KODLAMA ÖĞRENME MACERASI İLE GAMİFİCATİON

FET310 OYUN TASARIMI VE PROGRAMLAMA DERSİ

FİNAL RAPORU

YAZILIM MÜHENDİSLİĞİ

ŞULE KARTAL

22040301049

24-25 BAHAR DÖNEMİ

1.ÖZET

Unity oyun motoru kullanılarak geliştirilen, 3. şahıs kamera perspektifine sahip, eğitici ve etkileşimli bir oyun projesidir. Oyuncu, bir karakteri yön tuşlarıyla yönlendirerek, çeşitli tuzaklarla dolu parkurlarda ilerler. Oyunun temel amacı, oyuncunun hem fiziksel beceri hem de algoritmik düşünme yeteneğini kullanarak belirli noktalarda karşısına çıkan kodlama temelli soruları çözmesidir. Her doğru cevap, bir engelin veya tuzağın durmasına ya da ortadan kalkmasına neden olurken, yanlış cevap verildiğinde engeller aktif kalır ve oyuncunun ilerleyişi zorlaşır. Teknik olarak oyun, birden fazla sahneye sahip olup; UI sistemleri, olay tetikleyiciler (trigger), hareketli tuzaklar ve quiz tabanlı ilerleme mantığıyla oluşturulmuştur. Kodlamalar C# diliyle yazılmış, tüm UI yapısı Canvas sistemiyle düzenlenmiştir.Kod Koşusu, oyunculara hem refleks geliştirme hem de kodlama mantığını kavratma amacıyla kurgulanmış özgün ve öğretici bir projedir.

2.Hedef ve Amaç

Bu oyunun temel hedefi, oyunculara kodlama mantığını eğlenceli ve etkileşimli bir ortamda öğretmektir. Oyuncu, fiziksel reflekslerini kullanarak engelli parkurlarda ilerlerken, karşısına çıkan soruları çözerek algoritmik düşünme becerilerini geliştirir. Oyun boyunca karşılaşılan hareketli tuzaklar, engeller ve bilgi testleri sayesinde hem problem çözme yetenekleri hem de temel programlama kavramlarına olan hâkimiyetleri artar. Oyuncunun oyun içindeki ilerlemesi, doğru cevaplar vererek tuzakları etkisiz hale getirir. Amacı ise oyunculara hem eğlenerek öğrenebilecekleri hem de yazılım düşüncesini geliştirebilecekleri bir dijital öğrenme ortamı sunmaktır. Bu bağlamda oyun; değişkenler, koşullar, operatörler gibi yazılım temellerini örnek sorularla pekiştirmeyi ve bu bilgileri uygulamalı olarak oyun içinde kullandırmayı amaçlar. Oyuncular, düşme–yeniden başlama mekanikleriyle hatalarından öğrenme fırsatı elde ederken, doğru sorularla ilerleyerek bir başarı ve ödül hissi yaşar.

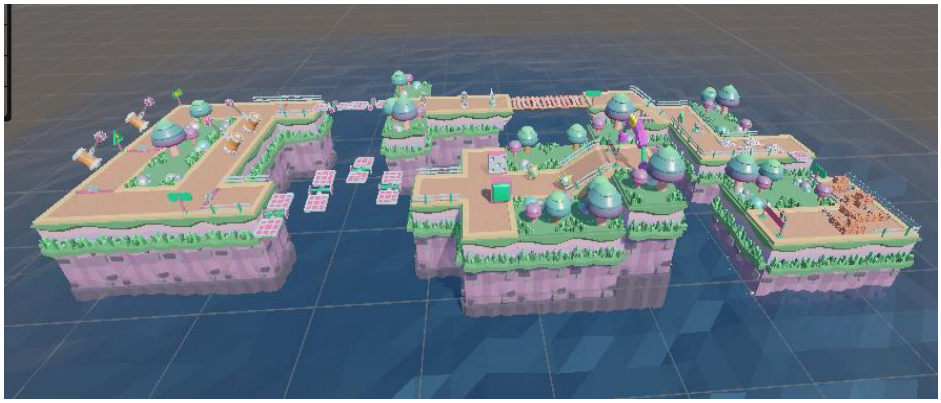
3.Levels

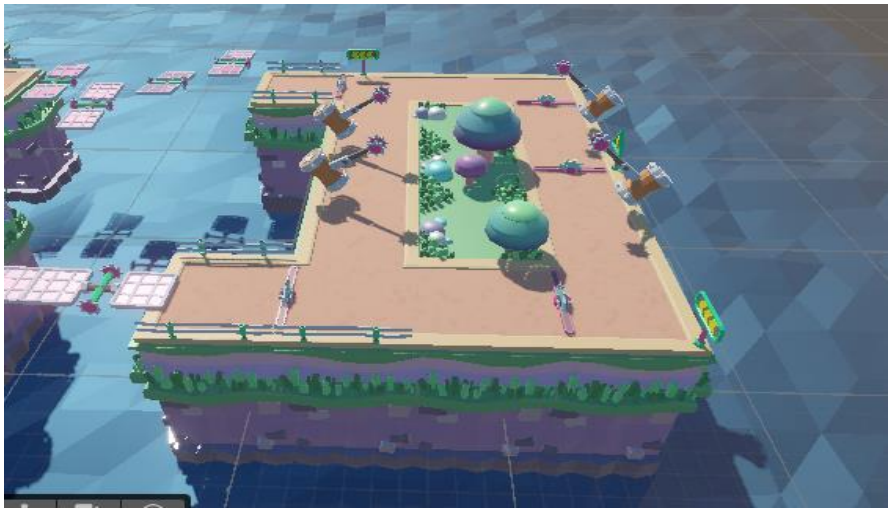
Oyuncu karşısına çıkan her engeli aştıkça yeni engeller ile karşılaşır ve çıkan engele göre önünde bir soru belirir. Bildiği sürece yoluna devam eder ve bitiş çizgisine ulaşmaya çalışır.

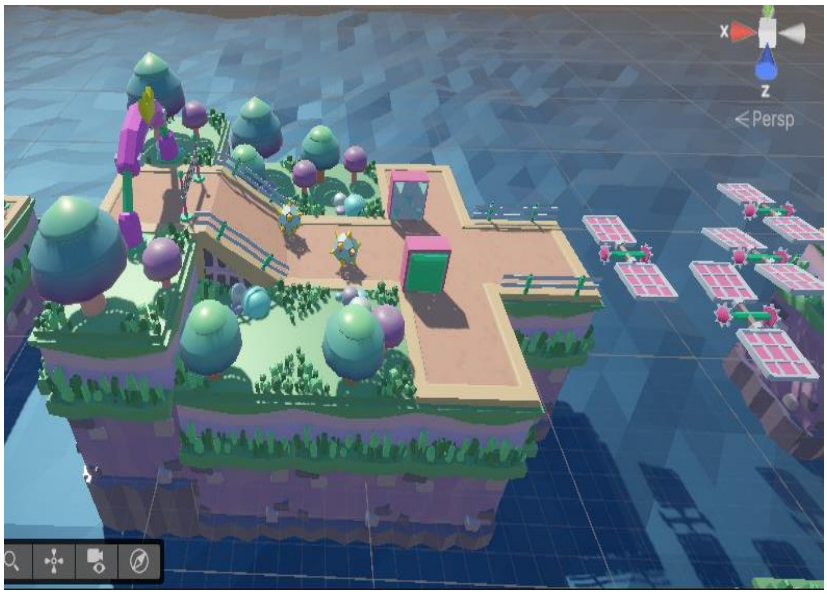
4.Asset Listesi

Asset olarak sadece oyun alanını ve karakteri kullandım

4.1.Oyun Alanı







4.2.Karakter

5.Kodlar

5.1 MainMenuManager.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenuManager : MonoBehaviour
{
    public GameObject YardimPanel;

    public void OyunaBasla()
    {
        SceneManager.LoadScene("SampleScene"); // kendi oyun sahnenin adı
    }
```

```
    public void YardimGoster()
    {
        YardimPanel.SetActive(true);
    }

    public void YardimKapat()
    {
        YardimPanel.SetActive(false);
    }
}
```

5.2.ObstacleTrigger.cs

```
using UnityEngine;

/// <summary>
/// Bu script, oyuncu belirli bir nesneye çarptığında quiz panelini açar.
/// Doğru cevap verilirse belirli nesneler devre dışı bırakılır.
/// </summary>
public class ObstacleTrigger : MonoBehaviour
{
    // Quiz sistemi referansı
    public QuizManager quizManager;

    // Gösterilecek soru
    [TextArea] public string question;

    // Seçenekler
    public string[] options;

    // Doğru cevap
    public string correctAnswer;

    // Doğru cevap verilirse devre dışı bırakılacak nesneler
    public GameObject[] objectsToDisableOnCorrect;

    // Oyuncu tetikledi mi kontrolü
    private bool triggered = false;

    // Oyuncu bu objeye çarptığında çağrılır
    private void OnTriggerEnter(Collider other)
    {
        // Daha önce tetiklendiyse veya çarpan oyuncu değilse çık
        if (triggered || !other.CompareTag("Player")) return;

        // Artık tetiklenmiş kabul edilir
        triggered = true;

        // Quiz'i başlat
        quizManager.ShowQuestion(
            question,
            options,
            correctAnswer,
            null, // tuzaklar (şu anda kullanılmıyor)
            objectsToDisableOnCorrect // doğru cevapta devre dışı
bırakılacaklar
        );
    }
}
```

5.3.HareketliTuzak.cs

```
using UnityEngine;

/// <summary>
/// Bu script, nesnenin yukarı-aşağı (Y ekseninde) ileri geri salınım yapmasını
sağlar.
/// Hareket durdurulabilir yapıdadır.
/// </summary>
public class HareketliTuzak : MonoBehaviour
{
    // Tuzak hareket hızı
    public float speed = 2f;

    // Tuzak hareket mesafesi (toplam yukarı-aşağı salınım)
    public float distance = 1f;

    // Başlangıç pozisyonunu tutar
    private Vector3 startPos;

    // Hareket aktif mi değil mi kontrolü
    private bool isActive = true;

    void Start()
    {
        // Nesneye ait başlangıç pozisyonu kaydedilir
        startPos = transform.position;
    }

    void Update()
    {
        // Eğer hareket durdurulduysa işlem yapma
        if (!isActive) return;

        // PingPong: Belirtilen aralıkta ileri-geri hareket sağlar
        // offset: pozisyon değişimi (yukarı-aşağı salınım)
        float offset = Mathf.PingPong(Time.time * speed, distance) - (distance
/ 2f);

        // Yeni pozisyonu uygula (sadece Y ekseni değişir)
        transform.position = startPos + new Vector3(0f, offset, 0f);
    }

    /// <summary>
    /// Bu fonksiyon çağrıldığında tuzak hareketi durur.
    /// </summary>
    public void HareketiDurdur()
    {
        isActive = false;
    }
}
```

5.4.TuzakTrigger.cs

```
using UnityEngine;

/// <summary>
/// Oyuncu bu trigger'a girdiğinde quiz başlatılır.
/// Oyuncu doğru cevabı verirse belirli tuzaklar devre dışı bırakılır.
/// </summary>
public class TuzakTrigger : MonoBehaviour
{
    // Quiz yöneticisine referans (soruları gösteren sistem)
    public QuizManager quizManager;
```

```
    // Soru metni
    [TextArea] public string question;

    // Şıklar
    public string[] options;

    // Doğru cevap
    public string correctAnswer;

    // Doğru cevap verilirse devre dışı bırakılacak tuzaklar (örn. 2. engel)
    public GameObject[] tuzaklarToDisable;

    // Bu trigger daha önce çalıştı mı?
    private bool triggered = false;

    // Oyuncu trigger'a girince çalışır
    private void OnTriggerEnter(Collider other)
    {
        // Daha önce çalıştıysa veya çarpan nesne oyuncu değilse çık
        if (triggered || !other.CompareTag("Player")) return;

        // Artık tetiklenmiş olarak işaretle
        triggered = true;

        // Quiz'i başlat, doğru cevapta belirli tuzaklar devre dışı kalacak
        quizManager.ShowQuestion(
            question,
            options,
            correctAnswer,
            tuzaklarToDisable, // Devre dışı bırakılacak tuzaklar
            null               // Devre dışı bırakılacak diğer objeler yok
        );
    }
}
```

5.5.DonenTuzak.cs

```
using UnityEngine;

/// <summary>
/// Bu script, bir objenin Y ekseninde ileri-geri dönüş hareketi (salınım)
yapmasını sağlar.
/// Hareket istenildiğinde durdurulabilir.
/// </summary>
public class DonenTuzak : MonoBehaviour
{
    // Dönme hızı (ne kadar hızlı salınsın)
    public float speed = 1f;

    // Maksimum dönme açısı (salınım aralığı)
    public float angleRange = 180f;

    // Başlangıç açısı (offset)
    private float angleOffset;

    // Tuzak aktif mi? (hareket ediyor mu?)
    private bool isActive = true;

    void Start()
    {
        // Başlangıçtaki Y rotasyonunu kaydet
```

```
        angleOffset = transform.eulerAngles.y;
    }

    void Update()
    {
        // Eğer aktif değilse hareket etmesin
        if (!isActive) return;

        // PingPong ile ileri-geri açı değeri hesapla
        float angle = Mathf.PingPong(Time.time * speed * 2f, angleRange) -
(angleRange / 2f);

        // Yeni rotasyonu uygula (X ve Z sabit, Y ekseninde salınma)
        transform.rotation = Quaternion.Euler(0f, angleOffset + angle, 0f);
    }

    /// <summary>
    /// Bu fonksiyon çağrıldığında tuzağın dönme hareketi durur.
    /// </summary>
    public void HareketiDurdur()
    {
        isActive = false;
    }
}
```

5.6.DonenTrigger.cs

```
using UnityEngine;

/// <summary>
/// Oyuncu bu trigger'a girdiğinde bir quiz başlatılır.
/// Oyuncu doğru cevabı verirse dönen tuzak(lar) devre dışı bırakılır.
/// </summary>
public class DonenTrigger : MonoBehaviour
{
    // Quiz sistemine referans
    public QuizManager quizManager;

    // Sorulacak soru
    [TextArea] public string question;

    // Cevap şıkları
    public string[] options;

    // Doğru cevap
    public string correctAnswer;

    // Doğru cevap verilirse devre dışı kalacak dönen tuzaklar
    public GameObject[] DonenTuzak;

    // Bu tetikleyici daha önce çalıştı mı?
    private bool triggered = false;

    // Oyuncu bu objeyle çarpışınca çalışır
    private void OnTriggerEnter(Collider other)
    {
        // Eğer tetiklenmişse veya çarpan oyuncu değilse çık
        if (triggered || !other.CompareTag("Player")) return;

        // Artık tetiklenmiş olarak işaretle
        triggered = true;
```

```
        // Soruyu göster ve doğruysa dönen tuzakları devre dışı bırak
        quizManager.ShowQuestion(
            question,
            options,
            correctAnswer,
            DonenTuzak, // devre dışı bırakılacak tuzaklar
            null        // diğer devre dışı bırakılacak objeler yok
        );
    }
}
```

5.7.DonmeyenTuzak.cs

```
using UnityEngine;

/// <summary>
/// Bu script, objenin Y ekseninde ileri-geri salınım hareketi yapmasını
sağlar.
/// Hareket istenildiğinde durdurulabilir. DonenTuzak'tan farkı, bu objenin
dönmesinin engellenmesi amaçlanır.
/// </summary>
public class DonmeyenTuzak : MonoBehaviour
{
    // Tuzak dönüş hızı
    public float speed = 1f;

    // Toplam salınım açısı (derece cinsinden)
    public float angleRange = 180f;

    // Başlangıç açısı
    private float angleOffset;

    // Tuzak aktif mi (dönüyor mu)?
    private bool isActive = true;

    void Start()
    {
        // Objeye ait ilk dönüş açısı kaydedilir
        angleOffset = transform.eulerAngles.y;
    }

    void Update()
    {
        // Eğer tuzak devre dışı bırakıldıysa dönüş yapmaz
        if (!isActive) return;

        // Zamanla ileri-geri dönen açı hesaplanır
        float angle = Mathf.PingPong(Time.time * speed * 2f, angleRange) -
(angleRange / 2f);

        // Hesaplanan açı Y eksenine uygulanır
        transform.rotation = Quaternion.Euler(0f, angleOffset + angle, 0f);
    }

    /// <summary>
    /// Bu fonksiyon çağrıldığında tuzak durur (dönmeyi bırakır).
    /// </summary>
    public void HareketiDurdur()
    {
        isActive = false;
    }
}
```

5.8.HareketliTestere.cs

```
using UnityEngine;

/// <summary>
/// Bu script, objenin ileri-geri salınım (ping-pong) hareketi yapmasını
sağlar.
/// Hareket, X veya Z ekseni boyunca ayarlanabilir. "HareketiDurdur()"
fonksiyonu çağrıldığında testere durur.
/// </summary>
public class HareketliTestere : MonoBehaviour
{
    [Header("Hareket Ayarları")]

    // Testerenin ne kadar mesafede ileri geri hareket edeceği
    public float hareketMesafesi = 3f;

    // Hareketin hızı
    public float hareketHizi = 2f;

    // Başlangıç pozisyonu
    private Vector3 baslangicPozisyonu;

    // Hareketin durdurulup durdurulmadığını kontrol eder
    private bool duruldu = false;

    void Start()
    {
        // Objenin başlangıç pozisyonunu kaydet
        baslangicPozisyonu = transform.position;
    }

    void Update()
    {
        // Eğer durdurulduysa hareket etme
        if (duruldu) return;

        // PingPong hareketini hesapla (ileri - geri)
        float delta = Mathf.PingPong(Time.time * hareketHizi, hareketMesafesi)
- (hareketMesafesi / 2f);

        // Objeyi Z ekseninde hareket ettir (X ekseni için New Vector3(delta,
0f, 0f) yazılabilir)
        transform.position = baslangicPozisyonu + new Vector3(0f, 0f, delta);
    }

    /// <summary>
    /// Bu fonksiyon çağrıldığında testere hareketi durdurulur.
    /// </summary>
    public void HareketiDurdur()
    {
        duruldu = true;
    }
}
```

5.9.Hareketli_Testere.cs

```
using UnityEngine;
```

```
/// <summary>
/// Bu script, objenin X ekseninde ileri-geri salınım (ping-pong) hareketi
yapmasını sağlar.
/// "HareketiDurdur()" fonksiyonu çağrıldığında hareket durdurulur.
/// </summary>
public class Hareketli_Testere : MonoBehaviour
{
    [Header("Hareket Ayarları")]

    // Testerenin ne kadar mesafede ileri-geri hareket edeceği
    public float hareketMesafesi = 3f;

    // Hareketin hızı (ne kadar hızlı gidip geldiği)
    public float hareketHizi = 2f;

    // Objeye ait başlangıç pozisyonu
    private Vector3 baslangicPozisyonu;

    // Testerenin durup durmadığını kontrol eden bayrak
    private bool durduruldu = false;

    void Start()
    {
        // Testerenin sahneye ilk yerleştirildiği pozisyonu al
        baslangicPozisyonu = transform.position;
    }

    void Update()
    {
        // Eğer hareket durdurulduysa işlem yapma
        if (durduruldu) return;

        // PingPong hareketiyle salınım miktarını hesapla
        float delta = Mathf.PingPong(Time.time * hareketHizi, hareketMesafesi)
- (hareketMesafesi / 2f);

        // Objeyi X ekseninde hareket ettir (X yönlü salınım)
        transform.position = baslangicPozisyonu + new Vector3(delta, 0f, 0f);
    }

    /// <summary>
    /// Bu fonksiyon çağrıldığında testere hareketi durdurulur.
    /// </summary>
    public void HareketiDurdur()
    {
        durduruldu = true;
    }
}
```

5.10.TestereTrigger.cs

```
using UnityEngine;

/// <summary>
/// Bu script, oyuncu belirli bir tetikleme alanına girdiğinde
/// bir quiz başlatır ve doğru cevap verilirse hareketli testereleri durdurur.
/// </summary>
public class TestereTrigger : MonoBehaviour
{
    // Quiz panelini yöneten script referansı
    public QuizManager quizManager;
```

```
    // Gösterilecek soru
    [TextArea] public string question;

    // Şıklar
    public string[] options;

    // Doğru cevap
    public string correctAnswer;

    // Z ekseninde hareket eden testereler
    public HareketliTestere[] hareketliTestereler;

    // X ekseninde hareket eden testereler
    public Hareketli_Testere[] hareketliTestereler2;

    // Tetikleme yalnızca bir kez gerçekleşsin diye kontrol değişkeni
    private bool triggered = false;

    // Oyuncu tetikleme alanına girdiğinde çalışır
    private void OnTriggerEnter(Collider other)
    {
        // Eğer daha önce tetiklendiyse veya çarpan nesne oyuncu değilse çık
        if (triggered || !other.CompareTag("Player")) return;

        // Tetiklendiğini işaretle
        triggered = true;

        // Quiz panelini başlat, testereleri gönder
        quizManager.ShowQuestion(
            question,               // Soru metni
            options,                // Seçenekler
            correctAnswer,          // Doğru cevap
            null,                   // Tuzaklar (kullanılmıyor)
            null,                   // Devre dışı bırakılacak objeler
            hareketliTestereler,    // Z ekseni testere dizisi
            hareketliTestereler2    // X ekseni testere dizisi
        );
    }
}
```

5.11.HareketliCekic.cs

```
using UnityEngine;

/// <summary>
/// Bu script, çekiç gibi salınım hareketi yapan bir tuzağın kontrolünü sağlar.
/// Obje Z ekseninde ileri geri dönerek salınım yapar.
/// Quiz'de doğru cevap verilirse `HareketiDurdur()` fonksiyonu ile hareketi
durdurulabilir.
/// </summary>
public class HareketliCekic : MonoBehaviour
{
    [Header("Dönme Ayarları")]

    // Çekiç salınım açısı: örneğin 20 derece ileri, 20 derece geri
    public float aciMiktari = 20f;

    // Salınımın hızı: sinüs fonksiyonuyla kontrol edilir
    public float donusHizi = 2f;

    // Başlangıç Z rotasyonunu tutar
```

```
    private float baslangicZRotasyonu;

    // Hareketin aktif olup olmadığını belirten kontrol değişkeni
    private bool durduruldu = false;

    void Start()
    {
        // Objenin başlangıç Z eksenindeki dönüş açısını kaydet
        baslangicZRotasyonu = transform.localEulerAngles.z;
    }

    void Update()
    {
        // Eğer durdurulmuşsa güncelleme yapılmaz
        if (durduruldu) return;

        // Sinüs fonksiyonu ile zaman bazlı bir açı değeri üret
        float aci = Mathf.Sin(Time.time * donusHizi) * aciMiktari;

        // X ve Y açıları sabit kalırken, sadece Z açısı salınım yapar
        Vector3 yeniAci = new Vector3(
            transform.localEulerAngles.x,
            transform.localEulerAngles.y,
            baslangicZRotasyonu + aci
        );

        // Objeye yeni dönüş açısı atanır
        transform.localEulerAngles = yeniAci;
    }

    /// <summary>
    /// QuizManager tarafından çağrılarak çekiç hareketini durdurur.
    /// </summary>
    public void HareketiDurdur()
    {
        durduruldu = true;
    }
}
```

5.12.CekicTrigger.cs

```
using UnityEngine;

/// <summary>
/// Oyuncu bu tetikleyiciye girdiğinde quiz başlatılır.
/// Oyuncu soruyu doğru cevaplarsa bağlı çekiç tuzakları durdurulur.
/// </summary>
public class CekicTrigger : MonoBehaviour
{
    // Quiz'i yönetecek olan QuizManager referansı
    public QuizManager quizManager;

    // Soru metni (Inspector'da çok satırlı girilebilir)
    [TextArea] public string question;

    // Cevap seçenekleri (3 adet olması beklenir)
    public string[] options;

    // Doğru cevabı belirten string
    public string correctAnswer;

    // Etkileşimde durdurulacak HareketliCekic component'lerine sahip objeler
```

```
    public HareketliCekic[] cekicler;

    // Tetikleme sadece bir kez gerçekleşmesi için kontrol
    private bool triggered = false;

    // Çarpışma kontrolü: oyuncu tetikleyiciye girdiğinde çalışır
    private void OnTriggerEnter(Collider other)
    {
        // Eğer zaten tetiklendiyse ya da çarpan nesne oyuncu değilse çık
        if (triggered || !other.CompareTag("Player")) return;

        triggered = true; // Artık tetiklenmiş sayılır

        // QuizManager üzerinden soru gösterimi başlatılır
        quizManager.ShowQuestion(
            question,       // Soru metni
            options,        // Şıklar
            correctAnswer,  // Doğru cevap
            null,           // Genel tuzaklar kullanılmıyor
            null,           // Devre dışı bırakılacak objeler yok
            null,           // Z ekseni hareketli testere yok
            null,           // X ekseni hareketli testere yok
            cekicler        // Hareketli çekiç dizisi (durdurulacaklar)
        );
    }
}
```

5.13.KapananTuzak.cs

```
using UnityEngine;

/// <summary>
/// Bu script, kapanan tuzakların ileri-geri (açılıp kapanma) hareketini
sağlar.
/// Lerp + PingPong ile yumuşak bir şekilde pozisyon değiştirir.
/// Quiz'de doğru cevap verildiğinde HareketiDurdur() çağrılarak hareket
durdurulur.
/// </summary>
public class KapananTuzak : MonoBehaviour
{
    [Header("Hareket Ayarları")]

    // Tuzak başlangıç pozisyonundan ne kadar kayacak (yön ve mesafe)
    public Vector3 hedefOffset = new Vector3(0f, 0f, 1f);

    // Hareketin hızı
    public float hiz = 2f;

    // Başlangıç pozisyonunu tutar
    private Vector3 baslangicPozisyonu;

    // Tuzak durdurulduğunda true olur
    private bool durduruldu = false;

    void Start()
    {
        // Başlangıç pozisyonunu kaydet
        baslangicPozisyonu = transform.position;
    }

    void Update()
    {
```

```
        // Eğer durdurulduysa hiçbir işlem yapma
        if (durduruldu) return;

        // Zamanla 0 ile 1 arasında gidip gelen t değeri üret
        float t = Mathf.PingPong(Time.time * hiz, 1f);

        // Pozisyonu, başlangıç ile hedef pozisyon arasında lineer olarak
değiştir
        transform.position = Vector3.Lerp(
            baslangicPozisyonu,
            baslangicPozisyonu + hedefOffset,
            t
        );
    }

    /// <summary>
    /// Quiz'de doğru cevap verildiğinde hareketi durdurur.
    /// </summary>
    public void HareketiDurdur()
    {
        durduruldu = true;
    }
}
```

5.14.KapakTrigger.cs

```
using UnityEngine;

/// <summary>
/// Bu script, oyuncu belirli bir tetikleme alanına (trigger) girdiğinde bir
quiz sorusu gösterir.
/// Oyuncu doğru cevabı verdiğinde `KapananTuzak` tipindeki engeller
durdurulur.
/// </summary>
public class KapakTrigger : MonoBehaviour
{
    public QuizManager quizManager; // Quiz sistemine referans

    [TextArea]
    public string question;         // Sorulacak soru metni
    public string[] options;        // Cevap seçenekleri (3 seçenek varsayılır)
    public string correctAnswer;    // Doğru cevap

    public KapananTuzak[] kapananTuzaklar; // Durdurulacak tuzaklar dizisi

    private bool triggered = false; // Oyuncu tetikleyiciyi bir kere
tetiklediyse tekrar tetiklenmesin

    private void OnTriggerEnter(Collider other)
    {
        // Eğer zaten tetiklendiyse veya oyuncu değilse, çık
        if (triggered || !other.CompareTag("Player")) return;

        triggered = true;

        // QuizManager üzerinden soruyu göster. Diğer tuzak türleri null
gönderiliyor.
        quizManager.ShowQuestion(
            question,
            options,
            correctAnswer,
            null,               // Genel tuzaklar
```

```
            null,               // Devre dışı bırakılacak objeler
            null,               // HareketliTestere[]
            null,               // Hareketli_Testere[]
            null,               // HareketliCekic[]
            kapananTuzaklar     // KapananTuzak[]
        );
    }
}
```

5.15.FinishLineTrigger.cs

```
using UnityEngine;
using TMPro;
using UnityEngine.SceneManagement;

public class FinishLineTrigger : MonoBehaviour
{
    public GameObject finishTextObject;
    public GameObject restartButtonObject;
    public bool stopTimeOnFinish = true;

    private bool hasFinished = false;

    void Start()
    {
        if (finishTextObject != null)
            finishTextObject.SetActive(false);

        if (restartButtonObject != null)
            restartButtonObject.SetActive(false);
    }

    private void OnTriggerEnter(Collider other)
    {
        if (hasFinished) return;

        if (other.CompareTag("Player"))
        {
            hasFinished = true;
            TriggerFinishSequence();
        }
    }

    private void TriggerFinishSequence()
    {
        Debug.Log("FinishLineTrigger: Oyuncu bitiş çizgisine ulaştı!");

        if (stopTimeOnFinish)
        {
            Time.timeScale = 0f;
        }

        // Fare imlecini görünür yap ve serbest bırak
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;

        if (finishTextObject != null)
            finishTextObject.SetActive(true);

        if (restartButtonObject != null)
            restartButtonObject.SetActive(true);
    }
```

```
    public void OnRestartButtonClicked()
    {
        Debug.Log("Restart butonuna tıklandı.");

        Time.timeScale = 1f;
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }
}
```

5.16.HealthManager.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using TMPro;  // TextMeshPro kullanıyorsanız

public class HealthManager : MonoBehaviour
{
    public int maxHealth = 3;
    private int currentHealth;

    [Header("Game Over Ekranı")]
    public GameObject GameOverPanel;
    public TMP_Text HealthText;

    void Start()
    {
        currentHealth = maxHealth;
        UpdateHealthUI();

        // Başlangıçta "Game Over Panel" kapalı olmalı
        GameOverPanel.SetActive(false);
    }

    /// <summary>
    /// Can azaltma işlemi. Eğer can 0'a iner ya da altına düşerse GameOver()
çağrılır.
    /// </summary>
    public void TakeDamage(int amount)
    {
        Debug.Log("TakeDamage çağrıldı, amount = " + amount);

        currentHealth -= amount;
        if (currentHealth <= 0)
        {
            currentHealth = 0;
            GameOver();
        }

        UpdateHealthUI();
    }

    /// <summary>
    /// Kalan can bilgisi UI üzerinde güncellenir.
    /// </summary>
    void UpdateHealthUI()
    {
        if (HealthText != null)
            HealthText.text = "Can: " + currentHealth.ToString();
    }

    /// <summary>
```

```
    /// Can sıfırlandığında oyun durur ve Game Over paneli açılır. Fare imleci
görünür yapılır.
    /// </summary>
    void GameOver()
    {
        Debug.Log("GameOver() tetiklendi");

        Time.timeScale = 0f;

        //  Fare imlecini serbest bırak ve görünür yap
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;

        // Paneli aktif et
        GameOverPanel.SetActive(true);
    }

    /// <summary>
    /// Restart butonuna tıklandığında sahneyi yeniden yükler, zamanı normale
çevirir.
    /// </summary>
    public void RestartGame()
    {
        Time.timeScale = 1f;
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

        // Not: Sahne yüklendiğinde başka bir yerde fareyi tekrar kilitleyen
kodunuz varsa
        // fare yeniden kilitlenir. Gerekirse buraya da Cursor.lockState =
Locked; Cursor.visible = false ekleyebilirsiniz.
    }
}
```

5.17.QuizManager.cs

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class QuizManager : MonoBehaviour
{
    [Header("Quiz UI Elemanları")]
    public GameObject quizPanel;
    public TMP_Text questionText;

    [Header("Seçenek Butonları")]
    public Button optionA;
    public Button optionB;
    public Button optionC;

    [Header("Sağlık Sistemi")]
    public HealthManager healthManager;   //  Burası Inspector'da atanacak

    private string correctAnswer;
    private GameObject[] currentTuzaklar;
    private GameObject[] currentObjectsToDisable;

    private HareketliTestere[] currentHareketliTestereler;
    private Hareketli_Testere[] currentHareketliTestereler2;
    private HareketliCekic[] currentCekicler;
    private KapananTuzak[] currentKapananTuzaklar;
```

```
    /// <summary>
    /// Yeni bir soru açıldığında, paneli aktif eder ve zamanı durdurur.
    /// </summary>
    public void ShowQuestion(string question, string[] options, string correct,
                             GameObject[] tuzaklar = null,
                             GameObject[] objectsToDisable = null,
                             HareketliTestere[] hareketliTestereler = null,
                             Hareketli_Testere[] hareketliTestereler2 = null,
                             HareketliCekic[] cekicler = null,
                             KapananTuzak[] kapananTuzaklar = null)
    {
        quizPanel.SetActive(true);
        questionText.text = question;
        correctAnswer = correct;

        // Hangi engeller ilgilenecekse referansları alıyoruz
        currentTuzaklar = tuzaklar;
        currentObjectsToDisable = objectsToDisable;
        currentHareketliTestereler = hareketliTestereler;
        currentHareketliTestereler2 = hareketliTestereler2;
        currentCekicler = cekicler;
        currentKapananTuzaklar = kapananTuzaklar;

        // Butonlardaki metinleri güncelle
        optionA.GetComponentInChildren<TMP_Text>().text = options[0];
        optionB.GetComponentInChildren<TMP_Text>().text = options[1];
        optionC.GetComponentInChildren<TMP_Text>().text = options[2];

        // Önceki event listener'ları temizle
        optionA.onClick.RemoveAllListeners();
        optionB.onClick.RemoveAllListeners();
        optionC.onClick.RemoveAllListeners();

        // Yeni listener ekle (yanıt kontrolü)
        optionA.onClick.AddListener(() => CheckAnswer(options[0]));
        optionB.onClick.AddListener(() => CheckAnswer(options[1]));
        optionC.onClick.AddListener(() => CheckAnswer(options[2]));

        // Zamanı durdur, mouse görünür yap
        Time.timeScale = 0f;
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }

    /// <summary>
    /// Kullanıcı bir şıkkı seçtiğinde çağrılır. Yanlışı kapatmaz, yalnızca
doğruysa paneli kapatır.
    /// </summary>
    public void CheckAnswer(string selected)
    {
        if (selected == correctAnswer)
        {
            Debug.Log("Doğru cevap!");

            //  Doğruysa ilgili tuzakları durdur:
            if (currentTuzaklar != null)
            {
                foreach (GameObject trap in currentTuzaklar)
                {
                    if (trap == null) continue;

                    var hareketli1 = trap.GetComponent<HareketliTestere>();
                    if (hareketli1 != null)
                    {
                        hareketli1.HareketiDurdur();
```

```
                continue;
            }

            var hareketli2 = trap.GetComponent<Hareketli_Testere>();
            if (hareketli2 != null)
            {
                hareketli2.HareketiDurdur();
                continue;
            }

            var cekic = trap.GetComponent<HareketliCekic>();
            if (cekic != null)
            {
                cekic.HareketiDurdur();
                continue;
            }

            var kapanan = trap.GetComponent<KapananTuzak>();
            if (kapanan != null)
            {
                kapanan.HareketiDurdur();
                continue;
            }

            trap.SetActive(false);
        }
    }

    if (currentObjectsToDisable != null)
    {
        foreach (GameObject obj in currentObjectsToDisable)
        {
            if (obj != null)
                obj.SetActive(false);
        }
    }

    if (currentHareketliTestereler != null)
    {
        foreach (var testere in currentHareketliTestereler)
        {
            if (testere != null)
                testere.HareketiDurdur();
        }
    }

    if (currentHareketliTestereler2 != null)
    {
        foreach (var testere in currentHareketliTestereler2)
        {
            if (testere != null)
                testere.HareketiDurdur();
        }
    }

    if (currentCekicler != null)
    {
        foreach (var cekic in currentCekicler)
        {
            if (cekic != null)
                cekic.HareketiDurdur();
        }
    }

    if (currentKapananTuzaklar != null)
```

```
            {
                foreach (var kapanan in currentKapananTuzaklar)
                {
                    if (kapanan != null)
                        kapanan.HareketiDurdur();
                }
            }

            //  Doğruysa paneli kapat ve zamanı normale döndür
            quizPanel.SetActive(false);
            Time.timeScale = 1f;

            Cursor.lockState = CursorLockMode.Locked;
            Cursor.visible = false;
        }
        else
        {
            // Yanlış cevap: panel kapanmasın, sadece can azalsın
            Debug.Log("Yanlış cevap! Can azalıyor.");

            if (healthManager != null)
            {
                healthManager.TakeDamage(1);
            }
            else
            {
                Debug.LogWarning("QuizManager: healthManager atanmadı!");
            }

            // Not: Burada Time.timeScale veya quizPanel.SetActive(false) yok,
            // böylece kullanıcı tekrar denemeye devam eder.
        }
    }
}
```

6.Game Project Management (Future work)

6.1. WBS (Work Breakdown Structure)

6.1.1. Tanım ve Hikâye

- 1.1. Proje amacı ve hedef kitlesi belirleme
- 1.2. Oyun akışı planı (engele yaklaş – soruyu doğru cevapla – parkuru bitir – tebrik mesajı)

6.1.2. Tasarım (Design)

- 2.1. 3D modellerin seçimi ve yerleştirilmesi
    - parkur, engeller (dikenli engeller, testere, hareketli çekiç)
- 2.2. Karakter ve kamera kontrol sistemi

2.3. UI/Canvas tasarımı (quiz paneli, butonlar, tebrik paneli)

6.1.3. Kodlama (Development)

- 3.1. Trigger sistemleri (yaklaşınca bilgi/soru gösterimi)
- 3.2. Quiz sistemi
    - Doğru cevap: engeller durur ya da ortadan kalkar, yanlış cevap:can sayısı azalır
- 3.3. Görev tamamlanması bitiş çizgisine varınca "Tebrikler" paneli
- 3.4. Sahne geçişi (ana menü → oyun sahnesi)

6.1.4. Test ve Düzeltme (Test & Fix)

- 4.1. Oyuncu hareketleri ve tuzaklara çarpma kontrolleri başarıyla test edildi.
- 4.2. Quiz paneli, doğru ve yanlış cevap senaryolarında sorunsuz çalıştı.
- 4.3. Doğru cevapta tuzaklar durdu veya devre dışı kaldı.
- 4.4. Yanlış cevapta panel açık kaldı, can sistemi düzgün işledi.
- 4.5. Tüm tuzak türleri (testere, çekiç, dönen, kapanan) çalıştı ve durdurulabilirliği test edildi.
- 4.6. Platformdan düşme durumunda "Tekrar Oyna" butonu işlevsel olarak test edildi.
- 4.7. Ana menüdeki butonların yönlendirmeleri doğru çalıştı.
- 4.8. Zaman ve mouse yönetimi (durma, görünürlük) sorunsuz biçimde test edildi.

6.1.5. Yayınlama ve Teslim

- 5.1.Android Build Alma (Ayarların Düzenlenmesi):
  Build Settings üzerinden platform Android olarak seçildi, gerekli SDK ayarları yapıldı ve başarıyla .apk dosyası oluşturuldu.

- 5.2. Rapor ve Sunum İçin Ekran Görüntüleri:
  Uygulamanın menü, quiz, tuzak, tekrar başlatma ve bitiş ekranlarından yüksek çözünürlüklü ekran görüntüleri alındı ve rapora eklendi.
- 5.3. Final Sunuma Hazırlanma:
  Projenin işleyişi, görev dağılımı, karşılaşılan zorluklar ve kazanımlar detaylı şekilde PowerPoint sunumuna aktarıldı ve prova sunumları yapılarak teslimata hazır hale getirildi.

6.2.Resource Sheet

	Kaynak Türü	Görevi/Kullanımı	Tahmini Süre	Notlar
Şule	İnsan Kaynağı	Proje yöneticisi, yazılımcı, tasarımcı	8 hafta	Tüm aşamalarda aktif
Unity	Yazılım Aracı	Oyun geliştirme platformu	8 hafta	URP kullanıldı
C# Script	Yazılım Dili	Tüm oyun mekaniği, trigger, sorular, sahne geçişi	6 hafta	Unity ile entegre
TMP	Unity Asset	UI metinleri (sorular, cevaplar, tebrik mesajı)	3 hafta	Ekrandaki yazılar ve quiz sistemi için

3D Model Kitleri	3D Asset Paketi	3D modeller (platform, engeller, tuzaklar)	7 hafta	Unity Asset Store'dan alındı
PC	Donanım	Geliştirme ortamı (Unity editörü)	8 hafta	Ana geliştirme ortamı
Word	Dokümantasyon Aracı	Rapor hazırlanması	1 hafta	Teslim için rapor bölümü

6.3. Gantt Chart (8 Haftalık, Haftalık İnterval)

Görev No	Görev Adı	Süre	Başlangıç	Bitiş	H1	H2	H3	H4	H5	H6	H7	H8
1	Proje planlaması ve kategori belirleme	1	H1	H1	X							
2	3D ortam kurulumu (platform, engeller, tuzaklar)	2	H1	H2	X	X						
3	Engellerin Hareketi	1	H2	H2		X						
4	Quiz sistemi kurulumu	1	H3	H3			X					

5	Canvas UI – Soru hazırlayıp gösterme	2	H3	H5			X	X	X			
6	Sorularda doğru cevaplarda engel durdurulması	1	H4	H5				X				
7	Sorularda yanlış cevaplarda can azalması	1	H5	H6					X	X		
8	"Tebrikler" ekranı – bitiş çizgisine varınca	1	H6	H7						X	X	
9	Son testler & hata düzeltmeleri	1	H8	H8								X

6.4. Risk Table

Risk Adı	Açıklama	Olasılık	Etkisi	Risk Seviyesi	Çözüm / Önlem
1. Kod Hataları	Oyun mekaniklerinde (quiz, sahne geçişi, tuzaklar) beklenmedik hatalar	Orta	Yüksek	Yüksek	Test senaryoları uygulanmalı, her tuzak türü ayrı test edilmeli
2. UI Sorunları	Quiz panelinin görünmemesi, butonların çalışmaması	Orta	Orta	Orta	UI öğeleri prefab olarak kontrol edilmeli, olay bağlamaları temiz
3. Build Alma Sorunu (APK)	Android export işlemlerinde hata veya eksik asset	Orta	Yüksek	Yüksek	Build ayarları önceden test edilmeli, eksik paketler kontrol edilmeli
4. Oyun İçi Can Sistemi Çalışmıyor	Yanlış cevapta can azalmasının gerçekleşmemesi	Orta	Yüksek	Yüksek	HealthManager bağlantısı kontrol

Risk Adı	Açıklama	Olasılık	Etkisi	Risk Seviyesi	Çözüm / Önlem
					edilmeli, her sahnede atanmalı
5. Kaynakların Eksikliği	3D modellerin veya seslerin yetişmemesi	Düşük	Yüksek	Orta	Yedek kaynaklar hazırlanmalı, gerekirse sadeleştirme yapılmalı