Dokuz Eylül Üniversitesi-Mühendislik Fakültesi
Fen ve Mühendislik Dergisi
Cilt 20, Sayı 58, Ocak, 2018

Dokuz Eylul University-Faculty of Engineering
Journal of Science and Engineering
Volume 20, Issue 58, January, 2018

DOI: 10.21205/deufmd. 2018205813

Akıllı Etmenler ile Izgara Tabanlı Bir Mobil Oyunda Yol Bulma

Semih UTKU*[1], Kadir TİLKİ[1]

[1]Dokuz Eylül Üniversitesi, Mühendislik Fakültesi, Bilgisayar Müh. Bölümü, İzmir.
(ORCID: 0000-0002-8786-560X, ORCID: 0000-0002-7271-3195)

(Alınış / Received: 21.04.2017, Kabul / Accepted: 21.12.2017,
Online Yayınlanma / Published Online: 20.01.2018)

Anahtar Kelimeler
Yapay Zekâ,
Akıllı Etmenler,
Yol Bulma,
Dinamik
Programlama

Özet: Büyük bir endüstri haline gelen oyun alanında, her alanda olduğu gibi yapay zekânın kullanımı yaygınlaşmaktadır. Oyunlar da bulunan yapay zekâ oyuna karakterleri hareket ettirme, nereye hareket etmesi gerektiğine karar verme ve taktiksel veya stratejik düşünme yeteneğini kazandırmak için kullanılmaktadır. Bu çalışmada; mobil platformda yapay zekâ kullanan bir oyun geliştirilmiştir. Geliştirilen yapay zekâ ile oyunda yer alan yapay karakterlere, mantıklı bir şekilde hareket edebilme özelliğinin verilmesi üzerine çalışılmıştır. Bunu sağlayabilmek için ilk olarak basit refleks etmeni yöntemi kullanılmıştır. Dinamik programlama yaklaşımıyla bir yol bulma algoritması kullanılarak hedef tabanlı etmen yöntemi geliştirilmiştir. Geliştirilen yöntemde hedef tabanlı oyuncuların hedefe odaklı hareket etmelerinden dolayı, hareketlerinin daha gerçekçi olduğu tespit edilmiştir. Yapılan test sonuçları değerlendirildiğinde, geliştirilen yöntem ile istenilen sonuca ulaşıldığı görülmüştür.

Grid-based Pathfinding in a Mobile Game with Intelligent Agents

Keywords
Artificial
Intelligence,
Intelligent Agents,
Pathfinding,
Dynamic
Programming

Abstract: Like in other areas, the use of artificial intelligence (AI) is spreading among computer games, which has become a huge industry. In most of the games, artificial intelligence enables the game characters to move, make decisions about where to move and generate tactical and strategic thinking capability. In this study, a game using artificial intelligence has been developed for mobile device platform. Stimulation of rational movements in game characters was worked on by using an advanced form of AI. To enable this, Simple Reflex Agents method was adopted. With a dynamic programming strategy approach, a goal finding algorithm was used to develop a Goal Based Agents method. It has been observed that movements of the goal based characters became more realistic, owing to their goal focused movements. The evaluation of the test results has shown that the intended result has been accomplished relying on this method.

*Sorumlu yazar: semih@cs.deu.edu.tr

1. Giriş

Yapay zekâ, insan aktivitesinde yer edinmiş birçok uygulamada kullanılan bir teknolojidir. Yapay zekâ ile bilgisayarın rasyonel ve insansı davranabilme, deneyimlerden bilgi edinebilme ve özgün problemleri çözebilme yetkinliği kazanması sağlanmaktadır. Yüksek başarı oranı ve hesaplamaların kısa süre içerisinde yapılması sayesinde, yapay zekâ olabildiğince her alanda kullanılmaya çalışılmaktadır. Başarıyla kullanıldığı alanlardan bazıları; hava tahmini, veri madenciliği, arama motorları, robotik, genetik, doğal dil işleme çalışmalarıdır [1, 2]. Yapay zekânın sıklıkla kullanıldığı bir diğer alan ise bilgisayar oyunlarıdır.

Bilgisayar oyunları dünya çapındaki eğlence endüstrisinin en büyük bölümlerinden biridir. Sunduğu grafikler, oynanış şekilleri, ses ve müzik efektleri ile bilgisayar oyunları kullanıcılara benzersiz deneyimler yaşatmaktadır. Geçmişte, oyunlarda grafik özelliğinin daha ön plana çıkması ve donanımsal yetersizlikten dolayı oyunlarda yapay zekâya yeterli önem verilmemiştir. Gelişen işlemciler, ana bellek ve grafik kartları, yapay zekânın da oyunlar içerisinde geliştirilmesine olanak sağlamıştır. Böylece iyi grafikler yanında, kullanıcılara daha gerçekçi bir oyun deneyimi yaşatabilmek adına, yapay zekâya verilen önem de artmıştır [3].

Bilgisayar oyunlarını tek oyunculu (single player) veya çok oyunculu (multiplayer) oyunlar olarak kategorilendirebiliriz. Tek oyunculu oyunlarda kullanıcı sayısındaki eksiklikler akıllı robotlarla giderilmektedir. Akıllı robotlar sadece oyun alanında değil, günümüzde yaygın olarak veri madenciliği, e-ticaret, e-posta, alışveriş ve arama motorlarında da kullanılmaktadır. Akıllı robotlar özel geliştirilen bir etmen (agent) yazılımı grubunu ortaya çıkarmıştır. Etmenler ortamı izleyerek karar verebilen yapılardır. Etmenler, algılayıcıları (sensors) aracılığıyla bulunduğu ortamı (environment) keşfeden ve bu ortamda gerçekleştiricileri (actuators) ile hareket eden bir varlık olarak görülmektedir [4]. Etmenlerin temel amacı algılayıcılar ile dış ortamdan algılanan olaylara otomatik ve uygun cevaplar veren bağımsız yapılar oluşturabilmektir. Bilgisayarlar, yapay zekânın temelini oluşturan etmenler sayesinde öğrenme ve kendini geliştirme yeteneği kazanarak, akıllılık özelliğine ulaşmaktadırlar. Oyunlarda kullanılan yapay zekâ sayesinde de kullanıcıya, gerçek oyuncularla oynuyormuş kanısı verilebilmektedir. Çoğu modern oyunda bulunan yapay zekâ üç adet temel ihtiyacı gidermek için geliştirilmektedir: (i) karakterleri hareket ettirme yeteneği, (ii) nereye hareket etmesi gerektiğine karar verme yeteneği, (iii) taktiksel veya stratejik düşünme yeteneği [5]. Bu ihtiyaçların giderilmesi için makine öğrenimi, karar ağaçları ve yol bulma algoritmaları gibi çeşitli yöntemler kullanılmaktadır [6]. Karakterlerin mantıklı bir şekilde hareketini sağlayabilmek için, bulundukları yerden varış noktasına ulaşabilecekleri olabildiğince iyi bir rota tasarlamak gerekmektedir. Bu rotayı tasarlamak için de etkin ve dinamik olarak değişebilen bir yol bulma algoritmasına ihtiyaç duyulmaktadır. Yol bulma konusunda kullanılan farklı ve etkin algoritmalar bulunmaktadır [7].

Bu çalışma kapsamında yapay zekâ teknikleri kullanılarak mobil ortamda oynanabilen çok oyunculu bir oyun geliştirilmiştir. Oyunda kullanıcılar yapay zekâ aracılığıyla hareket ettirilen karakterlere karşı, oynayabilecekleri bir oyun modu bulunmaktadır. Bu oyun modunu destekleyecek yapay zekânın

geliştirilmesinde, yapay zekânın yukarıda belirlenen üç temel ihtiyacın giderilmesi gözetilerek, özellikle karakterlerin hareket etmeleri gereken yön üzerinde kararlar verebilmelerine odaklanılmıştır.

Çalışma kapsamında mobil cihazlarda performansı olabildiğince yüksek tutmak adına, önce Basit Refleks Etmenleri - Simplex Reflex Agents (BRE) yapıları ve Hedef Tabanlı Etmenler - Goal Based Agents (HTE) [8] denenmiştir. BRE yapılarının uygulanması sonucunda oyun performansında yeterli başarı alınamadığı için, farklı bir yol bulma algoritmasının kullanılması gerekliliği görülmüştür. Geliştirilen oyunun sahip olduğu oyun alanının bir matris yapısında olması nedeniyle, yol bulma işlemi dinamik programlama ile gerçekleştirilmiştir. Manhattan Turist Problemini [9] çözmek adına kullanılan algoritma üzerinde bazı değişikliklerin yapılmasının ardından, bu algoritma yapay zekânın geliştirilmesinde kullanılmıştır. BRE ve HTE yapılarının farklı kullanıcı sayılarında başarı yüzdeleri karşılaştırılmış ve istenen performans ve başarı kıstası elde edilmiştir.

Çalışmanın organizasyonu şu şekildedir: 2. Bölümde ilişkili çözümler ve bu alanda yapılmış güncel akademik çalışmalar yer almaktadır. 3. Bölümde sisteminin motivasyon ve geliştirilen oyunun kısaca özellikleri aktarılmıştır. Bölüm 4'te oyun için geliştirlen yapay zekâ yöntemlerinden bahsedilmiştir. 5. bölümde yöntemlerin uygulandığı test sonuçları değerlendirilmiştir. Son olarak bölüm 6 da, sonuç kısmı yer almaktadır.

2. İlişkili Çalışmalar

Yapay zekâ geçmişte ve günümüzde çeşitli alanlarda ve çeşitli şekillerde uygulanmıştır. Çalışmalardan bazıları insan ve bilgisayar arasındaki ilişkiler üzerine, konuşma tanıma [10] veya el yazısı tanıma sistemleri [11] üzerine yapılmışken, bazı çalışmalar da güvenlik sistemlerini geliştirmek adına örüntü tanıma yöntemi kullanılarak yapılmıştır [12]. Günümüzde çok geniş bir kullanım alanına sahip olan yapay zekâ, günlük hayatın bir parçası haline gelmiş ve önümüzdeki dönemlerde daha da yaygınlaşacağı öngörülmektedir. Bilgisayar oyunları alanı ise yapay zekânın yaygın olarak kullanıldığı, büyük bir pazar payına sahip bir endüstridir. Oyunlar içerisinde yapay zekâ kullanım alanı çok geniştir. Geliştirilen yapay zekâ oyuna ve gerçekleştirilmesi istenen duruma göre değişiklik göstermektedir. Oyuncuların kişisel düşüncelerine ve davranışlarına göre, oyunu oyuncular için daha çok kişiselleştirmek amacıyla yapay zekâ kullanılabilmektedir [13]. 2016 yılında Silver ve ark. [14] tarafından geliştirilen GO isimli oyunda yapay sinir ağları kullanımıyla; oyunu ustalıkla oynayacak bir yapay zekâ geliştirilmiştir. Diğer bir öğrenme tabanlı oyunda 2001 yılında piyasa çıkmış olan Black and White oyunudur [15]. Oyun içerisinde, kullanıcının yönettiği bir varlık bulunmaktadır. Kullanıcı bu varlığın eğitiminden sorumlu olmakta ve varlık oyuncu tarafından kendine gösterilen davranışlara ve verilen komutlara göre yetişmektedir. Varlığın etrafındaki objelere ve insanlara karşı tutumu da yetiştirildiği tutuma bağlı olmaktadır. Örneğin, oyuncu bu varlığa sürekli insanlara saldırma komutu verirse, varlık insanlara saldırmanın yapılması gereken bir davranış olduğunu öğrenmekte ve bu davranışı sürekli uygulamaktadır. Bu tasarım karar ağaçları ve yapay sinir ağları ile oluşturulmuştur. Karar ağaçları, varlığın etrafındaki objelere karşı tutumunu sergilemesinde, yapay sinir ağları ise varlığın arzu ve isteklerini yönetmek ve karşılamak için kullanmıştır.

Oyunlar içerisinde yapay zekâ kullanımıyla çözülen bir diğer problemse yol bulma problemidir. Yol bulma problemiyle özellikle gerçek zamanlı strateji oyunlarında karşılaşılmaktadır [16]. Bu sorunun çözülmesi, oyun içerisindeki karakterlerin mantıklı bir şekilde hareket ettirilmesini sağlamaktadır. Yol bulmayı sağlamak amacıyla kullanılan çeşitli algoritmalar bulunmaktadır [7]. Kullanılan algoritmaların hesaplama süreçlerinde işlemci üzerine çok fazla yük binmektedir. Bu yükü indirgemek için algoritmanın çalışacağı alan veya alan içerisindeki noktalar azaltılması [17] gibi teknikler uygulanmaktadır. Bu algoritmalar arasında oyun endüstrisinde en yaygın olarak kullanılan algoritma, özel bir dinamik programlama durumuna sahip olan A* yol bulma algoritmasıdır [18].

Yapılan bir çalışmada çeşitli algoritmaların karşılaştırılması gerçekleştirilmiştir [19]. Çalışma sonucu A* algoritmasının kullanılabilecek daha uygun bir algoritma olduğuna sonucuna varılmıştır. Aynı zamanda yol bulmak için kullanılacak olan A* algoritmasının oyunun yapısına göre değişiklik gösterebileceği de belirtilmiştir.

Popüler gerçek zamanlı strateji oyunlarından biri olan StarCraft için yapılan çalışmada; oyun alanı dinamik bir yapıda olduğu için sadece A* algoritmasını kullanmak yerine melez bir yaklaşım kullanılmıştır [20]. Düşman askeri veya düşman binaları görülmedikçe A* algoritmasıyla ilerlenecek olan uygun yol bulunmuş, aksi takdirde Boids algoritması [21] ile askerlerin sürü hâlinde ilerlemesi ve saldırı yapması sağlanmıştır. Bilgisayar tarafından kontrol edilen oyuncunun yenme yüzdesi, melez olmayan yaklaşımlara göre çok az farkla daha iyi olduğu gözlenmiştir. Ancak istatistiksel olarak yüksek bir fark olmadığı için iki farklı yöntemi kullanan oyuncuların aynı kazanma yüzdesine sahip olacağı kararına varılmıştır. İki farklı yöntem performans açısından karşılaştırıldığında, melez yaklaşımın melez olmayan yaklaşıma göre 100 kat daha hızlı çalıştığı belirtilmiştir. Yapılan başka bir çalışmada, bir yarış oyunundaki araçların gidecekleri yolu en iyi şekilde bulabilmesi için A* algoritmasının varyasyonu kullanılmıştır. Geliştirilen algoritma ile araçların rastgele engellerden kaçınabilmesine olanak sağlanmıştır [22]. Bu algoritmaların statik pist içerisinde işlemciye fazla yük bindirmeden etkili bir şekilde yol bulma problemini çözdüğü görülmüştür.

Yapılan bir çalışmada mobil cihazlardaki ana bellek veya işlemci yetersizliğinden dolayı, A* algoritmasının işleyişi değiştirilerek algoritmanın performansının arttırılması hedeflenmiştir [23]. Yol bulma problemi, tüm alanın kapsanıp hesaplanmasının yerine, alan bölünerek gerektiği zaman çözülmüş ve bu şekilde performans arttırılmıştır. Ancak, problem bir bütün olarak ele alınmadığı için, algoritma ile bulunan yolların tam olmadığı ve yeterliliğin düştüğü belirtilmiştir.

3. Motivasyon

Bu çalışmanın amacı; bilgisayar oyunları alanında; çok kullanıcılı mobil yapay zekâ uygulaması geliştirilmesidir. Yapılacak çalışma ile oyunlar içerisindeki karakterlerin istenilen hedefe ulaşmada en uygun yolu bulabilmeleri istenmektedir. Karakterlere bu hedef doğrultusunda akıllı bir şekilde yön bulabilmeleri ve oyun içerisine konulan farklı engellere karşı yön bulabilmeleri planlanmıştır. Karakterlere verilecek farklı özelikler ile de bu yön bulma algoritmaların çalışma yapısının incelenmesi ve en uygun olan dinamik çözümün bulunması

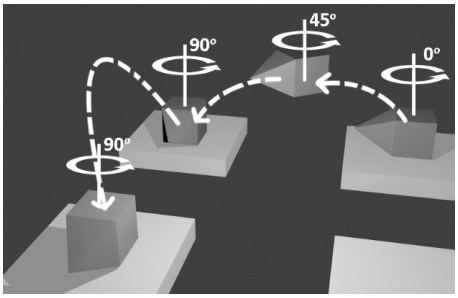
hedeflenmektedir. Yapa zekâ da dış dünyayı izleyerek karar verebilen sistemler için Etmen'ler kullanılmaktadır [8]. Karakterlere akıllı özelliği verilmesi için yapay zekâ etmenlerinden uygun olan çözümler incelenerek ve geliştirilen platforma yapay zekâ özelliği sağlanacaktır.

4. Materyal ve Metot

4.1 Genel oyun kuralları

Oyun üç boyutlu olarak mobil platformda geliştirilmiştir. Oyun alanında 100 adet rengi değiştirilebilen kareler (10x10), hayvan karakterleri ve birçok toplanabilir obje bulunmaktadır. Oyunun ana teması, dört adet oyuncunun en çok kareyi gezerek, en çok puanı alabilmek için birbirleriyle rekabet etmeleri üzerine kurulmuştur.

Oyun başlangıcında her oyuncuya rastgele bir renk (kırmızı, mavi, yeşil, sarı) verilmekte ve oyuncuların her biri oyun alanının farklı köşesine pozisyonlandırılmaktadır. Oyuncuların oyun alanı içerisindeki hareketleri, X ve Z koordinatları eksenlerinde ilerleme, Y koordinatı ekseninde alçalıp yükselme (zıplama) ve Y koordinatı ekseninde rotasyonu (sağa/sola dönüş) içermektedir. X veya Z koordinatı üzerinde karakterler otomatik olarak zıplayarak ilerler. Karakterlerin oyun alanı içerisindeki ilerlemesi saniyede yaklaşık olarak 2 kare olarak ayarlanmıştır. Kullanıcının yukarı, sağa veya sola dönüş hareketi girdisi vermesiyle yönlendirme hareketleri oluşmaktadır (Şekil 1'de gösterilmektedir). Bir oyuncunun zıpladıktan sonraki iniş noktası bir karedir ve bu kareyi kendine verilmiş olan renge boyar. Başka bir oyuncu tarafından daha önceden boyanmış bir kare, farklı bir oyuncu tarafından tekrar boyanabilmektedir.



Şekil 1. Karakterlerin hareketi

Oyun içerisinde yapay zekânın karar verme sürecini değerlendirmek adına her beş saniyede bir, bir adet güç objesi oluşturulmuştur. Oluşturulan bu obje, objeyi toplayan oyuncuya kısa süreli bir avantaj sağlamaktadır. Güç objesi; toplayan oyuncu hariç herkesi yavaşlatma, toplayan oyuncuyu hızlandırma, toplayan oyuncu hariç herkesi durdurma gibi özelliklerden birine sahiptir. Bu özelliklerinden dolayı, oyunu kazanmak için olabildiğince çok güç objesi toplamak gerekmektedir.

Oyuncuların kullandığı karakterler, üç boyutlu objeler olarak tasarlanmıştır. Toplamda oyun içerisinde 10 adet farklı karakter bulunmaktadır. Her karaktere kendine ait bir özelliği eklenmiştir. Bazı karakterler bu özelliklerini kullanıp oyun alanındaki karelerin üzerine bir obje bırakabilir. Bu objelerle çarpışan oyuncular, kısa süreliğine olumsuz bir etki altında kalırlar (yavaşlama, hareket edememe gibi). Şekil 2 de geliştirilen oyundan bir görüntü verilmiştir. Verilen görüntüde farklı karakterler hareket ettikleri platform içerisindeki kareleri kendi renkleri ile işaretlemektedirler. Oyun alanı içerisinde tüm alanlar işaretlendiğinde toplam renk sayısı sayılarak en fazla olan oyuncu oyunu kazanmaktadır.

Şekil 2. Oyundan bir görüntü

4.2 Yöntemler

Geliştirilen uygulamada yapay zekâ için temel mantık, karelerin renklerini ve oyun içi objelerin pozisyonlarını kullanarak oyuncular için bir yol veya hareket yönü hesaplanması üzerine kurulmaktadır. Hesaplamaların içerisine ayrıca karakterin Y ekseni açısının (radyan cinsinden) katılması gereklidir. Oyuncunun mevcut bulunduğu pozisyondaki açısı *(aci)* değişkeni olarak tanımlanmaştır. Bunun yanında, hesaplamaların kolaylaşması adına, dört dik açıyı (0, π/2, π, 3π/2) tutacak olan açı indeksi (*aciIndex*) tutulmuştur. Karakterlerin başlangıç açıları, açı indeksine Denklem 1 kullanılarak dönüştürülür. Başlangıç açı indeksi hesaplandıktan sonra, sola dönme komutları açı indeksini bir azaltır ve sağa dönme komutları açı indeksini bir artırmaktadır.

$$aciIndex = aci \,/\, (\pi/2) \qquad (1)$$

Yapa zekâ da dış dünyayı izleyerek karar verebilen sistemler için Etmen'ler kullanılmaktadır. Bu yapıların temelinde insana benzer davranmak ya da karar verme sürecinde yardımcı olmak yatmaktadır. Etmenler temel olarak 4 kategori altında değerlendirilmektedir. Bunlar; Basit Refleks Etmenleri (Simplex Reflex Agents), Model Tabanlı Refleks Etmenleri (Model- based Reflex Agents), Hedef Tabanlı Etmenler (Goal-based Agents), Fayda Tabanlı Etmenler (Utility-based Agents) 'dir [8]. BRE, sahip olduğu kurallardan, bulunduğu duruma ait olan kurala ait eylemi gerçekleştirir. Model tabanlı refleks etmenleri bulunduğu alanı ve bu alanın nasıl geliştiğine ve yapacağı eylemlerin bu alan içerisinde yapacağı etkiye (model) dair bilgiler dahilinde, BRE gibi bulunduğu duruma ait olan kurala ait eylemi gerçekleştirmektedir. HTE, sahip olduğu hedefe gitmek için bir arama ve planlama yaparak, hedefe ulaşmak için gerekli olan eylem dizisini bulurlar. Fayda tabanlı etmenler, HTE'de olduğu gibi bir hedefe sahiptir ve o hedefe ulaşabilmek için arama ve planlama yapar. Fayda tabanlı etmenlerin farklılığı ise, ajanların hedefe giderken yapacağı eylemlerden en fazla fayda sağlayanı uygulamaktır [8].

Çalışmamız kapsamında geliştirdiğimiz oyuna en uygun olan BRE ve HTE yapıları kullanarak incelemeler ve testler yapılmıştır.

4.2.1 Basit refleks etmeni (BRE)

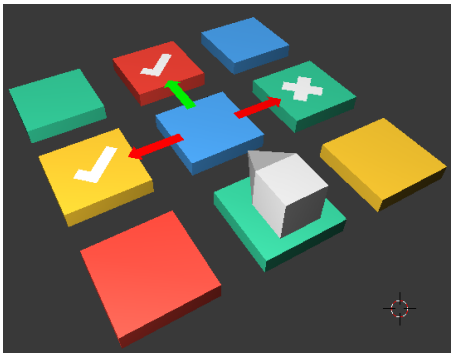
Bir basit refleks etmeni, bulunduğu duruma uyan bir kural bulur ve o kurala uyan koşula göre eylemi gerçekleştirir [8].

Çalışmamızda BRE yapısında toplamda üç adet yöntem denenmiştir. Her yöntem, bir önceki yöntemin üzerine eklenerek geliştirilmiştir.

4.2.1.1 BRE – yöntem #1

Bu yöntemde, oyuncu sadece ön tarafında bulunan kareye komşu olan kareleri kontrol eder ve rengi kullanıcının renginden farklı olan kareye doğru hareket eder. Eğer koşulu sağlayan birden fazla kare varsa, hareket yönü rastgele seçilmektedir (Şekil 3'de gösterilmektedir).

Hareket yönünün belirlenmesi karakterin önündeki kareye bağlı olduğu için, karakterin açı indeksine göre belirlenmektedir.



Şekil 3. Basit Refleks Etmeni #1 – hareket yönü kararı

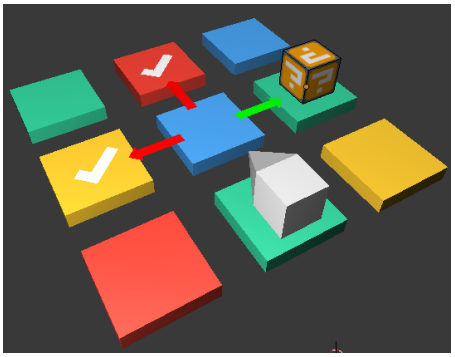
Örneğin, karakterin açı indeksi sıfırken önünde olan kare, açı indeksi bir olduğunda sağında olur. Bu sebepten ötürü, sol/sağ/ön karelerin renklerinin kontrol edilmesi açı indeksine göre değişim göstermektedir. Kare renklerinin açı indeksine göre kontrol edilmesi aşağıdaki gibi yapılmaktadır:

1. **if** aciIndeksi **is equal to** 0 **then**
2. RenkKontrol (SOL_UST, UST, SAG_UST)
3. **else if** aciIndeksi **is equal to** 1 **then**
4. RenkKontrol (SOL_ALT, SOL, SOL_UST)
5. **else if** aciIndeksi **is equal to** 2 **then**
6. RenkKontrol (SAG_ALT, ALT, SOL_ALT)
7. **else if** aciIndeksi **is equal to** 3 **then**
8. RenkKontrol (SAG_UST, SAG, SAG_ALT)

RenkKontrol fonksiyonu karakterin rengi ile karenin rengi arasındaki eşitliği döndürür. Eğer eşit değillerse, karenin indeksi bir listeye eklenir. Üç karenin renginin RenkKontrol fonksiyonu ile kontrol edilmesinden sonra, rastgele bir kare seçilir ve bu karenin indeksi, oyuncuya hedef olarak verilir.

4.2.1.2 BRE – yöntem #2

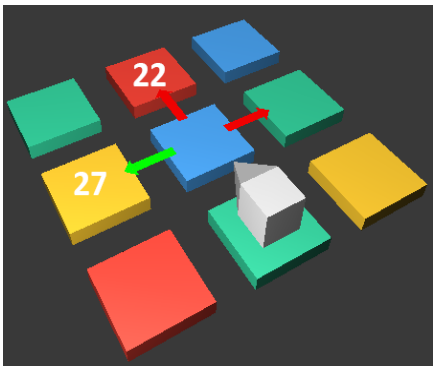
Bu yöntemde, birinci yönteme ek olarak oyun alanı içerisindeki güç objelerinin kontrolü eklenmiştir. Eğer kontrol edilen karelerin herhangi birinde güç objesi bulunuyorsa, oyuncu o kareye doğru hareket eder. Aksi takdirde, oyuncu kendi renginden farklı olan kareye doğru hareket eder (Şekil 4'te gösterilmektedir).



Şekil 4. Basit Refleks Etmeni #2 – hareket yönü kararı

4.2.1.3 BRE – yöntem #3

Bu yöntemde, birden fazla farklı renkli kare bulunduğunda, oyuncuyu rastgele hareket ettirmek yerine, o kareye sahip olan kullanıcıların skorlarını karşılaştırıp, en yüksek skora sahip olan karakterin yönüne hareket ettirmektir (Şekil 5'te gösterilmektedir). Bu yöntemde, Yöntem #2'de olduğu gibi güç objesi kontrolü de yapılmaktadır.



Şekil 5. Basit Refleks Etmeni #3 – hareket yönü kararı

4.2.2 Hedef tabanlı etmen (HTE)

HTE, Hedefe ulaşmak için ortamı algılayarak, hedefe en uygun şekilde ulaşmayı amaçlar. HTE tasarımlarında, ajan çevredeki eylemlerini açıkça

düşünmez; sadece önceden hesaplandığı şekilde eylemi gerçekleştirir. Fakat HTE'ler gelecekteki eylemlerinin nasıl sonuçlanacağını düşünebilir [8].

Bilgisayar tarafından kontrol edilen oyuncunun hedefi; olumsuz etkiye sahip olan objelere çarpmadan, en iyi oyuncuların skorunu düşüreceği, en iyi yolu bulmak için bir arama gerçekleştirmektir. Bu aramayı gerçekleştirmek için, Manhattan Turist Problemini çözmek için kullanılan dinamik programlama algoritması kullanılmıştır.

Manhattan Turist Problemi [9] şu şekilde tanımlanabilir: Manhattan'da gezi turu vardır. Burada çok fazla sayıda turistik yer bulunmaktadır ve turistler olabildiğince çok yeri görmek istemektedirler. Turistler yalnızca güneye veya doğuya hareket edebilirler ve farklı yollardan istediklerini seçebilirler. Problem içinde belirtilen her blok, içerisindeki turistik yer sayısını göstermektedir. Manhattan Turist Problemi, en çok turistik yere sahip olan yolu bulmayı hedefler.

Yapılan çalışma içerisinde, Manhattan Turist Problemi içerisindeki bloklar karelere ve turistik yer sayısı karelerin skoruna karşılık gelmektedir.

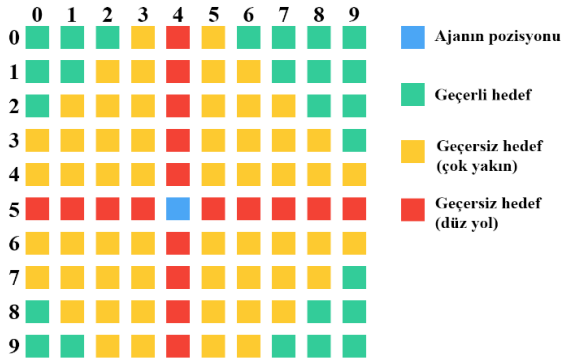
Hedef noktası yarı rastgele bir şekilde seçilir. Hedef noktası seçiminde bazı kısıtlamalar bulunmaktadır. Şekil 6'da bu kısıtlamalar gösterilmektedir. Bu kısıtlamaların amacı, olumsuz etkiye sahip olan oyun için objelerden kaçınmak ve oyuncunun düz bir çizgi üzerinde yol almasını engellemektir.

Bir hedef noktası seçildikten sonra, n × m boyutlarına sahip olan bir matris, en uygun yolu aramaya başlamak amacıyla yaratılır. n ve m değerleri Denklem 2 ve Denklem 3 kullanılarak hesaplanır.

$$n = \left|\frac{Poz\_Indeksi - hedef}{10}\right| + 1 \qquad (2)$$

$$m = |(Poz\_Indeksi - hedef)\ \%\ 10| \qquad (3)$$

İlk bakışta, tek bir problem ($S_{n-1,m-1}$'in hesaplanması), yerine n × m adet farklı problemin ($S_{i,j}$'nin $0 \le i \le n-1$ ve $0 \le j \le m-1$ değerleri için hesaplanması) yaratıldığı görünebilir, fakat genel problemin çözülmesi, Manhattan Turist Probleminin çözülmesi kadar kolay olması, dinamik programlamanın temelidir [9].



Şekil 6. Hedef noktası seçim kısıtları

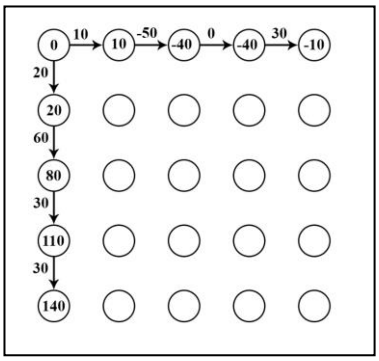
Bir karenin skoru, üzerinde bulunan objelere ve hesaplayan oyuncu ile kareye sahip olan oyuncu arasındaki skor farkına bağlıdır. Oyuncuların skorları ve karelerin üzerinde objeler gerçek zamanlı olarak değişim içerisinde olduğu için, bir karenin skorunun hesaplanması, $S_{i,j}$'nin

bulunması sırasında gerçekleşir. Karelerin skorları aşağıdaki verilen Algoritma 4.1 deki algoritma ile hesaplanmaktadır.

Algoritma 4.1

```
1. KareSkoru(int i, int j)
2. if kare[i,j] has olumsuzObje on it
   then
3.       kareSkoru -= 200
4. if ajan.renk is not equal to kare
   [i,j].renk then
5.       int kareSkoru =
   oyuncu[tile[i,j].color].skor - ajan.skor
6. if skorFarki is greater than 0 then
7. kareSkoru += skorFarki * skorFarki
8. else
9.       kareSkoru -= 5
10. else
11.      kareSkoru -= 50
12. return kareSkoru
```

Algoritma 4.1'e göre $S_{0,j}$'nin skoru ($0 \leq j \leq m-1$ değerleri için) için soldan sağa ilerleyerek ilk j karenin skorlarını toplanmasıyla hesaplanır. Benzer şekilde, $S_{i,0}$'ın skoru ($0 \leq i \leq n-1$ değerleri için) yukarıdan aşağı ilerleyerek hesaplanır. Hesaplama Şekil 7'de gösterilmiştir.

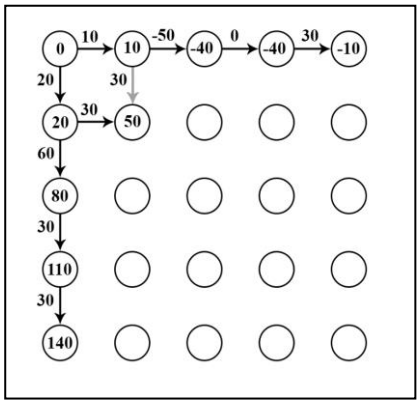


Şekil 7. $S_{0,j}$ ve $S_{i,0}$'ın değerlerinin hesaplanması

Şekil 7, 8, 9 ve 10 kare skorlarının farklılığından dolayı Neil & Pavel (2004) içerisindeki şekillerin tekrar çizilmesiyle oluşturulmuştur.

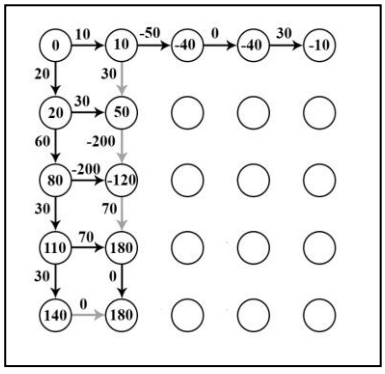
$S_{0,1}$ ve $S_{1,0}$'ın hesaplanmasından sonra, soldan sağa $S_{1,0}$'den veya yukarıdan aşağı $S_{0,1}$'den gelinerek $S_{1,1}$ hesaplanabilir. Oyuncu kendine en yararlı olan yolu seçmesi gerektiği için, en yüksek skora sahip olan yolu seçecektir (Şekil 8'de görülmektedir). Soldan sağa ve yukarıdan aşağı gelen yolların skorları aşağıdaki şekilde hesaplanmaktadır:

- $S_{0,1}$ + KareSkoru(1,1)
- $S_{1,0}$ + KareSkoru(1,1)



Şekil 8. $S_{1,1}$'in değerinin hesaplanması

Aynı mantık kullanılarak, $S_{2,1}$, $S_{3,1}$, $S_{4,1}$, $S_{5,1}$ ve geri kalan elemanların değerleri Şekil 9 ve Şekil 10 da gösterilmiştir.



Şekil 9. $S_{2,1}$, $S_{3,1}$, $S_{4,1}$ ve $S_{5,1}$'in değerlerinin hesaplanması

Daha sonra, oyuncunun kullanabilmesi için, hedeften başlangıç yerine geri gelerek ($S_{n-1,m-1}$) her eleman bir yığın

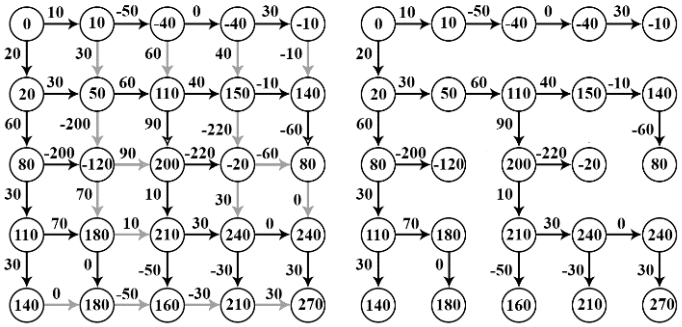
(stack) veri yapısına eklenir. Örneğin, Şekil 10'dan üretilen yığın yapısının içeriği S0,0 - S1,0 - S1,1 – S1,2 - S2,2 - S3,2 - S3,3 – S3,4 - S4,4 olur. Eğer birden fazla yol çözüm sağlıyorsa, bu yollardan bir tanesi rastgele seçilir. Oyuncu, bir sonraki hareketi için yığından bir eleman çıkarır ve o elemanın değerine göre sağa ya da sola döner veya düz gitmeye devam eder. Oyuncu yığından her eleman çıkarttığında, oyun alanı gerçek zamanda dinamik olarak değiştiği için yeni bir yol daha hesaplar. Eğer yeni hesaplanan yolun skoru daha büyükse, oyuncu o yolu takip eder; değilse eski hesapladığı yoldan ilerlemeye devam eder. Oyun alanında yeni bir güç objesi yaratıldığında, bilgisayar tarafından kontrol edilen üç oyuncudan en yakın oyuncu güç objesinin bulunduğu pozisyonu kendine hedef olarak alır. Bu sayede, hiçbir oyuncu güç objesini alabilmek için birbiriyle çatışmaz ve aynı kareler üzerinde sürekli hareket etmekten oluşacak skor kaybından da kaçınmış olurlar.

5. Bulgular

Uygulanan algoritmaların değerlendirilmesi için, BTE yöntemleri 20'şer kere, HTE 50 kere test edilmiştir. Bu testler özel bir yeteneğe sahip olmayan karakterler ile ve aynı kişi tarafından gerçekleştirilmiştir. Test eden kişi Bilgisayar Mühendisliği bölümü öğrencilerinden rastgele seçilmiştir. Test edilen oyun içerisinde, test eden kişi dışında, yapay zekâ tarafından hareket ettirilen üç adet oyuncu bulunmaktadır.

Testler içerisinde test eden kişinin; kaybettiği oyunların, tüm oyunlara oranı, başarı kıstası olarak alınmıştır. Test eden kişinin kazanma oranının düşük olması, testin başarılı olduğunu göstermektedir.

Yapılan her testin ardından, dört oyuncunun 100 kare içerisinde boyadıkları kare sayısı, oyun içerisinde toplayabildikleri güç objesi sayısı ve oyunu kimin kazandığı not edilmiştir .



Şekil 10. Yolun bulunması

Sonuçlar Tablo 1, Tablo 2 ve Tablo 3'te gösterilmiştir. Tablolar içerisindeki ortalama skor, oyuncunun her oyunda ortalama kaç kare boyadığı; ortalama güç objesi, oyuncunun her oyunda ortalama kaç adet güç objesi topladığı gösterilmektedir. Tabloda belirtilen kazanma yüzdesi ise oyuncunun kazandığı oyunların toplam oyun sayısına oranını göstermektedir.

5.1 Basit refleks etmeni değerlendirme

Yapılan testlerin sonunda, BRE yöntemlerinin her birinin hız açısından iyi performans göstermiş oldukları gözlenmiştir.

BRE-Yöntem#1 ile yapılan testler sırasında, oyuncuları sadece önündeki kareye bitişik olan karelerin renklerine göre hareket ettirmenin yetersiz olduğu görülmüştür. Bu yöntem ile kendi karelerine basarak puan alamama ihtimallerinin, tamamen rastgele hareket etmeye göre düşük olduğu, fakat oyun alanının küçük bir bölgesini gözlemleyip hareket etmekten dolayı, aynı bölgeler içerisinde dolaşıp puan alamadığı durumlar gözlenmiştir. Ayrıca, oyun alanındaki güç objelerinden tamamen habersiz oldukları için, ancak şans eseri güç objelerini toplayabilmişlerdir ve bu da test eden kişinin yapay zekâ ile hareket ettirilen oyunculara göre çok avantajlı konuma geçtiği görülmüştür. Bu sebepten dolayı, ikinci yöntemin geliştirilmesinde, birinci yöntemin üstüne güç objesi ile ilgili kontrollerin eklenmesi düşünülmüştür. Bu sebeplerden dolayı, Tablo 1'de görülebileceği gibi yapay zekâ tarafından kontrol edilen oyuncular kötü performans göstermiş olup, toplam oyunların ancak %20'sini kazanabilmişlerdir.

İkinci uygulanan yöntemde (BRE-Yöntem #2), birinci uygulanan yöntemdeki gibi aynı bölgeler içerisinde dolaşıp puan alamadığı durumlar gözlenmiştir. İkinci yöntemde etrafındaki karelerde güç objesi olup olmadığının kontrolünü yapıldığı için, güç objesi toplama oranı, Tablo 2'de görülebileceği gibi birinci yönteme oranla artmıştır. Fakat hâlâ rastgele dolaştığı için, güç objesi toplama oranı çok düşüktür. Güç objesi oyun alanının bir ucunda, test eden kişi diğer ucunda olduğu durumlarda dahi, yapay zekâ oyuncuları güç objesini alana kadar, test eden kişinin o güç objesini alabildiği gözlenmiştir. Yapay zekâ oyuncularının kazanma oranları, birinci yönteme göre %15 artmıştır. Bu artışın, oyuncuların artık daha fazla güç objesi toplayabilmelerinden dolayı olduğu düşünülmektedir.

Üçüncü yöntemin geliştirilmesinde (BRE-Yöntem#3); kullanıcının yapay zekâ oyuncularına göre daha iyi oynayıp, daha yüksek skorlara sahip olacağını düşünülüp, kullanıcının puanını aşağı çekmek adına oyuncuların puanlarına göre hareket sisteminin geliştirilmesi planlanmıştır. Bu yöntem için yapılan testlerin sonunda, yapay zekâ oyuncularının yenme yüzdesi ikinci yönteme oranla %15 artmıştır ve Tablo 3'te görülebileceği gibi, düşünülen yöntemin kullanıcının skorunu düşürmede etkili olduğu görülmüştür.

Üçüncü yöntemin başarı oranı istenilen değerlerde olmasına rağmen, denenen tüm yöntemlerde kullanılan oyuncuların bitişik karelere göre hareket etmesi yönteminden dolayı oluşan rastgele hareket edişin çok yapay olduğu (insansı olmadığı) gözlenmiştir ve kullanıcılara yeterli zorluğu sağlayamayacakları düşünülmüştür. Bu yüzden, sadece bitişik olan karelere göre hareket etmenin, problemi çözemeyeceğine karar verilmiştir. Problemi çözmek için oyuncuların bitişik karelerden fazla kareyi gözlemleyebileceği ve buna göre hareket edebileceği bir yöntem kullanmanın gereği görülmüştür.

Tablo 1. Basit Refleks Etmeni #1 Sonucu (YZ: Yapay Zekâ)

	User	**YZ #1**	**YZ #2**	**YZ #3**
Ortalama Skor	37,35	23	17,85	21,8
Ortalama Güç Objesi	8,35	0,75	0,75	0,8
Kazanma Yüzdesi	80%	15%	0%	5%

Tablo 2. Basit Refleks Etmeni #2 Sonucu (YZ: Yapay Zekâ)

	User	YZ #1	YZ #2	YZ #3
Ortalama Skor	34,45	21,75	20	23,75
Ortalama Güç Objesi	7,8	1,2	1,2	0,8
Kazanma Yüzdesi	65%	10%	5%	20%

Tablo 3. Basit Refleks Etmeni #3 Sonucu (YZ: Yapay Zekâ)

	User	YZ #1	YZ #2	YZ #3
Ortalama Skor	30,5	22,95	22,9	23,4
Ortalama Güç Objesi	6,4	1,55	1,55	1,4
Kazanma Yüzdesi	50%	10%	20%	20%

5.2 Hedef tabanlı etmen değerlendirme

Yapılan testlerin sonucunda HTE, BRE kadar hızlı olmasa da, oyunun performansını düşürmediği gözlenmiştir. Hedef noktasının uzaklaştırılmasıyla, oyuncular olumsuz etkiye sahip olan objelerden olabildiğince kaçınmış ve güç objesi toplama miktarı Tablo 4'te görülebileceği gibi, en iyi basit refleks ajanına göre yaklaşık olarak %40 oranında artmıştır. Tablo 4'te görülebileceği gibi, yapay zekâ ile hareket ettirilen oyuncuların ortalama skorları, kullanıcının ortalama skoruna çok yakındır. Bu sonuç göze alındığında, yol bulma algoritmasının basit refleks yapılarındaki rastgele harekete göre çok daha iyi sonuç verdiği söylenebilir. Algoritma içerisinde kullanılan kare skorlarının belirlenmesinde yapılan değişikliklerle (kullanıcının rengi sahip olan karelerin sağladığı puanı arttırıp, azaltarak), kazanma yüzdesinin kolayca arttırılıp, azaltılabildiği görülmüş; fakat kullanıcılara yeterli zorluğa sahip olan bir yapay zekâ sunulması istendiğinden, yapay zekânın kazanma yüzdesi bu oranda bırakılmıştır.

Bu yöntem ile hareket ettirilen oyuncular, rastgele hareket etmeyip, belirli bir yolu izlediklerinden dolayı daha doğal bir şekilde hareket etmişlerdir ve bu sayede kullanıcılara gerçek oyunculara karşı oynadığı hissini daha iyi vereceği düşünülmüştür.

Tablo 4. Hedef Tabanlı Etmen Sonucu (YZ: Yapay Zekâ)

	User	YZ #1	YZ #2	YZ #3
Ortalama Skor	27,12	23,54	23,44	24,8
Ortalama Güç Objesi	3,14	2,4	1,72	2,14
Kazanma Yüzdesi	44%	14%	18%	24%

6. Tartışma ve Sonuç

Bu çalışma ile yapay zekâ etmenlerinin denendiği mobil platformda, offline olarak da çalışabilecek çoklu oyunculu geliştirilen oyun sunulmuştur. Geliştirilen oyunda yapay oyuncuların mantıklı bir şekilde hareketini sağlamak için özel bir yapay zekâ metodu geliştirilmiştir. Önce BRE yöntemine dayanarak geliştirilen yapay zekâ test edilmiş ancak istenen performans alınamamıştır. Bunun üzerine, diğer bir yöntem olarak HTE geliştirilmiştir. HTE yaptığı hesaplamaların daha kompleks ve maliyetli olmasına karşın, oyun içerisinde gösterdiği başarıdan dolayı, BRE'ye olan üstünlüğü açıkça görülmüştür. Basit Refleks Etmeni #1 ile başlanan testlerde kullanıcıların %80'lerde olan kazanma oranı Hedef Tabanlı Etmen de % 44'lere gerilemiştir. Bunun yanında yapay zekâ oyuncuların kazanma oranlarının da %24'lere çıktığı görülmektedir. BRE oyun alanı içerisindeki hareketlerin gerçekçilikten uzak olduğu görülmüştür. HTE ile geliştirilen yapay oyuncuların hedef odaklı hareket etmelerinden dolayı, hareketlerinin daha insan

reaksiyonlarına benzer olduğu görülmüştür. Çalışmamızın ileriki aşamalarda oyuncuların kişisel özelliklerine göre hareketleri ve eylemleri incelenecektir. Öğrenme etmenleri ve veri madenciliği algoritmaları kullanılarak oyuncuların analizlerinin yapılması ve oyuncuların karşısına daha yetenekli yapay oyuncuların çıkarılması planlanmaktadır.

Kaynakça

[1] Strong, A.I. 2016. Applications of Artificial Intelligence & Associated Technologies, Proceeding of International Conference on Emerging Technologies in Engineering, Biomedical, Management and Science Jodhpur, India, ss 64-67.

[2] Nilsson, N.J. 2014. Principles of artificial intelligence, Morgan Kaufmann Publishers, 0-934613-10-9, California, A.B.D.

[3] Johnson, S. 2002. Wild Things, http://www.wired.com/2002 /03/aigames/ (Erişim Tarihi: 20.06.2016).

[4] Schölkopf, B. 2015. Artificial intelligence: Learning to see and act, Nature, 518(7540), s. 486-487.

[5] Millington, I., Funge, J. 2016. Artificial intelligence for games. CRC Press, 978-0-12-374731-0, Burlington, A.B.D.

[6] Alex, J. 2007. Top 10 Most Influential AI Games, http://aigamedev.com/open/highlights/top-ai-games/ (Erişim Tarihi: 26.06.2016).

[7] Stout, B. 1996. Smart moves: Intelligent pathfinding, Game developer magazine, Gamasutra.

[8] Russell, S., Norvig, P. 1996. Artificial Intelligence-A Modern Approach, Prentice Hall, Englewood Cliffs, New Jersey, A.B.D.

[9] Jones, N.C., Pevzner, P. 2004. An Introduction to Bioinformatics Algorithms, 0-262-10106-8, MIT press, A.B.D.

[10] Rabiner, L., Juang, B.H. 1993. Fundamentals of speech recognition, PTR Prentice Hall Inc., 0-13-285826-6, New Jersey, A.B.D.

[11] Xu, L., Krzyzak, A., Suen, C.Y. 1992. Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Transactions on Systems, Man, and Cybernetics, 22(3), s. 418-435.

[12] Javidi, B., Horner, J.L. 1994. Optical pattern recognition for validation and security verification, Optical engineering, 33(6), s. 1752-1756.

[13] Farooq, S.S., Kim, K.J. 2016. Game player modeling, Encyclopedia of Computer Graphics and Games, Springer International Publishing, s. 1-15.

[14] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Dieleman, S. 2016. Mastering the game of Go with deep neural networks and tree search, Nature, 529(7587), s. 484-489.

[15] Wexler, J. 2002. Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's "Black and White and where it can go and will go in the future, Doktora Tezi, University of Rochester.

[16] Buro, M., Furtak, T. 2004. RTS games and real-time AI research, In Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS), Arlington, Virginia, A.B.D.

[17] Graham, R., McCabe, H., Sheridan, S. 2015. Pathfinding in computer games, The ITB Journal, 4(2), 6.

[18] Ferguson, D., Likhachev, M., Stentz, A. 2005. A guide to heuristic-based path planning, 15th International Conference on Automated Planning & Scheduling, Monterey, California, A.B.D., 510.

[19] Anguelov, B. 2011. Video game pathfinding and improvements to discrete search on grid-based maps, Doktora Tezi, University of Pretoria.

[20] Hagelback, J. 2015. Hybrid pathfinding in StarCraft, IEEE Transactions on Computational Intelligence and AI in Games, 99, s. 1-10.

[21] Reynolds, C.W. 1987. Flocks, herds and schools: A distributed behavioral model, ACM SIGGRAPH Computer Graphics, 21(4), s. 25-34.

[22] Wang, J.Y., Lin, Y.B. 2012. Game ai: Simulating car racing game by applying pathfinding algorithms, International Journal of Machine Learning and Computing, 2(1), 13.

[23] Fayyazi, M., Ghazvini, F.F., Ramzi, A. 2014. Efficient Grid-Based Path Finding Techniques For Android Games Environments, Austrian E-Journals of Universal Scientific Organization, 4(12), s. 1072-1094.