THIRD PERSON SHOOTER (TPS) VE YAPAY ZEKA (AI) PROJESİ RAPORU

EMİRCAN DEMİR, 241307109 (2. GRUP)

1. GİRİŞ VE PROJE TANIMI

Bu proje, "Yazılım Geliştirme Laboratuvarı I" dersi kapsamında, Unity oyun motoru ve C# programlama dili kullanılarak geliştirilmiş bir Üçüncü Şahıs Nişancı (Third Person Shooter - TPS) oyununu konu almaktadır. Projenin temel amacı, modern bir TPS oyununun temel mekaniklerini (karakter hareketi, nişan alma, ateş etme) ve bu mekaniklerle etkileşime giren temel bir düşman yapay zeka (AI) sistemini hayata geçirmektir.

Çalışma, oyuncu tarafından kontrol edilen bir karakter, karakteri akıllıca takip eden bir kamera sistemi (Cinemachine) ve oyuncuyu algılayıp ona göre tepki veren (örneğin saldıran veya siper alan) düşman yapay zeka birimlerini içermektedir. Proje, yazılım geliştirme yaşam döngüsünün planlama, analiz, tasarım, uygulama ve test aşamalarını kapsamaktadır.

2. LİTERATÜR TARAMASI VE KARŞILAŞTIRMA

TPS oyunları ve bu oyunlardaki yapay zeka uygulamaları, oyun geliştirme literatüründe geniş yer tutan bir konudur. Bu alandaki çalışmalar genellikle iki ana başlık altında toplanır: karakter kontrolü ve yapay zeka davranış modellemesi.

- **Örnek Çalışmalar:** Literatürde, özellikle Unity platformu için yapay zeka geliştirmede "Durum Makineleri" (State Machines) ve "Davranış Ağaçları" (Behavior Trees) öne çıkan iki temel yaklaşımdır.
    - **Durum Makineleri (Finite State Machines - FSM):** Yapay zekanın "Devriye Atma", "Takip Etme", "Saldırma" veya "Siper Alma" gibi net ve ayrık durumlara sahip olduğu sistemlerdir. Bu proje gibi daha küçük ölçekli ve net kurallara bağlı düşman davranışları için ideal ve yaygın bir yöntemdir.
    - **Davranış Ağaçları (Behavior Trees):** Daha karmaşık ve modüler yapay zeka sistemleri için kullanılır. Birçok AAA (büyük bütçeli) oyunda, FSM'lerin karmaşıklaşmasını önlemek için bu yöntem tercih edilir.

- **Çalışmanın Literatürdeki Yeri ve Karşılaştırma:** Bu proje, literatürde "eğitim amaçlı oyun projesi" kategorisinde yer almaktadır. Geliştirilen yapay zeka sistemi, büyük ölçekli ticari oyunlardaki karmaşık Davranış Ağaçları yerine, temel bir Durum Makinesi (FSM) mantığına dayanmaktadır. Bu yaklaşım, dersin hedefleri doğrultusunda hem öğrenilmesi hem de uygulanması daha yönetilebilir bir çözüm sunmaktadır. Proje, Unity'nin sunduğu NavMesh (Navigasyon Ağı) sistemi gibi yerleşik araçları kullanarak, yapay zekanın çevre içinde yolunu bulması (pathfinding) gibi temel bir literatür problemini modern araçlarla çözmektedir.

3. SİSTEM TASARIMI VE MİMARİSİ

3.1. Kullanılan Yazılımsal Mimariler ve Teknikler

Projenin geliştirilmesinde, her bir oyun nesnesi (GameObject), belirli işlevleri yerine getiren bileşenlerin (Component) bir koleksiyonudur. Örneğin, oyuncu nesnesi bir "Hareket Kontrolcüsü" bileşeni, bir "Sağlık" bileşeni ve bir "Silah Kontrol" bileşeni içerebilir.

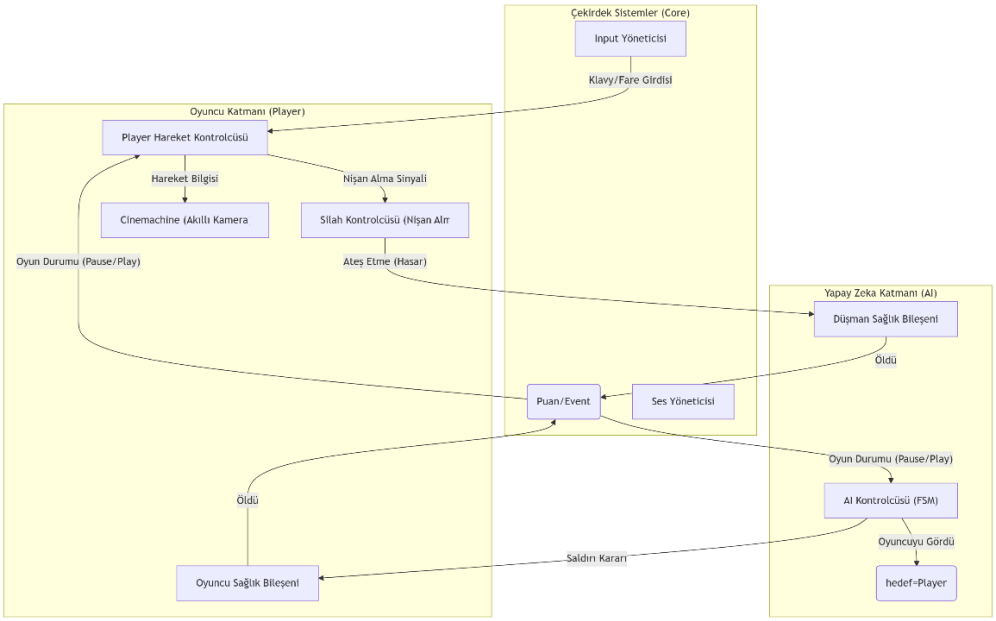
Bu temel mimariye ek olarak aşağıdaki tasarım desenleri ve prensiplerden faydalanılmıştır:

1. **Singleton (Yegane Nesne) Deseni:** GameManager, AudioManager veya UIManager gibi tüm oyun boyunca sadece bir adet olması gereken ve her yerden kolayca erişilmesi gereken sistemler için kullanılmıştır.
2. **State (Durum) Deseni:**
    - **Yapay Zeka:** Düşman yapay zekası, "Idle" (Bekleme), "Patrol" (Devriye), "Chase" (Takip) ve "Attack" (Saldırı) gibi farklı durumlara ayrılmıştır. Bu desen, her bir durumun kendi mantığını içermesini sağlayarak kod karmaşıklığını (if-else bloklarını) azaltmıştır.
    - **Oyuncu:** Oyuncu karakterinin "Normal Hareket", "Nişan Alma" (Aiming) ve "Ateş Etme" gibi farklı modları bu desenle yönetilmiştir.
3. **Observer (Gözlemci) Deseni:** Özellikle oyun içi olayların (events) yönetiminde kullanılmıştır. Örneğin, bir düşman öldüğünde "PuanSistemi"ne haber verilmesi veya oyuncunun sağlığı azaldığında "UI" (Arayüz) elementinin güncellenmesi bu desen aracılığıyla sağlanmıştır.

3.2. Sistem Şeması

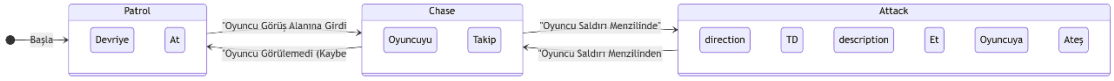
Sistemin genel şeması, üç ana katmandan oluşmaktadır:

1. **Çekirdek Sistemler (Core Systems)**:



- GameManager (Singleton): Oyunun genel akışını (Başlatma, Duraklatma, Bitirme) yönetir.
- InputManager: Oyuncu girdilerini (klavye, fare) okur ve ilgili birimlere iletir.
- AudioManager: Ses efektlerini ve müziği yönetir.

2. **Oyuncu Katmanı (Player Layer):**



- PlayerController: InputManager'dan gelen verilerle karakterin hareketini ve animasyonlarını yönetir.
- WeaponController: Nişan alma ve ateş etme mekaniklerini yönetir.
- Camera (Cinemachine): Oyuncuyu akıllı bir şekilde takip eden sanal kamera sistemi.

3. **Yapay Zeka Katmanı (AI Layer):**
   - AIController (FSM): Düşmanın Durum Makinesini (Patrol, Chase vb.) yönetir.
   - NavMesh Agent: Unity'nin yol bulma sistemini kullanarak hedefe (oyuncuya) nasıl gideceğini belirler.
   - Health/Damageable: Hem oyuncu hem de yapay zeka tarafından paylaşılan, hasar almayı sağlayan bileşen.

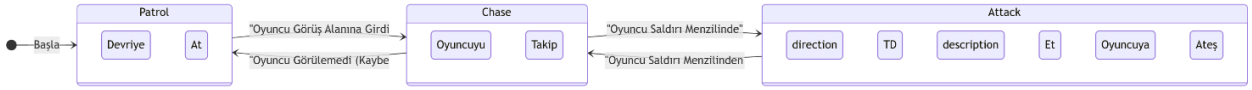
3.3. Oyun Mekanikleri Blok Diyagramı

Geliştirilen yapay zeka sistemi, bir Durum Makinesi (FSM) olarak çalışır. Aşağıdaki akış diyagramı bu mekaniği özetlemektedir:

1. **Başlangıç Durumu: [PATROL (Devriye)]**
   - Belirlenen noktalar arasında hareket et.
   - **Kontrol:** Oyuncu görüş alanına girdi mi?
     - **EVET:** Durumu [CHASE (Takip)] olarak değiştir.
     - **HAYIR:** Devriyeye devam et.

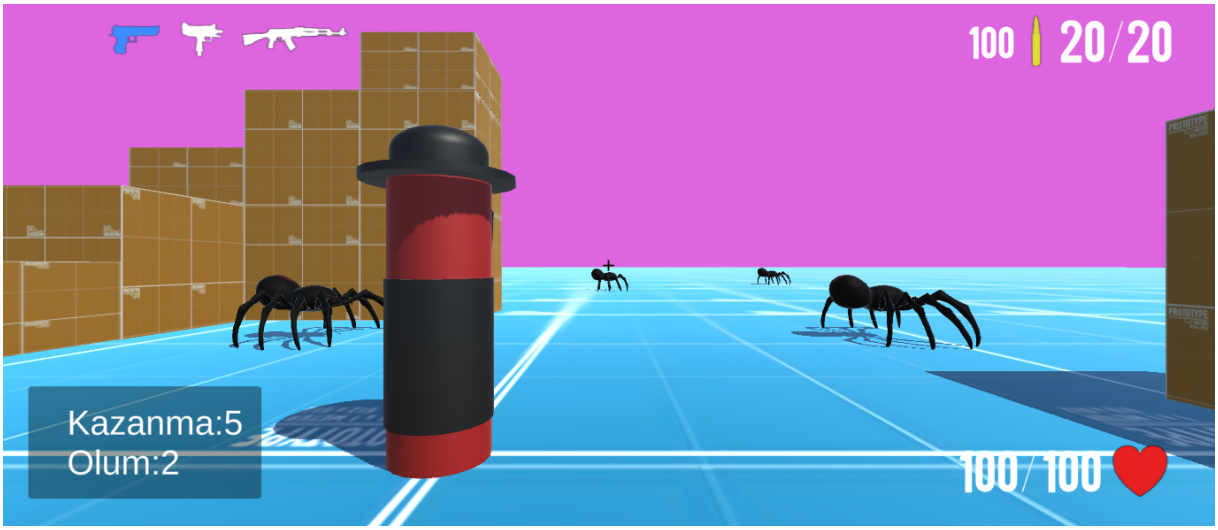
2. **Durum: [CHASE (Takip)]**
    - Oyuncunun konumuna doğru (NavMesh kullanarak) hareket et.
    - **Kontrol 1:** Oyuncu saldırı menzilinde mi?
        - **EVET:** Durumu [ATTACK (Saldırı)] olarak değiştir.
        - **HAYIR:** Takip etmeye devam et.
    - **Kontrol 2:** Oyuncu görüş alanından tamamen çıktı mı?
        - **EVET:** Son görülen konuma git, biraz bekle, sonra Durumu [PATROL] olarak değiştir.

3. **Durum: [ATTACK (Saldırı)]**
    - Dur ve oyuncuya ateş et.
    - **Kontrol:** Oyuncu saldırı menzilinden çıktı mı?
        - **EVET:** Durumu [CHASE] olarak değiştir.



3.4. Tasarlanan Sayfalar (Arayüz)

Proje, temel bir oyun döngüsünü desteklemek için aşağıdaki arayüz sayfalarını (Sahnelerini/UI Panellerini) içermektedir:



1. **Ana Menü (Main Menu):**
    - Oyunun açılış ekranı.
    - "Oyunu Başlat", "Ayarlar" (opsiyonel) ve "Çıkış" butonlarını içerir.
2. **Oyun İçi Arayüz (HUD - Heads-Up Display):**
    - Oyun esnasında ekranda görünen arayüz.
    - Oyuncunun "Sağlık Göstergesi" (Health Bar), "Mermi Sayısı" ve "Puan" (varsa) gibi kritik bilgileri gösterir.
3. **Duraklatma Menüsü (Pause Menu):**
    - Oyun sırasında (örn. 'ESC' tuşuyla) açılan menü.
    - "Devam Et", "Ana Menüye Dön" ve "Çıkış" seçeneklerini sunar.
4. **Oyun Bitti / Görev Başarısız Ekranı (Game Over Screen):**
    - Oyuncu sağlığı sıfıra düştüğünde belirir.
    - "Yeniden Başla" veya "Ana Menüye Dön" seçeneklerini sunar.

4. GELİŞTİRME SÜRECİ: ZORLUKLAR VE KAZANIMLAR

4.1. Karşılaşılan Zorluklar ve Çözümleri

- **Zorluk: Akıcı TPS Kamera Kontrolü**
    - *Problem:* Üçüncü şahıs kamerasının oyuncuyu takip ederken duvarların veya engellerin içine girmesi (clipping) ve nişan alma sırasında akıcı bir şekilde omuza yaklaşmaması.
    - *Çözüm:* Bu sorunu sıfırdan kodlamak yerine, Unity'nin sunduğu profesyonel bir çözüm olan **Cinemachine** paketi projeye dahil edilmiştir. Cinemachine'in "Virtual Camera" ve "State-Driven Camera" özellikleri kullanılarak, normal takip modu ile nişan alma modu arasında yumuşak geçişler sağlanmış ve "Collider" özelliği ile kameranın engellerin içinden geçmesi engellenmiştir.
- **Zorluk: Gerçekçi Yapay Zeka Hareketi ve Karar Verme**
    - *Problem:* Yapay zekanın oyuncuyu gördüğünde direkt olarak oyuncunun içinden geçmeye çalışması (fizik kurallarını ihlal etmesi) ve mantıksız kararlar vermesi.
    - *Çözüm:* Fiziksel hareket yerine Unity'nin **NavMesh (Navigation Mesh)** sistemi kullanılmıştır. Sahne "bake" edilerek yapay zekanın yürüyebileceği alanlar belirlenmiş ve NavMeshAgent bileşeni sayesinde yapay zekanın engellerin etrafından dolaşarak hedefine (oyuncuya) ulaşması sağlanmıştır. Karar verme mekanizması ise FSM (Durum Makinesi) deseni ile net kurallara bağlanarak çözülmüştür.

4.2. Projenin Sağladığı Kazanımlar

Bu proje, bir yazılım geliştirme öğrencisi olarak birçok önemli yetkinliğin pekiştirilmesini sağlamıştır:

1. **Teknik Kazanımlar:** Unity oyun motoru arayüzüne ve C# programlama diline olan hakimiyet artırılmıştır. Özellikle Nesne Yönelimli Programlama (OOP) prensiplerinin ve tasarım desenlerinin (State, Singleton) pratik uygulaması deneyimlenmiştir.
2. **Sistem Tasarımı:** Bir projenin sadece kod yazmaktan ibaret olmadığı; modüler, esnek ve genişletilebilir bir mimari kurmanın (Bileşen Tabanlı Mimari, FSM) projenin başarısı için ne kadar kritik olduğu anlaşılmıştır.
3. **Araç Kullanımı:** Cinemachine ve NavMesh gibi hazır araçları ve paketleri projeye entegre etme ve etkin kullanma becerisi kazanılmıştır.

5. SONUÇ

Bu raporda, Yazılım Geliştirme Laboratuvarı I dersi kapsamında geliştirilen TPS ve Yapay Zeka projesinin detayları sunulmuştur. Proje, Unity ve C# kullanarak temel TPS mekaniklerini ve basit bir yapay zeka sistemini başarıyla uygulamıştır. Literatürdeki benzer çalışmalara kıyasla eğitim odaklı bir yaklaşım benimsenmiş, Bileşen Tabanlı Mimari ve Durum Deseni gibi temel yazılım mimarileri kullanılmıştır. Geliştirme sürecinde karşılaşılan kamera ve yapay zeka zorlukları, Unity'nin sunduğu modern araçlarla (Cinemachine, NavMesh) aşılmıştır. Bu proje, teorik yazılım geliştirme bilgilerinin pratik bir oyun projesi üzerinde uygulanması açısından önemli kazanımlar sağlamıştır.