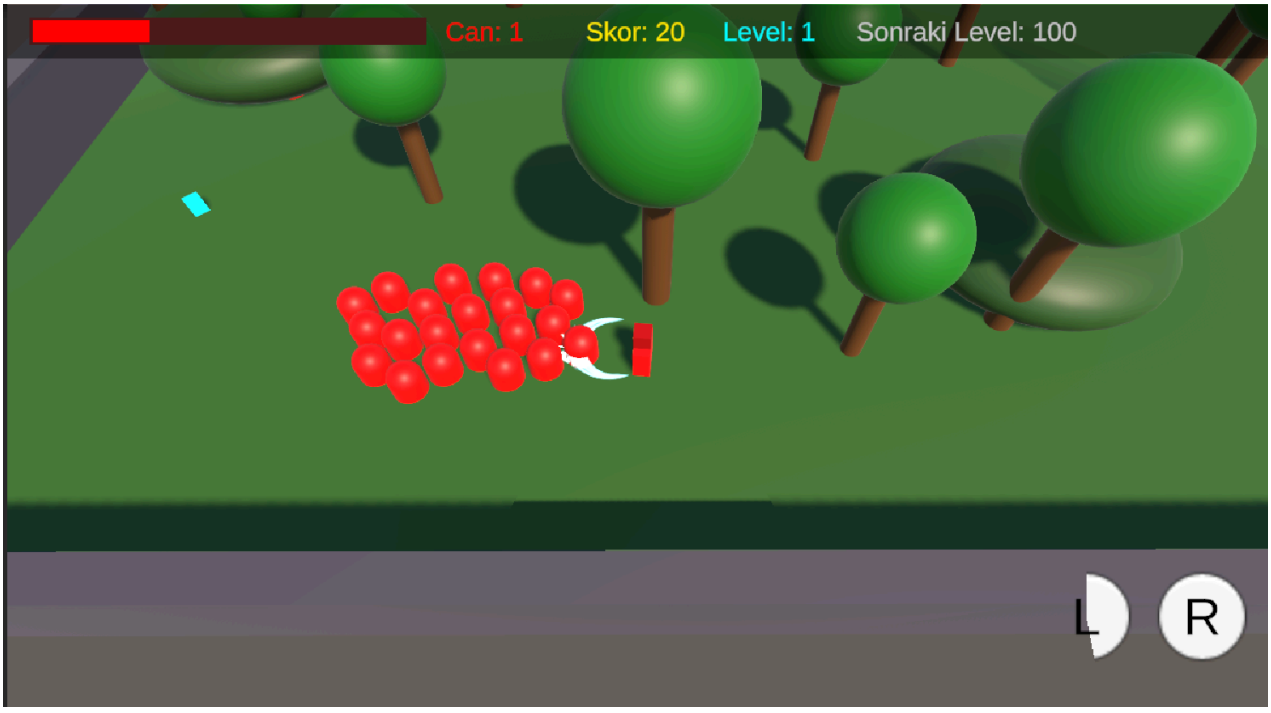
HACK'N'SLASH

Teknik Dokümantasyon

Hazırlayan : Muhammet Çiğdem

No : 2207231041



1. Genel Bakış

Oyuncu, üç farklı karakter sınıfından birini seçerek dalgalar halinde gelen düşmanlarla mücadele eder.

Temel Özellikler

Oyun üç oynanabilir karakter sunar: yakın dövüş odaklı Savaşçı, uzak mesafe saldırısına sahip Nişancı ve stratejik tuzaklar kurabilen Tuzakçı. Her karakter kendine özgü birincil ve özel saldırı mekaniklerine sahiptir.

Harita sistemi üç farklı ortam sunmaktadır: ağaç engellerinin bulunduğu Orman, kaya formasyonlarıyla dolu Çöl ve bina yapılarının yer aldığı Şehir. Haritalar prosedürel olarak oluşturulur ve her oyunda farklı engel dizilimleri sunar.

Kontrol Şeması

Girdi	İşlev
W, A, S, D	Karakter hareketi
Fare	Yön kontrolü
Sol tık	Birincil saldırı
Sağ tık	Özel saldırı
E tuşu	Manuel patlama (Tuzakçı)
Escape	Duraklatma menüsü

2. Mimari Yapı

Klasör Organizasyonu

Klasör	İçerik
Core	Oyunun temel yönetim sınıfları
Player	Karakter sınıfları ve hareket sistemi
Enemy	Düşman yapay zekası ve spawn mekanizması
Combat	Saldırı nesneleri: mermi, tuzak, patlayıcı
Map	Prosedürel harita oluşturma
Camera	Kamera takip sistemi
Effects	Görsel efekt bileşenleri
Items	Toplanabilir güç artırıcılar
UI	Arayüz yönetimi

3. Çekirdek Sistemler

GameManager

GameManager sınıfı, oyunun merkezi kontrol noktasıdır ve Singleton tasarım deseni kullanılarak implement edilmiştir. Bu sayede oyun boyunca tek bir örnek üzerinden erişim sağlanır ve sahneler arası veri kaybı önlenir.

Bu sınıf şu verileri yönetir:

- Oyuncunun mevcut puanı ve level bilgisi
- Sağlık değeri ve kalan can hakkı
- Oyun durumu (aktif, duraklatılmış, bitmiş)

Sınıf, C# event sistemi aracılığıyla diğer bileşenlerle iletişim kurar. Örneğin skor değiştiğinde OnScoreChanged eventi tetiklenir ve UI sistemi bu eventi dinleyerek ekranı günceller. Bu yaklaşım, bileşenler arası bağımlılığı minimize eder.

Zorluk sistemi level bazlı çalışır. Her level geçişinde düşman hasarı yüzde yirmi artar, spawn edilen düşman sayısı iki adet fazlalaşır ve spawn aralığı yüzde on kısalır. Bu değerler GetEnemyDamageMultiplier, GetEnemySpawnCount ve GetSpawnInterval metodları ile hesaplanır.

CharacterSelector

Karakter seçim sistemi, oyuncunun ana menüde yaptığı tercihi oyun sahnesine taşır. Seçilen karakter tipi static bir değişkende saklanır ve oyun sahnesinde SpawnCharacter metodu çağrıldığında ilgili prefab instantiate edilir.

Eğer karakter prefabı atanmamışsa, sistem CreateDefaultCharacter metodunu kullanarak temel geometrilerden oluşan bir placeholder karakter oluşturur. Bu özellik, geliştirme sürecinde prefab olmadan test yapılmasına olanak tanır.

GameBootstrap

Bootstrap sınıfı, oyun sahnesinin ilk yüklenişinde tüm sistemlerin doğru sırayla başlatılmasını sağlar. Awake metodunda yönetici nesneler oluşturulur, Start metodunda ise harita generate edilir, oyuncu spawn edilir ve kamera hedefi atanır.

4. Karakter Sistemi

Temel Sınıf: PlayerBase

Tüm oynanabilir karakterler PlayerBase abstract sınıfından türetilmiştir. Bu sınıf hareket, rotasyon ve saldırı cooldown gibi ortak mekanikleri içerir. Abstract olarak tanımlanması, doğrudan kullanılmasını engeller ve alt sınıfların belirli metodları implement etmesini zorunlu kılar.

Hareket sistemi Rigidbody tabanlıdır. FixedUpdate döngüsünde çalışan HandleMovement metodu, oyuncunun input değerlerini okuyarak fizik motoruna velocity atar. Bu yaklaşım, transform manipülasyonuna göre daha tutarlı çarpışma davranışı sağlar.

Karakter yönelimi fare pozisyonuna göre belirlenir. HandleRotation metodu, kameradan zemin düzlemine bir ray göndererek farenin işaret ettiği dünya koordinatını hesaplar ve karakteri bu noktaya döndürür. Quaternion.Slerp kullanılarak yumuşak bir dönüş sağlanır.

Savaşçı (MeleeCharacter)

Yakın dövüş karakteri, Physics.OverlapSphere kullanarak belirlenen yarıçap içindeki düşmanları tespit eder. Ancak her tespit edilen düşmana hasar verilmez; Vector3.Angle ile karakterin ileri yönü ve düşman arasındaki açı hesaplanır, sadece belirlenen açı içindeki hedefler vurulur. Varsayılan olarak bu açı doksan derecedir.

Özel saldırı olan Spin Attack, açı kontrolü olmaksızın çevredeki tüm düşmanlara hasar verir. Bu saldırı daha geniş bir yarıçapa sahiptir ancak verdiği hasar birincil saldırının yarısı kadardır.

Nişancı (RangedCharacter)

Uzak mesafe karakteri, her sol tıklamada bir Projectile nesnesi instantiate eder. Mermi, karakterin baktığı yönde düz bir çizgide ilerler. Projectile sınıfının Initialize metoduna hasar, hız ve ömür değerleri parametre olarak geçilir.

Özel saldırı olan Burst Fire, coroutine kullanılarak implement edilmiştir. Üç mermi art arda atılır ve her biri farklı bir açıda yayılır. Bu yayılma açısı, burstSpreadAngle değişkeni ile kontrol edilir ve fan şeklinde bir atış deseni oluşturur.

Tuzakçı (TrapperCharacter)

Bu karakter iki farklı yerleştirilebilir nesne kullanır. Sol tık ile diken tuzağı, sağ tık ile patlayıcı konumlandırılır. Her iki nesne tipi için maksimum aktif sayı sınırı bulunur, bu limite ulaşıldığında yeni yerleştirme yapılamaz.

Patlayıcılar varsayılan olarak belirli bir süre sonra patlar. Ancak oyuncu E tuşuna basarak tüm aktif patlayıcıları anında tetikleyebilir. Bu özellik, stratejik zamanlama gerektiren oyun durumlarında avantaj sağlar.

5. Düşman Sistemi

Yapay Zeka: Enemy

Düşman hareketi Unity'nin NavMeshAgent bileşeni ile sağlanır. Bu sistem, önceden bake edilmiş navigasyon mesh üzerinde yol bulma algoritması çalıştırır ve engelleri otomatik olarak dolaşır.

Düşman davranışı basit bir durum makinesi mantığıyla çalışır. Update döngüsünde oyuncuya olan mesafe hesaplanır. Eğer bu mesafe saldırı menzilinden küçükse hareket durdurulur ve saldırı başlatılır. Aksi halde SetDestination metodu ile oyuncunun pozisyonuna doğru hareket devam eder.

Hasar alındığında kısa süreli görsel geri bildirim verilir. DamageFlash coroutine'i, düşmanın materyalini anlık olarak kırmızıya çevirir ve ardından orijinal renge döndürür. Can sıfırın altına düştüğünde Die metodu tetiklenir; bu metod GameManager'a puan ekler ve belirli bir olasılıkla güç artırıcı düşürür.

Spawn Sistemi: EnemySpawner

Düşman oluşturma sistemi dalga bazlı çalışır. SpawnWaveRoutine coroutine'i sonsuz bir döngüde spawn işlemini yönetir. Her dalga arasındaki bekleme süresi ve dalga başına düşman sayısı, GameManager'dan alınan level bilgisine göre dinamik olarak hesaplanır.

Spawn pozisyonları oyuncunun etrafında dört yönden (kuzey, güney, doğu, batı) rastgele seçilir. Ancak ham pozisyon doğrudan kullanılmaz; NavMesh.SamplePosition metodu ile en yakın geçerli NavMesh noktası bulunur. Bu sayede düşmanların engellerin içinde veya harita dışında spawn olması önlenir.

6. Savaş Mekanikleri

Mermi Sistemi

Projectile sınıfı, ateşlendikten sonra bağımsız hareket eden nesneleri temsil eder. Update döngüsünde pozisyon, başlangıçta belirlenen yön vektörü ve hız çarpılarak güncellenir. Collider trigger modunda çalışır ve OnTriggerEnter ile çarpışmalar algılanır.

Çarpışma kontrolünde önce hedefin tag'i kontrol edilir. Oyuncu tag'ine sahip nesneler, diğer mermiler ve trigger collider'lar göz ardı edilir. Düşman tag'i tespit edildiğinde hedefin Enemy bileşeni alınır ve TakeDamage metodu çağrılır.

Tuzak Sistemi

Trap sınıfı, üzerinden geçen düşmanlara tekrarlayan hasar verir. OnTriggerEnter tetiklendiğinde bir coroutine başlatılır. Bu coroutine, düşman tuzak alanında kaldığı sürece belirli aralıklarla hasar uygulamaya devam eder. İlk hasar anında verilirken, sonraki hasarlar daha düşük miktardadır.

Patlayıcı Sistemi

Explosive sınıfı iki tetikleme modunu destekler: zamanlayıcı ve manuel. Varsayılan olarak spawn anından itibaren bir geri sayım başlar; bu süre dolduğunda Detonate metodu otomatik çağrılır. Alternatif olarak, Tuzakçı karakterin özel yeteneği ile erken tetikleme mümkündür.

Patlama hasarı mesafeye bağlı olarak azalır. Patlama merkezindeki düşmanlar tam hasar alırken, yarıçapın kenarındakiler minimum hasar alır. Bu hesaplama, mesafenin yarıçapa oranı üzerinden hesaplanır.

7. Harita Oluşturma

MapGenerator sınıfı, oyun başlangıcında prosedürel olarak harita içeriğini oluşturur. Üç farklı tema desteklenir ve her tema kendine özgü engel tiplerini barındırır.

Oluşturma Süreci

Harita oluşturma beş aşamada gerçekleşir. İlk olarak zemin düzlemi oluşturulur; bu büyük bir küp primitive'idir ve harita boyutlarına göre ölçeklendirilir. İkinci aşamada engeller rastgele pozisyonlara yerleştirilir. Üçüncü aşamada tepeler eklenir; bunlar küre primitive'lerinin yarısının zemine gömülmesiyle oluşturulur.

Dördüncü aşamada haritanın dört kenarına duvarlar eklenir. Bu duvarlar oyuncunun oyun alanı dışına çıkmasını engeller. Son aşamada NavMeshSurface bileşeni kullanılarak navigasyon mesh'i runtime'da bake edilir. Bu sayede düşmanlar oluşturulan engelleri hesaba katarak yol bulabilir.

Tema Farklılıkları

Orman temasında engeller silindir gövde ve küre yapraklardan oluşan ağaç formundadır. Çöl temasında engeller rastgele rotasyona sahip küp kayalardır. Şehir temasında ise engeller farklı yüksekliklerde dikdörtgen binalardır. Her tema ayrıca kendine özgü zemin rengine sahiptir.

8. Kullanıcı Arayüzü

Ana Menü

MainMenuManager sınıfı üç panel arasında geçişi yönetir. Ana panel oyunu başlatma, yardım ve çıkış seçeneklerini sunar. Oyun paneli karakter ve harita seçimini içerir. Seçimler yapıldığında değerler GameSettings static sınıfına kaydedilir ve sahne geçişinde korunur.

Oyun İçi Arayüz

UIManager sınıfı Singleton olarak implement edilmiştir ve GameManager'ın eventlerine abone olur. Skor, can veya level değiştiğinde ilgili event handler'lar tetiklenir ve UI elementleri güncellenir.

Sağlık barı, can yüzdesine göre renk değiştirir. Yüzde altmışın üzerinde yeşil, yüzde otuz ile altmış arasında sarı, altında ise kırmızı renk kullanılır. Bu görsel geri bildirim, oyuncunun sağlık durumunu hızlıca değerlendirmesine yardımcı olur.

Cooldown göstergeleri, saldırı hazır durumunu görsel olarak ifade eder. Her saldırı tipi için dairesel bir fill image kullanılır; saldırı yapıldığında boşalır ve cooldown süresi boyunca dolarak hazır duruma geçer.

9. Sistem Entegrasyonu

Veri Akışı

Sistemler arası iletişim üç mekanizma üzerinden sağlanır. Birincisi, GameManager ve UIManager gibi Singleton sınıflara doğrudan Instance özelliği üzerinden erişimdir. İkincisi, C# event sistemi kullanılarak loosely coupled iletişimdir. Üçüncüsü, GameSettings gibi static sınıflar aracılığıyla sahneler arası veri taşımadır.

Oyun Döngüsü

Tipik bir oyun döngüsünde şu akış gerçekleşir: Oyuncu karakteri hareket ettirir ve saldırır. Saldırı düşmana isabet ederse, düşmanın TakeDamage metodu çağrılır. Düşman ölürse, GameManager'a puan eklenir ve event tetiklenir. UIManager bu eventi dinler ve skor göstergesini günceller.

Eş zamanlı olarak EnemySpawner belirli aralıklarla yeni düşmanlar oluşturur. Düşmanlar spawn edildiğinde oyuncuyu hedef alır ve saldırı menziline girene kadar takip eder. Saldırı başarılı olursa GameManager'ın TakeDamage metodu çağrılır ve UI buna göre güncellenir.

Özet

Bileşen	Sorumluluk
GameManager	Oyun durumu ve zorluk yönetimi
CharacterSelector	Karakter oluşturma
PlayerBase	Ortak karakter mekanikleri
Enemy	Düşman yapay zekası
EnemySpawner	Dalga bazlı düşman üretimi
MapGenerator	Prosedürel harita oluşturma
UIManager	Arayüz güncelleme
Combat sınıfları	Saldırı nesneleri