**at, taslak, çizim içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**T.C**

**DOKUZ EYLÜL ÜNİVERSİTESİ**

**FEN FAKÜLTESİ**

**BİLGİSAYAR BİLİMLERİ BÖLÜMÜ**

**BİTİRME PROJESİ RAPORU**

**Abdülhalik GÜNEŞ**

**Alper KUM**

**Mirza ÖZMEN**

**Zuhal GÜRGEN**

**Danışman: Can ATILGAN**

**Mayıs, 2023**

**İZMİR**

**ÖZET**

Bu proje raporu, Unity, C# programlama dili ve MLAgents kullanarak 2 boyutlu bir kule savunma oyununun geliştirilmesini açıklamaktadır. Projemiz için, amacın düşman dalgalarına karşı harita üzerinde belirli bir noktayı savunmak olduğu bir kule savunma oyunu tasarlamaya karar verdik. Oyun bir Windows ortamında tasarlanmış ve Unity kullanılarak C# programlama dili ile yazılmıştır. Genel olarak, bu proje oyun geliştirmede AI ve makine öğreniminin potansiyelini vurgulayan ilgi çekici ve zorlu bir kule savunma oyunu sunmayı amaçlamaktadır.

**Anahtar kelimeler:** Makine Öğrenimi, MLAgents, Kule Savunma Oyunu, Derin Destekli Öğrenme…

**ABSTRACT**

This summary explains the development of a 2D tower defense game using Unity, C# and MLAgents. For this project, we’ve decided to make a game where the task is to defend a point against waves of enemies. The game has been designed in a Windows environment and written in C# programming language via Unity. In general, this project aims to present an interesting and challenging tower defense game that denotes the potential of AI and machine learning.

**Keywords:** Machine Learning, MLAgents, Tower Defense Game, Deep Reinforcement Learning

**ÖZET.………………………………………………………………………………....2**

**ABSTRACT.……………………………………………………………………….....3**

**İÇİNDEKİLER……………………………………………………………………....4**

**ÇİZELGELER…………………………………………………………………….....5**

**1. GİRİŞ……………………………………………………………………………..7**

**2. OYUN GENEL BAKIŞ……………………..……………………………………8**

**2.1. OYUN TANITIMI………………………………………………………8**

2.1.1 Intro……………………………………………...……………..9

2.1.2 How To Play Ekranı..…………………………………………..9

2.1.3 Düşmanlar………………………..…………………………......10

2.1.4 Haritalar…………………………….………………...…….….12

2.1.5 Kuleler………………………..…………………...……….…..15

2.1.6 Kullanım Kılavuzu.....................................................................16

**3. OYUN TASARIMI……………………………………………………..…….….15**

**3.1 MLAgents ve Derin Destekli Öğrenme………………………………..16**

3.1.1 Unity’de Nasıl Kullanılır?..........................................................19

3.1.2 Proje Kapsamındaki Amacı ve Entegrasyon Süreci…………...20

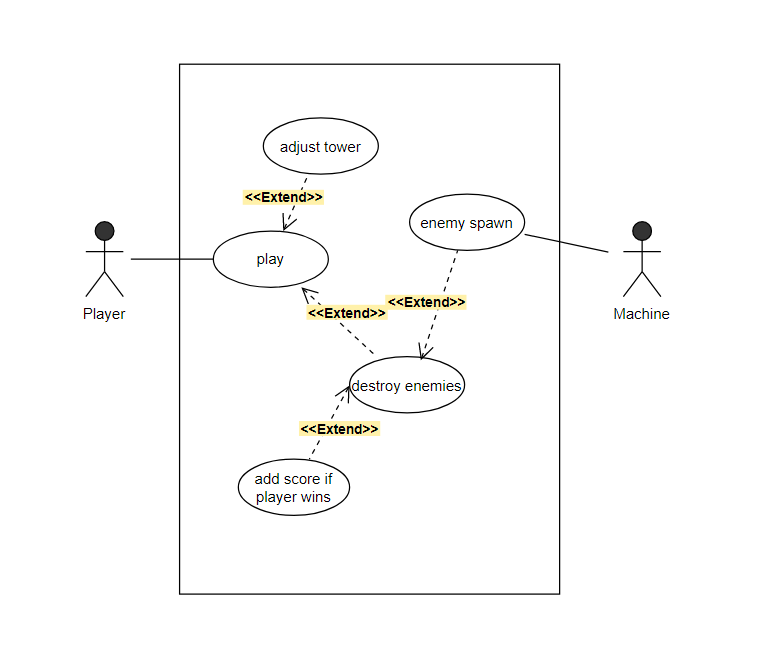
**3.2 Oyun İçi Satın Alma Sistemi……………………………………………23**

**4. SONUÇ……………………………………………………………………...……23**

**KAYNAKÇA…………………………………………………………………...….…24**

**ÇİZELGELER**

**Use Case Diyagramı:**

****

Bu diyagram, iş süreçlerinin yönetilmesi aşamasında ihtiyaç duyulan tüm fonksiyonları, bu fonksiyonları tetikleyecek aktörleri, fonksiyonlardan etkilenecek aktörleri ve fonksiyonlar arasındaki ilişkileri göstermek amacıyla seçilmiştir.

**Class Diyagramı:**

diyagram, metin, plan, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu diyagram, projemiz içindeki sınıflar, sınıfların tuttuğu özellikler ve bu sınıflar arasındaki ilişkileri basit ve anlaşılır bir şekilde görselleştirmeye yatkın olduğu için seçilmiştir

**Swimlane Diyagramı:**

**metin, diyagram, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Bu diyagram, oyuncu ve oyun arasındaki etkileşimleri, hangi etkileşimin neyi tetiklediğini ve oyun mekanizmalarını en sade biçimde göstermek üzere seçilmiştir.

1. **GİRİŞ**

Dungeon Vengeance, gelen düşman dalgalarına karşı üslerini savunan oyuncuların stratejik düşünme ve karar verme becerilerini zorlamak için tasarlanmış hem kişisel bilgisayarlarda hem de mobil cihazlarda oynanabilen bir kule savunma oyunudur. Oyunda, her biri kendi tasarımına ve farklı çevresel zorluklara sahip 5 benzersiz harita bulunmaktadır. Benzer şekilde 3 farklı düşman türü ve 3 farklı kule bulunmaktadır. Bu 3 farklı düşman türünün her biri, oyuncu için farklı zorluklar sunan benzersiz özelliklere ve yeteneklere sahiptir. Ek olarak, her düşmanın kendi saldırı tarzı vardır, bu da oyuncuların tek bir strateji ile tüm düşmanlara karşı savunma yapmasını zorlaştırır. Benzer şekilde, bu 3 farklı kulenin her biri, oyuncuya düşman saldırılarına karşı savunma yapabilmesi için çeşitli seçenekler sağlayan benzersiz yeteneklere sahiptir. Her kulenin güçlü ve zayıf yönleri vardır, bu da oyuncuların başarılı olmak için doğru kule kombinasyonunu inşa etmesini zorunlu kılar. Oyunumuzda amacımız, oyuncuların gelişen düşman taktiklerine uyum sağlayabilecekleri ve yenilikçi stratejiler geliştirebilecekleri dinamik ve büyüleyici bir oyun deneyimi sunmak için MLAgents'ın Derin Yardımlı Öğrenme algoritmasından yararlanmaktı. Hedefimiz, düşman gönderme modelini makine öğrenimi ile tasarlayarak öngörülemezliği ortaya çıkarmaktı. Ancak bazı aksilikler yaşadık ve istediğimiz sonuca ulaşamadık. Yine de sonuç olarak, bu projede, oyuncuları stratejik düşünmeye ve baskı altında hızlı kararlar almaya zorlayan çekici bir kule savunma oyunu yaratmayı amaçladık. Ek olarak, oyun içi satın alma sistemini oyuna derinlik ve karmaşıklık katacağını düşünerek tasarlamaya çalıştık fakat zaman kısıtlamaları nedeniyle dahil etmemeye karar verdik. Bunun yerine, normal oyun ilerlemesiyle kazanılabilecek çeşitli ve yükseltilebilir kuleler geliştirmeye odaklandık. Kule çeşitliliğine ve oyuncu becerisine odaklanarak oyun içi satın alımlara dayanmayan ilgi çekici ve ödüllendirici bir oyun deneyimi yaratmayı amaçladık. Bu raporun, oyunun geliştirme süreci ve tasarımına ilişkin içgörü sağlamasını umuyoruz.

1. **OYUN GENEL BAKIŞ**
   1. **Oyun Tanıtımı**

Bu projede, çeşitli düşmanlara karşı koymaya dayanan Dungeon Vengeance adlı bir kule savunma oyunu tasarlamayı planladık. Temel amacımız, kullanıcının istediği herhangi bir stratejiyi uygulayarak, düşmanların ilerlediği yolun yanı sıra çeşitli savunma kuleleri seçerek belirlenen bir noktaya düşmanın ulaşmasını engellemektir. Farklı sağlık seviyesi, saldırı teknikleri ve savunma yeteneği gibi özelliklere sahip çeşitli düşmanlar tasarladık. Örneğin bazı düşmanlar daha hızlı hareket ederken diğerleri daha yavaş olabilir ama daha fazla cana sahip olabilir, bazı düşmanlar belirli kule tiplerine karşı daha dirençli olabilirken bazıları belirli kuleler karşısında zayıf kalabilir. Oyuncu daha fazla puan kazandıkça ve belirli bir seviyeye ulaştıkça, düşman dalgaları güçlenir, bu da onlara karşı savunmayı giderek zorlaştırır ve oyuncunun stratejilerini buna bağlı olarak ayarlaması ve gerektiğinde daha yüksek seviyeli savunma kulesi inşa etmesi gerekir. Oyuncu, daha güçlü saldırı yeteneği elde etmek için hali hazırdaki kulelerini verdiği hasar, menzil ve diğer özelliklerinin daha yüksek olduğu kulelere yükselterek daha iyi bir savunma sağlayabilir bunun için düşmanları yok ettiğinde kazandığı parayı kullanmalıdır. Başlangıç noktasından hedef noktasına ulaşan saldırgan üssü yok ettiğinde oyuncu oyunu kaybeder ve oyun biter. Kullanıcı tekrar oyuna başlayabilir veya oyundan çıkmayı seçebilir. Oyuncu, savunmalarının etkinliğini en üst düzeye çıkarmak ve üssünü bu düşman türlerine karşı başarılı bir şekilde savunabilmek için hangi kuleyi seçeceğini, nereye yerleştireceğini ve ayrıca hangi kulelere yatırım yapacağını dikkatlice düşünmelidir. Bu, kaynakların dikkatli bir şekilde planlanması ve yönetiminin yanı sıra çeşitli düşman türlerinin güçlü ve zayıf yönlerinin anlaşılmasını gerektirir. Sonuç olarak Dungeon Vengeance, başarılı olmak için stratejik planlama ve kaynak yönetimi gerektiren ilgi çekici ve zorlu bir oyun deneyimi sağlar. Aralarından seçim yapabileceğiniz çeşitli düşman türleri ve savunma kuleleri ile oyuncular, gelen düşman dalgalarına karşı savunmanın en etkili yolunu bulmak için farklı stratejiler keşfedebilir.

* + 1. **Intro**

metin, ekran görüntüsü, bilgisayar oyunu, video oyunu yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

* + 1. **How To Play**

Aşağıdaki fotoğraflar oyunu ilk kez oynayan kullanıcılar için tasarlanmış olan how to play ekranını gösterir.

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

* + 1. **Düşmanlar**

Oyunda3 farklı düşman vardır. Her biri farklı özelliklere sahiptir.

ekran görüntüsü, çizgi film içeren bir resim

Açıklama otomatik olarak oluşturuldu

Priest

çizgi film, ekran görüntüsü, Animasyon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Runner

kırpıntı çizim, ekran görüntüsü, Çizgi film, Animasyon içeren bir resim

Açıklama otomatik olarak oluşturuldu

Warrior

Aşağıdaki tablo her bir düşmanın can, kazandırdığı altın değeri, hız gibi özelliklerini gösterir**.**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

* + 1. **Haritalar**

ekran görüntüsü, yeşil içeren bir resim

Açıklama otomatik olarak oluşturuldu

map for level 0

plan, dikdörtgen, devre içeren bir resim

Açıklama otomatik olarak oluşturuldu

map for level 1

devre içeren bir resim

Açıklama otomatik olarak oluşturuldu

map for level 2

devre, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

map for level 3

harita, devre, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

map for level 4

ekran görüntüsü, dijital kompozit oluşturma, 3B modelleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

map for level 5

* + 1. **Kuleler**

Oyunda her biri birbirinden farklı özelliklere sahip 3 kule çeşidi vardır.

çizgi film, kırpıntı çizim, Animasyon, grafik içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ranger Tower

çizgi film, kırpıntı çizim, ekran görüntüsü, simge, sembol içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bomber Tower



Laser Tower

Aşağıdaki tablo her bir kulenin verdiği saldırı zaman aralığını gösterir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**2.1.4 Kullanım Kılavuzu**

Oyuncu, oyunu fare ile kontrol eder.

* Main Menuden doğrudan Play butonuna tıklayarak oyunu oynamaya başlayabilir.
* Pause Menude sesi açıp kapatabilir.
* How To Play butonuna tıklayarak oyunun nasıl oynanacağını öğrenebilir.
* Escape butonuna tıklayarak oyunu duraklatabilir.
* Kaybederse Restart butonuna tıklayarak oyunu yeniden başlatabilir.
* Duraklama menüsünde Exit butonuna tıklayarak oyundan çıkabilir.
* Bir seviyeyi kazandıktan sonra, Next butonuna tıklayarak bir sonraki seviyeye geçebilir.

1. **OYUN TASARIMI**

**3.1 MLAgents ve Derin Destekli Öğrenme**

MLAgents, yapay zekanın bir dalı olan makine öğrenimini oyuna entegre etmek suretiyle oyuna rastgelelik, yeni meydan okumalar ve genel olarak daha doyurucu ve zorlayıcı bir oynanış tecrübesi elde etmek için konulabilir. Normal bir yapay zekaya kıyasla hangi şartlarda neler yapabileceğini öğrenen, bu bağlamda oyun ilerledikçe kendini geliştiren yapay zekaların entegrasyonunu sağlayan MLAgents paketinin, bu projede kulelerin durumuna göre hangi düşmanları nasıl göndereceğini belirlemek üzere kullanılmak amacı ile oyuna entegrasyonuna çalışılmıştır.

MLAgents, deep reinforcement learning (derin destekli öğrenme) algoritması kullanarak çalışmaktadır. Bu algoritmanın özeti, bu algoritmayı kullanan bir mekanizmaya sahip olan otonom özellikli bir objenin bulunduğu ortam üzerinde gözlem yapıp bu gözlemlere bağlı olarak hangi durumda nasıl bir davranış gerçekleştireceğini öğrenmesi ve bu öğrenme işlemini ödül ve ceza yöntemiyle harmanlayarak ve sürekli olarak gerçekleştirerek öğrenme işlemini kalıcı hale getirmesi, böylece aynı şartlar sağlandığı sürece aynı davranışa gitmesinin sağlanmasıdır.

Biraz daha açık anlatmak gerekirse bu algoritmanın “reinforcement” kısmı, fiziksel bağlamda şu maddeler ile anlatılabilir:

* Otonom özellikli obje, ortam ve koşullar aynı anda aynı yerde bulunur.
* Obje, ortam üzerinde sensörlerini kullanarak gözlem yapar ve bu gözlemlerin değerlerini ham veri olarak alır (kesikli ya da kesirli sayı). Bu veriler sıcaklık, pozisyon gibi veriler olabilir.
* Bu veriler kullanılarak bu verileri veren obje ile herhangi bir etkileşime karşılık olarak bir ödül sinyali alınır. Bu sinyal negatifse o koşul veya nesne ile o etkileşimden kaçınma, pozitifse o koşul veya nesne ile o etkileşime girme durumu güçlendirilmiş olur, ve benzeri bir duruma her girildiğinde o durum kümülatif ödül sistemiyle desteklenir.
* Bu algoritmanın öğrenme kısmı “ajan” adı verilen bir nesnenin problem üzerinde deneme yanılma yapması ile oluşur. Problem bu algoritmada Markov Karar İşlemi (MDP) olarak modellenir (MDP de Markov Zincirlerine dayanır).

**diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 1: Destekli öğrenme döngüsü (Unity MLAgents Toolkit/Background Machine Learning)

metin, ekran görüntüsü, yazı tipi, grafik içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 2: Destekli öğrenme döngüsü (İngilizce Vikipedi/Deep Reinforcement Learning)

Reinforcement learning algoritmasının en önemli özellikleri dinamik programlamayı destekliyor olması ve en baştan bir matematiksel modelin bilindiğini varsaymamasıdır, bu açıdan teorik olarak sınırsız modele ulaşılabilmektedir.

Reinforcement learning (destekli öğrenme) algoritması, her ne kadar bu projede tek ajan ile kullanılmış olsa da, birden fazla ajana sahip olan versiyonları da (multi-agent reinforcement learning, veya çok ajanlı destekli öğrenme) mevcuttur. Bu algoritmalarda birlikte öğrenen ve adapte olan birden fazla ajan vardır. Bu tarz bir destekli öğrenme algoritması daha büyük oyun projelerinde birbirine rekabetçi veya destekçi ajan grupları oluşturmak için kullanılabilir.

Bu algoritmanın “derin” olan kısmı ise, çok fazla sayıda gözlem ile (dolayısıyla çok fazla sayıda veri ile), oldukça karmaşık fonksiyonların çözümünün öğrenilebiliyor olmasıdır, yani deep reinforcement learning, standart reinforcement learning’in çok büyük verilerle ve veri setleriyle çalışan versiyonu olarak düşünülebilir. Bu durum reinforcement (takviye) kısmına derin sinir ağı eklenmesiyle elde edilmektedir. Doğal olarak, normal bir reinforcement learning algoritmasına kıyasla çok daha fazla sayıda koşul ve durum ile etkileşime girilip çok fazla ödül sinyali alınır, hatta bazı durumlarda bu etkileşim birden fazla ortamda gerçekleşebilir (örn. sıcak veya soğuk ortamlarda aynı objeyle karşılaşıldığında gerçekleşmesi istenen eylemler farklı olabilir).

Derin öğrenme algoritmaları genellikle çok (2’den fazla) katmanlı algoritmalar oldukları için kendilerinden istenen sonucu daha yüksek isabetle tuttururlar. Derin öğrenme algoritmalarının bu avantajını kullanan derin destekli öğrenme algoritmaları, elde edilen gözlemin ne ifade ettiğini birden fazla katman içerisinde gözlemler ve bu sayede öğrenme işlemi daha yüksek performansla gerçekleşir.

Derin takviyeli öğrenme algoritmaları yüksek boyutlu girdilerde (örneğin kameradan alınan bir fotoğrafın analizi) kullanılmaktadır. Bu algoritmalar model bazlı ve modelsiz (yani dinamik programlama bağımlı) olmak üzere ikiye ayrılırlar.

**3.1.1 Unity’de Nasıl Kullanılır?**

MLAgents kütüphanesi ise, derin destekli öğrenmenin Unity ortamında gerçekleştirilmesini sağlayan modül olarak karşımıza çıkmaktadır. Bizzat Unity destekli olan bu modül, altyapısında Python programlama dilini kullanmakta, böylece Python ile C# dillerinin birleşimini sağlayarak şunları sağlamaktadır:

* Python’un güçlü ellerinden olan makine öğreniminin Unity’e kolay entegrasyonu
* Hızlı bir öğrenme süreci ile daha dinamik bir oynanış
* Teknik anlamda çok dilli ve yüksek kapasiteli oyunların geliştirilmesi
* C# dilinin tanıdık ve kolay syntax’ı ile Python’da kod yazmaya gerek kalmaksızın gelişmiş makine öğrenimi sistemlerinin oturtulması (çok gelişmiş sistemlerde ekstra Python kodları yazılabilir de)

MLAgents, Unity projelerinde çalışabilmek için PyTorch kütüphanesine ihtiyaç duymaktadır. Ayrıyetten kurulum için protobuf kütüphanesinin 3.20.3 sürümüne ve yapay zeka denince akla gelen ilk kütüphanelerden olan numpy kütüphanesinin 1.21.2 sürümüne indirgenmesi gerekmiştir (mlagents 0.29.0 kütüphanesi o sürümü desteklemektedir). PyTorch içerisine gömülü olan TensorBoard ile ödül/ceza durumu görselleştirilir, bu durum da Unity ortamının hiperparametreleri üstünde etkiye sahiptir.

MLAgents’ın gözlem edinme ve gözlem sonucunda elde edilen sonuçları işleme işlevlerinin gerçekleştirilebilmesi için kullanılan başlıca fonksiyonlar CollectObservations, OnActionRecieved ve AddReward (veya SetReward) fonksiyonlarıdır. Bunlardan CollectObservations, MLAgents.Sensors kütüphanesine ihtiyaç duyar ve destekli öğrenmenin “gözlem” kısmını temsil eder. OnActionRecieved ise belirlenen büyüklükteki eylem dizisinden gelen değerlere göre objenin farklı eylemler yapmasını sağlar. Parametre olarak aldığı ActionBuffers’ın iki dizisi bulunur (DiscreteActions ve ContinuousActions), bunlardan genellikle biri kullanılmakla beraber yerine göre ikisi birden de kullanılabilir.

AddReward fonksiyonu, kümülatif ödül sistemleri için kullanılan ödül fonksiyonu olup, kendisine parametre olarak verilen miktarda ödülü arttırır (negatif değer alırsa azaltır). SetReward ise ödül miktarını belirler.

MLAgents kütüphanesi kullanılarak ajan nesnenin bulunduğu ortam üzerinde yapabileceği gözlemler ve gerçekleştirebileceği aksiyonlar tam (integer) veya kesikli (float) sayılardan oluşmaktadır. Bu gözlemler başta bir başka nesnenin pozisyonu, bir değişkeninin tuttuğu değer gibi sayılabilir değerler alan özelliklerdir. Ajan nesnenin gerçekleştirebileceği aksiyonlar, ajan nesnenin çevresine, gerçekleştirebileceği hareketlerin yönüne vs. bağlıdır.

metin, ekran görüntüsü, diyagram, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 3: Unity MLAgents blok diyagramı/çok ajanlı bir örnek (Unity MLAgents Toolkit/Deep Reinforcement Learning)

* + 1. **Proje Kapsamındaki Amacı ve Entegrasyon Süreci**

Bu proje bağlamında MLAgents kütüphanesi aracılığıyla düşmanların doğması işleminin yapay zeka tarafından halledilmesi amaçlanmıştır. Bu işlemi yapacak olan yapay zekanın kuleleri aşabilecek ve varış noktasına ulaşıp oyuncuya zarar verebilecek olan düşman gönderme düzenini keşfedip gönderme işlemini ona göre yapması hedeflenmiştir. Bunun için varış noktasına ulaşma ve varış noktasını yok etme (oyuncunun canının 0’a inmesi) durumları için ayrı seviyelerde ödüllendirmeler yapılarak yapay zekanın eğitilmesine çalışılmıştır.

MLAgents’in daha iyi çalışabilmesi için Grid Sistemi, grid sistemiyle bağlantılı olarak da düşmanların hareketini sağlayan bir pathfinding sistemi de oyuna entegre edilmiştir. Bu sistemle diagonal (8-yönlü) hareket sistemi mümkün kılınmıştır.

ekran görüntüsü, metin, 3B modelleme, dijital kompozit oluşturma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu Grid Sisteminin amacı üzerinde bulunan her oyun objesinin bilgisini üzerinde tutarak gerekli verilerin MLAgents için kullanılması, kullanımını kolaylaştırması ve Pathfinding Sistemi için gerekli temellerin atılmasıydı. Bu Grid Sisteminde MLAgents ile geliştirilen yapay zekanın kulelerin nerede olacağı bilgisini alacak ve ona göre bir düşman türü yollayacaktır ve böylece yaşanan sonuçlarda sahip olduğu veriler üzerinden daha iyi bir gelişim sağlayacaktır. Aynı zamanda farklı bölümlerin geliştirilmesinde gridlerin gerekli büyüklükte olmasını ayarlayabilmektir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Aynı zamanda bu Pathfinding Sistemi ile yapay zekanın hangi yoldan gideceği tercihini yaparken ve bu tercihlerini daha iyi seçmesinde yardımcı olmaktır. Böylece hem gerekli yol bulma sistemi kullanılacak hem de MLAgents’ın yolunu seçmesinde yardımcı olacaktır. Ayrıca hangi yolların üzerinden gidebileceğini ayarlarken kolaylık sağlayarak tasarımcılara yardımcı olabilmektir.

Oyunda MLAgents’ın kullanabileceği şekilde dört farklı state durumu oluşturulmuştur:

* Spawning: Oyunda tekil düşman çağırma işlemi gerçekleşmektedir.Hızlı bir öğrenme süreci ile daha dinamik bir oynanış
* SpawnAGroupOfRunner: Runner tipi düşmanları grupça çağırma işlemi gerçekleşmektedir.
* SpawnAGroupOfWarrior: Warrior tipi düşmanları grupça çağırma işlemi gerçekleşmektedir.

MLAgents’ın bu oyun projesindeki entegrasyonunda, CollectObservations fonksiyonu kalan düşman sayısını döndüren bir fonksiyonun değerini kontrol etmektedir. Akabinde OnActionReceived fonksiyonunda 0 veya 1 değeri almasına göre işlem yapmaktadır. Bu işlem alınan değerin 0 olması halinde Runner grubu, 1 olması halinde de tekil Warrior düşman çağırması şeklinde gerçekleşmektedir.

Bu oyun projesi için MLAgents’taki ödüllendirme kıstası düşmanın yolun sonuna ulaşması prensibine dayanmaktadır. Yolun sonuna ulaşan her bir düşman oyuncuya zarar verecek şekilde tasarlanmış olan bu sistem, oyuncu zarar gördüğünde veya oyuncunun merkezi yıkıldığında (canı 0 olduğunda) makine öğrenimi sistemine ödül sinyali göndermektedir. Bu ödül sinyaline bağlı olarak düşman taraf, asker gönderme düzenini değiştirmektedir. Bu ödülleri eklemek için AddReward fonksiyonu, sırayla 0.1 (oyuncunun merkezi hasar aldığında) ve 1.0 (oyuncunun canı 0 olduğunda) parametreleri verilerek kullanılmıştır.

Ancak donanımsal olduğu şüphe edilen, test aşamasında başlayıp daha sonra da farklı bilgisayarlarda ve projelerde çalıştırıldığı halde çözülememiş bazı problemlerden dolayı bu amaca ulaşılamamış olup, onun yerine hardcode mantığıyla kodlanmış ve dalgadan dalgaya güçlenen bir düşman gönderme düzeni oyuna eklenmiştir. Bu düzenin kod satırları, mevzuya açıklık getirilmesi açısından rapora eklenmiştir ve aşağıdaki gibidir:

Yukarıda görülen kod bloğunda, ilk iki dalga için tek bir tür düşman çağırma işlemi yapılmıştır. Üçüncü dalgadan itibaren enemySpawnIndex rastgele değişkeni aracılığıyla tekil (0, false) veya çoğul (1, true) düşman çağırma işlemleri yapılmıştır. Bu işlem, düşman türüne göre SpawnRunner ve SpawnWarrior fonksiyonlarıyla yapılmıştır ve her iki fonksiyon da tekillik veya çoğulluk durumunu belirten bir boolean parametre almaktadır. SpawnWarrior fonksiyonu

1. dalgadan itibaren çağırılmakta olup, SpawnRunner fonksiyonuyla çağırılabilecek runner tipi düşmanların sayısı 0’a ulaştığında warrior tipi düşman çağırma işlemini gerçekleştirmektedir.

**3.2 Oyun İçi Satın Alma Sistemi**

Oyun içi satın alma sistemleri genellikle birçok oyunda önemli, tamamlayıcı bir özellik olarak yer alır. Bu sistemler, oyuncuların, bu öğelerin nadirliğine dayalı olarak öğeleri rastgele satın almalarını sağlar. Bir oyun içi satın alma sisteminde öğelerin dağılımını belirlemek söz konusu olduğunda, her bir öğeye ayrı ayrı nadirlik atamak pratik olmayabilir. Potansiyel olarak yüzlerce benzersiz öğeye nadirlik atama görevi zor ve gereksizdir. Bu zorluğun üstesinden gelmek için, yaygın bir yaklaşım olan, bir gruptaki her öğenin eşit nadirliğe sahip olduğu farklı nadirlik grupları oluşturulur. Örneğin, "rare" bir grup 14 öğeden oluşuyorsa ve %28 seçilme şansı varsa, o grup içindeki her bir rare öğenin seçilme şansı %2 olacaktır. Biz bu oyunda, oyun içi satın alma sistemi ile kule yükseltmeyi ve bunu nadirlik kullanarak yapmayı planlamıştık. Fakat yeterli zaman olmaması sebebiyle bu sistemi yarıda bırakmak ve oyunun güncel halinden çıkarmak zorunda kaldık.

ekran görüntüsü, bilgisayar oyunu, video oyunu yazılımı, 3B modelleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Oyun içi satın alma sisteminin yarıda bırakılmış örnek görüntüsü

1. **SONUÇ**

Sonuç olarak, kule savunma oyunu projemiz, zorlu ve sürükleyici bir oyun deneyimi oluşturmak için Unity'de C# programlama dilinde MLAgents'ın Derin Destekli Öğrenme algoritması kullanılarak, oyuncuların savunma kulelerini seçerken ve yükseltirken göz önünde bulundurmaları için ek bir strateji katmanı ekleyecek akıllı ve dinamik bir düşman gönderme sistemi oluşturmamızı sağlayacaktı. Projenin şimdiki gidişatında bunu başaramamış bulunmaktayız, ancak bu durum oyunculara benzersiz ve ilgi çekici bir oyun deneyimi sunan bir oyun yaratmamıza engel olmamıştır. Nihai ürünle gurur duyuyoruz ve oyunculara saatlerce eğlence sağlama potansiyeline sahip olduğuna inanıyoruz. Gittikçe artan ve kısa sürede oyuncuyu daha çok düşman türüyle tanıştırıyor olması oyunu meydan okuyucu kılıyor ve farklı kuleler, düşmanlar ve arka plan müziği gibi çeşitli oyun özellikleri oyunu daha ilginç ve keyifli hale getiriyor.

KAYNAKÇA

[ML-Agents Overview - Unity ML-Agents Toolkit (unity-technologies.github.io)](https://unity-technologies.github.io/ml-agents/ML-Agents-Overview/#deep-reinforcement-learning)

[Machine Learning - Unity ML-Agents Toolkit (unity-technologies.github.io)](https://unity-technologies.github.io/ml-agents/Background-Machine-Learning/)

[PyTorch - Unity ML-Agents Toolkit (unity-technologies.github.io)](https://unity-technologies.github.io/ml-agents/Background-PyTorch/)

https://en.wikipedia.org/wiki/Deep\_reinforcement\_learning

[Markov decision process - Wikipedia](https://en.wikipedia.org/wiki/Markov_decision_process)

[Reinforcement learning - Wikipedia](https://en.wikipedia.org/wiki/Reinforcement_learning)

[Deep learning - Wikipedia](https://en.wikipedia.org/wiki/Deep_learning)

Rarity and Weighted Spawn Systems - <https://medium.com/geekculture/space-shooter-challenge-powerup-rarity-and-weighted-spawn-systems-13d439a5805e>