T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

PROJE KONUSU:Star Wars Temalı Labirent Oyunu

ÖĞRENCİ ADI:YERDİNAT ALİKHAN
ÖĞRENCİ NUMARASI:220502050

DERS SORUMLUSU:
PROF. DR./DR. ÖĞR. ÜYESİ FULYA AKDENİZ

TARİH:05.04.2025

1 GİRİŞ
1.1 Projenin amacı

Bu proje, **Star Wars** temasıyla tasarlanmış bir labirent oyunu aracılığıyla nesne yönelimli programlama (OOP) prensiplerini, dinamik yol bulma algoritmalarını ve grafiksel kullanıcı arayüzü (GUI) geliştirme becerilerini uygulamalı olarak göstermeyi amaçlamaktadır. Oyuncu, seçtiği bir karakterle labirentte hareket ederek hedefe ulaşmaya çalışırken, farklı yeteneklere sahip kötü karakterlerden kaçınmak zorundadır. Proje kapsamında:

- **Algoritma ve Veri Yapıları:** En kısa yol bulma algoritmaları (BFS, Dijkstra) ile karakterlerin zekâ davranışlarının simüle edilmesi,
- **OOP Tasarımı:** Kalıtım, çok biçimlilik ve soyutlama gibi OOP prensiplerinin karakter hiyerarşisinde kullanılması,
- **Kullanıcı Deneyimi:** Pygame kütüphanesiyle etkileşimli ve görsel bir oyun deneyimi oluşturulması hedeflenmiştir.

Projede Gerçekleştirilmesi Beklenenler

Aşağıdaki maddeler, projenin teknik ve fonksiyonel çıktılarını özetlemektedir:

1. Karakter Tabanlı Oyun Mekaniği

- **İyi Karakter Seçimi:** Kullanıcının **Luke Skywalker** (3 can) veya **Master Yoda** (6 can) arasından seçim yapması,
- **Kötü Karakterlerin Yetenekleri:**
  - **Darth Vader:** Dijkstra algoritmasıyla duvarları yok sayarak hareket,
  - **Kylo Ren:** İki adımlı BFS ile hızlı ilerleme,
  - **Stormtrooper:** Klasik BFS ile tek adımlı takip,
- **Can Yönetimi:**
  - Luke'un her çarpışmada 1 can, Master Yoda'nın 1 can (yarım hasar mantığı) kaybetmesi.

2. Dinamik Harita ve Yol Bulma

- **Harita Yükleme:** harita.txt dosyasından sabit labirent yapısının ve karakter konumlarının okunması,
- **Algoritma Performansı:** Kötü karakterlerin her hamlede en kısa yolu yeniden hesaplaması ve ekranda mesafe bilgisinin gösterilmesi.

3. Grafiksel Arayüz Tasarımı

- **Pygame Entegrasyonu:** Labirentin hücreler, karakterler ve kapılar ile görselleştirilmesi,

- **Etkileşimli Kontroller:** Klavye ok tuşlarıyla karakter hareketi,
- **Gerçek Zamanlı Feedback:**
    - Can barı ile can durumunun takibi,
    - Kötü karakterlerin mesafe bilgilerinin anlık gösterimi.

4. Nesne Yönelimli Programlama (OOP)

- **Sınıf Hiyerarşisi:**
    - Temel Karakter sınıfından türetilen LukeSkywalker, MasterYoda, DarthVader, KyloRen, Stormtrooper alt sınıfları,
    - Lokasyon sınıfıyla koordinat yönetimi,
- **Polimorfizm:** Kötü karakterlerin enKisaYol() metodunu kendi algoritmalarıyla ezmesi.

5. Hata Yönetimi ve Test

- **Dosya Okuma Kontrolleri:** Harita dosyası bulunamazsa varsayılan haritanın yüklenmesi,
- **Sınır Kontrolleri:** Karakterlerin labirent dışına çıkmasının engellenmesi,
- **Çarpışma Testleri:** Karakterlerin aynı hücrede bulunması durumunda can kaybı ve oyun sonu mekaniği.

6. Kullanıcı Dokümantasyonu

- **Başlangıç Menüsü:** Karakter seçimi, kontroller ve yardım ekranları,
- **Oyun Sonu Ekranı:** Kazanma/kaybetme durumunda animasyonlu geri bildirim.

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

1. Kullanıcı Arayüzü Gereksinimleri

1. **Grafiksel Menü Sistemi**
    - Başlangıç menüsünde karakter seçimi (Luke Skywalker veya Master Yoda) yapılabilmeli.
    - Kontrol talimatları ve yardım ekranlarına erişim sağlanmalı.
2. **Labirent Görselleştirme**
    - Sabit haritanın **14x11** hücre boyutunda grafiksel olarak gösterilmesi.
    - Renk kodları:
        - Duvarlar: **Siyah**, Yollar: **Açık Gri**, Başlangıç: **Sarı**, Hedef: **Yeşil Yıldız**, Kapılar: **Mavi**.
    - Karakterlerin konumlarının gerçek zamanlı güncellenmesi.
3. **Etkileşimli Kontroller**

- Klavye ok tuşlarıyla (↑, ↓, ←, →) iyi karakterin hareket ettirilmesi.
- **ESC** tuşuyla menüye dönme veya oyunu duraklatma özelliği.

4. **Gerçek Zamanlı Bilgi Paneli**
   - İyi karakterin can durumunu gösteren **can barı** (Luke: Kırmızı, Yoda: Yeşil).
   - Kötü karakterlerin iyi karaktere olan **mesafe bilgisinin** dinamik olarak gösterilmesi.
5. **Animasyon ve Ses Efektleri**
   - Karakter hareketlerinde ses efekti (hareket.wav).
   - Çarpışma anında **kırmızı flaş** efekti ve hasar sesi (hasar.wav).
   - Oyun sonu animasyonları (kazanma: yeşil gradyan, kaybetme: kırmızı gradyan).

2.2 Fonksiyonel gereksinimler

---

1. Karakter Yönetimi

1. **Karakter Seçimi:**
   - Kullanıcı, başlangıçta Luke Skywalker (3 can) veya Master Yoda (6 can) karakterlerinden birini seçebilmelidir.
2. **Karakter Hareketi:**
   - İyi karakter, klavye ok tuşları (↑, ↓, ←, →) ile duvarlara çarpmadan labirentte hareket ettirilebilmelidir.
3. **Can Mekaniği:**
   - Luke Skywalker, kötü karakterle çarpıştığında 1 can kaybetmeli.
   - Master Yoda, çarpışmada 1 can kaybetmeli (toplam 6 can hakkı olduğundan yarı hasar simüle edilir).
   - Canı 0'a düşen karakter için otomatik olarak Game Over ekranı gösterilmeli.

---

2. Düşman Davranışları

1. **Dinamik Takip:**
   - Stormtrooper: BFS algoritması ile tek adımlı en kısa yolu hesaplayarak hareket etmeli.
   - Kylo Ren: İki adımlı BFS ile her hamlede iki hücre ilerlemeli, ara hücrede duvar varsa hareket edememeli.
   - Darth Vader: Dijkstra algoritması ile duvarları yok sayarak en kısa yolu bulmalı.

2. **Gerçek Zamanlı Mesafe Hesaplama:**
    - Her kötü karakterin iyi karaktere olan anlık mesafesi (adım sayısı) ekranda gösterilmeli.

---

3. Oyun Mekaniği

1. **Labirent Yapısı:**
    - Harita, 14x11 hücre boyutunda sabit bir txt dosyasından (harita.txt) yüklenmeli.
    - Başlangıç noktası (5,6) ve hedef (4,10) konumları belirgin şekilde işaretlenmeli.
2. **Hedefe Ulaşma:**
    - İyi karakter hedefe ulaştığında otomatik olarak kazanma ekranı gösterilmeli.
3. **Çarpışma Kontrolü:**
    - İyi ve kötü karakterler aynı hücrede buluştuğunda can kaybı tetiklenmeli.

---

4. Grafiksel ve Ses Özellikleri

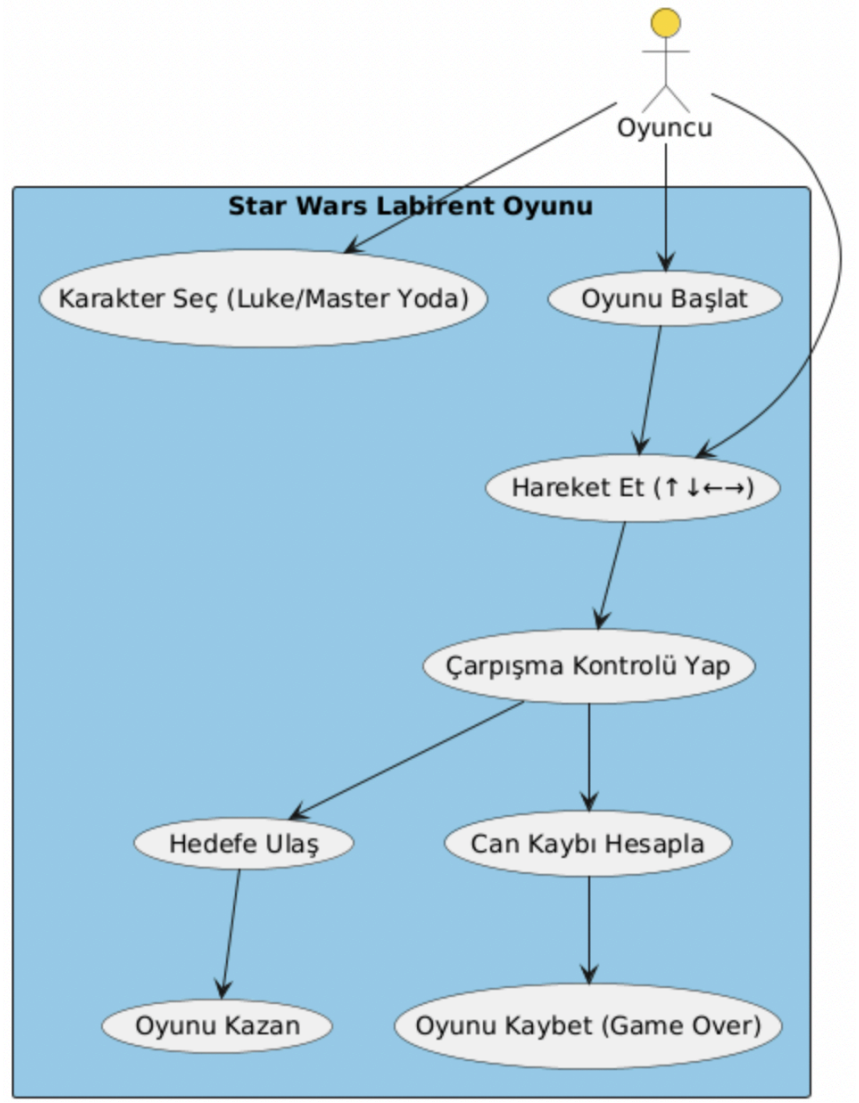
1. **Görsel Tasarım:**
    - Labirent hücreleri renklerle kodlanmalı:
        - Duvarlar: Siyah, Yollar: Açık Gri, Başlangıç: Sarı, Hedef: Yeşil Yıldız, Kapılar: Mavi Ok.
    - Karakterlerin konumları gerçek zamanlı güncellenmeli.
2. **Ses Efektleri:**
    - Karakter hareketi, çarpışma ve oyun sonu durumlarında ses dosyaları çalınmalı.

---

5. Veri ve Hata Yönetimi

1. **Harita Yükleme:**
    - harita.txt dosyası bulunamazsa varsayılan harita otomatik oluşturulmalı.
2. **Hata Kontrolleri:**
    - Karakterlerin labirent dışına çıkması engellenmeli.
    - Geçersiz dosya formatı veya eksik veri durumunda hata mesajı gösterilmeli.

2.3 Use-Case diyagramı



3 TASARIM

3.1 Mimari tasarım

Mimari Bileşenler:

1.Kullanıcı Arayüzü Katmanı (Presentation Layer):

Pygame Tabanlı GUI: Labirentin grafiksel gösterimi, karakterlerin animasyonları ve menü sistemi.

Girdi Yönetimi: Klavye ve fare etkileşimlerinin işlenmesi.

2.Uygulama Mantığı Katmanı (Application Layer):

Oyun Motoru: Oyun döngüsü, çarpışma kontrolü ve olay yönetimi.

Algoritma Yöneticisi: Kötü karakterlerin yol bulma algoritmalarının

çalıştırılması (BFS, Dijkstra).

3.Veri ve İş Katmanı (Domain Layer):

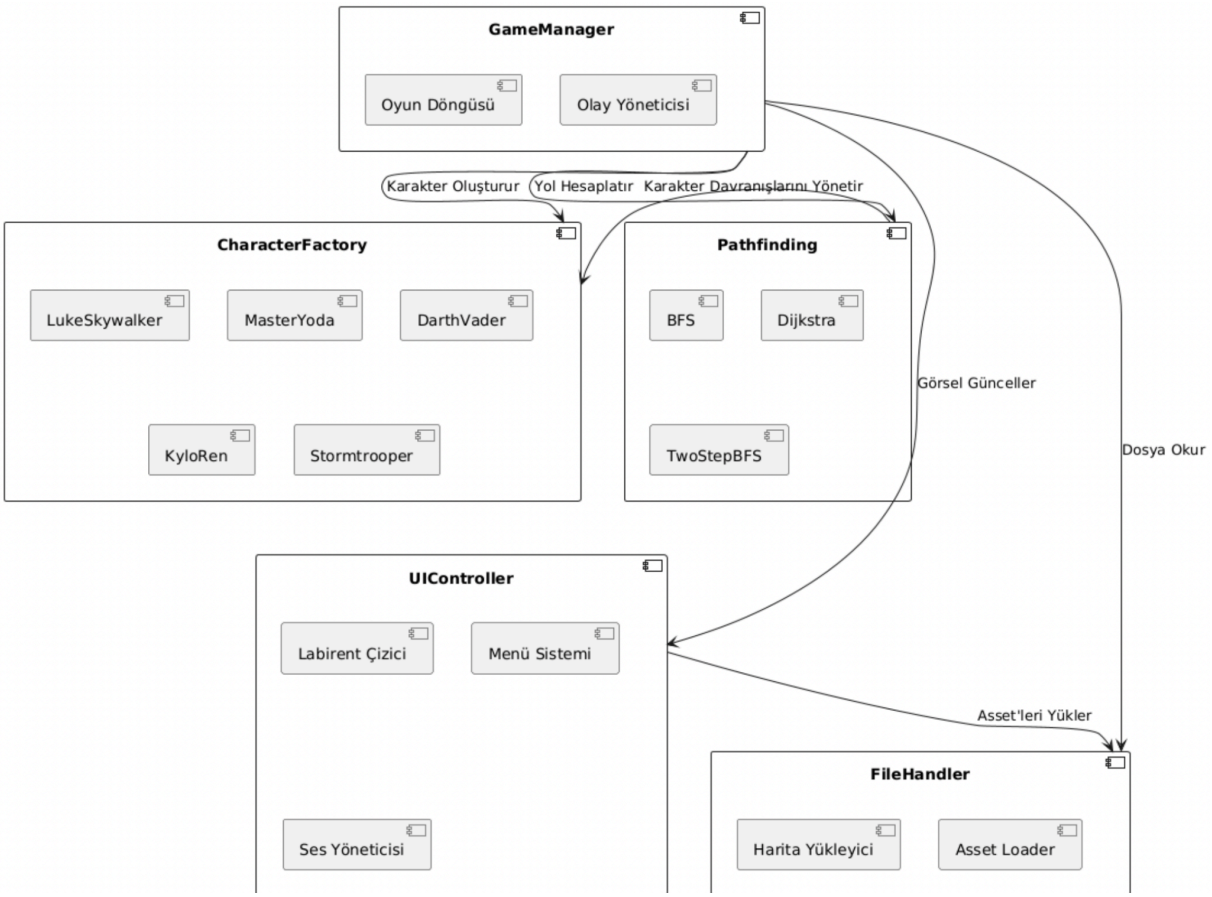
Karakter Sınıfları: İyi ve kötü karakterlerin davranışlarını yöneten sınıflar.

Harita Yükleyici: harita.txt dosyasının okunması ve labirent veri yapısının oluşturulması.

4.Veri Depolama Katmanı (Data Layer):

Dosya İşlemleri: Harita ve ses dosyalarının diskten okunması.

MODUL DİYAGRAMI:



3.2 Kullanılacak teknolojiler

1. Programlama Dili

- **Python 3.10+**: Projenin temel programlama dili olarak Python kullanılmıştır. Tercih sebebi, nesne yönelimli programlama (OOP) desteği, geniş kütüphane ekosistemi ve hızlı prototipleme imkânıdır.

2. Harici Kütüphaneler

1. **Pygame**:
    - **Kullanım Amacı**: Grafiksel kullanıcı arayüzü (GUI), karakter animasyonları, ses efektleri ve kullanıcı girdi yönetimi için kullanılmıştır.
    - **Versiyon**: 2.5.2+.
    - **Özellikler**: Pencerelerin oluşturulması, sprite yönetimi, çarpışma algılama.
2. **Heapq** (Standart Kütüphane):
    - **Kullanım Amacı**: Dijkstra algoritmasında öncelik kuyruğu (priority queue) yönetimi için kullanılmıştır.
3. **Collections** (Standart Kütüphane):
    - **Kullanım Amacı**: BFS algoritmasında kuyruk (deque) veri yapısının oluşturulması.

4. Veri Yapıları

- **Matris (2D Liste)**: Labirent yapısının saklanması.
- **Graf (Graph)**: Kötü karakterlerin yol bulma algoritmaları için soyut veri yapısı olarak kullanılmıştır.