



KÜTAHYA
SAĞLIK BİLİMLERİ
ÜNİVERSİTESİ

ML-AGENTS FINAL RAPORU

Yazar: Mert Koca
June 14, 2024



Özet

Bu proje, Unity'nin ML-Agents kütüphanesi kullanılarak Python ve PyTorch ile geliştirilen bir yapay zeka projesidir. Bu proje, Unity ortamında eğitilmiş bir yapay zeka modeli oluşturarak araba sürmeyi öğretmeyi amaçlamaktadır.

1 Giriş

Bu projenin amacı, Unity ortamında eğitilmiş bir yapay zeka modeli oluşturarak araba sürmeye öğretmektir. Bu rapor, projenin gereksinimlerini, literatür taramasını, teşvik öğrenme algoritmasını, kurulum aşamalarını, ML-Agents bileşenlerini, projenin farklı bölümlerini ve sonuçlarını detaylı bir şekilde ele almaktadır. Ayrıca, eğitim süreçleri ve elde edilen başarılar da raporda incelenmektedir. Bu çalışma, yapay zeka alanında ML-Agents gibi kütüphanelerin kullanımının ve uygulamalarının önemini vurgulamaktadır.

2 Gereksinimler

1. Python 3.9.13
2. Unity
3. PyTorch
4. Visual Studio
5. ML-Agents Kütüphanesi

3 Literatür

Python 3.9.13: Genel amaçlı, yüksek seviyeli, etkileşimli bir programlama dilidir. Basit ve okunabilir sözdizimine sahiptir. Python, modüler yapısı ve geniş standart kütüphanesiyle birçok alanda kullanılabilir.

PyTorch: Python tabanlı ve açık kaynaklı makine öğrenmesi kütüphanesidir.

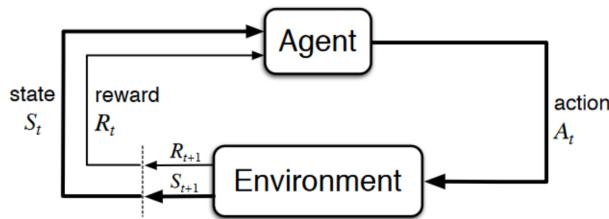
Unity: Oyun ve grafik uygulamaları geliştirmek için kullanılan bir oyun motorudur.

ML-Agents Kütüphanesi: Oyun ve simülasyonlarda ajanların eğitimi için ortamlar sağlayan açık kaynaklı bir projedir. Python API'si kullanılarak teşvik öğrenmesi yöntemiyle ajanlar eğitilebilir. PyTorch tabanlı uygulamalar sunar. Ajanlar, NPC davranışlarını kontrol etmek ve oyun yapılarının otomatik test edilmesi gibi çeşitli amaçlar için kullanılabilir.

4 Teşvik Öğrenme Algoritması

Teşvik öğrenme algoritmasında, ajan çevresiyle etkileşime girer, belirli eylemler gerçekleştirir ve bu eylemlerin sonuçlarına göre ödüller veya ceza alır. Resimde gösterilen şema, ajanın çevresiyle olan etkileşiminin ve bu süreçteki temel unsurları açıkça göstermektedir. Ajan, deneyimlediği ödül ve cezaları kullanarak en iyi eylem stratejilerini geliştirmeye çalışır.

Ibrahim Sobh Ibrahim'in yürüttüğü araştırma[1], Reinforcement Learning (RL) algoritmalarını kullanarak ajanların dinamik ortamlarda hızlı bir şekilde öğrenmesini ve adapte olmasını vurgulamaktadır. Derin RL'yi kullanarak, çalışma, özellikle müzakere gerektiren ve dinamik etkileşimler gerektiren senaryolarda otonom ajanların karar verme yeteneklerini artırmayı amaçlamaktadır.



Figür 1. Teşvik Öğrenme Algoritması Modeli [2]

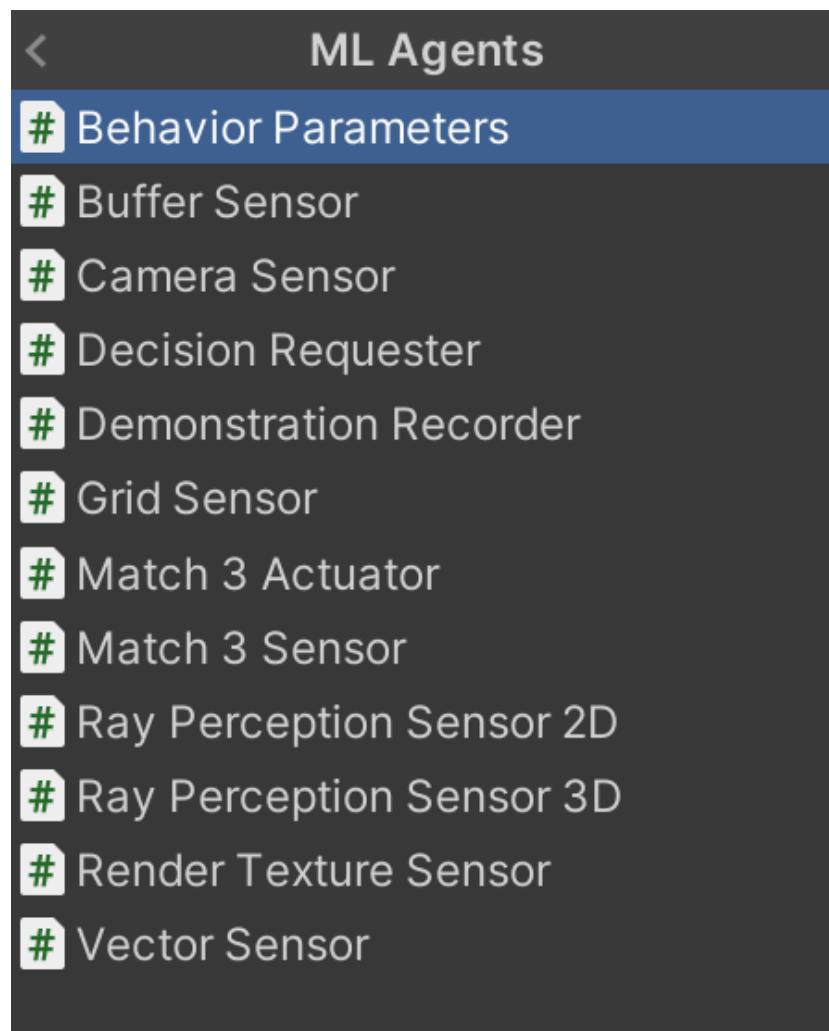
1. **Agent (Ajan):** Karar veren, öğrenen, eylemleri belirleyen ve ödüllü maksimize etmeye çalışan yapay zekadır.
2. **Environment (Çevre):** Ajanın kararlarının etkileşimde bulunduğu ve ona geri bildirim sağlayan simülasyon ortamıdır.
3. **State:** Zaman (t) anındaki çevrenin durumu ya da ajanın o andaki gözlemini temsil eder.
4. **Action:** Ajanın zaman (t) anında yapmayı seçtiği eylemdir.
5. **Reward:** Ajanın ($t+1$) anındaki eylemi sonucunda elde ettiği ödülüdür
6. **Next State:** Ajanın eylemi sonucunda çevrenin ulaştığı sonraki durumdur.

5 Kurulum Aşamaları

1. cmd yani komut istemi çalıştırılır ve cd (dizini değiştir) komutuyla projenin olduğu dosya açılır.
2. Dosyanın içinde venv adında bir virtual environment (sanal ortam) oluşturulur. Bu sayede python kullanılan başka bir proje ile karışıklık yaşanmasını önlenir.
3. Oluşturulan sanal ortam aktive edilir.
4. Python Package Installer (pip) güncel olup olmadığı kontrol edilir.
5. ML-Agents paketi kurulur.
6. Python torch kütüphanesi ve torch içindeki görüntü ve ses işleme kütüphanesi kurulur.
7. protobuf 3.20.3 sürümü kurlur. Protobuf, veri transfer protokolü olup diğer protokollere göre daha hızlıdır.
8. Packaging kütüphanesi kurulur. Bu kütüphane, python paketleri oluşturma, dağıtma ve sürümleme işlemlerini kolaylaştırır.
9. Unity projesi açılıp Package Manager kısmından mlagents paketi aktif edilir.
10. Boş bir nesne üretip bu nesnenin component kısmından ML-Agents aktif edilip edilmediği kontrol edilir.

6 ML-Agents Bileşenleri

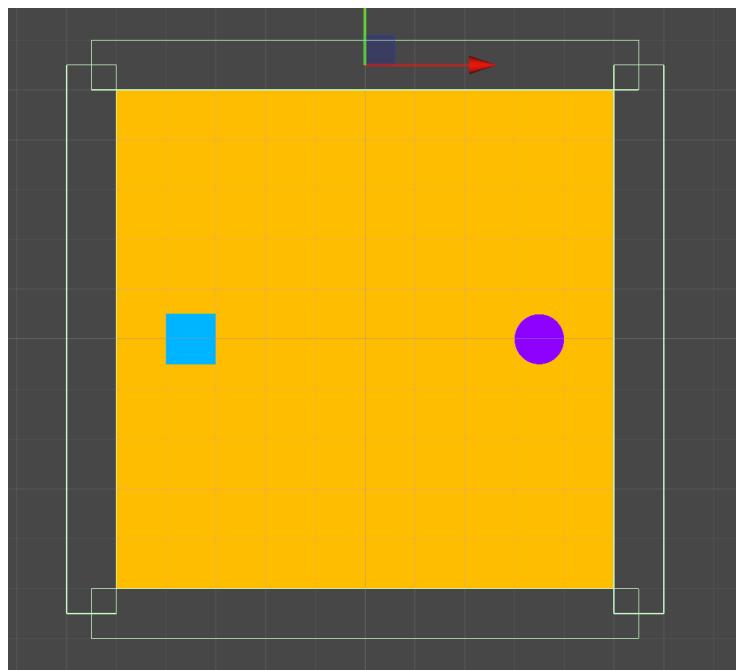
1. **Behavior Parameters:** Ajanların davranışlarını ve öğrenme süreçlerini kontrol etmek için kullanılan parametrelerin ayarlandığı bir bileşendir.
2. **Buffer Sensor:** Çeşitli veri türlerini depolamak ve işlemek için kullanılan bir sensördür.
3. **Camera Sensor:** Oyun dünyasını görsel veri olarak almak için kamera görüntülerini kullanan bir sensördür.
4. **Decision Requester:** Ajanların kararlarını gerektiğinde talep etmelerini sağlayan bir bileşendir.
5. **Demonstration Recorder:** Eğitim için insanların gerçekleştirdiği oyun hareketlerini kaydetmek ve kullanmak için bir kayıt cihazıdır.
6. **Grid Sensor:** Grid tabanlı oyunlarda çevre bilgisini algılamak için kullanılan bir sensördür.
7. **Match 3 Actuator:** Match 3 tarzı oyunlarda (Candy Crush) eylemleri gerçekleştirmek için kullanılan bir bileşendir.
8. **Match 3 Sensor:** Match 3 tarzı oyunlarda oyun durumunu algılamak için kullanılan bir sensördür.
9. **Ray Perception Sensor 2D/3D:** 2 boyutlu veya 3 boyutlu ortamlarda nesneleri algılamak için kullanılan bir sensördür.
10. **Render Texture Sensor:** Görüntüleri işlemek için render texture'ları kullanarak veri sağlayan bir sensördür.
11. **Vector Sensor:** Özel bir vektör verisiyle ajanlara çevre bilgisi sağlayan bir sensördür.



Figür 2. ML-Agents Bileşenleri

7 ML-Agents Hareket Projesi

Platform üzerinde mavi agentin hedeflediği mor renkli ödüller yer alır [3]. Bunların yanında görünmeyen ancak temas edildiğinde ceza uygulayan duvarlar da bulunmaktadır. Platformun rengi, Agent'ın etkileşimlerine göre değişmektedir. Ödül alındığında platform yeşile, duvara çarptığında kırmızıya dönüşür. Bu tasarımda, agentin belirlenen hedeflere ulaşmasına teşvik ederken, aynı zamanda negatif etkileşimlerden kaçınma yeteneğini geliştirmeyi amaçlamaktadır.



Figür 3. Hareket Projesi Bölüm Tasarımı

8 Proje Nesneleri

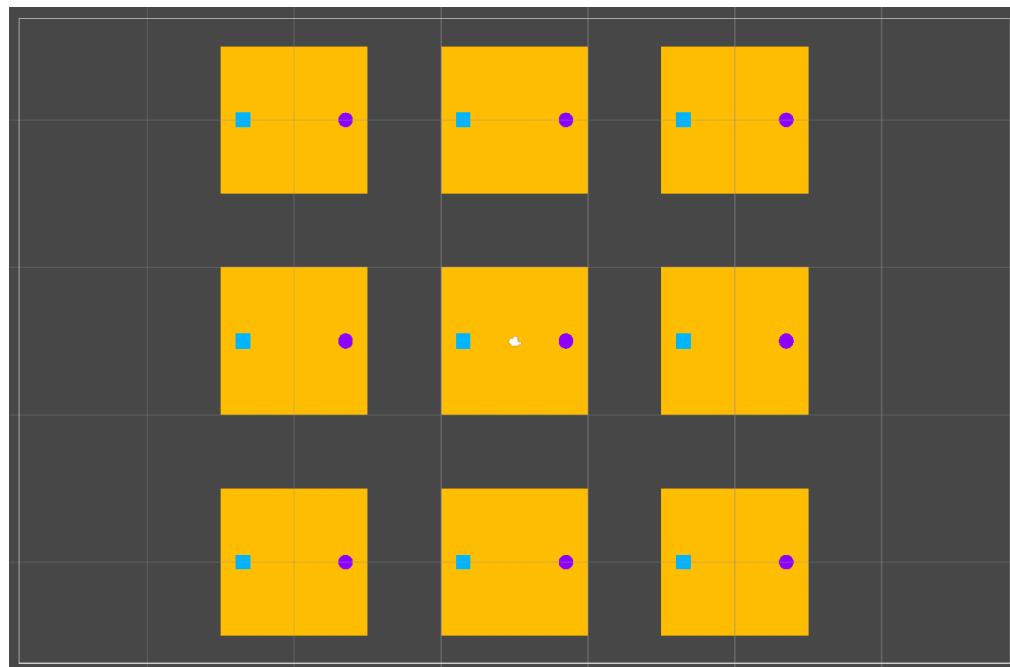
Projenin bu kısmında kullanmak üzere dört adet nesne seçilmiştir. Seçilen nesneler platform, agent, ödül ve duvarlardır. Agent hareket ederek ödüle ulaşmayı hedefler fakat bu süreçte duvarlara temas ederse en baştan başlamak zorundadır.



Figür 4. Agent ve Hedef

9 ML-Agents Davranışları

Agent hedefe ulaşmak için bir çok kez deneme yapmalıdır. Simülasyon hızlandırılmış olsada daha da hızlandırmak için oluşturulan bölüm koyanarak toplamda aynı bölümden dokuz tane oluşturulmuştur. Böylece dokuz ajan aynı anda eğitilir.

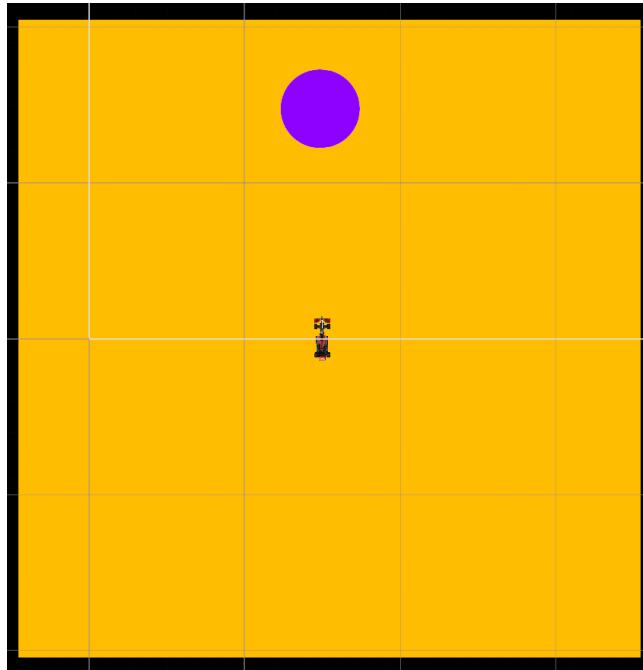


Figür 5. Çoklu Bölüm

10 Araç Kontrolleri Öğrenme Projesi

Hareket Projesine benzer olarak, platform üzerinde agentin hedeflediği mor renkli ödüller yer alır. Bu ödüllerin yeri her denemede rastgele olacak şekilde değişir. Araç duvara çarpara ceza puanı alır ve yeniden başlar, araç hedefe ulaşırsa ödül puanı alır ve yeniden başlar. Platformun rengi, Agent'ın etkileşimlerine göre değişmektedir. Ödül alındığında platform yeşile, duvara temas edildiğinde kırmızıya dönüşür. Bu tasarım, ajanların basit araç kontrollerini öğrenmesini sağlar.

flyyufelix adlı github kullanıcısının yaptığı "donkeyrl" projesinde[4] Donkey Car adı verilen bir arabanın bir pist etrafında RL kullanarak kendi kendine dönmesini sağlamıştır. Bu projede Double Deep Q Learning (DDQN) kullanılmıştır. DDQN, bir ajanın çevresiyle etkileşime girerek belirli bir durumda alabileceği eylemler arasında en uygun olanını seçmeyi öğrenen bir güçlendirme öğrenme algoritmasıdır.



Figür 6. Araç Kontrolleri Öğrenme Projesi Bölüm Tasarımı

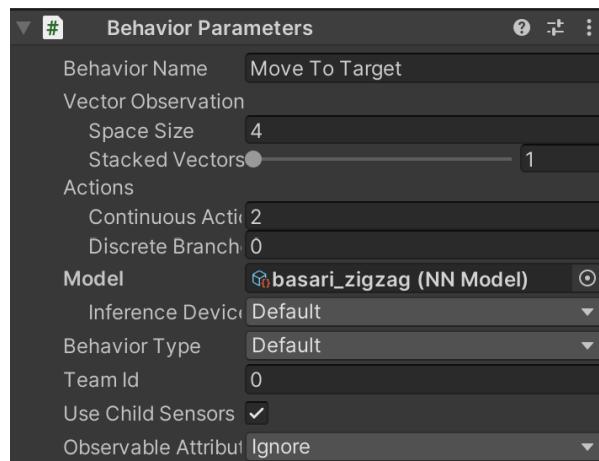
11 Araç ve Ajan Kontrolleri

Önceden carInput isminde başka bir koddan alınan girdiler, ajan kodunun içindeki X eksenine ve Y eksenine olarak ajan hareketlerinden alındı. Böylece ajanların araba kontrollerine erişimi sağlandı.

Ajanın kullancı tarafından kontrol edilip edilemediğini, ajanın kaç eksende hareket yaptığı, eğitilen beynin ajana aktarıldığı ve diğer bir çok ayarların yapıldığı kısım Behavior Parameters olarak adlandırılır. Bu özellik ML-Agents ile birlikte gelir.

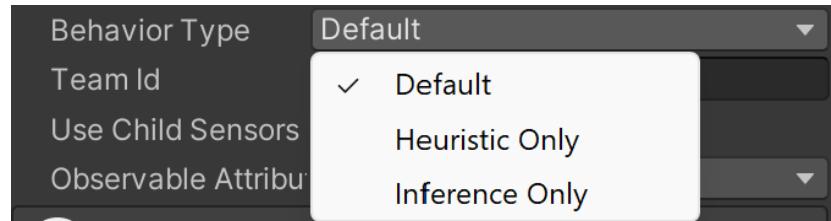
Car Settings	
Drift Mode	0.35
Acceleration Const	30
Turn Const	1.5
Max Speed	25
Kmh	152.9001
Acceleration Input	0.02137605
Turn Input	-0.3828245
Rotation Angle	163.6128
Velocity Vs Up	7.632071
Car Rigidbody 2D	Agent (Rigidbody 2D)

Figür 7. Araba Kontrolleri



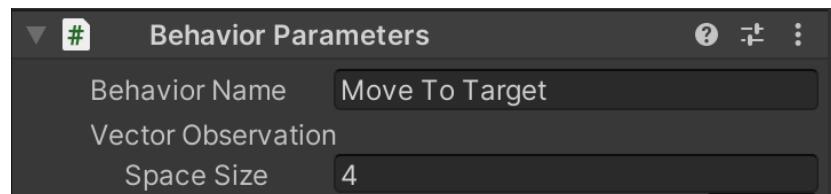
Figür 8. Behavior Parameters

Behavior Type: Agent'in kullancı tarafından kontrol edilip edilemediğini değiştiren bir özelliktir. Agent'i kullanicının yönetmesi isteniyorsa heuristic only seçeneği seçilmelidir. Agent'a dışarıdan herhangi bir müdahalede bulunmadan hedefe ulaşması isteniliyorsa inference only seçeneği seçilmelidir.



Figür 9. Behavior Type

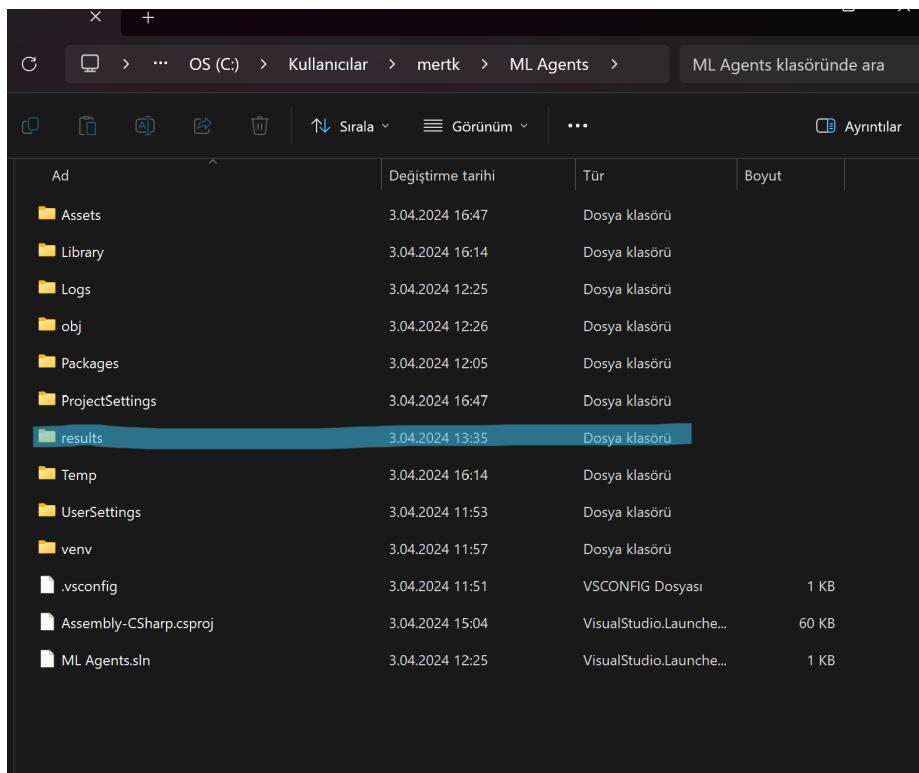
Space Size: Agent'in sahip olduğu davranışları belirleyen ve yönlendiren bir özelliktir. Bazı durumlarda Agent'in farklı davranışlar sergilemesi istenebilir. Bu durumda, her bir nesne için farklı davranışlar tanımlanması gereklidir. Bu projede Agent ve Ödül olarak iki hareketli nesne olduğundan her iki nesnenin de x ve y eksenindeki hareketleri almak için Space Size özelliği dört olarak ayarlanır.



Figür 10. Space Size

12 Beyin Oluşturma Kaydetme ve Yükleme

- Unity projesi çalıştırıldığından itibaren Agentlar öğrenmeye başlar. Öğrenmiş olan bu beyin, proje dosyasının içindedir

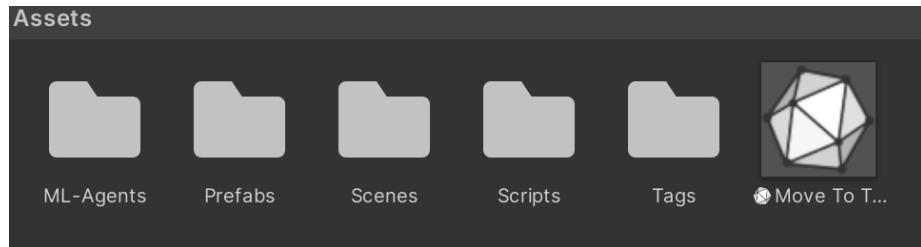


Figür 11. Proje Dosyaları



Figür 12. Beyin Dosyası

2. Result dosyasının içindeki beyin, proje dosyasının içindeki Assets klasörüne kopyalanır. Kopyalanan bu beyin Unity içerisinde Assets kısmında görülebilir.



Figür 13. Assets Klasörü

3. Agent altında, Behavior Parameters bileşeninin içindeki, Model adındaki değişkene Assets klasöründeki beyin atanır. Böylece önceden eğitilen bir Agent'ı projeye eklemiş oluruz.



Figür 14. Model Değişkeni

13 Başarılı ve Başarısız Oranı

Ekranın sol üst kısmında başarılı deneme sayısını, başarısız deneme sayısını, başarılı ve başarısız deneme sayısı arasındaki farkı ve başarı oranını gösteren bir panel yer almaktadır. Bu panel sayesinde deneme sayısı arttıkça başarı oranının arttığı gözlemlenebilir.

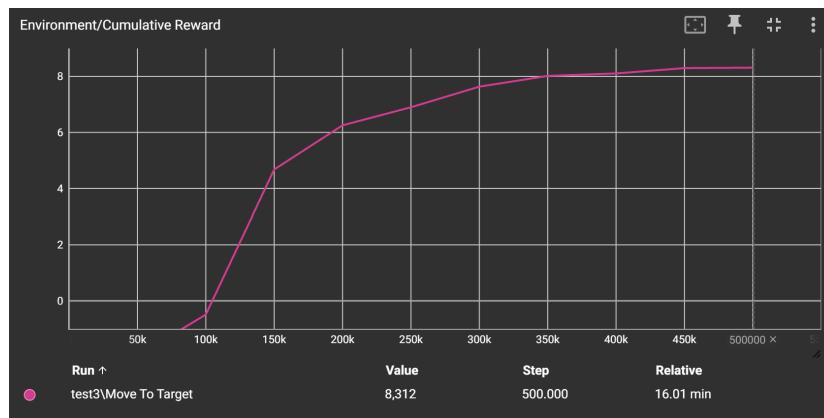


Figür 15. Az Deneme Sayısı

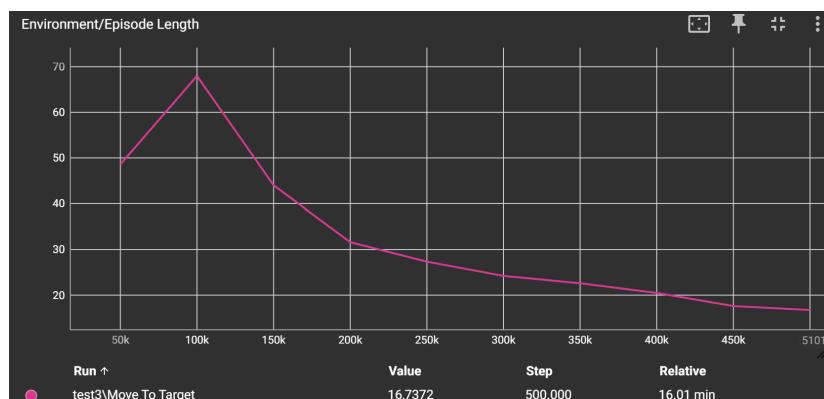


Figür 16. Çok Deneme Sayısı

Eğitim tamamlandıktan sonra TensorBoard yardımıyla sonuçlar grafik halinde görüntülenebilir. Aşağıdaki grafikler eğitilmiş bir beyine aittir.



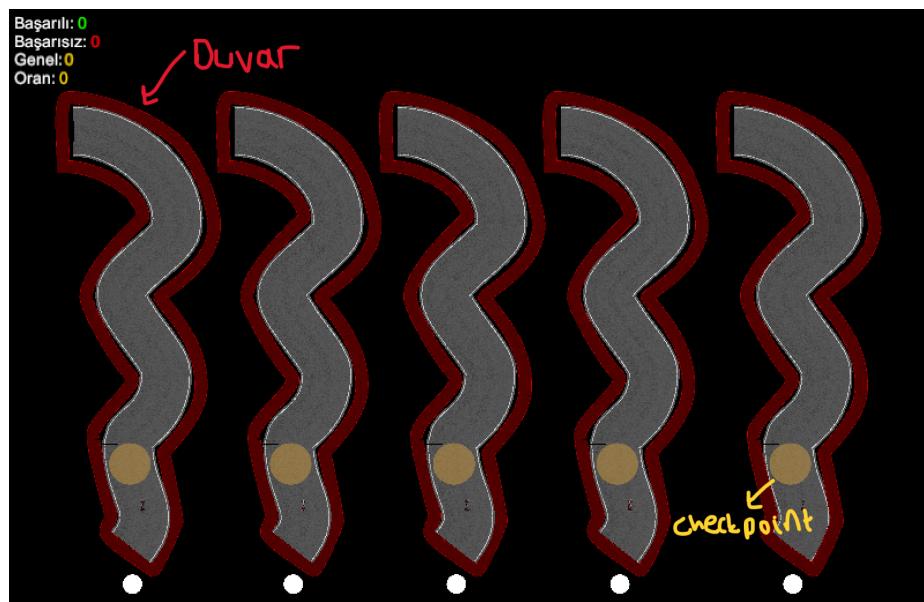
Figür 17. Toplam Ödül Sayısı Grafiği



Figür 18. Bölüm Uzunluğu Grafiği

14 Viraj Dönmeyi Öğretme Projesi

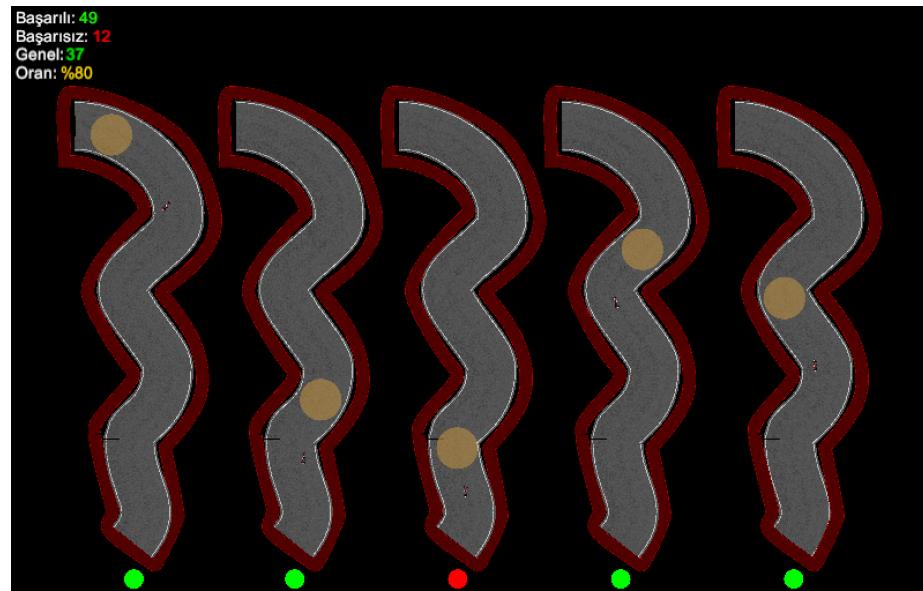
Ajamın sağa ve sola dönmemi öğrenmesi için basit virajlardan oluşan bir bölüm tasarlandı ve çoğaltıldı. Önceki örneklerle benzer şekilde ajan duvara temas ettiğinde negatif puan alıp baştan başlar. Kontrol noktasına temas ederse pozitif puan alır ve bir sonraki kontrol noktası belirir. Kontrol noktasının sırası arttıkça ajana vereceği ödül de artar.



Figür 19. Virajlı Bölüm Tasarımı [5]

14.1 Kontrol Noktaları

Ajan, kontrol noktalarından birine temas ettiğinde bir sonraki kontrol noktasını aktive eder. Kontrol noktalarının verdiği ödül sırasına göre katlanarak artar. Son kontrol noktasına ulaşan ajan en baştan başlar. Eğer ajan bu esnada duvarla temas ederse en baştan başlar ve kontrol noktaları sıfırlanır.



Figür 20. Sırayla Beliren Kontrol Noktaları

15 Önceden Tasarlanmış Pistin Öğrenime Hazır Hale Getirilmesi

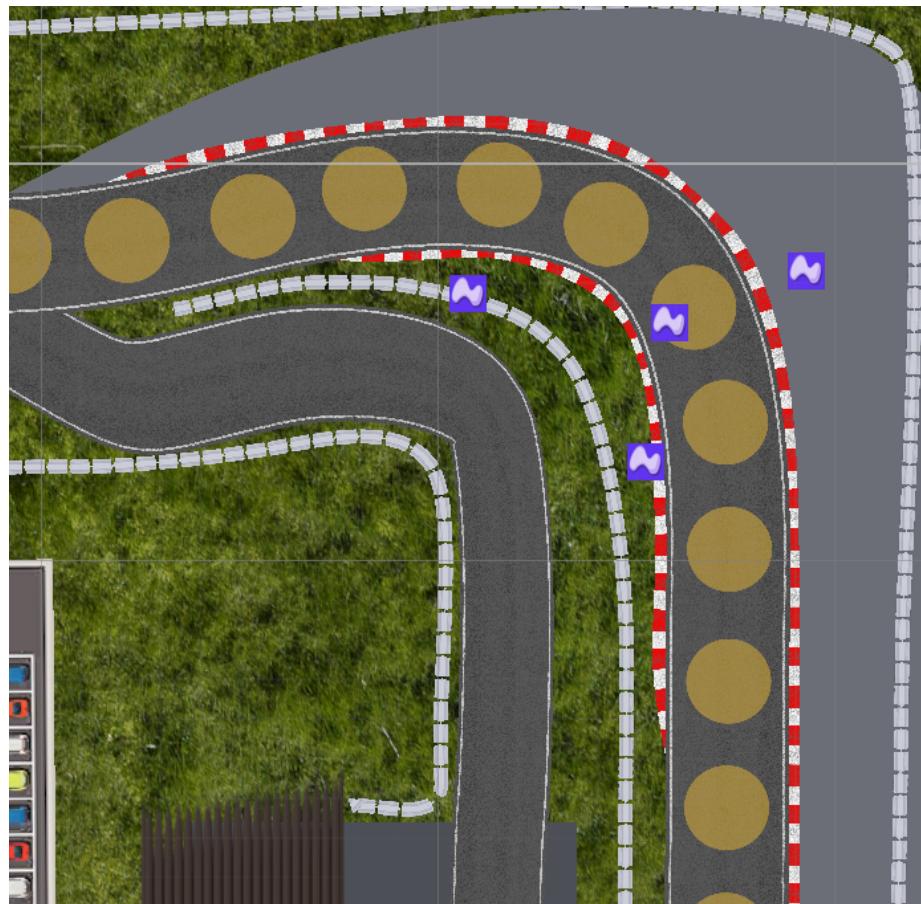
Ajanın önceden tasarlanan pist etrafında tur atabilmesi için piste oldukça sık şekilde kontrol noktaları yerleştirildi. Bu sayede ajan, bir kontrol noktasına ulaştığında, bir sonraki kontrol noktasına ulaşması için gereken çaba miktarı azaltıldı.



Figür 21. Pist

15.1 Pist Üzerindeki Kontrol Noktaları

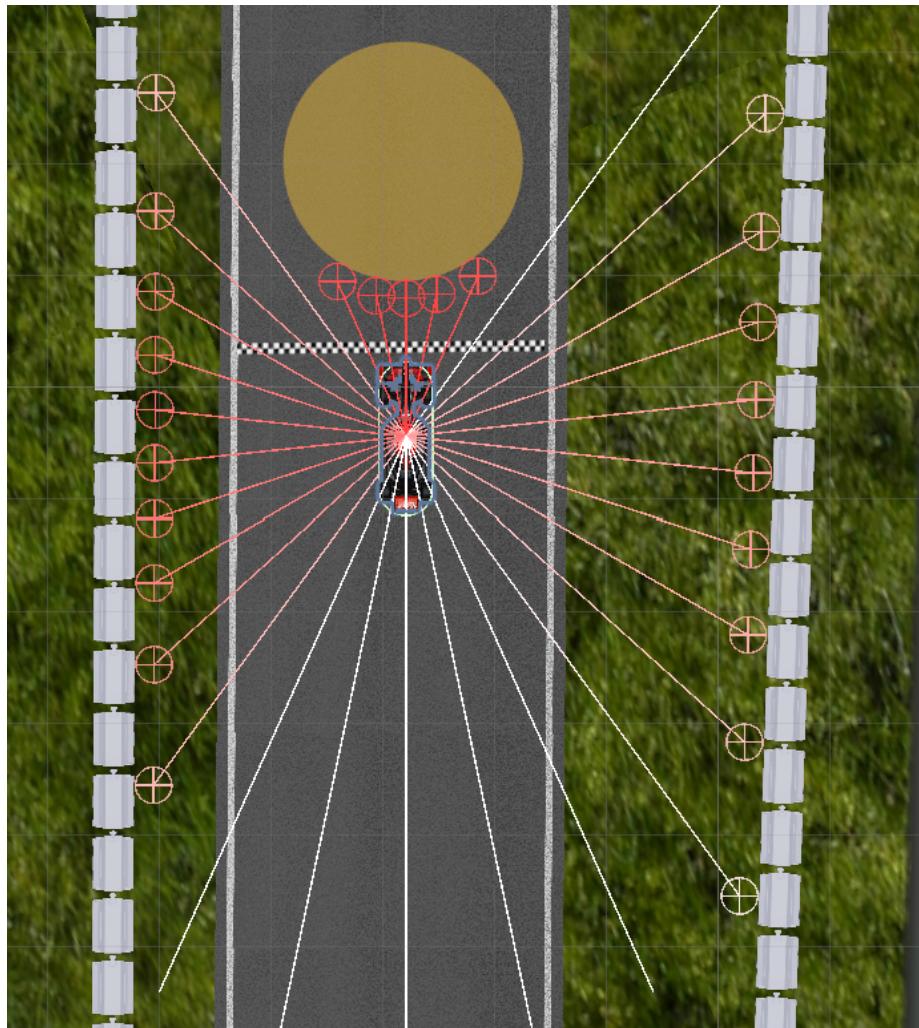
Ajan, ilk kontrol noktasına temas ettiğinde, temas ettiği kontrol noktası bir sonraki kontrol noktasına taşınır. İlk başta birinci kontrol noktası dışında aktif olmayan diğer kontrol noktaları, birinci kontrol noktasının önceden belirlenmiş pozisyonuna taşınması için kullanılır[6].



Figür 22. Pist Üzerindeki Kontrol Noktaları

16 Ray Perception Sensor 2D

Kullanıldığı objenin etrafındaki nesneleri algılamak için kullanılır[7]. Projede ajan üzerinde kullanılan bu sensör, kontrol noktalarını ve duvarları algılamak için kullanılır.



Figür 23. Ray Perception Sensor

17 Taklit Öğrenmesi (Imitation Learning)

Taklit öğrenmesi ajanın, önceden kullanıcı tarafından kaydedilen davranışları taklit ederek yeni davranışlar sergilemesini sağlayan algoritmadır. Ajanın, hedefin hangi yönde olduğunu anlaması için gerekli gözlemler eklendi.

17.1 Oluşturulan Dosyalar

Taklit öğrenme algoritmasını kullanmak için proje dosyasında "config" ve "Demos" adlı iki klasör oluşturulmalıdır[8].

17.1.1 Config Klasörü

Config Dosyasının içine MoveToTarget isimli .yaml uzantılı bir text dosyası oluşturuldu. Bu text dosyası ajanın eğitilmesi için gerekli olan parametreleri değiştirilmesine olanak sağlar. Kaydedilen demo dosyasının konumu bu dosya içinde belirtilmelidir.[9].

17.1.2 Demos Klasörü

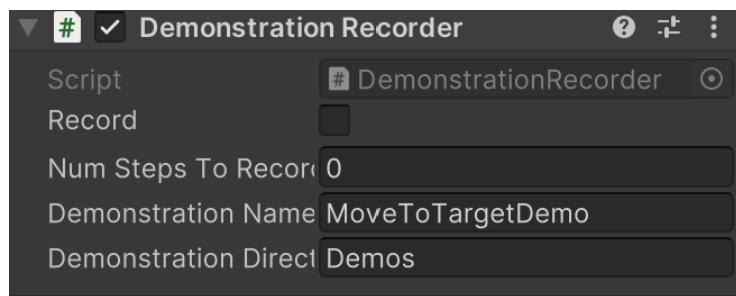
Demos klasörü kullanıcı tarafından kaydedilen davranışların kaydedildiği klasördür. .demo uzantısına sahiptir.

17.1.3 Ajanın Hedefe Göre Yönü

Ajanın gözlem topladığı kod kısmına, ajanın hedefe göre yön vektörünü alan bir gözlem eklendi[10].

18 Davranışların Kaydedilmesi

Kullanıcı davranışlarını kaydetmek için, bir ML-Agents bileşeni olan Demonstration Recorder kullanılır. Kayıt başlatmak için Record yazısının yanındaki kutucuk işaretlenmelidir.



Figür 24. Demonstration Recorder

19 Eğitimler

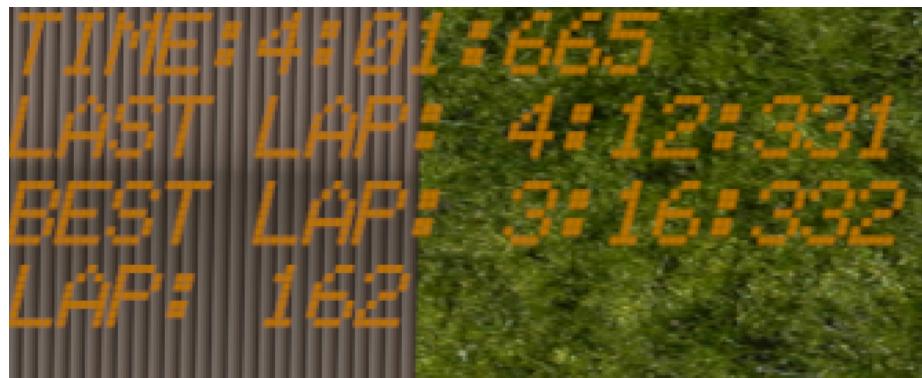
Doğru parametreler, doğru ödül, ceza miktarları girildikten sonra ajan hareket etmeye başlar. Başlarda çember şeklinde hareket eden ajan eğitim devam ettiğe düz gitmeyi, sonrasında virajları dönmeyi öğrenir. 33 turluk bir eğitim sürecinde 3 dakika 55 saniye 665 salislik en iyi tur zamanı elde edilmiştir.



Figür 25. En İyi Tur Zamanı

19.1 Eğitim-1

Önceki çalışmada ödüllü ve ceza sistemi değiştirmeden yapılan eğitimde ajan, direksiyon hareketleri ve geri gitmesi için ceza almaktadır. Bu eğitim sonucu elde edilen veriler aşağıdaki gibidir.



Figür 26. Eğitim-1

19.2 Eğitim-2

Önceki çalışmadan farklı olarak ajanın direksiyon hareketlerine ve geri gitmesine ceza verilmeyen sistemde alınan sonuçlar aşağıdaki gibidir.



Figür 27. Eğitim-2

19.3 Eğitim-3

Ödül ve ceza sistemi değiştirilmeden, pist dışına çıkışın ajanın hızının ve yol tutuşunun azaltıldığı eğitimde alman sonuçlar aşağıdaki gibidir.



Figür 28. Eğitim-3

Tablo 1: Yöntemler ve Zamanlar

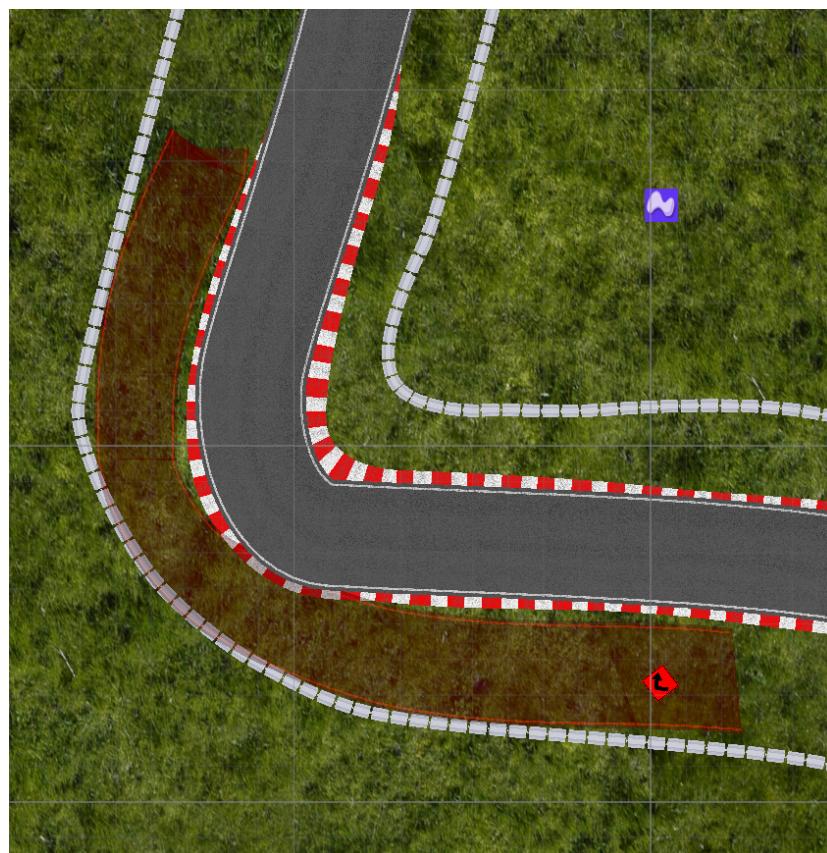
Eğitim	Yöntem	Zaman
1	Eski Ödül Ceza Sistemi	3:16:332
2	Yeni Ödül Ceza Sistemi	2:53:333
3	Hız ve Yol Tutuşu Ayarlamaları	2:46:666

20 Denemeler

Önceki eğitimlerdeki gözlemler ışığında yapılan denemeler aşağıda gösterilmiştir.

20.1 Deneme-1

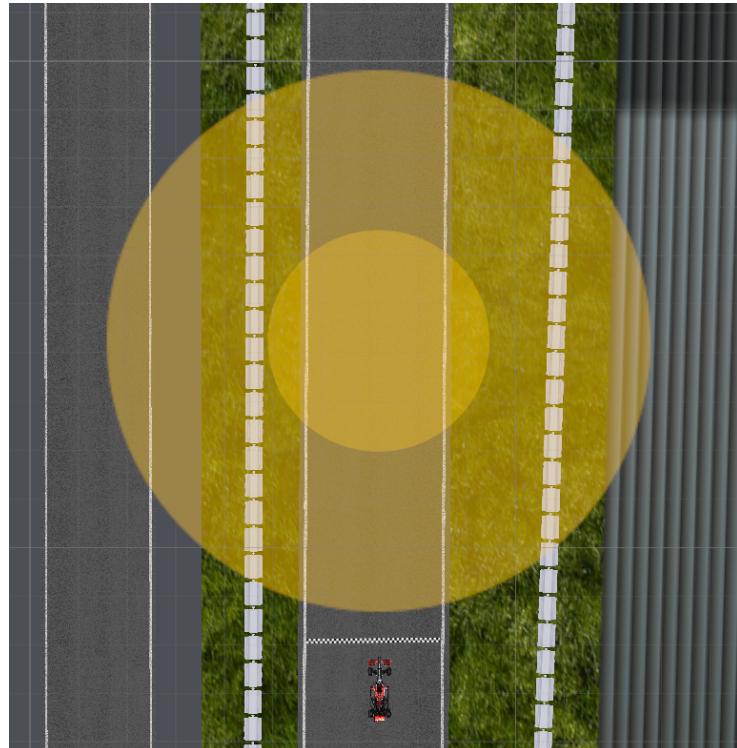
Önceki çalışmadaki ödül ve ceza sistemini değiştirmeden, ajanın sık sık hata yaptığı virajlara ceza puanı veren bölgeler yapılmıştır. Önceki çalışmalara göre bir gelişme gözlemlenmemiştir.



Figür 29. Hata Yapılan Virajdaki Ceza Bölgesi

20.2 Deneme-2

Önceki çalışmadaki ödül ve ceza sistemini değiştirmeden, kontrol noktalarının içine daha küçük kontrol noktaları eklenerek ajanın pisti daha doğru şekilde takip etmesi amaçlanmıştır. Önceki çalışmalara göre bir gelişme gözlemlenmemiştir.



Figür 30. Küçük Kontrol Noktası

20.3 Deneme-3

Yukarıda bahsedilen deneme-1 ve deneme-2 yöntemlerini birleştirerek yapılan denemedede önceki çalışmalara göre bir gelişme gözlemlenmemiştir.

20.4 Deneme-4

Deneme-1, deneme-2 ve deneme-3 tarafından kullanılan yöntemler sonucunda bir gelişme gözlemlenmediği için geçmiş çalışmada sisteme geri dönülmüştür. Farklı olarak, ajan 20 saniye içinde hedefe ulaşamazsa bir önceki kontrol noktasına geri dönmesi yerine artık ajan herhangi bir duvara çarptığında önceki kontrol noktasına geri döner. Bu ve önceki denemeler sonucu alınan süreler aşağıdaki gibidir.



Figür 31. Son Çalışmadaki En İyi Zaman

Tablo 2: Çalışmalar ve Zamanlar

Hafta	Yöntem	Zaman
10	Duvar & Kontrol Noktası Ayarlamaları	2:17:000
9	Hız & Yol Tutuşu Ayarlamaları	2:46:666
8	İlk Eğitim	3:55:665

Tablo 3: Ödüller ve Cezalar

Davranış	Ödül Artış Miktarı	Etken
Kontrol Noktasına Ulaşma	+100	Yok
Duvara Çarpma	-3500	Yok
20 Saniye İçinde Kontrol Noktasına Ulaşamazsa	-1000	Her Seferinde 2 Katı
Kontrol Noktasına Yaklaşıyorsa	+1	Her Frame Başı
Kontrol Noktasından Uzaklaşıyorsa	-3	Her Frame Başı
Kontrol Noktasının Ajana Uzaklığı	+5 / Uzaklık	Her Frame Başı
Hedefe Ulaşamadığı Süre Boyunca	-1	Her Frame Başı

21 Ajan Girdileri

Ajanın hangi girdileri gönderdiğini görsel olarak görebilmek için ekranın sağ üst köşesine ekleme yapıldı.



Figür 32. Ajan Girdileri

22 Sonuç

ML-Agents kullanılarak Unity içinde yapay zeka eğitimi yapılmaktadır. Bu proje kapsamında, ajanların araç kontrolünü öğrenmesi hedeflenmiştir. Bunun için ML-Agents kütüphanesi kullanılmış ve çeşitli eğitimler yapılmıştır. Eğitim sonuçlarına bakıldığında, ajanların belirli bir hedefe doğru hareket etme yeteneğinin geliştirildiği görülmektedir.

Kullanılan teknolojiler, Unity oyun motoru, ML-Agents kütüphanesi, Python programlama dili ve PyTorch kütüphanesidir. Bu teknolojilerin bir araya getirilmesiyle yapay zeka eğitimi için etkili bir platform oluşturulmuştur.

Proje, önceden tasarlanmış bir pist etrafında araç kontrolünü öğrenmek için çeşitli senaryoları içermektedir. Ajanların başarılı bir şekilde bu senaryoları tamamlaması, ML-Agents'in etkinliğini ve kullanışlığını göstermektedir.

Son olarak, proje ML-Agents ve Unity'nin yapay zeka eğitimi alanındaki potansiyelini ortaya koymaktadır.

Kaynakça

- [1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yoganmani, and P. Pérez, “Deep reinforcement learning for autonomous driving: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.
- [2] T. Simonini, “An introduction to unity ml-agents,” *Medium*, 2020.
- [3] T. A. Bot, “How to use unity ml agents 2.0.1,” *Youtube*, 2023.
- [4] flyyufelix, “Train donkey car in unity simulator with reinforcement learnin,” *GitHub*, 2018.
- [5] A. Thirslund, “Make organic levels!! unity sprite shape,” *Youtube*, 2018.
- [6] unknown1050, “Unity checkpoints, laps, and times,” *YouTube*, 2022.
- [7] jpschneider, “Unity ml agents ray perception sensor 2d not showing collisions,” *stackoverflow*, 2021.
- [8] CodeMonkey, “How to use machine learning ai in unity! (ml-agents),” *YouTube*, 2020.
- [9] CodeMonkey, “Teach your ai! imitation learning with unity ml-agents!checkpoints, laps, and times,” *YouTube*, 2020.
- [10] Berenger, “How to calculate direction between 2 objects(),” *Unity Discussions*, 2014.