Özge ÇITAKOĞLU 2118121023



## YAPAY ZEKA RAPORU

### Reinforcement Learning Kullanarak Robotun Labirentte Gezinmesi

**kütahya sağlık bilimleri üniversitesi mühendislik ve doğa bilimleri fakültesi**

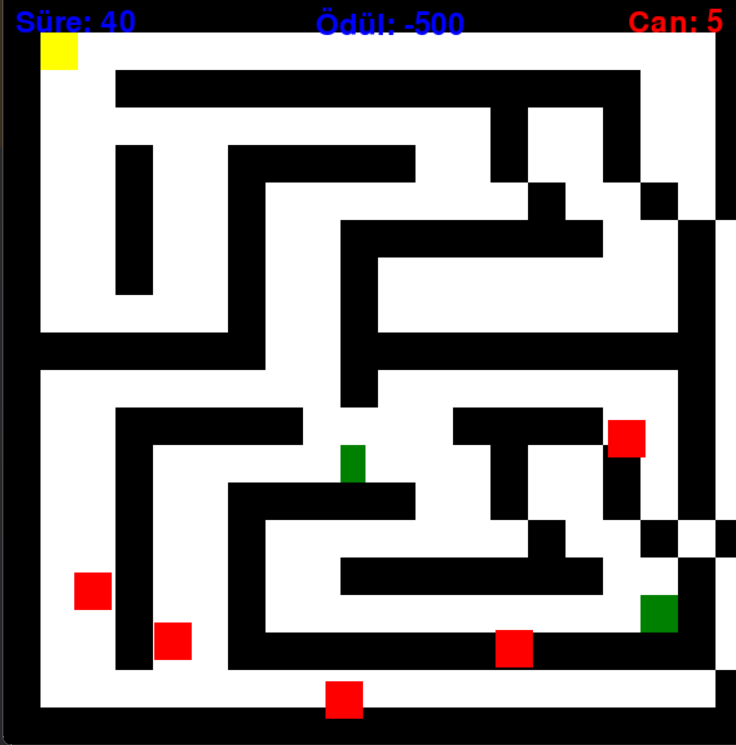
## 1. Giriş

Bu proje, A\* algoritması ve Q-Learning algoritmasının labirent çözme problemlerindeki performanslarını karşılaştırmak amacıyla gerçekleştirilmiştir. A\* algoritması, bir graf veya grid üzerinde en kısa veya en maliyet etkin yolu bulmak için kullanılan deterministik bir arama algoritmasıdır. Genellikle labirent veya haritalarda yol bulma problemleri için kullanılır. Q-learning ise bir ajanın bir ortamda nasıl hareket edeceğini öğrenmesini sağlayan pekiştirmeli öğrenme algoritmasıdır ve dinamik çevrelerde kullanılarak ajanın ödüller ve cezalar üzerinden en iyi aksiyonu seçmesini amaçlar

. **A\*Star Algoritması:**Deterministik ve planlama bazlı bir yaklaşımdır. Hedefe ulaşmak için önceden tanımlanmış bir yol bulur. Genellikle statik ve belirli bir çevrede kullanılır.

**Q-Learning:** Deneyim tabanlı ve öğrenme odaklı bir yaklaşımdır. Ajan, ödüller ve cezalar üzerinden deneyim kazanarak en iyi stratejiyi öğrenir. Dinamik ve değişen çevrelerde kullanılabilir.

Bu kod, Pygame kullanarak bir labirent oyunu oluşturur. Oyunda bir oyuncu (Q-learning algoritmasını kullanan) ve düşmanlar bulunmaktadır. Oyuncunun amacı, labirentin başlangıç noktasından bitiş noktasına ulaşmaktır.

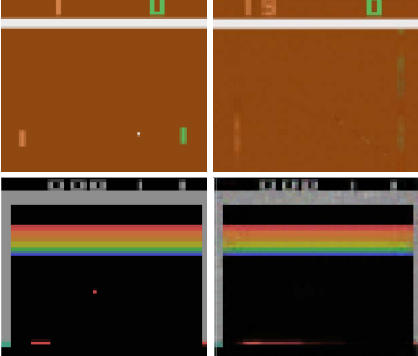
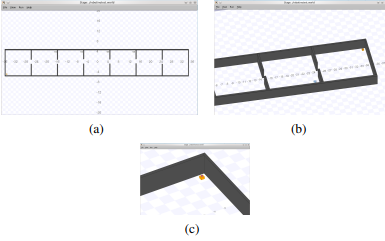


## 2. Literatür Araştırması

a) **Reinforcement Learning ile Çocuklarda Eğitsel Oyunların Geliştirilmesi:** Bu çalışmada, çocukların öğrenme süreçlerini desteklemek amacıyla eğitsel oyunların geliştirilmesinde reinforcement learning (RL) yöntemlerinin kullanımı incelenmiştir. RL algoritmaları, oyunların dinamik olarak zorluk seviyesini ayarlayarak çocukların performansını optimize eder.

b) **Yapay Zeka Destekli Öğrenme Oyunları ve Öğrenci Motivasyonu:** Bu çalışma, yapay zeka (AI) destekli öğrenme oyunlarının öğrenci motivasyonunu artırmadaki etkisini araştırmaktadır. RL, öğrencilerin eğitim materyalleriyle etkileşime geçerek öğrenme süreçlerini optimize etmeye yardımcı olabilir.

c) **Robotik Eğitiminde Güçlendirme Öğrenme Yaklaşımı:** Bu araştırma, robotik eğitiminde güçlendirme öğrenme yaklaşımının etkisini incelemektedir. RL, robotların çevreleriyle etkileşime girerek yeni beceriler öğrenmelerini sağlar ve öğrencilerin problem çözme becerilerini geliştirmede önemli bir rol oynar.

**3.Metodoloji**

Bu projede, A\* ve Q-learning algoritmaları kullanılarak bir labirent oyunu geliştirilmiştir. Oyunda bir oyuncu (Q-learning algoritmasını kullanan) ve düşmanlar bulunmaktadır. Oyuncunun amacı, labirentin başlangıç noktasından bitiş noktasına ulaşmaktır.

##### A\* Algoritması

A\* algoritması, başlangıç ve bitiş noktaları arasındaki en kısa yolu bulmak için kullanılır. Heuristic fonksiyonu olarak Manhattan mesafesi kullanılmıştır. Bu mesafe, iki nokta arasındaki yatay ve dikey mesafelerin toplamını ifade eder ve özellikle doğrudan gidilemeyen durumlarda en kısa yolu tahmin etmek için kullanılır.

KOD:

def astar(maze, start, end):  
 neighbors = [(0, 1), (0, -1), (1, 0), (-1, 0)]  
 close\_set = set()  
 came\_from = {}  
 gscore = {start: 0}  
 fscore = {start: heuristic(start, end)}  
 oheap = []  
  
 heapq.heappush(oheap, (fscore[start], start))  
  
 while oheap:  
 current = heapq.heappop(oheap)[1]  
  
 if current == end:  
 data = []  
 while current in came\_from:  
 data.append(current)  
 current = came\_from[current]  
 data.append(start)  
 data.reverse()  
 return data  
  
 close\_set.add(current)  
 for i, j in neighbors:  
 neighbor = current[0] + i, current[1] + j  
 tentative\_g\_score = gscore[current] + heuristic(current, neighbor)  
 if 0 <= neighbor[0] < len(maze):  
 if 0 <= neighbor[1] < len(maze[0]):  
 if maze[neighbor[0]][neighbor[1]] == 'X':  
 continue  
 else:  
 continue  
 else:  
 continue  
  
 if neighbor in close\_set and tentative\_g\_score >= gscore.get(neighbor, 0):  
 continue  
  
 if tentative\_g\_score < gscore.get(neighbor, 0) or neighbor not in [i[1] for i in oheap]:  
 came\_from[neighbor] = current  
 gscore[neighbor] = tentative\_g\_score  
 fscore[neighbor] = tentative\_g\_score + heuristic(neighbor, end)  
 heapq.heappush(oheap, (fscore[neighbor], neighbor))  
  
 return False

##### Q-Learning Algoritması

Q-learning, oyuncunun ödül ve cezalar üzerinden en iyi stratejiyi öğrenmesini sağlar. Ajan, belirli durumlarda aldığı ödüllere göre en uygun aksiyonu seçer ve bu aksiyonlar üzerinden çevreyi keşfeder.

class QLearningPlayer(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, maze):  
 super().\_\_init\_\_()  
 self.image = pygame.Surface((20, 30))  
 self.image.fill(GREEN)  
 self.rect = self.image.get\_rect()  
 self.rect.topleft = (30, 30)  
 self.maze = maze  
 self.health = MAX\_HEALTH  
 self.reward = 0  
  
 self.q\_table = np.zeros((20, 20, 4)) # 20x20 grid, 4 actions (up, down, left, right)  
 self.alpha = 0.1  
 self.gamma = 0.9  
 self.epsilon = 0.1  
 self.actions = [(-1, 0), (1, 0), (0, -1), (0, 1)]  
 self.state = (self.rect.y // 30, self.rect.x // 30)  
 self.path = []

## 4. VeriTabanı ve Veriler

#### . Veri ve Yöntem

Veri, labirent haritaları ve oyuncu ile düşmanların hareket verilerini içermektedir. Bu veriler, algoritmaların performansını değerlendirmek için kullanılmıştır. Oyun süresi boyunca oyuncunun hareketleri ve aldığı ödüller kaydedilmiş ve bu verilere dayanarak algoritmaların etkinliği analiz edilmiştir.

##### Labirent Haritaları

. # Labirent haritaları  
MAZE\_1 = [  
 "XXXXXXXXXXXXXXXXXXXX",  
 "XS X",  
 "X XXXXXXXXXXXXXX X",  
 "X X X X",  
 "X X XXXXX X X X",  
 "X X X X X X",  
 "X X X XXXXXXX X",  
 "X X X X X",  
 "X X X X",  
 "XXXXXXX XXXXXXXXXX",  
 "X X X",  
 "X XXXXX XXXX X",  
 "X X X X X",  
 "X X XXXXX X X X",  
 "X X X X X X",  
 "X X X XXXXXXX X",  
 "X X X EX",  
 "X X XXXXXXXXXXXXX",  
 "X X",  
 "XXXXXXXXXXXXXXXXXXXX"  
]  
  
MAZE\_2 = [  
 "XXXXXXXXXXXXXXXXXXXX",  
 "XS ",  
 "X XXXXXXXXXXXX X",  
 "X X X X",  
 "X X XXXXX X X X",  
 "X X X X X X",  
 "X X X XXXXX X",  
 "X X X X X",  
 "X X X X",  
 "XXXXX XXXXXXXX",  
 "X X",  
 "X XXXXX XXXX X",  
 "X X X X X",  
 "X X XXXXX X X X",  
 "X X X X X X",  
 "X X X XXXXXXX X",  
 "X X X X",  
 "X X XXXXXXXXXXXXX",  
 "X EX",  
 "XXXXXXXXXXXXXXXXXXXX"  
]

Bu çalışmada, Q-learning algoritmasının performansını değerlendirmek için kullanılan veri, oyuncunun her bir adımda aldığı ödülleri ve cezaları içermektedir. Ayrıca, oyun süresi boyunca ajan tarafından yapılan hareketler ve öğrenme süreci de analiz edilmiştir. Veriler, ajan'ın oyun ortamıyla etkileşiminden elde edilmiştir ve bu etkileşimler, Q-learning algoritmasının güncellenmesi için kullanılmaktadır.

Oyunun her bir adımında, ajan'ın mevcut durumu, yaptığı hareket, aldığı ödül ve ulaştığı yeni durum kaydedilir. Bu veriler, Q-learning algoritmasının performansını izlemek ve optimize etmek için kullanılır. Veritabanı, aşağıdaki verileri içerir:

* **Durum (State):** Oyuncunun labirentteki mevcut konumu.
* **Aksiyon (Action):** Oyuncunun belirli bir durumdan yaptığı hareket (yukarı, aşağı, sola, sağa).
* **Ödül (Reward):** Oyuncunun yaptığı hareket sonucu aldığı ödül (örneğin, hedefine ulaşırsa yüksek ödül, duvara çarparsa düşük ödül).
* **Yeni Durum (New State):** Oyuncunun yaptığı hareket sonucu ulaştığı yeni durum.

Veritabanı bu bilgileri kullanarak Q-learning algoritmasının performansını optimize etmek ve ajan'ın öğrenme sürecini hızlandırmak için kullanılır. Oyuncunun her adımda aldığı ödül ve cezalar, Q-table'ın güncellenmesi için kullanılır ve bu süreç, oyuncunun zaman içinde en iyi hareketleri öğrenmesine yardımcı olur.

## 5. Sonuçlar ve Tartışma

Bu projede, A\* algoritması ve Q-learning algoritması kullanılarak bir labirent oyununda oyuncunun en kısa yolu bulması ve öğrenmesi sağlanmıştır. A\* algoritması, belirli bir başlangıç ve bitiş noktası arasındaki en kısa yolu bulmak için etkili bir yöntemdir ve bu çalışmada başarılı bir şekilde uygulanmıştır.

**A\* Algoritması Sonuçları**

A\* algoritması, labirentteki en kısa yolu başarıyla bulmuştur. Kullanılan Manhattan mesafesi, iki nokta arasındaki yatay ve dikey mesafelerin toplamını hesaplayarak yol bulma işlemini hızlandırmıştır. Bu yöntem, özellikle engellerle dolu labirentlerde etkili sonuçlar vermiştir. A\* algoritmasının avantajları şunlardır:

* **Hızlı ve Verimli:** Algoritma, hedefe en kısa sürede ulaşmak için optimal yolu bulur.
* **Deterministik:** Belirli bir başlangıç ve bitiş noktası verildiğinde, her zaman aynı sonucu üretir.
* **Uygulanabilirlik:** Statik ve belirli çevrelerde mükemmel çalışır, bu nedenle labirent oyunlarında yaygın olarak kullanılır.

**Q-Learning Sonuçları**

Q-learning algoritması, ajan'ın (oyuncunun) dinamik ve değişen bir ortamda en iyi hareketleri öğrenmesini sağlamıştır. Oyuncu, çevresindeki durumları ve aldığı ödülleri kullanarak en uygun stratejiyi geliştirmiştir. Q-learning'in avantajları ve sonuçları şunlardır:

* **Öğrenme Yeteneği:** Ajan, ödüller ve cezalar üzerinden deneyim kazanarak çevresine adapte olmuştur.
* **Dinamik Ortam:** Değişen labirent yapıları ve hareket eden düşmanlara karşı oyuncunun hareketlerini optimize etmiştir.
* **Deneyim Tabanlı:** Oyuncunun her hareketi, gelecekteki hareketlerini optimize etmek için kullanılacak deneyim olarak kaydedilmiştir.

**Genel Tartışma**

Bu çalışmanın en önemli katkılarından biri, A\* algoritması ve Q-learning algoritmasının bir arada kullanılarak hem planlama hem de öğrenme yeteneklerinin birleşimidir. A\* algoritması, statik ve belirli bir çevrede en kısa yolu bulmak için kullanılırken, Q-learning algoritması, dinamik ve değişen çevrelerde en iyi stratejiyi öğrenmek için kullanılmıştır.

Bu kombinasyon, oyuncunun hem sabit bir hedefe ulaşmasını sağlamakta hem de çevresindeki değişikliklere adapte olmasına yardımcı olmaktadır. Bu tür bir yaklaşım, oyun geliştirme, robotik, otonom araçlar ve diğer pek çok alanda uygulanabilir.

**Gelecekteki Çalışmalar**

Bu çalışmanın sonuçları, gelecekteki araştırmalar için çeşitli yönler önermektedir:

* **Algoritma İyileştirmeleri:** Q-learning algoritmasının daha gelişmiş versiyonları, örneğin Double Q-learning veya Dueling Q-networks gibi yöntemler kullanılabilir.
* **Çevresel Değişkenlikler:** Oyuncunun daha karmaşık ve dinamik ortamlarda performansını değerlendirmek için farklı labirent yapıları ve engeller kullanılabilir.
* **Çoklu Ajan Sistemleri:** Birden fazla oyuncunun işbirliği veya rekabet içinde olduğu senaryolar araştırılabilir.

Bu projede elde edilen bulgular, A\* algoritması ve Q-learning algoritmasının birlikte kullanılarak dinamik ve statik çevrelerde etkin yol bulma ve öğrenme yeteneklerinin nasıl geliştirilebileceğini göstermektedir. Bu yöntemler, çeşitli uygulama alanlarında etkili çözümler sunabilir ve gelecekteki araştırmalar için değerli bir temel oluşturabilir.