**Q-Learning ile Labirentte Yol Bulma**

*Furkan Kurnaz, Furkan Yiğit Yılmaz*

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[140202029@kocaeli.edu.tr](mailto:140202029@kocaeli.edu.tr) , [140202043@kocaeli.edu.tr](mailto:140202043@kocaeli.edu.tr)

**Özet**

*Projede Q-Learning algoritmasının iteratif olarak çalışması gerçeklenmiştir. Proje Swift dili ile kodlanmış ve MVC(Modal-View-Controller) tasarım kalıbı yöntemi ile mimarisi kurulmuştur. Uygulamanın kullanıcı arayüzünde girilecek olan başlangıç, bitiş ve iterasyon sayısının olduğu ekran bir iOS uygulaması olarak gerçeklenmişken, input değerlerinin alındığı input.txt dosyası MacOS işletim sisteminden okunarak projede işlenmiştir. Ayrıca outPath.txt outQ.txt ve outR.txt dosyalarına programın çalıştırılması sonucu çıkan sonucun yazdırılması da projenin isterlerinde olduğu şekilde MacOS işletim sisteminde gerçeklenmektedir. Proje animasyon ekranı çıktılarını iOS’te gerçeklerken outPut çıktılarını ise MacOS’te gerçeklemektedir.*

# Giriş

Q-learning pekiştirmeli bir öğrenme algoritmasıdır. Ortam hakkında hiçbir şeyin bilinmediği durumlarda, Q-learning algoritması ortamı brute-force seklinde, her ortam için olası tüm aksiyonları takip ederek, problem çözümü icin en karlı yolu bulmaya çalışır. Q-learning algoritmasının girdileri kazanç matrisi olarak adlandırılan R matrisidir. Bu matrisin satır ve sütunlar ortamları temsil etmekte, R[i][j] değeri ise i durumundan j durumuna geçildiğinde elde edilen anlık kazanç değeridir. Eğer i durumunda j durumuna bir geçiş yoksa R[i][j] değeri -1, geçiş var ancak j durumu hedef durum değilse değeri 0, j hedef durum ise değeri kullanıcı tarafından belirlenen bir kazanç değeridir. [1]

Yapay zeka çalışmaları yapan Google DeepMind’ın son uygulamalarından birisi olan “derin öğrenmeyi güçlendirme” başlıklı çalışmasında Q-Learning algoritması uygulanmaktadır. Bu uygulama ile bazı Atari 2600 oyunları ileri seviyelerde başarılı bir şekilde oynanmaktadır.

Q-learning ile labirentte yol bulma başarılı bir şekilde gerçeklenebilmektedir. Yolun başarılı bir şekilde bulunabilmesi için algoritmanın gerçeklenmesinde iterasyon sayısı ve labirentin odacık sayısının etkisi vardır. Az iterasyon sayısı hesaplama zamanındaki maliyeti düşürürken, sonuca ulaşmada kesin sonuçların verilemediği zamanlara yol açabilmektedir. Ayrıca labirentin odacık sayısı arttıkça, Q-Learning öğrenmesi daha da zorlaşmaktadır. Örnek olarak dört odacıklı bir matriste yüz iterasyon ile doğru sonuçları elde edebilirken, yirmi beş odacıklı bir matriste altı bin iterasyondayken doğru sonucun elde edilememesi gibi durumlar meydana gelebilmektedir. Bu açıdan algoritmanın doğru gerçeklenebilmesi için ortam hakkında bilgi ve doğru iterasyon sayısı büyük önem arz etmektedir.

# Temel Bilgiler

Proje Apple tarafından 2014 yılında yayınlanan, iOS, macOS, watchOS ve tvOS işletim sistemlerine yönelik uygulamalar geliştirmede kullanılan SWIFT dilinin V3.0.1 versiyonu ile kodlanmıştır. Kodlama XCode 8.1 bütünleşik geliştirme ortamında geliştirilmiştir. Ayrıca çıktıların ve kullanıcı girişlerinin kontrol edilmesi için XCode ile çalışan simulasyon uygulaması da kullanılmıştır.

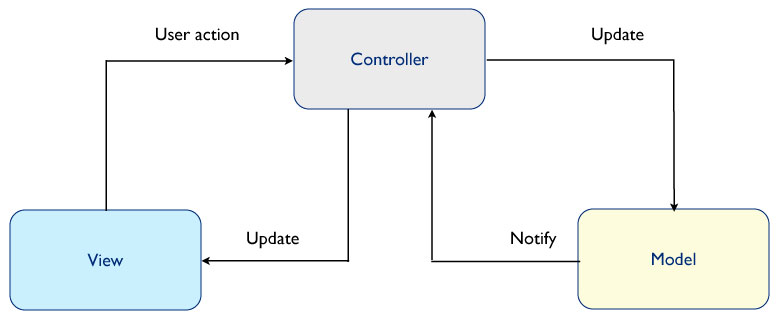
# Diğer Bölümler

## 

## Proje geliştirilmesinde MVC kısaltmasıyla adlandırılan Modal - View - Controller yazılım geliştirme mimarisi kullanılmıştır.

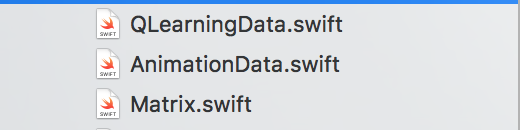
**3.1 Modal - View - Controller (MVC)**

Model-View-Controller (MVC), yazılım mühendisliğinde kullanılan bir "mimari desen"dir. Kullanıcıya yüklü miktarda verinin sunulduğu karmaşık uygulamalarda veri ve gösterimin soyutlanması esasına dayanır. Böylece veriler ([İng.](https://tr.wikipedia.org/wiki/%C4%B0ng.) *model*) ve kullanıcı arayüzü ([İng.](https://tr.wikipedia.org/wiki/%C4%B0ng.) *view*), birbirini etkilemeden *controller* adı verilen ara bileşenle veri gösterimi ve kullanıcı etkileşiminden veri erişimi ve iş mantığını çıkarma suretiyle çözümlenmektedir. [2]



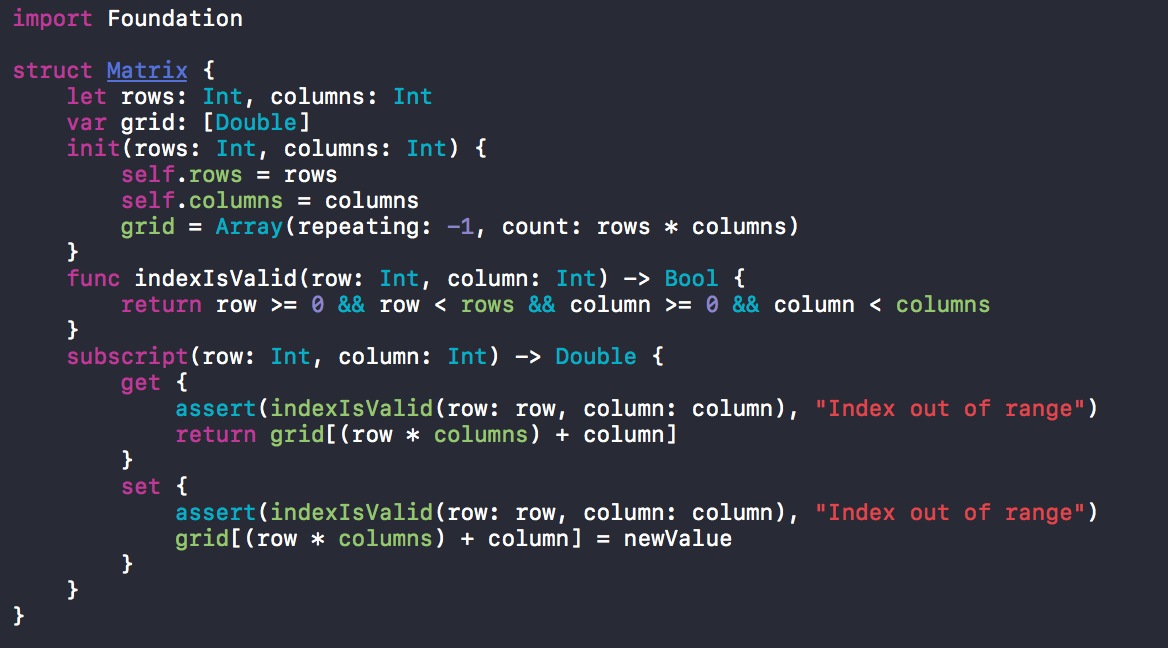
**3.2 Modal Sınıfları**

Modal sınıfları MVC yapısı içerisinde verileri ve veri yapılarının ihtiva edildiği sınıflardır. Bu proje kapsamında üç adet Modal sınıfı kullanılmıştır:



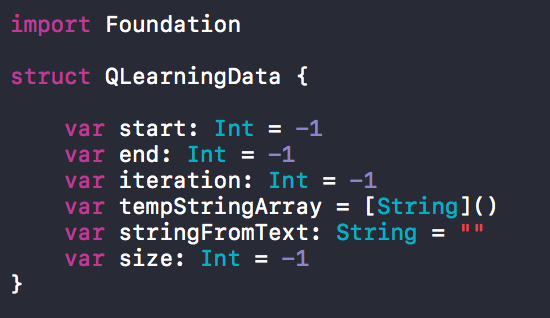
**3.2.1 Matrix.swift**

Swift dilinde varsayılan olarak matris veri yapısının gerçeklenebileceği bir yapı yoktur. Bunun yerine dizi yapısından kalıtım ile matris şeklinde diziler üretilmektedir. Matrix.swift isimli sınıfta bu veri yapısı barındırılmaktadır.



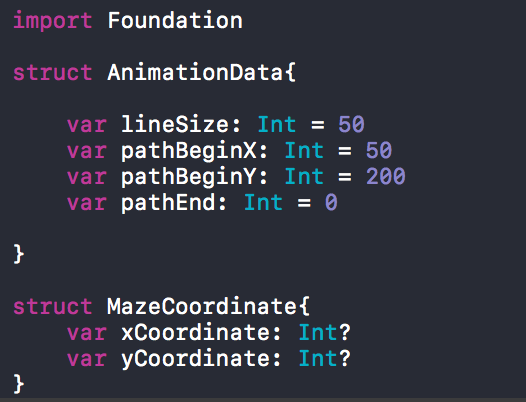
**3.2.2 QLearningData.swift**

QLearningData.swift sınıfında kullanıcının uygulama içerisinde karşılaşacağı ilk ekrandaki girdileri tutan ve bununla ilgili controller sınıfında işlemlerin yapılacağı veriler bulunmaktadır. Ayrıca input.txt dosyasından okunacak olan veriler de bu sınıfın içerisindeki veri yapılarında tutulmaktadır.



**3.2.3 AnimationData.swift**

Bu sınıf içerisinde labirentin boyutunu ve şeklini tutan ayrıca Q-Learning algoritması sonucu ortaya çıkan yolun çizilmesi için gerekli olan veri yapıları barındırılmaktadır.



**3.3 Controller Sınıfları**

**3.3.1 ViewController.swift**

Bu sınıf içerisinde kullanıcının giridileri ve input.txt’den okunan verilere göre algoritmanın gerçeklenmesi yapılmaktadır. İlk olarak input.txt’den mevcut labirentin komşulukları okunur ve R-matrisi buna göre oluşturulur. Ardından ortaya çıkan R-matrisine ve kullanıcının girmiş olduğu başlangıç, bitiş, iterasyon verilerine göre Q-matrisi oluşturulur. Q-matrisinin oluşturulmasında

**Q(durum,aksiyon)=R(durum,aksiyon)+ Max{Q(sonrakidurumlar,tümaksiyonlar)}**

formülü kullanılır. Ayrıca Q-Matrisinden yararlanarak ortaya çıkan yol da yeni bir diziye atanır. Daha sonra ortaya çıkan R-matris, Q-matris ve yol sırasıyla outR.txt, outQ.txt, outPath.txt isimli .txt uzantılı dosyalara yazdırılır.

MVC mimarisinde iki controller sınıfı birbirleriyle haberleşebilen yapılardır. Tüm bu hesaplamalar bittikten sonra MazeViewController.swift sınıfında labirenti oluşturmak için ViewController.swift sınıfından iterasyon, başlangıç odacığı, bitiş odacığı gibi bilgiler gönderilir. Gönderme işleminde UIKit kütüphanesinde tanımlı olan prepare(for segue:) ve shouldPerformSegue(withIdentifier identifier:) isimli fonksiyonlar kullanılır.

Ayrıca bu sınıf içerisinde kullanıcı girişlerinde hata ayıklaması da yapılır. Kullanıcı başlangıç odacığı, bitiş odacığı ve iterasyon sayısından herhangi birini eksik girmesi halinde ekranda bir uyarı gösterilmektedir ve kullanıcıdan bu bilgilerin hepsini eksiksiz bir şekilde girmesi istenilmektedir. Bu uygulamayla Q-learning algoritmasının gerçeklenmesi için mutlaka ihtiyaç duyduğu verilerin eksiksiz bir şekilde sağlanması hedeflenir.

**3.3.2 MazeViewController.swift**

MazeViewController.swift sınıfı içerisinde ViewController.Swift sınıfında gerçeklenen Q-Learning algoritması sonucunda ortaya çıkan yolun ve yine ViewController.swift içerisinde bulunan labirentin animasyonunun gerçeklemesi yapılmıştır.

MazeViewController.swift sınıfında labirentin boyutuna göre çizgi uzunluğunun belirlenmesi böylece kullanıcıya labirentin boyutundan bağımsız olarak hep aynı büyüklükte ve ekranı ortalamış bir labirent görüntüsü gösterilmektir.

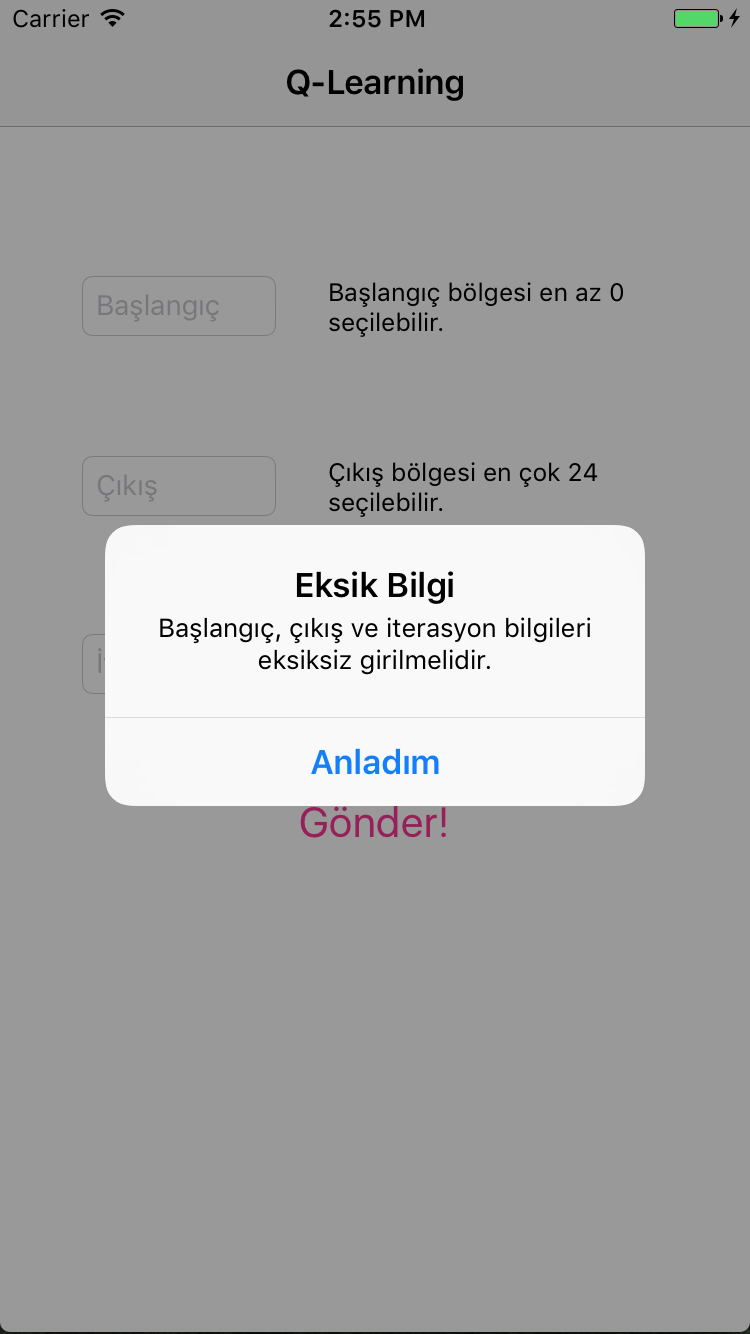
ViewController.Swift sınıfındaki hesaplamalar sonucu ortaya çıkan yolun animatik bir şekilde gösterimini sağlamak için UIKit kütüphanesinde ihtiva edilen UIView.animation(withDuration:) isimli fonksiyon kullanılmıştır.

**3.4 View**

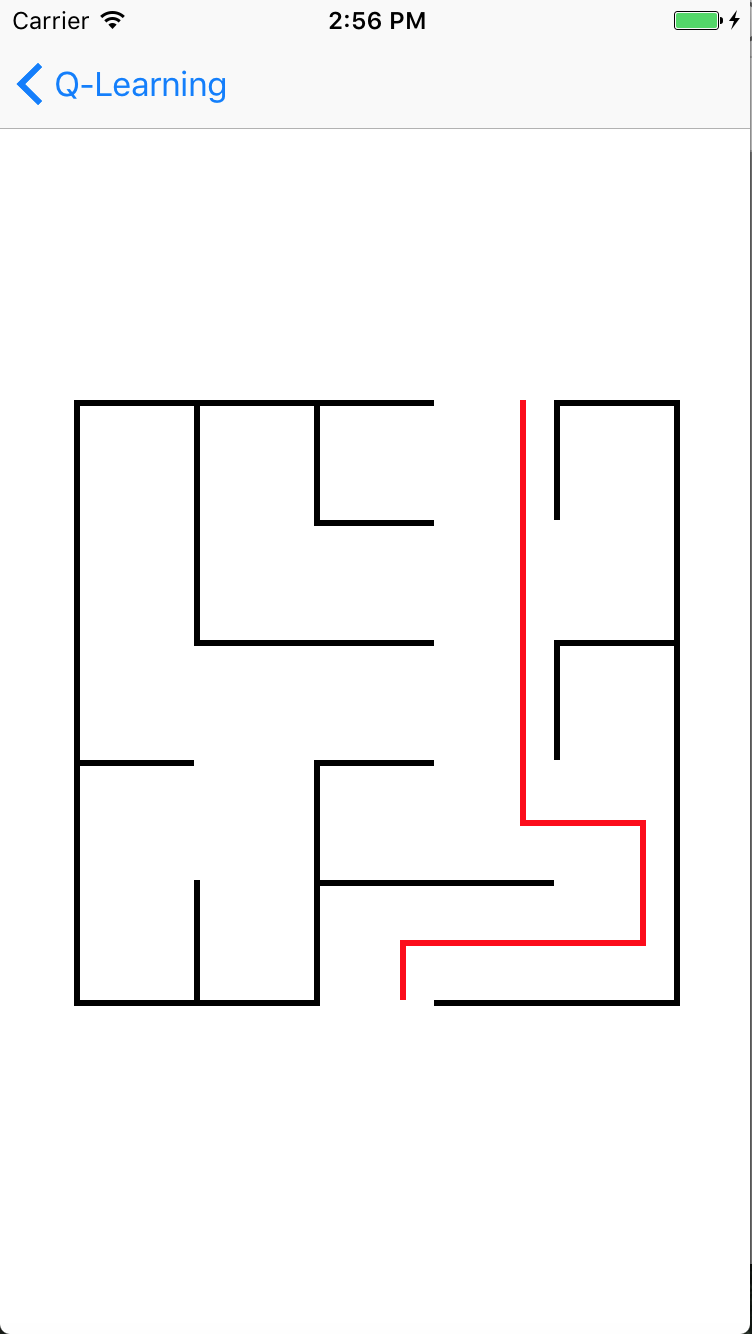
MVC mimarisi içerisinde View yapısı son kullanıcının etkileşimde bulunduğu her şeyi kapsamaktadır. Q-Learning ile labirentte yol bulma uygulamasında kullanıcıya gösterilen iki farklı View ekranı vardır. İlk ekranda kullanıcı başlangıç odacığını, bitiş odacığını ve iterasyon sayısının girilebileceği alanlar vardır:



Ayrıca kullanıcı bu üç alanın herhangi birisinde boşluk bırakması durumunda hata mesajı gösterilecektir:



İkinci View ekranında ise kullanıcının MazeViewController.swift sınıfında ihtiva edilen fonksiyonlar ile oluşturmuş olduğu labirent ve yol animasyonları gösterilmektedir:



# Sonuçlar

Proje isterlerde verilen maddelerin tamamını karşılamakta ve Q-Learning uygulamasını başarılı bir şekilde gerçeklemektedir. Proje verilmiş inputların hepsinde doğru iterasyon değerleri girildiğinde sorunsuz bir şekilde çalışmış ve projenin hedeflemiş olduğu çıktıların hepsini yerine getirmiştir.

# Kaynakça

1. YazLab proje açıklaması
2. Modal-View-Controller,Vikipedi https://tr.wikipedia.org/wiki/Model-View-Controller

[3] Q-Learning, Wikipeadia

<https://en.wikipedia.org/wiki/Q-learning>

[4] Modal-View-Controller in iOS, Apple

https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html