Proje 3

Q-learning ile Yol Planlaması

Altan TUĞFAN
170202123
Mühendislik Fakültesi
Bilgisayar Mühendisliği

Arda Talu
170202037
Mühendislik Fakültesi
Bilgisayar Mühendisliği

***Özet*—Bu uygulamamızda bizden istenen pekiştirmeli öğrenme (reinforcement learning) kullanarak ajanların bir görevi en yüksek kazanç, yani en düşük maliyetle tamamlamak için hangi eylemleri gerçekleştirmeleri gerektiğini bulan, 25*25'lik bir gridde rastgele belirlenmiş engelleri ajanın hesap ederek onlara çarpmadan başlangıç node'undan bitiş node'una gitmesi beklenmesi. Bunu yaparken de maliyeti en aza indirebilmesi ve en yüksek kazanç ile yolculuğunu tamamlayabilmesidir.**

Anahtar kelimeler—q-learning, pekiştirmeli öğrenme, görsel uygulamalar, C#, Unity, A*algorithm.

I. GİRİŞ

Bu Projeyi, bilgisayarlar, konsollar ve mobil cihazlar için video oyunları ve simülasyonları geliştirmek için kullanılan ve Unity Technologies tarafından geliştirilen çapraz platform bir oyun motoru olan Unity, ve scriptlerine destek olarak da C# dili ile gerçekleştirdik. Projenin amacı; Q-learning algoritması kullanarak, pekiştirmeli öğrenme (reinforcement learning) ile ajanların bir görevi en yüksek kazanç, yani en düşük maliyet ile gideceği yolu tamamlamak için hangi eylemleri gerçektirmeleri gerektiğini bulan, 25*25'lik bir gridde rastgele tanımlanmış engelleri ajanın hesap ederek bu engellere çarpmadan, yine kullanıcının belirttiği başlangıç node'undan bitiş node'una gitmesi beklenmektedir. Bu işlemi yaptıktan sonra maliyet hesaplarını kullanıcıya gösteren, grafiğe çizdiren ve tüm engelleri bir txt dosyasına yazdıran bir algoritma çalışması beklenmektedir.

Yaptığımız uygulama çalıştığında öncelikle bir ajan ve 25*25'lik boş bir grid kullanıcıyı karşılamaktadır. Öncelikle ajanın hangi node'dan başlayacağını kullanıcı seçmektedir. Ardından engelleri belirttikten sonra gideceği nodu'un seçilmesi beklenmektedir. Bu işlemden hemen sonra ajan belirttiğimiz algoritmaya göre en az maliyetli yolu hesaplamakta ve harekete başlamayıp bitiş node'una varmaktadır. Bitiş node'una vardıktan sonra "space" tuşu ile gidilen yolu nasıl aradığını adım adım kullanıcıya gösterip en sonunda en kısa yolu yeşil renk ile kullanıcıya göstermektedir. Gösterirken de o adıma gelene kadarki maliyeti (gCost), o noktadan bitiş node'una kadarki hesaplanan maliyeti(hcost) ve başlangıçtan bitiş node'una kadar olan toplam maliyeti her node'da o node'a özel göstermektedir.

Programı kodlamadan önce projede bizden istenenleri, yani kullanmamız gereken q-learning algoritmasını anladık, bu algoritmada işimizi en iyi görecek yöntemi araştırdık ve A* algoritmasını kullanmaya karar verdik. Arayüz tasarımının nasıl olması gerektiğini, hangi kütüphaneleri kullanmamız gerektiğini ve nasıl kullanabileceğimize dair fikir birliğine vardık. Bu algoritmayı nasıl ve nerde kullanacağımızı kararlaştırıp scriptleri yazmaya başladık.

II. YÖNTEM

A. *Grid Oluşturumu*

Programda öncelikle bizden istenen 25*25 boyutunda bir grid oluşturulması ve bu board üzerinde nesnelerin hareket edebilmeleri için bir dizi algoritması oluşturumu idi.

Öncelikle bir Grid sınıfı oluşturduk. Bu sınıfımız gridimizde vektörel aksiyonları mümkün kılan özellikler ekledik. Genişlik, yükseklik verilerini, vektörün orjinini, hücre boyutunu tutuyor ve üzerinde işlem yaptırıyor. Aynı zamanda gridimizin dizaynını, rengini de burada gerçekleştirdik.

B. *Pathfinding.cs*

Program yol bulma algoritmasını Q-learning'in A* algoritması ile gerçekleştirdik. Burda en önemli adım da pathfinding classı ile gerçekleştirdik. Pathfinding classı öncelikle ajanımızın o an konumlanan node'dan bitiş node'una kadar yolu belirttiğimiz algoritmasıyla bulmasıdır.listeAcik komutuyla arama için ssıraya alınan düğümleri tutar. ListelemeKapat komutuyla da zaten aranmış nodelar tutulur. Sürekli aynı nodeları aramaması için buna ihtiyacımız vardı. Maliyet hesabını yukarı, aşağı, sağa ve sola hareketlerde "10" maliyet, çarpraz hareketlerde "14 maliyet olarak işlem yapmaktadır.
Sınıfımız en düşük maliyetten başlayarak tüm komşu nodelara bakar. Eğer en az maliyetli değilse komşuluktan siler ve bakılanlar listesine ekler. Son node'a kadar bu işlemi gerçekleştirir ve en kısa olduğunu belirlediği tüm nodeları listesine alır.

C. *PathNode.cs*

Bu sınıfımız path'imizin x ve y değerlerini, bool değer olarak yürünebilir yani engel olmayan alanları, fcost'u yani toplam maliyeti tutar ve hesaplar.

D. *PathfindingDebugStepVisual.cs*

Bu sınıfımız yol bulurken atılan tüm adımları, algoritmamızın denediği tüm adımları görsel olarak kullanıcıya gösteren bir classtır. Bu adımları visualNodeList listesinde tutar. Dizisini de visualNodeArray'de tutarak gerçekleştirilecek işlemleri Gridimizin currentNode'una göre işlem yapmasını sağlar Space tuşuna her bastığımızda başlangıç nodundan bitiş noduna kadar her denenen ihtimalin gcost, hcost ve fcostunu gösterir.

E. PathfindingVisual.cs

Objenin hareketini görsel olarak görmemizi sağlayan sınıfımızdır. Bu sınıfı kullanarak, oluşturduğumuz ajanın bulunan konumundan en kısa yol hesaplandıktan sonra bitiş node'una kadar katettiği yolu yürümesini sağlar. Mesh diye belirttiğimiz ağımızın vektörel uzantısında hareket eder.

F. MeshUtils.cs

Bu sınıfımızda hesaplanan en kısa yolun vektörel kaydı tutulur. Bir vektör çizimi yaparak görsel olarak kullanıcıya sunar. Bu adımı diğer classlar ile kullandığımızda ajanımızın hareket güzergahı da çizilmiş ve uygulanmış olur.

G. Testing.cs

Testing.cs kısmı programın grid değerlerini, program açıldıktan sonra gridin boyutunu, adım başına griddeki değişimleri, mouse click fonksiyonlarını içinde barındıran bir arayüz sınıfıdır.

III. SONUÇ

Sonuç olarak kazanımlar;

- "Unity" motorunda, C# dilinde yazılmıştır.
- Grid.cs, patfinding.cs, PathNode.cs, PathfindingDebugStepVisual.cs, PathfindingVisual.cs, MeshUtils.cs ve Testing.cs dosyaları tutulmuştur.
- Q-learning ile maliyet hesabı öğrenilmiştir
- Unity ile oyun motorunda nasıl çalışılır, görsel nesnelerin nasıl projemize dahil olabileceği ve kullanılabileceği öğrenilmiştir.
- A* algoritmasıyla yine nasıl maliyet hesabı yapılacağı öğrenilmiştir.

Kullanılan diller, sınıflar ve kütüphaneler

- Unity
- C#
- Grid.cs
- Pathfinding.cs
- PathNode.cs
- PathfindingDebugStepVisual.cs
- PathfindingVisual.cs
- MeshUtils.cs
- Testing.cs
- System.Collections
- System.Collections.Generic
- UnityEngine
- CodeMonkey.Utils
- TMPro
- System.IO

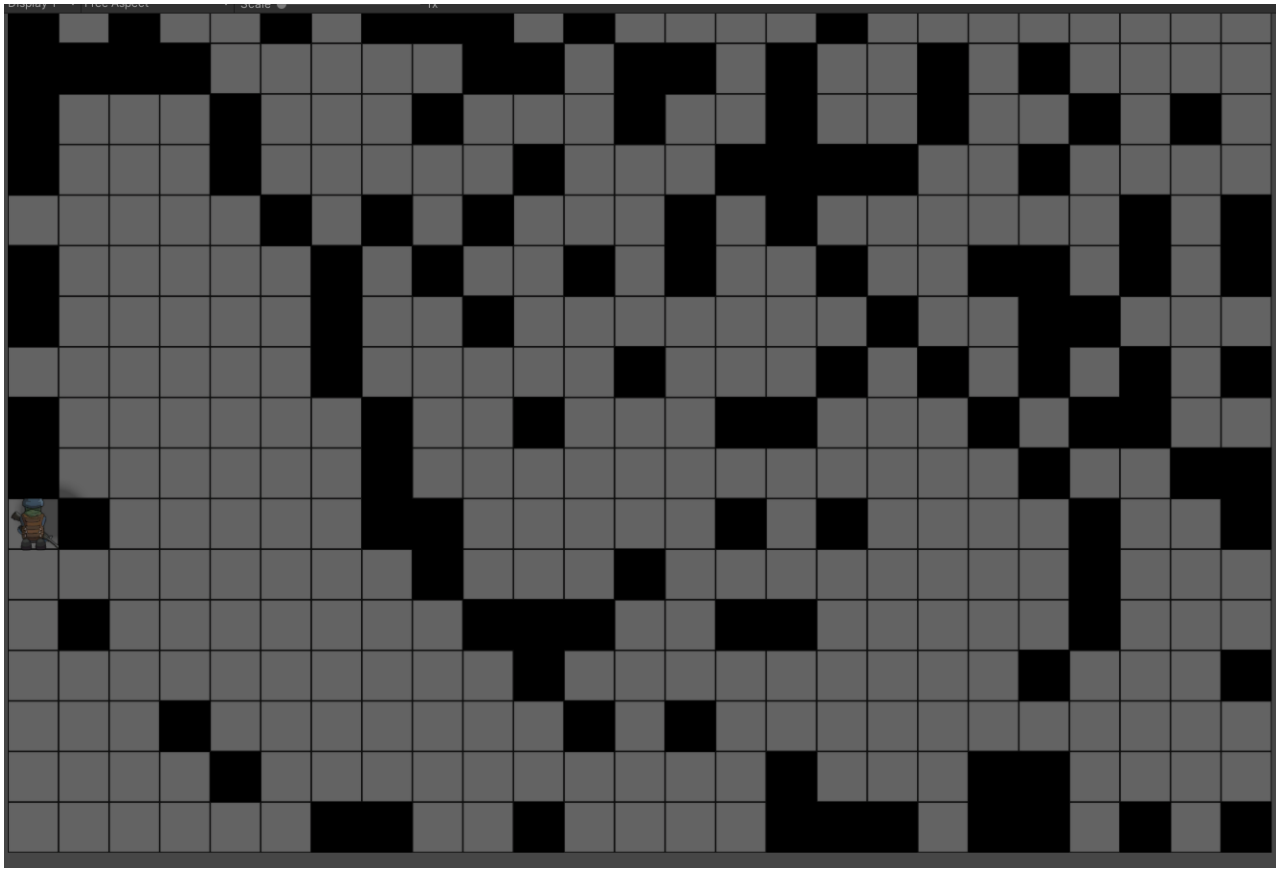
Karşılaştığımız problemler

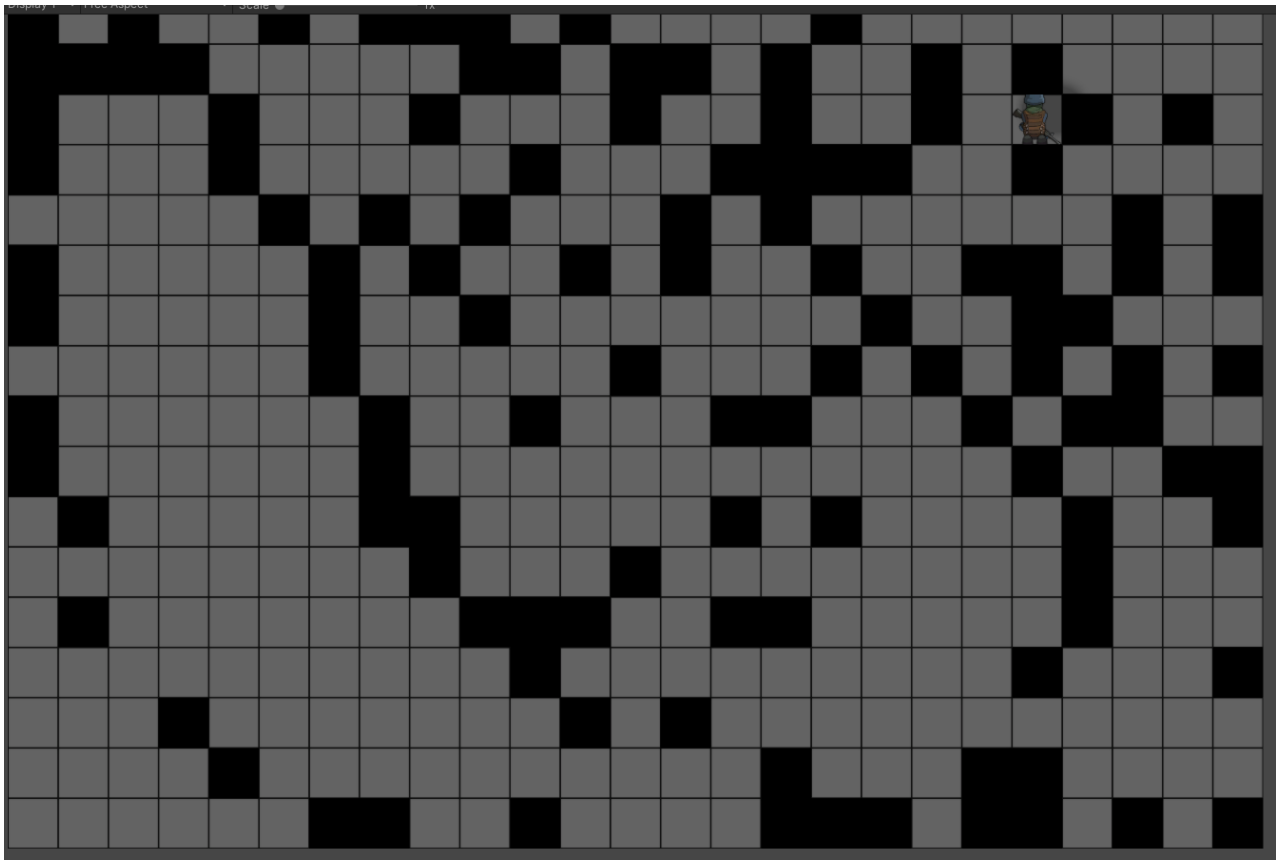
Programı yazarken Gridi oluştururduktan sonra kullanırken başta zorlandık. İstediğimiz şekilde karelere işlem yapamıyor, hangi path'de olduğunu kodumuza tanıtamıyorduk. Birkaç araştırma sonucunda çözdük

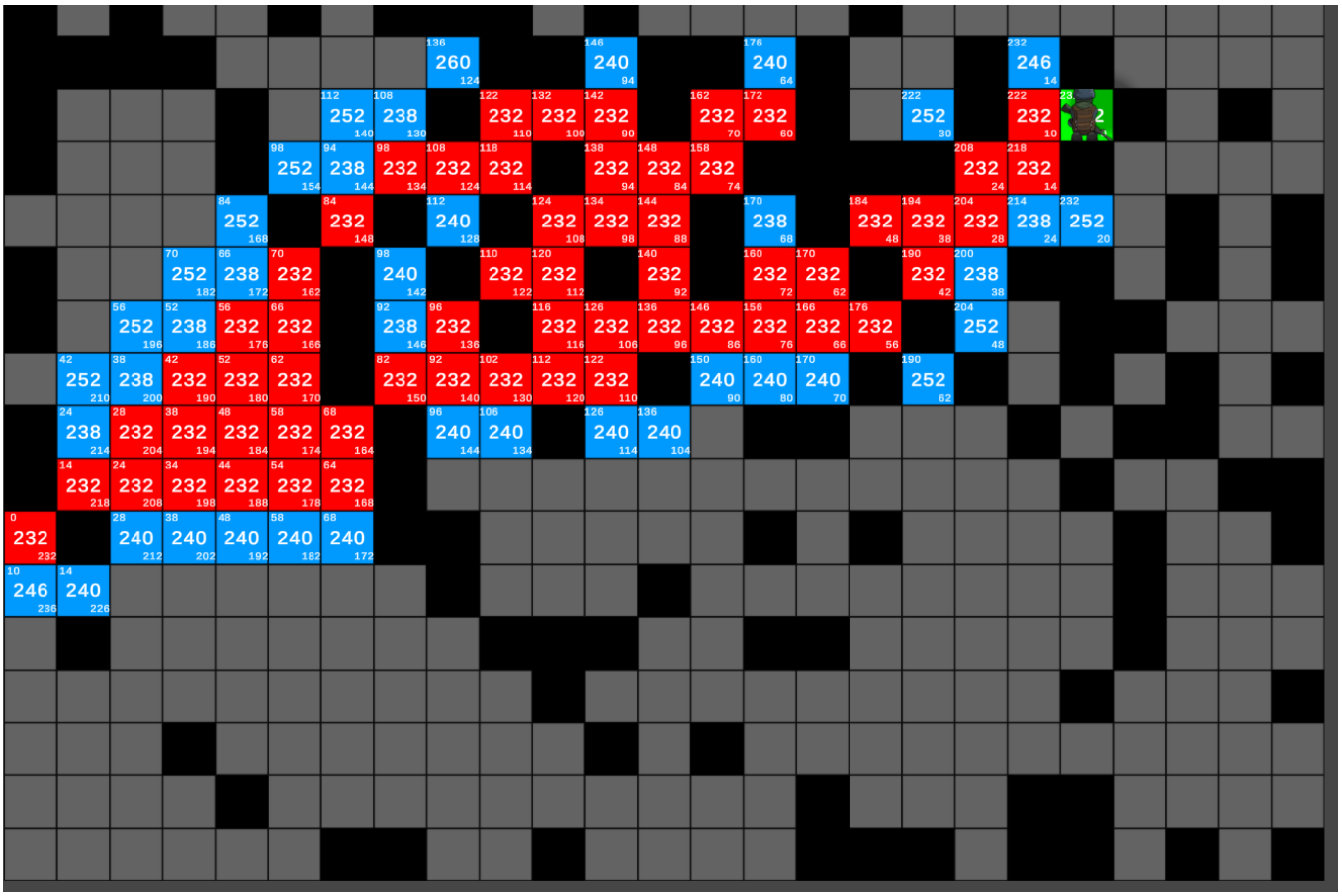
Maliyet grafiği ve bölüm adım sayısı grafiği oluştururken çok zorlandık. Unity'de grafik oluşturma toollarını bizim bulduğumuz sonuçlara uyarlayamadık ve bu kısmı yapmamaya, projemizdeki diğer adımlara daha çok özen göstermeye çalıştık.

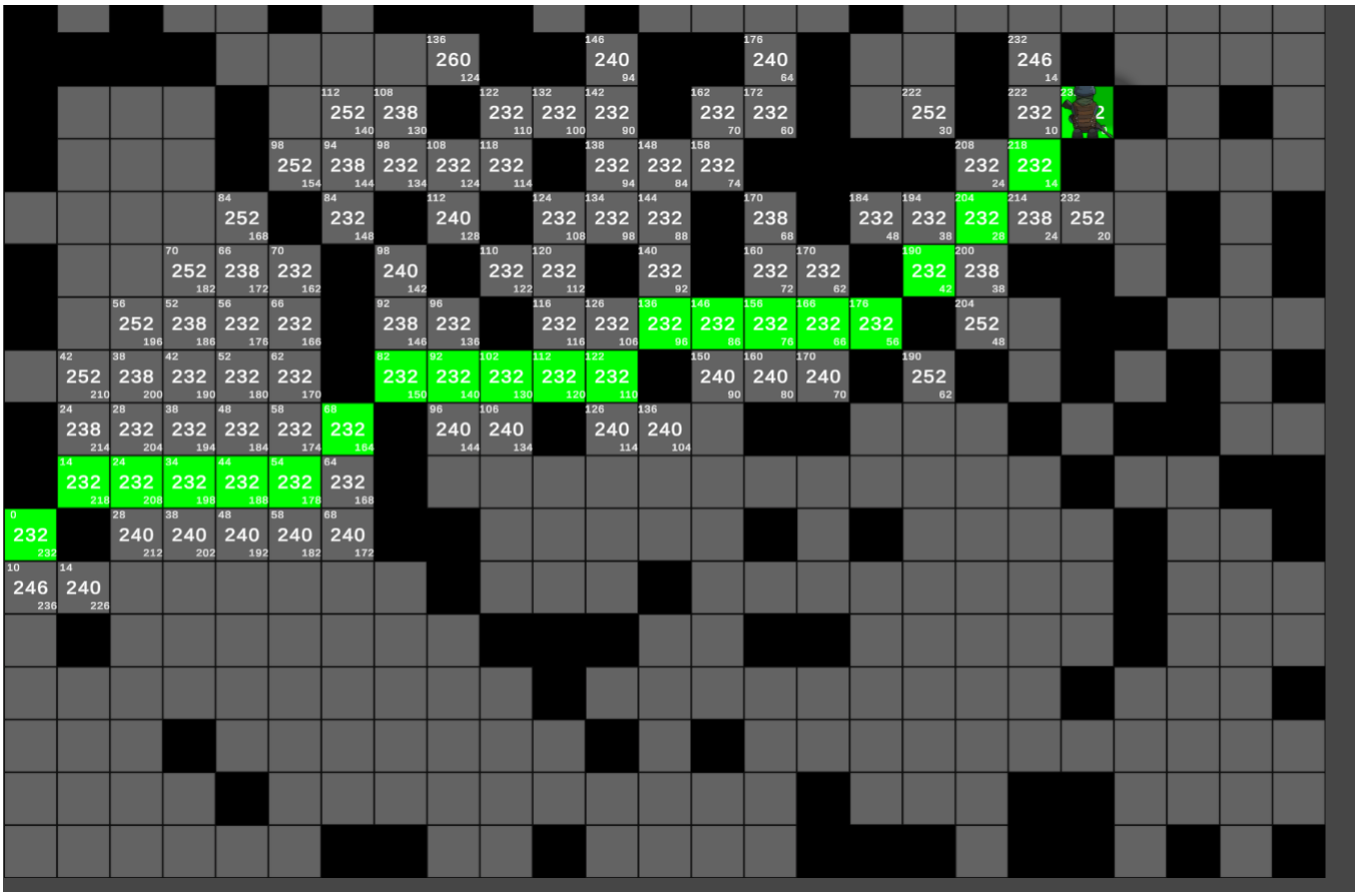
Ajanın görsel kısmınıyapmayı denedik. Sürekli hata aldık. Vektörel hareketinde zorlandık. Bu yüzden hazır bir görsel ve algoritma kullandık görsel kısmı için.

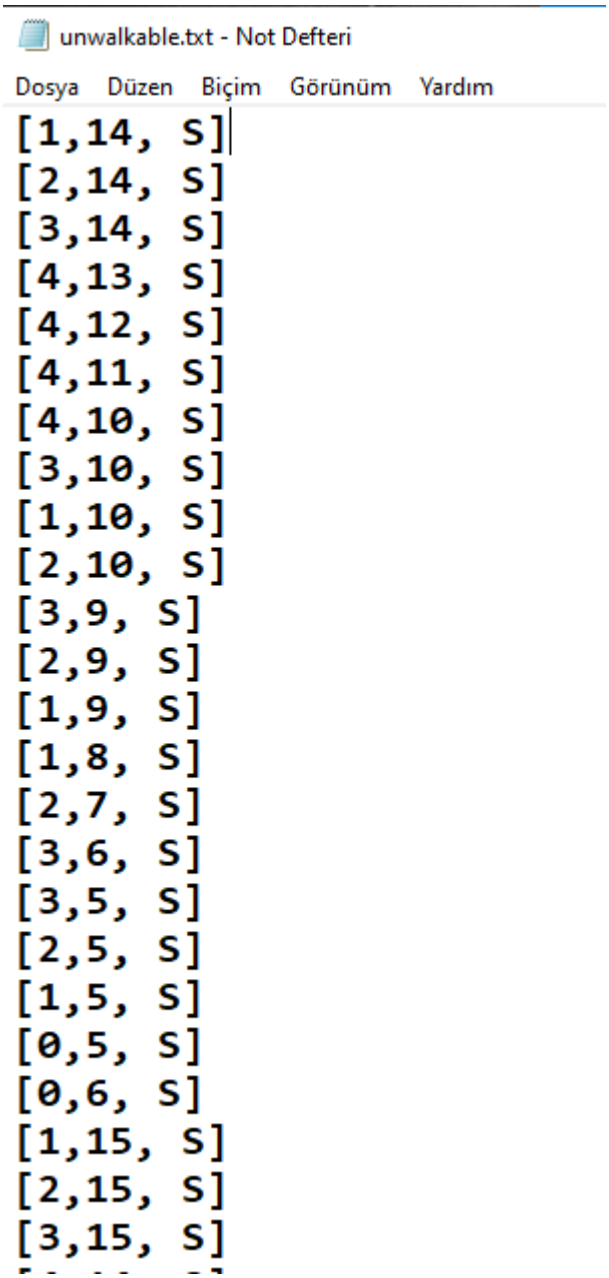
IV. ARAYÜZ GÖRÜNTÜLERİ



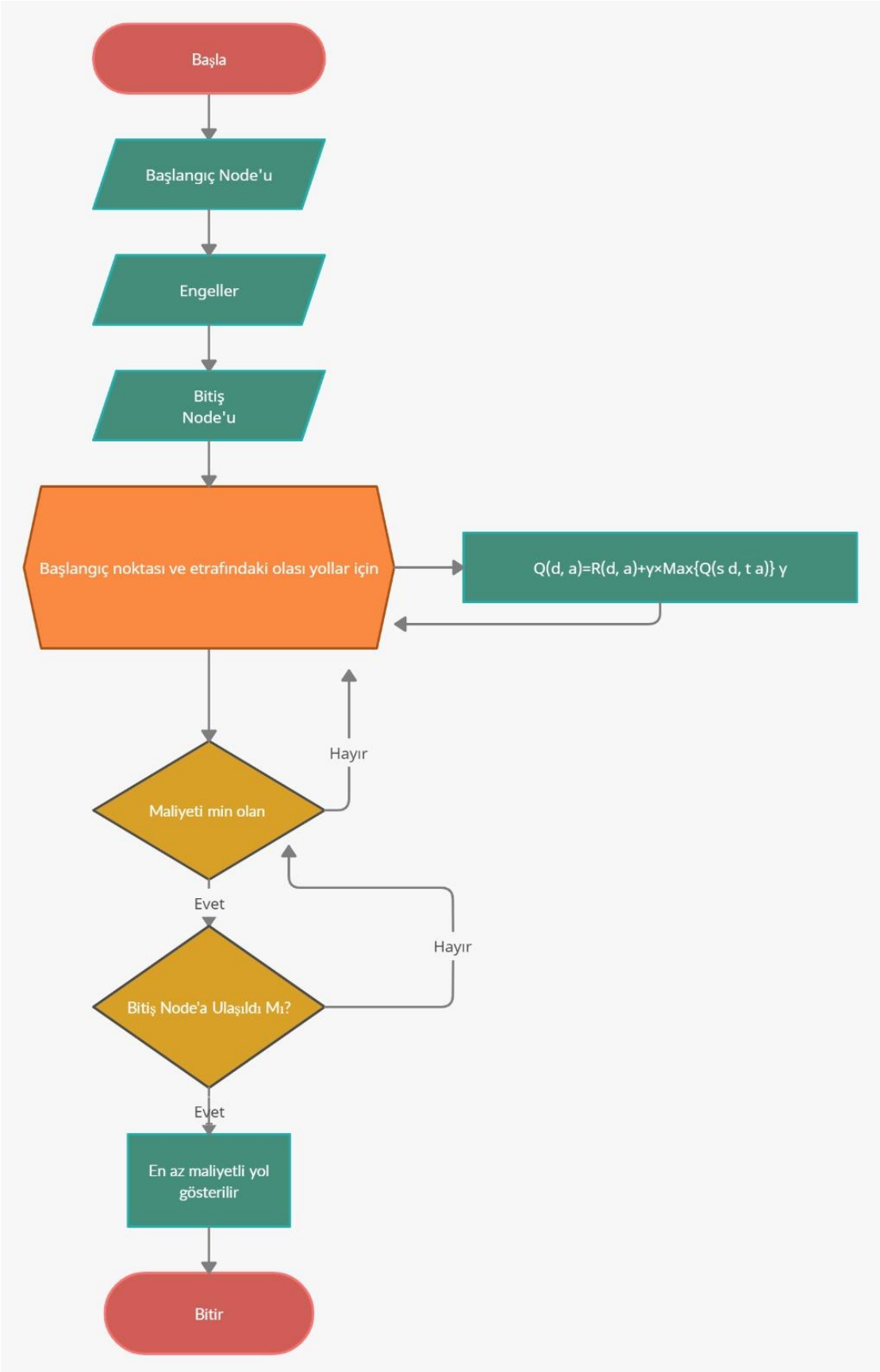







unwalkable.txt - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
[1,14, S]
[2,14, S]
[3,14, S]
[4,13, S]
[4,12, S]
[4,11, S]
[4,10, S]
[3,10, S]
[1,10, S]
[2,10, S]
[3,9, S]
[2,9, S]
[1,9, S]
[1,8, S]
[2,7, S]
[3,6, S]
[3,5, S]
[2,5, S]
[1,5, S]
[0,5, S]
[0,6, S]
[1,15, S]
[2,15, S]
[3,15, S]

V. AKIŞ DIYAGRAMI



VI. KAYNAKÇA

1. https://docs.unity3d.com/ScriptReference/Input.GetKeyDown.html
2. https://docs.microsoft.com/tr-tr/cpp/dotnet/how-to-specify-an-out-parameter?view=msvc-160
3. https://docs.microsoft.com/tr-tr/dotnet/api/system.console.writeline?view=net-5.0
4. https://blog.unity.com/technology/unity-ai-reinforcement-learning-with-q-learning
5. https://www.kodlamamerkezi.com/c-net/c-ile-dosya-okuma-ve-yazma-islemleri/
6. https://unitycodemonkey.com/video.php?v=alU04hvz6L4