T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAPAY ZEKÂ İLE OYUN SEVİYELEME

YÜKSEK LİSANS TEZİ

YUNUS SARICA

DENİZLİ, AĞUSTOS - 2019

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAPAY ZEKÂ İLE OYUN SEVİYELEME

YÜKSEK LİSANS TEZİ

YUNUS SARICA

DENİZLİ, AĞUSTOS - 2019

KABUL VE ONAY SAYFASI

Yunus SARICA tarafından hazırlanan "**Yapay Zekâ İle Oyun Seviyeleme**" adlı tez çalışmasının savunma sınavı 26.08.2019 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Dr. Öğr. Üyesi Meriç ÇETİN	.....
Üye Prof. Dr. Sezai TOKAT	.....
Üye Doç. Dr. Emre ÇOMAK	.....

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 04/09/2019 tarih ve ....35/15... sayılı kararıyla onaylanmıştır..

.....

Prof. Dr. Uğur YÜCEL

Fen Bilimleri Enstitüsü Müdürü

Bu tez çalışması Pamukkale Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi (PAUBAP) tarafından 2018FEBE003 nolu proje ile desteklenmiştir.

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

Yunus SARICA

ÖZET

YAPAY ZEKÂ İLE OYUN SEVİYELEME
YÜKSEK LİSANS TEZİ
YUNUS SARICA
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
(TEZ DANIŞMANI:DR. ÖĞR. Ü. MERİÇ ÇETİN)

DENİZLİ, AĞUSTOS - 2019

Oyun geliştirim süreci, her geçen gün daha ayrıntılı bir yapıya sahne olmaktadır. Kapsamlı bir bilişim teknolojisi olan yapay zekâ uygulamaları, oyun teknolojilerini de yakından etkilemiştir. Her geçen gün artan etkileyici çalışmaların başında gelmektedir. Bu çalışmada ise 2 boyutlu bir platform oyununun seviyeleme işlemleri incelenmiştir. Renga, çalışmada kullanılan ve tez için tasarlanmış bir oyun olup, basit bir oynanışa sahiptir. Oyun verileri, elde edilen oynanış ve kullanıcı bilgileri ile yapay sinir ağları modeli, en yakın k-komşu, rassal orman ve derin öğrenme yöntemleri ile işlenmiştir. Çıkan sonuçlar, yapılan sınıflama işlemi, bir sonraki oyun bölümüne yönelik, seviyeleme için kullanılabilecek sonuçlar sunmaktadır. Böylelikle geliştiricilerin, oyun kullanıcıları için, oyuna göre sunabileceği en verimli oynanabilirlik algısı şekillenecektir. Bir oyunu oynanabilir kılan en önemli özellik ise zorluktur. Her kullanıcının farklı oyun becerilerine sahip olması, her oyuncuya farklı zorluk sunumunun bir gereklilik olduğunu ortaya çıkaracaktır. Bu çalışma dinamik verilerle hesap edilen zorluğun, genel bir kullanıcı kitlesine göre şekillenebilirliğini Renga üzerinde sağlamaktır. Bu sayede yeni bölümler, varlıkların, bu kullanıcılara özel değerleriyle oluşturulacaktır. Kullanıcıya en verimli oyun tecrübesi aktarılmış olacaktır.

ANAHTAR KELİMELER: Yapay Zekâ, Oyun Programlama, Bölüm Zorluğu, İçerik Üretimi, Zorluk Tespiti, Makine Öğrenmesi.

ANAHTAR KELİMELER:

ABSTRACT

GAME LEVELLING WITH ARTIFICIAL INTELLEGENCE
MSC THESIS
YUNUS SARICA
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
COMPUTER ENGİNEERİNG
(SUPERVISOR: ASSİS. PROF. DR. MERİÇ ÇETİN)

DENİZLİ, AUGUST 2019

The game development process is becoming a more detailed structure every day. The applications of artificial intelligence (AI), which is a comprehensive information technology, have been closely related to game technologies. In this project, the levelling process of a 2-dimensional (2D) platform game was investigated. The game called “Renga” has a basic gameplay and had developed for this work. Game data was processed through an artificial neural network (ANN), k nearest neighbour, random forest algorithms and deep learning model that is trained with gameplay and user information. The classification process with the output data provides results for the next game section and level. In this way, the most effective playability impression that the developers offer to the game users is created according to game. The most important feature that makes a game playable is difficulty. The fact that each user has a different game skill requires each player to have a game with different difficulty. In this project, the variety of difficulty calculated with dynamic data by the user is provided by Renga, in which new sections/levels are created with user-specific assets. Thus, the most efficient gaming experience has been transfered to the users.

KEYWORDS:Artificial Intelligence, Game Programming, Level Difficulty, Content Generation, Measuring Difficulty, Machine Learning.

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

KISALTMALAR LİSTESİ

PCG : Procedural Content Generation
YSA : Yapay Sinir Ağları
FPS : Frame per Second
k-EYK : En Yakın K Komşu
NPC : Non-player Character

ÖNSÖZ

Video oyunları, günümüzde; bilim, eğlence ve sanat alanlarının buluştuğu, en önemli ortamlardan biri hâline gelmiştir. Her ne kadar eğlence odaklı gibi görünse de oyunların hareket yakalama (motion capture), ışın izleme (ray tracing) gibi zamanın teknolojik gelişmeleri ile bir araya gelmesi, bu alanın yalnızca eğlence odaklı olmadığını gözler önüne sermektedir. Oyunların süregelen kendi hikâyeleri veya kabul görmüş etkileri sayesinde, oyunu edinebilmek için günlerce sırada beklemeyi göz önüne alan kitleler bulunmaktadır. Bu durum oyunların, toplumu yönlendirebilecek bir araç hâline geldiğini kanıtlayabilir niteliktedir.

Böyle bir çalışmada, tasarladığım Renga oyununu kullanmamın en önemli sebebi, ufak adımlarla da olsa büyütmek istediğim hayallerimin gerçekleşmesinde ilk basamağı aşabilmektir. Geri dönüp baktığımda, bu yolu izleyecek meslektaşlarım için bir yol haritası sunmak, yaşayabilecekleri olumsuzları en aza indirmek yine en büyük isteklerimden biridir.

Lisans ve yüksek lisans bilimsel katkıları ile bana destek olup, eğitimim süresince her daim güler yüzle ve sabırla, yardımlarını esirgemeyen değerli hocam, Dr. Öğr. Üyesi Meriç ÇETİN'e çok teşekkür ediyorum. Görüşmelerimizde yardımcı olabilmek için her türlü imkânı sunan bölüm hocalarım Prof. Dr. Sezai TOKAT ve Öğr. Gör. Şevket Umut ÇAKIR'a ve adını sayamadığım, tez süresince beni destekleyen bölümümün değerli öğretim üyelerine teşekkür ederim.

Üniversite hayatım boyunca maddi manevi desteklerini hissettiren ve profesyonel hayatımda emeği bulunan üniversitemiz Tiyatro Topluluğu üyelerine ve tez süresince destek ve yönlendirmeleri için Vasfi TATAROĞLU'ya teşekkür ediyorum.

Çalışmalarım sırasında gösterdiği sabır, anlayış, manevi destek ve çabaları ile beni yetiştirip bugünlere getiren kıymetli aileme ve akrabalarıma, moral kaynağım sevgili kardeşime ve başka deyişle; akademik önderim Doç. Dr. Mustafa SARICA'ya, her anımın vazgeçilmez destekçisi Prof. Dr. Nurten SARICA'ya ve ne olursa olsun yüzümü güldürmeyi başaran kardeşim Bilgenur SARICA'ya teşekkürlerimi sunarım.

1. GİRİŞ

Oyunlar, güncel teknolojide, sürekli olarak karşılık bulabilecek birer uygulama haline gelmişlerdir. Platformlar arasında, kendi hedef kitlesinin algısını açık tutabilecek, farklı türlerde oyunlar her geçen gün artmakta ve gelişmektedir. Bu gelişime en çok ayak uyduran video oyunları; çevre tasarımı ve oynanamaz karakterlerin (NPC) gerçekçi davranışlarını en iyi aktarabilenlerdendir (Sarıca, 2018). Non-player Character isimli bu nesnelerin davranışlarının meydana getirdiği zorluklar ise bir kullanıcının oyundan alacağı keyfi en üst düzeye çıkarmak için yapay zekâ ile desteklenmektedir.

Yakın dönem teknolojilerinin sağladığı, oyun motoru sistemleri geliştirilen oyunların ince ayrıntılarını ön plana çıkarmaktadır. Yapay zekânın oyun alanlarında kullanım kapsamı gün geçtikçe çeşitlenmekte ve derinleşmektedir. Yannanakis ve Togelius, bir seminer boyunca edindiği izlenimleri aktarırken 10 temel alan ortaya koymaktadırlar (Yannakakis ve diğ., 2014);

- Oynanamaz karakterlerin davranış öğrenimi
- Arama ve planlama uygulamaları
- Oyuncu modellemeleri
- Yapay zekâya bağlı oyun yeterlilik uygulamaları
- Yöntemsel içerik üretimi (bu tezde incelenen bir alandır)
- Değişiklik gösterebilen hikâyeler
- Gerçekçi çevre hareketlerinin oluşumu
- Yapay zekâya dayalı oyun tasarımı
- Ticari oyunların kullanımı
- Genel oyun içi yapay zekâ işlemleri

Kullanıcılara sunulabilen ayrıcalıklar kadar geliştiricilerin de zamanını verimli kullanabilmesi her oyun süreci için önem taşımaktadır. Bu çalışmada zorluk algısı sonsuz-sürekli oyunlarda incelenmiş, dengeli bir seviyeye getirilerek, kullanıcı kitlesine özel uyumlu bölümler tasarlanmaya çalışılmıştır.

Bu çalışmanın yapılmasında kararlaştırılan süreç için öncelikli amaçlardan biri platform oyun bölümleri yapılırken geliştiriciye en kısa zamanda yeterli verileri sunarak bölümü tasarlamasına yardımcı olabilmektir. Ortak nesnelerin varlığı üzerinden düşünülecek olursa, platform oyunları genel olarak bir karakter, düşman, etkileşimli nesneler ve çeşitli platform gruplarından meydana gelebilmektedir.

Karakterin amacı bölümün sonuna gelebilmek olsa da kullanıcının karakteri ile yapacağı işlemlerin sonucu, oyun başarımlarında uygun zorluk seviyelerine ulaşması önem teşkil etmektedir. Oyun zorluğunun olağandan az olması kullanıcının sıkılmasına, çok olması ise bıkkınlığa sebep verebilmektedir. Ancak her oyuncu için farklı tutarlılıkta bölümler ortaya çıkabilmektedir.

Bu çalışmada, sonsuz sürekli bir platform oyunu olan Renga'nın, tasarlanması ve işlenmesi ile yeni bir çalışma ve veri kümeleri oluşturulmak istenmiştir. Renga hem kullanımı basit hem de çeşitlenebilecek oyun varlıkları ile kullanıcıyı güncel tutacaktır. Kullanıcılardan gelen verilerin de işlenmesiyle, bulunduğu zorluk katsayısının kendine göre en verimli aralığa ulaşması istenmiştir. Tüm bu süreçte kullanılan Renga da hem küresel anlamda hayata geçirilmiştir hem de gelecekte yapılacak, olası benzer çalışmalar için temel oluşturulmuştur.

Günümüzde pek çok oyun, geliştirim sürecinde yapay zekâdan faydalanmaktadır. Bu çalışmada bahsedilen çalışmaların da diğer yapay zekâ çalışmalarına destek olması beklenmektedir. En büyük amaç, PCG (Procedural Content Generation), yani yöntemsel içerik üretimi ile sonsuz sürekli oyunlar için yeni çalışmaların tasarlanmasında faydalı olabilmek ve yol gösterebilmektir.

Tez çalışmasının ilk bölümünde geliştirilen 2 boyutlu bir platform oyunu üzerinden yapay zekâ temelli oyun seviyeleme işleminin amacına değinilmiştir. İkinci bölümde platform oyunlarının süreci ve teknolojik gelişmeler anlatılmış, üçüncü bölümde ise teknik ayrıntılara yer verilerek bu çalışmadaki konumu aktarılmıştır. Dördüncü bölümde yapay zekâ modellerinde işlenecek verilerin içeriği aktarılmış ve çalışmada kullanılan teknolojiler açıklanmıştır. Aynı zamanda kullanılan yöntemler ve elde edilen seviye tanılama bulguları sunulmuştur. Beşinci bölümde ise kullanıcı kitlesine yönelik seviyeleme verilerinden ve bunların kullanıcılara etkisinden bahsedilmiştir. Çalışmanın sonuç bilgilerine ve olası yeni çalışmalara değinilmiştir.

2. PLATFORM OYUNLARI

Bu bölümde, video oyun tarihinin başından beri gündemden düşmemiş platform oyunlarının süreci ve teknolojik gelişmeleri aktarılmıştır.

2.1 Bilinen Örnekler

Platform oyunları video oyun tarihinin ilk evresidir. Brtie the Brain (Kates, 1950), 1950 yılında ortaya çıkan ilk endüstriyel oyun özelliğini taşımaktadır. Bu süreç, 8 yıl sonra William Higinbotham'ın Tennis for Two isimli oyunu ile devam etmiştir (Higinbotham,1958). Pong ise bu oyunun ilk gerçekleştirimi olarak platform oyunları arasında yer edinmiştir (Kent, 2001). İlerleyen zamanlarda oyun teknolojileri günümüze kadar gelişip farklı kategorilere ayrılmıştır. Jumpman adıyla anılacakken bir efsaneye dönüşen Super Mario (Togelius, 2009), Contra (Konami,1987) ve Metal Slug (Nazca Corporation, 1996) serileri gibi yapımların yanında, 3 boyutlu oyunların öncüsü olan Wolfenstein 3D'nin (ID Software, 1992) ortaya çıkmasına rağmen özelliklerini yitirmemişlerdir.

Şekil 2.1: Ori and the Blind Forest (Moon Studios, 2015)

Aksine, platform oyunları da gelişerek Limbo (Playdead, 2010), Inside (Playdead, 2016) gibi yapımlar ile yepyeni bir video oyun çağı başlatılmıştır (Simon, 2012). Günümüzde değer taşıyan platform oyunlarından biri olan, Moon Studios'un geliştirdiği Ori and the Blind Forest isimli oyun görseli Şekil 2.1'de verilmiştir (Moon Studios, 2015).

Bu oyunların çoğunluğu, günümüzde, Bağımsız Yapım Oyunlar (Independent Games) tarafınca geliştirilmekte ve örneklerine rastlanmaktadır. Bu terim, bağımsız yapımcıların geliştirdiği oyunların genel kategorisi olup, oyun motorlarının geniş kitleli kullanıcılara yayılması ve dijital oyun platformaları sayesinde seslerini duyurabilmesi sayesinde günümüzde özel bir yer edinmiştir.

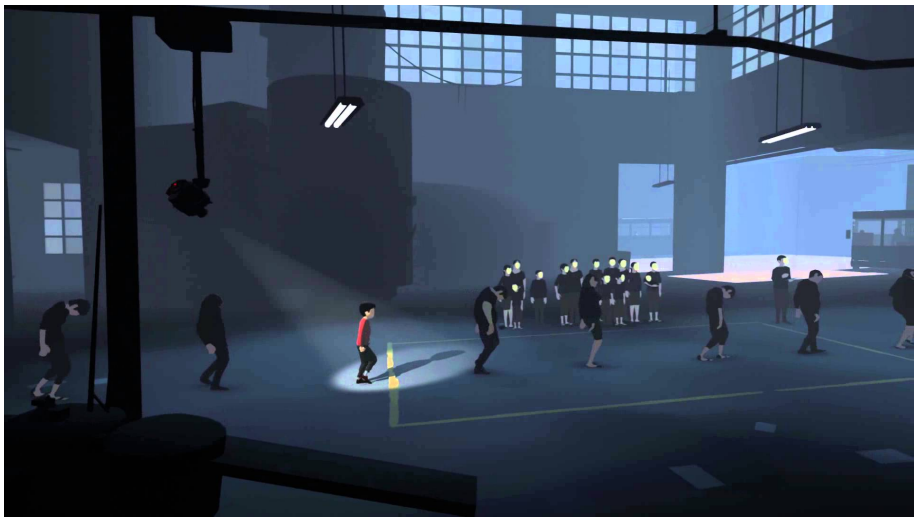
2.2 Platform Oyunlarının Gelişimi

Super Mario World ve Inside kendi dönemlerinin teknolojik özelliklerine göre incelenebilen platform oyunlarıdır. Super Mario World için Şekil 2.2 başlıklı görsel, grafiksel ayrıntıları yansıtmaktadır. Şekil 2.3 ise Inside isimli oyunun görsel örneğini içermektedir. Yan yana koyulduklarında göze çarpacak olan ilk özellik, grafiksel yenilikler olacaktır. Aynı zamanda gerçek ortam algısını yansıtan seslerin, Inside üzerinde daha etkin ve çarpıcı olduğu açıktır. Fizik kurallarının kıyaslanması; görsel anlamda karakter ve çevre animasyonlarındaki büyük değişiklikler dikkat çekmektedir.

Şekil 2.2: Super Mario World (Nintendo, 1990)

Yapısal anlamda ise, yapılan iş azalmış ancak verim artmıştır. Teknolojik gelişmelerin getirdiği süreç platform oyunlarını yeniliklere sürüklese de karakter, düşman, etkileşimli nesne ve ritim platform grupları, türün tüm oyun ve bölümlerinde temel taslak olarak kalmaya devam etmiştir. Yakın zamanda mobil platformlarda da boy gösteren platform oyunları bir kez daha türünün vazgeçilmezleri arasına yerleştiğini göstermiştir.

Tüm bu benzerlikler ve türünü devam ettiren örneklerden sonra, bu çalışmanın da platform oyunları üzerinden gerçekleştirilmesi en uygun seçenek olarak belirlenmiştir. Genel anlamda yapılan karşılaştırmalar, belirli özellik ve değerlerin kullanılacak yapay zekâ yöntemleri ile uyumlu çalışabilecek bir düzene büründürülmüştür.



Şekil 2.3: Inside (Playdead, 2016)

2.3 Literatür Araştırması

Akademik değerle yapılan çalışmaların yanında, yapay zekâ ile oyun sektörü arasında, özellikle mobil oyunlar üzerine gerçekleştirilen çalışmalar mevcuttur.

Spronck ve diğ. (2004), yaptıkları çalışma ile oyunların zorluk algısının kullanıcının düzeyi ve oyun değerlerine göre değişiklik gösterebileceğini belirtmişlerdir. Neverwinter Nights oyunun bir bölümünde, kullanıcının oyun içindeki karakterine karşı dinamik olarak karşılık verebilmesi için düşman özelliklerine,

geliştirilebilir bir taktik oluşturulmaya çalışılmıştır. Bu çalışmada ise tasarlanan sonsuz-sürekli bir platform oyununun tüm süreci ele alınmaya çalışılmıştır.

Yöntemsel içerik üretimine yönelik bir adım niteliğinde olan bu çalışmada, içeriklerin üretimine dair, Togelius ve diğ. (2011) tanımı dikkat çekmektedir. Bu doğrultuda, yöntemsel içerik üretimi herhangi bir insanın oluşturduğu, çevrimiçi veya çevrimdışı olarak oyuna kendisinin koyduğu içerik olmamalıdır. Aksine bu süreci gerçekleştiren bir algoritmaya bağlanması gerekmektedir. Çalışmada da yer alacak en önemli ayrıntılardan biri bu işlemdir. Renga isimli oyun ile elde edilen tanılama bulguları, oyun içi dinamiklerinin değişimine katkı sağlamaktadır.

2.4 Renga Platform Oyununun Ortaya Çıkışı

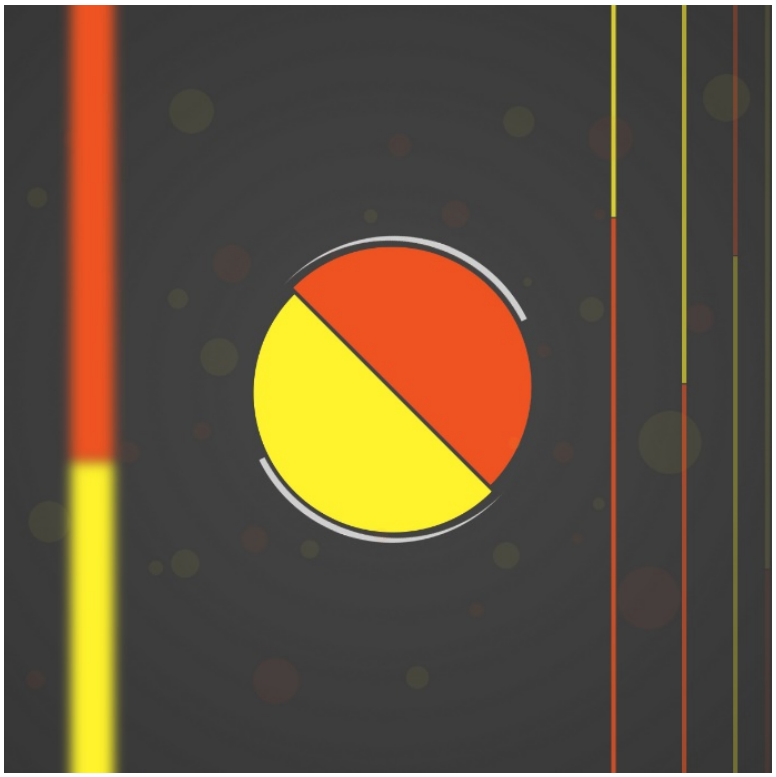
Yapay zekâ uygulamalarının, platform oyunlarındaki yönü incelenirse pek çok içeriği olağanlaştırmak gerekir. Super Mario World için yapılan en etkili işlemlerden birisi, uygulama yapılacak bölümün karakter bazında tekrar oluşturulmasıdır. Böylece matrisler elde edilebilmiş her oyun nesnesi karakterlere atabilmiş ve bu bölümler üzerinde işlem yapılabilmiştir.

Bu çalışmada da kullanılması düşünülen yapay zekâ yöntemleri için oyun tasarımı temel adımlara dayandırılmıştır. Farklı zorluk özelliklerini kapsayabilecek bir platform oluşturmak ve zorlukları ise kullanılan oyun varlıklarının sayısal özelliklerine bağlamak temel amaçtır.

Tüm bu sürecin etkileşiminde dikkat edilen başlıca özellikler için;

- Basit bir oynanış,
- Varlıklar arasında etkileşim,
- Karakter yapısının güncellenmesi,
- Oyun skoruna katkı
- Akademik çalışma
- Küresel kitleye yönelik eğlence uygulaması sunabilme

işlevlerini yerine getirebilecek düzeyde, 2 boyutlu bir platform oyunu tasarımı düşünülmüştür. Oyun için dil ise İngilizce olarak belirlenmiştir. Oyunun küresel çapta kitlelere hitap edebilmesi için tasarım bu düşünceyle şekillendirilmiştir. Şekil 2.4 geliştirilen Renga'nın tanıtım görseli olarak ortaya çıkmıştır.



Şekil 2.4: Renga, tanıtım görseli

2.5 Unity

Unity, mevcut sürümü ile oyunlar geliştirilebilen, aynı zamanda 3 boyutlu modelleme ve animasyonların tasarlanabileceği, 2 boyut desteği de bulunan bir oyun motorudur (Haas, 2018). Oyundaki her alan birer sahneden oluşur ve bu sahneler "scene" adı ile bilinmektedir. Sahnelerin içeriği oyun nesneleriyle doludur. Bu oyun nesneleri; "sprite", "texture", "prefab" ve "object" olarak isimlendirilmiştir. Her bir varlığın kendi içinde özel isimlendirmesi bulunmaktadır. "Texture", "object" yapılarının katmanını oluşturur, her biri bir desendir. Kamera, kullanıcının bakış açısını belirler. Her sahne kamera aracılığıyla bir bakış alanı kazanmaktadır. Renga aktif olarak 4 sahneye sahiptir. Bunlar; "menu", "howToPlay", "gameplay" ve "end" sahneleridir. Oyun Android işletim sistemli telefonlarla uyumlu çalışmaktadır. İlgili

telefondaki Android sürümünün 4.0'dan yüksek olması gerekmektedir. Şekil 2.5'te "menü" isimli sahneden alınmış bir görsel aktarılmıştır.

Şekil 2.5: menu sahnesi

Kullanıcıların ilk karşısına gelen sahne "menu" sahnesidir. Bu sahne akademik çalışmanın test edileceği verilerin girişlerinin sağlandığı sahnedir. Kullanıcılar, numara, rassallık değeri, yatay hız, yer çekimi, zıplama değeri ve engeller arası mesafe bilgilerinin girişini sağlayarak Unity içerisinde herhangi bir sahneye ait kod dosyasından erişilebilirlik hakkına sahip olur.

Oyunun başlangıç sahnesi "menu**"**, oyun akışına geçmeden kullanıcıların son hazırlıklarını tamamlaması gereken sahnedir. Gerekli özelliklerin girişi sağlanmıştır. Kullanıcı "PLAY" tuşuna dokunduğunda, kendisine gösterilecek şekilde, 3 saniye

sonrasında oyun başlatılmaktadır. Oyun süreci "gameplay" sahnesinde gerçekleşmektedir.

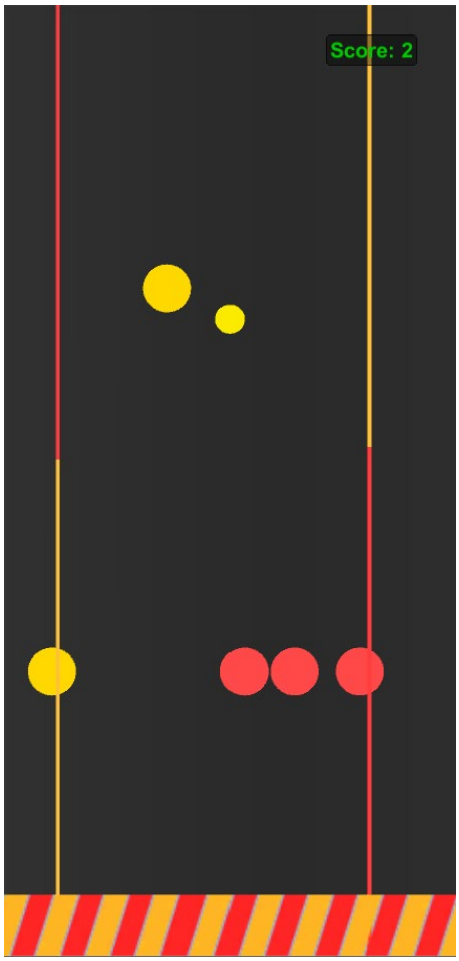
Şekil 2.6: howToPlay sahnesi

Kullanıcıları oyun için terimler ve oynanış hakkında önceden bilgilendirmek için Şekil 2.6'da görülen "howToPlay" sayfası oluşturulmuştur. Kullanıcı oyunda ihtiyacı olacak açıklamaları ve ipuçlarını bu sayfa sayesinde öğrenebilmektedir. Ayrıca çalışmada da yer bulacak oyun terimlerinin açıklamasının bulunduğu sayfadır.

"gameplay**"** sahnesi, Şekil 2.7'de gösterilmiştir ve kullanıcının oyun ile ilgili oynanış faaliyetlerinin tümünü gerçekleştirdiği sahnedir. Renga hata yapılana dek

oynanışı devam eden sonsuz sürekli bir oyundur. Kullanıcı tarafından yapılabilecek hatalar şu şekildedir:

- Bakış açısının dışına çıkmak
- Zemine çarpmak
- Farklı renkli bir Rengeçe çarpmak

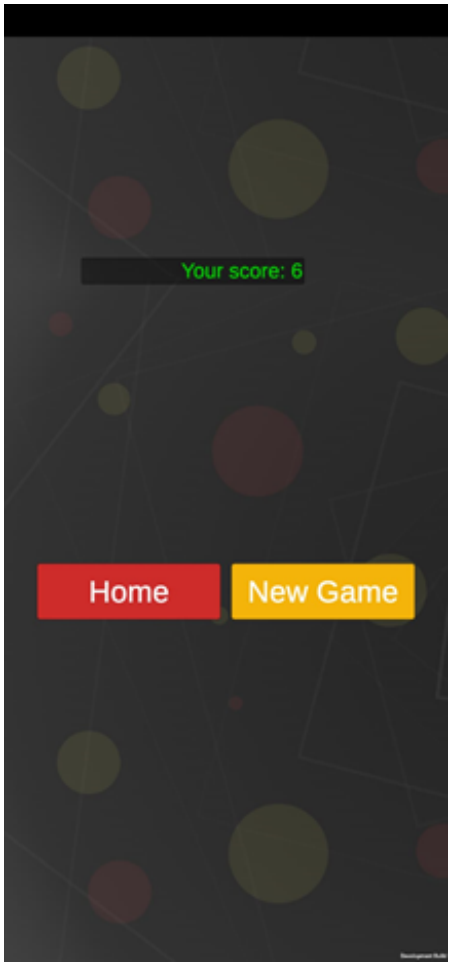


Şekil 2.7: gameplay sahnesi

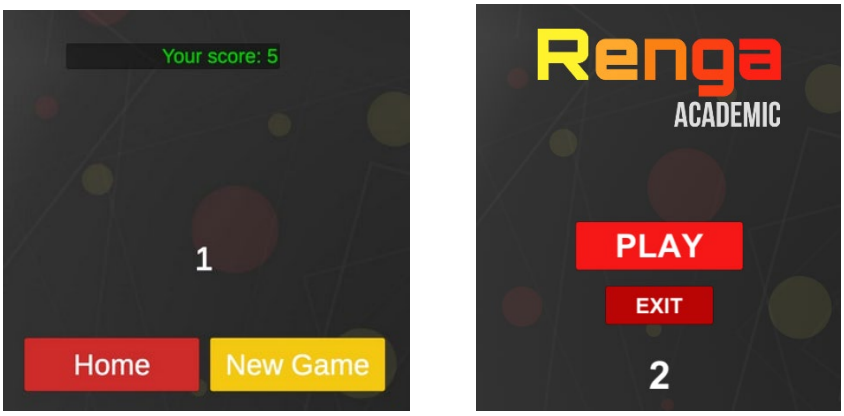
Bu sahnede oyun varlığı olarak yalnızca Renga mevcuttur. Arkaplan, zemin, Ahengler ve Rengeçler dinamik olarak oluşmaktadır. Oyun içi varlıkların ayrıntılı açıklamaları ilerleyen bölümlerde belirtilmiştir.

Şekil 2.8'de görülen **"end"** sahnesi ise oyunu sıradaki değerler ile baştan başlatır ya da kullanıcıyı ilk Şekil 2.5'te görülen menü sayfasına gönderebilecek tuşları barındırır. Bu ekranda oyunun sonucu da yazmaktadır.

Renga'nın yanlış etkileşimde bulunmasından sonra ekran, ses efektlerinin geçişi boyunca durdurularak, end sahnesine otomatik bir geçiş sağlamaktadır. Ancak kullanıcı tıklama veya dokunma ile de bu geçişi kendi sağlayabilmektedir.



Şekil 2.8: end sahnesi



Şekil 2.9: 3'ten geriye sayım ile gameplay sahnesine geçiş yapılmaktadır.

Şekil 2.5’teki “menu” sahnesinde “PLAY” tuşu ve Şekil 2.8’deki “end” sahnesinde bulunan “New Game” tuşlarına basıldığında, Şekil 2.9’da belirtilen geri sayım işlemleri gösterilmiştir.

3. RENGA

Yapay zekâ, oyunların geliştirme sürecindeki karakter davranışları, modelleme, oyun tasarımı, içerik üretimi gibi pek çok konuya hitap edebilecek bir sistem içermektedir. Bu çalışmada ise yapay zekâ tekniklerinin Renga isimli 2 boyutlu platform oyununda, nasıl kullanılacağı ayrıntılı olarak yer edinmiştir. Rassal değerler ile dinamik içeriklere sahip olan oyunun bölümleri, varlık özelliklerine göre bir zorluk değeri göstermektedir. Bu deneyim, dinamik varlıkların özellikleri değiştirilerek kullanıcı işlemleri ile elde edilmektedir ve bölüm sonunda seviyeleme seçenekleri, yeni zorluk değerlerini oluşturan veri fikirleri sunmaktadır.

Akademik çalışmalarda kullanılmak için tasarlanan Renga, aynı zamanda küresel olarak kitlelere hitap edebilecek bir yapıya sahiptir. Google Play Store üzerinde de kullanıcıların erişim imkânı bulunmaktadır. Tüm bu aşamaların öncesinde elde edilen veriler, oyun mimarisinin temelini en basit şekilde oluşturmak amacıyla tasarlanmıştır. Kolay bir oynanabilirlik içerisinde, kişisel beceri sağlayabilmek üzere geliştirilmiştir.

3.1 Oyun Özeti

Renga, hedef kapıların renkleri üzerinden oynanan 2 boyutlu bir platform oyunudur. Oyunun genel varlıklarını Rengeçler, Ahengler ve Renga oluşturmaktadır. Platform üzerinde; Ahengler Renga'nın rengini değiştirirler. Rengeçler ise Renga'nın geçişini ve puan kazanmasını sağlayan nesnelerdir. Renga'nın bu etkileşimlerinin dışında; yanlış bir Rengeç'e ya da zemine çarpma, oyun görselinin dışına çıkma gibi olumsuz etkileşimleri bulunmaktadır.

3.2 Geliştirim Süreci

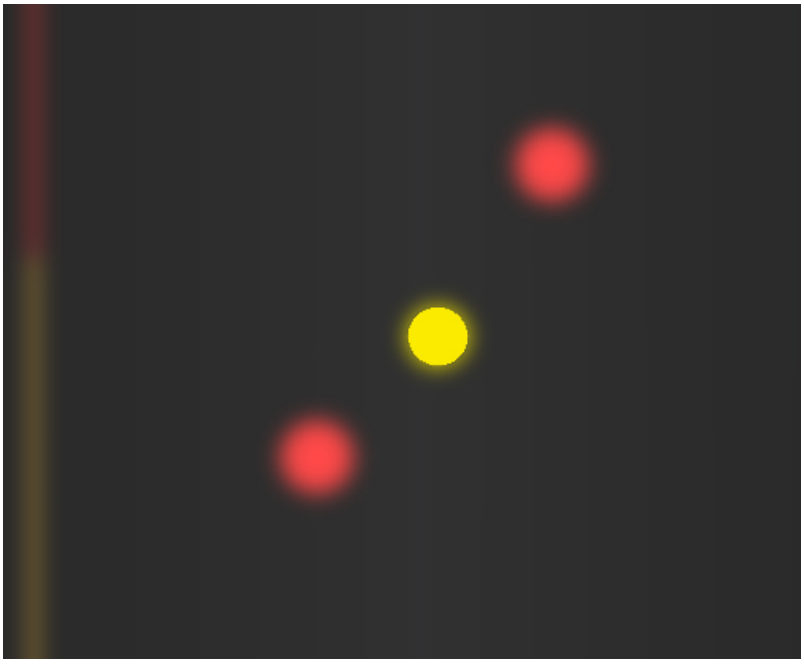
Öncelikle varlıkları belirlenen oyunun tasarım sürecinde, varsayımsal bir görseli gerçekleştirilmiştir. Görselde Renga, Aheng ve Rengeçler'in genel şeması bulunmaktadır. Oyun tasarımı bölümünde, varlıkların ayrıntılı tanıtımı yapılmıştır.

Renga'da, yapay zekâ yöntemlerinde kullanmak üzere veri oluşturulabilecek, zorluk algısı da kullanıcı bazında tespit edilebilecek bir yapıya uygun olmalıdır. Bu düşünceyle şekillenen temel noktalar aşağıdaki gibidir.

- Yapay zekânın oyun geliştiriciliği üzerindeki alt başlıklarından biri olan, yöntemsel içerik üretiminin kullanımını mümkün kılmak,
- İçerik üretimi için minimum gereklilikleri belirleyen oyun tasarımını oluşturmak,
- Oyunu oynayan kullanıcılardan elde edilen verilerin tutulduğu bir veri tabanı oluşturmak,
- Bu verileri yapay zekâ yöntemleriyle işleyerek oyuncuya yeni veriler sunmak.

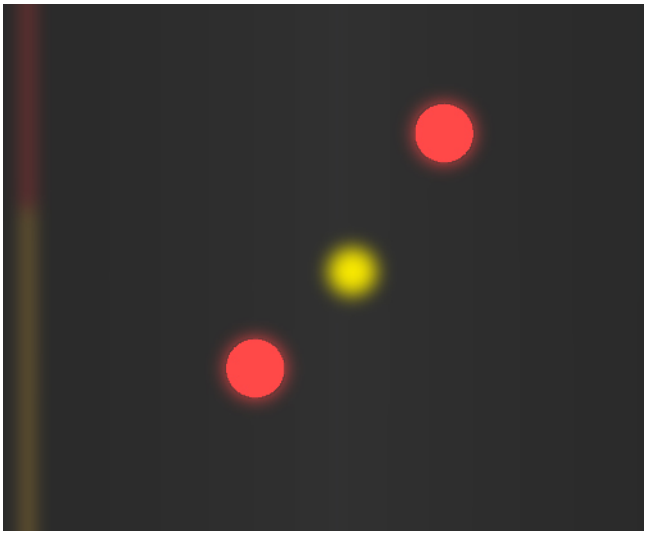
3.3 Oyun Tasarımı

Varlıkların özellikleri ve birbirleri arası etkileşimlerinin, yapay zekâya aktarımları düşünüldüğünde aşağıdaki açıklamalardan yararlanılmaktadır;



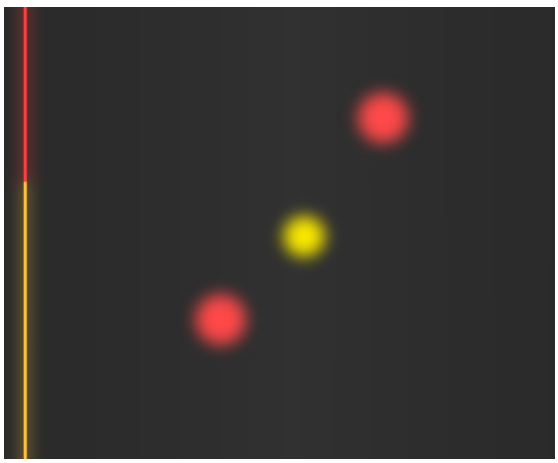
Şekil 3.1: Sarı renkli Renga

Renga; sarı olarak başlamaktadır. Kırmızı veya sarı Ahengler ile etkileşime geçebilmektedir. Fiziksel kurallara sahiptir. Yer çekimi değerine göre serbest düşüş hareketi yapmaktadır. Yatay ve sabit bir hız değerine sahiptir. Kendi rengine denk Rengeçler'den geçerek 1 puan kazanmaktadır. Şekil 3.1 Renga oyun varlığını göstermektedir.



Şekil 3.2: Kırmızı Ahengler

Aheng; Sarı veya kırmızı renkte olabilmektedir. Temas ettiği anda kendi rengini Renga'ya vermektedir. Verdiği renk ile Renga'nın sarı veya kırmızı kapıdan geçebilmesini sağlamış olmaktadır. Platform üzerinde rassal bir şekilde çifter adet bulunmaktadır. Şekil 3.2 kırmızı renkli Aheng oyun varlığını göstermektedir.



Şekil 3.3: Kırmızı alanı üste denk gelen rassal kesişim noktalı Rengeç

Rengeç; Şekil 3.3'te görülen çizgisel kapı şeklindeki Rengeçler, Renga'nın bir sonraki engele ulaşmasını sağlayan geçittir. Bölümde bulunduğu sayı kadar skora puan ekler. Birbirleri arasında mesafe değerine sahiptir. Bulunduğu sütunda rassal oranda sarı ve kırmızı renklerin uç uca eklenmesiyle oluşmaktadır.

Şekil 3.1'deki Renga karakteri, oyun içinde sarı renk ile başlar. Kırmızı bir Aheng'e denk gelmediği sürece sarı renkte kalır. Kullanıcılar ise oyun akışı boyunca her tıklama veya dokunmada, Renga'nın sahip olduğu zıplama değeri kadar yukarı, yer çekiminin aksi yönünde, hareket eder. Bu esnada yatay hız veya yer çekimi değeri değişmeyecektir. Renga her geçtiği Rengeç başına 1 puan kazanacaktır. Eğer mevcut ekran dışına çıkar, zemine düşer veya Rengeç'in farklı renk bölümünden geçer ise oyun sonlanacak ve "end" sahnesine geçecektir. Kendi rengindeki Aheng'i alması herhangi değişiklik oluşturmamaktadır. Ahengler'in her biri toplandığında, rassal koordinatlarda yeni bir Aheng oluşmaktadır.

3.4 Kullanılan Teknolojiler

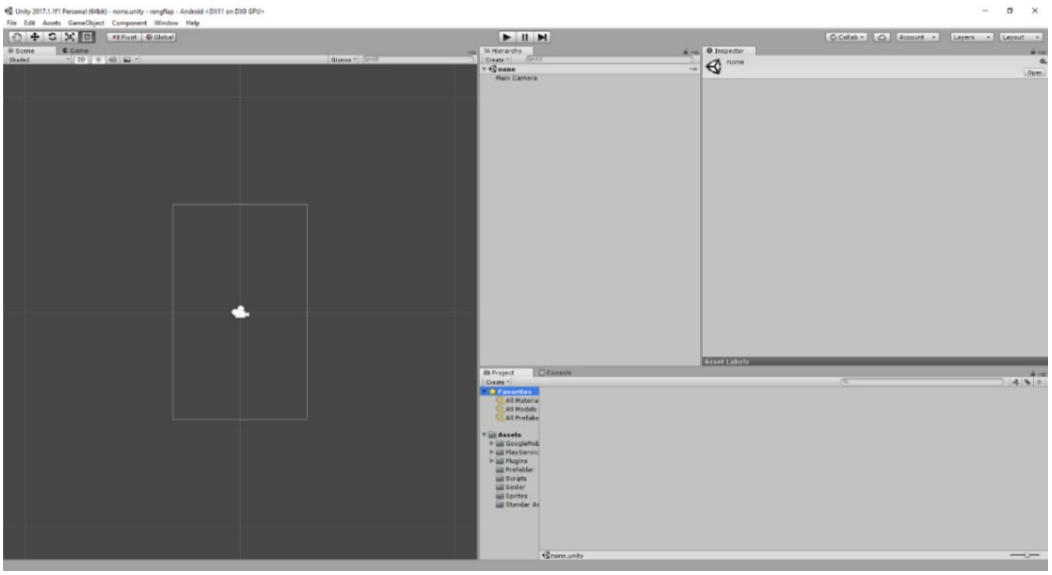
Renga; Unity oyun motoru ve Visual Studio 2017 Community programları üzerinde geliştirilmiştir. Veritabanı işlemleri için gerekli kodlar Sublime Text programında php olarak yazılmıştır. Kod dosyalarının içeriği C# ve javascript kodlarından oluşmaktadır. Herhangi iki ayrı programlama dili ile yazılan kod dosyaları arası doğrudan bağlantı kurulmamıştır. Unity arayüzü ile düzenlenmiş olan sahne, varlık ve ayrıntıların dinamik olarak oluşmasını sağlayan kodlar javascript ile sağlanırken, sahneler arası geçiş kodları ise C# ile yazılmıştır. Unity programının 2018 yılı içerisindeki güncellemelerinden itibaren javascript desteğini sonlandırması sebebiyle çalışmalar iki dil desteği üzerinden yürütülmüştür.

Veri toplama işlemleri için MYSQL veritabanı ve PHP kullanılmıştır. İşlemler yerel sunucu aracılığıyla tamamlanıp "gameonyou.tk" alan adı ile ortak erişilebilir bir sisteme taşınmıştır.

Süreç boyunca, Git isimli sürüm sisteminden yararlanılmıştır. Renga'nın gelişme süreçleri bu sayede takip edilmiş ve geriye dönük iyileştirmeler yapılmıştır.

Oyun içerisinde Unity'nin sağladığı düzenleme ortamı sayesinde çeşitli parçalarla bölüm ve sahneler oluşturulmuştur. Dosyalar içinde, "assets" tüm varlıkları kapsar; kodlar, prefab'lar, nesneler, desenler vb. Şekil 3.4'te görülen arayüz Unity programından alınmıştır.

Verilerin toplanması için ise MYSQL üzerinden yerel bir veri tabanı oluşturulmuş, bunlar daha sonra farklı sistemler üzerinde kullanılmak üzere hazır bir şekilde bulundurulmuştur. Örnek veri toplama işlemleri sonrasında, belirlenen formatlarda, "gameonyou.tk" adresi üzerine aktarılmıştır.



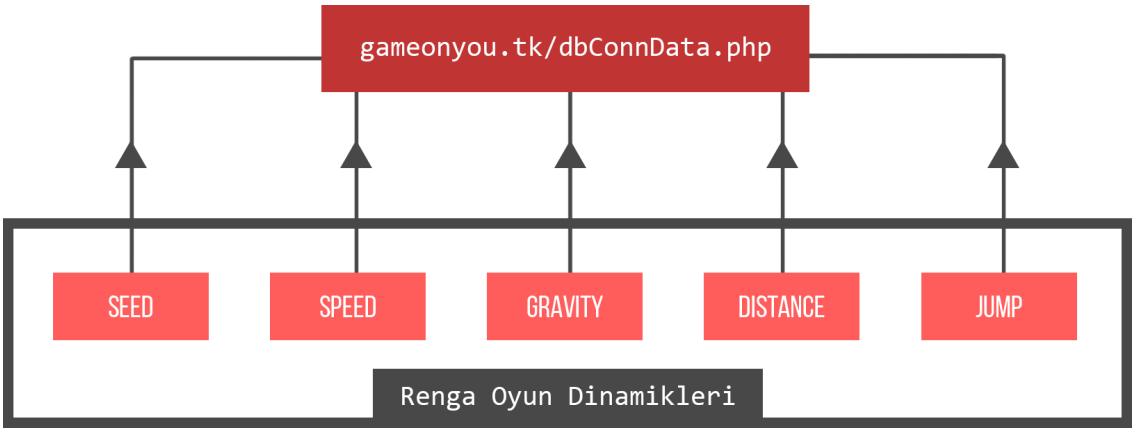
Şekil 3.4: Unity 2017.1.1f1. temel arayüz görünümü

Renga süreci boyunca versiyonlama sistemi kullanılmıştır (GİTHUB). Oyun versiyonları kullanım ve geliştirme adına dallara ayrılmıştır. Bu çalışmada bilimsel hesaplamalar için hazırlanan versiyon temel alınmıştır. İlgili sürümde oyun içi arayüz Türkçe olarak ayarlanmış ancak sonraki sürümlerde, oyun tasarımı da dâhil olmak üzere İngilizce isimlendirmelerle düzenlenmiş ve çevrilmiştir. Bu da uygulama içi hazırlanacak veri kümelerinin evrensel nitelik taşımasını sağlayacaktır.

"menu", verilerin toparlanarak kaydedilmesi ve diğer sahnelerde aktarımların yapılması için test kullanıcılarına sunulmuştur. Kullanıcılar hakkında bilgiler ilerleyen bölümlerde ele alınmıştır. Bu veriler "sendToDb.js" dosyasında aşağıdaki şekillerde kaydedilmiştir:

- **Cihaz Numarası;** phoneId
- **Rassallık Değeri;** seed
- **Yatay Hız;** speed
- **Zıplama Kuvveti;** jump
- **Yer Çekimi;** gravity
- **Engeller Arası Uzaklık;** distance

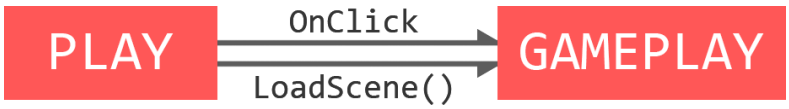
İlgili akış görseli Şekil 3.5'de sunulduğu gibidir:



Şekil 3.5: sendToDb.js

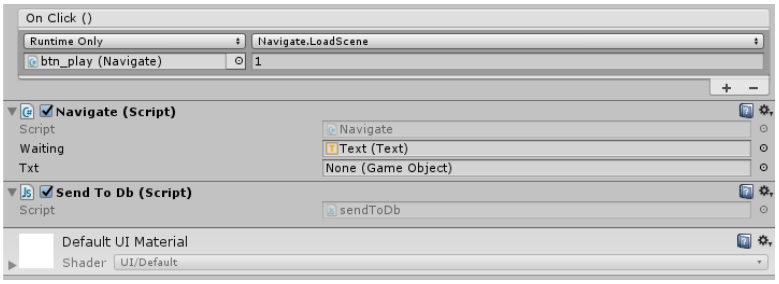
Elde edilen veriler "PlayerPrefs" özelliği ile sahneler arası paylaşılabilmektedir. String değerli veriler gerekli görülen duruma göre double ve float türü değişkenlere çevrilmiştir.

"menu", Renga'nın ana sahnesidir. Kullanıcının oyunu gördüğü ilk sahnedir. Sahnesinin açılmasıyla oyun içi zorluğu belirleyecek veriler, önceden hazırlanmış ve kullanıcının cihazı için belirlenen özel bir numara ile veri tabanına aktarılmaktadır. PLAY tuşu ile başlayan oynanış zamanı hem verileri oyuna aktaracaktır hem de gameplay sahnesine geçişi sağlayacaktır. Sahne değişiminin sağlanımı Şekil 3.6'da gösterilmiştir. OnClick olayı ile tetiklenir ve LoadScene() fonksiyonu ile geçiş sağlanır.



Şekil 3.6: Sahne değişi

Unity'de bir arayüz nesnesine veya herhangi bir nesneye kod üzerinden erişebilmek için, Şekil 3.7'de görüldüğü üzere nesnenin içerik bölümüne, erişimi istenen kod dosyası tanımlanmalıdır. Bu sayede "LoadScene" fonksiyonunu "btn_play" ile bağlayarak, ekranın "gameplay" sahnesine aktarımı gerçekleştirilmektedir. Navigate.cs içerisinde bulunan LoadScene() fonksiyonu "SceneManager" sınıfı üzerinden çalışmaktadır. Şekil 3.6'da belirtilmiştir ve btn_play (button olarak geçmektedir ve kullanıcı arayüzü içerisinde tanımlıdır) nesnesine bağlı bir fonksiyondur. Bu bağ "OnClick" olayı ile sağlanmıştır. Nesneye atanan kod dosyaları, Unity'de properties isimli arayüz aracılığı ile sağlanabilmektedir. "OnClick" isimli bölme ise button kontrollerinde bulunmaktadır. Yalnızca button kontrolü etkinleştirildiğinde harekete geçecek fonksiyonu barındırmaktadır.



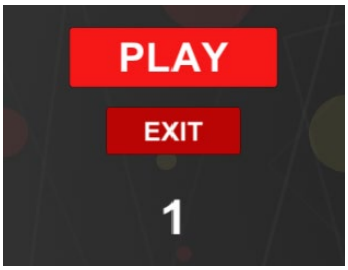
Şekil 3.7: Tıklanması ile tetiklenecek fonksiyonlar.

"Menu" sahnesinde "Play", "How To Play", "Exit" tuşları bulunmaktadır. Kullanıcı "Play"e bastığında oyuna geçmeden 3 saniye boyunca her saniyede bekleme bildirimi alır ve sonra "gameplay" sahnesine geçer. Bu etkileşimin sağlanımı Şekil 3.8'te belirtilmiştir.

```
1  public void LoadScene(int Level)
2  {
3      StartCoroutine(delay(level));
4  }
5  IEnumerator delay(int Level)
6  {
7      waiting.text = "3";
8      yield return new WaitForSeconds(1.0f);
9      waiting.text = "2";
10     yield return new WaitForSeconds(1.0f);
11     waiting.text = "1";
12     yield return new WaitForSeconds(1.0f);
13     SceneManager.LoadScene(level);
14
15 }
```

Şekil 3.8: Navigate.cs üzerinde aktarım ve bekleme işlemleri

"LoadScene" fonksiyonun parametresi sıradaki sahnenin hangi konumda olduğunu bildiren "level" adındaki parametreyi bekletme fonksiyonuna (delay) göndererek Unity'e ait "SceneManager" metoduyla "gameplay" sahnesine geçiş sağlamaktadır.



Şekil 3.9: Menu sahnesi ve bekletme bildirimi

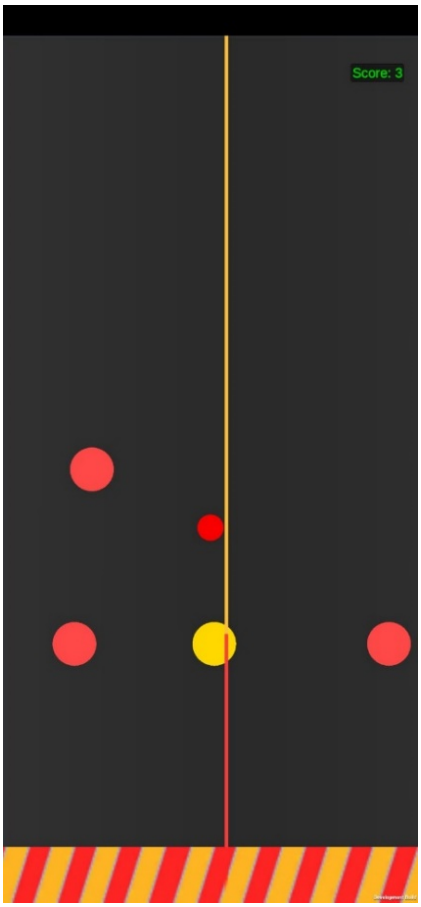
"WaitForSeconds" ise içine verilen ondalık saniye değeri kadar sistemi bekletebilir. Bu işlem IEnumarator ile sağlanmaktadır. Kullanıcı btn_play isimli button nesnesinin "OnClick" olayını aktif ettiğinde 3 saniye boyunca bekletilerek, bildirim alır. Sonrasında sahne, "gameplay"e geçerek oyun başlatılmaktadır. Geri sayım bildirimleri Şekil 3.9'daki gibi gösterilmektedir.

Kullanıcı bu sahnede ise yalnızca ekrana dokunarak Renga'nın zıplamasını sağlar. Ahengler toplanarak Renga'nın rengi değiştirilebilmekte ve Renga aynı renkli Rengeçler'den geçebilmektedir. Sağ üst köşede gösterilen metin ile kullanıcı kendi puanını takip edebilmektedir.

Veritabanında yer alan değerler, sırasıyla "menu" sahnesinde kullanılmıştır. Oyun başladığında "gameplay" sahnesinde kullanılmak üzere çağırılacaktır. Bunun için de "PlayerPrefs" özelliği kullanılmaktadır. Şekil 3.10'da bu özelliğin nasıl sağlandığı görselleştirilmiştir.

Şekil 3.10: RengaMove.js, verilere erişiminin sağlanması

Update fonksiyonu sürekli bir işlem içerisinde olduğu için verileri süre boyunca işlemeye devam etmektedir. GetMouseBotton(0), bilgisayarlarda sol fare tuşunu ifade ederken, Android işletim sistemli cihazlarda dokunma ile temasın etkin kılındığı eylemi sağlamaktadır. İki işletim sistemi aynı işlevi desteklemektedir.

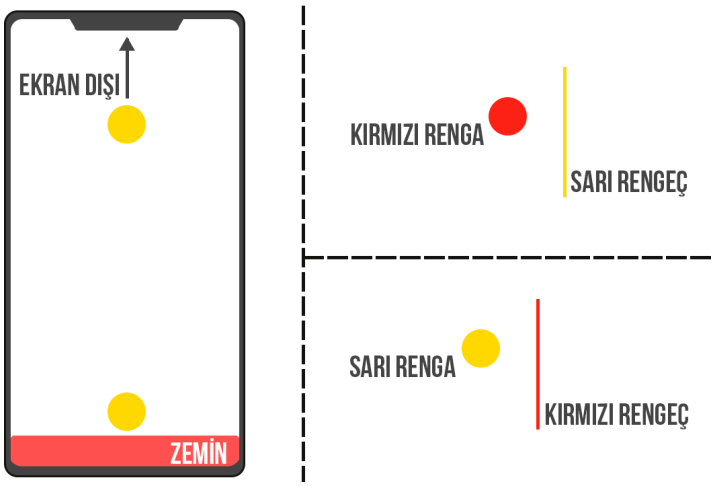


Şekil 3.11: gameplay sahne akışı

Şekil 3.11 Renga'nın oyun içi görüntülerinden birini barındırmaktadır. Oynanış süresi boyunca Renga, yer çekimine yönelik değerle düşecektir. Her dokunuş ile zıplama değeri kadar dikeyde yukarı hareket edecektir. Dikey hareket için "jumpSFX" isimli değer, ilgili hareket için kullanılan ses parçasının dengidir. Transform fonksiyonları, yazılan bu metotların atandığı "gameObject" için işlev sürdürmektedirler. Renga bu sayede yatay hareketlerini ekran kartezyeninde sağlarken (Space.World adı verilen bu terim ekrandaki konum baz alınarak işlenmiştir), düşey hareketlerini kendi ekseninde gerçekleştirmiştir. 3 parametreli "translate" fonksiyonları; x, y ve z değerlerini ifade etmektedir. Program bazında oyun varlıklarının duruşunu ve oluşumunu temsil etmektedirler.

Oyun süresince, başta verilen değerler "update" fonksiyonu boyunca işlenmektedir. Şekil 3.12'de belirtildiği gibi oyun bitiş şartları 3 farklı durumda oluşabilmektedir, Renga;

- Zemine düştüğünde
- Ekran dışına çıktığında,
- Farklı renkten bir Rengeç'e çarptığında, oyun sonlanacaktır



Şekil 3.12: RengaMove.js, update fonksiyonu

Update fonksiyonu ele alındığında, Renga'nın y kartezyenindeki pozisyonu ile ekranın dışında kalması kontrol edilmiştir. Renga'nın boyutsal değeri 0,8 olarak verilmiştir. Bu sebeple kontroller, basit hataların önüne geçilebilmesi için 1 tamsayı değeri üzerinden verilmiş ve 0,2'lik hata payı ile ekran dışına çıkılması tolere edilmiştir. İleride de bahsedilecektir ki "hit" yalnızca Rengeçler ile olan çarpışmaların doğruluk değerini kontrol etmektedir. Aktif olmadığı zamanlarda hitFallSfx(); fonksiyonu çalıştırılmıştır.

```
function OnTriggerEnter2D(temas: Collider2D)
{
    if (temas.tag == "ground" && !hit) {
        Destroy(Camera.main.GetComponent(Arkaplan));
        speed = 0;
        if (!hit) {
            gameOver = true;
            hit = true;
            isDie = true;
             hitFallSfx();
        } } ...
```

Şekil 3.13: RengaMove.js, Renga'nın zemine düşme kontrolü

Şekil3.13 görseli, Renga'nın zemin ile oyunun sonlandığı bilgi kontrollerini içerir. "OnTriggerEnter2D", javascript kodlarının atandığı nesne olan Renga'nın, başka bir nesneye temasıyla tetiklenmektedir. Kontrol mekanizmasında bulunan tag kontrolü, zemine atanmış olan "ground" etiketi ile eşleşirse aktif oyun bileşenlerini durdurmaktadır. Arka plan, oyun süresince arkada belirli bir hız ile devam eden arka planın işlemlerini durdurur. Renga'nın yatay hızı da sıfıra eşitlenerek, tek aktif değer "gravity" olarak kalır. Belirtildiği gibi mantıksal değerli "hit" kontrolü Renga'nın daha önce bir nesneye çarpıp çapmadığını öğrenebilmek için uygulanmıştır. Şekil 3.14'te gösterilen hitFallSfx() fonksiyonu, "end" sahnesine yönlendirilmeden çalan ses dosyalarını çağırarak oyunun etkin son işlevlerini "gameplay" sahnesinde yerine getirmektedir.

```
function hitFallSfx() {

    GetComponent.<AudioSource>().PlayOneShot(fallSFX);
    yield WaitForSeconds(1.1);

    GetComponent.<AudioSource>().PlayOneShot(endSFX);
    yield WaitForSeconds(1.8);

    Application.LoadLevel("end");
}
```

Şekil 3.14: RengaMove.js, hitFallSfx() fonksiyonu

Debug.Log() fonksiyonları, oyun yapım süresince pek çok hata kavramını giderebilmek amacıyla kullanılmıştır. Ses efektleri PlayOneShot(AudioSource) fonksiyonu ile bir kere baştan sona oynatılabilmektedir. Tüm bu işlemler sonrası Renga'nın yere düşmesiyle yapılan son işlem kullanıcının "end" sahnesine aktarılmasıdır ve bu durum yine Şekil 3.14'te gösterilmiştir.

```
... if (temas.tag == "altEngel" || temas.tag == "altEngelBot") {
        if (gameObject.GetComponent.<Renderer>().material.color == Color.red) {
            Destroy(Camera.main.GetComponent(Arkaplan));
            Destroy(temas);
            speed = 0;
            hit = true;
            hitRengecSfx();
        }
        else {
            if (!hit) {
                skor++;
                GetComponent.<AudioSource>().PlayOneShot(pointSFX);
            }
        } } ...
```

Şekil 3.15: Rengeçler üzerinde temas kontrolü

Bir diğer kontrol mekanizmasında belirtilen etiketler, Rengeçin kırmızı veya sarı renkte olması ve ilgili renkteki Rengeç'in yukarıda-aşağıda olması üzerine sağlanmıştır. Eğer "OnTriggerEnter2D" tetiklendiğinde meydana gelen çarpışma "hit" değerini etkilemiyorsa, skor da 1 sayı değeri artırılmaktadır. Bu durum Renga'nın etkileşime geçtiği Rengeç'in renklerinin aynı olduğunu bildirir. Aynı işlemler sarı ve kırmızı Rengeçler için kullanılmıştır. Bu durumun program ifadesi Şekil 3.15'te belirtilmiştir.

Renga'nın renk değiştirebilmesi Ahengler aracılığıyla sağlanmaktadır. Sarı ve kırmızı renkteki Ahengler için benzer kontrol mekanizmalarından biri Şekil 3.16'daki gibidir.

```
if (temas.tag == "sari" && !hit) {
        gameObject.GetComponent(Renderer).material.color = Color.yellow;
       GetComponent.<AudioSource>().PlayOneShot(changeSFX);

        var val1: int = Random.Range(1, 5);
       var tempVal: int = val1;
       val1 = Random.Range(1, 5);

        if (tempVal == val1) {
           val1 = Random.Range(1, 5);
        }
           temas.gameObject.transform.position.x += val1 + 0.1;
}
```

Şekil 3.16: Aheng mekanizması

Şekil 3.16'da Renga'nın temasa geçtiği Aheng etiketinin eşleşmesiyle, mekanizma gerçekleşecektir. "gameObject" ile çağırılan Renga, sarı renge, materyal özelliği kullanılarak döndürülmüştür. Sonrasında yapılan işlemler ile Aheng'in bir sonraki yeri rastsal olarak belirlenmiştir. Ahengler silinmeyerek yalnızca kartezyen düzlemdeki yeri, rassal olarak değiştirilmiştir.

Tüm bu işlemlerin yanında Rengeç ve Ahengler'in oluşturulması ise verilen rassal değerine göre başlar ve devam eder. Bu işlemler ise "spawners.js" dosyası ile sağlanmıştır.

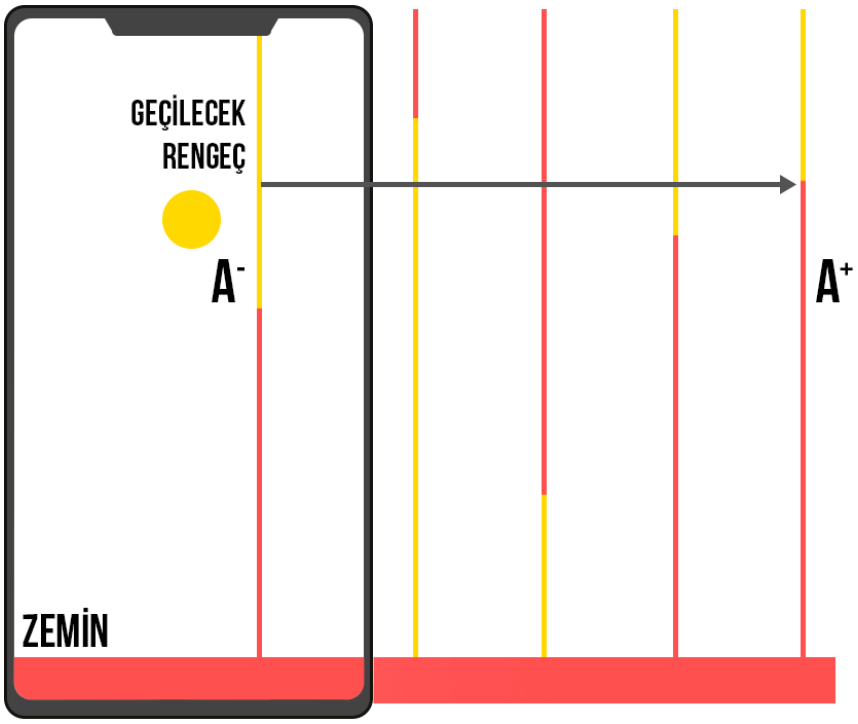
```
Random.seed = parseInt(PlayerPrefs.GetString("Seed"));        (1)
```

(1) numaralı kod parçacığı, seed ifadesinde, rassal değere yönelik bir bölüm sunmak için kullanılmıştır. Tüm kullanıcılarda aynı bölüm nesneleri ve veri kümelerine göre şekillenmiş ayarlar ile başlatılacaktır.

Rengeç ve Ahengler sahnenin başından itibaren yer almamışlardır. “Sprite” nesnesi üzerinden “prefab” yapıları oluşturularak dinamik bir şekilde bölüm tasarlanmasında kullanılmışlardır. Oyun bu oluşumlarla çarpışmalar olmadığı sürece devam edecektir. Öncelikle ekran boyutu hesabıyla engelin alabileceği yükseklik belirlenmiştir. Sonrasında oyuna üst ve altta denk adet olmak üzere Rengeçler’in ataması yapılmıştır. Bu atama transform fonksiyonu ile sağlamıştır.

İşlemlerin ardından engel oluşumu for döngüsü ile sahnede var edilmeye başlatılmış, veri tabanından gelen “distance” değerince de aralıklar bırakılmıştır.

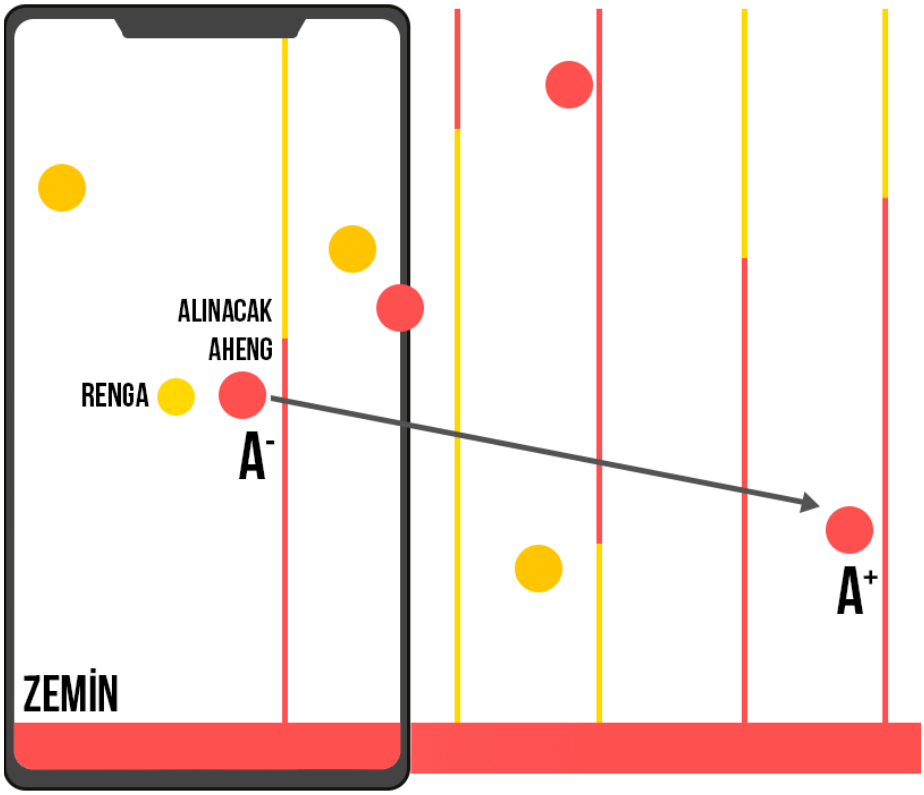
Bu işlemler Şekil 3.17’de görselleştirilmiş ve anlatılmaya çalışılmıştır. Renga’nın teması ile A Rengeç’i ok ile gösterilen alana farklı kesim noktaları ve farklı renk paylarıyla taşınmıştır.



Şekil 3.17: spawners.js, Rengeçler’in oluşum ve düzenlemeleri

Farklı kontroller ile yapılan bu işlem aslında tek bir dayanağa bağlanmaktadır. Rengeçler’in, oluşan 1 veya 2 değerinden birinin, rastgele değerine göre çaprazlanması sonucu renk durumları üstte veya altta bulunabilir. “Instantiate” fonksiyonu ilgili “prefab”ın ortaya çıkmasını sağlarken içeriğinde bulunan değerler kartezyeni ve orijin noktasının yerelliğini ifade eder. “Quaternion.identity” de bu lokalizasyonun ekran

üzerinde olmasını sağlayan başka bir özelliktir. Rengeçler için oluşturulan alt ve üst prefabları ("bot" ve "top" olarak isimlendirilmiştir. Örneğin; ustEngelTop, altEngelBot, vb.) kendi ağırlık merkezlerinin ya en üstte ya da en altta olmasına göre isimlendirilmiş ve uygulanmıştır. Ağırlık merkezleri yerel konum değerlerinin bu merkezlere göre değiştirilebilmesinde kullanılmaktadır. Sarı ve kırmızı bölümlere sahip olan Rengeçler'in, dikey duruşlarına göre üstte ve allta kalacak renk bölümlerinin belirginliği bu yapıyla sağlanmıştır.



Şekil 3.18: spawners.js, Ahengler'in oluşum ve düzenlemeleri

Ahengler, başlangıç sayıları kadar ekran sınırları kapsamında, oluşumlarını "spawners.js" içerisinde sağlarlar. Her 2 renkteki Aheng de bölüm içerisinde bulunmaktadır. İki engel arasında meydana gelebilecek boşlukta Ahengler aynı renk olabilmektedir. Herhangi bir Aheng ile Renga'nın etkileşime geçmesi Aheng'i yeni bir noktaya rassal olarak aktaracaktır. Bu işlemler de Şekil 3.18'de görselleştirilmiştir. A isimli Aheng, sarı Renga ile etkileşime girmiştir. Bu da örnek olarak verilen yeni A^+ pozisyonuna taşımıştır. Renga da bu işlem ile kırmızı renge bürünmüş olacaktır.

4. YAPAY ZEKÂ TEMELLİ OYUN SEVİYELEME SONUÇLARI

Bu bölümde Renga oyununun kullanıcı için etkileri ve ne gibi değerlerle nasıl bir ortam oluşturulabileceği belirtilmiştir. Verilerin nasıl toplanıldığı ve bu süreçte kullanılan teknoloji ve yöntemlerle nasıl sonuçların oluşturulduğu gösterilmeye çalışılmıştır. Karışıklık matrisi aracılığıyla veriler incelenerek, çıkarımlar bu doğrultuda yapılmış ve ileriye dönük çalışmalar için bulgular detaylandırılmıştır.

4.1 Verilerin Toplanması

Verilerin toplanması için çeşitli kullanıcılarla oyunun oynanması gerekmektedir. Bu çalışmada, her kullanıcı rastgele oluşturulmuş başlangıç değeri ile en az 5 defa oyunu oynamıştır. Tablo 4.1'de örnek veri tablosu gösterilmektedir.

Tablo 4.1: Kullanıcıya ait 5 oyun örneği

PHONEID	SEED	SPEED	GRAVITY	JUMP	DISTANCE	STATUS	SCORE
ABCDEF9876	50	1.0	12	4.0	1.8	0	0
ABCDEF9876	50	1.0	10	4.0	1.8	0	0
ABCDEF9876	50	1.0	14	4.0	1.8	0	0
ABCDEF9876	50	1.0	12	5.5	1.8	0	0
ABCDEF9876	50	1.0	12	2.5	1.8	0	0

Daha önce verilerin saklandığı "PlayerPrefs" yapısı boyunca veritabanında alınmış bilgiler datas.php dosyası ile elde edilmiş, Unity'de belirtilen **WWWForm()** nesnesi ile sağlanmıştır. Veriler daha sonra sunucuya aktarılmak üzere, "http://gameonyou.tk/" adresinden işlem yapılmıştır. Renga, Google Play Store'da kullanıcılara açıktır, ancak akademik özellikler aktif değildir. Bu sebeple veriler öncelikle "xampp" ile oluşturulan yerel sunucuda barındırılmaktadır. Ayrıca akademik sürüm için harici olarak uygulama dosyası "http://gameonyou.tk/Renga.apk" adresi üzerinden kullanıcılarla paylaşılmıştır.

dbConnData.php isimli dosya dökümünde, veri tabanına, **pdo** veri işleme yapısı kullanılarak bağlantı kurulmakta ve kayıtlar işlenmektedir.

Öncelikle, kullanıcı oyunu ilk defa çalıştırdığında, oyun için verileri toplanarak veritabanında kayıtlar açılmaktadır. Sabit değerler haricinde oynanış verileri genel bir veri yapısı dışında, her değerin belli oranlarda artırılıp azaltılması ile incelenmiştir. Tablo 4.1'de verilen bir satır ele alınacak olursa; değişen değerler oldukça, diğer değerler sabit kalmış ve bu durum kombinasyonlar ile devam etmiştir. Maddelerce ayrıntılı olarak anlatılan bu kayıtlar;

- **seed:** 50 olarak belirlenmiş ve sabit bir değerdir. Tüm kullanıcıların test verileri yapay zekâ işlemlerinde bu rassallık değeri ile kullanılmıştır. Tüm test değerlerini tamamlayan kullanıcılar farklı bir seed değeri ile farklı bölümler oynamaya başlarlar.
- **speed:** Asıl değeri 1.0 olan değer, oynanış süresince Renga'nın hızını belirleyen özelliğidir 0.8 ve 1.2 değerleri ile kullanıcılardaki oynanış durumu gözlenmek istenmiştir. Rengeçler'e ulaşmak Renga'nın ilerlemesinde gerekmektedir.
- **gravity:** Asıl değeri 10.0 olan değer, oynanış süresince Renga'nın sabit bir doğrultuda bulunması yerine düşme hareketini sağlamaktadır. 12.0 ve 8.0 değerleri ile kullanıcıdaki etkileşimleri incelenmiştir. Bu değer sebebiyle Renga zemine çarpacak olursa oyun sonlanacaktır. jump değeri ile ters orantılı bir fayda zarar eğrisine sahiptir. Kullanıcı oyun esnasında ekrana dokunarak Renga'yı zeminden korumaktadır.
- **distance:** Asıl değeri 1.6 olarak belirlenmiştir. Rengeçler arasındaki mesafenin değerini belirtmektedir. 1.0 ve 3.0 değerleri ile farklılaştırılmıştır. Hız değeri ile ters orantıda olup zorluk değeri değiştirebilmektedir.
- **jump:** Asıl değeri 4.0 olarak kullanılmıştır. 5.5 ve 2.5 ikili değerleri ile farklılıkların etkisi kullanıcılarda gözlenmiştir. Kullanıcı düşey hareket eden Renga'nın yükseklik kazanmasını bu değerin büyüklüğü kadar sağlayabilir.

Rengeçler'in kesim noktalarının üstünden veya altından geçerken "jump" ve "gravity" değerlerinin önemi göz önünde bulundurulmuştur. Renga'nın geçemeyeceği kesim noktalarının meydana gelmesi engellenmiştir.

Kullanıcı yeterli veri düzeyini doldurdukça oyun içi değerler değişmiş ve her değer 5 farklı zamanda kullanıcı karşısına gelmiştir.

Tüm bu toplanan veriler sonrası yapay zekâ modellerinde, ilgili kullanıcının edinimi kolay, orta veya zor olarak sınıflandırılacaktır. Eğitim verileri oranına göre bilgiler doğrultusunda kullanıcıya sunulan başlangıç verilerinin değişimi ile kullanıcı oyunu daha verimli bir şekilde oynamaya devam edebilecektir. Bu çalışma toplanan veriler ve yapay zekâ ile karşımıza gelen edinimleri sunmaktadır. Sonuçları yorumlayabilmek için verilen çeşitli yapay zekâ yöntemleri ile analiz edilmiştir.

4.2 Renga Verilerinin İncelenmesi

Renga'dan hazırlanmış veriler; kullanıcı bazlı zorluk denetim ve ayarının yapılması, kullanılan yöntemlerde zorluğun takip edilmesini sağlayabilecek yeterliliktedir.

Tablo 4.2: Örnek oyun verileri

CİHAZ NUMARASI	RASTGELELİK	YATAY HIZ	YER ÇEKİMİ	ZIPLAMA	ENGELLER ARASI MESAFE	SKOR
ABCDEF123	42	1.5	10	2.5	3	26
ABCDEF123	42	1.8	7	2.5	3	9
ABCDEF123	42	1.5	4	2.5	3	16
ABCDEF123	42	1.2	7	2.5	3	10
ABCDEF123	42	1.5	7	3.5	1	18
ABCDEF123	42	1.5	7	4.5	3	4
ABCDEF123	42	1.5	7	3.5	5	2

Tablo 4.2'de örnek olarak verilen giriş setine göre, oluşturulacak yapay zekâ modelini anlatmadan önce, algoritma girdilerini oluşturan verilerden ve bu verilerin oyun içi özelliklerinden bahsedilmesi uygun olacaktır.

Rastgelelik: Yenileme değeri olarak tanımlanabilmektedir. Oyun içindeki tüm varlıklar bir rastgelelik değerine göre oluşturulmaktadır. Kullanıcı aynı rastgelelik değerinde her bir değerleme için en az 5 oyun oynamaktadır. Tablo 4.2 ele alındığında ortaya çıkan veri bu şekildedir;

$$4! = 24 \rightarrow 24 * 5 = 120 \quad (1)$$

(1) numaralı hesaplamaya göre yapay zekâ modellerince, bir kullanıcının, tam test ortamında minimum 120 sonuç çıktısı sunabilmesi mümkündür. Çıkan sonuçlar ise bir farklı rastgelelik değerinde, farklı oyun içi değerleriyle, zorluk denetimine yeni veriler oluşturabilmektedir.

Yatay Hız: Oyun başlangıç anından itibaren, saniyenin ekran yenileme hızına karşı aldığı yoldur. Tablo 4.2'de "YATAY HIZ" değerlerine göre saniyede 60 kare (FPS) için;

$$Yatay\ Hız = 0{,}016 * 1{,}5 = 0{,}024 \quad (2)$$

(2) numaralı hesaplamada her kare için 0,024 birim değişiklik meydana gelmektedir. Bu değerin artırılması, oyundaki zorluk algısını artıracaktır. Yapay zekâ modelindeki girdilerden biri olan yatay hız ağırlığının artması skor durumuna etki etmekte ve kullanıcıyı kolay sınıfına iletmektedir.

Yer Çekimi: Oyun başlangıç anından itibaren, Renga nesnesine atanmış "Rigidbody" özelliği ile etki eden fiziksel düşüş kuvvetidir. Tablo 4.2'de "YER ÇEKİMİ" değerlerine göre Saniyede 60 kare için;

$$Yer\ Çekimi = 0{,}016 * 7\ = 0{,}112 \quad (3)$$

(3) numaralı hesaplamada her kare için 0,112 birim değişiklik meydana gelmektedir. Bu değerin artırılması, zıplama değeriyle ters etkide ve dengeli olmalıdır. Değerin artması zorluğu artıracaktır, ancak gereğinden fazla düşük olması oyunu imkânsız duruma sokacaktır.

Zıplama: Kullanıcının etkileşimde bulunduğu hareketin değeridir. Kullanıcı her tıkladığında, Renga 2.5 birim kadar dikey pozisyonda konum artıracaktır. Bu hareket esnasında yer çekimi de düşüş hareketine devam etmektedir. İvmesi

tamamlanan Renga tekrar aşağıya doğru düşey hareket etmeye başlar. Değerin gereğinden fazla olması en düşük kazanımı dahi imkânsız oyuna yönlendirebilecektir.

Engeller Arası Mesafe: Rengeçler arası mesafe birimidir. Rastgelelik değerine göre bir sütunda, iki renk parçasıyla oranlanan Rengeçler arasında 3 birimlik mesafe bulunmaktadır. Değerin artışı oyunu kolaylaştıracaktır, azaltılması ise zorluk algısını kullanıcı deneyiminde önemli derecede farklılaştırabilmektedir. Ancak kullanıcıların da deneyimlerine göre optimize edilmiş uzaklığın yakınlığı ve uzaklığı göreceli bir düzey oluşturabilmektedir.

Skor: Oyuncunun bir oyun sürecindeki en yüksek kazanımını belirlemektedir. Böylelikle, geçilebilen Rengeç sayısını da temsil etmiş olmaktadır. Zorluk algısını belirleyen en önemli değerdir. Yapay zekâ modeline aktarılırken girdi olarak kullanıldığında, 5 oynanışa ait sonuçlar dâhil edilmiştir.

Bu veriler çalışma sonuç bölümünde farklılaştırılarak kullanıcılara sunulduğunda, genel kullanıcı kitlesinin durumu hakkında bilgiler de bulunacaktır.

Oyun tasarımı esnasında paket programların yanı sıra pek çok farklı teknik alandan fayda sağlanmıştır. Bir sonraki bölümde bu platformlar kısaca anlatılmıştır. Yöntemler ve bulgular alanında ise kullanılan yapay zekâ yöntemleri ele alınmış ve bunların sonuçları paylaşılmıştır. Sonuçlar bölümünde analizler ile Renga oyununa ait verilerin bu çalışmada nasıl bir etki bırakabildiğinin ilk adımı atılmaya çalışılmıştır. Geleceğe yönelik çalışmaların dahlinin neler olabileceği, yine bu bölümde irdelenmiş ve aktarılmıştır.

4.3 Programlama Dilleri ve Platformlar

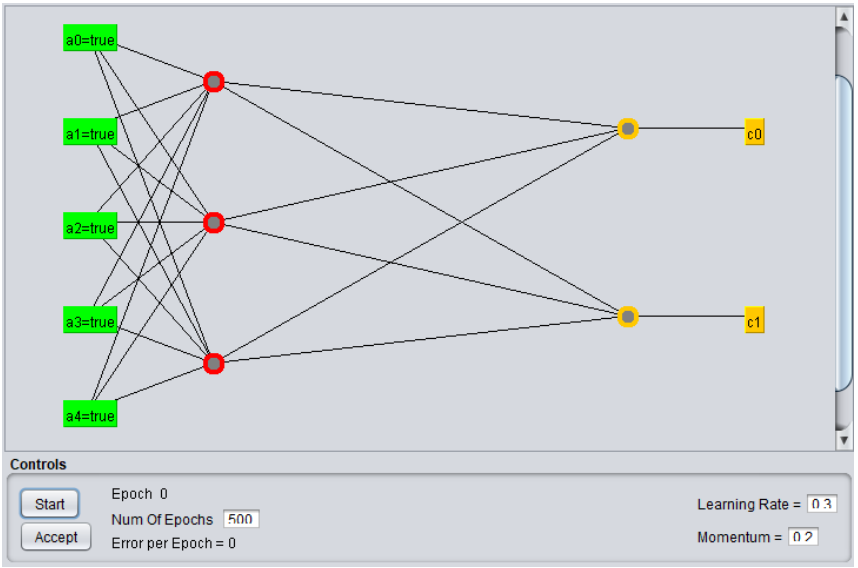
Çalışma süresince pek çok teknolojik platform ve programlama dili üzerinde araştırmalar yapılmıştır. Ancak aşağıda bahsedilen konular Renga isimli oyun çalışmaları için bir ilk adım oluşturulmuştur.

4.3.1 Python

Nesneye yönelik bir programlama dili olan python, diğer programlama dillerine göre daha yüksek verimlilikle çalıştığı için yapay zekâ işlemlerinde kullanılan önemli bir platform olarak görülebilmektedir. NumPy, SciPy, scikit-learn ve matplotlib kütüphaneleri yapay zekâ sınıflama algoritmaları için kullanılmaktadır. (Joshi, 2017).

4.3.2 Weka

Weka, makine öğrenmesi için geliştirilmiş ve dünya üzerinde fark edilen problemlerin çözümünde kullanılması düşünülen makine öğrenmesi yöntemlerinin uygulanabildiği bir sanal tezgâh olarak nitelendirilmiştir. Yeni Zelanda'nın Hamilton şehri, Waikato Üniversitesi'nde geliştirilmiştir (Holmes, 1994). Güncellemelerle desteklenen program Şekil 4.1'deki gibidir. Görsel, tek gizli katmanlı bir yapay sinir ağları modelinin Weka üzerindeki modelini belirtir.



Şekil 4.1: Weka; örnek çift katmanlı bir YSA modeli

Renga'nın geliştirim sürecinde, YSA modelinde uygulanmak üzere örnek veri setleri oluşturulmuştur. Veriler Weka ile incelenmiş, test edilmeye çalışılmıştır.

Oyun tamamlandıktan sonra Rapid Miner'ın sonuç analizleri ve işleyişinde kullanıcı deneyimi ölçütlerinin daha verimli olacağı gözlenmiştir. Çalışma Rapid Miner üzerinden sürdürülmüştür. Ancak Weka açık kaynaklı olup Rapid Miner ise birtakım ileri düzey araştırma sonuçları için ücretli bir sistem sunmaktadır. Rapid Miner'ın sunduğu özellikle bu çalışma için yeterli görülmüştür.

4.3.3 Rapid Miner

Makine öğrenmesi, veri madenciliği gibi bilimsel çalışmaların yürütülebileceği bir yazılım platformudur. Kendi akış arayüzü bulunan sistem, verilerin hazırlanabileceği ve yapay zekâ yöntemlerince işlenebileceği tüm düzeyleri görselleştirerek kullanıcıya büyük resim sunabilmektedir. Çıkan sonuçların model dökümü sayesinde veri analizleri daha rahat okunabilmektedir. (Rapid Miner, 2013)

4.4 Yöntemler

Çalışma içerisinde yaklaşık 1566 adet veri bulunmaktadır. Test kullanıcıları oyun içi değerlerin her biriyle en az 5 kere oynamıştır. Bu durumda bir kişiden gelen toplam veri sayısının 125 olduğu öngörülmüştür. Oluşan toplam kayıtın 10 bine yaklaştığı bilinmektedir. Kayıtlı verilere bakıldığında ise yapay zekâ modellerinde kullanılmak üzere verilerin bir ön araştırmaya dayandırılması uygun görülmüş ve bu ölçüt dahilinde veriler değerlendirilmiştir. Sonuç olarak kişisel değil kitlesel bir veri seti ortaya çıkarılmıştır.

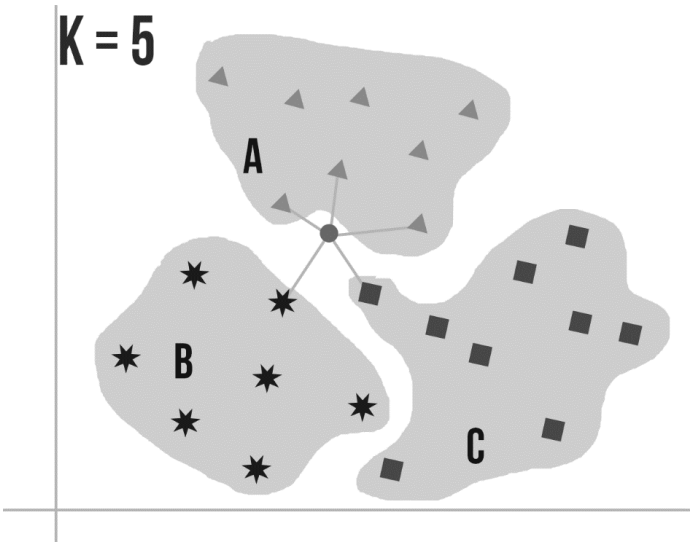
Kullanılan yapay zekâ yöntemlerinin ortak özelliği, hepsinin birer sınıflandırma algoritması kategorisinde yer almasıdır. Bu algoritmalar aynı zamanda danışmalı öğrenme kategorisinde yer almaktadır. En Yakın k-Komşu, Rassal Orman, Derin Öğrenme ve Yapay Sinir Ağları algoritmaları, düzenlenen verilerce, Rapid Miner plaformunda çalıştırılmıştır. Sonuçlar karışıklık matrisi ile incelenerek aktarılmıştır. Her yöntemin kendi içindeki değişkenleri baz alınarak başarı ve hata katsayıları belirlenmiştir. Bu çıkarımlar, oyun içi verilerin, belirtilen kullanıcı kitlesi için bir seviye tanılama belirtmektedir. Çıkarımlar, bu verilerin sonucunda kullanıcılara aktarılarak sonuçlar gözlemlenmiştir.

4.4.1 En Yakın k-Komşu

Sınıflandırmanın temelinde yatabilecek kullanımın en basit ifadesi, benzerlikten yola çıkmaktır. k-EYK (en yakın k komşu) yöntemi de bu durumun en basit örneği olarak bilinmektedir. Eğitim verileri için gereken kayıtların aktarılması ile kartezyen düzlemde kümelenmeler sağlanarak ilk aşama oluşturulmaktadır. Seçilecek k değerine göre düzleme yeni gelecek veriler beklenmektedir. Bu verilerin de k değerince düzlemde yakın olduğu k kadar nokta ele alınarak hangi kümeye ait olduğu aktarılacaktır. Algoritma için sözde kod ifadesi şu şekilde açıklanabilir:

1. *Eğitim verilerinin yüklenmesi*
2. *K değerinin belirlenmesi*
3. *Bilinmeyen her bir veri için yapılacak işlemler:*
    a. *Bilinmeyen veriyi yerleştir*
    b. *k değerince ona en yakın yerleşmiş eğitim verilerini bul.*
    c. *Seçilen k eğitim verilerinin çoğunluğuna göre bilinmeyen veriyi etiketle*
4. *Bitir.*

Şekil 4.2 görselinde, kümelenmiş bir alan giren yeni verinin, k=5 için yerleştirileceği değer, A kümesi olacaktır. Ancak k değerinin değişimi, en yakın komşu sayısının dengesini değiştirebilmektedir. Bu sebeple k-EYK yönteminin en etkili parametresi k komşuluk sayısı olarak bilinmektedir. k değerinin değişimi yeni gelen bir elemanın değerleri doğrultusunda, yakın komşuluğu fazla olan kümenin içerisine dahil olacaktır.



Şekil 4.2: k=5 için yeni bir değerin A-B-C kümelerince sınıflandırılması

4.4.2 Rassal Orman

Rassal orman yöntemi çalışma verilerinin uyumluluğu düşünüldüğünde daha farklı bir bakış açısı sağlayabilecek bir ağaç algoritmasıdır. Çünkü veriler içerisinde; farklı kişilerin aynı verilerle elde edeceği farklı sonuçların barınması veya oyuna alışma sürecinde 5 kez oynanacak verilerin aynı kişilerde açık puan farklılıkları göstermesi gibi durumların rassal bakış açısıyla modellenmesi sunulabilecek verimli bir ağaç algoritmasında önemli veriler ortaya çıkarmaktadır.

Algoritmanın sözde koduna aşağıda değinilmiştir:

1. ***m*** *kadar özellik içerisinden bir* ***k*** *özelliği seçilir*
    a. ***k << m***
2. ***k*** *özelliği için en iyi ayrım noktasının hangi değer olacağını bularak bir* ***d*** *düğümü oluşturulur.*
3. *Bu düğüm en iyi değerli iki farklı düğüm veya etikete gidene kadar devam eder.*
4. *1 ve 3 adımları boyunca özellikler kadar devam eder.*
5. *4. adıma kadar oluşturulması istenen n kadar ağaç modeli oluşturulur.*

Rassal Orman algoritmasındaki en iyi ayrım noktasını bulan değerlerin olduğu her bir ağaç karar ağaçlarını belirtmektedir. Aşağıdaki sözde kod ise karar ağaçları algoritmasını belirtmektedir:

1. *Veri setinde bulunan sonuca etki edecek özellikleri, ağacın kök düğümüne yerleşir.*
2. *Eğitim setlerini alt setlere böl. Bölünen alt setlerin her biri, bir özellik için aynı değeri taşımalıdır.*
3. *Yaprak düğümleri bulunana kadar 1 ve 2 numaralı adımlar özyinelemeli olarak devam eder.*

Karar ağaçlarında kök düğüme yerleştirilecek verinin hangisi olabileceğinin hesaplanması için seçilebilecek yöntemler mevcuttur. En bilinen özellik hesaplamalarından biri Bilgi kazanımıdır. Sınıflandırmaya etki eden verilerin hangisinin olduğu eğitim verilerince belirlenerek kök düğüm bu özelliğe göre

bölünebilmekte ve belirli değerin ne olduğuna göre ikiye ayrılarak farklı düğümlere yaprak düğümü kalıncaya kadar özyinelemeli olarak aktarılmaktadır. Bu ölçüt için kullanılan hesaplama yöntemi entropi olarak bilinmektedir.

$$H(X) = E\big(I(X)\big) = \sum_{i=1}^{n} p(Xi) \log_2 \left(\frac{1}{p(Xi)}\right) \quad (4)$$

(4) numaralı denklemlerin açıklamasına bakıldığında;

- I(X) X özelliğinin bilgisini gösterir,
- p(Xi) = Pr(X=Xi) X özelliğinin götürdüğü ihtimali belirtmektedir.

Denkleme uygulanan özelliklere göre çıkan değer entropi değeridir. Sonuçta K veri kümesinin bilgi kazancı (5) numaralı denklemde belirtilmiştir;

$$Kazanç(K, A) = Entropi(K) - \sum P(V)\ Entropi(T(v)) \quad (5)$$

- V: A özelliğinin değeri
- P(V): A özelliğine yönelik olasılığın hesabıdır. Sonucun a özelliğince değerini belirtir.

Algoritmaya kazandırılarak oluşan ağaç modelinin pek çok özellik bazında uygulanması ile rassal orman algoritması çeşitliliği daha fazla sağlayarak daha farklı yorumlayıcı sonuçlar verebilir. Bundaki en önemli etken ise verilerin ileriye dönük şekillendirilmesi ve işlenmesinde gerekebilecek çeşitliliğin nasıl sağlanması gerektiğini gösterebilmesidir.

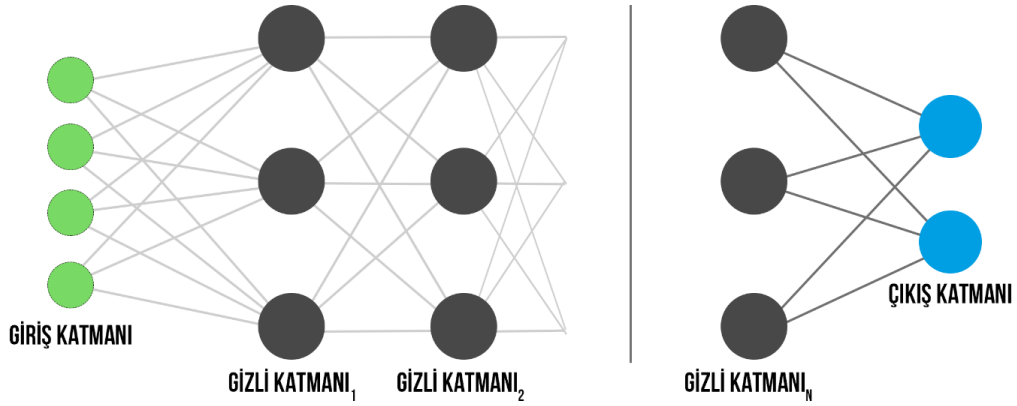
4.4.3 Derin Öğrenme

Çok katmanlı Yapay Sinir Ağı (YSA) modelinin makine öğrenmesindeki algoritmalarını belirtmektedir. YSA'nın en küçük birimi algılayıcılardır. Çeşitli algoritmalar sonucu geliştirilmesi mümkün olmuştur. XOR kapılarının gerçekleştirilememesi sonrasında geri yayılımlı ağların ortaya çıkması gibi zaman içinde yetersiz kalması, süreci derin öğrenmenin ortaya çıkmasına kadar taşımıştır. En büyük farkı ise daha çok işlem yapabilme gücü ile, çok katmanlı mimarilere sahip olmasıdır. (Yao, 1999)

Çalışmada geri yayılımlı öğrenmenin etkisini görebilmek ve her eğitim çağı boyunca eğitime yeni verilerin katılmasının ne gibi değişimlere denk olabileceğinin görülebilmesi amacıyla kullanılmıştır. Bu sayede ileriye dönük hazırlanacak veri setleri elde edilirken fayda sağlayacaktır.

4.4.4 Yapay Sinir Ağı

Bir yapay sinir ağını oluşturan en küçük yapı sinir hücresidir. Hepsi birleşerek, oluşturdukları sistem içinde ortaya çıkan, sonucu bir aktivasyon fonksiyonuna göre ayırt edilerek aktarılmaktadır. Şekil 4.3'teki gibi mimarisi katmanlara dayanır ve her katman birbirini izleyen algılayıcılara bağlıdır.



Şekil 4.3: YSA model örneği

Model üzerinde giriş katmanındaki özellikler gizli katmanlar aracılığıyla işlemlere tabi tutulmaktadır:

$$z_j^i = w_j^i\, x + b_j^i \tag{6}$$

- z: özellik
- w: ağırlık
- b: eğilim

Yapay sinir ağları (6) numaralı denklem ile perceptronlar arası iletişimi sağlamaktadır. Sonuçlar etkinleştirme fonksiyonları üzerinden elde edilmektedir.

Derin öğrenme yöntemi için, çalışmada kullanılan aktivasyon fonksyionları tanh (tanjant hiperbolik) ve maxout fonksiyonlarıdır;

$$\max(w_1^T + b_1, w_2^T + b_2) \quad (7)$$

(7) numaralı denklem, maxout fonksiyonunu formülünü ifade etmektedir. ReLU (Rectifier Linear Unit) ve Leaky ReLU fonksiyonlarının genelleştirilmiş hâlidir. ReLU fonksiyonundaki perceptron kayıpları yoktur. İleri beslemeli ağlarda formüle göre iletilen en yüksek değeri kullanır.

$$TanH = g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (8)$$

(8) numaralı denklem, tanh aktivasyon fonksiyonunu ifade eden formülü gösterir.

4.5 Bulgular

Veri setleri içerisinde birden fazla kullanıcıdan gelmiş veriler bulunmaktadır. Uygulama için değerler aynı olsa da puan farklılıkları, sonuçları etkileyen temel rolü oynamıştır. Bunun en büyük sebebi "score" isimli veri içeriğine göre kayıtların zor, orta ve kolay etiketlerine ayrılmasıdır. "score"u etkileyen en önemli etmenlerin ise yer çekimi ve hız değişkenlerinin olduğu anlaşılmıştır. Rassal orman algoritması için bu iki değerin yer aldığı karar ağacı yapısı paylaşılmıştır.

Tüm kullanıcılar hayatlarında bu oyunu ilk defa oynamış olarak kabul edilmiştir. Bulgular, çalışmaya Renga oyunu için bir seviye tanılama bilgisi sunmaktadır. Bu veriler, kullanıcı kitlesine daha verimli bir bölüm ve oyun içi dinamiği sunmak için kullanılmıştır.

Aşağıdaki bulgular belirtilen yapay zekâ yöntemlerinin uygulanmasıyla elde edilmiştir. Çalışmada yer alan bulgular, çalışılan veri kümelerinin en verimli sonuçları baz alınarak paylaşılmıştır. Pek çok parametre denenmiş ancak en verimli ve başarılı olan sonuçlar paylaşılmıştır.

Tablo 4.3: Derin öğrenme, tanh fonksiyonlu bulgu

DERİN ÖĞRENME	ZOR	ORTA	KOLAY	TUTARLILIK
TAHMİNİ ZOR	39	48	60	26.53%
TAHMİNİ ORTA	63	110	146	34.48%
TAHİMİNİ KOLAY	11	22	149	81.87%
TEKİL BAŞARIM	34.51%	61.11%	41.97%	
PARAMETRELER	BAŞARI: %45,99 | EĞİTİM ORANI: 0.6 | AKTİVASYON: tanh			

Tablo 4.3'te, anlaşıldığı üzere tanh aktivasyon fonksiyonunun kullanımında %50'lik değerin aşağısında kalan bir sonuç kümesi ortaya çıkmıştır. En başarılı tahmin değerinin kolay etiketi olduğu anlaşılmıştır. %81.87'lik değer ile toplam 182 tahminden 149'u başarılıdır. Ancak en doğru tahmin "orta" etiketi ile yapılan sonuçlarda başarı sağlanmıştır.

Tablo 4.4: Derin öğrenme, maxout fonksiyonlu bulgu

DERİN ÖĞRENME	ZOR	ORTA	KOLAY	TUTARLILIK
TAHMİNİ ZOR	9	20	0	31.03%
TAHMİNİ ORTA	12	9	10	29.03%
TAHİMİNİ KOLAY	92	151	345	58.67%
TEKİL BAŞARIM	7.96%	5.00%	97.18%	
PARAMETRELER	BAŞARI: %56,02 | EĞİTİM ORANI: 0.6 | AKTİVASYON: maxout			

Bu derin öğrenme yönteminde ise maxout fonksiyonu kullanılmıştır. Aşırılıkların önüne geçilerek elde edilen yakın değerler boyunca başarı ortamı sağlanmıştır. Bu değerin %56,02 ile ortalamanın üzerine çıktığı görülmektedir. En başarılı tahminlerin ise kolay etiketinde olduğu görülmüştür. Tablo 4.3 ve Tablo 4.4 bulgularında, 0,8 oranlı eğitim test verileri sunulmuştur. Derin öğrenme algoritması, ortalama %50 civarında bir başarı oranı sunmaktadır.

Yapay sinir ağlarından, daha fazla katman içermesi özelliği ile ayrılan derin öğrenmenin yanında, verilerin 2 gizli katmanlı yapay sinir ağı modelince de sonuçlar

oluşturulması uygun görülmüştür. Bu bölümde yapay sinir ağları işlemleri de yer almıştır.

Tablo 4.5: k-EYK yöntemli, k=3 ayraçlı bulgu

k-EYK	ZOR	ORTA	KOLAY	TUTARLILIK
TAHMİNİ ZOR	26	26	0	50.00%
TAHMİNİ ORTA	73	138	0	65.40%
TAHİMİNİ KOLAY	21	40	0	0.00%
TEKİL BAŞARIM	31.86%	82.22%	0.00%	
PARAMETRELER	BAŞARI: %50,62 | EĞİTİM ORANI: 0.8 | k = 3			

Tablo 4.5'te verilen, k-EYK yönteminde göze çarpan en önemli bulgu; "kolay" etiketine yönelik hiçbir tahminde bulunamamasıdır. Bu da verilerin, k=3; k=5 için yeterli yakınsak noktalara sahip olmadığını göstermektedir. Ancak başarı ortalaması %50'nin üzerine çıkarak verimli bir sonuç verebilecek bir düzeye yakınlaşabileceği anlaşılmıştır.

Tablo 4.6: k-EYK yöntemli, k=5 ayraçlı bulgu

k-EYK	ZOR	ORTA	KOLAY	TUTARLILIK
TAHMİNİ ZOR	42	50	0	45.65%
TAHMİNİ ORTA	78	154	0	66.38%
TAHİMİNİ KOLAY	0	0	0	0.00%
TEKİL BAŞARIM	35.00%	75.49%	0.00%	
PARAMETRELER	BAŞARI: %60,49 | EĞİTİM ORANI: 0.8 | k = 5			

Tablo 4.6, k=5 için ortaya çıkan sonuçlarda daha verimli bulgular göze çarpmaktadır. Sonuç olarak kolay etiketi yine bulunamamasına rağmen, tekil başarımlarda orta düzeyin belirginliği önemli bir sonuç vermiştir. Genel olarak k-EYK sisteminin, değişken roller için farklı veriler kazanılması sebebiyle tutarsız verilerde kullanılmaması gerektiği düşünülmektedir.

Tablo 4.7: Rassal orman yöntemli doğrusal örneklem ayraçlı bulgu

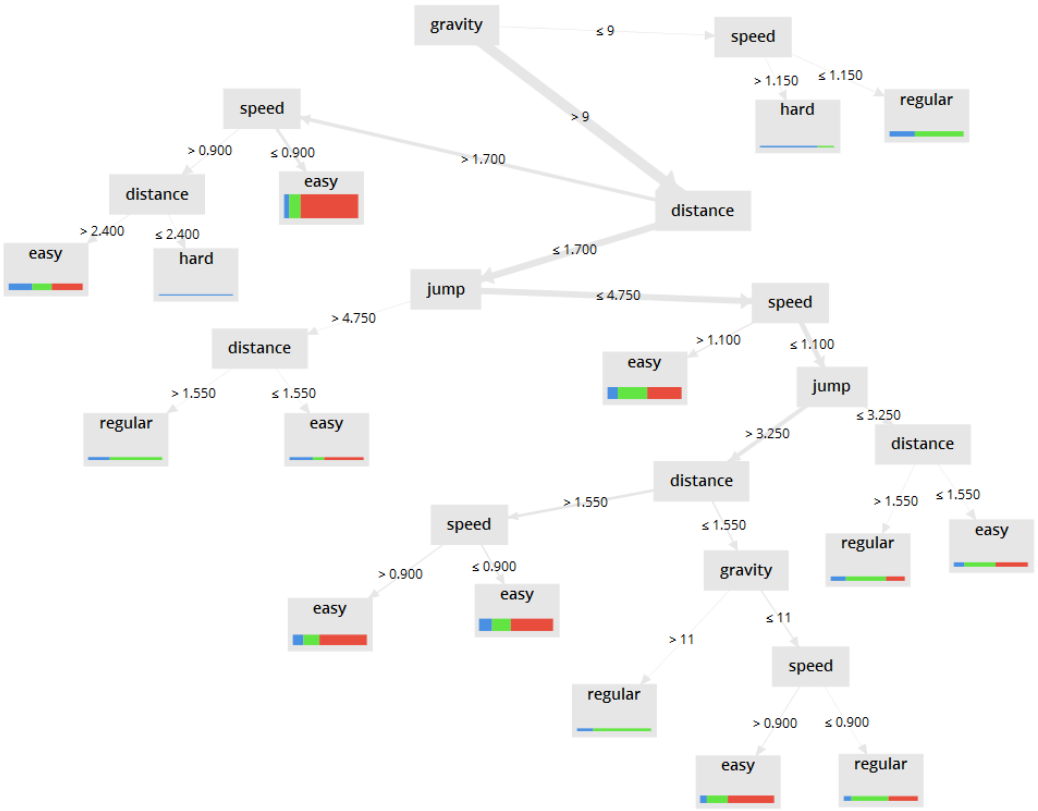
RASSAL ORMAN	ZOR	ORTA	KOLAY	TUTARLILIK
TAHMİNİ ZOR	4	1	0	80.00%
TAHMİNİ ORTA	21	42	0	66.67%
TAHİMİNİ KOLAY	95	161	0	0.00%
TEKİL BAŞARIM	3.33%	20.59%	0.00%	
PARAMETRELER	BAŞARI: %14,20 | EĞİTİM ORANI: 0.8 | ÖRNEKLEM: Doğrusal			

Tablo 4.7'de Rassal Orman için algoritmik parametrelerin değişiminde, bulgulara yönelik etkili bir farklılığa rastlanmamıştır. Bu sebeple örneklemler farklılaştırılmış ve bulgular doğrusal-karmaşık örneklemlere göre ayrıştırılmıştır. Tahminler içerisinde tutarlılık olmasına rağmen, doğrusal örneklemlerde (linear sampling) başarım oranı %14,20 olarak belirlenmiştir. Kolay etiketine hiç yansımamış değerler de doğrusal örneklemlerin eksiğidir.

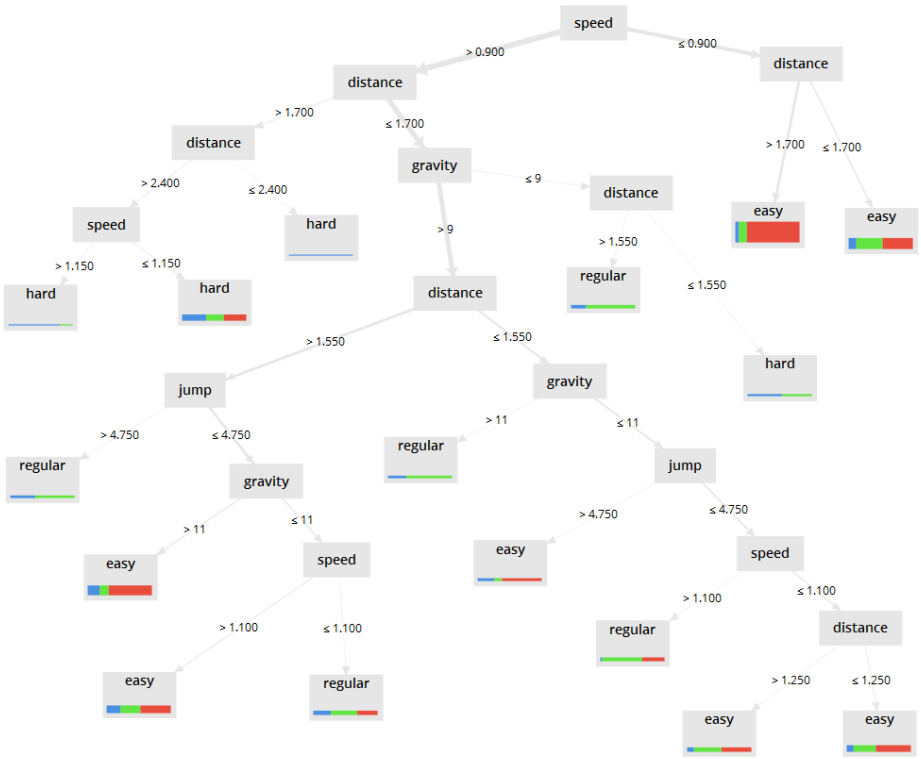
Karar ağaçları yerine kullanılan rassal orman yöntemi farklı özellikleri üzerinden sonuçların nasıl etkilenebileceği veya yönlendirilebileceğini modellemek için kullanılmıştır. Ancak etki alanları birbirinden farklı sonuçlara yol açmadığı için ayrıca yer verilmemiştir. Tablo 4.8'de yer alan sonuçlar, 0,8 eğitim-test oranlı olup, örneklemlerin karışık bir şekilde algoritmaya sunulmasıyla sağlanmıştır. Elde edilen en verimli ve başarı oranı en yüksek bulgulardır.

Tablo 4.8: Rassal orman yöntemli karışık örneklem ayraçlı bulgu

RASSAL ORMAN	ZOR	ORTA	KOLAY	TUTARLILIK
TAHMİNİ ZOR	4	1	0	80.00%
TAHMİNİ ORTA	13	40	17	57.14%
TAHİMİNİ KOLAY	27	48	174	69.88%
TEKİL BAŞARIM	9.09%	44.94%	91.10%	
PARAMETRELER	BAŞARI: %67,28 | EĞİTİM ORANI: 0.8 | ÖRNEKLEM: Karışık			



Şekil 4.4: Yer çekimi özelliği üzerinden karar ağacı yöntemi



Şekil 4.5: Yatay hız özelliği yönünden karar ağacı modeli

İki ağaç modeli için de ortaya çıkan en önemli sonuç; oyun seviyesini tanılamaya götüren süreçte, belirtilen değerler ile tekrar denenmesiyle karşılaştırılabilecek zorluk algısını tekrar kullanıcılara sunarak kesin veri sağlayabilmektir.

Tablo 4.9: YSA yöntemi ile elde edilen bulgular

YSA	ZOR	ORTA	KOLAY	TUTARLILIK
TAHMİNİ ZOR	0	0	0	0.00%
TAHMİNİ ORTA	15	28	12	50.91%
TAHİMİNİ KOLAY	29	61	179	66.54%
TEKİL BAŞARIM	0.00%	31.46%	93.72%	
PARAMETRELER	BAŞARI: %63,89 | EĞİTİM ORANI: 0.8 | 2 GİZLİ KATMAN			

YSA En verimli sonucu veren ikinci algoritma olmuştur. Ancak belirtilen veri setlerince zor etiketinin tahmin edilemediği görülmüştür. 2 gizli katman kullanılan algoritmada eğitim oranı Tablo 4.9'da görüldüğü gibi 0,8 olarak ele alınmıştır. Karmaşık örneklem (shuffled sampling) değeri kullanılmıştır.

5. SONUÇLAR ve ÇIKARIMLAR

Bu tez çalışmasında, farklı yöntemler ve farklı parametrelerle yapılan sınıflandırmada sonsuz sürekli bir oyun olan Renga'nın kullanıcıları üzerinde kolaylık ve zorluk bilgileri edinilmiştir. Seviyeleme öncesi kitleye uygun bir kullanıcı-zorluk değerleri tanılanmıştır. Kitleye uygun bir sonuç kümesi elde edilmiştir.

Tablo 5.1: Seviyeleme sonucunun kullanıcılara etkisi

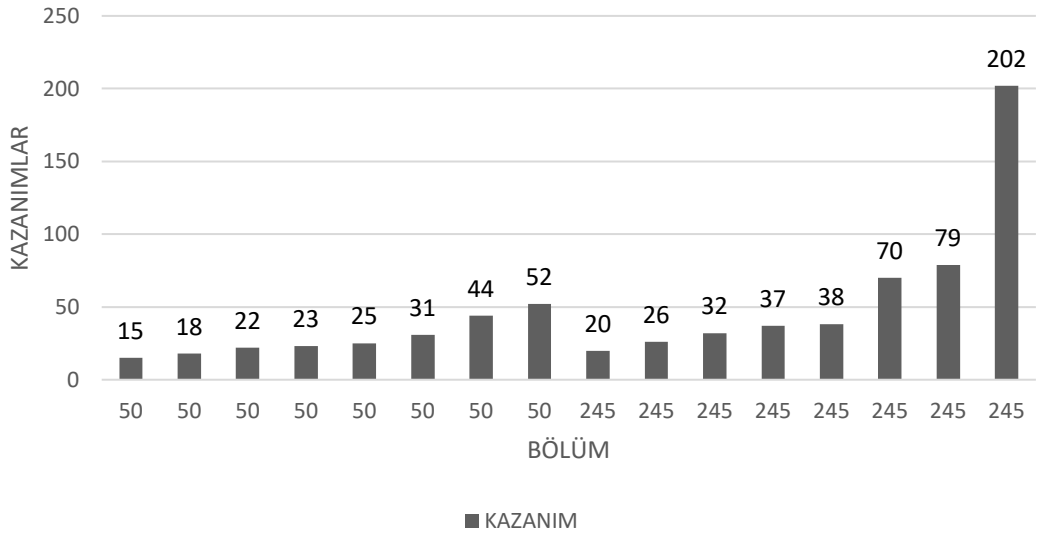
KULLANICI	BÖLÜM	YATAY HIZ	YER ÇEKİMİ	ENGEL ARALIĞI	ZIPLAMA	KAZANIM
USER1	245	0.8	10	1.8	2.7	202
USER2	50	0.8	10	1.6	4	89
USER4	245	0.8	10	1.8	2.7	77
USER3	245	0.8	10	1.8	2.7	76
USER5	245	0.8	10	1.8	2.7	38

Tablo 5.1'e bakıldığında; 245 numaralı bölümlerin kullanıcıların kazanım etkisini büyük oranda artırdığı görülmektedir. Sonuç olarak, bu değerlere göre kullanıcıların, oyun içi kazanımlarını sağlarken oyun deneyimini en verimli şekilde edinebilmişlerdir. Kullanıcı bazında yapılacak çalışmalar sayesinde ise kişisel sonuçlar elde etmeye yönelik verilerin toplanabilmesinin önü açılmıştır. Çıkan sonuçlara göre kullanıcılara sunulan oyun içi değerlerin olumlu bir sonuç verdiği ve keyif duygusunun da en iyilendiği gözlenmiştir. Geliştiricilere ise bu deneyimi sunabilecekleri kısa bir yol gösterilmiştir. Bu sayede zamandan kazanç sağlanabilmiş ve oyun iyileştirmeleri hız kazanmıştır.

Tablo 5.2: "User1" kullanıcısının seviyeleme öncesi bilgileri

KULLANICI	BÖLÜM	YATAY HIZ	YER ÇEKİMİ	ENGEL ARALIĞI	ZIPLAMA	KAZANIM
USER1	50	1	12	1.6	4	18
USER1	50	0.8	10	1.6	4	14
USER1	50	1	10	1.6	2.5	8
USER1	50	1	10	3	4	4
USER1	50	1.2	10	1.6	4	23
USER1	50	1	8	1.6	4	10
USER1	50	1	10	1.6	5.5	11
USER1	50	1	10	1	4	14

Tablo 5.2'ye bakılacak olursa, "User1" isimli kullanıcının seviyeleme öncesi edindiği kazanımların, kullanıcı için yeterli bulunmadığı anlaşılmıştır. Ancak Seviyeleme sonrası oyundan alınan keyif daha fazla artmış ve kullanıcının kazanımlarına yüksek oranda bir getiri sağladığı gözlemlenmiştir.



Şekil 5.1: User1 kullanıcısının oyunda elde ettiği kazanımlar

Şekil 5.1 incelendiğinde, 50 ve 245 değerli bölüm ve her bölüm için 8 adet oyun verisi bulunmaktadır. 50 değerli bölüm, seviye tanılamada kullanıcıların test değerlerini barındıran zorluklara sahiptir. Şekilde, bu test verileri, User1 kullanıcısının ilgili bölümde elde ettiği en iyi çıktılardır. 50 değerli bölümün yapay zekâ yöntemlerince sağladığı tanılamalar sonrasında, 245 değerli yeni bir bölüm kullanıcıya sunulmuş ve bulgulara göre oyun içi değerler şekillendirilmiştir. Bu bölümde elde edilen ilk 8 kazanımın tamamı, `kolay` olarak etiketlenmiştir. Kazanımların bölümlere göre aritmetik ortalaması alınıp oranlandığında;

$$\sum_{i=1}^{8} Kazanım_i \Big/ \sum_{j=1}^{8} Kazanım_j \tag{9}$$

- **i:** 50 değerli bölümlerin kazanım değerleri
- **j:** 245 değerli bölümlerin kazanım değerleri

Olmak üzere, (9) numaralı denkleme göre User1 kullanıcısının elde ettiği kazanımların %45 oranında artış sağladığı gözlenmiştir.

Bu oyun üzerinden yapılabilecek değerlemeler, diğer platform oyunlarına da uygulanabilmek için bir araç niteliğindedir. Çalışmaların ilerleyen süreçlerinde farklı tür ve zenginlikte platform oyunlarında uygulanabilmesinin önü açılmıştır.

Çalışmada kullanılan en iyi yöntemin ise Rassal Orman algoritması olduğu görülmektedir. Ayrıca Rassal Orman algoritmasının en iyi dağılımı sağlaması, başarı oranının yanında, sınıflandırma için de verimli bir yöntem oluşturduğu anlaşılmaktadır. Sunduğu karar ağaçları modeli, yol gösterici nitelikte olup yüksek fayda sağlayabilmektedir.

Kullanıcıların, oyun kimliklerinin içinde yer alacağı bir veri seti sınıflandırmasında, kullanıcı tiplerinin de ortaya çıkabileceği farklı sınıflandırma problemlerinin, yeni çalışmalar ortaya koyabileceği öngörülmektedir. Bu sayede oyunun belirli değerlerle birkaç temel kullanıcı üzerinden oynatılarak veri toplanması, seviyeleme çalışmalarını daha verimli bir hâle getirebilmektedir. Çalışmada ise bu durum kitlesel bir değer taşımaktadır.

Her oyun içi değerlere sahip uygulamaların da bu sürece dâhil edilebilmesi tezin evrenselliğini ortaya koyabilmektedir. Çünkü basit bir adım olarak atılan bu çalışma, olağanüstü sıradan (hypercasual) oynanışlara sahip oyunlar için bir ön adım oluşturabilmektedir.

6. KAYNAKLAR

Aponte, M. V., Levieux, G., Natkin, S. “Measuring the level of difficulty in single player video games”, *Entertainment Computing*, 2(4), 205-213, (2011).

Baghdadi, W., Eddin, F. S., Al-Omari, R., Alhalawani, Z., Shaker, M., & Shaker, N. “A procedural method for automatic generation of spelunky levels”, *European Conference on the Applications of Evolutionary Computation*, pp. 305-317, (2015).

Csikszentmihalyi, M., “Flow and the psychology of discovery and invention”, *Harper Perennial, New York*, 39, (1997).

Ferreira, L., & Toledo, C. “A search-based approach for generating angry birds levels”, *Computational intelligence and games*, (2014).

Galway, L., Charles, D., & Black, M. “Machine learning in digital games: a survey.”, *Artificial Intelligence Review*, 29(2), 123-161, (2008).

Gee, J. P., “What video games have to teach us about learning and literacy”, *Computers in Entertainment*, 1(1), 20-20, (2003).

Haas J. K., “A History of the Unity Game Engine”, (7 Kasım 2018) https://digitalcommons.wpi.edu/iqp-all/3207, (2014).

Hendrikx, M., Meijer, S., Van Der Velden, J., & Iosup, A., “Procedural content generation for games: A survey”, *ACM Transactions on Multimedia Computing, Communications, and Applications*, 9(1), 1, (2013).

Holmes, G., Donkin, A., & Witten, I. H., “WEKA: A Machine Learning Workbench”, *Working Paper,* 94/9, (1993).

Hunicke, R., “The case for dynamic difficulty adjustment in games.”, *2013 ACM SIGCHI International Conference on Advances in computer entertainment technology* (pp. 429-433), (2005).

Goodfellow, I., “Maxout Networks.”, *arXiv:1302.4389*, (2005).

Johnson, D., & Wiles, J., “Computer games with intelligence. Fuzzy Systems”, *The 10th IEEE International Conference*, Vol. 3, pp. 1355-1358, (2001).

Kaelbling, L. P., Littman, M. L., & Moore, A. W., “Reinforcement learning: A survey”, *Journal of artificial intelligence research*, 4, 237-285, (1996).

Kent, S. L., “The Ultimate History of Video Games: From Pong to Pokemon-TheStory Behind the Craze That Touched Our Lives and Changed the World [Digital Edition]”, *New York: Three Rivers Press*., (2001).

Koster, R., “Theory of fun for game design”, *O’Reilly Media*, Inc., (2013).

Mourato, Fausto, Próspero dos Santos, Manuel., “Measuring difficulty in platform videogames”, *4a Conferência Nacional Interacção humano-computador*, (2010).

Mawhorter, P., & Mateas, M., “Procedural level generation using occupancy-regulated extension.”, *Computational Intelligence and Games, IEEE Symposium*, pp. 351-358, (2010).

Millington, I., & Funge, J., “Artificial intelligence for games”, *CRC Press*, (2016).

Missura, O., & Gärtner, T., “Player modeling for intelligent difficulty adjustment”, *International Conference on Discovery Science*, (pp. 197-211). Springer, Berlin, Heidelberg, (2009).

Mourato, F., dos Santos, M. P. & Birra, F., “Automatic level generation for platform videogames using genetic algorithms”, *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, p. 8, (2011).

Nakamura, J. Csikszentmihalyi. M., “The concept of flow”, *CR Snyder & SJ Lopez, Handbook of positive psychology*, 89-105, (2002).

Pedersen, Chris, Togelius, Julian, Yannakakis, Georgios N., “Modeling player experience in Super Mario Bros”, *CIG2009-IEEE Symposium on Computational Intelligence and Games,* (2009).

Persson, M., “Infinite mario bros”, *Online Game*, (2008).

Joshi P., “Artificial Intelligence with Python”, *Pact Publishing*, (2017).

Rouse III, R., “Game design: Theory and practice”, *Jones & Bartlett Publishers*, (2008).

Y. Sarıca, M. Çetin, "Dynamic Difficulty Methodologies in Games", *ICAT, 7th International Conference on Advanced Technology & Sciences*, 28 Nisan-1 Mayıs 2018, Antalya, TURKEY.

Schmidhuber J., “Deep learning in neural networks: An overview”, Volume 61, 85-117, (2015).

Shaker, N., Yannakakis, G. N., & Togelius, J., “Towards Automatic Personalized Content Generation for Platform Games”, *AIIDE*, (2010).

Simon B., “*Indie Eh? Some Kind of Game Studies*”, *Technoculture, Art and Games (TAG)*, (2012).

Sinclair, Jeff, "Feedback control for exergames", *Theses: Doctorates and Masters*, (2011).

Smith, G., Treanor, M., Whitehead, J., & Mateas, M., “Rhythm-based level generation for 2D platformers”, *4th International Conference on Foundations of Digital Games* (pp. 175-182), (2009).

Soucy P. and Mineau G. W., “A simple KNN Algorithm for Text Categorization”, *Proceedings 2001 IEEE International Conference on Data Mining*, (2002)

Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E., “Online adaptation of game opponent AI in simulation and in practice”, *4th International Conference on Intelligent Games and Simulation*, pp. 93-100, (2003).

Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E., “Difficulty scaling of game AI”, *5th International Conference on Intelligent Games and Simulation*, pp. 33-37, (2004).

Togelius, J., Karakovskiy, S., Koutník, J., & Schmidhuber, J., “Super mario evolution. Computational Intelligence and Games”, *CIG 2009. IEEE Symposium,* 156-161, (2009).

togelius J., Kastbjerg E., Schedl D., Yannakakis N., “*What is Procedural Content Generation?Mario on the borderline*”, (2011).

V. der Linden, R. R. Lopes and R. Bidarra, “Designing procedurally generated levels”, *second workshop on Artificial Intelligence in the Game Design Process*, (2013).

Wheat, D., “Dynamically adjusting game-play in 2D Platformers using Procedural Level Generation”, Thesis: *Individual retirement savings behavior: evidence from Malaysia*, (2012).

Wiegand, R. P., Liles, W. C., & De Jong, K. A., “Analyzing cooperative coevolution with evolutionary game theory. Evolutionary Computation”, *CEC'02, Vol. 2*, 600-1605, (2002).

Woodcock, S., Laird, J. E., & Pottinger, D., “Game AI: The state of the industry. Game”, *Developer Magazine*, 7-8, (2000).

Yao X., “Evolving artificial neural networks”, *Volume: 87* , Issue: 9, (1999).

Yannakakis G., Togelius J., “A Panorama of Artificial and Computational Intelligence in Games”, *IEEE Transactions*, (2014).

Zhang, Y., He, S., Wang, J., Gao, Y., Yang, J., Yu, X., & Sha, L., “Optimizing player's satisfaction through DDA of game AI by UCT for the Game Dead-End”, *Natural Computation, Sixth International Conference,* 4161-4165. IEEE, (2010).

ÖZGEÇMİŞ

Adı Soyadı	:	Yunus SARICA
Doğum Yeri ve Tarihi	:	Van, 09.14.1993
Lisans Üniversite	:	Pamukkale Üniversitesi
Y. Lisans Üniversite	:	Pamukkale Üniversitesi
Elektronik posta	:	ysarica11@posta.pau.edu.tr
İletişim Adresi	:	Gerzele Mah. Geriz Cad. Yağmurkent Sitesi A Blok 1 Numara Merkezefendi/Denizli

Konferans yayını:

Y. Sarıca, M. Çetin, "Dynamic Difficulty Methodologies in Games", ICAT, 7th International Conference on Advanced Technology & Sciences, 28 Nisan-1 Mayıs 2018, Antalya, TURKEY.