BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
SEMİNER DERSİ PROJESİ

Oyun Programlamada Yapay Zeka

Mehmet Emir ERDEM

2022-2023 BAHAR DÖNEMİ

ÖNSÖZ

Bu kılavuz, Bursa Teknik Üniversitesi Bilgisayar Mühendisliği 3. Sınıf Bahar Dönemi Seminer dersi kapsamında hazırlanmıştır. Amacı, oyun programlama ve yapay zeka kavramlarının temel düzeyde tanıtılması ve bu kavramların oyunlara nasıl entegre edilebileceğinin anlatılmasıdır. Kılavuz, Unity oyun motoru üzerinden proje oluşturma sürecini içermekte ve oluşturulan projenin yapay zeka bileşenlerinin adım adım nasıl eklenmesi gerektiğini açıklamaktadır.

Mehmet Emir ERDEM

Bursa 2023

İÇİNDEKİLER

ÖZET

Bu kılavuz içerisinde, oyun programlamada yapay zeka konusu ele alınmıştır. Yapay zekanın oyunlardaki önemi ve rolü incelenerek, oyun karakterlerinin davranışlarından düşman yapay zekasına kadar birçok alanda yapay zekanın nasıl kullanıldığı açıklanmıştır. Kılavuzda ayrıca oyun programlama kavramı, yapay zekanın temel kavramları, algoritmaları ve veri yapılarına değinilmiştir. Yapay zeka ile oyunlarda karşılaşılan zorluklar ve bu zorlukları aşmak için kullanılan çözümler ele alınmış, gelecekte yapay zekanın oyun programlamasındaki potansiyeli tartışılmıştır.

Uygulama kısmında, Unity oyun motorunu kullanarak yapay zekanın bir projeye nasıl entegre edildiği örneklenmiştir. Bu süreçte, Python programlama dilinin gerekli kütüphaneleri kullanılarak yapay zeka teknikleri uygulanmıştır. Unity'nin sunduğu olanaklarla birlikte, yapay zeka özelliklerinin projeye nasıl eklenmesi gerektiği adım adım gösterilmiştir.

1 GENEL BİLGİLER

1.1 Oyun Programlama Nedir?

Oyun programlama, video oyunlarının tasarlanması ve geliştirilmesinde kullanılan bir yazılım disiplinidir. Oyun programlama, bilgisayar oyunları veya mobil oyunlar gibi çeşitli oyun türlerini içerebilir. Bu süreçte, geliştiriciler, oyun dünyasını oluşturmak, oyun mekaniğini uygulamak ve oyuncuların etkileşimde bulunabileceği bir deneyim sunmak için yazılım kodları kullanır. Oyun programlaması, grafiklerin, ses efektlerinin, yapay zekanın, kullanıcı arabirimlerinin ve diğer oyun bileşenlerinin oluşturulmasını da içerebilir. Programcılar, oyun motorları veya oyun geliştirme araçları gibi yardımcı teknolojileri kullanarak, oyunun tüm yönlerini kontrol etmek ve gerçekleştirmek için programlama dilleri ve algoritmaları kullanır. Oyun programlaması, yaratıcı bir süreç olarak kabul edilirken, aynı zamanda teknik zorluklar ve optimizasyon gereksinimleri de içerir. Oyun programcıları, eğlenceli ve sürükleyici oyun deneyimleri sağlamak için sanat ve bilimi bir araya getirerek, oyun endüstrisine yenilikçi ve ilgi çekici oyunlar sunmada önemli bir rol oynarlar.

1.2 Oyun Programlama Tarihçesi

Oyun programlama tarihi, bilgisayar ve video oyunlarının gelişimiyle birlikte şekillenmiştir. İlk oyunlar basit grafikler ve sınırlı işlevsellikle başladı. 1950'lerdeki bilgisayarlar, oyunları çalıştırmak için yeterli güce sahip değildi ve genellikle basit matematiksel problemlerin çözümüne odaklanıyordu.

1960'ların sonlarında ve 1970'lerin başlarında, arcade makineleri ve ev bilgisayarları popülerlik kazandı. Bu dönemde Spacewar!, Pong ve Atari gibi oyunlar, oyun programlamasının temel taşları oldu. Oyunlar genellikle donanımın sınırlamalarına uygun olarak programlanıyordu ve görsel ve işitsel öğeler oldukça basitti.

Şekil 1: *PDP-1 bilgisayarında "Spacewar!" isimli video oyunu*

1980'ler ve 1990'lar, oyun programlamasının büyük bir ilerleme kaydettiği dönemlerdi. Ev bilgisayarları, video oyun konsolları ve kişisel bilgisayarlar oyun endüstrisinin büyümesini tetikledi. Bu dönemde Super Mario Bros., Pac-Man, The Legend of Zelda gibi ikonik oyunlar ortaya çıktı.

Şekil 2: *"Super Mario Bros" oyunundan bir ekran görüntüsü*

2000'ler ve sonrasında ise oyun programlaması büyük bir devrim geçirdi. Oyun motorları ve geliştirme araçları daha gelişmiş hale geldi. Unreal Engine ve Unity gibi oyun motorları, geliştiricilere güçlü araçlar ve hızlı prototipleme imkanı sağladı. Yapay zeka, daha sofistike ve gerçekçi oyun karakterleri ve düşmanlarının geliştirilmesinde önemli bir rol oynamaya başladı.

Şekil 3: *Unity oyun motoru logosu*

Günümüzde oyun programlaması, yüksek performanslı grafikler, karmaşık fizik simülasyonları, etkileyici yapay zeka ve çok oyunculu oyun deneyimleri gibi birçok gelişmiş özelliği içeren büyük ve karmaşık projeleri kapsamaktadır. Sanal gerçeklik, artırılmış gerçeklik ve bulut tabanlı oyunlar gibi yeni teknolojiler, oyun programlamasının geleceğini şekillendiren alanlardır.

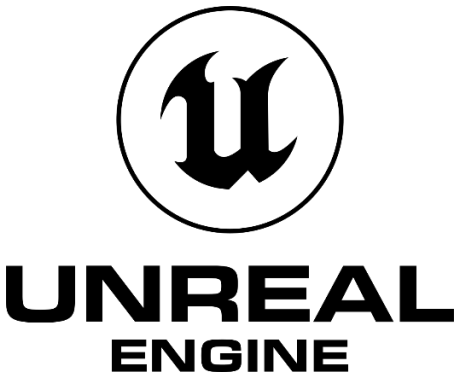
Oyun programlama tarihçesi, teknolojik ilerlemeler ve yaratıcı yeniliklerle birlikte sürekli olarak evrim geçirmektedir. Bugün, oyun programlaması büyük bir endüstri haline gelmiş ve oyunlar milyonlarca oyuncuya ulaşan popüler ürünler haline gelmiştir.

1.3 Oyun Programlamada Kullanılan Teknolojiler

Oyun programlamasında kullanılan teknolojiler, oyunların geliştirilmesi ve işlevsel bir şekilde çalışması için önemlidir. Bu teknolojiler oyun türüne, platformuna ve geliştirici tercihlerine göre değişiklik gösterebilir. Aşağıda bahsedilen teknolojiler, genel olarak oyun programlamasında yaygın olarak kullanılan ve temel öneme sahip olanlardır.

1.3.1 Oyun Motorları

Oyun motorları, oyunların geliştirilmesinde temel yapıları sağlayan yazılımlardır. Popüler oyun motorları arasında Unity, Unreal Engine ve CryEngine bulunur. Oyun motorları, grafiklerin, fizik simülasyonunun, sesin, animasyonların ve kullanıcı girişinin yönetimi gibi temel işlevleri sunar. Bu motorlar, oyun geliştirme araçları ve programlama dilleriyle entegre çalışarak geliştiricilere kolaylık sağlar.



Şekil 4: Unreal Engine oyun motoru logosu

1.3.2 Programlama Dilleri

Oyun programlamasında birçok farklı programlama dili kullanılır. C++ ve C# gibi diller, performans ve verimlilik sağlama açısından tercih edilen dillerdir. Python, JavaScript, Lua gibi diller de oyun programlamasında kullanılan diğer seçenekler arasında yer alır. Bu diller, oyunun belirli bölümlerini veya oyun senaryosunu yönetmek için kullanılabilir.

1.3.3 Grafik Motorları ve API'ler

Grafik motorları ve API'ler, oyunların görsel yönünü yönetmek için kullanılır. DirectX ve OpenGL gibi API'ler, grafiklerin ve 3D efektlerin oluşturulması ve işlenmesinde kullanılır. Vulkan ve Metal gibi yeni nesil API'ler ise yüksek performans ve çoklu platform desteği sağlar. Oyun motorları içinde bulunan grafik motorları da oyunların görsel kalitesini artırmak için kullanılan önemli araçlardır.

1.3.4 Veritabanı ve Ağ Teknolojileri

Çok oyunculu veya çevrimiçi oyunlar gibi durumlarda veritabanı ve ağ teknolojileri kullanılır. Veritabanı teknolojileri, oyuncu verilerini saklamak, oyun ilerlemesini yönetmek veya liderlik tablolarını tutmak gibi işlevlerde kullanılır. MySQL, PostgreSQL gibi ilişkisel veritabanları veya Redis, MongoDB gibi NoSQL veritabanları bu amaçla kullanılabilir. Ağ teknolojileri ise oyuncular arasındaki etkileşimi yönetir, çok oyunculu oyunlar için sunucu tarafı kodlamayı ve ağ bağlantısını sağlar. TCP/IP, UDP, WebSocket, Photon Server gibi teknolojiler bu alanda kullanılan örneklerdir.

1.4 Yapay Zeka Nedir?

Yapay zeka, bilgisayar sistemlerinin insan benzeri zekaya sahip olabilmesini hedefleyen bir disiplindir. Yapay zeka, bilgisayarların karmaşık problemleri çözebilmesi, öğrenme yeteneği kazanabilmesi, kararlar alabilmesi ve insan benzeri düşünme süreçlerini taklit edebilmesi üzerine odaklanır.

Yapay zeka, veri analizi, desen tanıma, örüntü tanıma, doğal dil işleme ve makine öğrenimi gibi alanları içerir. Bilgisayarlar, büyük miktardaki veriyi analiz ederek örüntüler ve ilişkiler keşfedebilir, dilin anlamını anlayabilir, görüntüleri tanıyabilir ve öğrenme süreciyle bilgi ve deneyim kazanabilir.

Makine öğrenimi, yapay zekanın temel bir parçasıdır. Makine öğrenimi, bilgisayarların veriye dayalı olarak öğrenme yeteneği kazanmasını sağlar. Algoritmalar ve istatistiksel modeller kullanılarak, bilgisayarlar veri setlerini analiz eder, kalıpları tanır ve gelecekteki verileri tahmin edebilir.

Şekil 5: *Yapay zeka, makine öğrenmesi ve derin öğrenme ilişkisinin açıklanması*

Yapay zeka, birçok sektörde kullanılmaktadır. Örneğin, otomotiv endüstrisinde sürücüsüz araçlar, sağlık sektöründe tanı ve tedavi destek sistemleri, finansal hizmetlerde risk analizi ve pazarlama alanında kişiselleştirilmiş öneriler gibi birçok uygulama yapay zekaya dayanmaktadır.

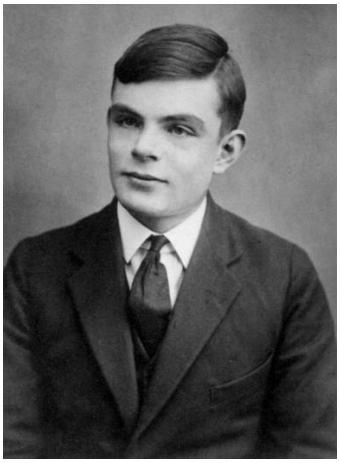
Yapay zekanın ileri düzey formları, derin öğrenme ve sinir ağları gibi teknikleri içerir. Bu teknikler, insan beyninin işleyişine benzer bir şekilde çalışan yapay sinir ağları kullanarak karmaşık problemleri çözebilme yeteneğini sağlar.

Yapay zeka, teknoloji alanında sürekli olarak gelişmekte olan bir alan olup gelecekte daha da önem kazanması beklenmektedir. Potansiyeli, insan yaşamını daha da kolaylaştırma, yenilikçilik ve verimlilik alanlarında sınırsız olanaklar sunmaktadır.

1.5 Yapay Zeka Tarihçesi

Yapay zeka (YA) kavramı, insan benzeri düşünme ve zeka yeteneklerini bilgisayar sistemlerine aktarma fikriyle ortaya çıkmıştır. Yapay zeka çalışmaları, 1950'lerin ortalarında başlamış ve o zamandan beri büyük bir evrim geçirmiştir.

Yapay zeka çalışmalarının temelleri, 1950'lerin ortalarında Alan Turing, John McCarthy, Marvin Minsky, Nathaniel Rochester ve Claude Shannon gibi bilim insanları tarafından atıldı. Turing Testi, bilgisayarların insan gibi düşünebilme yeteneğine sahip olup olmadığını belirlemek için geliştirildi. 1956'da Dartmouth Konferansı, yapay zeka araştırmalarının resmi olarak başladığı dönüm noktası oldu.



Şekil 6: Alan Turing

1970'ler ve 1980'ler döneminde sembolik yapay zeka yöntemleri popülerlik kazandı. Bilgisayarlar, semboller ve sembolik işlemler kullanarak problemleri çözmeye çalıştı. Örnek olarak, uzman sistemler ve dil işleme alanında çalışmalar yapıldı. Ancak sembolik yaklaşımın sınırlamaları nedeniyle, yapay zeka alanında bir hayal kırıklığı yaşandı ve "yapay zeka kışı" dönemi olarak adlandırılan bir gerileme dönemi başladı.

1990'lar ve 2000'ler döneminde yapay zeka araştırmaları yeniden canlandı. İstatistiksel yöntemlerin ve makine öğreniminin gelişmesiyle birlikte yeni yaklaşımlar ortaya çıktı. Destek vektör makineleri, karar ağaçları ve derin öğrenme gibi teknikler, yapay zekanın sınıflandırma, tanıma ve tahmin gibi alanlarda büyük ilerlemeler kaydetmesini sağladı.

2010 ve sonrasında yapay zeka, büyük bir ivme kazandı. Derin öğrenme ve sinir ağları gibi teknikler, otonom araçlar, sesli asistanlar, görüntü tanıma sistemleri ve doğal dil işleme uygulamaları gibi birçok alanda çığır açıcı gelişmeler sağladı. Büyük veri ve daha güçlü bilgisayar donanımları, yapay zekanın potansiyelini daha da artırdı. Geliştiriciler, açık kaynaklı yapay zeka kütüphaneleri ve bulut tabanlı hizmetlerle daha kolay erişilebilir ve uygulanabilir çözümler geliştirebilmektedir.

Bugün, yapay zeka, otomasyon, sağlık, ulaşım, güvenlik, finans ve eğitim gibi birçok sektörde kullanılmaktadır. Yapay zeka teknolojileri, insan yaşamını kolaylaştırmak, problemleri çözmek ve yeni keşifler yapmak için büyük bir potansiyele sahiptir. Gelecekte yapay zeka alanında daha da büyük ilerlemeler ve yenilikler beklenmektedir.

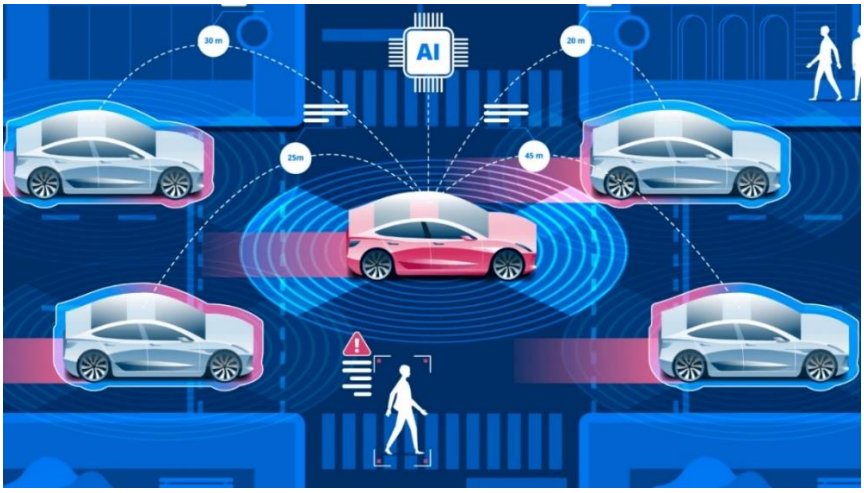
1.6 Yapay Zekanın Kullanım Alanları

Otomasyon: Yapay zeka, endüstriyel otomasyon süreçlerinde büyük bir rol oynamaktadır. Makine öğrenme algoritmaları ve yapay sinir ağları, üretim hatlarının optimize edilmesi, robotik sistemlerin kontrol edilmesi, tedarik zinciri yönetimi ve veri analizi gibi birçok otomasyon görevinde kullanılmaktadır. Bu sayede iş süreçlerinde verimlilik artmakta, hata oranları düşmektedir.

Şekil 7: *Fabrikalarda yapay zeka ile kontrol edilen robotik sistemler*

Sağlık Hizmetleri: Yapay zeka, sağlık sektöründe birçok alanda kullanılmaktadır. Tıbbi görüntüleme alanında, yapay zeka tabanlı algoritmalar, röntgen, MR ve CT taramaları gibi görüntüleme verilerini analiz ederek hastalıkların teşhis edilmesine yardımcı olmaktadır. Ayrıca, hastaların sağlık geçmişlerinden veri analizi yaparak risk faktörlerini tahmin etme ve kişiselleştirilmiş tedavi planları oluşturma gibi görevlerde de yapay zeka kullanılmaktadır.

Otomotiv: Yapay zeka, otonom araç teknolojilerinin temelini oluşturmaktadır. Yapay zeka algoritmaları, araçların çevresini algılamak için kullanılan sensör verilerini işleyerek araçların güvenli bir şekilde seyretmelerini sağlamaktadır. Ayrıca, trafik optimizasyonu, kaza önleme sistemleri ve sürücülerin tercihlerini anlama gibi alanlarda da yapay zeka kullanılmaktadır.



Şekil 8: *Otonom araçların çalışma prensibi*

Doğal Dil İşleme: Yapay zeka, doğal dil işleme (NLP) teknikleriyle metin tabanlı verileri analiz etme ve anlama yeteneğine sahiptir. Metin analizi, metin çevirisi, metin sınıflandırma, konuşma tanıma ve metin tabanlı soru-cevap sistemleri gibi birçok uygulama, yapay zeka algoritmalarıyla desteklenmektedir. Bu sayede, otomatik metin oluşturma, müşteri hizmetleri ve dil çevirisi gibi işlemler gerçekleştirilebilmektedir.

Finans: Yapay zeka, finansal hizmetler sektöründe büyük bir etkiye sahiptir. Yüksek hızda veri analizi yapabilme yeteneği sayesinde, yapay zeka algoritmaları risk analizi, dolandırıcılık tespiti, portföy yönetimi, otomatik ticaret ve müşteri hizmetleri gibi alanlarda kullanılmaktadır. Bu sayede finansal kararlar daha hızlı ve daha doğru bir şekilde alınabilmektedir.

Oyun Programlama: Yapay zeka, oyunlarda yapay oyuncuların davranışlarını kontrol etmek için kullanılmaktadır. Yapay zeka algoritmaları, yapay oyuncuların karar verme süreçlerini simüle ederek gerçekçi ve zorlu bir oyun deneyimi sağlamaktadır. Ayrıca, yapay zeka oyunlarda düşman karakterlerin davranışlarını yönetmek, yapay zeka destekli karakterlerin oyunculara rehberlik etmesi gibi görevlerde de kullanılmaktadır.

Bu alanlar, yapay zekanın çeşitli uygulamalarının sadece birkaç örneğini temsil etmektedir. Yapay zeka teknolojileri, birçok sektörde verimliliği artırmak, kararları iyileştirmek ve yeni inovasyonlar sağlamak için yaygın bir şekilde kullanılmaktadır. Yapay zeka teknolojileri, verimlilik, doğruluk ve otomatizasyon gibi avantajlarıyla birçok sektörde büyük bir etki yaratmaktadır.

2 OYUNLARDA YAPAY ZEKA

2.1 Oyunlarda Yapay Zeka Neden Kullanılır?

Oyunlar, yapay zekanın yaygın olarak kullanıldığı alanlardan biridir. Yapay zeka, oyunlarda gerçekçi ve zorlu bir oyun deneyimi sunmak, oyunculara karşı rekabetçi bir ortam sağlamak ve oyuncuların becerilerini test etmek için kullanılır. Oyunlarda yapay zekanın kullanımına dair bazı örnekler:

Gerçekçi Oyun Deneyimi: Yapay zeka, oyunculara gerçekçi bir oyun deneyimi sunmak için kullanılır. Oyuncular, yapay zeka kontrollü karakterlerle etkileşime geçerek daha dinamik ve akıllı bir oyun dünyasında yer alırlar. Bu, oyunun daha zorlu ve tatmin edici olmasını sağlar.

Rekabetçi Oyun Ortamı: Yapay zeka, oyunculara karşı rekabetçi bir oyun ortamı oluşturmak için kullanılır. Yapay zeka kontrollü düşmanlar, oyuncuların becerilerini test etmek ve meydan okumak amacıyla tasarlanır. Bu, oyuncuların daha fazla adrenalin yaşamalarını sağlar ve oyun deneyimini daha heyecanlı hale getirir.

Oyuncu Desteği: Yapay zeka, oyunculara rehberlik etmek ve yardımcı olmak için kullanılır. Özellikle büyük ve karmaşık oyun dünyalarında, yapay zeka karakterleri oyunculara ipuçları, yönlendirmeler ve görevler sunarak oyunun ilerlemesini kolaylaştırır. Bu, oyuncuların oyunu daha iyi anlamalarına ve daha keyifli bir deneyim yaşamalarına yardımcı olur.

Oyun Dünyasının Dinamikliği: Yapay zeka, oyun dünyasının dinamik bir şekilde değişmesini sağlar. Yapay zeka algoritmaları, oyuncuların kararlarına ve eylemlerine tepki vererek oyun dünyasını sürekli olarak değiştirir. Bu, oyunun tekrar oynanabilirliğini artırır ve her seferinde farklı bir deneyim sunar.

Yaratıcı Oyun İçeriği: Yapay zeka, oyun yapımcılarına yaratıcı bir şekilde oyun içeriği oluşturma imkanı sunar. Yapay zeka algoritmaları, otomatik olarak dünya haritaları, karakterler, görevler ve diyaloglar oluşturabilir. Bu, oyun yapım sürecini hızlandırır ve daha geniş bir oyun içeriği yaratma potansiyelini ortaya çıkarır.

İnsan Oyuncuların Yetersizliği: Oyunlarda yapay zeka kullanılmasının bir diğer nedeni, çok oyunculu oyunlarda yetersiz insan oyuncu sayısıdır. Yapay zeka kontrollü karakterler, oyuncuların eksikliğini tamamlar ve oyun dünyasını canlı tutar. Böylece, oyuncuların her zaman bir rekabet veya işbirliği ortağı bulmaları sağlanır.

Öngörülemezlik: Yapay zeka, oyunculara karşı öngörülemezlik sağlamak için kullanılır. Oyunlarda tekrarlanan veya önceden belirlenmiş bir oyun süreci, oyuncular için sıkıcı hale gelebilir. Yapay zeka kontrollü karakterler, algoritmalara dayalı kararlar alarak ve oyuncuların eylemlerine gerçek zamanlı olarak tepki vererek öngörülemezlik oluşturur. Bu, oyuncuların her seferinde farklı ve şaşırtıcı deneyimler yaşamalarını sağlar.

Yapay Oyuncu Davranışları: Oyunlarda yapay zeka, yapay oyuncuların davranışlarının kontrol edilmesinde kullanılır. Yapay oyuncular, oyunculara karşı rekabet eder, stratejik kararlar alır ve oyunun dinamiklerine uyum sağlar. Yapay zeka algoritmaları, yapay oyuncuların düşmanları takip etmeleri, siperlere saklanmaları, taktiksel manevralar yapmaları gibi görevleri yerine getirmelerini sağlar.

Adaptasyon Yeteneği: Yapay zeka, oyuncular için adaptasyon yeteneği sağlamak amacıyla kullanılır. Oyunlar genellikle farklı zorluk seviyelerine sahiptir ve oyuncuların beceri düzeyleri değişebilir. Yapay zeka kontrollü karakterler, oyuncuların beceri seviyelerine ve performanslarına uyum sağlayarak oyun deneyimini optimize eder. Örneğin, yapay zeka, oyuncuların başarısız olduğu alanlarda daha kolay bir oyun deneyimi sunabilir veya daha deneyimli oyunculara daha zorlu bir meydan okuma sunabilir. Bu şekilde, yapay zeka oyunculara özelleştirilmiş bir oyun deneyimi sunar ve oyuncuların kendilerini daha rahat hissetmelerini sağlar.

Oyun Sürekliği ve Uzun Ömürlülük: Yapay zeka, oyunlara sürekliği ve uzun ömürlülüğü sağlamak için kullanılır. Bir oyunun tekrarlanabilirlik değeri, oyuncuların oyunu tekrar tekrar oynamak istemesini etkileyen önemli bir faktördür. Yapay zeka kontrollü karakterler, her oyun seansında farklı davranışlar sergileyerek oyunun sürekliğini artırır. Oyuncular, her seferinde farklı bir deneyim yaşayarak oyunun tekrar tekrar oynanabilirliğini deneyimler.

Oyuncu Etkileşimi ve Duygusal Bağ: Yapay zeka, oyuncu etkileşimi ve duygusal bağ oluşturmak için kullanılır. Oyuncular, yapay zeka kontrollü karakterlerle etkileşimde bulunarak duygusal bir bağ kurabilir. Yapay zeka karakterleri, gerçekçi tepkiler ve duygusal ifadelerle oyuncuların duygusal tepkilerini uyandırabilir. Bu, oyuncuların karakterlere bağlanmalarını, empati kurmalarını ve hikayenin gelişimiyle ilgili hisler yaşamalarını sağlar.

2.2 Oyunlarda Yapay Zeka Kullanımının Tarihçesi

Oyunlarda yapay zeka kullanımı, oyun endüstrisinin gelişimiyle birlikte büyük bir evrim geçirmiştir. İlk dönem oyunlarında, yapay zeka daha basit formda ve sınırlı bir şekilde kullanılıyordu. Öncelikle düşman karakterlerin basit hareket modelleri ve sınırlı tepkileriyle simüle ediliyordu.

Ancak, oyun teknolojilerinin ilerlemesiyle birlikte yapay zeka, daha sofistike ve gerçekçi hale gelmeye başladı. 1980'lerdeki oyunlarda, yapay zeka düşman karakterlerin daha karmaşık davranışlar sergilemesini sağlayacak şekilde geliştirildi. Örneğin, düşmanlar oyuncunun hareketlerini takip edebilir, saldırı stratejilerini ayarlayabilir ve ortamı etkin bir şekilde kullanabilir hale geldi.

1990'lı yıllarla birlikte, yapay zeka algoritmaları daha da geliştirildi ve oyunlarda daha fazla kullanılmaya başlandı. Bu dönemde, strateji tabanlı oyunlar ve yapay zeka kontrollü karakterlerin bulunduğu rol yapma oyunları popüler hale geldi. Oyuncular, yapay zeka tarafından kontrol edilen karakterlerle etkileşime girebilir, onlara talimatlar verebilir ve takım çalışması yapabilirlerdi.

Yapay zeka kullanımı, 2000'li yıllarda ve sonrasında da hızla arttı. Gelişmiş algoritmalar, büyük veri analizi, makine öğrenimi ve derin öğrenme gibi teknolojik ilerlemeler, yapay zekanın oyunlarda daha gerçekçi ve akıllı bir şekilde kullanılmasını sağladı. Oyuncular artık yapay zeka kontrollü karakterlerle daha karmaşık etkileşimlerde bulunabiliyor, gerçek zamanlı strateji oyunlarında daha sofistike kararlar alabiliyor ve yapay zeka tabanlı rakiplerle rekabet edebiliyor.

Bugün, yapay zeka oyunlarda yaygın bir şekilde kullanılmaktadır ve oyun endüstrisindeki büyük bir öneme sahiptir. Yapay zeka, oyunların daha gerçekçi, zorlu, etkileyici ve dinamik bir deneyim sunmasını sağlar. Gelecekte, yapay zeka teknolojilerinin daha da ilerlemesiyle birlikte, oyunlardaki yapay zeka kullanımının daha da gelişmesi ve daha inovatif deneyimlerin ortaya çıkması beklenmektedir.

2.3 Oyunlarda Yapay Zeka Kullanımının Avantajları

Paralel İşlem Kapasitesi: Yapay zeka, genellikle karmaşık hesaplamalar ve büyük veri işleme gerektirdiğinden paralel işlem kapasitesi avantajı sunar. Gelişmiş yapay zeka algoritmalarının yüksek performanslı işlemcilerle çalışması, oyunlarda daha hızlı ve verimli yapay zeka hesaplamaları gerçekleştirmeyi mümkün kılar.

Özel Donanım Hızlandırma: Bazı yapay zeka algoritmaları, özel donanım hızlandırma teknolojileri kullanarak daha yüksek performans elde etmek için optimize edilebilir. Örneğin, grafik işlem birimleri (GPU'lar) veya tensor işlem birimleri (TPU'lar), yapay zeka hesaplamalarını hızlandırmak için özel olarak tasarlanmış donanımlar sağlar. Bu, oyunlarda daha karmaşık yapay zeka modellerinin daha hızlı ve verimli bir şekilde çalışmasını sağlar.

Gerçek Zamanlı Performans: Oyunlarda yapay zeka, gerçek zamanlı performans gerektiren bir alan olarak önemli bir rol oynar. Donanımın yüksek performansı, oyunlarda anlık tepkiler ve akıcı bir oyun deneyimi sunmayı mümkün kılar. Yüksek performanslı işlemciler, hızlı hesaplamalar ve yapay zeka algoritmalarının hızlı çalışması için gereken işlem gücünü sağlar.

Dağıtılmış Hesaplama: Yapay zeka algoritmaları, karmaşık hesaplamalar ve veri işleme gerektirebilir. Ancak, dağıtılmış hesaplama yöntemleri kullanılarak bu yük CPU üzerinde daha etkin bir şekilde dağıtılabilir. Örneğin, çoklu işlemci veya çok çekirdekli işlemciler kullanılarak yapay zeka hesaplamaları paralel olarak yürütülebilir, bu da CPU kullanımını optimize eder ve performansı artırır.

Bellek Yönetimi: Yapay zeka algoritmalarının çalışması için büyük veri setlerine ve model parametrelerine ihtiyaç duyulabilir. Ancak, gelişmiş bellek yönetimi teknikleri kullanılarak bu verilerin ve parametrelerin bellek üzerinde daha verimli bir şekilde yönetilmesi sağlanabilir. Bellek yönetimi optimizasyonları, RAM kullanımını azaltır ve daha verimli bir yapılandırma sağlar.

Ön İşleme ve Veri Sıkıştırma: Yapay zeka algoritmalarının veri işlemesi genellikle büyük veri setleri üzerinde yapılır. Ancak, önceden işlenmiş veya sıkıştırılmış veri formatları kullanılarak bu verilerin boyutu ve işleme gereksinimleri azaltılabilir. Veri ön işleme ve sıkıştırma teknikleri, CPU ve RAM kullanımını düşürerek oyun performansını artırır.

Yapay zeka temelli görüntü iyileştirme teknolojileri, oyunlarda grafikleri ve efektleri geliştirmek için kullanılan önemli araçlardan birkaçıdır. Bunlardan bazıları:

DLSS (Deep Learning Super Sampling): DLSS, yapay zeka ve derin öğrenme algoritmalarını kullanarak oyunlardaki grafikleri daha yüksek çözünürlükte ve daha keskin bir şekilde sunmayı hedefleyen bir görüntü iyileştirme tekniğidir. DLSS, oyun motoruna entegre edilmiş yapay zeka modellerini kullanarak düşük çözünürlüklü bir görüntüyü yüksek çözünürlüğe dönüştürerek oyunlarda daha iyi görüntü kalitesi sağlar. Bu teknoloji, yüksek performanslı oyun deneyimi sunarken GPU yükünü azaltarak daha akıcı bir oyun performansı sağlar.

FSR (FidelityFX Super Resolution): FSR, AMD tarafından geliştirilen bir görüntü iyileştirme teknolojisidir. Yapay zeka ve algoritma tabanlı bir yaklaşım kullanarak oyunların düşük çözünürlükteki görüntülerini yüksek çözünürlüğe yakın bir kaliteye yükseltir. FSR, GPU performansını artırırken düşük sistem gereksinimlerine sahip oyunlarda daha akıcı bir deneyim sunar.

Bu yapay zeka temelli görüntü iyileştirme teknolojileri, oyunlarda daha yüksek görüntü kalitesi ve performansını mümkün kılar. Yapay zeka algoritmaları, oyunların grafiklerini optimize ederken aynı zamanda donanım kaynaklarını daha verimli bir şekilde kullanır. Sonuç olarak, oyunlarda daha gerçekçi ve etkileyici görseller elde edilirken, oyunculara daha akıcı bir oyun deneyimi sunulur.

2.4 Oyunlarda Yapay Zeka Kullanımının Dezavantajları

Yüksek Geliştirme Maliyeti: Yapay zeka tabanlı oyunlar geliştirmek genellikle daha fazla zaman, kaynak ve uzmanlık gerektirir. Yapay zeka algoritmalarının tasarımı, eğitimi ve entegrasyonu karmaşık olabilir ve bunlar için deneyimli ekip üyeleri ve kaynaklar gerekebilir. Bu da geliştirme maliyetlerini artırabilir ve oyun stüdyoları için finansal bir zorluk oluşturabilir.

Performans ve Optimizasyon Sorunları: Yapay zeka tabanlı oyunlar, daha fazla hesaplama gücü ve bellek kullanımı gerektirebilir. Bu da oyunun performansını olumsuz etkileyebilir, özellikle düşük donanım özelliklerine sahip sistemlerde. Yapay zeka algoritmalarının optimize edilmemesi durumunda, oyunun kasma, takılma veya düşük kare hızları gibi sorunlarla karşılaşılabilir.

Yapaylık ve Tahmin Edilebilirlik: Yapay zeka tabanlı düşman veya NPC davranışları bazen oyuncular tarafından yapay veya tahmin edilebilir olarak algılanabilir. Yapay zeka algoritmalarının kısıtlamaları veya belirli desenlerin tekrar etmesi nedeniyle, oyunun zorluk seviyesi veya düşmanların taktikleri bazen öngörülebilir olabilir. Bu durum, oyunun tekrar oynanabilirliğini azaltabilir ve oyuncu deneyimini etkileyebilir.

Yapay Zeka Hataları: Yapay zeka algoritmaları, yanlış kararlar verebilir veya beklenmedik hatalar yapabilir. Bu hatalar, oyunun akışını veya dengeyi etkileyebilir ve oyuncuların gerçekçilik hissini bozabilir. Örneğin, yapay zeka karakterlerin duvarlara takılması veya mantıksız hareketler yapması gibi hatalar, oyuncuların oyun dünyasına olan bağlılığını zayıflatabilir.

Daha Az İnsan Etkileşimi: Yapay zeka tabanlı oyunlarda, bazen insan etkileşimi ve oyuncu-oyun etkileşimi azalabilir. Yapay zeka karakterlerin daha az sosyal etkileşim sunması veya daha az anlamlı hikaye ilişkileri oluşturması, oyun deneyimini daha soğuk veya yalıtılmış hale getirebilir.

Sınırlı Gerçeklik Algısı: Yapay zekanın gerçeklik algısı, insanların algıladığı kadar kapsamlı ve derin olmayabilir. Oyun dünyasındaki detayları, nesnelerin özelliklerini veya ortamın karmaşıklığını tam olarak anlamak ve yorumlamak konusunda bazı sınırlamaları olabilir. Bu da yapay zekanın bazen hatalı kararlar vermesine veya mantıksız davranışlar sergilemesine neden olabilir.

Öğrenme Süreci: Yapay zeka algoritmalarının eğitim süreci zaman alabilir ve kaynak gerektirebilir. Yapay zeka, bir oyunun içinde doğru kararları verebilmek için önceden eğitilmelidir. Bu eğitim süreci, oyunun gerektirdiği karmaşıklık ve değişkenlik göz önüne alındığında zaman ve kaynak açısından maliyetli olabilir.

İnsan Yeteneklerinin Yansıtılması: Yapay zeka, bazı durumlarda insan yeteneklerini tam olarak yansıtmakta zorluklar yaşayabilir. Örneğin, bir oyunun içinde karmaşık bir strateji oluşturmak veya yaratıcılık gerektiren bir görevi yerine getirmek gibi insan becerilerini tam anlamıyla taklit etmek zor olabilir. Bu nedenle, yapay zeka bazen sınırlı veya sınırlı bir şekilde insan yeteneklerini yansıtabilir.

Etik Sorunlar: Yapay zekanın kullanımıyla ilgili etik sorunlar ortaya çıkabilir. Örneğin, yapay zeka tabanlı oyunlarda karakterlerin duygusal tepkileri veya ahlaki değerleri doğru bir şekilde yansıtmak zor olabilir. Ayrıca, yapay zeka algoritmalarının yanlış kullanılması veya kötü niyetli bir şekilde manipüle edilmesi durumunda, oyunlarda adaletsizlik veya haksız avantajlar ortaya çıkabilir.

Bu sınırlamalar, yapay zekanın oyunlarda kullanımının bazı zorlukları olduğunu gösterir. Ancak, ilerleyen teknoloji ve araştırmalarla birlikte bu sınırlamaların aşılabileceği veya en azından minimize edilebileceği düşünülmektedir.

3 OYUNLARDAKİ YAPAY ZEKA TEKNİKLERİ

3.1 Oyunlarda Genellikle Kullanılan Yapay Zeka Teknikleri

Oyunlarda yapay zeka kullanımı için çeşitli teknikler ve yöntemler bulunmaktadır. Bu tekniklerden en çok kullanılanlardan bazıları şunlardır:

Makine Öğrenmesi: Makine öğrenmesi, yapay zekanın deneyimlerden öğrenerek davranışlarını geliştirmesini sağlayan bir tekniktir. Oyunlarda, makine öğrenmesi algoritmaları kullanılarak yapay zeka, oyuncuların hareketlerini, stratejilerini ve oyun dünyasındaki etkileşimleri analiz ederek kendini geliştirir.

Pekiştirmeli Öğrenme: Pekiştirmeli öğrenme, yapay zekanın bir çevreyle etkileşim içinde olduğu durumları modellemek için kullanılan bir tekniktir. Oyunlarda, yapay zeka, belirli bir hedefe ulaşmak için doğru veya yanlış hareketlerle çevreyle etkileşimde bulunur ve bu deneyimlerden öğrenerek daha iyi stratejiler geliştirir.

Karar Ağaçları: Karar ağaçları, yapay zekanın bir dizi karar noktasından geçerek sonuca ulaşmasını sağlayan bir yapıdır. Oyunlarda, yapay zeka karakterlerinin karar verme sürecini modellemek için kullanılır. Karar ağaçları, oyuncuların eylemlerini analiz ederek en uygun kararı veren bir dizi karar noktası oluşturur.

Yapay Sinir Ağları: Yapay sinir ağları, biyolojik sinir ağlarının matematiksel modelleridir. Bu teknik, yapay zekanın öğrenme ve karar verme süreçlerini modellemek için kullanılır. Oyunlarda yapay sinir ağları, karmaşık desenleri tanımak, oyuncuların davranışlarını tahmin etmek ve uygun yanıtlar vermek gibi görevlerde kullanılır.

Bulanık Mantık: Bulanık mantık, karmaşık veya belirsiz durumları işlemek için kullanılan bir yöntemdir. Oyunlarda bulanık mantık, yapay zekanın oyunculara daha doğal ve gerçekçi tepkiler verebilmesini sağlar. Örneğin, yapay zeka karakterleri, oyuncuların hareketlerini ve durumlarını analiz ederek uygun tepkileri vermek için bulanık mantık sistemlerini kullanabilir.

3.2 Yapay Zeka Tekniklerinin Detayları

3.2.1 Makine Öğrenmesi

Makine öğrenmesi, yapay zeka alanında kullanılan bir yöntemdir ve bir bilgisayarın deneyimlerden öğrenme yoluyla kendini geliştirmesini sağlar. Makine öğrenmesi, veri analizi ve istatistiksel modellerin kullanımıyla bilgisayarlara öğrenme yeteneği kazandırır.

Makine öğrenmesi süreci genellikle şu adımları içerir:

Veri Toplama: Makine öğrenmesi algoritması için kullanılacak verilerin toplanması gerekmektedir. Bu veriler, oyunlarda oyuncuların hareketlerini, stratejilerini, tercihlerini veya oyun dünyasındaki etkileşimleri içerebilir. Bu veriler genellikle oyuncu davranışlarına veya oyunun içeriğine dayanan büyük veri kümeleridir.

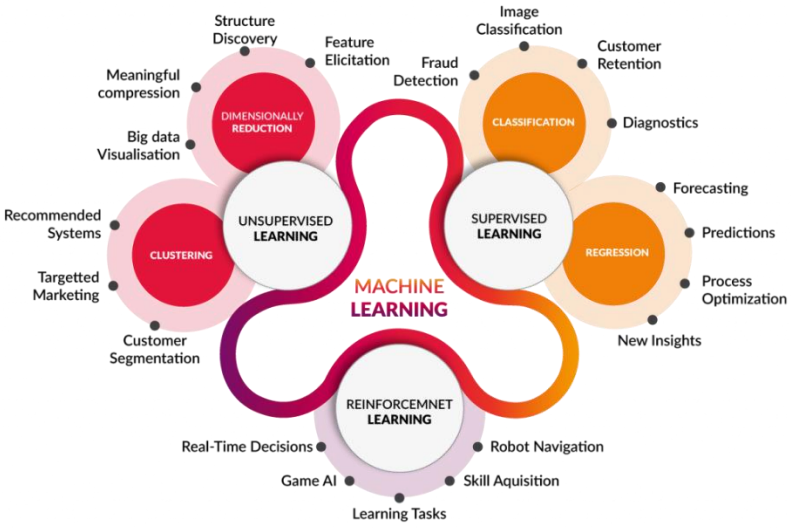
Veri Ön İşleme: Toplanan veriler, makine öğrenmesi algoritmalarının işleyebileceği uygun bir formata getirilir. Bu adımda veriler filtrelenir, düzenlenir, eksik değerler doldurulur veya normalleştirilir. Veri ön işleme aşaması, verilerin daha doğru ve tutarlı sonuçlar üretmek için hazır hale getirilmesini sağlar.

Model Seçimi ve Eğitimi: Makine öğrenmesi modeli, veri üzerinde eğitilerek öğrenme sürecini gerçekleştirir. Bu aşamada, uygun bir makine öğrenmesi algoritması seçilir ve veri kümesi üzerinde bu model eğitilir. Eğitim süreci, veriye dayanan örüntüleri ve ilişkileri tanımak ve modelin performansını iyileştirmek için gerçekleştirilen iteratif bir süreçtir.

Model Değerlendirme ve Ayarlama: Eğitilen model, doğruluk ve performans kriterleri kullanılarak değerlendirilir. Modelin tahmin yeteneği, hata oranı ve diğer performans metrikleri ölçülür. Modelin başarı düzeyine bağlı olarak, ayarlamalar yapılabilir veya farklı modeller denenerek daha iyi sonuçlar elde edilmeye çalışılır.

Modelin Uygulanması: Eğitilen model, oyunlarda yapay zeka uygulamalarında kullanılmak üzere gerçek zamanlı olarak uygulanır. Bu aşamada, oyuncuların gerçek zamanlı etkileşimleri veya oyun ortamının değişkenleriyle modelin etkileşimi sağlanır. Modelin çıktıları, oyunculara karşı akıllı bir davranış veya gerçekçi bir tepki olarak yansıtılır.

Makine öğrenmesi, oyunlarda yapay zekayı geliştirmek ve daha zengin, gerçekçi deneyimler sunmak için önemli bir araçtır. Yapay zeka, oyuncuların hareketlerini analiz ederek stratejik kararlar alabilir, oyuncu tercihlerini tahmin edebilir veya oyun dünyasını dinamik ve öngörülemez hale getirebilir. Makine öğrenmesi algoritmaları, bu süreçte verilerden öğrenir ve zamanla daha iyi sonuçlar üretmek için kendini geliştirir.



Şekil 9: *Makine öğrenmesi ile ilgili görsel*

3.2.2 Pekiştirmeli Öğrenme

Pekiştirmeli öğrenme, yapay zeka alanında kullanılan bir öğrenme yaklaşımıdır ve bir bilgisayarın çevreyle etkileşim içinde olduğu durumları modellemesini sağlar. Pekiştirmeli öğrenme, bir ajanın belirli bir hedefe ulaşmak için deneyimlerden öğrenme yoluyla en iyi eylemi seçmesini amaçlar.

Pekiştirmeli öğrenme süreci genellikle şu adımları içerir:

Durumları ve Eylemleri Tanımlama: Pekiştirmeli öğrenme probleminde, bir ajanın bulunduğu çevre durumu ve yapabileceği eylemler tanımlanır. Oyunlarda, çevre durumu genellikle oyun dünyasının mevcut durumunu, oyunun ilerlemesini ve diğer oyuncuların durumunu içerir. Eylemler ise ajanın oyun içinde yapabileceği hamleleri, stratejileri veya kararları temsil eder.

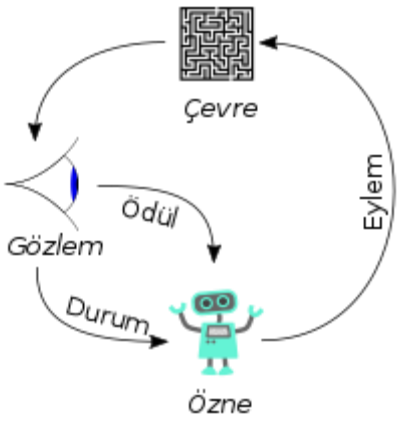
Ödül ve Cezaların Tanımlanması: Pekiştirmeli öğrenme, bir ajanın hedefe ulaşmak için aldığı eylemlere bağlı olarak ödül veya ceza verme prensibine dayanır. Oyunlarda, ödüller genellikle puanlar, seviye atlama, başarılar veya diğer oyun içi avantajlar şeklinde temsil edilir. Cezalar ise istenmeyen durumlar veya başarısızlıklar sonucunda ajanın karşılaştığı negatif sonuçları ifade eder.

Deneyimlerin Toplanması: Pekiştirmeli öğrenme, ajanın çevreyle etkileşime geçerek deneyimler topladığı bir süreçtir. Oyunlarda, ajanın oyun dünyasıyla etkileşimi, oyuncu hareketleri ve stratejileri üzerinden gerçekleşir. Ajan, çevresel durumu gözlemleyerek bir eylem seçer, bu eylemi gerçekleştirir ve çevrenin tepkisini alır.

Değer Fonksiyonu veya Q-Fonksiyonunun Güncellenmesi: Pekiştirmeli öğrenmede, ajanın aldığı eylemlerin değeri veya kalitesi bir değer fonksiyonu veya Q-fonksiyonu kullanılarak değerlendirilir. Bu fonksiyon, ajanın her durumda alabileceği eylemlerin değerlerini tahmin eder ve gelecekteki ödüllerle ilişkilendirir. Deneyimlerin toplanması ve çevrenin tepkileri üzerine yapılan gözlemlemelerle, değer fonksiyonu veya Q-fonksiyonu güncellenir ve iyileştirilir.

Eylem Seçimi ve İyileştirme: Ajan, güncellenmiş değer fonksiyonu veya Q-fonksiyonunu kullanarak en iyi eylemi seçer. Bu süreç, ajanın hedefe ulaşmak için en uygun kararları almasını sağlar. Eylem seçimi genellikle keşfetme ve sömürme stratejileriyle dengelemeye çalışılır. Ajan, keşfetme yoluyla yeni eylem seçeneklerini keşfederken, öğrenilen bilgiyi de kullanarak en iyi eylemleri seçmeye çalışır.

Pekiştirmeli öğrenme, oyunlarda yapay zeka geliştirmek için güçlü bir araçtır. Oyunlarda ajanların stratejilerini geliştirmek, akıllı düşmanları simüle etmek veya oyun dünyasını dinamik ve öngörülemez hale getirmek için pekiştirmeli öğrenme teknikleri kullanılabilir.



Şekil 10: *Pekiştirmeli Öğrenme ile ilgili görsel*

3.2.3 Karar Ağaçları

Karar ağaçları, makine öğrenmesi ve yapay zeka alanında sıkça kullanılan bir sınıflandırma ve regresyon yöntemidir. Bir karar ağacı, bir dizi karar kuralı ve karar düğümüyle temsil edilen bir yapıdır. Her düğüm, bir özelliği temsil eder ve bir karar kuralıyla bağlantılıdır.

Karar ağaçları, veri setlerini kullanarak kararlar almak ve tahminler yapmak için kullanılır. Temel prensip, bir veri örneğinin özelliklerine dayanarak sınıflandırma veya regresyon işlemi yapmaktır. Karar ağacı, veri setindeki özelliklerin değerlerine göre bir dizi karar kuralı kullanarak sınıflandırma veya regresyon sonuçlarını üretir.

Karar ağacının temel bileşenleri şunlardır:

Kök Düğüm: Karar ağacının başlangıç noktasıdır. Veri setindeki en önemli özelliği temsil eder.

Dallanma Düğümleri: Kök düğümden gelen yolları temsil eder. Bir dallanma düğümü, bir özelliği ve o özellik için alınabilecek farklı değerleri temsil eder.

Yaprak Düğümleri: Karar ağacının sonuçlarını temsil eder. Her yaprak düğümü, bir sınıf etiketi veya regresyon değeriyle ilişkilendirilir.

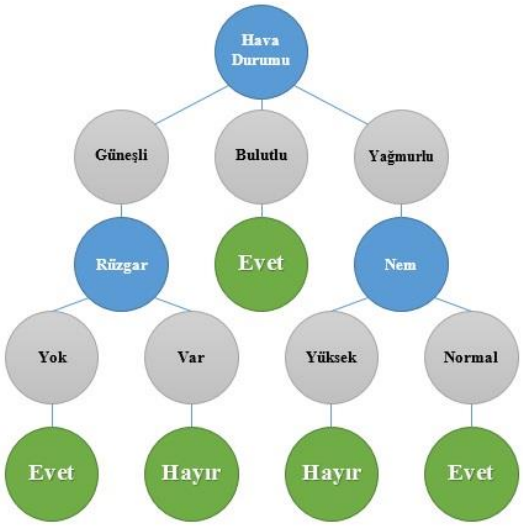
Karar ağaçlarının oluşturulması için genellikle aşağıdaki yöntemler kullanılır:

Karar Kuralı Oluşturma: Veri setindeki özelliklerin değerleri dikkate alınarak karar kuralı oluşturulur. Örneğin, "Yaş" özelliği için 30'dan büyükse ve "Gelir" özelliği için 50.000'den fazlaysa, bir karar kuralı oluşturulabilir.

Karar Kuralı Oluşturma: Veri setindeki özelliklerin değerleri dikkate alınarak karar kuralı oluşturulur. Örneğin, "Yaş" özelliği için 30'dan büyükse ve "Gelir" özelliği için 50.000'den fazlaysa, bir karar kuralı oluşturulabilir.

Karar Ağacı Oluşturma: Veri seti ve oluşturulan karar kuralları kullanılarak karar ağacı oluşturulur. Bu süreç, veri setinin ayrıntılarına ve özelliklerine göre ilerler. Her düğümde, en iyi karar kuralı ve değer seçilerek ağaç genişler.

Karar ağaçları, sınıflandırma ve regresyon problemlerinde kullanılan basit ve etkili bir yöntemdir. Veri setinin yapısal özelliklerini ve ilişkilerini anlamak için görsel ve anlaşılması kolay bir yapı sunar. Ayrıca, karar ağaçları genellikle hızlı bir şekilde eğitilir ve tahmin yapar.



Şekil 11: *Karar ağaçları ile ilgili görsel*

3.2.4 Yapay Sinir Ağları

Yapay sinir ağları (YSA), yapay zeka ve makine öğrenmesi alanında kullanılan güçlü bir modelleme ve öğrenme yaklaşımıdır. İnsan beyninin çalışma prensiplerinden esinlenen YSA'lar, karmaşık veri yapılarını analiz etmek, desenleri tanımak ve örüntüleri tahmin etmek için kullanılır.

YSA'lar, temelde birbirine bağlı yapay sinir hücrelerinden (nöronlar) oluşur. Bu yapay nöronlar, girdi değerlerini alır, bu değerleri bir aktivasyon fonksiyonu kullanarak işler ve çıktı değerlerini üretir. Bu çıktılar, bir sonraki katmandaki nöronlara giriş olarak iletilir ve işlem ağ üzerinde tekrarlanır. Bu şekilde, YSA, veri setlerindeki karmaşık ilişkileri modellemek için çok katmanlı bir yapı oluşturur.

YSA'ların ana bileşenleri şunlardır:

Girdi Katmanı: YSA'nın başlangıç noktasıdır ve veri setinden girdi değerlerini alır. Her girdi, bir yapay nöron tarafından temsil edilir.

Gizli Katmanlar: Bir veya daha fazla gizli katman, YSA'nın karmaşıklığını artıran katmanlardır. Her gizli katmanda, bir dizi yapay nöron bulunur. Bu katmanlar, verilerdeki örüntüleri ve ilişkileri yakalamak için kullanılır.

Çıktı Katmanı: YSA'nın son katmanıdır ve genellikle sınıflandırma veya regresyon sonuçlarını temsil eder. Her çıktı, bir yapay nöron tarafından temsil edilir ve tahmin veya sınıflandırma sonuçlarını üretir.

YSA'ların eğitimi ve çalışması için genellikle aşağıdaki adımlar izlenir:

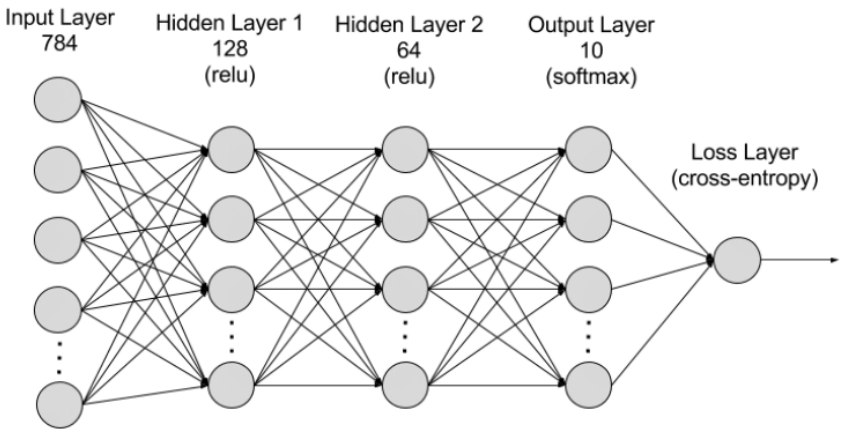
Ağırlık Atama: YSA'nın başarılı bir şekilde çalışması için nöronlar arasındaki bağlantıların ağırlıklarının başlangıçta rastgele atanması gerekir.

İleri Yayılım (İleri Hesaplama): Veri setindeki örnekler, girdi katmanından başlayarak ağın içinden ilerler ve çıktı katmanında tahmin sonuçları üretilir. Bu işlem, yapay nöronlardaki ağırlıkların kullanılmasıyla gerçekleşir.

Geri Yayılım (Geri Hesaplama): Yapay sinir ağının ürettiği çıktı sonuçları, gerçek sonuçlarla karşılaştırılır ve hata değerleri hesaplanır. Bu hata değerleri, ağın geriye doğru ilerleyerek ağırlıklarını güncellemesini sağlar.

Ağırlık Güncelleme: Geri yayılım algoritması kullanılarak elde edilen hata değerleri, ağırlıkların güncellenmesinde kullanılır. Bu güncelleme süreci, hata değerlerini azaltmaya ve ağın performansını iyileştirmeye yöneliktir.

YSA'lar, oyun programlamasında yapay zeka geliştirmek için de yaygın olarak kullanılır. Örneğin, yapay sinir ağları, oyunlarda düşman karakterlerin davranışlarını simüle etmek, oyun stratejilerini öğrenmek ve gerçekçi oyun deneyimleri sunmak için kullanılabilir.



Şekil 12: *Yapay sinir ağları ile ilgili görsel*

3.2.5 Bulanık Mantık

Bulanık mantık, belirsizlik içeren problemlerin modellemesi ve çözümü için kullanılan bir matematiksel yöntemdir. Bu yöntemde, kesin sınıflandırmalar yerine bulanık kümeler ve bulanık kural tabanları kullanılır.

Bulanık mantık, doğal dilde ifade edilen belirsiz kavramları işlemek için idealdir. Örneğin, "sıcak" veya "soğuk" gibi kavramlar belirsiz ve nesnel olmayan değerlere sahiptir. Bulanık mantık, bu tür belirsiz kavramları işleyebilir ve analiz edebilir.

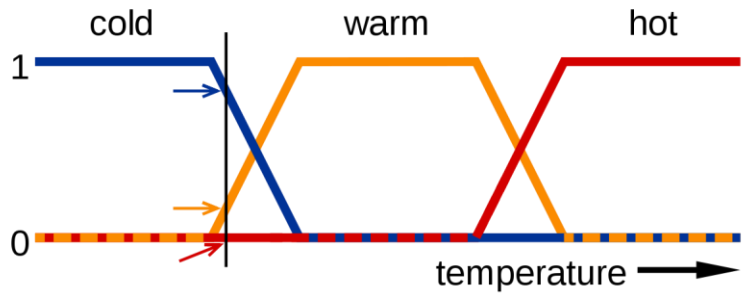
Bulanık mantığın anahtar bileşenleri şunlardır:

Bulanık Kümeler: Bulanık kümeler, belirsizlik içeren verileri temsil etmek için kullanılan matematiksel kavramlardır. Bir bulanık küme, belirli bir özellik veya değişken için üyelik fonksiyonu ile tanımlanır. Bu fonksiyon, bir değerin o kümeye ne kadar uyduğunu belirler. Örneğin, "sıcaklık" özelliği için "sıcak" bulanık kümesi, 0 ile 1 arasında bir üyelik değeriyle temsil edilebilir.

Bulanık Kurallar: Bulanık kurallar, belirli koşullar altında belirli eylemleri belirlemek için kullanılan ifadelerdir. Bu kurallar, "Eğer X, o zaman Y" şeklinde ifade edilir. Her bir kural, bir veya daha fazla girdi değişkenine ve bir çıktı değişkenine sahiptir. Örneğin, "Eğer sıcaklık yüksekse ve nem düşükse, o zaman klimayı aç" gibi bir bulanık kural olabilir.

Bulanık Mantık Operasyonları: Bulanık mantık, bulanık kümeler üzerinde işlem yapmak için belirli operasyonları kullanır. Bunlar arasında birleşme (union), kesişme (intersection), komplement (tamamlayıcı) gibi işlemler bulunur. Bu operasyonlar, bulanık kümelerin birlikte işlenmesi ve sonuçların elde edilmesi için kullanılır.

Bulanık mantık, oyunlarda yapay zeka ve karar verme sistemlerinde sıklıkla kullanılır. Özellikle, oyun karakterlerinin davranışlarını simüle etmek, kararlar almak ve oyun stratejilerini geliştirmek için bulanık mantık kullanılabilir. Bu sayede, belirsizlik ve değişkenlik içeren oyun ortamlarında daha gerçekçi ve akıllı hareket eden karakterler oluşturulabilir.

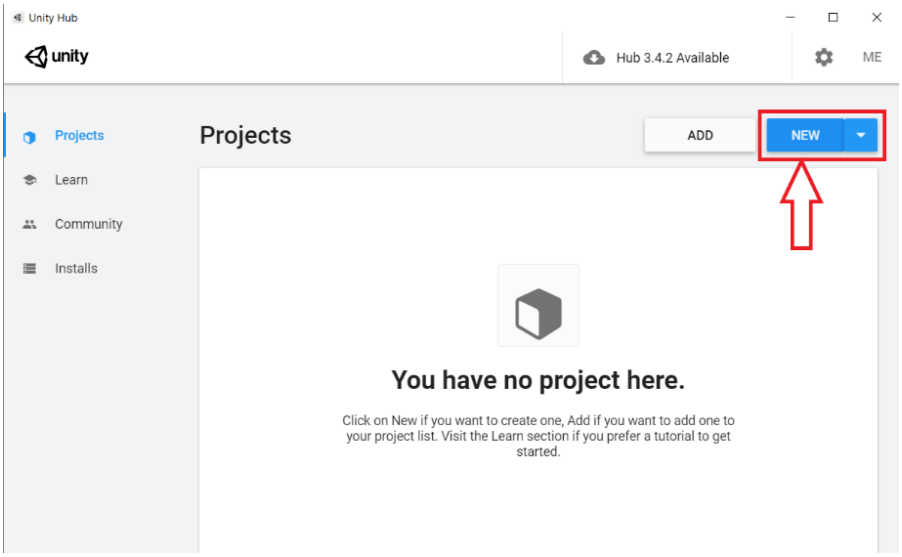


Şekil 13: *Bulanık mantık ile ilgili görsel*

4 UYGULAMA

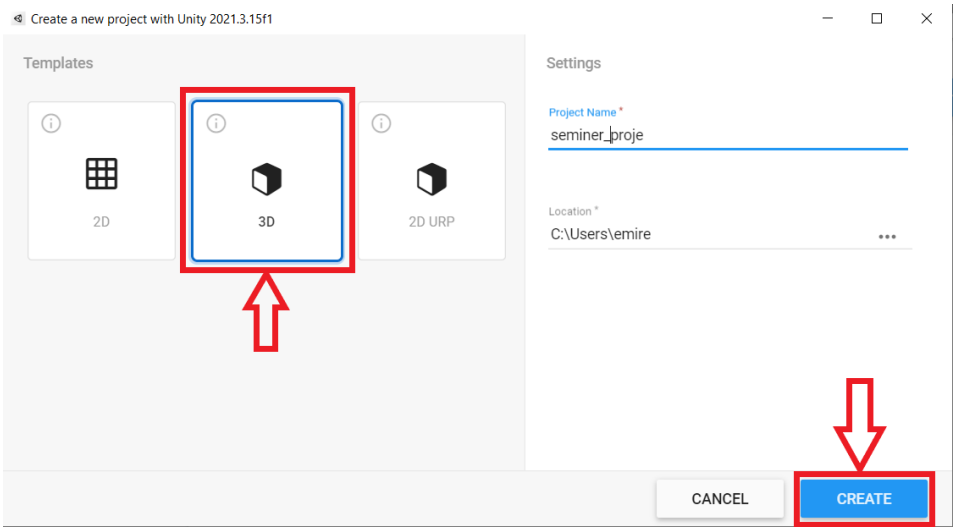
4.1 Proje Oluşturulması

Bu uygulama, "2021.3.15f1" sürümüne sahip Unity editörü kullanılarak geliştirilmiştir. Proje oluşturmak için öncelikle Unity Hub üzerinden "Projeler" sekmesinden yeni bir proje açmanız gerekmektedir. Unity Hub, Unity projelerinizi yönetmek ve farklı Unity sürümlerini yönetmek için kullanılan bir araçtır. Yeni proje oluşturmak için Unity Hub'a giriş yaparak "Yeni Proje Oluştur" seçeneğini seçmelisiniz. Ardından projenizin adını ve konumunu belirleyerek Unity editöründe çalışmaya başlayabilirsiniz. Bu şekilde, Unity Hub'un projeleri yönetme ve Unity editörüne erişim sağlama kolaylığından faydalanarak projenizi oluşturabilirsiniz.

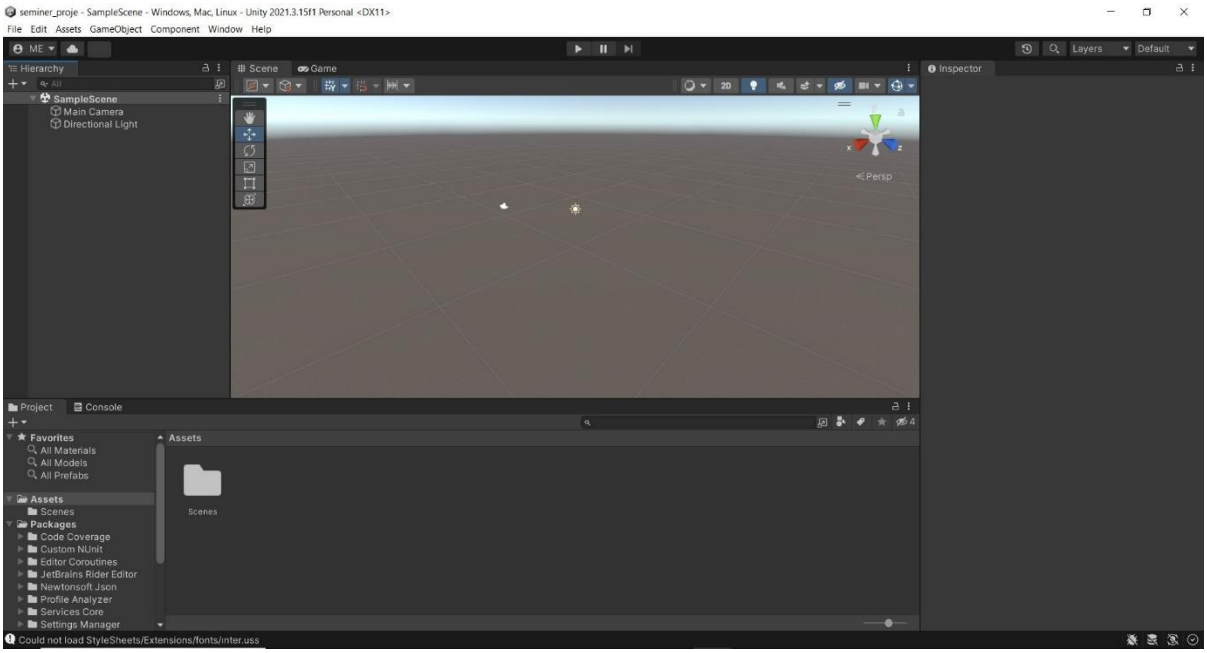


Şekil 14: *Unity Hub üzerinden yeni proje oluşturma seçimi*

Yeni bir proje açmak için seçim yapıldıktan sonra açılan pencerede, projenin hangi şablona (template) göre oluşturulacağı seçilmelidir. Bu uygulama 3 boyutlu (3D) bir oyun projesi olduğu için "3D" seçeneği tercih edilmelidir. Ayarlar bölümünde proje adı ve dosyaların konumu istenirse değiştirilebilir. Bu adımlar tamamlandıktan sonra "Create" butonuna basılmalıdır. Bu şekilde, seçilen şablona göre yeni bir 3D proje oluşturulmuş olur. "Create" butonuna tıklandıktan sonra Unity editörü proje dosyalarını oluşturacak ve projenin ana ekranı açılacaktır.



Şekil 15: *Açılan pencerede seçim işlemi*



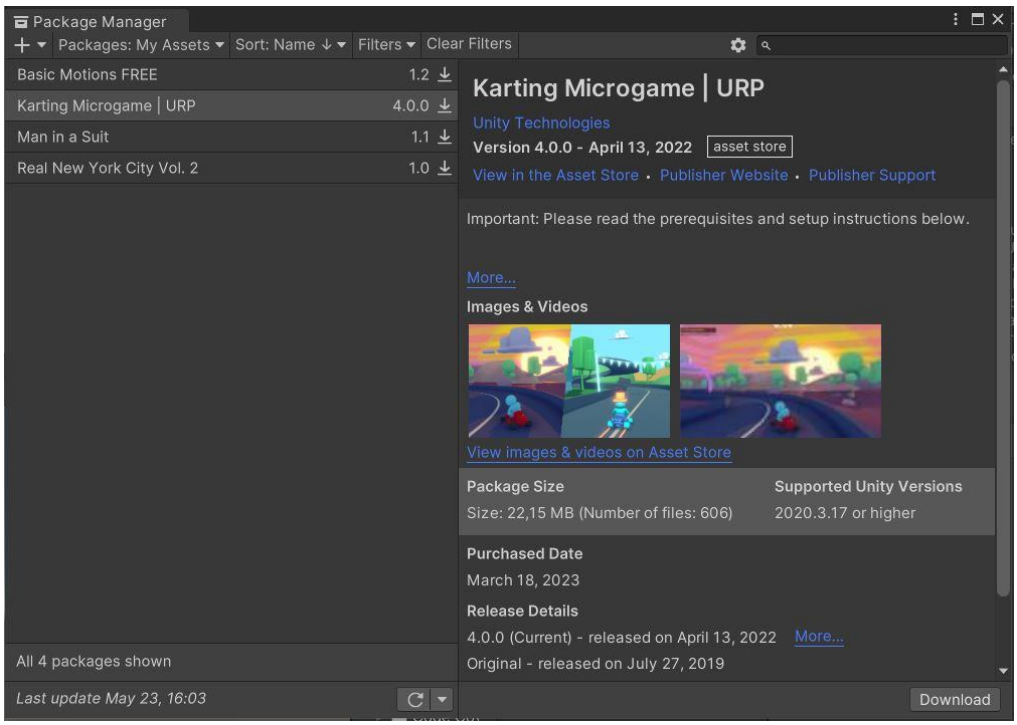
Şekil 16: *Unity Editor kullanıcı arayüzü*

"Hierarchy" kısmı, sahnelerin bulunduğu ve bu sahnelerin içerisine gerekli nesnelerin eklenebildiği bir bölümdür. İlk açıldığı halde, ana kamera ve ışık gibi temel nesneler bulunmaktadır. "Inspector" kısmı ise bu nesnelerin özelliklerinin değiştirilebildiği ve detaylı ayarların yapılabildiği bir bölümdür. Örneğin, bir nesnenin konumunu x, y, z koordinat sisteminde belirli bir konuma taşımak isterseniz, gerekli sayıları girerek konumunu değiştirebilirsiniz. Sağ üst köşedeki "Default" bölümünden ise arayüz görünümü değiştirilebilir. Uygulama sırasında "2 by 3" arayüz düzeni kullanılacaktır, bu düzen sayesinde çalışma alanınızı daha verimli bir şekilde kullanabilirsiniz.

4.2 Proje Uygulaması

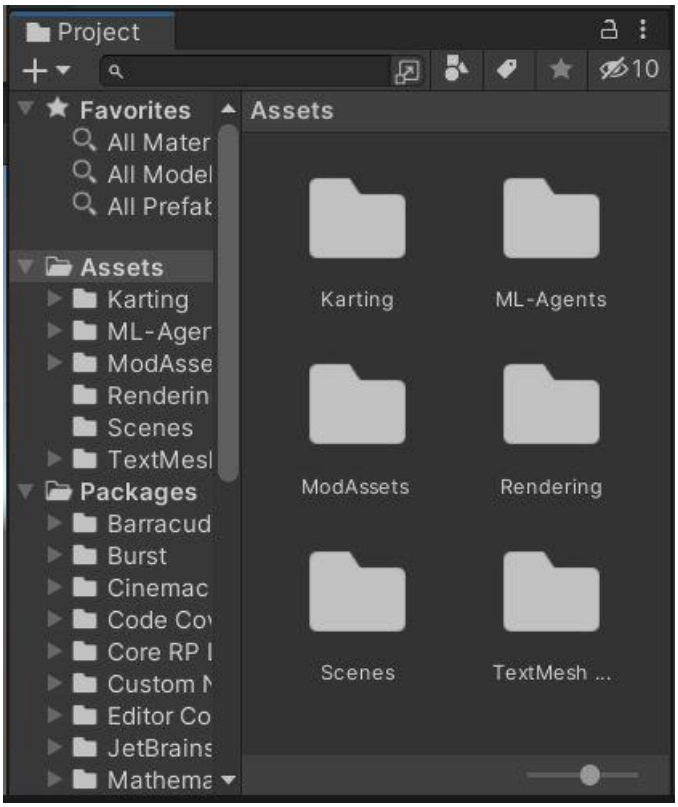
Bu uygulamada, sıfırdan proje oluşturma yerine hazır assetler kullanılarak yapay zeka eklemesi üzerinde durulacaktır. Hazır assetler, önceden oluşturulmuş ve kullanıma hazır olan oyun nesneleri, karakterler, animasyonlar veya diğer öğelerdir. Bu kılavuzda, yapay zeka eklemesi için kullanılacak hazır assetlerin seçimi, entegrasyonu ve yapılandırması adım adım açıklanacaktır. Böylece, projenize yapay zeka eklemek için başlangıçtan itibaren bir proje oluşturma sürecine gerek kalmadan hızlıca ilerleyebilirsiniz. Hazır assetler, projenize yapay zeka özellikleri kazandırmanızı kolaylaştırarak zaman ve emek tasarrufu sağlar.

Arayüzdeki üst menü çubuğunda bulunan "Window" sekmesi altında "Package Manager" kısmı bulunmaktadır. Bu bölümde, kullanıcının önceden eklediği hazır paketler görüntülenebilir. Bu uygulamada Unity Learning tarafından hazırlanan Karting Microgame paketi kullanılacaktır. Bu paketi indirdikten sonra, projeye eklemek için "Import" butonuna tıklamanız gerekmektedir. Bu şekilde, Karting Microgame paketi projenize başarıyla eklenmiş olacaktır. Bu paket, oyununuzda kullanmak için gerekli olan hazır nesneler, sahneler, özellikler ve diğer kaynakları içermektedir.



Şekil 17: *Package Manager yardımıyla hazır paketlerin eklenmesi*

İndirilen paket, projeye başarıyla eklendikten sonra "Assets" kısmında gerekli assetlerin geldiği görülecektir. "Assets" kısmı, projenizin içerisinde bulunan tüm kaynak dosyalarını ve hazır assetleri içeren bir bölümdür. İndirilen paket, bu kısımda ilgili klasör altında yer alacak ve içerisinde gerekli assetler bulunacaktır. Bu assetler, oyununuzda kullanacağınız karakterler, nesneler, animasyonlar, sesler veya diğer öğeler olabilir. Assets kısmında doğru klasörü bulduktan sonra, eklenen paketin içeriğini gözlemleyebilir ve projenizde kullanmaya başlayabilirsiniz.

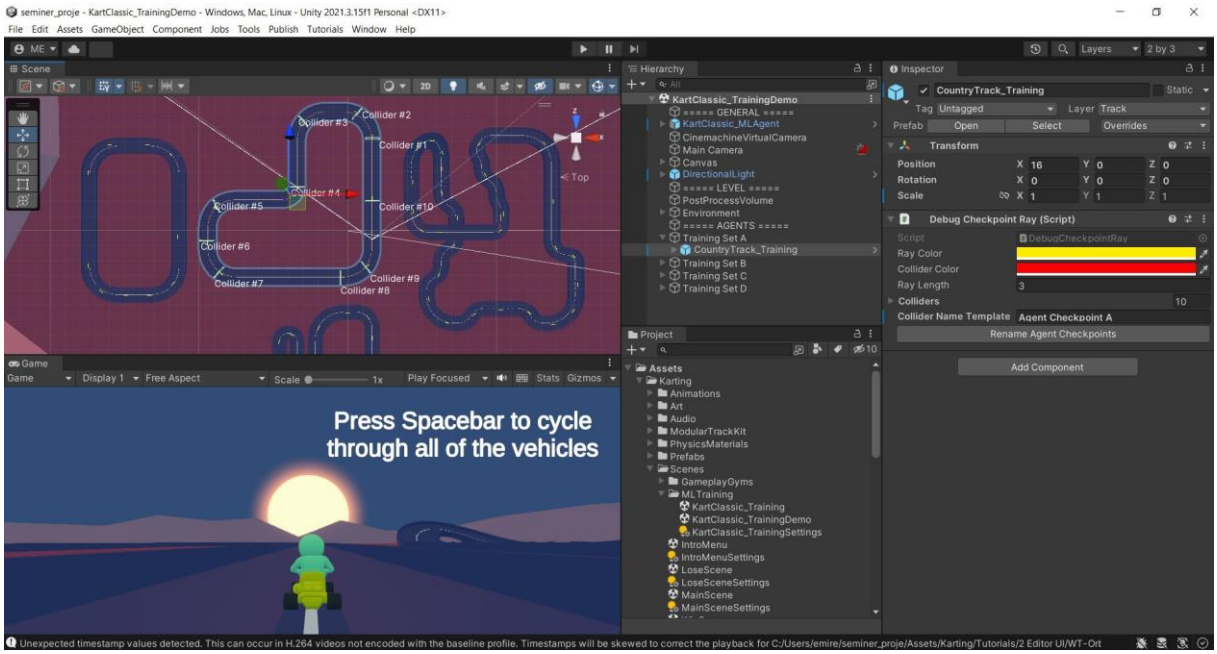


Şekil 18: *Assets klasöründeki Karting Microgame paketinin gerekli dosyaları*

Uygulamada, ajanların eğitimi sırasında yarışılacak harita yerine farklı haritaların kullanılması tercih edilmektedir. Bunun sebebi, ajanların aşırı öğrenme veya ezberleme durumuna düşmesini engellemektir. Farklı haritaların kullanılması, ajanların çeşitli ortamlarda farklı koşullarla karşılaşmasını sağlar ve daha genel bir eğitim elde etmelerini sağlar.

Bu projede, eğitim için özel olarak oluşturulmuş farklı haritalar bulunmaktadır. Bu haritalara ajanlar eklenir ve eğitim süreci gerçekleştirilir. Her harita farklı bir zorluk seviyesi veya yapıya sahip olabilir, böylece ajanlar farklı senaryolarla karşılaşarak çeşitli yetenekler geliştirebilir.

Farklı haritaların kullanılması, ajanların genel bir beceri kazanmasını sağlar ve uygulamada daha iyi bir performans elde edilmesine yardımcı olur. Aynı zamanda, gerçek dünyadaki değişkenlikleri ve farklı koşulları simüle etmek için bu yaklaşım önemlidir.



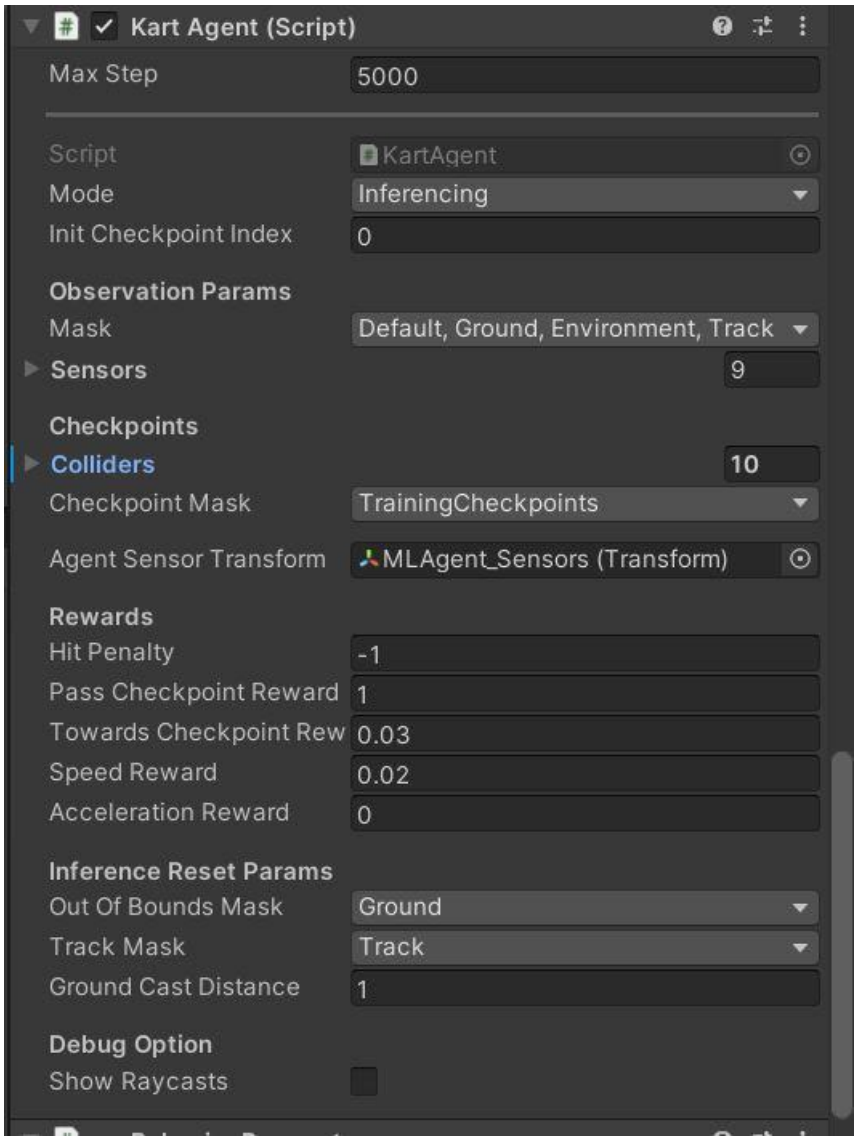
Şekil 19: *Training kısmına dair ekran görüntüsü*

Yukarıdaki ekran görüntüsünde gösterilen haritalarda, pekiştirmeli öğrenme mantığıyla çalışan bir eğitim sistemi kullanılmaktadır. Bu sistemin temel amacı, ajanın belirlenmiş checkpointlere ulaşarak belirli bir görevi yerine getirmesini sağlamaktır.

Harita üzerinde yer alan checkpointler, colliderlar aracılığıyla oluşturulmuştur. Ajan, checkpointlere ulaştıkça ödüller kazanır. Bu ödüller, ajanın doğru davranışları takip etmesini teşvik etmek amacıyla verilir. Örneğin, checkpointlere hızlı bir şekilde ulaşmak veya belirli bir yolu takip etmek gibi. Bu durumda ajan, hedefe yönelik doğru davranışları sergiledikçe daha fazla ödül kazanır.

Ancak ajan, yoldan çıkarak kenarlara çarptığında cezalandırılır. Bu çarpmalar, ajanın hatalı veya istenmeyen davranışlarına karşılık gelir. Ajanın hızlı ve hatasız bir şekilde checkpointlere ilerlemesi, kazandığı ödüllerin katlanmasını sağlar. Böylece, ajanın arzu edilen davranışları öğrenerek performansını geliştirmesi hedeflenir.

Bu eğitim sistemi, ajanın çeşitli durumlarda belirlenmiş hedeflere ulaşmasını sağlayarak yapay zekanın oyun içerisindeki davranışlarını geliştirmeye yönelik bir yaklaşımı temsil eder. Pekiştirmeli öğrenme prensipleri ve checkpoint tabanlı ödüllendirme sistemi kullanılarak, ajanın oyun içerisinde öngörülen görevleri yerine getirmesi ve başarılı bir şekilde ilerlemesi hedeflenir.



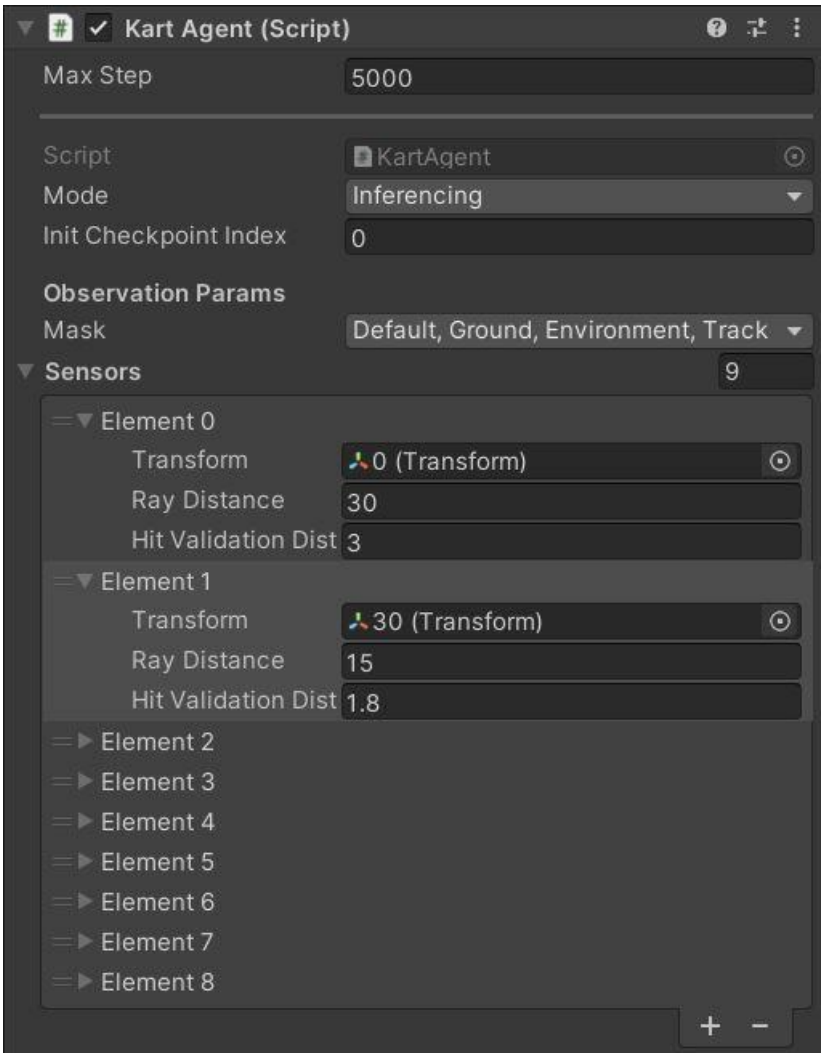
Şekil 20: *Inspector penceresinde Kart Agent Script'i altında Unity arayüzünden rewardları görüntüleme veya değiştirme*

Ajanların yol boyunca etrafını algılayabilmesi için Unity'de bulunan Raycast sistemi kullanılmaktadır. Raycast, bir noktadan başka bir noktaya doğru bir ışın çizerek çizginin üzerindeki nesneleri algılamamızı sağlar. Bu projede ise Raycast, ajanın çevresini algılamak ve çevredeki engellere tepki vermek için kullanılmaktadır.

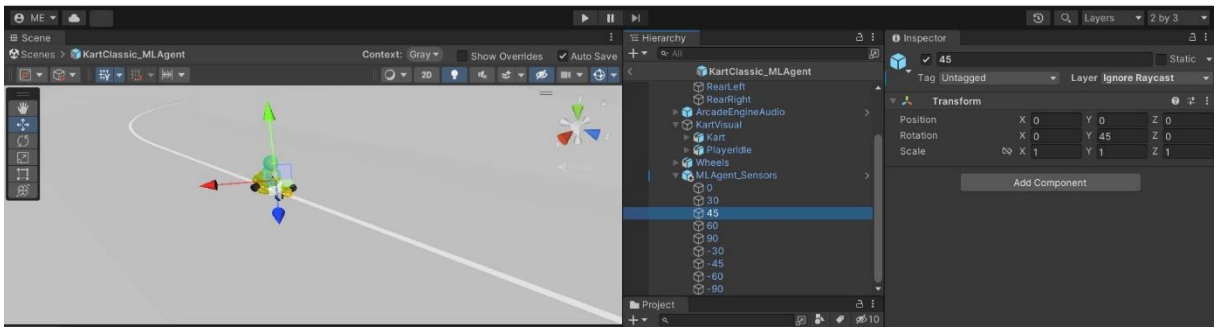
Ajanın etrafında belirli açı aralıklarıyla yerleştirilen ışınlar, ajanın etrafındaki engelleri tespit etmesine ve uygun şekilde tepki vermesine olanak tanır. Bu ışınlar, birer sensör gibi davranarak ajanın çevresindeki nesneleri algılamasını sağlar. Örneğin, ajanın önünde bir engel varsa, ışınlar bu engeli tespit eder ve ajanın engelden kaçınması gerektiğini anlamasını sağlar.

Raycast sistemi, ajanın çevresini gerçek zamanlı olarak tarayarak etkileşimlerini ve tepkilerini kontrol etmesine yardımcı olur. Bu sayede ajan, oyun içerisinde çevresel faktörlere duyarlı bir şekilde hareket edebilir ve uygun kararlar alabilir.

Yapay zeka için önemli olan çevre algısı, Raycast gibi tekniklerle sağlanır. Bu sayede ajanlar, oyun dünyasındaki nesneleri ve engelleri fark edebilir, çevresel koşullara uyum sağlayabilir ve doğru hareketleri gerçekleştirebilir. Bu da oyunlarda yapay zekanın daha gerçekçi ve akıllı bir şekilde davranmasını sağlar.



Şekil 21: *Inspector penceresi ve Kart Agent Script'i altında sensor kısmı*



Şekil 22: *Arayüzdeki Sensorlerin arka planda nasıl olduğu*

Projedeki yapay zeka sisteminin çalışabilmesi ve eğitilebilmesi için Python'un kullanılması gerekmektedir. Python, popüler bir programlama dilidir ve yapay zeka alanında yaygın olarak kullanılmaktadır. Eğer kullanılan cihazda Python yüklü değilse, öncelikle Python'un indirilip kurulması gerekmektedir.

Python, resmi web sitesi üzerinden indirilebilir. Web sitesine giderek en son sürümün indirilmesi ve kurulum adımlarının takip edilmesi gerekmektedir. İndirilen Python sürümüne uygun kurulum dosyası çalıştırılarak Python kurulumu gerçekleştirilir.

Python kurulduktan sonra, projede kullanılacak olan Python kütüphanelerini de yüklemek gerekmektedir. Bu kütüphaneler, yapay zeka algoritmalarının uygulanmasını ve eğitimini desteklemektedir. Örneğin, TensorFlow, PyTorch veya Scikit-learn gibi kütüphaneler yapay zeka için sıklıkla kullanılan kütüphanelerdir.

Kurulum tamamlandıktan sonra Python, projedeki yapay zeka sistemini çalıştırmak ve eğitmek için kullanılabilir hale gelir. Python, güçlü bir dil olması ve geniş bir kütüphane ekosistemi sunması nedeniyle yapay zeka projeleri için tercih edilen bir seçenektir.

```
C:\Users\emire>cd C:\Users\emire\seminer_proje

C:\Users\emire\seminer_proje>python -m venv venv

C:\Users\emire\seminer_proje>venv\Scripts\activate

(venv) C:\Users\emire\seminer_proje>pip install -Iv mlagents==0.12.0
```

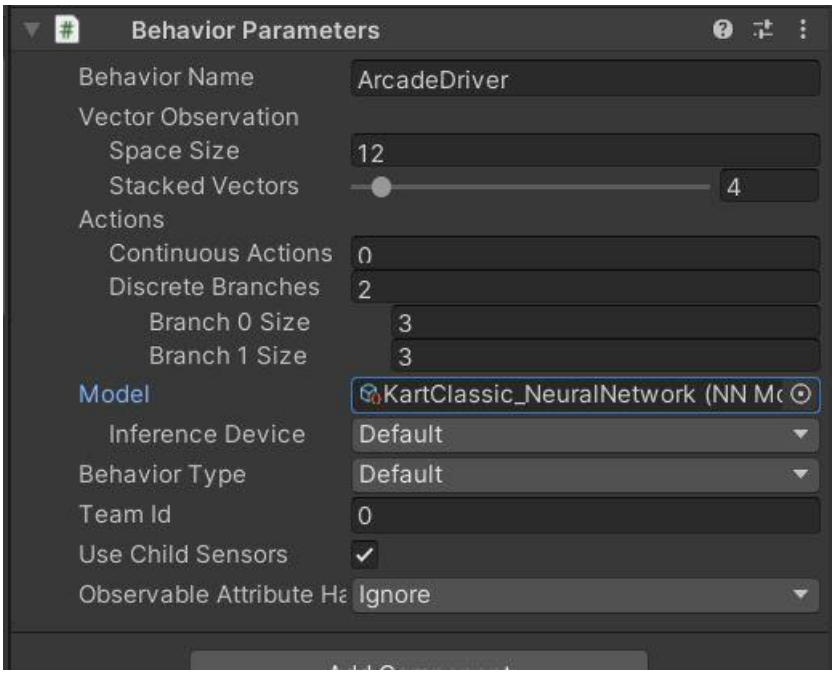
Şekil 23: *CMD üzerinde kullanılması gereken kodlar*

CMD açıldıktan sonra Unity projesinin bulunduğu konuma değiştirilmelidir. Bu konuma gelince, “python -m venv venv” komutuyla sanal çevre (virtual environment) oluşturulmaktadır. Daha sonra gerekli script aktive edilir ve proje için kullanılması gereken “mlagents” toolkit indirilir. Bu komut sayesinde gerekli kütüphaneler (pandas, numpy, tensorflow vb.) indirilir (bu projede uygulamasında mlagents==0.16.0 kullanılmıştır).

```
(venv) C:\Users\emire\seminer_proje>mlagents-learn Assets\Karting\Prefabs\AI\kart_mg_trainer_config.yaml --train --run-i
d=custom-track-1
```

Şekil 24: *Özel oluşturulmuş bir sahnede eğitim yapılabilmesi için gereken kod*

Kod CMD üzerinden çalıştırıldıktan sonra unity arayüzünden play butonuna basılması gerekir ve model eğitilmeye başlar. Sonlandırmak için play butonuna tekrar basılır ve modelin eğitilmesi tamamlanmış olur. Eğitilmiş model ".nn" (neural network) uzantılı bir dosyada kaydedilir. Daha sonra bu modeli, unity arayüzünde davranışsal parametreler kısmında model yerine eklenebilir.



Şekil 25: *Modelin eklenmesi gereken kısım*

Artık işlem tamamdır. Eğitilen model sayesinde NPC(non-playable character)'ler hazırdır. Bu proje için, diğer yarışçılar hazır olmuştur. Başka bir insana gerek kalmadan gerçek bir insan gibi yarışabilen, en azından onları taklit edebilen yapay zeka üretilmiştir.

5 SONUÇ

Bu kılavuzda oyun programlamada yapay zeka konusu ele alındı ve yapay zekanın oyunlardaki önemi ve rolü incelendi. Yapay zeka, oyunlarda gerçekçi ve akıllı davranışlar sergilemek, oyunculara meydan okumak ve oyun deneyimini geliştirmek için kullanılmaktadır.

Yapay zeka tekniklerinin oyunlarda kullanılması, oyunlara öngörülemezlik ve çeşitlilik kazandırırken, aynı zamanda daha karmaşık düşman yapay zekası, daha akıllı oyun karakterleri ve daha gerçekçi düşman davranışları gibi avantajlar sunmaktadır. Yapay zeka, oyunlardaki düşmanların stratejik kararlar almasını, oyuncu davranışlarını analiz etmesini ve oyun dengelemesini sağlamasını mümkün kılar.

Oyunlarda yapay zeka kullanmanın donanımsal avantajları da bulunmaktadır. Yapay zeka işlemleri, oyuncunun donanımını daha az yormakta ve daha akıcı bir oyun deneyimi sunmaktadır. Ayrıca yapay zeka teknikleri, grafikler ve efektler gibi görsel unsurların daha gelişmiş olmasını sağlayarak, oyunların görsel kalitesini artırmaktadır.

Ancak yapay zeka kullanımının dezavantajları da vardır. Yapay zeka algoritmalarının karmaşıklığı, geliştirme sürecini zorlaştırabilir ve daha fazla kaynak gerektirebilir. Ayrıca yapay zeka sistemlerinin tamamen hata yapmaması veya beklentilerin tam olarak karşılanmaması gibi durumlar da olabilir.

Bu kılavuzda, yapay zeka konusunda temel kavramlar ve teknikler, oyunlarda yapay zeka kullanımının avantajları ve dezavantajları, yapay zeka tekniklerinin uygulanması için gerekli adımlar ve kullanılan araçlar hakkında bilgi verilmiştir. Yapay zekanın oyun programlamasındaki potansiyeli ve önemi üzerinde durulmuş ve gelecekte yapay zeka kullanımının daha da yaygınlaşması beklenmektedir.

Bu kılavuzun, oyun programlama ve yapay zeka konularına ilgi duyan herkes için faydalı bir kaynak olmasını umulmaktadır. Yapay zeka tekniklerini oyunlara entegre etmek, oyun deneyimini zenginleştirmek ve oyun dünyasına yeni bir boyut katmak için heyecan verici bir fırsat sunmaktadır.

6 KAYNAKÇA

1. URL https://youtu.be/gYwWolRFt98
2. URL https://forum.unity.com/threads/using-imitation-learning-in-karting-microgame.1104724/
3. URL https://github.com/tensorflow/tensorflow/issues/39130
4. ChatGPT
5. Millington I. Ve Funge J. D. (2006) *Artificial Intelligence for Games*
6. URL https://tr.wikipedia.org/wiki/Makine_%C3%B6%C4%9Frenimi
7. URL https://tr.wikipedia.org/wiki/Bulan%C4%B1k_mant%C4%B1k