

BİLGİSAYAR GRAFİKLERİ

Seher KARAGÖZ

Benim geliştirdiğim **Balık Macerası** adlı oyun, oyuncunun balığı kontrol ettiği ve denizdeki engellerden kaçmaya çalıştığı bir aksiyon oyunu. Amacım, oyuncunun ne kadar uzun süre hayatta kalabileceğini görmek ve yüksek skor yapmasını sağlamak. Oyun ilerledikçe zorluk artıyor, engeller hızlanıyor ve daha sık spawn oluyor.

Oyuncu, balığı dikey olarak yönlendiriyor ve engellere çarpmadan ilerlemeye çalışıyor. Eğer engellere çarparsa, oyun sona eriyor. Oyun başladığında engeller daha yavaş geliyor ama zamanla hızlanıyor. Ayrıca, oyuncu kazandığı puanlarla yüksek skor elde edebiliyor.

Oyun, sürekli olarak refleksleri test eden ve strateji gerektiren bir yapıya sahip. Her yeni oyun, oyuncuya daha zorlu bir deneyim sunuyor. Benim amacım, oyuncuların sürekli olarak daha iyi bir skor yapabilmesi ve oyun sonunda elde edilen yüksek skorlarla motive olmaları.

BackgroundMusic

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

// Bu script, arka plan müziğinin sahneler arasında kesintisiz devam etmesini sağlar

// ve birden fazla müzik çalar oluşmasını engeller.

public class BackgroundMusic : MonoBehaviour

{

    // Singleton desenini uygulamak için statik bir BackgroundMusic referansı.

    private static BackgroundMusic backgdMusic;

    void Awake()

    {

        // Eğer BackgroundMusic sınıfından daha önce bir örnek oluşturulmamışsa

        if (backgdMusic == null)

        {

            // Bu örneği statik referansa atar.

            backgdMusic = this;

            // Bu GameObject'in yeni bir sahne yüklendiğinde yok edilmemesini sağlar.

            DontDestroyOnLoad(backgdMusic);

        }

        else

        {

            // Eğer zaten bir örnek varsa, bu GameObject'i yok eder

            // böylece müzik çalarlar çoğalmaz.

            Destroy(gameObject);

        }

    }

}

BackMenu

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

// Bu script, kullanıcıyı bir önceki sahneye geri döndüren bir fonksiyon içerir.

public class BackMenu : MonoBehaviour

{

    // Kullanıcının bir önceki sahneye dönmesini sağlar.

    public void BackGame()

    {

        // Aktif olan sahnenin indeksini alır ve bir önceki sahneyi yükler.

        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);

    }

}

CameraMovement

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

// Bu script, kameranın yatay eksende belirli bir hızla hareket etmesini sağlar.

public class CameraMovement : MonoBehaviour

{

    // Kameranın hareket hızı (kullanıcı tarafından ayarlanabilir).

    public float cameraSpeed;

    // Update metodu her bir frame'de bir kez çağrılır.

    void Update()

    {

        // Kameranın pozisyonunu yatay eksende (x ekseni) hız değerine göre günceller.

        // Time.deltaTime, frame'e bağlı olmayan sabit bir hız sağlar.

        transform.position += new Vector3(cameraSpeed \* Time.deltaTime, 0, 0);

    }

}

GameOver

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

// Bu script, oyuncunun oyunda kaybettiği durumu yönetir ve oyun sonu panelini açar.

public class GameOver : MonoBehaviour

{

    // Oyun sonu panelini temsil eden GameObject.

    public GameObject gameOverPanel;

    // Oyunun bitip bitmediğini kontrol eden bir değişken.

    public bool isGameOver = false;

    // Update metodu her bir frame'de bir kez çağrılır.

    void Update()

    {

        // Oyuncu etiketi ("Player") olan bir GameObject sahnede yoksa (null ise).

        if (GameObject.FindGameObjectWithTag("Player") == null)

        {

            // Oyun sonu panelini aktif hale getirir.

            gameOverPanel.SetActive(true);

            // Oyunun bittiğini belirtmek için değişkeni true yapar.

            isGameOver = true;

        }

    }

    // Oyunu yeniden başlatan fonksiyon.

    public void Restart()

    {

        // Mevcut sahneyi tekrar yükler (oyunu yeniden başlatır).

        SceneManager.LoadScene(SceneManager.GetActiveScene().name);

    }

}

HealthManager

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

// Bu script, oyuncunun sağlık durumunu yönetir ve kullanıcı arayüzündeki kalp ikonlarını günceller.

public class HealthManager : MonoBehaviour

{

    // Oyuncunun mevcut sağlık durumu (başlangıçta 3 can).

    public static int health = 3;

    // Sağlık durumunu göstermek için kullanılan kalp görselleri.

    public Image[] hearts;

    // Tam dolu bir kalbi temsil eden sprite.

    public Sprite fullHeart;

    // Boş bir kalbi temsil eden sprite.

    public Sprite emptyHeart;

    // Oyun başladığında sağlık durumunu varsayılan değere (3 can) ayarlar.

    void Awake()

    {

        health = 3;

    }

    // Her bir frame'de sağlık durumunu kontrol eder ve görselleri günceller.

    void Update()

    {

        // Tüm kalpleri boş olarak ayarlar.

        foreach (Image img in hearts)

        {

            img.sprite = emptyHeart;

        }

        // Oyuncunun mevcut sağlık durumuna göre dolu kalpleri günceller.

        for (int i = 0; i < health; i++)

        {

            hearts[i].sprite = fullHeart;

        }

    }

}

LoopingBackground

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

// Bu script, arka planın sürekli olarak kaymasını sağlar ve hızını artırır.

public class LoopingBackground : MonoBehaviour

{

    // Arka planın kayma hızı.

    public float backgroundSpeed;

    // Arka planın materyali üzerinden erişilen Renderer bileşeni.

    public Renderer backgroundRenderer;

    // Update metodu her bir frame'de bir kez çağrılır.

    void Update()

    {

        // Arka planın dokusunun kaymasını sağlar (x ekseninde).

        backgroundRenderer.material.mainTextureOffset += new Vector2(backgroundSpeed \* Time.deltaTime, 0f);

        // Eğer arka planın hızı 2'den küçükse, hızı zamanla artırır.

        if (backgroundSpeed < 2)

        {

            backgroundSpeed += 0.1f \* Time.deltaTime;

        }

    }

}

MainMenu

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

// Bu script, ana menüdeki oyun başlatma fonksiyonunu yönetir.

public class MainMenu : MonoBehaviour

{

    // Oyun başlatıldığında bir sonraki sahneye geçiş yapar.

    public void PlayGame()

    {

        // Aktif sahnenin indeksini alır ve bir sonraki sahneyi yükler.

        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

    }

}

Obstacle

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Obstacle : MonoBehaviour

{

    // Oyuncu nesnesine referans.

    private GameObject player;

    // Start metodu, ilk frame'den önce bir kez çağrılır.

    void Start()

    {

        // "Player" etiketiyle etiketlenmiş oyuncu nesnesini bulur.

        player = GameObject.FindGameObjectWithTag("Player");

    }

    // Collider2D nesnesi ile çarpışma tespit edildiğinde çağrılır.

    private void OnTriggerEnter2D(Collider2D collision)

    {

        // Eğer çarpışan nesne "Border" etiketi taşıyorsa

        if (collision.tag == "Border")

        {

            // Bu engeli yok eder.

            Destroy(this.gameObject);

        }

        // Eğer çarpışan nesne "Player" etiketi taşıyorsa

        else if (collision.tag == "Player")

        {

            // Eğer oyuncunun sağlık durumu sıfırsa, oyuncu nesnesini yok eder.

            if (HealthManager.health <= 0)

            {

                Destroy(player.gameObject);

            }

            else

            {

                // Aksi takdirde, oyuncunun sağlığını bir azaltır.

                HealthManager.health--;

            }

        }

    }

}

PauseMenu

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

// Bu script, oyun duraklatma ve devam ettirme işlevselliklerini ve ana menüye dönme işlemlerini yönetir.

public class PauseMenu : MonoBehaviour

{

    // Duraklatma menüsünü temsil eden GameObject.

    [SerializeField] GameObject pauseMenu;

    // Oyun duraklatıldığında çağrılır.

    public void Pause()

    {

        // Duraklatma menüsünü aktif hale getirir.

        pauseMenu.SetActive(true);

        // Oyun zamanını durdurur (oyun duraklatılır).

        Time.timeScale = 0f;

    }

    // Oyun devam ettirildiğinde çağrılır.

    public void Resume()

    {

        // Duraklatma menüsünü kapatır.

        pauseMenu.SetActive(false);

        // Oyun zamanını normal akışına döndürür.

        Time.timeScale = 1f;

    }

    // Ana menüye dönmek için çağrılır, belirtilen sahneyi yükler.

    public void Home(int sceneID)

    {

        // Oyun zamanını normal akışına döndürür.

        Time.timeScale = 1f;

        // Belirtilen sahneyi yükler (ana menü gibi).

        SceneManager.LoadScene(sceneID);

    }

}

Player

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Player : MonoBehaviour

{

    // Oyuncunun hareket hızını belirleyen değişken.

    public float playerSpeed;

    // Oyuncunun Rigidbody2D bileşenine referans.

    private Rigidbody2D rb;

    // Oyuncunun hareket yönünü temsil eden vektör.

    private Vector2 playerDirection;

    // Start metodu, ilk frame'den önce bir kez çağrılır.

    void Start()

    {

        // Rigidbody2D bileşenini alır.

        rb = GetComponent<Rigidbody2D>();

        // Oyun hızı arttırma coroutine'ini başlatır.

        StartCoroutine("IncreaseGameSpeed");

    }

    // Update metodu her frame'de bir kez çağrılır.

    void Update()

    {

        // Kullanıcının dikey hareket girişini alır.

        float directionY = Input.GetAxisRaw("Vertical");

        // Yönü normalize eder ve oyuncu hareketini belirler (yalnızca dikey).

        playerDirection = new Vector2(0, directionY).normalized;

    }

    // FixedUpdate metodu, sabit aralıklarla çalışır ve fiziksel hesaplamalar için kullanılır.

    void FixedUpdate()

    {

        // Rigidbody2D'yi oyuncu yönü ve hızına göre hareket ettirir.

        rb.velocity = new Vector2(0, playerDirection.y \* playerSpeed);

    }

    // Coroutine, oyun hızını zamanla arttırmak için kullanılır.

    IEnumerator IncreaseGameSpeed()

    {

        // Sonsuz bir döngüde 10 saniyede bir çalışır.

        while (true)

        {

            // 10 saniye bekler.

            yield return new WaitForSeconds(10);

            // Eğer oyuncu hızının üst sınırına ulaşılmadıysa, hızı arttırır.

            if (playerSpeed < 8)

            {

                playerSpeed += 0.9f;

            }

            // Engelleri spawn etme aralığını azaltır.

            GameObject.Find("Spawn Point").GetComponent<SpawnObstacles>().timeBetweenSpawn -= 0.1f;

        }

    }

}

ScoreMenager

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using TMPro;

public class ScoreManager : MonoBehaviour

{

    // Skoru göstermek için kullanılan TextMeshPro bileşeni.

    public TextMeshProUGUI scoreText;

    // En yüksek skoru göstermek için kullanılan TextMeshPro bileşeni.

    public TextMeshProUGUI highScoreText;

    // Mevcut skoru tutan değişken.

    private float score;

    // Update metodu her frame'de bir kez çağrılır.

    void Update()

    {

        // Eğer oyuncu hala sahnede varsa

        if (GameObject.FindGameObjectWithTag("Player") != null)

        {

            // Skoru zamanla arttırır.

            score += 1 \* Time.deltaTime;

            // Skoru ekran üzerinde gösterir, tam sayıya yuvarlanır.

            scoreText.text = ((int)score).ToString();

        }

        // Eğer mevcut skor, en yüksek skordan büyükse

        if (score > PlayerPrefs.GetFloat("HighScore", 0))

        {

            // En yüksek skoru PlayerPrefs'e kaydeder.

            PlayerPrefs.SetFloat("HighScore", ((int)score));

        }

    }

    // Start metodu, ilk frame'den önce bir kez çağrılır.

    void Start()

    {

        // En yüksek skoru günceller.

        updatehighscore();

    }

    // Yüksek skorun ayarlanması için kullanılan metod, şu an yorum satırına alınmış.

    void sethighscore()

    {

        // Eğer skor daha büyükse, yüksek skoru ayarlamayı sağlar.

        // if(score > PlayerPrefs.GetFloat("HighScore", 0))

        //{

        //    PlayerPrefs.SetFloat("HighScore",score);

        //}

        // Yüksek skoru metin olarak gösterir.

        // highScoreText.text = PlayerPrefs.GetFloat("HighScore", 0).ToString();

    }

    // En yüksek skoru ekran üzerinde gösterir.

    void updatehighscore()

    {

        highScoreText.text = $"BEST : {PlayerPrefs.GetFloat("HighScore", 0)}";

    }

}

SoundMenager

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class SoundManager : MonoBehaviour

{

    // Ses seviyesini ayarlamak için kullanılan Slider.

    [SerializeField] Slider volumeSlider;

    // Start metodu, ilk frame'den önce bir kez çağrılır.

    void Start()

    {

        // Eğer PlayerPrefs'te müzik sesi seviyesi bilgisi varsa

        if (PlayerPrefs.HasKey("musicVolume"))

        {

            // Varsayılan olarak müzik ses seviyesini 1 (tam) yapar.

            PlayerPrefs.SetFloat("musicVolume", 1);

            Load(); // Kaydedilen sesi yükler.

        }

        else

        {

            // PlayerPrefs'te müzik sesi seviyesi kaydedilmemişse, yükler.

            Load();

        }

    }

    // Ses seviyesini Slider'dan alınan değere göre değiştirir.

    public void ChangeVolume()

    {

        // Slider değeriyle ses seviyesini ayarlar.

        AudioListener.volume = volumeSlider.value;

        // Yeni ses seviyesini kaydeder.

        Save();

    }

    // Kaydedilen müzik ses seviyesini slider'a yükler.

    private void Load()

    {

        // PlayerPrefs'ten kaydedilen müzik ses seviyesini slider'a atar.

        volumeSlider.value = PlayerPrefs.GetFloat("musicVolume");

    }

    // Ses seviyesini PlayerPrefs'e kaydeder.

    private void Save()

    {

        // Müzik ses seviyesini PlayerPrefs'e kaydeder.

        PlayerPrefs.SetFloat("musicVolume", volumeSlider.value);

    }

}

SpawnObstacle

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SpawnObstacles : MonoBehaviour

{

    // Spawn edilecek engel nesnesi.

    public GameObject obstacle;

    // Engellerin spawn edilebileceği maksimum ve minimum X ve Y koordinatları.

    public float maxX;

    public float maxY;

    public float minX;

    public float minY;

    // Engeller arasındaki zaman aralığını belirler.

    [HideInInspector]

    public float timeBetweenSpawn;

    // Son engel spawn edilen zaman.

    private float spawnTime;

    // Update metodu her frame'de bir kez çağrılır.

    void Update()

    {

        // Eğer geçerli zaman, spawn edilen engel zamanından daha büyükse

        if (Time.time > spawnTime)

        {

            // Yeni engel spawn eder.

            Spawn();

            // Spawn edilme zamanını günceller.

            spawnTime = Time.time + timeBetweenSpawn;

        }

    }

    // Yeni engel spawn eden metod.

    void Spawn()

    {

        // X ve Y koordinatları için rastgele değerler seçer.

        float randomX = Random.Range(minX, maxX);

        float randomY = Random.Range(minY, maxY);

        // Engeli belirli bir pozisyonda ve dönüşte instantiate eder.

        Instantiate(obstacle, transform.position + new Vector3(randomX, randomY, 0), transform.rotation);

    }

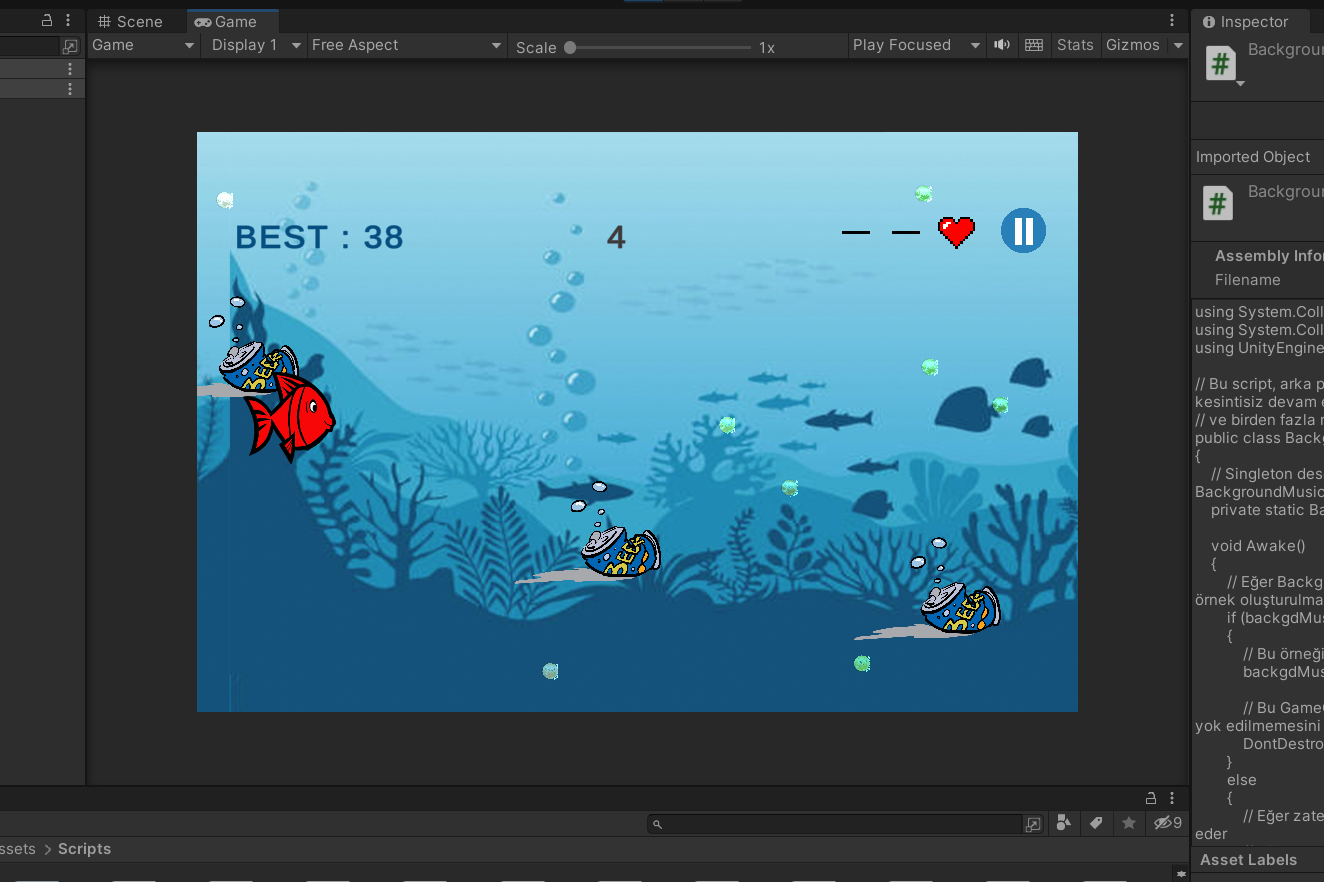
}



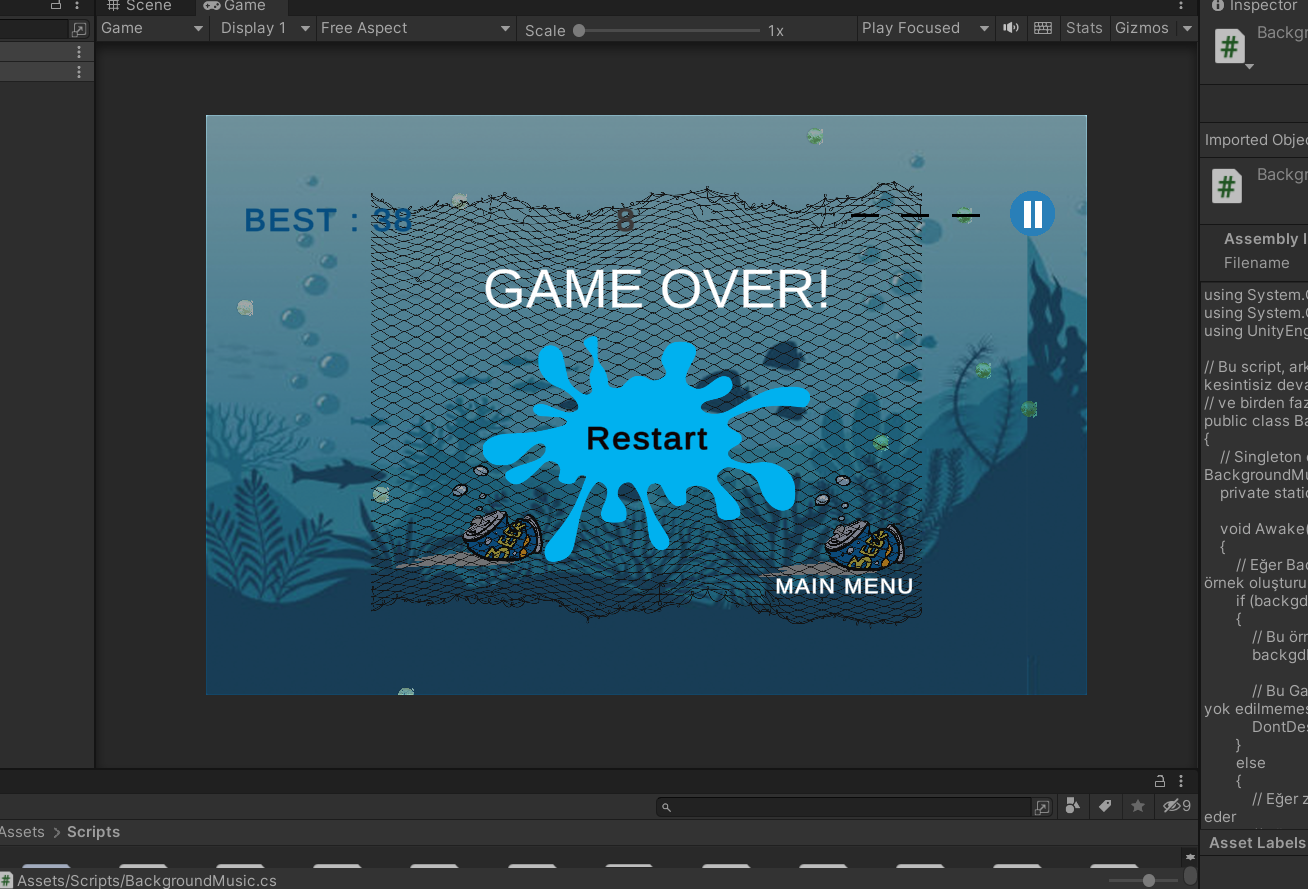
Oyun ana ekranı



Ayarlar menüsü



Oyun akışı



Oyun sonlandırma ekranı