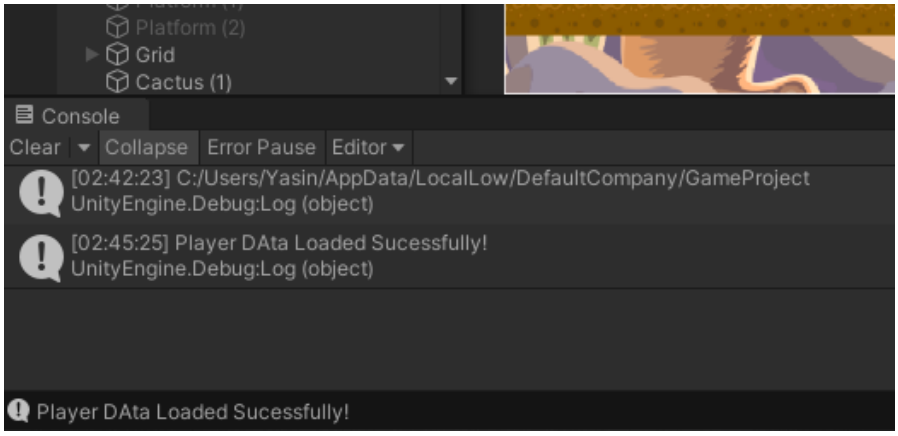
Save butonu sonrası çıktı Şekil8’de, kayıt dosyası Şekil 9’da, Load butonu sonrası çıktı Şekil 10‘da, oyun ekranı görüntüleri Şekil 11’de görülmektedir.



Şekil 8 Kayıt Butonu Sonrası Çıktı



Şekil 9 Kayıt Dosyası Dizin Görünümü

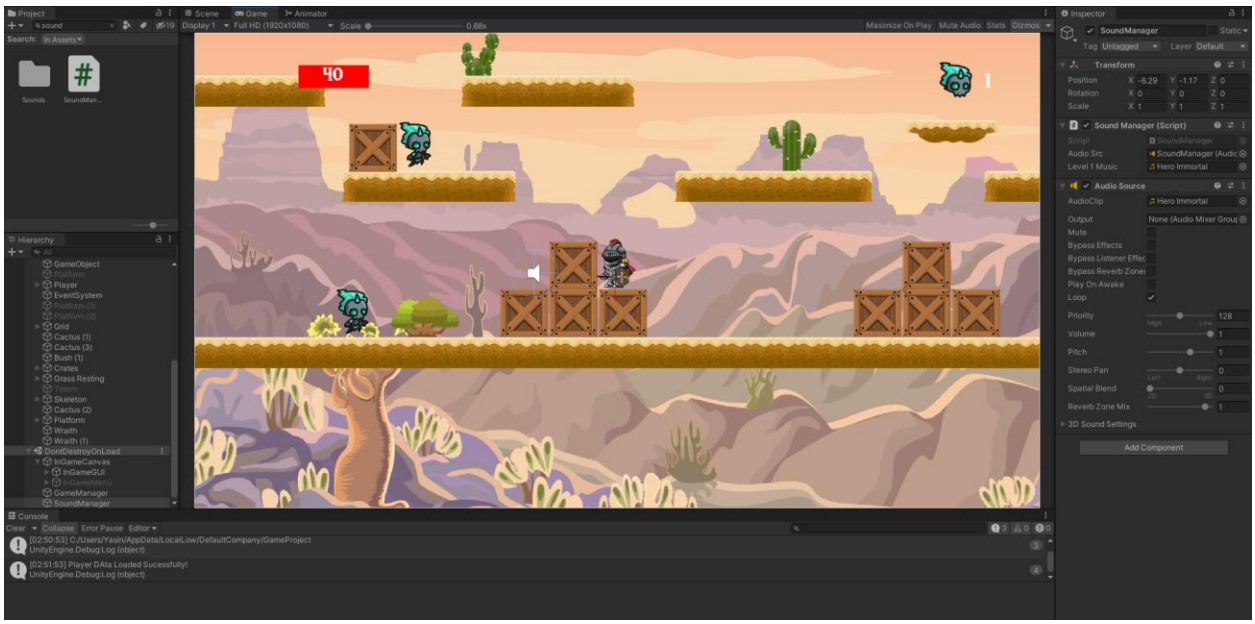


Şekil 10 Load Butonu Sonrası Çıktı

Şekil 11 Save - Lokasyon Değişikliği - Load

Ses/Müzik Ekleme

Ses veya müzik eklemek için SoundManager oyun nesnesi oluşturulmuştur. Eklenen oyun nesnesi üzerine bir de SoundManager.cs isimli script dosyası eklenmiştir. SoundManager oyun nesnesine, müzikleri oynatması için Audio Source isimli component eklenmiştir. Oyun ekranı Şekil 12'de görülmektedir.



Şekil 12 Müzik Ekleme

Kodlar

BgParallaxScroll.cs

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

public class BgParallaxScroll : MonoBehaviour
{
    [SerializeField] private float parallaxEffectMultiplier = 0.1f;
    [SerializeField] private float scrollSpeed = 0.01f;

    private Vector2 _savedOffset;
    private Renderer _renderer;

    private Rigidbody2D playerRB;

    //[SerializeField] private Vector2 playerVelocity;

    //Camera
    private Transform cameraTransform;
    private Vector3 lastCameraPosition;
    [SerializeField] private Vector3 deltaCamMovement;
```

```
    // Start is called before the first frame update
    void Start()
    {
        _renderer = GetComponent<Renderer>();
        _savedOffset = _renderer.material.mainTextureOffset;

        //playerRB =
GameObject.FindWithTag("Player").GetComponent<RigidBody2D>();
        //playerRB =
PlayerHealth.Instance.gameObject.GetComponent<Rigidbody2D>();
        //playerVelocity = playerRB.velocity;

        cameraTransform = Camera.main.transform;
        lastCameraPosition = cameraTransform.position;
    }
    // Update is called once per frame
    void LateUpdate()
    {
        deltaCamMovement = cameraTransform.position - lastCameraPosition;
        lastCameraPosition = cameraTransform.position;
        //playerVelocity = playerRB.velocity;

        float x=0f;
        float diffx=0f;

        diffx = ( parallaxEffectMultiplier*( Math.Sign(deltaCamMovement.x) *
(Time.deltaTime*scrollSpeed) ) );
        x = _savedOffset.x + diffx;
        x = Mathf.Repeat(x,1);
        _savedOffset = new Vector2(x, _savedOffset.y);
        _renderer.material.mainTextureOffset = _savedOffset;
    }
}
```

GameManagerSc.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using Cinemachine; // To make camera follow player

public class GameManagerSc : Singleton<GameManagerSc>
{
    [SerializeField] private GameObject PlayerPrefab;
    private GameObject PlayerGO;
    private PlayerHealth PlayerHealthSc;
```

```
    [SerializeField] private GameObject InGameCanvasPrefab;
    private GameObject InGameCanvas;
    private InGameCanvasSc InGameCanvasScript;

    private uint sth = 0;

    public static GameManagerSc instance;

    Scene currentScene;

    private bool startAnewGame=false;
    PlayerData PxData;

    private void Start()
    {
        startAnewGame=false;
        currentScene = SceneManager.GetActiveScene();

        //Instantiate Player
        PlayerGO = Instantiate(PlayerPrefab, new Vector3(0f,0f,0f),
Quaternion.identity);
        PlayerHealthSc = PlayerGO.GetComponent<PlayerHealth>();

        //Instantiate InGameCanvas
        InGameCanvas = Instantiate(InGameCanvasPrefab, new Vector3(0f,0f,0f),
Quaternion.identity);
        InGameCanvasScript = InGameCanvas.GetComponent<InGameCanvasSc>();


        if(PlayerGO != null)
        {
            PlayerHealthSc = PlayerGO.GetComponent<PlayerHealth>();
        }
        else{

            //PlayerGO = GameObject.Find("Player");
            PlayerGO = GameObject.FindWithTag("Player");
            if(PlayerGO == null)
                Debug.LogError("No Player Game Object in the Scene");
            else
            {
                Debug.Log("Player found in the Scene");
                PlayerHealthSc = PlayerGO.GetComponent<PlayerHealth>();
            }
        }

        if(currentScene.buildIndex == 0)
        {
            PlayerGO.SetActive(false);
            InGameCanvas.SetActive(false);
```

```
        }

    }

    //After Scene Load
    [RuntimeInitializeOnLoadMethod]
    static void OnRunTimeMethodLoad()
    {
        Debug.Log("After Scene Loaded");
    }

    void OnSceneLoaded(Scene scene, LoadSceneMode mode)
    {
        Debug.Log("OnSceneLoad " + scene.name);
        Debug.Log("Scene Mode" + mode);

        if(startAnewGame==false)
        {
            //Load Game
            PlayerHealthSc.setPlayerData(PxData);
            PlayerGO.transform.position = PlayerHealthSc.getPlayerPos();
        }
        else
        {
            PlayerGO.transform.position = new Vector3(0f,4f,0f);
        }

        InGameCanvasScript.SetPlayerGUIobjects();

        GameObject cineMachine = GameObject.FindWithTag("Cinemachine");
        if(cineMachine != null)
        {
            CinemachineVirtualCamera vcam =
cineMachine.GetComponent<CinemachineVirtualCamera>();
            vcam.Follow = PlayerGO.transform;

        }
        else
        {
            Debug.LogError("Not Found Cinemachine");
        }
    }
    // Start is called before the first frame update
    public void MainMenuStart()
    {
        startAnewGame=true;
        Debug.Log("Start Game");

        PlayerGO.SetActive(true);
        InGameCanvas.SetActive(true);
```

```
        SceneManager.LoadScene(1);

        SceneManager.sceneLoaded += OnSceneLoaded;
    }

    public void MainMenuSettingsUI()
    {
        //MainMenu_Default.SetActive(false);
        Debug.Log("Settings Game");
    }

    public void MainMenuLoad()
    {
        //MainMenu_Default.SetActive(false);
        PxData = SaveLoad.LoadData();

        startAnewGame = false;

        InGameCanvas.SetActive(true);
        Debug.Log("Load Game");

        SceneManager.LoadScene(PxData.getCurrentSceneId());
        SceneManager.sceneLoaded += OnSceneLoaded;
    }

    // Update is called once per frame
    public void MainMenuExit()
    {
        Debug.Log("Exit Game");
        Application.Quit();
    }
}
```

InGameCanvas.cs

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

public class InGameCanvasSc : Singleton<InGameCanvasSc>
{
    private bool isActive_InGameMenu = false;

    [SerializeField] private GameObject InGameGUI;
    [SerializeField] private GameObject InGameMenuObj;

    [SerializeField] private GameObject PlayerGO;
    [SerializeField] private PlayerHealth PlayerHealthSc;
```

```
    private void Start()
    {
        InGameGUI     =  this.gameObject.transform.GetChild(0).gameObject; //Get
First Child Game Object
        InGameMenuObj =  this.gameObject.transform.GetChild(1).gameObject;

        isActive_InGameMenu = false;
        InGameMenuObj.SetActive(isActive_InGameMenu);


        PlayerGO = GameObject.FindWithTag("Player");
        if(PlayerGO)
            PlayerHealthSc =PlayerGO.GetComponent<PlayerHealth>();
        else
            Debug.LogError("InGameCanvas - Start - Player Object cannot be
found");
    }

    public void SetPlayerGUIobjects()
    {
        if(PlayerGO == null && PlayerGO == null)
        {
             if(PlayerGO)
                 PlayerHealthSc =PlayerGO.GetComponent<PlayerHealth>();
        }

        if(InGameGUI == null)
        {
            InGameGUI     =  this.gameObject.transform.GetChild(0).gameObject;
//Get First Child Game Object
             InGameMenuObj =  this.gameObject.transform.GetChild(1).gameObject;
        }

        if(InGameGUI != null && PlayerGO != null)
        {
            GameObject HealthBarGO = InGameGUI.transform.GetChild(0).gameObject;
            GameObject HealthTextGO = InGameGUI.transform.GetChild(1).gameObject;
            GameObject SoulTextGO = InGameGUI.transform.GetChild(3).gameObject;
            PlayerHealthSc.setGUIObject(HealthBarGO, HealthTextGO, SoulTextGO);
        }
    }

    void Update(){

        if(Input.GetButtonDown("Cancel") || Input.GetKeyDown(KeyCode.P))
        {
            InGameMenu_Continue();
        }
    }
```

```
    public void InGameMenu_Continue()
    {
        if(isActive_InGameMenu==false){isActive_InGameMenu=true;
Time.timeScale=0f;}
        else{isActive_InGameMenu=false; Time.timeScale=1f;}
        InGameMenuObj.SetActive(isActive_InGameMenu);
    }

     public void InGameMenu_Save()
    {
        if(PlayerGO != null)
        {
            SaveLoad.SaveData(PlayerHealthSc.getPlayerData());

        }
        else{
            Debug.Log("Player cannot to save the game!");
        }

       Debug.Log(Application.persistentDataPath);
    }

    public void InGameMenu_Load()
    {
        if(PlayerGO != null)
        {
            PlayerHealthSc.setPlayerData(SaveLoad.LoadData());

        }
        else{
            Debug.Log("Player cannot to load the game!");
        }
    }

    public void InGameMenu_Settings()
    {

    }

     public void InGameMenu_Exit()
    {
        Debug.Log("Exit Game");
        Application.Quit();
    }
}
```

Platform2Dmove.cs

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

public class Platform2Dmove : MonoBehaviour
{

    [SerializeField] private List<GameObject> targets;
    [SerializeField] private int currentTarget, incrementDir=1;
    [SerializeField] private float moveSpeed=2.5f;

    // [SerializeField] private List<GameObject> ObjcetsOnPlatform;
    // Start is called before the first frame update
    void Start()
    {
        currentTarget = 0;

        foreach(GameObject target in targets)
        {
            if(target == null)
            {
                Debug.LogError("Platform target is null");
            }
        }
    }

     void FixedUpdate()
    {
        PlatformMove();
    }

    private void PlatformMove()
    {
        this.transform.position =
Vector3.MoveTowards(this.transform.position,targets[currentTarget].transform.posi
tion,moveSpeed*Time.deltaTime);

        float distance =
Vector3.Distance(this.transform.position,targets[currentTarget].transform.positio
n);

        if(distance < 0.25f)
        {
            currentTarget=nextTarget();
        }
    }
```

```
    private int nextTarget()
    {
        int tempTarget = currentTarget;

        if(tempTarget + incrementDir == targets.Capacity)
        {
            incrementDir=-1;
        }
        else if(tempTarget+incrementDir==-1)
        {
            incrementDir=1;
        }
        tempTarget+=incrementDir;
        return tempTarget;
    }

    void OnCollisionEnter2D (Collision2D col)
    {
        //Debug.Log(col.gameObject.name);
        //ObjcetsOnPlatform.Add(col.gameObject);
        col.gameObject.transform.SetParent(this.transform);
    }
    void OnCollisionExit2D (Collision2D col)
    {
        //ObjcetsOnPlatform.Remove(col.gameObject);
        col.gameObject.transform.SetParent(null);
    }
}
```

SaveLoad.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

public static class SaveLoad
{
    public static void SaveData(PlayerData PlayerSaveData)
    {
        //Debug.Log(Application.persistentDataPath);

        BinaryFormatter Bformatter = new BinaryFormatter();
        string path = Application.persistentDataPath + "/GameSaveData.save";

        FileStream fstream = new FileStream(path, FileMode.Create);

        PlayerData PxData = new PlayerData(PlayerSaveData);
```

```
        Bformatter.Serialize(fstream, PxData);
        fstream.Close();

        Debug.Log("Player Data Saved Succesfully!");
    }

    public static PlayerData LoadData()
    {
        string path = Application.persistentDataPath + "/GameSaveData.save";

        if(File.Exists(path))
        {
            BinaryFormatter Bformatter = new BinaryFormatter();
            FileStream fstream = new FileStream(path, FileMode.Open);

            PlayerData  Pxdata = Bformatter.Deserialize(fstream) as PlayerData;

            fstream.Close();
            Debug.Log("Player DAta Loaded Sucessfully!");

            return Pxdata;
        }
        else
        {
            Debug.Log("Error: Save file not found in path: "+ path);
            return null;
        }
    }
}
```

Singleton.cs

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

public abstract class Singleton<T> : MonoBehaviour where T: MonoBehaviour
{
    private static T instance;

    public static T Instance{
        get
        {
            if(instance == null)
            {
                instance = FindObjectOfType<T>();
                if(instance == null)
                {
```

```
                    GameObject obj = new GameObject();
                    obj.name = typeof(T).Name;
                    instance = obj.AddComponent<T>();
                }
            }
            return instance;
        }
    }

    public virtual void Awake()
    {
        if(instance == null)
        {
            instance = this as T;
            DontDestroyOnLoad(this.gameObject);
        }
        else{
            Destroy(gameObject);
        }
    }
}
```

SoundManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SoundManager : Singleton<SoundManager>
{
    [SerializeField] private AudioSource AudioSrc;
    [SerializeField] private AudioClip Level1Music;

    private Scene scene;

    // Start is called before the first frame update
    void Start()
    {
        scene = SceneManager.GetActiveScene();

        if(scene.buildIndex == 1){

            //AudioSrc = this.GetComponent<AudioSource>();
            if(AudioSrc != null)
            {
                playBackgroundMusic(Level1Music);
            }
        }
```

```
    }
    // Update is called once per frame
    void Update()
    {

    }
    void playBackgroundMusic(AudioClip audioClp)
    {
        this.AudioSrc.clip = Level1Music;
        AudioSrc.Play();
        AudioSrc.loop = true;
    }
}
```

Kaynakça

1) "2d Fantasy Knıght Free Character Sprite" isimli şövalye karakterine linki verilen site üzerinden ulaşılabilir: https://craftpix.net/freebies/2d-fantasy-knight-free-sprite-sheets/

2) "Free Desert Platformer Tıleset"e linki verilen site üzerinden ulaşılabilir: https://www.gameart2d.com/uploads/3/0/9/1/30917885/deserttileset.zip

3) "The Wild Breath of Zelda" yazı tipine linki verilen site üzerinden ulaşılabilir: http://www.dafont.com/the-wild-breath-of-zelda.font

4) Arkaplan görsellerine linki verilen site üzerinden ulaşılabilir: https://craftpix.net/freebies/free-desert-scrolling-2d-game-backgrounds/

5) Wraith isimli düşman karakterine linki verilen site üzerinden ulaşılabilir: https://craftpix.net/freebies/free-wraith-tiny-style-2d-sprites/?num=1&count=3&sq=wraith&pos=2

6) Unity Particle Pack 5.x'e linki verilen site üzerinden ulaşılabilir: https://assetstore.unity.com/packages/essentials/asset-packs/unity-particle-pack-5-x-73777

7) Hero Immortal isimli oyun müziğine linki verilen site üzerinden ulaşılabilir: https://opengameart.org/content/hero-immortal