**T.C.**

**İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA**

**MÜHENDİSLİK FAKÜLTESİ**



**BİTİRME PROJESİ**

UNITY ÜZERİNDE C# İLE OYUN geliştirme

Elektronik ANABİLİM DALI

YASİN ENES ÖZDEMİR

1316130047

**Haziran, 2020**

**DANIŞMAN**

**Yrd. Doç. Dr. Abdurrahim Akgündoğdu**

**İSTANBUL**

# ÖNSÖZ

Yapay zekâ neredeyse ilkel bilgisayarların çıkışından beri insanlığın yaratmak istediği unsurlar arasındadır. Geçmiş tarihe bakıldığında Yapay zekâ kavramını ilk olarak 1308 yılında Katalan şair ve teolog Ramon Llull Ars, Generalis Ultima (The Ultimate General Art) adlı kitabında kullanmıştır. Burada kavramların kombinasyonlarından oluşan yeni bir bilgi türünden ilk kez bahsedilmiştir. Yapay zekanın ilk olarak kavramlaştırılması Ramon Llull tarafından yapılmıştır.

Hesap makinelerinin babası sayılan matematikçi Blaise Pascal 1642 yılında, vergi memuru olan babasına yardımcı olması için ilk mekanik hesap makinesini geliştirmiştir. Alan Mathison Turing “Makineler düşünebilir mi?” sorunsalını ortaya atarak makine zekâsını tartışmaya açan kişi olmuştur. 1943’te II. Dünya Savaşı sırasında Kripto analizi gereksinimleri ile üretilen elektromekanik cihazlar sayesinde bilgisayar bilimi ve yapay zekâ kavramları oluşmuştur.

İspanyol mühendis Leonardo Torres y Quevedo, 1914 yılına gelindiğinde ilk satranç oynayabilen makineyi tanıtmıştır. Makine hiçbir insan etkisi olmadan oyunu kendi devam ettirebilmektedir. 1951 yılında Nim isimli bir strateji oyunu sanal ortama uyarlanmıştır. Oyun için yaratılan yapay zekâ, insanları yenebilecek düzeyde olmuştur. 1997 yılında yapılan Deep Blue isimli bilgisayar yapay zekâsı sayesinde başlangıçta orta düzey satranç oyuncularını yenmekteydi. İlerleyen zamanlarda yapılan geliştirmeler sonucunda dünyanın en ünlü satranç oyuncusu olan Garry Kasparov’u yenmiştir.

Günümüze bakıldığında ise yapay zekâ konusunda birçok ilerleme kaydedilmiştir. Kullanım alanları arasında sağlık sektörü, sesli asistanlar, e-ticaret, otonom araçlar ve iletişim ön plana çıkmış durumdadır. Çağımızda yapay zekanın hemen hemen kullanılmadığı yer yoktur. Dijital oyunlarda yapay zekâ tekniklerinden olan derin öğrenme modelli yapay zekâ ve kural tabanlı yapay zekâ kullanımı yaygınlaşmıştır.

|  |  |  |
| --- | --- | --- |
| Haziran 2020 |  | Yasin Enes Özdemir |

# İÇİNDEKİLER

**Sayfa No**

|  |  |
| --- | --- |
| **ÖNSÖZ** | **i** |
| **İÇİNDEKİLER** | **ii** |
| 1. **GİRİŞ**    1. **Tanım**    2. **Tarihçe**    3. **Gelişim süreci**       1. **Sembolik Yapay Zekâ**       2. **Sibernetik Yapay Zekâ**       3. **Uzman Sistemler** | **1 2 2 3 4 5 5** |
| 1. **YAPAY ZEKA**    1. **Ajanlar**    2. **Arama Yöntemleri**       1. **Bilgisiz Arama Yöntemleri**          1. **Enine/Yayılımlı Arama**          2. **Eşit maliyetli Arama**          3. **Derinlik öncelikli Arama**          4. **Derinlik sınırlı Arama**          5. **Yinelemeli Derinleşen Arama**          6. **İki Yönlü Arama**       2. **Bilgili Arama Yöntemleri**          1. **1. Aç Gözlü Arama**          2. **A\* Arama** | **7 7 8 9 9 10 10 12 13 13 14 14 16** |
| 1. **DAVRANIŞ AĞAÇLARI**    1. **Operatörler**       1. **Ve Operatörü - Sıra Görevi**       2. **Veya Operatörü - Seçim Görevi**       3. **Paralel Görev**    2. **Dekaratörler**       1. **Tekrarlayıcı**       2. **Çevirici**       3. **Hata Döndür**       4. **Başarı Döndür**       5. **Başarısızlığa Kadar**       6. **Başarıya Kadar** | **18 18 18 20 21 23 23 23 24 25 26 27** |
| 1. **Uygulamalar**    * 1. **Nesneyi Gör Görevi**      2. **Nesneyi Duy Görevi**      3. **Nesneyi Yakala Görevi**      4. **Nesneyi Takip Et Görevi**      5. **Nesneyi Ara Görevi**      6. **Devriye Gez Görevi**      7. **Beklemede Kal Görevi**      8. **Nesneye Saldır Görevi**      9. **Nesneyi Savun Görevi** | **29  29 30 31 33 34 36 38 40 40** |
| 1. **ÖRNEK SAHNE**    1. **Sahne Yapısı**    2. **Ajan Çeşitleri**       1. **Savunan Ajanlar**       2. **Saldıran Ajanlar**       3. **Devriye Gezen Ajanlar** | **42 42 43 43 44 46** |
| **KAYNAKLAR** | **47** |

# GİRİŞ

Bilgisayar ve Video Oyunları pazarı son yıllarda giderek ivmelenen bir gelişme göstermeye başlamıştır. Bunun temel nedenlerinden birisi gelişmiş oyunların artık sadece belirli oyun salonlarından çıkıp masaüstü sistemlerin ve kullanılan özelleşmiş grafik hızlandırıcıları ile hemen her alanda karşımıza çıkmaya başlamasıdır. Hemen her yıl ikiye katlanan işlemci ve grafik işleyici yongaların hızı oyun konusundaki rekabeti körüklemiştir. Özellikle son yıllarda yeni pazarlar arayışına giren dev şirketlerin oyun programlama ve özelleşmiş oyun konsolu tasarımına ağırlık vermeleri de pazardaki rekabetin ve potansiyelin açık bir göstergesidir.

Yıllar geçtikçe işlem gücünün artması sadece oyunların daha hızlı çalışmasını değil oyunların biçimlerini de değiştirmeye başlamıştır. Önceleri sadece hamle tabanlı (satranç, dama ya da basit zeka oyunları) ve düşük kaliteli iki boyutlu karakterlerin tekdüze bir mantığa göre hareketlerine dayanan oyunlar mevcutken günümüz bilgisayarlarında gerçek zamanlı ve neredeyse gerçeğe yakın görünümde ortamlarda onlarca karakterin birbiri ile insan davranışına yakın hareketlerle etkileşimini sergilemektedir. Oyunların ve oyun şirketlerinin her geçen gün artması da kullanıcıların daha gerçekçi grafik ve daha akıllı oyun karakterlerine olan isteğini arttırmıştır. Bu noktada ise oyun programlanın en zor ve en etkileyici yönlerinden yapay zeka devreye girer.

Oyun programlamada kullanılan yapay zekanın amacı oyuncu ile etkileşimde bulunan karakterlerin ya da oyunun geçtiği ortamın mümkün olduğunca gerçek insan, topluluk ve dünya (ya da kimyasal- fiziksel – biyolojik olarak anlamlı ortam) yaşam ortamına benzetilmeye çalışılmasıdır. Başka bir anlamda da bilgisayarın yetenekli bir oyuncuya aynı derecede yetenekli karşılık vermesi, oyuncunun kurduğu planlara benzeyen planlar ya da tuzaklar kurması, gerektiğinde oyuncuyu zor duruma düşürüp onu yenebilmesi oyun programındaki yapay zeka kalitesini belirler.

Ancak burada gözden kaçırılmaması gereken bir nokta oyunlardaki yapay zeka dozunun iyi ayarlanmasıdır. Eğer bilgisayarı oyuncuya karşı çok hızlı reaksiyon verecek şekilde ayarlarsanız ya da oyuncuyu bunaltacak şekilde iyi taktikler gerçekleştirirse oyunun “oynanabilirlik” seviyesi düşer ve kaçınılmaz bir pazar başarısızlığı ortaya çıkabilir. Bunun önüne geçmek için oyunlarda çeşitli yöntemler izlenir. Bunlardan bir tanesi oyuna “seviye” özelliği eklenir ve oyuncu bilgisayarın ne kadar “zeki” davranacağını kendisi belirler. Yukarıda bahsedilen gerekçeden dolayı modern oyunlarda günümüz akademik yapay zeka araştırmalarının popüler konularından olan Yapay sinir ağları (Artificial Neural Networks) ve Genetik Algoritmalar gibi konular birkaç oyun türü hariç halen yeni oyunlarda tercih edilmemektedir.

* 1. **Tanım**

İdealize edilmiş bir yaklaşıma göre yapay zekâ, insan zekâsına özgü olan, algılama, öğrenme, çoğul kavramları bağlama, düşünme, fikir yürütme, sorun çözme, iletişim kurma, çıkarımsama yapma ve karar verme gibi yüksek bilişsel fonksiyonları veya otonom davranışları sergilemesi beklenen yapay bir işletim sistemidir. Bu sistem aynı zamanda düşüncelerinden tepkiler üretebilmeli (eyleyici yapay zekâ) ve bu tepkileri fiziksel olarak dışa vurabilmelidir.

* 1. **Tarihçe**

"Yapay zekâ" kavramının geçmişi modern bilgisayar bilimi kadar eskidir. Fikir babası, "Makineler düşünebilir mi?" sorunsalını ortaya atarak makine zekâsını tartışmaya açan Alan Mathison Turing'dir. 1943'te II. Dünya Savaşı sırasında Kripto analizi gereksinimleri ile üretilen elektromekanik cihazlar sayesinde bilgisayar bilimi ve yapay zekâ kavramları doğmuştur.

Alan Turing, Nazilerin Enigma makinesinin şifre algoritmasını çözmeye çalışan matematikçilerin en ünlü olanlarından biriydi. İngiltere, Bletchley Park'ta şifre çözme amacı ile başlatılan çalışmalar, Turing'in prensiplerini oluşturduğu bilgisayar prototipleri olan Heath Robinson, Bombe Bilgisayarı ve Colossus Bilgisayarları, Boole cebirine dayanan veri işleme mantığı ile Makine Zekâsı kavramının oluşmasına sebep olmuştu.

Modern bilgisayarın atası olan bu makineler ve programlama mantıkları aslında insan zekâsından ilham almışlardı. Ancak sonraları, modern bilgisayarlarımız daha çok uzman sistemler diyebileceğimiz programlar ile gündelik hayatımızın sorunlarını çözmeye yönelik kullanım alanlarında daha çok yaygınlaştılar. 1970'li yıllarda büyük bilgisayar üreticileri olan Microsoft, Apple, Xerox, IBM gibi şirketler kişisel bilgisayar (PC Personal Computer) modeli ile bilgisayarı popüler hale getirdiler ve yaygınlaştırdılar. Yapay zekâ çalışmaları ise daha dar bir araştırma çevresi tarafından geliştirilmeye devam etti.

Bugün, bu çalışmaları teşvik etmek amacı ile Turing'in adıyla anılan Turing Testi ABD'de Loebner ödülleri adı altında makine zekâsına sahip yazılımların üzerinde uygulanarak başarılı olan yazılımlara ödüller dağıtılmaktadır.

Turing Testinin içeriği kısaca şöyledir: birbirini tanımayan birkaç insandan oluşan bir denek grubu birbirleri ile ve bir yapay zekâ diyalog sistemi ile geçerli bir süre sohbet etmektedirler. Birbirlerini yüz yüze görmeden yazışma yolu ile yapılan bu sohbet sonunda deneklere sorulan sorular ile hangi deneğin insan hangisinin makine zekâsı olduğunu saptamaları istenir. İlginçtir ki, şimdiye kadar yapılan testlerin bir kısmında makine zekâsı insan zannedilirken gerçek insanlar makine zannedilmiştir.

Loebner Ödülü kazanan yapay zekâ diyalog sistemlerinin yeryüzündeki en bilinen örneklerinden biri A.L.I.C.E'dir. Carnegie üniversitesinden Dr.Richard Wallace tarafından yazılmıştır. Bu ve benzeri yazılımlarının eleştiri toplamalarının nedeni, testin ölçümlendiği kriterlerin konuşmaya dayalı olmasından dolayı programların ağırlıklı olarak diyalog sistemi (chatbot) olmalarıdır.

Türkiye'de de makine zekâsı çalışmaları yapılmaktadır. Bu çalışmalar doğal dil işleme, uzman sistemler ve yapay sinir ağları alanlarında Üniversiteler bünyesinde ve bağımsız olarak sürdürülmektedir. Bunlardan biri, D.U.Y.G.U. - Dil Uzam Yapay Gerçek Uslamlayıcı 'dır.

* 1. **Gelişim süreci**

İlk araştırmalar ve yapay sinir ağları. İdealize edilmiş tanımıyla yapay zekâ konusundaki ilk çalışmalardan biri McCulloch ve Pitts tarafından yapılmıştır. Bu araştırmacıların önerdiği, yapay sinir hücrelerini kullanan hesaplama modeli, önermeler mantığı, fizyoloji ve Turing'in hesaplama kuramına dayanıyordu. Herhangi bir hesaplanabilir fonksiyonun sinir hücrelerinden oluşan ağlarla hesaplanabileceğini ve mantıksal ve ve veya işlemlerinin gerçekleştirilebileceğini gösterdiler. Bu ağ yapılarının uygun şekilde tanımlanmaları hâlinde öğrenme becerisi kazanabileceğini de ileri sürdüler. Hebb, sinir hücreleri arasındaki bağlantıların şiddetlerini değiştirmek için basit bir kural önerince, öğrenebilen yapay sinir ağlarını gerçekleştirmek de olası hale gelmiştir.

1950'lerde Shannon ve Turing bilgisayarlar için satranç programları yazıyorlardı. İlk yapay sinir ağı temelli bilgisayar SNARC, MIT'de Minsky ve Edmonds tarafından 1951'de yapıldı. Çalışmalarını Princeton Üniversitesi'nde sürdüren Mc Carthy, Minsky, Shannon ve Rochester'le birlikte 1956 yılında Dartmouth'da iki aylık bir açık çalışma düzenledi. Bu toplantıda birçok çalışmanın temelleri atılmakla birlikte, toplantının en önemli özelliği Mc Carthy tarafından önerilen yapay zekâ adının konmasıdır. İlk kuram ispatlayan programlardan Logic Theorist (Mantık kuramcısı) burada Newell ve Simon tarafından tanıtılmıştır.

Yeni yaklaşımlar. Daha sonra Newell ve Simon, insan gibi düşünme yaklaşımına göre üretilmiş ilk program olan Genel Sorun Çözücü (General Problem Solver)'ı geliştirmişlerdir. Simon, daha sonra fiziksel simge varsayımını ortaya atmış ve bu kuram, insandan bağımsız zeki sistemler yapma çalışmalarıyla uğraşanların hareket noktasını oluşturmuştur. Simon'ın bu tanımlaması bilim adamlarının yapay zekâya yaklaşımlarında iki farklı akımın ortaya çıktığını belirginleştirmesi açısından önemlidir: Sembolik Yapay Zekâ ve Sibernetik Yapay Zekâ.

* + 1. **Sembolik yapay zekâ**

Simon'ın sembolik yaklaşımından sonraki yıllarda mantık temelli çalışmalar egemen olmuş ve programların başarımlarını göstermek için bir takım yapay sorunlar ve dünyalar kullanılmıştır. Daha sonraları bu sorunlar gerçek yaşamı hiçbir şekilde temsil etmeyen oyuncak dünyalar olmakla suçlanmış ve yapay zekânın yalnızca bu alanlarda başarılı olabileceği ve gerçek yaşamdaki sorunların çözümüne ölçeklenemeyeceği ileri sürülmüştür.

Geliştirilen programların gerçek sorunlarla karşılaşıldığında çok kötü bir başarım göstermesinin ardındaki temel neden, bu programların yalnızca sentaktik süreçleri benzeşimlendirerek anlam çıkarma, bağlantı kurma ve fikir yürütme gibi süreçler konusunda başarısız olmasıydı. Bu dönemin en ünlü programlarından Weizenbaum tarafından geliştirilen Eliza, karşısındaki ile sohbet edebiliyor gibi görünmesine karşın, yalnızca karşısındaki insanın cümleleri üzerinde bazı işlemler yapıyordu. İlk makine çevirisi çalışmaları sırasında benzeri yaklaşımlar kullanılıp çok gülünç çevirilerle karşılaşılınca bu çalışmaların desteklenmesi durdurulmuştu. Bu yetersizlikler aslında insan beynindeki semantik süreçlerin yeterince incelenmemesinden kaynaklanmaktaydı.

* + 1. **Sibernetik yapay zekâ**

Yapay sinir ağları çalışmalarının dahil olduğu sibernetik cephede de durum aynıydı. Zeki davranışı benzeşimlendirmek için bu çalışmalarda kullanılan temel yapılardaki bazı önemli yetersizliklerin ortaya konmasıyla birçok araştırmacılar çalışmalarını durdurdular. Buna en temel örnek, Yapay sinir ağları konusundaki çalışmaların Minsky ve Papert'in 1969'da yayınlanan Perceptrons adlı kitaplarında tek katmanlı algaçların bazı basit problemleri çözemeyeceğini gösterip aynı kısırlığın çok katmanlı algaçlarda da beklenilmesi gerektiğini söylemeleri ile bıçakla kesilmiş gibi durmasıdır.

Sibernetik akımın uğradığı başarısızlığın temel sebebi de benzer şekilde Yapay Sinir Ağının tek katmanlı görevi başarması fakat bu görevle ilgili vargıların veya sonuçların bir yargıya dönüşerek diğer kavramlar ile bir ilişki kurulamamasından kaynaklanmaktadır. Bu durum aynı zamanda semantik süreçlerin de benzeşimlendirilememesi gerçeğini doğurdu.

* + 1. **Uzman sistemler**

Her iki akımın da uğradığı başarısızlıklar, her sorunu çözecek genel amaçlı sistemler yerine belirli bir uzmanlık alanındaki bilgiyle donatılmış programları kullanma fikrinin gelişmesine sebep oldu ve bu durum yapay zekâ alanında yeniden bir canlanmaya yol açtı. Kısa sürede Uzman sistemler adı verilen bir metodoloji gelişti.

Uzman sistemler bir konuda belli ön koşullar aynı anda var olduğunda konunun bir uzmanın (bazen ne olasılıkla) ne karar alacağını belirleyen kuralların tümünü içeren bir programı gelen problemlere uygulamak temellidir. Bunun bir avantajı her verilen kararın hangi kurallar uygulanarak verildiğinin kolayca bilinmesi idi. Bu birçok kuralcı bürokratik karar örgütleri için kolayca uygulamalar geliştirilebilmesi demekti. Bu doğal olarak bir otomobilin tamiri için önerilerde bulunan uzman sistem programının otomobilin ne işe yaradığından haberi olmaması da demekti. Buna rağmen uzman sistemlerin başarıları beraberinde ilk ticari uygulamaları da getirdi.

Yapay zekâ yavaş yavaş bir endüstri hâline geliyordu. DEC tarafından kullanılan ve müşteri siparişlerine göre donanım seçimi yapan R1 adlı uzman sistem şirkete bir yılda 40 milyon dolarlık tasarruf sağlamıştı. Birden diğer ülkeler de yapay zekâyı yeniden keşfettiler ve araştırmalara büyük kaynaklar ayrılmaya başlandı. 1988'de yapay zekâ endüstrisinin cirosu 2 milyar dolara ulaşmıştı.

# Yapay zeka

Bu bölümde oyun geliştirme aşamalarında bilinmesi gereken yapay zeka teorik bilgileri incelenecektir.

* 1. **Ajanlar**

Algıladıklarını davranışlara haritalayan programlardır. Ortamdan bilgiyi alarak davranışları hakkında karar verirler ve bu kararları uygularlar. Zeki etmenlerin yazılım etmenleri, özerk etmenler, uyarlanabilir ara yüzeyler, kişisel ara yüzeyler, ağ etmenleri gibi farklı isimleri vardır. Tüm bu kavramlar arasında küçük farklılıklar olmasına rağmen, bu kavramların tamamı zeki etmenleri ifade etmede kullanılırlar. Bir zeki etmen, muhakeme yapma ve öğrenme davranışlarının derecelenmesi olarak ifade edilen zekâ ve sistem içinde etmenler ve diğer varlıklar arasındaki etkileşimin doğası ile nitelenebilen özerklik boyutları ile tanımlanır.

En az düzeyde ortamı, durumu algılama ve yorumlama, kararlar verme ve davranışları kontrol etme yeteneği olan zekâ kavramı; ‘Algılayıcılar aracılığı ile çevresini fark edebilen ve çevresini etkileyebilen herhangi bir şey’ olarak tanımlanan akıllı ajanların temelini oluşturmaktadır.

Temel amaç ortamdan algılanan olaylara otomatik ve uygun cevaplar veren özerk sistemler oluşturabilmektir.

Ajanlar algıları tepkilere dönüştüren fonksiyondur ve yeni algılar geldikçe güncellenecek olan veri yapılarını içermektedir. Bu veri yapıları ajanın karar verme yöntemlerine göre işlenerek verilecek tepki üretilir ve algılayıcılardan elde edilen algıları programa aktararak programı çalıştırıp, programın tepki seçenekleri etkileyicilere iletilir.

metin, çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 1 Ajanın Yapısı

* 1. **Arama Yöntemleri**

Arama algoritmaları yapay zeka uygulamalarında önemli yere sahiptirler. Birçok problemin çözümünde arama algoritmaları hızlı ve etkili çözümler sunmaktadır. Yapay zeka problemlerinin çözümünde arama evrensel bir tekniktir.

Bir problem için doğru arama stratejisini seçmek gerekir. Burada arama stratejilerinden hangisini seçeceğini belirlerken aşağıdaki kriterlere bakmak gerekir:

Tamlık-Bütünlük (Completeness) : Problemin bir çözümü olduğunda seçilecek olan strateji bu çözümü bulabilecek mi?

Zaman Karmaşıklığı (Time Complexity) : Bir çözüm bulmak ne kadar zaman alacak?

Uzay Karmaşıklığı (Space Complexity) : Aramayı yapmak için ne kadarlık bir hafıza(memory) gerekli olacak? Rekürsif olan algoritmalarda bir önceki durum saklanacağından dolayı problemin çözümü için gerekli hafıza alanı rekürsif olmayan algoritmalara göre daha fazla olacaktır.

Optimalite-Optimallık-Eniyileme (Optimality) : Farklı çözümler mevcut iken seçilen stratejinin çözümü optimal olan mı?

Körüne Arama-Bilgisiz Arama (Blind Search) : Burada sonuca varmak için içinde bulunan durumun sonuç ile arasındaki adım sayısı veya mesafesinin bilinmemesidir. Yani çözüme ne zaman varılacağı bilinmemektedir. Burada sadece bilinen şey içinde bulunan durumun hedef durum olup olmadığıdır.

* + 1. **Bilgisiz Arama Yöntemleri**

Arama işleminin bilmeyerek yapılması demek, arama algoritmasının, probleme özgü kolaylıkları barındırmaması demektir. Yani her durumda aynı şekilde çalışan algoritmalara uninformed search (bilmeden arama) ismi verilir.

* + - 1. **Enine/Yayılımlı Arama**

Yaygın olan bir arama stratejisidir. Bu stratejide bulunan durum yani düğüm başlangıçta kök düğüm olduğundan ilk önce kök düğüm genişletilir. Genişletmekten kasıt kök düğümden giden bütün yolların ziyaret edilmesidir. Böylece kök düğümün bütün komşuları keşfedilmiş olur. d derinliğinde bulunan tüm düğümler arama ağacında d+1 derinlikte bulunan düğümlerden önce açılırlar(genişletilirler). Bir problemin çözümü varsa enine arama algoritması çözümü garanti eder. Eğer birkaç çözüm var ise en üstte olan yani ilk bulunan çözümü çözüm olarak kabul eder. Yukarıdaki kriterlere göre enine arama tamdır, çünkü çözüm var ise bulmayı garanti eder.

Enine arama algoritması, problemin çözümü daha derinlerde ise verimsiz bir algoritmadır. Çünkü açılan düğüm sayısı artacaktır. Çözümün derinde olması dallanma etmeni(branching factor) olarak bilinmektedir. Başlangıç düğümü de dahil olmak üzere her düğümün b sayıda çocuğu olduğunu varsayalım. Kök ilk genişletildiğinde b sayıda dal olacaktır bir sonraki derinlikte b² dallanma olacaktır. Bu şekilde d. derinlikte b üzeri k olacaktır.

Toplamda değerlendirilmesi gereken düğüm sayısı=1+b+b²+b³… olacaktır. Bu da zaman karmaşıklığı bakımından önemlidir.

Örnek olarak eğer her düğümün dallanma sayısı b=10 alırsak ve derinliği de d=10 alırsak,

toplam düğüm sayısı=10 000 000 000

gereken zaman=128 gün

gerekli bellek=1 terabyte

Yukarıdaki örnekte de görüldüğü gibi enine arama algoritması dallanma faktörüne göre verimli olup olamadığı belli olmaktadır.

Algoritmanın adımları:

* Öncelikle bir başlangıç düğümü seçilir ve bu düğüm işaretlenir.
* Bu düğümün komşuları sırasıyla bir K listesine yazılır. Bu K listesi komşulukları tutan bir listedir.
* Bu komşu düğümler teker teker ziyaret edilir ve işaretlenerek Z listesine eklenir. Z listesi ziyaret edilen düğümleri tutan listedir.
* K listesindeki ilk düğüm alınarak işaretlenir ve K listesinden silinir.
* Silinen düğümün komşuları K listesine eklenir. Bu ekleme yapılırken eğer eklenecek düğüm Z listesinde varsa yani daha önce ziyaret edilmişse eklenmeyecektir.
* Silinen düğümün ziyaret edilen düğümleri Z listesine eklenir (var olanlar eklenmeyecek).
* Bu şekilde K listesindeki tüm düğümler silininceye kadar devam edilir.
  + - 1. **Eşit maliyetli Arama**

Bir düğüme olan yolları maliyetlerine göre artacak şekilde sıralar. Genişlemeyi yani komşuları ziyaret etmeyi maliyeti en az olan, en yakın olanı seçer ve genişlemeyi bu düğümle yapar.

Eğer tüm yolların maliyeti aynı ise enine arama gibidir. Yolları, maliyeti artan şeklinde genişletir(ziyaret eder)

* + - 1. **Derinlik öncelikli Arama**

Derinlik öncelikli aramada, arama ağacı fazla dallanmadan kökten gidilebilecek en uzak düğüme kadar ilerlenir. Burada kök olarak herhangi bir düğüm alınabilir. Kök düğümden yaprak düğüme giden, tek yol gibi düğümler her iterasyonda saklanır ve düğümler doğrusal bir yapıda depolanırlar. Bundan dolayı kullanılan veri yapısı LIFO Last Input First Out(son giren ilk çıkar)’dur. Bu şekilde kökten yaprağa tek yol olarak tutulmasından dolayı bellek gereksinimi oldukça makuldür.

Dallanma faktörüne baktığımızda ise b dallanma sayısı ve d derinliği için gerekli bellek alanı b\*d olacaktır. Enine aramaya göre ne kadar verimli olduğu görülmektedir.

Derinine aramada olası sorunlar problemin çözümünde gidilen yanlış yolda sıkışıp kalmaktır. Birçok problemin sonsuz veya çok derin ağaçları olabilmektedir böyle durumlarda derinine arama uygun bir çözüm bulamayacaktır. Bu şekilde arama bir önceki duruma dönmeden sürekli devam edecektir, hatta bir önceki seviyede bir çözüm bulunmuşsa bile yine devam edecektir. Böylece derinine arama algoritması bu tür sonsuz veya çok derin ağaçlarda sonsuz döngüye girer ve asla bir çözüme ulaşamaz çözüm bir önceki seviyede olsa bile. Bir diğer durum ise algoritma bir çözüm bulabilir ama bu optimal çözüm olmayabilir. Yani bir üst seviyedeki çözüm ile en son bulunan çözüm arasında zaman, bellek bakımından farklar olacaktır. Derine arama yukarıdaki kriterlere göre tam değildir. Rekürsif çalışan bir fonksiyon olarak yazılabilir.

Algoritmanın adımları:

* Başlangıç düğümü seçilir ve ziyaret edilir.
* Başlangıç düğümünün ziyaret edilmemiş komşularından bir tanesi seçilir ve ziyaret edilir.
* Bu ziyaret edilen komşunun da komşularında gidilir ve ve ziyaret edilmeyen bir komşusu seçilerek ziyaret edilir.
* Bu şekilde sürekli derine inilir, en son derinlikte ziyaret edilmeyen düğüm kalmayınca tekrar yukarı çıkılır ve işlemler aynı şekilde devam edilir.

Özet olarak; verilen tüm düğümler gezilecek şekilde, sırayla ziyaret edilir, gezilmemiş bir komşusu var ise ona gidilir ve daha sonra sıradaki kendi komşusundan önce komşusunun komşusuna gidilir böyle devam eder.

* + - 1. **Derinlik sınırlı Arama**

Bu algoritma esas olarak derin öncelikli arama (depth first search DFS) ile aynı çalışmaktadır ancak tek farkı arama işlemi sırasında özellikle dairelere (cycles) takılma ihtimaline karşı sınır önlemi alınmış olmasıdır.

Kod yapısı şu şekildedir.



Şekil 2 Derinlik sınırlı Arama Kod Yapısı

Özyineli fonksiyonda (recursive function) bakılan düğüm hedef olana kadar dolaşma işlemi devam etmektedir. Dolaşma işlemi sırasında klasik derin öncelikli aramalarda kullanılan yığın (stack) kullanılmış ve geçilen düğümler geri dönülüp aranmak üzere yığında tutulmuştur.

Şayet aranan düğüm verilen derinlikten daha derin değilse arama işlemi devam etmektedir ancak verilen derinlik geçildiği zaman arama işlemi daha derine gitmemekte ve artık o ana kadar aranmak üzere yığınladığı düğümleri işlemektedir.

* + - 1. **Yinelemeli Derinleşen Arama**

Derinlik öncelikli drama (depth first search) üzerine kurulu olduğu için, literatürde “iterative deepening depth first search (yinelemeli derinleşen, derin öncelikli arama)” olarak da geçmektedir.

Derinlik değerini bir değişkende tutmakta ve bu değeri her adımda arttırmaktadır.

Yineleme yapısı (iteration) basit bir döngü (loop) olarak düşünülebilir ve her adımda derinliğin, bir döngü değişkeni (loop variable) gibi düşünülerek derinleştiği kabul edilebilir.

* + - 1. **İki Yönlü Arama**

bir bilgi kaynağı veya veri yapısı üzerinde problemi her adımda iki parçaya bölerek yapılan arama algoritmasının ismidir. Bu anlamda bazı kaynaklarda bölerek arama olarak da geçmektedir.

Aramaya başlangıç durumundan ileri doğru ve sondan geriye doğru arama yapar belirlenen bir durumda ikisi karşılaşana kadar devam eder. Baştaki arama ve sondaki arama her birisi toplam yolun yarısını almış olur.

Algoritmanın şu şekildedir:

* Problemde aranacak uzayın tam orta noktasına bak
* Şayet aranan değer bulunduysa bit
* Şayet bakılan değer aranan değerden büyükse arama işlemini problem uzayının küçük elamanlarında devam ettir.
* Şayet bakılan değer arana değerden küçükse arama işlemini problem uzayının büyük elemanlarında devam ettir.
* Şayet bakılan aralık 1 veya daha küçükse aranan değer bulunamadı olarak bitir.

Problem her seferinde aranan sayıya göre ortadan ikiye bölünmektedir. Bu anlamda problemin log2(n) adımda çözülmesi beklenir. Burada logaritmik fonksiyonların üssel fonksiyonların tersi olduğu ve her adımda problemi iki parçaya böldüğümüzü hatırlayınız.

Arama algoritmasının bir dizi üzerinde başarılı çalışması için dizinin sıralı olması gerekir. Yukarıdaki koddan anlaşılacağı üzere ortadaki elemana bakıldığında, aranan sayı ya bakılan sayıdan büyük ya da küçüktür. Dolayısıyla algoritmamız, ya bakılan sayıdan küçük olan sayılar arasında arama yapacak ya da büyük olan sayılar arasında arama yapacaktır.

* + 1. **Bilgili Arama Yöntemleri**

Arama işleminin bilerek yapılması, algoritmanın probleme ait bazı özellikleri bünyesinde barındırması ve dolayısıyla arama algoritmasının problem bazlı değişiklik göstermesi demektir.

* + - 1. **Aç Gözlü Arama**

Algoritma üretme yöntemlerinden birisi olan açgözlü yaklaşımına göre mümkün olan ve sonuca en yakın olan seçim yapılır. Yani basitçe bir seçim yapılması gerektiğinde sonuca en çok yaklaştıracak olan seçimin yapılmasını önerir. Ancak mâlum olduğu üzere bu seçim her zaman için en iyi seçim değildir.

Örneğin para üzeri verilmesi (coin exchange problem) için açgözlü yaklaşımının kullanılmasını düşünelim. Bu problemde bir satıcı kendisinden alışveriş yapan kişiye para üzeri vermektedir. Ödenmesi gereken miktar bu örnekte 24 olsun ve para birimlerimiz 20, 19, 5, 1 olsun. (yani para birimi olarak bu para birimleri bulunuyor)

açgözlü yaklaşımına göre satıcı 24’ü tamamlamak için elindeki para birimlerinden sonuca en çok yaklaştıran 20lik birimi seçecektir. Daha sonra geriye kalan boşluğu (24-20=4) doldurmak için elindeki tek imkan olan 4 tane 1lik birimle dolduracaktır ve toplamda 5 adet bozuk parayı müşteriye geri verecektir. Oysaki aynı problem bir 19luk bir de 5lik bozuk paralar ile çözülerek 2 bozuk para vermek mümkün olabilirdi.

Hedef düğüme en yakın olduğu tahmin edilen düğümü genişletir. Düğümleri f(x)=h(x) baz alarak açar. Yani toplam maliyet sezgisel(tahmini)’dir. Öncelikli kuyruk veri yapısı uygulanır.

Kod yapısı şu şekildedir:

ekran görüntüsü, oturma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 3 Aç Gözlü Arama Kod Yapısı

* + - 1. **A\* Arama**

Maliyeti fazla olan yolların genişletilmesini engeller, sadece en faydalı olabilecek yolları genişletir.

Başlangıç s düğümünden x düğümüne kadar olan yolu g(x) ile x düğümünden ise f hedef düğümüne olan yol değerini h(x) ile ifade edersek bu algoritma için aşağıdaki bağıntı elde edilir.

f(x)=g(x)+h(x),

g(x) x durumunun gerçek olan o anki değeri,

h(x) ise x düğümünden çözüme olan gidişlerin sezgisel değeridir.

f(x) ise toplam n yolun içinden hedefe olan toplam tahmini maliyettir. Artan sırada öncelikli kuyruk veri yapısı uygulanır.

Kod yapısı şu şekildedir:



Şekil 4 A Star Kod Yapısı

# DAVRANIŞ AĞAÇLARI UYGULAMALARI

Davranış ağaçları yukarıdan aşağı ve soldan sağa doğru öncelikli çalışır.

* 1. **Operatörler**
     1. **Ve operatörü - Sıra görevi**

Sıra görevi \ve\ işlemine benzer. Eğer alt görevlerinden biri hata döndürür ise Sıra görevi de hata döndürür ve karar ağacında bulunan bir sonraki göreve geçilir. Bir alt görev başarı döndürürse sırasıyla ve varsa sonraki alt görevler hata döndürene kadar çalışmaya devam eder. Tüm alt görevler başarı döndürürse Sıra görevi de başarı döndürür. Örnek üzerinden inceleyelim.

siyah, oturma, işaret, tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 5 Sıra Görevi Davranış Ağacı Örnek Şeması

Sıra görevi aktif olduğunda bir alt görevleri çalıştırmaya başlar. Öncelik soldaki görevin olduğu için Nesneyi Görme çalışır. Eğer görev başarısız olursa bir sonraki görev olan Kovalama çalışmaz ve Sıra Görevi hata döndürür. Aynı katmandaki Beklemede Kal Görevi çalışmaya başlar. Eğer Nesneyi Görme başarılı olursa aynı katmanda bulunan Kovalama adlı görev çalışır. Eğer Kovalama başarısız olursa Sıra Görevi hata döndürür ve aynı katmandaki Beklemede Kal Görevi çalışmaya başlar. Eğer Kovalama başarılı olursa Sıra Görevi de başarı döndürür ve aynı katmandaki Beklemede Kal Görevi çalışmaya başlar.

Sıra görevinin kod yapısı şu şekildedir.

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 6 Sıra Görevi Kod Yapısı

* + 1. **Veya operatörü - Seçim görevi**

Seçimgörevi \veya\ işlemine benzer. Alt görevlerinden biri başarıya döndüğünde başarı döndürür. Bir alt görev başarısızlık döndürürse, sıralı olarak bir sonraki görevi çalıştırır. Hiçbir alt görev başarı döndürmezse başarısızlık döndürür.

Seçim görevinin kod yapısı şu şekildedir.

**ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 7 Seçim Görevi Kod Yapısı

* + 1. **Paralel görev**

Sıra görevine benzer şekilde, paralel görev, alt görev başarısız olana kadar her alt görevi çalıştırır. Aradaki fark, paralel görevin tüm alt görevlerini aynı anda yürütmesi ile her görevi aynı anda yürütmesi. Dizi sınıfı gibi, paralel görev de tüm alt görevleri geri döndüğünde başarı döndürür. Bir görev hata döndürürse, paralel görev tüm alt görevleri sonlandırır ve hata döndürür.

oturma, tablo, telefon, dizüstü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 8 Paralel Görev Kod Yapısı

* 1. **Dekaratörler**
     1. **Tekrarlayıcı**

Tekrarlayıcı , alt görev belirtilen sayıda çalıştırılana kadar alt görevinin yürütülmesini tekrarlar. Alt görev başarısız olsa bile alt görevi yürütmeye devam etme seçeneği var.

Kod yapısı şu şekildedir.

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 9 Tekrarlayıcı Kod Yapısı

* + 1. **Çevirici**

Çevirici görevi, yürütme işlemi tamamlandıktan sonra alt görevin dönüş değerini ters çevirir. Alt görev başarı döndürürse sürücü görevi başarısız olur. Alt görev başarısızlık döndürürse sürücü görevi başarıya döner.

Kod yapısı şu şekildedir.

ekran görüntüsü, telefon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 10 Çevirici Kod Yapısı

* + 1. **Hata Döndür**

Hata döndür görevi, alt görevin çalıştığı durumlar dışında her zaman başarısızlık döndürür.

Kod yapısı şu şekildedir.

ekran görüntüsü, oturma, telefon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 11 Hata Döndür Kod Yapısı

* + 1. **Başarı Döndür**

Başarı döndür görevi, alt görevin çalıştığı durumlar dışında her zaman başarı döndürür.

Kod yapısı şu şekildedir.

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 12 Başarı Döndür Kod Yapısı

* + 1. **Başarısızlığa Kadar**

Başarısızlığa kadar görevi, alt görev başarısızlık döndürene kadar alt görevini yürütmeye devam eder.

Kod yapısı şu şekildedir.

ekran görüntüsü, telefon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 13 Başarısızlığa Kadar Kod Yapısı

* + 1. **Başarıya Kadar**

Başarıya kadar görevi, alt görev başarıya dönene kadar alt görevini yürütmeye devam eder.

Kod yapısı şu şekildedir.

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 14 Başarıya Kadar Kod Yapısı

# uygulamalaR

Visual studio üzerinde C# programlama dili kullanılarak geliştirilmiştir. Unity NavMesh’i kullanarak oyunlara uyarlanmış görevler geliştirdim. Bu geliştirme sürecinde önceden test edilmiş ve kütüphaneye eklenmiş olan bazı fonksiyon ve sınıfları ve bu sınıflara ait değişken ve fonksiyonları kullandım. Bu görevleri karar ağaçlarında mantıklı kombinasyonlar ile bir araya getirip oyun içinde yapay zeka yada bilgisayar diye adlandırılan botları oluşturmayı planlıyorum.

* 1. **Nesneyi Gör Görevi**

Ajanın açısı ve alanı daha önce belirlenmiş olan menzilinde bir nesne varsa görür.

Kod yapısı şu şekildedir.

ekran görüntüsü, oturma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 15 Nesneyi Gör Kod Yapısı

* 1. **Nesneyi Duy Görevi**

Mevcut nesnelerin işitme menzilinde herhangi bir nesne olup olmadığını kontrol eder.

Kod yapısı şu şekildedir.

oturma, telefon, tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 16 Nesneyi Duy Kod Yapısı

* 1. **Nesneyi Yakala Görevi**

Unity NavMesh'i kullanarak belirtilen hedefi arar.

Kod yapısı şu şekildedir.

ekran görüntüsü, telefon, cep telefonu, oturma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 17 Nesneyi Yakala Kod Yapısı

* 1. **Nesneyi Takip Et Görevi**

Unity NavMesh'i kullanarak belirtilen hedefi takip eder.

Kod yapısı şu şekildedir.

ekran görüntüsü, telefon, oturma, cep telefonu içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 18 Nesneyi Takip Et Kod Yapısı

**4.5.Nesneyi Ara Görevi**

Unity NavMesh'i kullanarak dolaşmayı, işitme menzili içinde ve menzil içindeki görevleri birleştirerek bir hedef aramayı gerçekleştirir.

Kod yapısı şu şekildedir.

oturma, telefon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 19 Nesneyi Ara Kod Yapısı

* 1. **Devriye Gez Görevi**

Unity NavMesh'i kullanarak belirtilen geçiş noktalarında devriye gezer.

Kod yapısı şu şekildedir.

oturma, tablo, telefon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 20 Devriye Gez Kod Yapısı

* 1. **Beklemede Kal Görevi**

Belirli bir süre bekler. Görev, bekleyen iş bitene kadar çalışmaya devam eder. Bekleme süresi geçtikten sonra başarıya döndürür.

Kod yapısı şu şekildedir.

oturma, telefon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 21 Beklemede Kal Kod Yapısı

* 1. **Nesneye Saldır Görevi**

En yakın hedefe gider ve ajan belirlenen mesafe içinde olur olmaz saldırmaya başlar

Kod yapısı şu şekildedir.

ekran görüntüsü, telefon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 22 Nesneye Saldır Kod Yapısı

**4.9.Nesneyi Savun Görevi**

Nesneyi savunma yarıçapı içinde korur. Belirli bir yarıçap içindeki bir hedefi arar ve saldırır.

Kod yapısı şu şekildedir.

oturma, tablo, telefon, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Örnek Sahne

elektronik eşyalar, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 24 Unity Arayüzü

* 1. **Sahne Yapısı**

Oluşturduğum sahnede bir labirent ve çeşitli görevlere sahip ajanlar bulunmaktadır. Ajanlar devriye gezen, savunan ve saldıran olarak gruplanmaktadır ve sayıları çeşitlilik göstermektedir. Sahnedeki görev ise mavi renkli takımın bayrağa ulaşması ve belirlenen noktaya taşımasıdır. Kırmızı takımın görevi ise bayrağı korumak ve belirlenen alanı düşman takıma karşı savunmaktır.

oturma, siyah, beyaz, kapat içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 25 Labirent

* 1. **Ajan Çeşitleri**

Sahnede bulunan ajanlar 2 takıma ve 3 farklı göreve ayrılmışlardır.

**5.2.1. Savunan Ajanlar**

Kırmızı renkli olan bu ajanlar 3 tanedir. Labirentin iç bölgesinde konumlandırılmışlardır. Görevleri görüş alanına giren mavi renkli düşman birimlerine karşı bölgeyi savunmak ve sonrasında düşmanı takip edip yakalamaktır.

Davranış ağacı şeması şekildeki gibidir.

iç mekan, siyah, küçük, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 26 Savunan Ajan Davranış Ağacı Şeması

Şema üzerinden anlaşılacağı gibi seçim görevi ile sol ve sağ kollar arasında seçim yapılmaktadır. Sol kolda ise görüş alanı kontrolü yapılır ve görüş alanına önceden belirlenen tipteki ajanlar girerse başarı döndürür ve Nesneyi Savun görevi çalışmaktadır. Burada önceden belirlenen ajan tipleri ister kod ister Unity ara yüzü ile değiştirilebilmektedir. Ben bu durumda mavi renkli ajanları seçtim. Bölge savunmasından sonra da bayrak düşman birimler tarafından ele geçirilirse Nesneyi Yakala görevi aktifleşmekte ve davranış ağacı sonlanmaktadır.

**5.2.2. Saldıran Ajanlar**

Mavi renkli olan bu ajanlar 2 tanedir. Labirentin dış bölgesinde konumlandırılmışlardır. Görevleri labirentin merkezinde bulunan bayrağı almak ve belirlenen bitiş noktasına düşman birimlerine yakalanmadan ulaşmaktır.

Davranış ağacı şeması şekildeki gibidir.

tablo, ışık, çift, siyah içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 27 Saldıran Ajan Davranış Şeması

Şemadan da anlaşılacağı gibi seçim görevi ile sol ve sağ kollar arasında seçim yapılmaktadır. Sol kolda sıra görevi Nesneyi Takip Et hata döndürdüğünde başarı döndürür ve Nesneyi Yakala görevi aktifleşir. Sağ kolda ise Nesneden Uzaklas görevi ancak Nesneyi Takip Et hata döndürdüğü zaman çalışır.

**5.2.3. Devriye Gezen Ajanlar**

Kırmızı renkli olan bu ajanlar 3 tanedir. Labirentin orta bölgesinde konumlandırılmışlardır. Görevleri önceden belirlenmiş olan güzergah üzerindeki kontrol noktaları arasında hareket etmek ve düşman birimlerinin görüş alanına girip girmediğini kontrol etmektir. Yine aynı şekilde düşman birimlerinin tanımlanması kod üzerinden yada Unity ara yüzü üzerinden yapılabilmektedir.

Davranış ağacı şeması şekildeki gibidir.

genel, tuvalet, çift, cadde içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 28 Devriye Gezen Ajan Davranış Şeması

Şemadan da anlaşılacağı gibi sol kolda Devriye Gez görevi gerçekleşirken sağ kolda ise Nesneyi Yakala görevi yerine getirilir.

# KAYNAKLAR

Yapay Zeka Ders notları – Dr. Abdurrahim Akgündoğdu, 2019

<https://tr.wikipedia.org/wiki/Yapay_zekâ>

<https://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php>

<http://bilgisayarkavramlari.sadievrenseker.com/2009/11/23/arama-algoritmalari-search-algorithms/>

<https://medium.com>

<https://www.sciencedirect.com/science/article/pii/S2452315118301516>

<https://docs.unity3d.com/Manual/bolt-connections.html>

<https://docs.microsoft.com/tr-tr/dotnet/csharp/programming-guide/classes-and-structs/inheritance>

<https://app.diagrams.net>

<https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games>

<https://www.aithority.com/computer-games/understanding-the-role-of-ai-in-gaming/>

https://www.cc.gatech.edu/~riedl/pubs/cig13.pdf

# EKLER