*2. Grup Yapay Zeka Projesi*

Bu proje, bir trafik simülasyonu ortamında çeşitli yol bulma algoritmalarını entegre ederek gerçek dünya koşullarında algoritmaların performansını incelemeyi amaçlamaktadır. Proje, *Python programlama dili* ve *çeşitli yardımcı kütüphaneler* kullanılarak geliştirilmiştir. **Kullanıcı etkileşimli bir arayüz üzerinden algoritmaların seçimi, çalıştırılması ve analiz edilmesi sağlanmıştır**. Simülasyon, dinamik trafik ışıkları, yayalar ve tümsekler gibi çevresel etkileri içererek ***algoritmaların bu durumlara karşı davranışlarını*** gözlemlemeye olanak tanımıştır.

**Kullanılan Teknolojiler ve Kütüphaneler**

1. **Programlama Dili: Python**
   * **Neden Python? Python, hem hızlı prototipleme hem de geniş kütüphane desteği sayesinde simülasyon geliştirme süreçlerinde tercih edilmektedir.**
2. **Kütüphaneler:**
   * **Pygame:**
     + **Kullanım Alanı: Grafiksel arayüz ve animasyonların oluşturulması.**
     + **Trafik ışıklarının, araçların ve haritanın dinamik görselleştirilmesinde kullanılmıştır.**
   * **Heapq:**
     + **Kullanım Alanı: Öncelik sırası olan veri yapılarını yönetmek.**
     + **A\* ve Dijkstra algoritmalarında öncelikli düğüm seçimi için kullanılmıştır.**
   * **Random:**
     + **Kullanım Alanı: Trafik ışıklarının başlangıç durumları ve sürelerinin rastgele belirlenmesi.**
     + **Tümseklerin ve yayaların rastgele konumlandırılmasında da kullanılmıştır.**
   * **Dataclasses:**
     + **Kullanım Alanı: Algoritmalarda kullanılan düğüm veri yapılarının tanımlanması.**
     + **Kodun daha okunabilir ve düzenli hale getirilmesini sağlamıştır.**

**Geliştirme Sürecinin Detayları**

1. **Simülasyon Ortamının Oluşturulması**

* **2D bir ortam hazırlanmıştır. Projenin ekipçe verimli olması için dosyalara ayrılmıştır. Bu dosyalar şu şekildedir:**

**~ bump.py: Görseldeki tümsekleri temsil eder.**

**~ cars.py: Araçları temsil eder.**

**~ grid\_map.py: Haritanın koordinatları ve grid yapısı buradan oluşturulmuştur.**

**~ pedestrian.py: Yayaları temsil eder.**

* **Ana Bileşenler ve İşlevler:**
* **Pygame Ayarları:**
  + Simülasyon ekranı, genişlik, yükseklik ve yazı tipi gibi genel ayarlar tanımlanır.
  + Ekran arka plan rengi ve başlık belirlenir.
* **Harita ve Trafik Işıkları:**
  + Harita görseli yüklenir ve simülasyona ölçeklendirilmiş olarak yerleştirilir.
  + Trafik ışıkları koordinatları ve özellikleriyle tanımlanır. Trafik ışıkları arasında eşleşme yapılır (örneğin, bir sinyalin kırmızı ise diğeri yeşil olabilir).
* **Tümsekler ve Yayalar:**
  + Rastgele konumlar seçilerek tümsekler ve yayalar eklenir.
  + Döndürme (rotate) gibi özel durumlar kontrol edilir ve görsel ayarlamalar yapılır.
* **Oyun Akışı:**
  + **Duraklatma:** "Space" tuşuyla simülasyon duraklatılabilir ve duraklatma durumunda ekranda "Paused" yazısı görüntülenir.
  + **Ana Döngü:** Simülasyon, her karede (frame) ekranı güncelleyerek tüm unsurları çizer ve animasyonları çalıştırır.
  + Tüm bileşenlerin çarpışma, zamanlayıcı ve hareket gibi işlevleri sürekli güncellenir.
* **Arabalar ve Hareket:**
  + Arabalar belirli koordinatlara göre eklenir ve görselleştirilir.
  + Gelecekte araba hareketleri için hız ve yön bilgileri kolaylıkla eklenebilir.

***Şimdi Kullanılan Fotoğraflar İçin:***

A map of a road

Description automatically generated

1. **Algoritmaların Seçimi ve Uygulanması**

**Trafik simülasyonunda beş farklı yol bulma algoritması uygulanmıştır:**

* ***A Algoritması:*\***
  + **Heuristic (sezgisel) ve maliyet fonksiyonlarını birleştirerek en kısa ve verimli yolu bulur.**
  + **Manhattan mesafesi sezgisel fonksiyonu olarak kullanılmıştır.**
* **Dijkstra Algoritması:**
  + **Tüm düğümleri eşit öncelikte değerlendirerek garantili en kısa yolu bulur.**
  + **Özellikle maliyetlerin değişmediği durumlarda etkili sonuçlar üretmiştir.**
* **Greedy Best-First Search:**
  + **Hedefe olan en kısa sezgisel mesafeyi temel alır. Ancak her zaman küresel optimuma ulaşmayabilir.**
* **Breadth-First Search (BFS):**
  + **Her seviyeyi sırayla değerlendirerek hedefe ulaşır.**
  + **Uniform maliyet varsayımları altında etkili bir çözüm sunar.**
* **Depth-First Search (DFS):**
  + **Derinlemesine arama stratejisiyle çalışır ancak optimal çözüm garantisi yoktur.**

**Her algoritma, grid tabanlı bir harita üzerinde bağımsız modüller halinde geliştirilmiş ve arayüze entegre edilmiştir.**

1. **Çevresel Faktörlerin Entegrasyonu**

**Gerçek dünya koşullarını simüle etmek için trafik ışıkları, yayalar ve tümsekler gibi dinamik faktörler simülasyona dahil edilmiştir:**

* **Trafik Işıkları:**
  + **Rastgele kırmızı veya yeşil olarak başlatılmıştır.**
  + **Işık süreleri rastgele 10-15 saniye arasında belirlenmiştir.**
  + **Her ışığın yalnızca belirli bir etki alanına sahip olması sağlanmıştır. Örneğin, bir ışık yalnızca bulunduğu konumun solundaki iki hücreyi etkiler.**
* **Tümsekler:**
  + **Rastgele yerleştirilen hız kesici bölgeler olarak tanımlanmıştır.**
  + **Bu alanlar araçların hızını düşürerek algoritmaların maliyet hesaplamalarını değiştirmiştir.**
* **Yayalar:**
  + **Harita üzerinde rastgele hareket eden yayalar oluşturulmuştur.**
  + **Araçlar, yayaların konumlarına göre durmak zorunda kalmıştır.**

1. **Arayüz Geliştirme ve Kullanıcı Etkileşimi**

**Projenin kullanıcı dostu bir arayüzle sunulması, algoritmaların çalışma mantığını görselleştirmek için kritik bir adımdı:**

* **Butonlar:**
  + **Her bir algoritmayı başlatan ayrı butonlar tasarlanmıştır.**
  + **Kullanıcı, bir algoritmayı seçip aracı hareket ettirebilir.**
* **Harita ve Trafik Işıkları Görselleştirmesi:**
  + **Harita, grid tabanlı olarak çizilmiş ve araçların, trafik ışıklarının ve çevresel faktörlerin konumları kullanıcıya açıkça gösterilmiştir.**
  + **Işıkların durumları (kırmızı/yeşil) gerçek zamanlı olarak yansıtılmıştır.**

1. **Teknik Detaylar**

* **Modüler Kodlama:**
  + **Algoritmalar, trafik ışıkları, araçlar ve harita ayrı ayrı modüller halinde kodlanmış, böylece kodun genişletilebilirliği sağlanmıştır.**
* **Performans Optimizasyonu:**
  + **Algoritmalar, grid tabanlı haritalar üzerinde test edilmiş ve her birinin performansı ölçülmüştür.**

1. **Butonlar:**

* **Algoritmaların butonlara göre hareket etmesi için “*button.py*” dosyası oluşturuldu.**
* **Bu dosya sayesinde istediğimiz algoritmalarımızı istediğimiz zaman yönlendirebiliyoruz.**

1. **Veri Analizi:**

Bu projede kullanılan *Dijkstra, Breadth First Search(BFS), Greedy, A\* ve Depth First Search(DFS) algoritmaları* kullanılmıştır. **Her birisi 10 kez çalıştırılarak** algoritmaların varış noktasına ulaştıkları süreler, karşılaştıkları engeller ve kullandıkları süreler not edilip veri seti oluşturulmuştur. Bu veri seti ***Python*** dili kullanılarak analiz edildi.

***Veri Setinin Analiz Edilme Süreci:***

1. *Öncelikle Pandas ve Matplotlib kütüphaneleri ekliyoruz. Pandas istatistiksel işlemler yapmamız için gerekli iken Matploblib grafik oluşturmak için gereklidir.*



*Şekil 1. Gerekli Kütüphaneler*

1. *Veri setinin okunmasını gerçekleştirdikten sonra verilerimizi görebiliriz.*

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 2. Veri Okuma

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 3. Algoritmaların Verileri

1. *Sonrasında kullanılan algoritmaları, kullanılan yolları ve kaç farklı yol kullanıldığının çıktısı alındı.*

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4. Algoritmalar, Yollar ve Sayılarının Verisi

1. Kullanılan algoritmaların süre ortalamaları alındı. Süre Ortalaması **En Düşük Algoritma** 9.08 ile ***'Greedy Algoritması'*** çıkmıştır.

metin, elektronik donanım, ekran görüntüsü, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 5. Algoritma Süre Ortalamaları Tablosu

1. Daha sonrasında algoritmaların seçtikleri yollarda ne kadar engelle karşılaştıkları bulundu. **En Az Engelli Yola Giden Algoritma** 29 ile ***'A\* Algoritması'*** çıkmıştır.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 6. Algoritmaların Karşılaştığı Engeller Tablosu

1. **Son olarak**, süre ortalamalarının bir *çizgi grafiği* oluşturuldu.

metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 7. Algoritma Ortalama Süreleri Çizgi Grafiği

Sonuç ve Çıkarımlar

**Projenin sonunda, trafik simülasyonu gerçek dünya koşullarını modelleyen bir test ortamı sunmuştur. Algoritmalar, çevresel faktörlerin etkisi altında performans açısından incelenmiştir. Özellikle:**

* ***A Algoritması*\*, hem zaman hem de maliyet açısından en optimal sonuçları üretmiştir. Bu projemizde A\* Algoritması, en verimli algoritmalarda 1.sıradadır.**
* **BFS Algoritması, doğru ancak nispeten yavaş bir çözüm sunmuştur. Projemizde bu algoritma, 5.sıradadır.**
* **DFS Algoritması, ortalama süreler göz önüne alınarak en verimli 3. Algoritma olmuştur.**
* **Greedy ve Dijkstra, farklı durumlara karşı farklı avantajlar göstermiştir. Greedy en hızlı algoritma olmasına karşın A\* Algoritmasına göre daha fazla engeli seçmiştir bu nedenle 2.sıradadır. Dijkstra ise 4.sıradadır.**
* **Trafik Işıkları ve Yayalar, algoritmaların performansını doğrudan etkileyen faktörler olarak öne çıkmıştır.**

**Proje, hem akademik hem de mühendislik açısından pratik bir referans ve öğrenme kaynağı oluşturmuştur.**