OYUN PROJESİ ANALİZİ

Dalga Tabanlı Savunma Oyunu

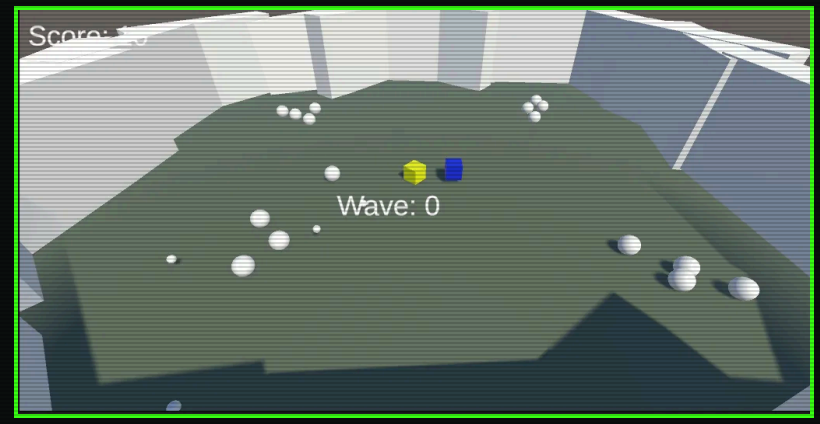
Teknik Mekanik İncelemesi

> SYSTEM INITIALIZED
> ARCADE TERMINAL v1.0
> READY FOR ANALYSIS

PROJE ÖZETİ

OYUN KONSEPTI

Oyuncu (Sarı Küp), Base'i (Mavi Küp) dalgalar halinde gelen düşmanlara (Beyaz Küreler) karşı korur.



TEMEL MEKANİKLER

- > Wave Sistemi: Her 1 dakikada dalga atlanır
- > Düşman Sayısı: Dalga ilerledikçe artar
- > Oyuncu Kontrolü: Space ile ateş, LeftShift ile hızlı mod
- > Puan Sistemi: Her düşman 10 puan değerinde

ZORLUK ARTIŞI

Dalga geçişlerinde düşman hızı ve sayısı dinamik olarak artırılır, oyunu giderek zorlaştırır.

TEMEL OYNANIS MEKANİKLERİ

WAVE SİSTEMİ

Her 1 dakikada bir dalga atlanır. Dalga sayısı arttıkça düşman sayısı ve hızı dinamik olarak artırılır.

OYUNCU KONTROLÜ

Sarı Küp oyuncuyu temsil eder. W/A/S/D veya yön tuşları ile hareket, fare ile dönüş yapılır.

ATEŞ ETME

Space tuşu ile mermi atılır. Her mermi 10 birim hızla ileriye doğru hareket eder ve 5 saniye sonra yok olur.

PUAN SİSTEMİ

Her düşman 10 puan değerindedir. Mermiler düşmanlara çarptığında puan eklenir ve ekranda güncellenir.

TARAMAYLI MOD (BURST FIRE)

- → LeftShift tuşu ile etkinleştirilir
- → 5 saniye boyunca hızlı ateş yapılabilir
- → Kullanım sonrası 10 saniye bekleme süresi (cooldown)
- → Stratejik avantaj sağlayan güçlü bir mekanik

MERMİ SİSTEMİ

MERMI PARAMETRELERİ

```
public float speed = 10f;
public float lifeTime = 5f;
public float bulletScale = 2f;
```

Mermi hızı 10 birim/saniye, yaşam süresi 5 saniye, boyut 2 birim olarak ayarlanmıştır.

HAREKET MANTIGI

```
// Update() metodunda
transform.Translate(Vector3.forward * speed * Time.deltaTime);
```

Mermi her frame'de ileri doğru hareket eder. Belirtilen yaşam süresi sonunda otomatik olarak yok edilir.

ÇARPIŞMA SİSTEMİ

- → Enemy tag'ine sahip objeye çarpma algılanır
- → Çarpışmada 10 puan eklenir
- → Hem mermi hem düşman yok edilir

DÜŞMAN HAREKETI VE HIZ YÖNETİMİ

ENEMYMOVE SCRIPT

Düşmanları Base hedefine doğru hareket ettirir ve dalga ilerledikçe hızlarını dinamik olarak artırır.

HEDEF BELİRLEME

```
target = GameObject.Find("Base");
// Veya tag ile bulma
```

Düşman, Base objesini hedef olarak belirlenir. Eğer atanmamışsa, tag veya isim ile otomatik bulunur.

HIZ PARAMETRELERİ

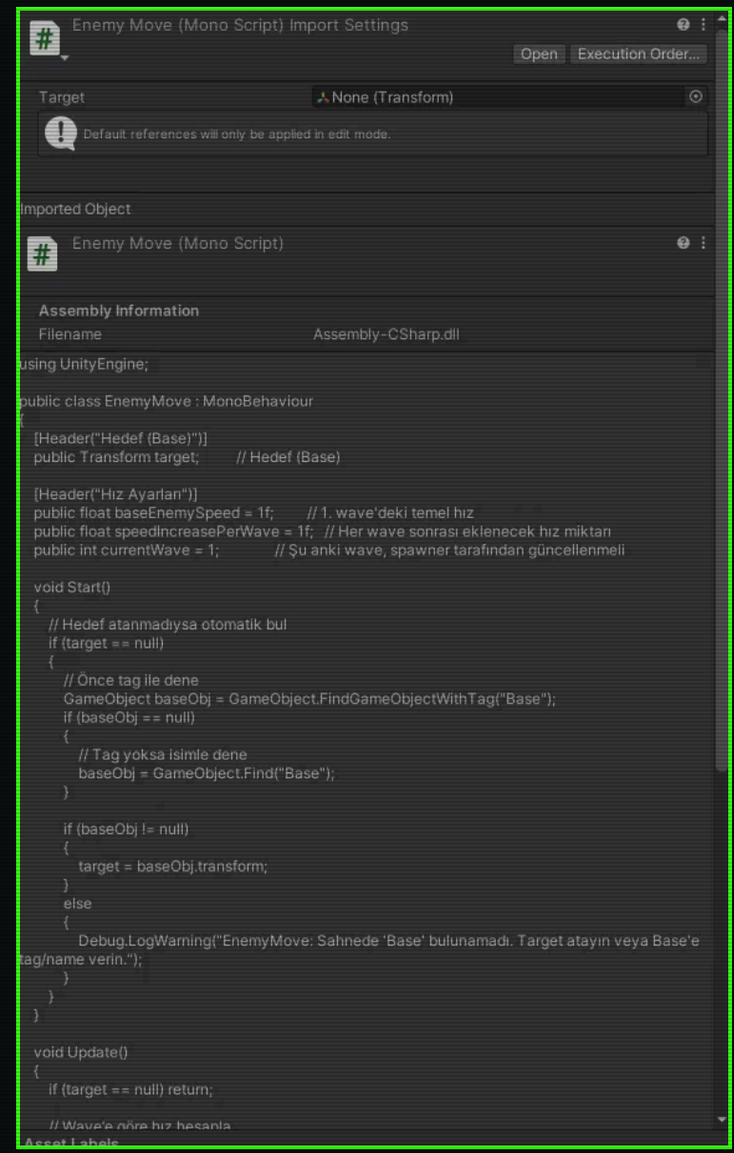
- → baseEnemySpeed = 1f (1. Wave hızı)
- → speedIncreasePerWave = 1f (Wave başına artış)
- → currentWave = Mevcut dalga numarası

DINAMIK HIZ HESAPLAMASI

```
Düşman Hızı = baseEnemySpeed + (currentWave × speedIncreasePerWave)
Örnek: Wave 5 → 1 + (5 × 1) = 6 birim/saniye
```

HAREKET MANTIGI

- → Düşman, Base'e doğru sürekli hareket eder
- → Hız, mevcut wave'e göre otomatik güncellenir
- → Oyun zorlaştıkça düşmanlar hızlanır



DÜŞMAN ÜRETİMİ VE DALGA SİSTEMİ

ENEMYSPAWNER SCRIPT

Düşmanları belirli noktalardan (Spawner) üretir ve dalga ilerledikçe zorluk seviyesini dinamik olarak artırır.

SPAWN PARAMETRELERİ

- → enemyPrefab: Spawnlanacak düşman objesi
- → target: Düşmanların hedefi (Base)
- → spawnDelay: 0.5f (Spawnlar arası bekleme)
- → enemySpeed: 3f (Düşman hızı)

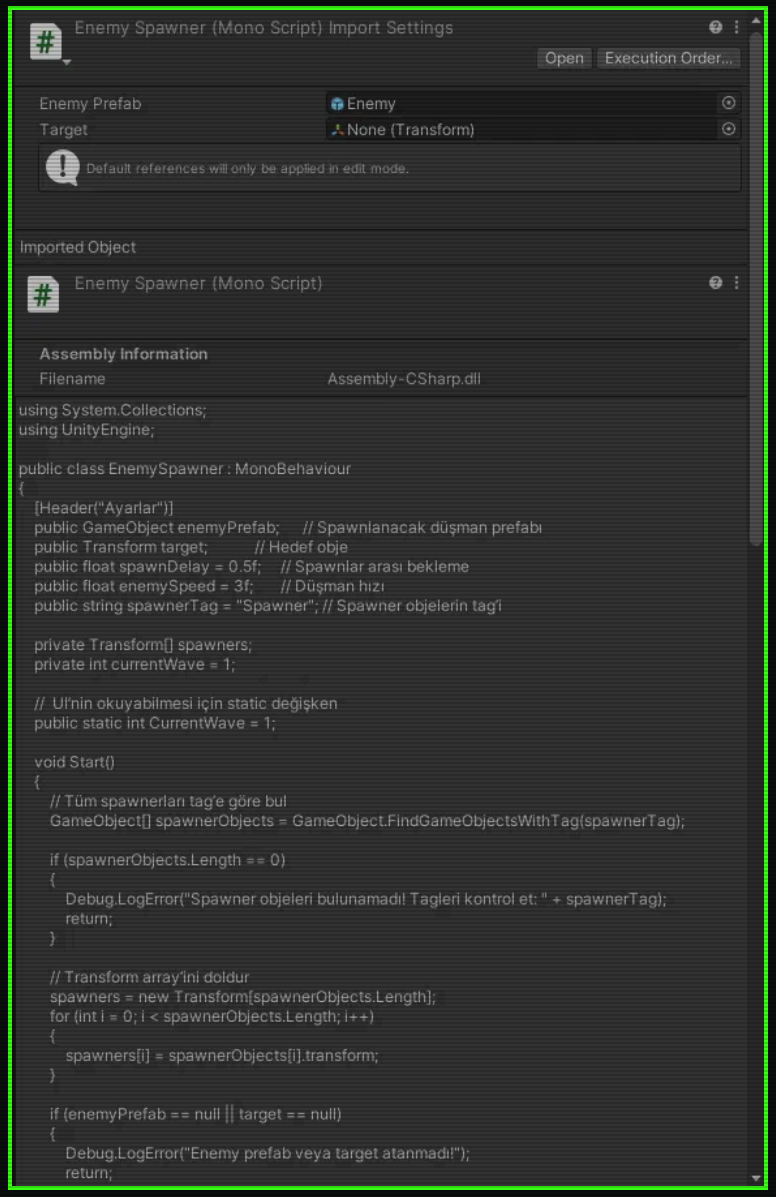
WAVE YÖNETİMİ

currentWave statik değişkeni tüm spawnerler ve düşmanlar tarafından paylaşılır. Her wave'de zorluk parametreleri güncellenir.

ZORLUK ARTIŞI

- → Spawn hızı azalır (spawnDelay düşer)
- → Düşman sayısı artar
- → Düşman hızı artar (EnemyMove ile entegre)
- → Oyun giderek zorlaşır ve hızlanır

```
InvokeRepeating(nameof(AddWave), 60f, 60f);
// 60 saniyede bir wave artırılır
```



OYUNCU KONTROLÜ VE ATEŞ ETME

TowerMove Script

HAREKETE PARAMETRELER

- moveSpeed = 10f (Hareket hızı)

KONTROL TUŞLARI

- W / UpArrow = İleri
- S / DownArrow = Geri
- A / LeftArrow = Sol
- D / RightArrow = Sağ

> **Update() Metodu**
> Tuş girişlerini kontrol eder, moveDirection vektörü oluşturur ve transform.Translate() ile uygulanır.

TowerAimAndShoot Script

ATEŞ PARAMETRELER

- rotationSpeed = 10f (Dönüş hızı)
- fireRate = 0.25f (Ateş aralığı)

HIZLI ATEŞ MODU

- rapidFireDuration = 3f
- rapidFireCooldown = 7f

> **RotateWithMouse()**
> Fare konumuna göre oyuncu döner. Raycast ile hedef belirlenir ve dönüş hesaplanır.

Kontrol Mekanizmaları

ATEŞ ETME

- Space tuşu = Normal ateş
- Mermi hızı = 10 birim/saniye
- Mermi yaşam süresi = 5 saniye

HIZLI ATEŞ (BURST FIRE)

- LeftShift tuşu = Etkinleştir
- 5 saniye boyunca hızlı ateş yapılabilir
- Kullanım sonrası 10 saniye bekleme süresi

ENTEGRASYON

TowerMove ve TowerAimAndShoot scriptleri birlikte çalışarak oyuncunun hem hareket etmesini hem de ateş etmesini sağlar. Fare kontrolü ile dönüş ve Space/LeftShift tuşları ile ateş etme, oyuncu için tam kontrol sağlar.

ARAYÜZ SİSTEMLERİ

SCORE MANAGER

- Puan Yönetimi
- Mermi-Enemy çarpışması
- UI Güncelleme

STATIK DEĞİŞKEN:

public static int score

METODLAR:

AddScore(int amount)

UpdateText()

REFERANS:

TextMeshProUGUI

WAVE UI

- Dalga Gösterimi
- 1 dakika aralığı
- Zorluk Göstergesi

STATIK DEĞİŞKEN:

public static int wave

METODLAR:

AddWave()

UpdateText()

ZAMANLAMA:

InvokeRepeating(60f)

GAME OVER UI

- Oyun Sonu Paneli
- Restart Butonu
- Sahne Yenileme

REFERANSLAR:

GameOverPanel

RestartBtn

METODLAR:

Show()

Restart()

ZAMANLANDIRıCı:

Time.timeScale

SONUÇ VE DEĞERLENDİRME

PROJE BAŞARILARI

- ✓ Dalga tabanlı savunma mekaniği başarıyla uygulandı
- ✓ Dinamik zorluk artışı sistemi entegre edildi
- ✓ Temiz ve modüler kod yapısı sağlandı
- ✓ Kullanıcı dostu arayüz tasarımı gerçekleştirildi
- ✓ Çarpışma ve fizik sistemi düzgün çalışıyor

TEKNİK BAŞARILAR

- ✓ 8 Script başarıyla entegre edildi
- ✓ Statik değişkenler ile global veri yönetimi
- ✓ Mermi-Enemy çarpışma sistemi optimize edildi
- ✓ Wave tabanlı hız ve spawn artışı
- ✓ Time.timeScale ile oyun durdurma/devam

GELİŞTİRİLMESİ GEREKEN ALANLAR

- → Taramalı mod için görsel/ses geri bildirimi eklenmesi
- → Wave geçişlerinde daha kademeli zorluk artışı
- → Farklı enemy türleri ve davranışları eklenmesi
- → Power-up sistemi ve özel abilityler
- → Yüksek skor tablosu ve başarı sistemi
- → Ses efektleri ve müzik entegrasyonu

GENEL DEĞERLENDİRME

Proje, dalga tabanlı bir savunma oyununu başarıyla uygulamıştır. Dinamik zorluk artışı, temiz kod mimarisi ve kullanıcı dostu arayüz ile oyun döngüsü tamamlanmıştır. Teknik olarak sağlam bir temel oluşturulmuş olup, geliştirilmesi gereken alanlar ise oyun derinliğini ve oyuncu deneyimini artıracak yeni mekanikler ve sistemlerdir.

8	10	∞
SCRIPT ENTEGRE	MEKANIK DETAYI	POTANSIYEL