Reinforcement Learning ile Yapay Zekaya Hayatta Kalmayı Öğretme

Hüseyin Buğra Taştan

25 Nisan 2024

Özet

"AI Learns To Survive" adlı proje, derin güçlendirme öğrenme (deep reinforcement learning) kullanarak yapay zeka ajanlarının hayatta kalma becerilerini öğrenmeye odaklanan bir araştırma veya uygulama projesidir. Projede amaç, yapay zeka ajanlarını belirli bir ortamda hayatta kalmaya yönlendirmek ve bu süreçte karşılaştıkları zorluklarla başa çıkabilmelerini sağlamaktır.

1 Giriş

Günümüzde, yapay zeka teknolojisinin hızla gelişmesiyle birlikte, yapay zeka ajanlarının gerçek dünya ortamlarında başarılı bir şekilde işlev görmesi ve hatta hayatta kalabilmesi giderek daha büyük bir önem kazanıyor. "AI Learns To Survive" (Yapay Zeka Hayatta Kalmayı Öğreniyor) projesi, bu bağlamda derin güçlendirme öğrenme tekniklerini kullanarak yapay zeka ajanlarının hayatta kalma becerilerini geliştirmeye odaklanan heyecan verici bir araştırma ve uygulama girişimidir.Projemizin temel amacı, yapay zeka ajanlarını belirli bir ortamda hayatta kalmaya teşvik etmek ve bu süreçte karşılaşacakları çeşitli zorluklarla başa çıkabilmelerini sağlamaktır. Bu zorluklar, gerçek dünya senaryolarından esinlenmiş olabileceği gibi sanal ortamlarda da simüle edilebilir.Örneğin, bir robotun doğal afetler, engeller veya diğer tehlikelerle karşılaştığı bir ortamda hayatta kalabilme yeteneği üzerine odaklanabiliriz.Derin güçlendirme öğrenme algoritmaları, projemizin temelini oluşturur. Bu algoritmalar, yapay zeka ajanlarının ortama uyum sağlamak ve belirlenmiş hedefleri başarmak için optimal eylemleri öğrenmelerini sağlar. Öğrenme süreci genellikle bir ödül sistemiyle desteklenir; ajanlar belirli görevleri başarıyla tamamladıklarında veya belirli zorlukları aştıklarında ödüllendirilirler. Bu ödül sistemi, ajanların istenilen davranışları öğrenmelerini teşvik eder."Aİ Learns To Survive" projesi, yapay zeka alanında önemli bir boşluğu doldurmayı hedefliyor. Gerçek dünya uygulamalarında kullanılabilen yapay zeka ajanlarının, değişen ve bazen de tehlikeli ortamlarda başarılı bir şekilde işlev görebilmesi, teknolojinin ilerlemesinde kritik bir adımdır. Bu proje, bu hedefe ulaşmada bir adım daha atmaktadır.

2 Literatür Çalışması

Bu makalede, derin pekiştirmeli öğrenme yöntemlerini kullanarak, yüksek boyutlu duyusal girdilerden (örneğin, görüntü veya konuşma gibi) doğrudan ajanları kontrol etmeyi öğrenmektir. Peşin başarılı pekiştirmeli öğrenme uygulamaları, bu tür alanlarda genellikle elle hazırlanmış özelliklerle birleştirilmiş doğrusal değer fonksiyonları veya politika temsillerine dayanmıştır[1].

Bu makalede,derin pekiştirmeli öğrenme modellerini daha hızlı ve daha verimli bir şekilde eğitmek ve genişletmek için asenkron yöntemlerin etkin bir şekilde kullanılmasıdır. Bu sayede, daha karmaşık ve gerçekçi problemler üzerinde daha iyi performans gösteren yapay zeka sistemleri geliştirmek mümkün olabilir[2].

Bu projede, derin sinir ağları ve ağaç araması gibi ileri öğrenme tekniklerinin kullanılmasıyla, Go oyununu oynamak için daha etkili bir yaklaşım geliştirilmiştir. Derin sinir ağları, oyun tahtasındaki durumu analiz etmek ve olası hamleleri tahmin etmek için kullanılırken, ağaç araması algoritması, bu tahminleri daha ileri seviyeye taşımak ve olası hamlelerin sonuçlarını değerlendirmek için kullanılır[3].

Bu proje, derin takviyeli öğrenme tekniklerini kullanarak genel video oyunları için yapay zeka geliştirmeyi amaçlamaktadır. Derin takviyeli öğrenme, yapay zeka ajanlarının belirli bir görevi gerçekleştirmek için çevreleriyle etkileşimde bulunarak deneyimlerinden öğrenmelerini sağlayan bir makine öğrenimi yaklaşımıdır.[4]

3 ML Agents Nedir?

ML Agents, Unity'nin yapay zeka ve makine öğrenimi modellerini eğitmek için geliştirilmiş bir araç setidir. Bu araç seti, Unity oyun motorunda oyun nesnelerini kontrol etmek için kullanılan yapay zeka algoritmalarını ve modellerini eğitmek için bir dizi araç ve kütüphane sunar. ML Agents, karmaşık oyun senaryolarında yapay zeka davranışlarını modellendirme ve eğitme sürecini kolaylaştırır.

3.1 ML Agents Kullanımı

ML Agents kullanarak yapay zeka modelleri eğitmek ve entegre etmek oldukça basittir. İşte temel adımlar:

3.1.1 Çevrenin Tanımlanması

İlk adım, yapay zeka modelini eğitmek için bir çevre tanımlamaktır. Bu çevre, oyun sahnesindeki nesneleri, hedefleri ve etkileşimleri içerir.

3.1.2 Eğitim Verilerinin Toplanması

Sonraki adım, çevrede yapay zeka modelini eğitmek için kullanılacak verilerin toplanmasıdır. Bu genellikle insan ya da önceden belirlenmiş stratejilerle oluşturulmuş verileri içerir.

3.1.3 Yapay Zeka Modelinin Eğitimi

Toplanan veriler kullanılarak yapay zeka modeli eğitilir. ML Agents, çeşitli makine öğrenimi tekniklerini destekler ve eğitim sürecini kolaylaştırır.

3.1.4 Yapay Zeka'nın Entegrasyonu

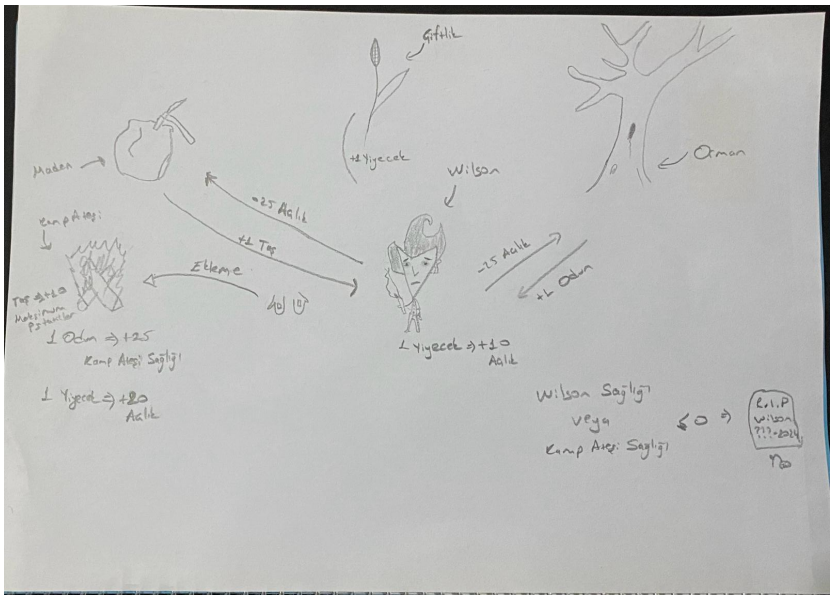
Eğitilen yapay zeka modeli, Unity oyun motoruna entegre edilir ve gerçek zamanlı olarak oyun sahnesinde kullanılır.

4 Kurulum Aşamaları

1. Anaconda Kurulumu
2. Unity Kurulumu
3. ML-Agents Kurulumu
4. Sanal Orrtam Oluşumu
5. Gerekli Eklenti Kurulumları
6. Eğitimin Başlatılması

5 Proje Taslağı

Hayatta kalma oyunları genellikle oyuncuların doğal kaynakları toplamalarını, inşa etmelerini ve düşmanlardan kaçınmalarını veya onlarla savaşmalarını gerektirir. Yapay zeka, bu tür oyunlarda oyuncuların karşılaşabileceği düşman davranışlarını, kaynak toplama stratejilerini ve oyun dünyasıyla etkileşimlerini simüle etmek için kullanılabilir.

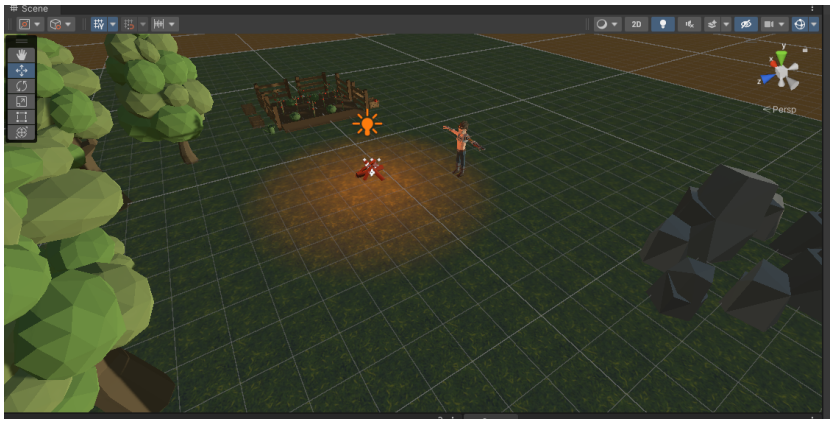


Resim 1: Proje Taslağı

6 Ortam Hazırlama Adımları

6.1 Temel Oyun Dünyası Oluşturma

- **Proje Oluşturma:** Unity'de yeni bir proje oluşturun ve gerekli ayarları yapın.
- **Sahne Oluşturma:** Sahneye bir arazi veya ortam ekleyin. Bu, oyunun geçeceği dünyayı temsil eder. Toprak, ağaçlar, kayalar gibi doğal unsurları içerebilir.
- **Işıklandırma:** Işıklandırmayı ayarlayın. Gündüz ve gece döngüsü gibi dinamik ışıklandırma efektleri eklemek, oyun dünyasını daha gerçekçi hale getirebilir.
- **Detaylar ve Atmosfer:** Ortama detaylar ekleyin; bitkiler, taşlar, su vb. Bunlar oyunun atmosferini zenginleştirecek ve oyuncuların etkileşimini artıracaktır.



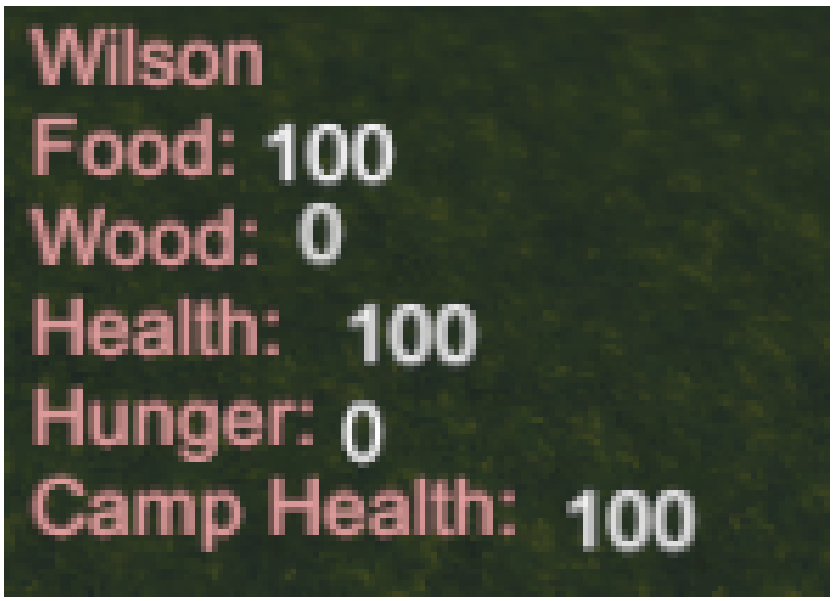
Resim 2: Ortam Oluşturma

6.2 Yapay Zeka İçin Temel Unsurlar

- **Düşman Karakterler:** Düşman karakterlerinizi tasarlayın ve oluşturun. Bu karakterler, oyuncuya karşı saldırabilir veya onlardan kaçabilir.
- **Davranış Kodlaması:** Düşmanların hareket ve davranışlarını belirleyen temel kodları yazın veya hazır yapay zeka çözümlerini kullanın. Örneğin, düşmanların oyuncuyu takip etmesi veya belirli bir alanı savunması gerekebilir.
- **Düşman AI Entegrasyonu:** Oyun içinde düşman yapay zekasını etkinleştirecek ve onların oyuncularla etkileşimini sağlayacak kodları entegre edin.

6.3 Kaynak Toplama ve İnşa Etme Sistemleri

- **Kaynakları Tanımlama:** Oyuncunun kaynakları toplayabileceği ve bunları kullanarak yapılar inşa edebileceği bir sistem oluşturun. Bu kaynaklar odun, taş, besin gibi şeyleri içerebilir.
- **Yapıları Tanımlama:** Oyuncunun inşa edebileceği yapıları belirleyin. Bu yapılar barınaklar, savunma kuleleri, üretim tesisleri vb. olabilir.
- **İnşa ve Üretim Mekanikleri:** Oyuncunun kaynaklarını kullanarak yapılar inşa etmesini ve bu yapılar aracılığıyla kaynakları işlemesini sağlayacak mekanikleri tasarlayın ve kodlayın.



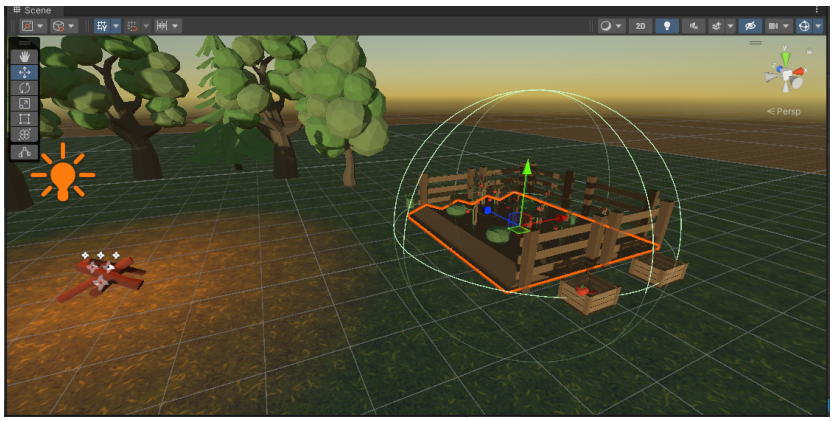
Resim 3: Kaynak Belirleme

6.4 Yapay Zeka ile Etkileşimler

- **Düşman Etkileşimleri:** Yapay zekanın kaynakları toplaması ve inşa etmesi için gerekli kodları yazın veya entegre edin. Ayrıca, düşmanların oyuncunun inşa ettiği yapıları hedef alması veya onları yok etmeye çalışması için gerekli kodları ekleyin.
- **Oyuncu-Yapı Etkileşimleri:** Oyuncunun inşa ettiği yapılarla yapay zeka arasında etkileşimleri sağlayın. Örneğin, düşmanların oyuncunun inşa ettiği yapıları hedef alması veya bu yapıları savunması gerekebilir.

7 Trigger Kullanımı

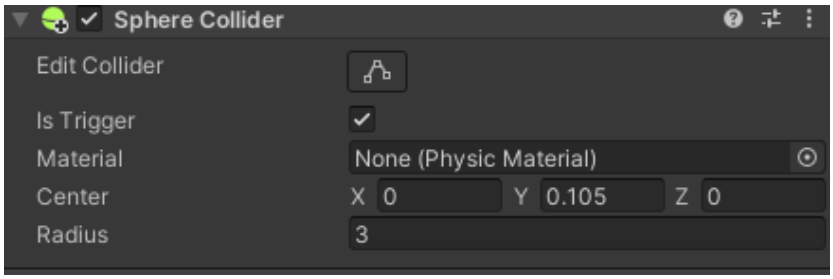
Trigger'lar, bir nesnenin içine giren diğer bir nesne tarafından algılanan alanlardır. Bir nesne bir Trigger alanına girdiğinde, bu olay bir tetikleyici (trigger) olarak adlandırılır ve belirli bir işlevi başlatır. Örneğin, bir oyuncunun bir tarlaya girdiğinde bir tetikleyici tetiklenerek hasat yapabilir.



Resim 4: Trigger ve Collider Bileşeni

7.1 Trigger'ı Etkinleştirme

Unity'de bir nesnenin bir Trigger olarak işlev görmesi için, bir Collider bileşenine sahip olması ve "Is Trigger" özelliğinin etkinleştirilmiş olması gerekir.



Resim 5: Trigger Etkinleştirme

7.2 Trigger Olayları

Bir Trigger ile ilişkili olaylar genellikle OnTriggerStay, OnTriggerEnter ve OnTriggerExit fonksiyonlarıyla ele alınır. OnTriggerEnter, bir nesne Trigger alanına girdiğinde tetiklenir. OnTriggerStay, bir nesne Trigger alanında kaldığı sürece devam eder. Son olarak, OnTriggerExit, bir nesne Trigger alanından çıktığında tetiklenir.

8 Collider Kullanımı

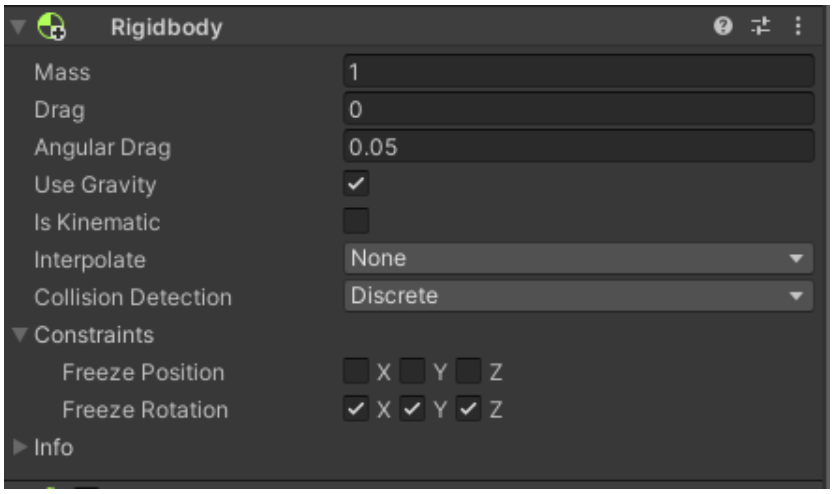
Collider, nesnelerin fiziksel çarpışmalarını algılamak için kullanılır. İki nesne çarpıştığında, Collider'lar bu çarpışmayı algılar ve Unity'nin fizik motoruna bilgi gönderir, böylece nesneler arasındaki etkileşim gerçekleşir.

8.1 Collider Türleri

Unity'de farklı Collider türleri vardır, bunlar arasında BoxCollider, SphereCollider, CapsuleCollider, ve MeshCollider bulunur. Her biri farklı şekil ve formlara sahiptir ve nesnelerin çarpışma algılama şekillerini etkiler.

8.2 Fiziksel Özellikler

Collider'lar, nesnelerin fiziksel özelliklerini belirlemek için kullanılabilir. Örneğin, bir nesnenin ağırlığını, sürtünmesini ve elastikiyetini ayarlamak için Collider bileşeninin özellikleri kullanılabilir.



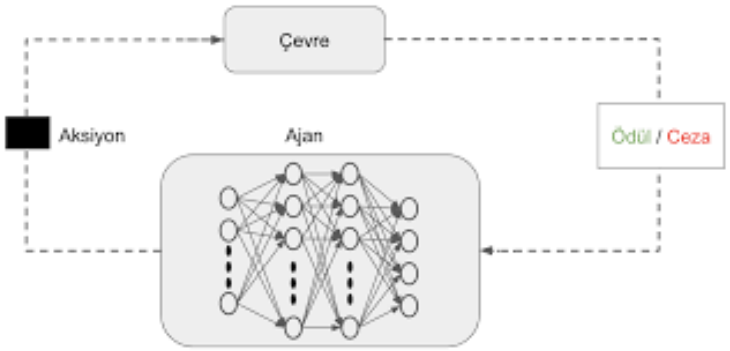
Resim 6: Rigidbody

9 Örnek Kod Parçası

```
using UnityEngine;

public class yemek_hasat : MonoBehaviour
{

    private void OnTriggerEnter(Collider other)
    {
        Envanter.foodCount += 1;

    }
}

using UnityEngine;
using TMPro;

public class Envanter : MonoBehaviour
{
    public static int foodCount = 0;

    [SerializeField] private TextMeshProUGUI foodText;

    // Güncelleme kare başına bir kez çağrılır
    void Update()
    {
        foodText.text = foodCount.ToString();

    }
}
```

Yukarıdaki örnek, çiftliğin bir oyuncu tarafından hasat edilmesini sağlamak için bir Trigger kullanır. OnTriggerEnter fonksiyonu, oyuncu bir Trigger alanına girdiğinde tetiklenir ve foodCounta 1 ekler.

10 Pekiştirmeli Öğrenme Algoritması



Resim 7: Makine Öğrenimi [5]

1. **Gözlemleme:** Model, çevresindeki ortamı gözler ve bu ortam hakkında bilgi toplar. Örneğin, bir video oyunundaki bir karakter, oyunun dünyasını ve karakterin durumunu gözlemleyebilir.

2. **Karar Verme:** Model, mevcut gözlemlerine dayanarak bir eylem seçer. Bu eylem, belirli bir hedefe ulaşmak için tasarlanmıştır. Örneğin, bir video oyun karakteri, bir düşmanla savaşmak veya bir engeli aşmak için bir eylem seçebilir.

3. **Eylemi Uygulama:** Model seçilen eylemi gerçekleştirir ve çevrede bir değişiklik oluşturur. Örneğin, bir video oyununda karakter hareket eder veya bir silah ateş eder.

4. **Geri Bildirim Alınması:** Model, geri bildirim alır. Bu geri bildirim, seçilen eylemin ne kadar başarılı veya başarısız olduğunu belirtir. Örneğin, karakterin canını azaltan bir düşman saldırısı veya bir hedefi vurma başarısı olabilir.

5. **Ödül veya Cezalandırma:** Model, aldığı geri bildirime göre bir ödül veya ceza alır. Eylem başarılıysa, genellikle bir ödül alır; başarısızsa bir ceza alır. Örneğin, düşmanı yenmek başarı olarak kabul edilirken, bir engelde takılıp kalmak cezalandırılır.

6. **Öğrenme:**Model, aldığı geri bildirime dayanarak öğrenir. Başarılı eylemleri teşvik eden davranışları öğrenirken, başarısız eylemlerden kaçınmayı öğrenir.

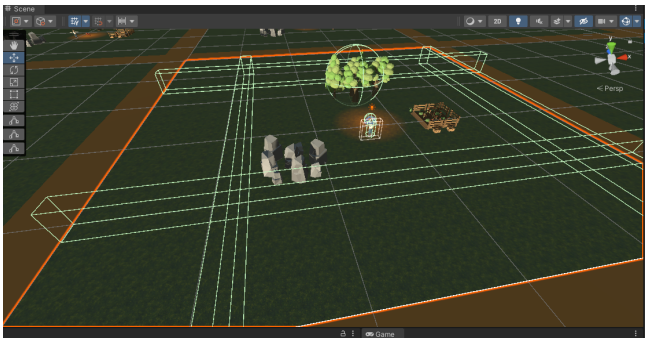
7. **Yeniden Gözlemleme ve Yeniden Karar Verme:** Model, güncel gözlemlerine dayanarak yeni bir eylem seçer ve süreç tekrarlanır.

11 ML-Agents İçin Gelişmiş Kavramlar

Unity ML-Agents'in temel kavramlarına hakim olduktan sonra, gelişmiş teknikler ve kavramlarla daha derinlemesine bir anlayış geliştirebiliriz. Bu bölümde, ML-Agents için önemli olan birkaç gelişmiş kavramı ele alacağız:

11.1 Reward Shaping

Reward Shaping, ajanın davranışlarını yönlendirmek ve hızlandırmak için ödüllendirme işlemine ek olarak kullanılan bir tekniktir. Bu teknik, ajanın istenen davranışları daha hızlı öğrenmesini sağlayabilir, ancak dikkatlice uygulanmalıdır çünkü yanlış bir şekilde kullanıldığında ajanın performansını olumsuz etkileyebilir.



Resim 8: Ödül Ceza Sistemi

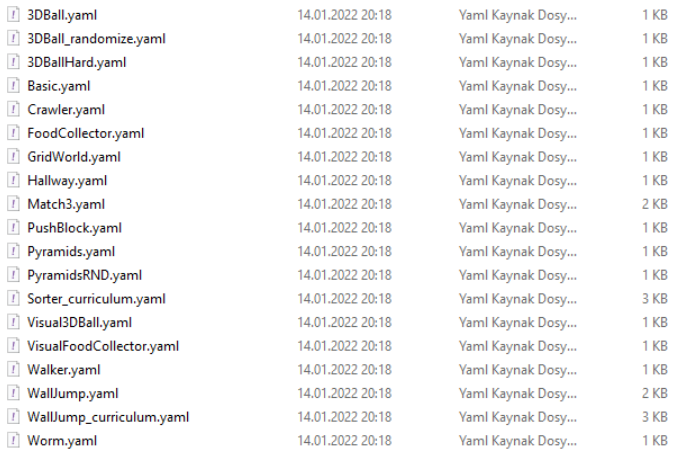
11.2 Curriculum Learning

Curriculum Learning, ajanın öğrenme sürecini daha etkili hale getirmek için kullanılan bir öğrenme stratejisidir. Bu strateji, ajanın daha basit görevlerle başlamasını ve daha sonra giderek zorlaşan görevlere geçiş yapmasını sağlar. Bu, ajanın daha hızlı ve daha kararlı bir şekilde öğrenmesini sağlayabilir.

```
private void OnTriggerEnter(Collider other)
{
    if(other.gameObject.tag=="odun")
    {
        AddReward(10f);
        EndEpisode();
    }
    if(other.gameObject.tag=="wall")
    {
        AddReward(-5f);
        EndEpisode();
    }
}
```

11.3 Proximal Policy Optimization (PPO)

PPO, ML-Agents'te sıkça kullanılan bir eğitim algoritmasıdır. Policy gradient metotlarının bir türü olan PPO, stabil ve hızlı bir şekilde ajanları eğitmek için tasarlanmıştır. Ayrıca, genellikle büyük ölçekli eğitim setleri üzerinde etkili bir şekilde çalışır.



Resim 9: PPO Örnek Dosyalar

12 Gelişmiş Ortam Tasarımı

Gelişmiş bir ML-Agents ortamı tasarlarken dikkate alınması gereken birkaç önemli faktör bulunmaktadır. Bu bölümde, ML-Agents ile gelişmiş ortamların tasarımı ve yönetimi üzerine bazı ipuçlarını ele alacağız:

12.1 Kompleks Ortamların Oluşturulması

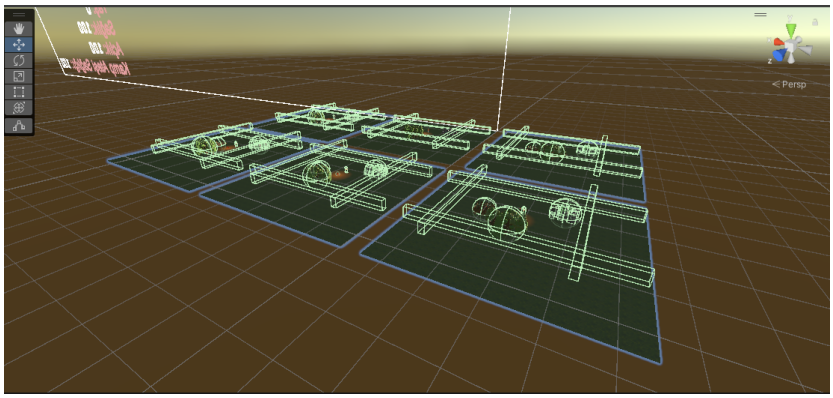
ML-Agents ile kompleks ve etkileşimli ortamlar oluşturmak için Unity'nin özelliklerinden yararlanabiliriz. Bu, ajanların gerçek dünya senaryolarına daha yakın bir şekilde eğitilmesini sağlayabilir.

12.2 Çoklu Ajan Ortamları

ML-Agents, birden fazla ajanı aynı ortamda eğitmeyi ve simüle etmeyi destekler. Bu, işbirliği ve rekabet temelli senaryolar için idealdir. Çoklu ajan ortamları tasarlarken dikkat edilmesi gereken özel durumlar bulunmaktadır.

13 Ajanların Gelişmiş Eğitimi

Ajanların gelişmiş eğitimi, ML-Agents kullanıcılarının sık sık karşılaştığı bir konudur. Bu bölümde, ajanların daha hızlı, daha stabil ve daha verimli bir şekilde eğitilmesini sağlamak için kullanılan bazı gelişmiş eğitim tekniklerini ele alacağız:



Resim 10: Çoklu Eğitim Örneği

13.1 Eğitim Hiperparametrelerinin Ayarlanması

ML-Agents eğitim sürecinde birçok hiperparametre vardır ve bu parametrelerin doğru bir şekilde ayarlanması önemlidir. Örneğin, öğrenme hızı, eğitim devri sayısı, ödül şekillendirme faktörleri gibi parametrelerin dikkatlice ayarlanması gerekebilir.

13.2 Eğitim Verimliliğini Artırmak İçin Teknikler

Ajanların eğitim verimliliğini artırmak için birçok teknik mevcuttur. Bunlar arasında deneyim tekrarı, örnek çeşitliliği sağlama, eğitim verilerinin dengelemesi gibi teknikler bulunabilir. Bu teknikler, ajanların daha hızlı ve daha stabil bir şekilde öğrenmelerini sağlayabilir.

13.3 Hızlandırılmış Eğitim için Paralel Eğitim

ML-Agents, paralel eğitim modunu destekler. Bu, birçok ajanın aynı anda eğitilmesini ve eğitim sürecinin hızlandırılmasını sağlar. Paralel eğitim, büyük ölçekli eğitim setleri üzerinde çalışırken özellikle faydalıdır ve eğitim süresini önemli ölçüde azaltabilir.

14 Model Yönetimi ve Hata Ayıklama

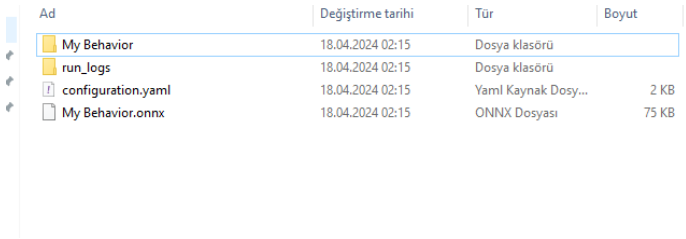
ML-Agents ile çalışırken, eğitilen modellerin yönetimi ve hata ayıklama süreçleri önemlidir. Bu bölümde, model yönetimi ve hata ayıklama için bazı önemli konuları ele alacağız:

14.1 Model Kontrolü ve Yönetimi

Eğitilen modellerin kontrolü ve yönetimi, ML-Agents kullanıcılarının sıklıkla karşılaştığı bir konudur. Bu bölümde, eğitim sırasında ve sonrasında modellerin nasıl yönetileceği, kaydedileceği ve yüklenip kullanılacağı gibi konuları ele alacağız.

14.2 Eğitim Sırasında Hata Ayıklama

ML-Agents ile çalışırken, eğitim sırasında ortaya çıkan hataları tanımlamak ve çözmek önemlidir. Bu bölümde, eğitim sırasında sık karşılaşılan hataları ve bunların nasıl çözülebileceğini ele alacağız. Ayrıca, eğitim sürecini izlemek ve modelin performansını değerlendirmek için kullanılabilecek araçlar hakkında bilgi vereceğiz.



Resim 11: Model Kontrolü

15 Performans Optimizasyonu

ML-Agents ile çalışırken, performans optimizasyonu önemli bir konudur, özellikle büyük ölçekli eğitim setleri veya karmaşık ortamlar kullanırken. Bu bölümde, performansı artırmak için kullanılan bazı teknikleri ele alacağız:

15.1 Eğitim Süresini ve Maliyeti Azaltma Yöntemleri

Eğitim süresini ve maliyetini azaltmak için kullanılabilecek çeşitli teknikler vardır. Bu teknikler arasında veri paralelleştirmesi, model paralelleştirmesi, dağıtık eğitim ve donanım optimizasyonu gibi yöntemler bulunabilir.

15.2 İleri Algoritmaların ve Tekniklerin Kullanımı

ML-Agents'in yanı sıra, performansı artırmak için kullanılabilecek daha gelişmiş makine öğrenimi algoritmaları ve teknikler de bulunmaktadır. Bu bölümde, daha gelişmiş algoritmaların ve tekniklerin nasıl kullanılabileceğini ve ML-Agents ile entegrasyonunun nasıl sağlanabileceğini ele alacağız.

Kaynakça

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, "Deep reinforcement learning for general video game ai," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, IEEE, 2018.

[5] M. F. D. Bakhtiyar Ospanov, "Implementation of tennis game simulation with reinforcement learning," *DergiPark*, 2023.