



KASTAMONU ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ



UNITY 2D OYUN PROGRAMLAMA

(UNITY 2D OYUN PROGRAMLAMA TEMELLERİ)

2021

LİSANS BİTİRME TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

METEHAN BULUT

DOÇ.DR.SALİH GÖRGÜNOĞLU

UNİTY 2D OYUN PROGRAMLAMA

Metehan BULUT

(174410040 / N.Ö)

Kastamonu Üniversitesi

Mühendislik ve Mimarlık Fakültesi

BİLGİSAYAR Mühendisliği Bölümü

Lisans Tezi Olarak Hazırlanmıştır

KASTAMONU

TEMMUZ 2021

TEZ ONAYI

METEHAN BULUT tarafından hazırlanan “**UNİTY 2D OYUN PROGRAMLAMA**” adlı tez çalışması Kastamonu Üniversitesi Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü Bitirme Tezi Komisyonu tarafından Bitirme Tezi olarak kabul edilmiştir.

Danışman

Doç.Dr. SALİH GÖRGÜNOĞLU
Kastamonu Üniversitesi

.....

TAAHHÜTNAME

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu; ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını, bilimsel etiğe uygun olarak kaynak gösterildiğini bildirir ve taahhüt ederim.

METEHAN BULUT

İmza

ÖZET

LİSANS BİTİRME TEZİ

UNİTY 2D OYUN PROGRAMLAMA

METEHAN BULUT

KASTAMONU ÜNİVERSİTESİ

MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DANIŞMAN:DOÇ.DR.SALİH GÖRGÜNOĞLU

ÖZET

Bu projede günümüzde en yaygın kullanılan eğlence sektörü olan video oyun sektörü için içerik üretmenin aşamaları, oyun programlama ve bu oyun programlamanın yapay zekâ ile desteklenerek, bilgisayarın kullanıcıyı insan zekâsına daha yakın bir şekilde karşılaması için oluşturulacak görsel ve işlevsel kodlamalar aşamaları anlatılmıştır. Bu yazılımın oluşturulması sürecinde kullanılacak olan yöntemler, teknikler ve uygulamalar konusunda bilgi verilmiş ve kısaca tanıtılmıştır. Unity Yazılımı, Android proje çıkarımı,C# Programlama, Nesneye Yönelik Programlama, Sınıf-Kalıtım ilişkisi, Boyut kavramı , AutoDesk3DSMax teknolojisi, Visual Studio 2019 Teknolojisi, A* Algoritması hakkında tanımlamalar ve derlemeler yapılmıştır.

ANAHTAR KELİMELER:A*Algoritması, Yapay zekâ, Unity Yazılımı, AutoDesk3DSMax, Visual Studi, Android Build, C#, NYP(Nesneye Yönelik Programlama),Class,2D,3D

TEŞEKKÜR

Bitirme projesi kapsamında bize fikir ve görüşleriyle, yol gösteren danışman hocamız Sayın Doç.Dr. SALİH GÖRGÜNOĞLU (Kastamonu Üniversitesi / Bilgisayar Mühendisliği)’na teşekkür ederim.

METEHAN BULUT

Kastamonu, 2021

İÇİNDEKİLER

TEZ ONAYI	ii
TAAHHÜTNAME	iii
ÖZET.....	iv
TEŞEKKÜR	v
İÇİNDEKİLER	v
ŞEKİL LİSTESİ.....	viii
TABLO LİSTESİ	Hata! Yer işareti tanımlanmamış.
SİMGELER VE KISALTMALAR LİSTESİ	x
1. GİRİŞ ve TANITIM.....	12
2. PROJEDE KULLANILAN MATERYALLER.....	14
2.1. Unity	14
2.1 Unityhub	15
2.2. Android Studio	16
2.3 AutoDesk 3DSMax	17
2.4 Visual Studio 2019	18
2.5 Asset Studio Nedir ?.....	19
2.6 Boyut Nedir, Boyut Kavramı Nedir?	20
2.6.1 Boyut Nedir?.....	20
2.6.2 Oyun Programlamada Boyut Kavramı Nedir Nasıl Kullanılır?.....	21
2.6.3 Boyutlarına Göre Oyunların İncelenmesi	22
2.6.5 TEPEDEDEN BAKIŞ AÇISI	24
3.PROJE GELİŞTİRME SÜRECİ.....	26
3.1 PROJENİN PLANI VE AMACI.....	26
3.2 UNITY OYUN MOTORU ÇALIŞMA MANTIĞI	26
3.3 PROJENİN OLUŞTURULMASI	27
3.4 PROJEDE OBJE OLUŞTURULMASI.....	28
3.5 OBJEYE SPRİTE EDİTOR VERİLMESİ.....	28
3.6 OBJEYE COLLIDER VERİLMESİ	28
3.7 OBJEYE RIGIDBODY VERİLMESİ	29
3.8 OBJEYE SCRIPT VERİLMESİ	30
3.9 SCRIPT OLUŞTURULMASI.....	31
3.10 KARAKTERİN YURUYUSU SCRİPT EKLENMESİ.....	31
3.10.1 KARAKTERİN YURUYUSU SCRİPT AÇIKLAMASI	32
3.11 KARAKTERE YON VERİLME SCRİPTİ	32
3.11.1 KARAKTERİN YONVERME SCRİPT AÇIKLAMASI	33
3.12 TİLEMAP OLUŞTURULMASI.....	33
3.13 KAMERA TAKİBİ SCRIPT EKLENMESİ.....	34
3.13.1 KAMERA TAKİBİ SCRİPT AÇIKLAMASI.....	35
3.14 OBJE KOPYALAYARAK NESNE ÇOĞALTILMASI	35
3.15 PROTECTED PROVIDE ÖZELLİKLERİ.....	36
3.16 İLERLEMİYİ KONSOLDA GÖRMEK.....	37
3.17 PREFAB KULLANIMI	38
3.18 TİLEMAP PALET KULLANIMI.....	39
3.19 UNITY LAYER KULLANIMI.....	40
3.20 HARİTA OLUŞTURULMASI	40
3.21 COMPOSITE COLLİDER İLE TİLEMAP COLLİDER BİRLEŞTİRİLMESİ.....	41
3.22 ÇARPRAZ HARAKETLENME.....	42

3.22.1 ÇAPRAZ HAREKETLENME AÇIKLAMASI	43
3.23 IŞIK EKLEME	45
3.23.1 IŞIK AYARLARI	46
3.23.2 IŞIKLARIN TANITIMI	48
3.24 OBJEYE BAŞKA BİR OBJENİN EKLENMESİ.....	49
3.24.1 ANA KARAKTERE FENER EKLENMESİ	50
3.25 TRİGGER VE COLLİSON KAVRAMLARI	50
3.26 KODLAR İLE İŞLEMLERİN YÖNETİLMESİ	52
3.26.1 FENER OBJESİNE FENER ALANI EKLENMESİ	52
3.26.2 FENER ALANI SCRIPT KODU EKLENMESİ	52
3.26.2.1 FENER ALANI SCRIPT KODU AÇIKLAMASI	53
3.26.3 BAKIŞ ALANI OBJESİNİN ANA KARAKTERE EKLENMESİ.....	53
3.26.4 FENER KONTROL SCRIPTİNİN ANA KARAKTERE EKLENMESİ	
54	
3.26.4.1 FENER KONTROL SCRIPT AÇIKLAMASI	55
3.26.5 DUVAR YANSIMA KONTROL SCRIPTİ EKLENMESİ.....	56
3.26.5.1 DUVAR YANSIMA KONTROL SCRIPT AÇIKLAMASI ...	57
3.26.5.2 SERIALIZEFIELD KULLANIMI.....	58
3.26.6 KARAKTERİN KOŞU VE YAVAŞ YÜRÜME FONKSİYONLARININ	
KAZANDIRILMASI.....	58
3.26.7 KARAKTERE SES ÇEMBERİ OBJESİNİN EKLENMESİ.....	59
3.26.7.1 SES ÇEMBERİ SCRIPTİNİN KARAKTERE EKLENMESİ	60
3.26.7.2 SES ÇEMBERİ SCRIPT AÇIKLAMASI	60
3.26.7.3 PROJEDE MANTIKSAL HATA ÇÖZÜMÜ.....	61
3.26.7.4 SES ÇEMBERİNİN FARKLI ODAYA SES İLETME SORUNU	
ÇÖZÜMÜ	61
3.26.8 RAYCAST NEDİR	62
3.26.8.1 RAYCAST OLUŞTURMA ve DUVAR KONTROL SCRIPTİNİ	
ANA KARAKTERE EKLEME.....	62
3.26.8.2 RAYCAST SCRIPT AÇIKLAMASI	63
3.27 GARDİYAN GENEL İŞLEYİŞ MODELİ	64
3.28 YAPAY ZEKA NEDİR?.....	64
3.28.1 UNITY OYUN MOTORU İLE YAPAY ZEKÂ	65
3.28.2 PROJE İÇERİSİNDE YAPAY ZEKÂ KULLANIM AMACI	65
3.28.3 A* ALGORTİMASI NEDİR.....	67
3.29 GARDİYAN OBJESİNE HAREKET KAZANDIRILMASI.....	68
3.29.1 RASTGELE HAREKET ETME KODLARI	68
3.29.2 TAKİP ALGORİTİMASI KODLARI VE A* PAKETİ KURULUMU	70
3.30 PROJEDEKİ OBJELERE ANİMASYON EKLEME.....	72
3.30.1 ANİMASYON PANELİ KULLANIMI	72
3.30.2 ANİMATÖR PANELİ KULLANIMI	73
3.30.3 ANİMASYON SCRIPTİ EKLENMESİ	74
3.31 PROJEYE ANA MENU EKLEME	74
3.31.1 ANA MENU KODLAMASI.....	75
4.SONUÇ VE ÖNERİLER.....	76
5.KAYNAKÇA	77
ÖZGEÇMİŞ.....	78

ŞEKİL LİSTESİ

Sayfa

Şekil 1 Unity Hub	15
Şekil 2 Unity	15
Şekil 3 Android Studio.....	16
Şekil 4 Autodesk	17
Şekil 5 Unity Kabul Edilen Dosya Tipleri	17
Şekil 6 Visual Studio.....	19
Şekil 7 Visual Studio Versiyon	19
Şekil 8 Unity Asset Store	20
Şekil 9 Boyut Kavramı.....	21
Şekil 10 Boyut Kavramı.....	21
Şekil 11 Side Scroller.....	22
Şekil 12 Side Scroller Angled	22
Şekil 13 Top Down Direct	22
Şekil 14 Top Down Direct	23
Şekil 15 2d&3d Hybrid Top.....	23
Şekil 16 Third Person.....	23
Şekil 17 First Person	24
Şekil 18 Tepe Bakış açısı	24
Şekil 19 25	25
Şekil 20 External Tools	27
Şekil 21 Proje Oluşturma Ekran.....	27
Şekil 22 Karakter objesi	28
Şekil 23 Collider Objesi	29
Şekil 24 Poligon Çarpıştırıcı	29
Şekil 25 Rigidbody Bileşeni.....	29
Şekil 26 İç görüntü	30
Şekil 27 Rotasyon Değerleri	30
Şekil 28 Script ekranı	30
Şekil 29 Kod.....	31
Şekil 30 Kod Objesi	31
Şekil 31 Kod.....	31
Şekil 32 Kod.....	32
Şekil 33 Kod.....	32
Şekil 34 Kod.....	32
Şekil 35 Kod.....	33
Şekil 36 Kod.....	33
Şekil 37 Hiyerarşi.....	34
Şekil 38 Kod.....	34
Şekil 39 Editör Paneli	35
Şekil 40 Harita.....	36
Şekil 41 Hiyerarşi.....	36
Şekil 42 Sınıf ilişkisi	36
Şekil 43 Kod.....	37
Şekil 44 Konsol Ekranı	38

Şekil 45 Dosya Boyutu	39
Şekil 46 Hiyerarşi.....	39
Şekil 47 39	
Şekil 48 39	
Şekil 49 Layer Paneli	40
Şekil 50 Harita Tasarım-2	41
Şekil 51 Harita Tasarım-1	41
Şekil 52 Composite Collider 2D	42
Şekil 53 Tilemap Collider 2D	42
Şekil 55 Kod.....	43
Şekil 54 Kod.....	43
Şekil 56 Kod.....	44
Şekil 57 LightWeight Rp	46
Şekil 58 Ayarlar	48
Şekil 59 Ayarlar	48
Şekil 60 Parent-Child İlişkisi	49
Şekil 61 Işık Özellikleri	50
Şekil 62 Oyun içi Görüntü	50
Şekil 63 Collison Trigger Farkı.....	51
Şekil 64 Oyun içi Görüntü	52
Şekil 65 Kod.....	52
Şekil 66 Kod.....	53
Şekil 67 Kod.....	54
Şekil 68 Kod.....	55
Şekil 69 Kod.....	55
Şekil 70 Oyun içi görüntü	56
Şekil 71 Oyun içi Görüntü	56
Şekil 72 Kod.....	57
Şekil 73 Kod.....	58
Şekil 74 Kod.....	59
Şekil 75 Circle Collider.....	60
Şekil 76 Kod.....	60
Şekil 77 Kod.....	61
Şekil 78 Anlatım Paneli	62
Şekil 79 Kodlama mantığı.....	62
Şekil 80 Oyun içi Görüntü	63
Şekil 81 Kod.....	63
Şekil 82 Kod.....	64
Şekil 83 Yapay zeka inceleme Modeli.....	65
Şekil 84 Yapay zeka Şeması	66
Şekil 85 A* Algoritması	67
Şekil 86 A* Algoritması Unity	68
Şekil 87 Kod.....	68
Şekil 88 Kod.....	69
Şekil 89 Kod.....	69
Şekil 90 A* paket içeriği.....	70
Şekil 91 A* Yüzeyi Oyun içi Uygulanmış Görüntü	71
Şekil 92 Kod.....	71
Şekil 93 A* Sonuç Görüntüsü.....	72

Şekil 94 Animasyon paneli	72
Şekil 95 Animator Paneli	73
Şekil 96 Kod.....	74
Şekil 97 Editör Paneli	75
Şekil 98 Kod.....	75

SİMGELER VE KISALTMALAR LİSTESİ

Kısaltmalar

NPC : Non Playable Character
API : Application Programming Interface
IDE : Tümüleşik Geliştirme Ortamı

1. GİRİŞ ve TANITIM

21.Yüzyıl içerisinde teknoloji hızla ilerliyor ve insanların amaç ve ihtiyaçları doğrultusunda gelişiyor ve ilerliyor. Bu değişim teknoloji açısından araçlar ve uygulamalar bazında çeşitli şekillerde gelişmesini sürdürüyor. Bu değişim ve gelişim şirketlerin geride kalması ve zirveye çıkması gibi farklı sonuçlar doğurabiliyor. Hızla gelişen teknolojiyi her anlamda yakalamak firmalar ve teknolojinin bir parçası olan yazılım şirketleri için oldukça önemlidir. Bu yarış içerisinde günün ve çağın teknolojisini yakalamak tüm firmaların amacıdır. Bu sebeplerden dolayı çağın teknolojisi diye tabir ettiğimiz kavramı yakalamak firmaların amacı olduğu gibi mühendislerin ve yazılım şirketlerinin de amacıdır. Diğer teknoloji firmaları ve şirketleri çağın teknolojisini, bir makinenin yeni sürümünü yaparak, parça temin ederek veyahut sahip olduğu teknolojiye yeni işlevsellikler kazandırarak yaparken yazılım şirketlerinde bu işlemin adı “Güncelleme” diye tabir ettiğimiz olaydır. Güncellendikçe yenilenen ve daha ileri seviyelere erişen uygulamalar sayesinde insanlara hizmet sunan web siteleri, uygulamalar vb. projeler arasında güncelliği yakalaması en önemli sektörlerden birisi de oyun sektörüdür. Video oyunları geliştirmek ve oynamak için çeşitli teknolojik aletler ve yazılımlar gün geçtikçe insanlığa kazandırılıyor. Popüler oyun firmalarının sıklıkla kullandığı sektörün ileri gelen yazılımlarına örnek olarak “Unity, Unreal Engine, CupperCube, GameMaker Studio” gibi yazılımlar örnek verilebilir. Unity; öncelikli olarak bilgisayarlar, konsollar ve mobil cihazlar için video oyunları ve simülasyonları geliştirmek için kullanılan ve Unity Technologies tarafından geliştirilen çapraz platform bir oyun motorudur. Tasarımcının 2 ve 3 boyut arasında rahatlıkla geçiş yapabildiği ve her ikisini çapraz olarak destekleyen, sürükleyip bırakma metodunu kullandıran ve C# Scripts(Senaryo-Kodları) ile komut dosyası yazmayı destekleyen çok yönlü bir platformdur. Başlıca kendi kütüphanesini yazılım dünyasına kazandırmış ve ‘Unity’ isimli kütüphanesi ile yazılımcının yapmak istediği çoğu işlevi daha rahat bir şekilde işleme alabilen bir platform olarak sektördeki konfor seviyesini korumuştur. Unity C# ve Boo isimli İki programlama dili desteklemektedir. Boo dili Unity 5’in yayınlanmasından sonra kullanımı tavsiye edilmeyen Unity 2017.1’in piyasaya sürülmesinden sonra Ağustos 2017’de kullanımı pek tavsiye edilmeyen tescilli bir

betik dilidir. Unity oyun motoru ařaęıdaki grafik ortamlarını ve API'lerini hedeflemektedir: Windows için DX10,DX11,DX12 ve bazı işlemciler, MacOS için; Intel ve AMD işlemcileri, Linux için; OpenGL3.2, Vulkan gibi Api'leri destekler.

2. PROJEDE KULLANILAN MATERYALLER

Bu kısımda projede kullanılan materyallerin tanıtımı ve kullanım amaçları ile ilgili kısaca bilgiler verilmiştir. Kullanılan materyallerin işleyişleri projenin gelişim süreci kısmında daha detaylı olarak yazılmıştır.

2.1.Unity

Unity 3D, oyun geliştirmek üzere üretilmiş bir oyun motorudur. Unity piyasaya sürüldüğündeki çıkış hamlesi ve onu yukarılara taşıyan sebep, yazılımın oyunun bilgisayara kurulmadan oynanabilir kılması ve editör destekli oyun paneli kullanması olmuştur. Ayrıca Unity 3D motorunu ile üretilen oyunlar istenildiği takdirde Unity Web Player eklentisi sayesinde herhangi bir programa veya eklentiye gerek kalmadan tarayıcı üzerinden direk oynanabilir projelerdir. Kullanılan tarayıcı ve donanımın sahip olduğu RAM belleğin yeterliliğine göre istenildiği çaptaki bir oyun tarayıcı üzerinden rahatlıkla oynanabilir. Bu sayede çoğu proje için kurulum gerektirmeden server üzerinden oyun oynanabilir kıldığından, depolama alanında fazla yer işgalinden kaçınarak oyuncuların dostu bir platforma dönüşmüştür. Unity bu geliştirme sayesinde kendi platformu üzerinde çalıştırabildiği projeler ve oyunların sertifika ve korsan sıkıntısının bir şekilde önüne geçmiştir. Unity oyun motorunun oyun yapımcılarına sağladığı en büyük kolaylıklardan birisi de oluşturulan projenin birkaç ufak ayar ile istenilen ortama (Bilgisayarlar, Mac Bilgisayarlar, Web-Sunucuları, iOS, Android, Windows işletim sistemine sahip telefonlar(Lumia gibi), Playstation, Xbox vb.) derlenip sıkıntısız çalışabilmesidir. Bu sayede oyun yapımcıları derlediği projeyi farklı ortamlarda derleyip sunmaktan çekinmez ve rahat bir geliştirme ortamı sağlanmış olur. Bu sistem sayesinde aslında android için veya Windows için hazırlanan bir proje bir tık ile Playstation ortamı için de çalışır hale getirilebilir.

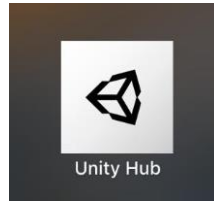
Unity diğer oyun geliştirme motorlarının yanında ücretsiz olması ve yine çoğu ücretsiz oyun yapım işlemleri olan culling, occlusion gibi özellikleri oyun yapımcılarına sunuyor. Unity teknolojisinin bir diğer oyun yapımcısına yararı da proje çalıştırıldığı

süreç içerisinde kod yazma imkânı vermesidir. Oyun aktif haldeyken kodu aktif olarak sürekli derlediği için kod değiştiğinde içeriğin değişimini yapımcının izlemesine olanak sağlar.

Bu çalışma mantığı oyun yapımcısına hem esnekliği artırma hem de zamandan tasarruf şansını yaratır. Bu çalışma mantığı geliştiriciye esneklik sağlamakta, geliştirme süresini kısaltmaktadır. Aynı zamanda Murat Delen'in şu anlatımı ile Unity kolaylığına bir örnek daha çıkarılabilir.” Unity bütün bu avantajlarının yanında Unity 3D'de yazılmış oyunlar düşük ve orta seviye bilgisayarlarda (en düşük 1.6 Ghz işlemci, 500 MB ram) rahatlıkla oynanabilmektedir. Unity3D oyun motoru Unity Engines tarafından C/C++ ile geliştirilmiştir. Unity 4.0 sürümü ile C#, JavaScript, Boo ve DirectX dilleri ile oyun geliştirmenize destek vermektedir.[4]“ Unity3D ile birçok oyun tasarlanmıştır ve bu oyunlara örnek olarak ‘Galactica’, ‘Legends of Aethereus’ gibi birçok meşhur oyun Unity3D ile tasarlanmıştır. Run 2, Dead Trigger 2 gibi Mobil oyunlarda Unity3D ile tasarlanmıştır.

2.1 Unityhub

Unity Hub, Unity Projelerinizi ve kurulumlarınızı bulma, indirme ve yönetme şeklinizi kolaylaştıran bağımsız bir uygulamadır. Ayrıca, makinenize önceden yüklemiş olduğunuz Düzenleyici sürümlerini manuel olarak Hub'ınıza ekleyebilirsiniz.[3]



Şekil 1 Unity Hub



Şekil 2 Unity

2.2. Android Studio

Android Studio , Android işletim sistemine sahip cihazlar için uygulamalar ve yazılımlar geliştirmek üzere üretilmiş bir IDE'dir. 16 Mayıs 2013 tarihinde bir Google etkinliğinde tanıtılarak piyasaya sürülmüştür. Ağırlıklı olarak mobil cihazlar için uygulama geliştirmekte kullanılsa da tabletler, televizyonlar ve Android işletim sistemine sahip tüm cihazlar için Android Studio tümleşik geliştirme ortamı kullanılabilir. Java , Kotlin, C++ dillerinin geliştirilmesine olanak sağlar. Çapraz platformlarda çalışabilme özelliğinden dolayı bilgisayara ihtiyaç duymadan uygun işletim sistemi ile programcılara dilediği donanımda kod yazma imkanı sunmaktadır. Apache lisansı ile lisanslanmış ve ücretsiz olarak piyasaya sürülmüş ve sürüldüğü günden bugüne dek ücretsiz kalmıştır. Projede Android Studio, otomatik olarak android ortamına derlenecek olan projenin gerekli arayüz ekipmanlarını tasarlamak amacı ile projeye dahil edilmiştir.



Şekil 3 Android Studio

2.3 AutoDesk 3DSMax

3DSmax olarak bilinen 3D Studio Max programı, Autodesk tarafından tasarlanan, geliştirilen ve kullanıma sunulan bir bilgisayar grafik geliştirme yazılımıdır. 3ds Max'in ilk sürümü, Nisan 1990 yayınlanan 3ds Max 1990 versiyonudur. Yazının yazıldığı tarih itibari ile yayımlanan son güncel sürümü 2017 SP3 sürümüdür. 3DSMax Programı MSDOS ortamında çalışan 3D Studio yazılımının devamıdır. 3DStudio Max 3D modeller ve 2D modeller oluşturmak için daha doğrusu çizim yapmak için oluşturulan bir modelleme programıdır. Gelişmiş eklenti desteği ve kullanım kolaylığı ile 3ds Max, 3D ve 2D modelleme yazılımlarında en çok kullanılan uygulama haline gelmiştir. Gelişmiş karakter modelleme yetenekleri ile oyun geliştiricilerin gözdesi haline gelmiştir. Ayrıca film özel efektlerinde, mimari gösterilerde ve endüstriyel tasarım gösterilerinde yaygın olarak kullanılmaktadır



Şekil 4 Autodesk

Kaynakça :Alan Thorn (auth.) - Learn Unity for 2D Game Development-Apress (2013)

Table 1-1. File Formats Accepted by Unity

Meshes	Textures	Audio	Movies
.FBX	.PSD ●	.MP3	.MOV
.MA	.TIFF	.OGG	.AVI
.MB	.PNG ●	.MOD	.OGG
.MAX ●	.BMP	.IT	.ASF
.BLEND	.JPG ●	.XM	.MPG
.3DS ●	.TGA	.S3M	
.DXF	.DDS/PVR	.WAV	
.C4D			

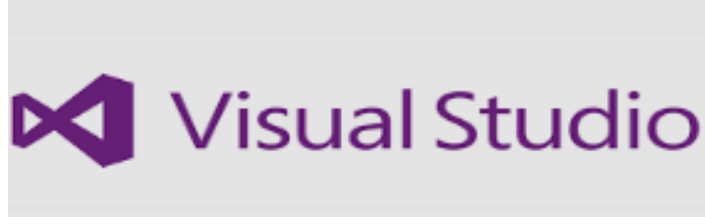
Şekil 5 Unity Kabul Edilen Dosya Tipleri

Burada projenin ana yazılım geliştirme ortamı olan Unity Yazılımı içerisinde kabul edilebilen dosya tipleri listelenmiştir ve bu liste içerisindeki tip dosyaları oluşturulabilecek yazılımlardan faydalanarak oyun geliştiricisi kendi kaynak dosyalarını üretebilir.

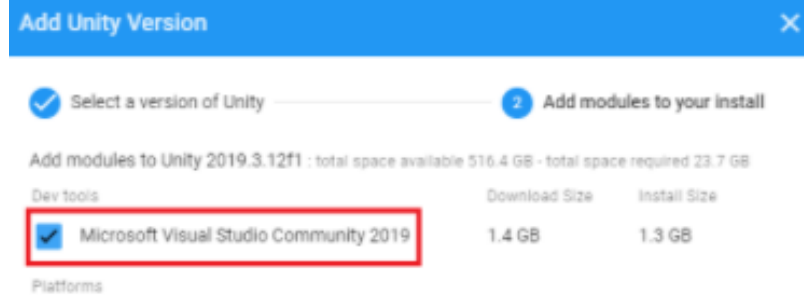
2.4 Visual Studio 2019

Visual Studio geliştiriciler tarafından kullanılan bir tümleşik geliştirme ortamıdır. Programlar, uygulamalar veya web siteleri oluşturmak için birden fazla programlama dili kullanabileceğiniz entegre bir geliştirme ortamı olması yanı sıra, Microsoft Windows için bilgisayar programları, web siteleri, web uygulamaları, web hizmetleri ve mobil uygulamalar geliştirmek için en çok kullanılan tümleşik geliştirme ortamlarından biridir. Visual Studio, Windows API, Windows Forms, Windows Presentation Foundation, Windows Store ve Microsoft Silverlight gibi Microsoft yazılım geliştirme altyapı platformlarını kullanır. Yazılımın temelinde proje oluşturulurken yerel(kaynak) kod ve yönetilen kod üretebilir. Yönetilen kod programcının dinamik olarak değiştirerek projeyi tasarladığı kodlar olurken kaynak kodlar Visual Studio taban kütüphane kodlarıdır. Ayrıca Microsoft Visual Studio'nun ücretsiz bir "topluluk" ve ücretli bir "ticari" sürümü vardır. Ücretsiz olması da kullanıcıların bu ortama kolayca ulaşabilmesi için önemli bir etkidir. Programın geçmişi 1995 yılına kadar uzanır. Yazılım endüstrisindeki bazı kişilerin veya tüm insanların hayatında bir yeri vardır, şu anki mevcut sürümü 2019 yılında üreticisi Windows tarafından kullanıcılara sunulmuştur.

Visual Studio, farklı programlama dillerini destekler ve dile özgü bir hizmet olması koşuluyla, kod düzenleyicisinin ve hata ayıklayıcının hemen hemen tüm programlama dillerini desteklemesine olanak tanır. Yerleşik diller arasında C, C ++ ve C ++ / CLI (Visual C ++ ile), VB.NET (Visual Basic .NET ile), C # (Visual C # ile), F # (Visual Studio 2010'dan itibaren) ve TypeScript (Visual Studio 2013 Update 2'den itibaren) bulunur.



Şekil 6 Visual Studio



Şekil 7 Visual Studio Versiyon

2.5 Asset Studio Nedir ?

Asset Store Unity yazılımının içerisinde kolaylıkla ulaşılabilen Unity geliştiricileri tarafından özgür geliştiricilere kaynak sunumu yapılması amacıyla kurulmuş bir internet sitesidir. İnternet sitesine erişim program içerisinde (dahili) veya program dışarısından (harici) olarak gerçekleştirilebilir. Projeler için paketler, görseller, ve altyapı varlıklarının ücretli veya ücretsiz olarak satın alındığı bir çevrimiçi market, e ticaret sitesidir.

Asset Store aynı zamanda özgür yazılımcılar tarafından geliştirilen varlıkların da eklenmesine imkan sağladığından bir kazanç kapısıdır. Bazı firmalar, StudioArtists gibi bu sitelerdeki satışlarıyla şirket gelirlerini artırmaktadır.

Örnek olarak özgür yazılımcı oluşturduğu projede bir efekt kullanmak istediği zaman (örnek olarak ateş efekti) bu efekti kendisinin geliştirmesi yerine mağazadan elde edebilir ve projesine ekleyip kullanabilir. Buradan satın alınan veya ücretsiz olarak yayımlanan varlıkların lisansları mağazaya bağlıdır ve kullanılmasında bir sorun teşkil edilmemektedir.

Ayrıca özgür yazılımcıların yaptığı varlıkların, yeterli satışlara ulaşıldığında geliştiricisine ciddi kazançlar sağladığı bilinmektedir.

Asset Store her geçen gün artan içeriği ile oyun geliştiricilerine büyük katkı sağlamaktadır.



Şekil 8 Unity Asset Store

2.6 Boyut Nedir, Boyut Kavramı Nedir?

2.6.1 Boyut Nedir?

Boyut basit olarak bir uzayın kaç adet düzleme sahip olduğunun cevabıdır. Nokta sıfır boyutlu, çizgi tek boyutlu, düzlem iki boyutlu ve uzay üç boyutludur.

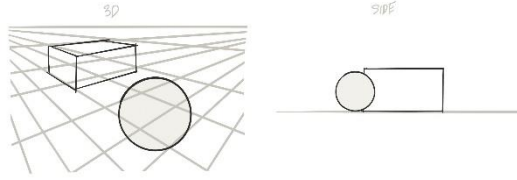
Bir nesnenin boyutu, nesne üzerindeki bir noktayı temsil etmek için gereken minimum koordinat sayısı olarak tanımlanabilir. Dolayısıyla bir düzlem eğer iki boyutlu ise içerisindeki her noktanın koordinatı (X,Y) olarak tanımlanabilir. Tek boyutlu bir uzay düşünmek imkansızdır çünkü minimum uzay iki boyutludur. Bir nesnenin sıfır boyutlu olduğundan bahsediyorsak o nesne bir noktadır. Çünkü nokta bir uzay üzerinde belirlenemez. Çizgi tek boyutludur çünkü bir alanı hesaplanamayan geometrik bir şekildir dolayısıyla sadece X için bahsederiz ve bu değer tek boyuta tekabül eder. Eğer bir kare daire veya benzer alanlı bir objeden bahsediyorsak iki boyutlu bir objeden bahsedildiği anlaşılır. (X,Y,Z) gibi bir koordinattan bahsediyorsak, uzay içerisine bir derinlik katmanı eklenir ve bu sefer 3 boyutlu bir uzay içerisinde bir nesneden bahsedildiği anlaşılır. Basit olarak Koordinat sistemindeki her sütun bir boyut eklenmesi anlamına gelir ve matematiksel düşünüldüğünde boyut kavramı 3 boyut ile

sınırlı değildir. İnsan gözünün algılayabildiği 3 boyut varken analitik düzlemde bunun bir sınırı yoktur.

2.6.2 Oyun Programlamada Boyut Kavramı Nedir Nasıl Kullanılır?

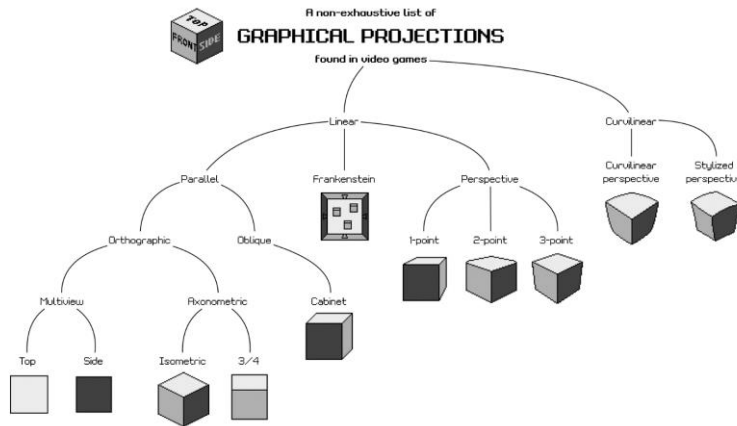
Öncelikle oyun programlamada bahsedilen boyut kavramı bir bakış alanı veya bir gözlemden ziyade, bu terim aslında Perspektif 'ir. Perspektifler 3 Boyut için de 2 boyut programlama için de var olan bir kavramdır ve oyun geliştiricisinin oyun tasarım ve plan aşamasında dikkat etmesini ve belirli seçimler ile projeye başlamasını sağlayan kılavuz niteliğinde proje temelleridir.

Kaynakça :- GameDesignInitiative Drg. Cornell University Lecture 15 Perspective in 2D Games



Şekil 9 Boyut Kavramı

Şekilde görülebildiği üzere basit olarak iki nesnenin 3 boyutlu bir uzayda nasıl durduğu ve 2 boyutlu bir perspektife nasıl iz düşüm sergilediği gözükmemektedir.



Şekil 10 Boyut Kavramı

2.6.3 Boyutlarına Göre Oyunların İncelenmesi



Şekil 11 Side Scroller

2D | Side Scroller (Direct)

Bu perspektif tipi genellikle platform zıplama oyunlarında veya yatay atış oyunlarında kullanılır x ve y axisleri aktiftir.



Şekil 12 Side Scroller Angled

2D | Side Scroller (Angled)

2D ‘ ye açı ile bakışlı bu tarzda ana obje yukarı ve aşağı hareket edebilir ve land (zemin) sabit x doğrultusunda değil bir düzlem parçasıdır. Macera oyunları için geliştirilmiştir Multi karakterlere olanak

tanır.



Şekil 13 Top Down Direct

2D | Top-Down (Direct)

Bulmaca oyunları için tasarlanmış bir bakış açısıdır. Ana objemiz zemin üzerinde tüm yönlerde hareket edebilmektedir. Fakat bir öncekinden farkı artık tüm düzlem bir zemindir. Proje X ve Y axisleri üzerinde oluşturulur



Şekil 14 Top Down Direct

2D | Top-Down (Slight Angle)

Bu perspektif de 2D Top-Down ile aynıdır, ancak hareketli grafikler ve nesneler birbiriyle örtüşme eğilimindedir. Bu stil daha iyi grafiklere izin verir, ancak derinlikteki küçük bir kafa karışıklığı nedeniyle oyuncu kontrolünün bir kısmını kaybeder. X ve Y axisleri

yine düzlemdir fakat çizim ile eğik bakış açısına izin vermeyi amaçlar .



Şekil 15 2d&3d Hybrid Top

2D & 3D Hybrid | Top-Down (Isomorphic)

Bu oyun tarzında, çizimler iki boyutlu iken zemin üç boyutludur. Ayrıca perspektif, hem sol-sağ eksende hem de üst-alt eksende 45 derece kamera açısına göre ayarlanır.



Şekil 16 Third Person

3D | Third Person

Bu oyun türü, oyuncunun karakterinin dışından bazı açıları gösterir. Genellikle bu açı omuz üzerindedir, ancak kontrol stiline bağlı olarak herhangi başka bir açı veya tüm açıları olabilir. Genellikle bu, oyunun

oyuncunun istediğini düşündüğü açıyı yorumlamasını içerir ve bu da oyuncu kontrolünü önemli ölçüde azaltır.

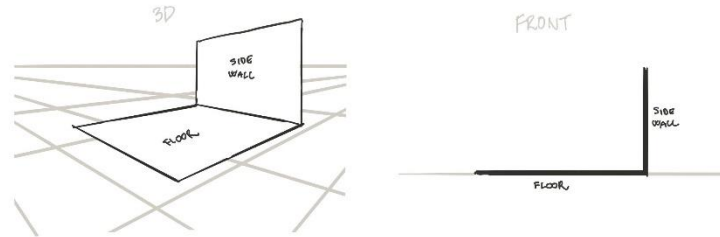


Şekil 17 First Person

3D | First Person

Bu perspektif, oyuncunun her zaman karakterin gözlerinden baktığı anlamına gelir. Oyuncunun çok daha iyi kontrole sahip olmasını sağlar ancak kamera açısı nedeniyle oyuncunun görme yeteneğini sınırlar.

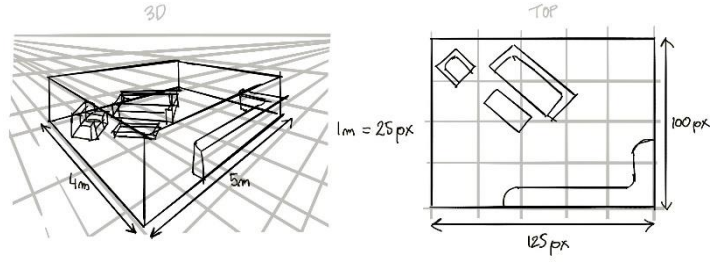
2.6.5 TEPEDEN BAKIŞ AÇISI



Şekil 18 Tepe Bakış açısı

Top-Down Perspective(Tepeden Bakış Perspektifi) ile çalışılan projede düzlem yatay bir zemin üzerinde oluşur. Objeler bu düzlem üzerinde belirli bir koordinatlar ile verilir. Tepeden bakış açısı ile oluşturulan projelerde zemine kamera açısı tavandan olur ve kameranın her zaman 3 boyutta bir varlığı olduğundan bahsedilir.

Diğer objelerin 3 boyutta bir varlığı olması için bir **Z** koordinatına sahip olması gerekir.



Şekil 19

Şekilde görüldüğü üzere 3 boyutlu bir uzaydaki oluşturulma planı gösterilmiştir.

3.PROJE GELİŞTİRME SÜRECİ

3.1 PROJENİN PLANI VE AMACI

Projenin amacı Türkiye yazılım sektörüne Unity programlama ile ilgili bir proje kazandırmak ve oyun geliştiricileri için oyun yazılım sektörüne ön gösterebilecek bir oyun oluşturmak ve bu projenin adımlarını izlenebilir ve uygulanabilir şekilde açıklanarak kılavuz niteliğinde Türkçe Oyun geliştirme Yazılımları Literatürüne kazandırmaktır.

Projenin planı; basitçe bir kaçış oyunu üzerinde labirent yapısını andıracak şekilde harita üretimi ve bu haritanın içinde yapay zeka destekli NPC karakterlerin, oyuncunun yönettiği karakteri yakalaması ve ana karakteri yöneten oyuncunun bilgisayar yönetimli oyunculara yakalanmadan haritadan çıkabilmesi üzerine kurulmuştur.

Oyun programlamada oluşturulan nesneler, birbirini takip eden olaylar ve çoğu zaman tekrar işlemleri içerdiğinden konu başlıklarında işlemlerin nasıl yapıldığından bahsedilmiş fakat benzer işlemin farklı obje için tekrarı açıklanmamıştır.

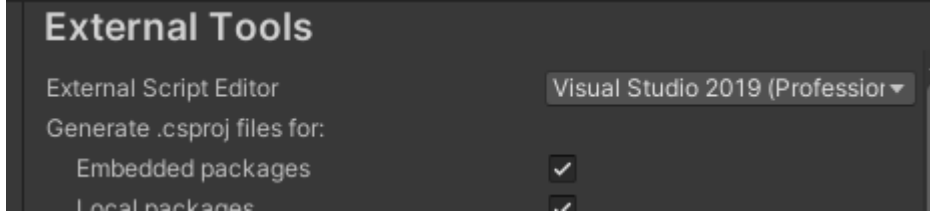
3.2 UNITY OYUN MOTORU ÇALIŞMA MANTIĞI

Unity Objesi – Kod ilişkisi:

Unity yazılımı içerisinde sahne içerisinde oluşturulan her dosya bir Unity Oyun objesi olarak adlandırılır. Bu Oyun objelerinin özellikleri olur ve sahip olduğu bileşenlere göre obje şekillenir.

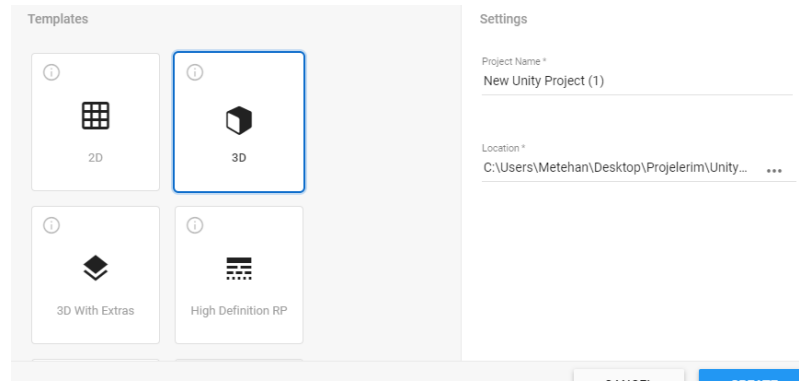
Bileşenlerin bir araya gelerek oluşturduğu objelerin oyun çalıştırılmadan yapımcı panelinden özellikleri ayarlanabiliyor olsa bile oyun çalıştırıldıktan sonra oluşacak davranışları kodlar ile oluşturulur. Bunun sebebi oyun projesinin bitiminde oyuncunun editör paneline erişmiyor olması ve yapımcının oynanışı daha konforlu oyunlar üretme amacı gütmesidir.

Unity içerisinde Kod dosyalarına çoğu literatürde olduğu gibi Script adıyla erişim sağlanır. Birbirinden iki farklı program bu bağlantıyı Script'ler Unity yazılım ayarlarındaki External(Dış) Araçlar ile gerçekleştiriyor.



Şekil 20 External Tools

3.3 PROJENİN OLUŞTURULMASI



Şekil 21 Proje Oluşturma Ekran

Unity için bir oyun projesi oluşturulurken ilk seçim için boyut seçimi yapılır. Buradaki boyut seçimi kalıcı bir boyut seçimi değildir, proje başlangıcında 2 boyut seçildiğinde proje içerisinde tekrar 3 boyuta geçiş sağlanabilir. Bunun sebebi Unity yazılımının dinamik geliştirme imkânı sunmasıdır. Buradaki seçimin önemi sahne tasarımı ve uygulama yapımçı için gerekli materyalleri otomatik olarak seçmesinden kaynaklanır.

3.4 PROJEDE OBJE OLUŞTURULMASI

Unity’de bir obje oluşturulması için hiyerarşi sekmesi içerisinde sağ tıklanarak “Create Empty” komutu ile veya CTRL+SHIFT+N kısayolu ile oluşturulabilir. Yazılımcının özgür olduğu bir alan olduğundan, oyun programlamada oluşturulan dosyaların sırası önemli değildir.

3.5 OBJEYE SPRİTE EDİTOR VERİLMESİ

Projede sahneye obje eklendikten sonra o objeyi editörün görmesi için onun grafik işlemci tarafından **render edilmesi (çizilmesi)** gerekir. Bu bileşen PNG formatında bir görüntü olarak Unity Asset Store içerisindeki **CodeArtist.Mx Studio** mağazasındaki ücretsiz varlıkları içerisinde elde edilmiştir.

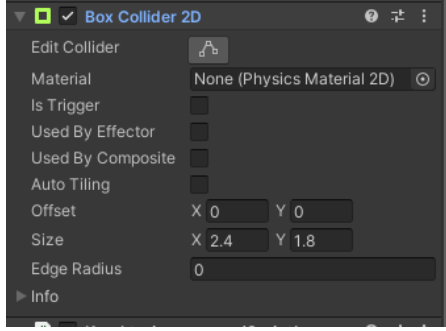


Şekil 22 Karakter objesi

3.6 OBJEYE COLLIDER VERİLMESİ

Unity içerisinde projede oluşturulan bir obje soyut varsayılan olarak gelir. Sahne içerisindeki diğer objeler ile iletişimi yoktur. Yapımcının bu iletişimi sağlaması için **Çarpıştırıcı(Collider)** bileşenini objeye eklemesi gerekir.

Bileşen ekle kısmından herhangi bir çarpıştırıcı tipi seçilir ve Unity otomatik olarak geliştiricinin seçtiği çarpıştırıcıyı sprite ‘a uygun şekilde şekillendirerek nesneye ekler. Daha sonrasında düzenlemeye, geliştirilmeye ve değiştirilmeye açıktır.



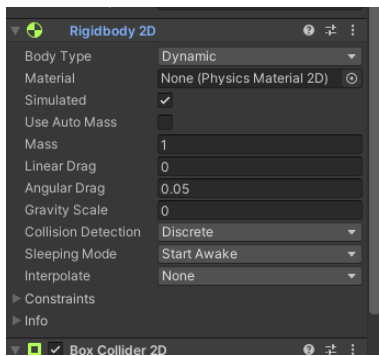
Şekil 23 Collider Objesi



Şekil 24 Poligon Çarpıştırıcı

3.7 OBJEYE RIGIDBODY VERİLMESİ

Unity içerisinde projede oluşturulan bir obje soyut varsayılan olarak gelir. Dolayısıyla bir fizik kuralına sahip değildir. Herhangi bir kuvvetten etkilenmez ve yerçekimi varsayılanından etkilenmez. Objenin diğer kuvvetler içerisinde etkilenebilir olması, objeye bir kuvvet eklenmesi veya etkilenmesi istendiğinde objenin alması gereken bileşen **RIGIDBODY** dir.

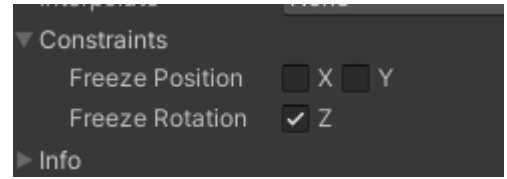


Şekil 25 Rigidbody Bileşeni

Objeye Rigidbody bileşeni eklerken geliştiricinin dikkat etmesi gereken husus o objenin diğer nesneler ile çarpıştığında vereceği tepkiyi düzenlemek olacaktır. Bu bileşenin özelliklerinde Z ekseninde dönüş kapatılmıştır ve bu sayede karakterin duvarlara çarptığında aldığı etki tepki vektörü ile anlamsız dönmelerinin önüne geçilmiştir.



Şekil 26 İç görüntü

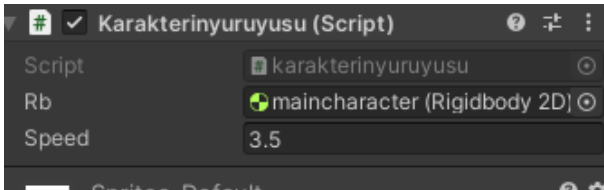


Şekil 27 Rotasyon Değerleri

3.8 OBJEYE SCRIPT VERİLMESİ

Unity içerisinde projede oluşturulan bir obje etkisiz varsayılan olarak gelir. Sahne içerisindeki diğer objeler ile iletişimi olsa kuvvet uzayında varlığı olsa bile objenin herhangi bir şekilde proje çalışırken özellik değiştirmesi, form değiştirmesi olaylarının gerçekleşmesi için oyun geliştiricisi objeye script atamak zorundadır.

Projede oluşturulan ana karakter nesnesine eklenecek script ilk olarak karakterin hareket edilmesi ve oyuncu tarafından kontrol edilebilmesidir.



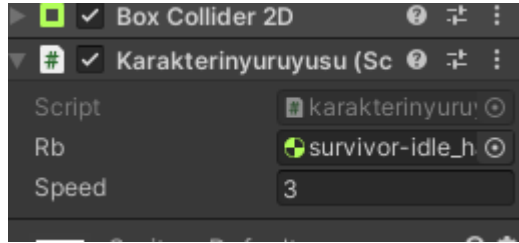
Şekil 28 Script ekranı

3.9 SCRIPT OLUŞTURULMASI

Scriptler sahneden bağımsız dosyalardır. Dolayısıyla oluşturulan bu dosyanın sahneyi tanıması için bağlantı oluşturulması gerekir. Bu bağlantı **UnityEngine** kütüphanesinin dosyaya **import** edilmesi ile sağlanır. Bu aşamadan sonra kod içerisinde kontrol edilecek bileşenin script dosyasına tanıtılması gerekir. Oyun geliştiricisi bu bileşeni dosyaya dilerse editör panelinden seçerek (şekil 2), dilerse sahne oluşturulduğu anda kod üzerinden tanıtabilir.(Şekil-3). Ayrıca Scriptler çalışmak için sahnede aktif bir bileşen olarak bulunmak zorundadır çünkü Unity yazılımı projenin dosyalarının tamamını değil sahnedeki dosyaların tamamını çalıştırarak sahneyi oluşturur.

```
public Rigidbody2D rb; // Fizik yasalarını uygulamak için 2d Fizik motoru
Vector2 move; // hareket etmeyi sağlayan vektör
```

Şekil 29 Kod



Şekil 30 Kod Objesi

```
{
    print("objeninadi: "+GameObject.FindGameObjectWithTag("anakaraktertag"));
    // print(speed);
    // print(vucud.name+" = "+vucud.position.ToString()+"Ana Karakter = "+anakarakter.position.ToString())
}
```

Şekil 31 Kod

3.10 KARAKTERİN YURUYUSU SCRIPT EKLENMESİ

Projede karakterin hareket etmesi oyuncunun kontrolünde olacaktır. Bunun için oyuncunun bastığı tuşlar, dokunduğu butonlar gibi gerekli giriş elemanlarının bileşen ile bağlantısı oluşturulması gerekir. Bu aşamada projede şekil 2 deki **Input** fonksiyonu bu görevi üstlenir. Oluşturulduktan sonra devamlı çalışması isteniyorsa Unity

fonksiyonlarından Sürekli çağrılmak istenen **FixedUpdate** veya **Update** içerisinde gerekli kodların yazılması gerekir. Sadece sahne oluşturulduğunda çalışması isteniyorsa **Start** fonksiyonu içerisinde yazılır.

```
void FixedUpdate()
{
    hareket();
}
```

Şekil 33 Kod

```
void hareket()
{
    move.x = Input.GetAxisRaw("Horizontal");
    move.y = Input.GetAxisRaw("Vertical");
    rb.MovePosition(rb.position + move * speed * Time.deltaTime);
}
```

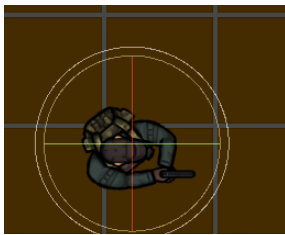
Şekil 32 Kod

3.10.1 KARAKTERİN YURUYUSU SCRIPT AÇIKLAMASI

Microsoft Visual Studio’da değersiz bıraktığımız public değişkenler Unity tarafından otomatik 0 olarak belirleniyor, burada da speed değişkenimiz 0 olarak sabit gelirken onu 3 ile değiştiriyoruz ve karakterimiz hareket etmeye başlıyor. Basacağımız klavye tuşunun algılanması Unity tarafından Horizontal Keys : “W, S” ve ↑ ve ↓ olarak ; Vertical Keys: “A,D” ve “→ ,←“ olarak belirlendiğinden tekrardan bir KeyPress event’i eklemiyoruz projeye.

3.11 KARAKTERE YON VERİLME SCRIPTİ

Unity yazılımında objelerin bir Rotation değeri vardır. Tepeden bakış oyunlarında oluşturulan objelerin Z eksenindeki dönüş değerleri o objenin görüntüsünün istenildiği tarafa çevrilmesinde ana rolü oynar. Tepeden bakış olmayan oyunlarda bu işlemler animasyonlar ile yönetilebiliyorken tepeden bakış oyunlarında bu rotasyon değeri kullanılır.



Şekil 34 Kod

```

32 void yonverme()
33 {
34
35     if ((move.x) < 0 && nereyebakıyor!="sol") // SOLA DÖNÜŞ
36     {
37         if(nereyebakıyor!="asagi"&&nereyebakıyor!="yukari")
38             transform.Rotate(0,0,180); //sağ bakıyorken sola dönüş
39         else if( nereyebakıyor=="asagi")
40             transform.Rotate(0, 0, -90);
41         else if (nereyebakıyor == "yukari")
42             transform.Rotate(0, 0, 90);
43         nereyebakıyor = "sol";
44     }
45     else if ((move.x) > 0 && nereyebakıyor!="sag") // SAĞA DÖNÜŞ
46     {
47         if (nereyebakıyor != "asagi" && nereyebakıyor != "yukari")
48             transform.Rotate(0, 0, 180);
49         else if (nereyebakıyor == "asagi")
50             transform.Rotate(0, 0, 90);
51         else if (nereyebakıyor == "yukari")
52             transform.Rotate(0, 0, -90);
53         nereyebakıyor = "sag";
54     }
55 }
56

```

Şekil 35 Kod

```

57
58     if ((move.y) < 0 && nereyebakıyor!="asagi" )
59     {
60         if(nereyebakıyor=="sag")
61             transform.Rotate(0, 0, -90);
62         else if (nereyebakıyor == "sol")
63             transform.Rotate(0, 0, 90);
64         else if (nereyebakıyor=="yukari")
65             transform.Rotate(0, 0, 180);
66         nereyebakıyor = "asagi";
67     }
68     else if ((move.y) > 0 && nereyebakıyor!="yukari")
69     {
70         if (nereyebakıyor == "sag")
71             transform.Rotate(0, 0, 90);
72         else if (nereyebakıyor == "sol")
73             transform.Rotate(0, 0, -90);
74         else if (nereyebakıyor == "asagi")
75             transform.Rotate(0, 0, 180);
76         nereyebakıyor = "yukari";
77     }
78

```

Şekil 36 Kod

3.11.1 KARAKTERİN YONVERME SCRIPT AÇIKLAMASI

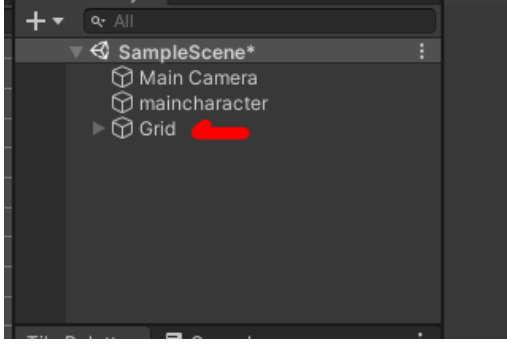
Unity ‘de dinamik proje kodlama sistemi olduğundan oluşturulan kodlar devamlı olarak işleyişe göre şekillenir ve değişiklik gösterir. Temel olarak bir fonksiyon içerisinde objenin yapması istenilen olaylar eklenir ve bu fonksiyon gerekli Unity ana fonksiyonunda çağrılarak kodlama tamamlanır. Buradaki kodlamada ise bu yapı baz alınmıştır.

Yön verme için bu projede kullanılan hazır fonksiyon Transform.Rotate() oluyor. Aldığı 3 parametre’den (X, Y, Z) Z ekseninde dönüş gerçekleştiriyor sebebi ise nesnelerin X ekseninde dönüş yapması 2D projeler için söz konusu değildir.

3.12 TİLEMAP OLUŞTURULMASI

Unity projelerinde bir harita veya obje uzayı oluşturma konusu çok çeşitlidir. Bakış açısına göre karşıdan bakış açısına sahip oyunlarda arka plana bir fotoğraf konularak, sahne içerisinde hareket edilebilecek uzayın tilemap ile oluşturulması söz konusu iken, tepeden bakış açısına sahip projelerde Tilemap direk uzayın kendisi olup tüm nesne uzayını içerisinde barındırabilir. Kare tabanlı bir düzleme tek tek görüntü

render'layarak oluşturulan ve bu kareler sayesinde objelere somutluk kazandırılan tilemap objesi bu projede aktif uzay olarak seçilmiştir.



Şekil 37 Hiyerarşi

Tilemap objesi oluşturulduğunda bir düzlem katman objesi (**Grid**) ile birlikte gelir. Bu objede nesnenin somut kısımları ile işlemler sürdürülürken görsel-soyut işlemler alt nesnelerde gerçekleşir.

Tilemap oluşumunda kaynak nesneleri almak için yararlanılan kaynak www.Icith.io sitesinde CMX Studio nun ücretsiz olarak yayınladığı Items2D paketidir.

3.13 KAMERA TAKİBİ SCRIPT EKLENMESİ

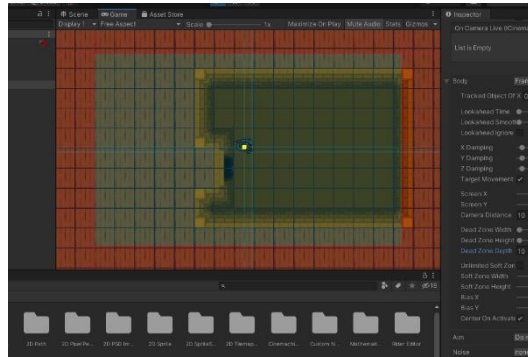
Unity yazılımında sahne oluşturulduğunda varsayılan bir ana kamera nesnesi ile gelir. Ekstra bir kamera kullanıldığında oyun sahnesi hangi ekranı seçeceği belirtilmemişse görüş iptal edilir ve oyuncuya siyah bir ekran içerisinde kamera hatası mesajı gösterilir. Kameranın hareket etmesi ise yine bir obje olarak gelen kamera nesnesinin oyun çalışırken hareket etmesine dayanır. Bu işlem temel olarak yine objenin çalışırken istenilen olayı gerçekleştirmesi için bir script'e bağlanır.

```
@ Unity İletisi | 0 bagvuru
void FixedUpdate()
{
    transform.position = new Vector3(anakarakter.position.x, anakarakter.position.y, -6);
}
```

Şekil 38 Kod

3.13.1 KAMERA TAKİBİ SCRIPT AÇIKLAMASI

Bu kod tanımladığınız nesnenin bulunduğu konumu değiştirmek için kullanılıyor ve 3 adet değişken alıyor. Burada X ve Y ile ana karakterin koordinatları giriliyor ve -6 değeri ile kamera uzaklığı tekrar değiştirilemeyecek şekilde ayarlanıyor. Vector3 sınıfı kullanılmasının sebebi kameranın Z ekseninde konumlanmış olması ve geliştiricinin Z değişkenine ihtiyaç duymasıdır. Kamera dışında diğer nesnelerin Z eksen koordinatı 0 iken Kamera nesnesinin Z eksen koordinatı 0 dan farklı bir değer olmak zorundadır.



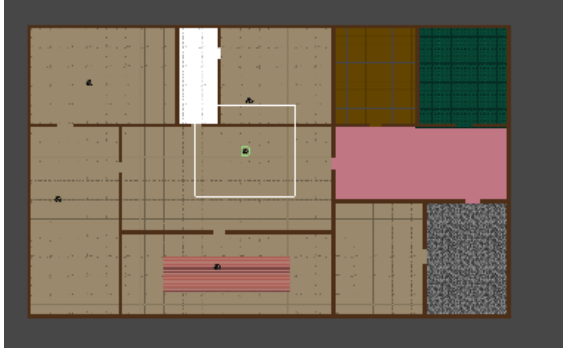
Şekil 39 Editör Paneli

Kamera ayarlarından sonra derinlik ayarından kamera derinliği ve **Dead Zone** (Ölü Bölge) kameranın gecikme ayarını ayarlamak için Ölü bölgeye 0.02 +0.02 değerleri veriliyor ki karakter hareket eder etmez kamera direkt harekete geçmesin. Bu şekilde ayarlandığı zaman kamera ile karakter hareketi arasında bir oynatım süresi farkı oluşuyor ve görsel olarak daha uygun bir görünüş sunuyor. Bu aşamada çeşitli olarak eklenen **CinemachineVirtualCamera** bileşeni ile çeşitli ayarlar yapılarak görsel sunum daha da iyileştirilebilir.

3.14 OBJE KOPYALAYARAK NESNE ÇOĞALTILMASI

Unity içerisinde bir objenin işlevselliği oluşturulduğunda veya bir objeye birden fazla alanda ihtiyaç duyulduğunda nesne dilenildiği gibi çoğaltılabilir ve çoğaltma işleminde bileşenlerini varsayılan olarak kaybetmez. CTRL+D kombinasyonu ile bir objenin **Duplicate** edilmesi sahne içerisinde mümkündür. Windows kısayollarından kopyalama kısayolu da yine aynı şekilde hiyerarşi içerisinde uygulanabilir.

Burada NPC olarak tabir edilen, düşman görevini üstlenecek ‘gardiyan’ olarak isimlendirdiğim nesneleri üretmek için tekrar tekrar obje oluşturma aşamaları işlenmesine gerek yoktur. Obje çoğaltma yöntemi ile gerekli bileşenler kapatılarak düşman objeleri oluşturulabilir.



Şekil 40 Harita



Şekil 41 Hiyerarşi

3.15 PROTECTED PROVIDE ÖZELLİKLERİ

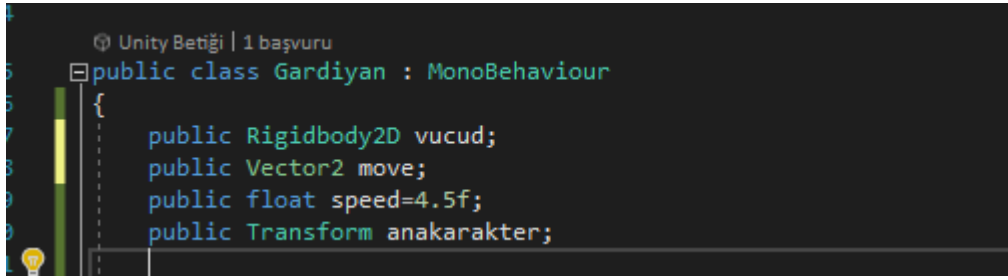
Protected Provide özellikleri C# programlama dilinin Nesneye yönelik programlama temellerine dayanan bir konudur. Unity içerisinde objelerin de kalıtım aldığı ve ona göre özellikler taşıdığı script aleminde objelerin ne şekilde tanımlanacağı geliştirici tarafından belirlenmeli ve ona uygun bir etiket ile oluşturulmalıdır.

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	X
no modifier	Y	Y	X	X
private	Y	X	X	X

Şekil 42 Sınıf ilişkisi

Bir değişkenin oluşturulma etiketi olarak korumalıdan korumasıza doğru giden bir tablo **şekilde gösterilmiştir**. Bu bir değişkenin editör panelinde gözükp gözükmeyeceği ve diğer scriptler tarafından bu sınıfın ulaşılabilir olup olmadığını belirleyen faktördür.

Unity içerisinde oluşturulan kodlar varsayılan olarak **MonoBehaviour** sınıfından kalıtım alarak gelir. Bu sınıf bir oluşturulan sınıfın Unity nesnesi olduğunu ve Unity değişkenlerini kullanmasına imkan veren bir taban kod sınıfıdır.



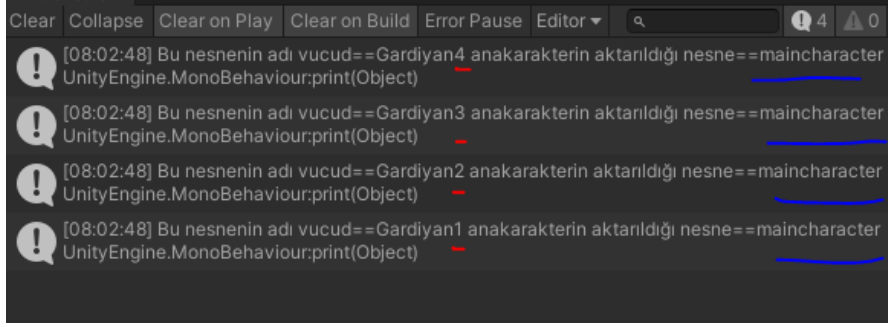
```
Unity Betiği | 1 başvuru
5 public class Gardiyan : MonoBehaviour
6 {
7     public Rigidbody2D vucud;
8     public Vector2 move;
9     public float speed=4.5f;
10    public Transform anakarakter;
```

Şekil 43 Kod

şekilde gardiyan nesnesinin aldığı kalıtlar ve özelliklerinin etiket almaları gösterilmiştir.

3.16 İLERLEMİYİ KONSOLDA GÖRMEK

Unity yazılımında her tümleşik geliştirme ortamında olduğu gibi bir konsol ekranı bulunmaktadır ve bu konsol ekranı geliştiricinin süreci takip etmesini kolaylaştırmaktadır. Geliştiricinin konsolda görmek istediği bir değişken veya bir olayı **Print** veya **Debug.Log** kodlarını gerekli Unity Fonksiyonu içerisinde yazması, ve diğer scriptlerde olduğu gibi bu kodun aktif olarak sahnede bileşen olarak bulunması gerekmektedir.



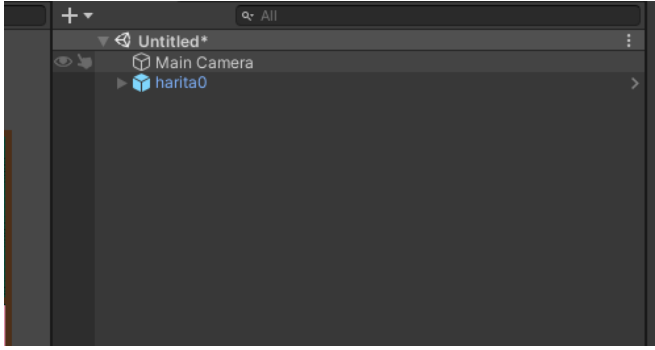
Şekil 44 Konsol Ekranı

3.17 PREFAB KULLANIMI

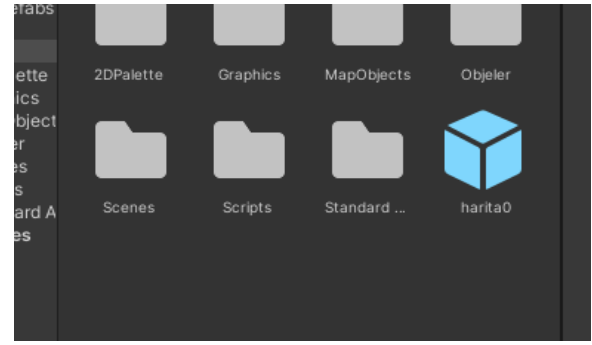
Unity yazılımı içerisinde dosyaların paketlenme sistemi mevcuttur. Bu paketleme sistemi sayesinde bir obje kalıp olarak kaydedilir ve tekrar tekrar kopyalanmak yerine ana bir kalıptan yönetilebilir, aynı zamanda başka sahnelerde obje taşıma işlemini kolaylaştırmak açısından oyun geliştiriciye sunulmuş bir teknolojidir. Prefab Unity3D programı için geçerli bir terim olmasına rağmen programlama dillerinde bazı terimlere benzetilerek açıklanabilir.

Unity içerisinde oyun geliştiricisi herhangi bir nesneyi tekrar kullanmak istediğinde onu Prefab haline getirir. Kısaca aynı nitelikte objeleri tekrar tekrar oluşturmak yerine bir defa oluşturup aynı sahnede veya farklı sahnelerde bu objeyi kullanma işlemine Prefab denir.

Unity’de Prefab oluşturmak için objenin hiyerarşiden proje dosyalarına sürüklenip bırakılması yeterli olduğu gibi objeye sağ tıklanarak da obje Prefab haline getirilebilir.



Şekil 46 Hiyerarşi

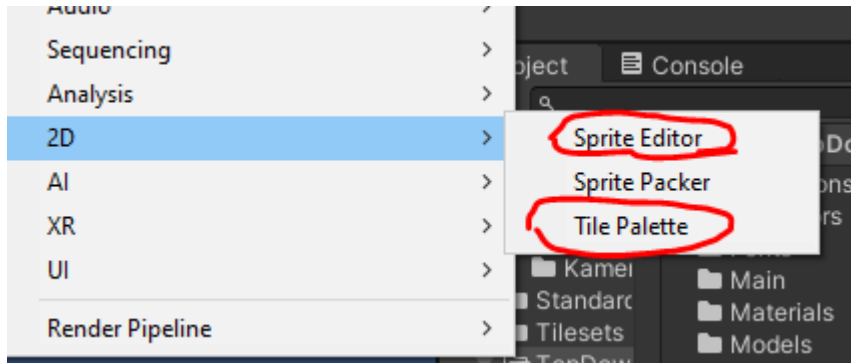


Şekil 45 Dosya Boyutu

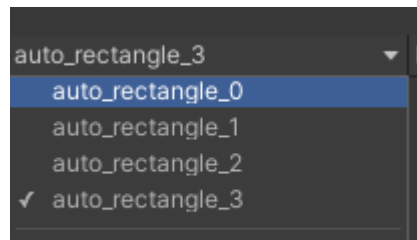
(Unity’de Prefablar mavi boyalı küpler ile gösterilir / temsil edilir.)

3.18 TİLEMAP PALET KULLANIMI

Unity içerisinde tilemap oluşturulurken diğer çizim-grafik yazılımlarında olduğu gibi bir palet kullanımı söz konusudur. Burada tilemap üzerindeki objelerin sprite’larını düzenlemek için **Sprite Editor** kullanılır. Görüntüler eklendikten sonra düzlem üzerinde düzenlemesini yapmak için **Tile Palette** kullanılır.



Şekil 47



Şekil 48

Bu projede“Backyard-Free“ isimli varlık Asset Store’dan projeye import edilmiştir.

(“Kittens and Elves Works” isimli bu çalışmalarından dolayı teşekkür ederim.)

<https://assetstore.unity.com/publishers/10908> Studyo’nun linki;

3.19 UNITY LAYER KULLANIMI

Unity yazılımında **Layer** (katman) kavramı oyun geliştiricisinin objelerin aktif olduğu uzayları birbirinden ayırması ve görüntülerin hangisinin daha önde gösterileceği gibi konularda yardımcı olması amacıyla kullanılan bir kavramdır. Layer katmanı aynı olan objelerin aynı düzlemde olduğu varsayılabilir. Layer katmanı daha yüksek olan bir obje diğer objeden daha sonra render’lanacağından harita dizaynı ve obje ilişkilerinde bu kavramın kullanımına dikkat edilmesi gerekir.

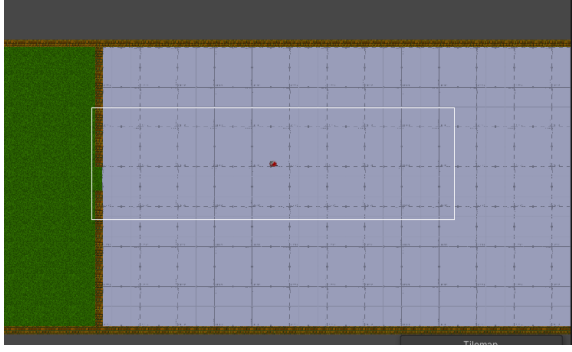


Şekil 49 Layer Paneli

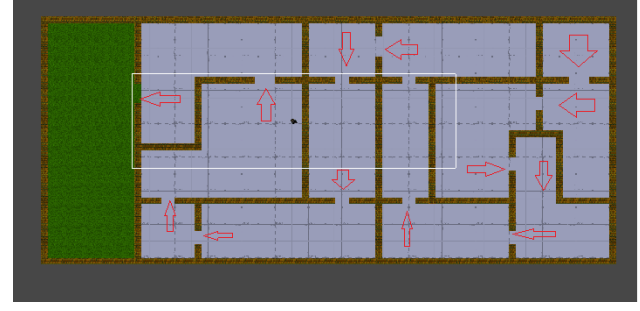
Şekilde bir objenin kaçınıcı katmanda render edileceği belirlenmiştir.

3.20 HARİTA OLUŞTURULMASI

Harita şekildeki aşamalarla bir dikdörtgen içerisinde ilk hatta çimlerin, sonra parkelerin, sonra da duvarların olacağı şekilde 3 farklı tilemap üzerinde oluşturulmuş ve boyama işlemi gerçekleştirilmiştir.



Şekil 51 Harita Tasarım-1



Şekil 50 Harita Tasarım-2

Haritanın tasarımı bittikten sonra şekil-2 de gözüktüğü hali alıyor. Platform oyunlarında haritanın başlangıç noktasından, bölümü bitirme noktasına kadar dışarı çıkabilmesi mümkün olmayacak şekilde oyuncuya belli bir rotayı izletmek gerekir. Aksi takdirde oyun ilerlemesini imkansız hale getirecek tercihler oyuncu tarafından yapılabilir ve bu programcı tarafından engellenmelidir. Her ne kadar bu proje bir platform oyunu olmasa da projede, oyuncunun çizilen binanın dışına çıkması istenilen bir olay değildir dolayısıyla tasarım bu şekilde yapılmıştır.

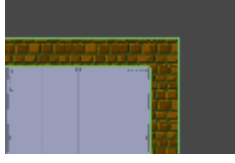
3.21 COMPOSITE COLLİDER İLE TILEMAP COLLİDER BİRLEŞTİRİLMESİ

Collider bileşeninin gelişmiş versiyonu olan Composite Collider bileşeni eklendiği nesnenin parça parça olarak sahnede bulunmasına rağmen oyun motoru tarafından parça parça bölünmesinin önüne geçerek bir bütün olarak ele almasını sağlayan bileşendir. Bu bileşen uygulandığı nesnenin dış çevresini birleştirerek bir bütün haline getirir ve bu sayede istenmeyen bazı durumların önüne geçilmiş olur.

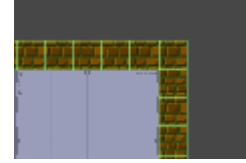
Bu projede Composite Collider Tilemap üzerine uygulanan collider ile birleştirilerek ‘duvarlar’ nesnesinin bir bütün olarak çarpıştırıcıya sahip olmasını sağlamıştır.

Burada dikkat edilmesi gereken husus Composite Collider bir Rigidbody bileşeni ile varsayılan olarak beraber gelir ve bu bileşen olmadan çalışamaz. Kuvvet vektörlerine

maruz kalması açık olacağından dolayı bu bileşen duruma göre statik ve dinamik olarak ayarlanmalıdır.



Şekil 52 Composite Collider 2D



Şekil 53 Tilemap Collider 2D

3.22 ÇARPRAZ HARAKETLENME

Bu aşamada daha önce 4 ana temel yön üzerine kurulmuş hareket teması 8 ‘ e çıkarılmış ve yönler

$X > 0$ için; $Y > 0 \rightarrow$ Kuzeydoğu, $Y = 0 \rightarrow$ Doğu, $Y < 0 \rightarrow$ Güneydoğu

$X < 0$ için; $Y > 0 \rightarrow$ Kuzeybatı, $Y = 0 \rightarrow$ Batı, $Y < 0 \rightarrow$ Güneybatı

$X = 0$ için; $Y > 0 \rightarrow$ Kuzey, $Y = 0 \rightarrow$ Durağan, $Y < 0 \rightarrow$ Güney

Tabloda gözüktüğü şekilde ayarlanıp script’e işlenmiş ve ana karakter objesine eklenmiştir.

```

if (move.x > 0)
{
    if (move.y > 0 )
    { //sag yukari gidiyor yani kd aktif
        this.transform.rotation = Quaternion.Euler(0, 0, 45);
    }
    else if (move.y < 0 )
    {
        // sag asagi gidiyor yani gd aktif
        this.transform.rotation = Quaternion.Euler(0, 0, -45);
    }
    else
    {
        // sag gidiyor yani d aktif
        this.transform.rotation = Quaternion.Euler(0, 0, 0);
    }
}
else if (move.x < 0)
{
    if (move.y > 0 )
    { //sol yukari gidiyor yani kb aktif
        this.transform.rotation = Quaternion.Euler(0, 0, 135);
    }
    else if (move.y < 0 )
    {
        // sag asagi gidiyor yani gb aktif
        this.transform.rotation = Quaternion.Euler(0, 0, -135);
    }
    else
    {
        // sag gidiyor yani b aktif
        this.transform.rotation = Quaternion.Euler(0, 0, 180);
    }
}
}

```

Şekil 54 Kod

```

}
}
else
{
    if (move.y > 0)
    {
        // karakter yukari gidiyor yani k aktif
        this.transform.rotation = Quaternion.Euler(0, 0, 90);
    }
    else if (move.y < 0)
    {
        //karakter asagi gidiyor yani g aktif
        this.transform.rotation = Quaternion.Euler(0, 0, -90);
    }
    else
    {
        //karakter duruyor.
    }
}
}

```

Şekil 55 Kod

3.22.1 ÇAPRAZ HAREKETLENME AÇIKLAMASI

```

/*
 * ---- Rotasyon z eksenini ----
 * kd =kuzeydogu (45)
 * k = kuzey (90)
 * kb =kuzeybatı (135)
 * b = batı (180)
 * gb =guneybatı (-135)
 * g = guney (-90)
 * gd =guneydogu (-45)
 * d =dogu Rotation (0)
 */

```

Şekil 56 Kod

Projedeki yön verme algoritmasını zenginleştirmek adına 8 yön matematiksel olarak oluşturuldu ve bu yönler için objenin dönüş değerleri ayarlandı. Kodlamada daha verimli ve hızlı çalışmak adına gerekli hesaplamalar yapılarak kod içerisine yorum satırı olarak şekildeki gibi eklendi.(Şekil-1)

Burada kullanılan algoritmada

1-)X>0 , 2-)X<0 , 3-)X=0

Koşullarına sahip 3 ana koşul var. Ve her koşulun içerisinde

1-)Y>0 , 2-)Y<0 , 3-) Y=0

Şeklinde 3 iç koşul var.

Dış koşullarda karakterin X eksenindeki -1,0,1 değerleri tespit edilip iç koşullarda yine aynı şekilde karakterin Y eksenindeki 1,0,-1 değerleri tespit edilmiştir ve buna göre yön koordinatları oluşturulmuştur. Daha sonra bu konumlara uygun açılar 'maincharacter' nesnesinin Transform Component'i içerisindeki Rotate:Z değerine aktarılmıştır.

3.23 IŞIK EKLEME

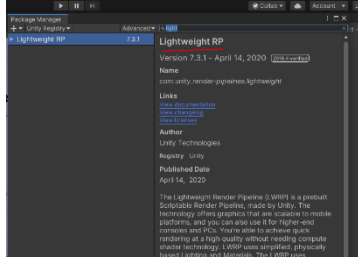
Oyun yazılımı konusunda geliştiriciyi en çok zorlayan konulardan birisi de ışık eklemektir. Sanal bir uzay olan oyun evrenine ışık kazandırmak için çeşitli yazılımlar ve hazır kütüphaneler bulunur. Unity içerisinde diğer tüm nesnelerde olduğu gibi ışık da sahnede boş bir objedir ve sahip olduğu bileşenler ile ışık objesine dönüşür.

Işık bir uzayda oluştuğunda ilk olarak aydınlıklar ve karanlıklar oluşur. Gerçek hayatta bir ışığı fark etmek için o ışığın bir nesneye çarpması gerekir. Çarptığı nesnede aydınlık oluşturur ve görülebilir hale getirir. Bu uzayda da aydınlıklar ve karanlıklardan bahsedebiliriz fakat bunlar el ile ayarlanmıyor, projeye eklenen ışık Unity Motoru ile etkileşimli bir şekilde çalışıyor ve otomatik ayarlanıyor.

Fakat ışık bileşeni 2D projede gerçekleştirmek istenince oyun geliştiricisi şu anki güncellemeye sahip Unity içerisinde bir şekilde çaresiz kalıyor. Sebebi ise Unity 2D için yeterli bir uygulama gibi gözüксе de bazı noktalarda eksik yanları kalıyor ve programcuyu zorluyor. Programcuyu çözmesi gereken olağandan daha fazla problemle karşı karşıya bırakıyor. Unity kütüphanelerinde gerekli düzenlemeler yaptırmak zorunda bırakıyor.

Sebebi ise Unity yazılımının iki boyuttaki ışık bileşenini hala geliştiriyor olması ve bu bileşenin deneysel olarak program içerisinde kullanılıyor olmasıdır. Bu paketi sorunsuz bir şekilde geliştiricinin proje paketine dâhil etmesi ve ayarlarını projeye göre yapması gerekiyor.

Unity yazılımında sektördeki diğer tümleşik geliştirme ortamlarında olduğu gibi bir paket yöneticisi var ve bu paket yöneticisi ile gerekli paketler projeye dâhil ediliyor.



Şekil 57 LightWeight Rp

Şekilde Unity içerisindeki güncel ışık paketi gösterilmiştir.

Package Menager



LightWeight RP



Download



Import

Tabloda eklenme aşamaları İngilizce dili için varsayılan olarak gösterilmiştir.

3.23.1 IŞIK AYARLARI

Projede kullanılan ışık bileşeni projeye eklenmesi için birkaç ayar yapılması gerekiyor. Projeye ekleme kısmı bittikten sonra, bu kütüphaneyi temel alan nesneler oluşması gerekiyor. Proje dosyaları içerisinde boş bir klasöre alttaki işlemler uygulanarak sorunsuz takip edilebilir.

Fare Sağ tık



Rendering



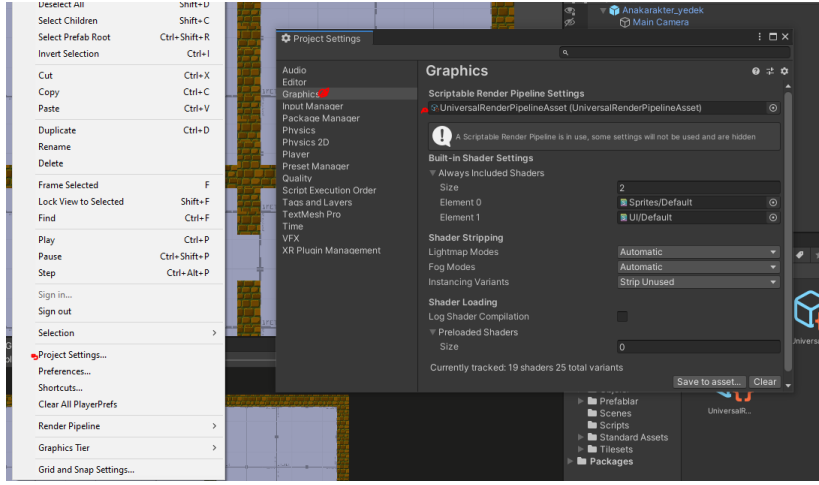
Universal Render Pipeline



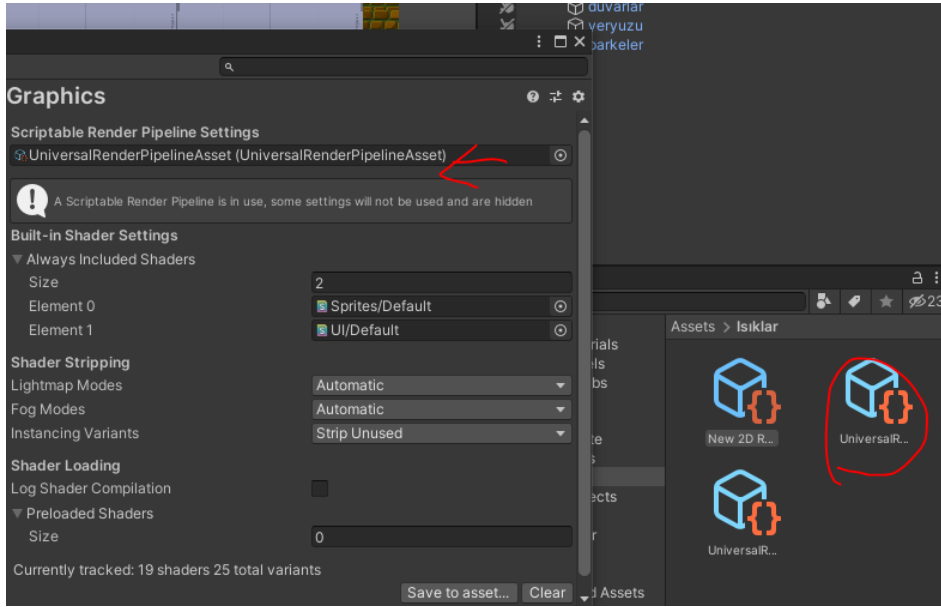
2D Renderer

Dosyası oluşturuluyor.

Sonraki aşamada ise yüklenen ışık dosyalarını proje içerisine kurulması gerekiyor. Çünkü Unity default olarak bu değerleri değil kendi ışık değerlerini kullanıyor. Bu aşama yapılmadığı takdirde eklediğimiz nesne sahnede işlenmiyor (render). Bu bir problem değil bu bir Unity ayarı olarak geçiyor çünkü deneysel olan kodlamaları farklı cihazlarda çalıştırabilmek için yoğun bir optimizasyon gerekebiliyor ve genelde büyük firmalar bu tarz deneysel bileşenlerden uzak durabiliyor.



Şekil 58 Ayarlar



Şekil 59 Ayarlar

3.23.2 IŞIKLARIN TANITIMI

Unity yazılımındaki ışık paketi içerisinde güncel olarak sahneye eklenebilecek 5 farklı ışık türü bulunuyor.

Freeform Light: Bu ışık tipinde istenilen formu geliştirici Editor ile ışığa verebilir.

Sprite Light: Bu ışık tipinde istenilen görüntüyü geliştirici ışığa verebilir. Varsayılan olarak Freeform ışık tipinin şeklini alıyor.

Parametric Light: Bu ışık tipinde geliştirici n sayılı istenilen bir poligon üreterek bu alanı aydınlatan bir ışık verebiliyor.

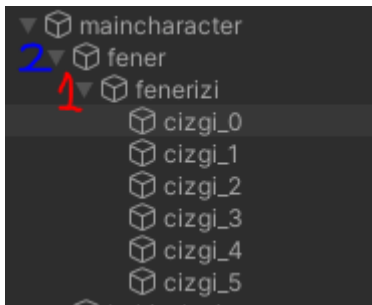
Point Light: Bu ışık tipinde geliştirici r yarıçaplı bir küresel verebiliyor.

Global Light: Global ışık sahnenin her objesi ve her nesnesi üzerinde tepeden oyun içinde evrensel bir ışık oluşturulmasını sağlıyor.

3.24 OBJEYE BAŞKA BİR OBJENİN EKLENMESİ

Unity içerisinde bir objenin çalışma süresi içerisinde veya sahne içerisinde bir başka obje ile birlikte çalışmasına gerek duyulduğunda geliştiricinin bu iki objeyi birbirine eklemesi gerekir. Bir obje eklendiği başka bir obje olduğu zaman yerel yeni değerler kazanır ve o objenin eklendiği editör panelindeki ayarlanmış koordinatlarını bozmayacak şekilde onunla birlikte hareket eder.

Bu eklenme gerçekleştiğinde aralarında bir **Parent- Child** ilişkisi doğuyor ve eklenen obje eklendiği objenin çocuğu olarak oyun motoruna otomatik olarak tanıtılıyor.

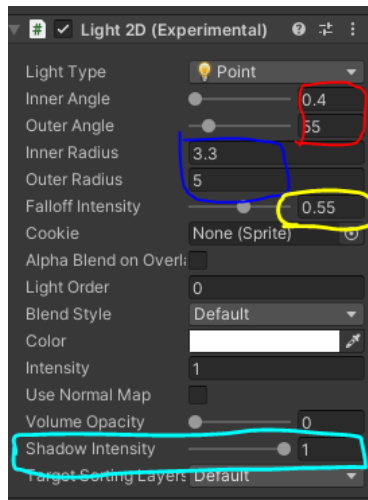


Şekil 60 Parent-Child ilişkisi

3.24.1 ANA KARAKTERE FENER EKLENMESİ

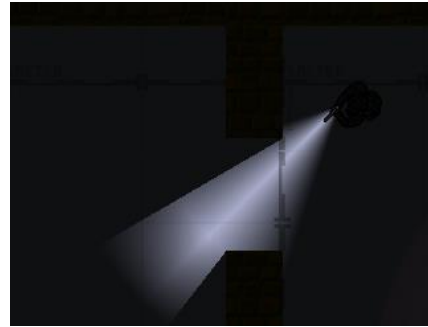
Ana karakterin oyun işleyişini kontrol açısından bir fenera ihtiyacı var. Bu fener objesini projede eklemek için bir objeye başka bir objenin eklenmesi yöntemine başvuruluyor. Bu yöntem ile ana karakterin bir fener tutması sağlanıyor.

Eklenen objenin bileşen olarak noktasal ışık kaynağı seçiliyor bu sayede küresel olan ışığın açısı değeri ayarlanarak fener görüntüsü veriliyor.



Şekil 61 Işık Özellikleri

Eklenen ışık için bileşen değerleri şekilde verilmiştir.



Şekil 62 Oyun içi Görüntü

3.25 TRİGGER VE COLLİSON KAVRAMLARI

Oyun geliştirici yazılımlarında geliştirici objelerin birbirine temasını kontrol ederek bunlara olaylar ekler. Bu olaylar oyun yapısını ve algoritmasını kontrol eden temel etmenleri oluşturur. Unity yazılımında bu çarpışmalar iki ana Unity taban fonksiyonu ile işlenir. Yazılımcı bu fonksiyonlardan uygun olanını script olarak objeye atmasını gerçekleştirir ve uygun olanını kullanır.

Collision detection occurs and messages are sent upon collision						
	Static Collider	Rigidbody Collider	Kinematic Rigidbody Collider	Static Trigger Collider	Rigidbody Trigger Collider	Kinematic Rigidbody Trigger Collider
Static Collider		Y				
Rigidbody Collider	Y	Y	Y			
Kinematic Rigidbody Collider		Y				
Static Trigger Collider						
Rigidbody Trigger Collider						
Kinematic Rigidbody Trigger Collider						

Trigger messages are sent upon collision						
	Static Collider	Rigidbody Collider	Kinematic Rigidbody Collider	Static Trigger Collider	Rigidbody Trigger Collider	Kinematic Rigidbody Trigger Collider
Static Collider					Y	Y
Rigidbody Collider				Y	Y	Y
Kinematic Rigidbody Collider				Y	Y	Y
Static Trigger Collider		Y	Y		Y	Y
Rigidbody Trigger Collider	Y	Y	Y	Y	Y	Y
Kinematic Rigidbody Trigger Collider	Y	Y	Y	Y	Y	Y

Şekil 63 Collison Trigger Farkı

Her iki fonksiyon da çarpışmayı kontrol eder aralarındaki temel fark şudur;

Trigger:

Bir objenin diğer objenin içinden geçtiğinde oluşacak durumları kontrol eder.

Collison:

İç içe giremeyen iki nesnenin çarpıştığında oluşacak durumları kontrol eder.

Her iki kavramın da alt 3 fonksiyonu vardır.

Enter: Çarpışma olduğu anda bir defa çalışması gereken olayı temsil eder.

Exit: Çarpışma bittiği anda bir defa çalışması gereken olayı temsil eder.

Stay: Çarpışma olduğu süre boyunca sürekli çalışması gereken olayı temsil eder.

3.26 KODLAR İLE İŞLEMLERİN YÖNETİLMESİ

3.26.1 FENER OBJESİNE FENER ALANI EKLENMESİ

Bu eklemenin amacı ışığın duvarlardan geçtiğindeki algılama sistemini yapay zekâya da tanıtmaktır.

Fener alanı için 5 adet kutu çarpıştırıcı fener alanı objesini parent olarak oluşturuluyor



Şekil 64 Oyun içi Görüntü

ve uygun değerler ile objeye ekleniyor

3.26.2 FENER ALANI SCRIPT KODU EKLENMESİ

Fener alanı kodu her bir kutu çarpıştırıcı nesnesine ekleniyor.

```
}  
© Unity İletisi | 0 başvuru  
private void OnTriggerEnter2D(Collider2D digernesne)  
{  
    if(digernesne==cevre_0)  
    {  
        if(cizgialani.size.x>0.20f)  
        {  
            cizgialani.size = new Vector2(cizgialani.size.x-0.15f, ilkhali_size_y);  
            cizgialani.offset = new Vector2(cizgialani.offset.x +0.075f, ilkhali_offset_y);  
            Debug.Log("Kücülüyorum"+this.name+". için x degerim: "+cizgialani.size.x);  
        }  
    }  
}
```

Şekil 65 Kod

```

void Update()
{
    if (!carpisiyor)
    {
        if (cizgialani.size.x < ilkhali_size.x)
        {
            cizgialani.size = new Vector2(cizgialani.size.x + 0.30f, ilkhali_size.y);
            cizgialani.offset = new Vector2(cizgialani.offset.x - 0.15f, ilkhali_offset.y);
        }
    }
}

```

Şekil 66 Kod

3.26.2.1 FENER ALANI SCRIPT KODU AÇIKLAMASI

Çarpışmanın aktif olup olmadığı kontrolü dışarıdaki if koşulu ile kontrol ediliyor. BoxCollider2D nesnesinin **size.x** değeri kutunun x eksenindeki boyutunu belirlediğinden, ilk halindeki boyutundan kısa olup olmadığı içerideki if döngüsü ile sağlanıyor. Bu döngü algoritmada eğer Collider, ilk halinden başka bir boyuta geçtiyse, Collider'ın tekrardan kendi formuna geçmeye çalışmasını sağlıyor, sistemi sürekli bunun için çalıştırıyor. Bu sayede ilk boyutuna sürekli dönmeye çalışan bir Collider elde etmiş oluyoruz.

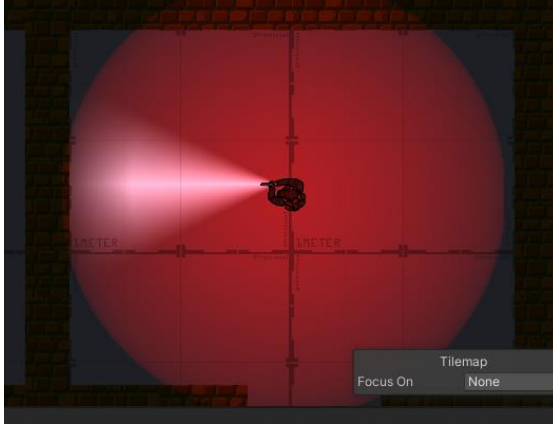
0.30f değeri ile kısalma hızı

0.15f değeri ile de çizginin karakter dışına çıkmaması sağlanıyor.

Burada ne olursa olsun kısalma hızının, iz düşüm değerinin 1 / 2 değerinde olması gerekiyor.

3.26.3 BAKIŞ ALANI OBJESİNİN ANA KARAKTERE EKLENMESİ

Bu eklemenin amacı ana karakterin farkındalık seviyesini oyuncuya göstermek oluyor.



Şekil 67 Kod

Bakış alanı objesi farklı kodlar ile çalışacağından kendinden sahip olduğu bir script tanımlanmıyor.

3.26.4 FENER KONTROL SCRIPTİNİN ANA KARAKTERE EKLENMESİ

Oyun geliştirmede oyuncu karakteri yönetmek için kullandığı yön tuşlarının yanı sıra ekstra tuşlar veya tıklamalar ile yönettiği karakterin veya sistemin farklı işlevlerini kullanır. Bu projede eklenen fener objesinin açılıp kapanma olayı bir script ile karakterin uygun child nesnesine bağlandı.

Bu aşamada dikkat edilmesi gereken husus nesnenin aktif ve pasif halini kodlamada kullanırken bu aktifliğin child'lara da etki edeceği ve senaryo başladığında aktif olmayan nesneye erişim sağlanamayacağıdır.


```

@ Unity İletisi | 0 başvuru
void Start()
{
    bakisalaniobjesi = this.transform.GetChild(1).gameObject;
    globalisikobjesi = GameObject.FindGameObjectWithTag("globalisik");
    fenerobjesi = this.transform.GetChild(0).gameObject;

    if (this.gameObject.activeSelf == false) //fener acik kapali kontrol
    {
        feneracikmi = false;
    }
    else
        feneracikmi = true;
}

```

Şekil 68 Kod

```

void Update()
{
    if (Input.GetKeyDown(fenertusu))//Tuş kontrol
    {
        if (feneracikmi == true)
        {
            bakisalaniobjesi.SetActive(true);
            fenerobjesi.SetActive(false); //fener kapandığında olanlar
            globalisikobjesi.SetActive(false);
            feneracikmi = false;
        }
        else
        {
            bakisalaniobjesi.SetActive(false);
            fenerobjesi.SetActive(true); //fener açıldığında olanlar
            globalisikobjesi.SetActive(true);
            feneracikmi = true;
        }
    }
}

```

Şekil 69 Kod

3.26.4.1 FENER KONTROL SCRİPT AÇIKLAMASI

Bool “feneracikmi” değişkeni fenerin açık olup olmadığını kontrol edecek;

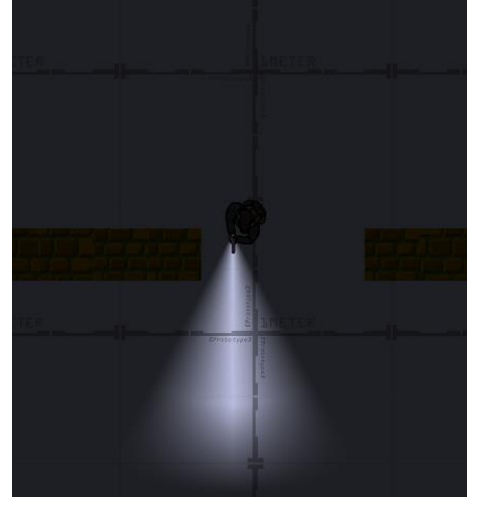
GameObject “bakisalaniobjesi,globalisikobjesi,fenerobjesi,cevre” değişkenleri sırası ile “bakisalani,globalisik,fener,cevre_0” nesnelerini temsil etmek ve kod içerisinde kullanmak için kodlandı.

Start() içerisinde kullanılacak nesneler temsil ettiği objelere atandı.

İf kontrolü ile fenerin oyun başlangıcında açık olup olmadığı belirlendi.



Şekil 71 Oyun içi Görüntü



Şekil 70 Oyun içi görüntü

Fener kapalı iken oluşan görüntü

Fener açıkken

3.26.5 DUVAR YANSIMA KONTROL SCRIPTİ EKLENMESİ

Fener objesinin açılıp kapanması ile ışık objesi açılıp kapanırken diğer objelerin görüntülerini kontrol edecek dosya projeye eklendi.

```

public class duvaryansimakontrol : MonoBehaviour
{
    GameObject globalisikobjesi;
    KeyCode fenertusulokal;
    // Start is called before the first frame update
    @ Unity İletisi | 0 bayvuru
    void Start()
    {
        globalisikobjesi = GameObject.FindGameObjectWithTag("globalisik");
        fenertusulokal = GameObject.FindGameObjectWithTag("anakaraktertag").GetComponent<fener_kontrol>().fenertusu;
    }

    // Update is called once per frame
    @ Unity İletisi | 0 bayvuru
    void Update()
    {
        //Debug.Log("selfshadow= "+this.GetComponent<TilemapShadowCaster.Runtime.TilemapShadowCaster2D>().m_SelfShadows);
        if (globalisikobjesi.activeSelf && Input.GetKeyDown(fenertusulokal))
        {
            this.GetComponent<TilemapShadowCaster.Runtime.TilemapShadowCaster2D>().m_SelfShadows = true;
            this.GetComponent<TilemapShadowCaster.Runtime.TilemapShadowCaster2D>().ReinitializeShapes();
        }
        else if (!globalisikobjesi.activeSelf && Input.GetKeyDown(fenertusulokal))
        {
            // Debug.Log("setShadow=1");
            this.GetComponent<TilemapShadowCaster.Runtime.TilemapShadowCaster2D>().m_SelfShadows = false;
            this.GetComponent<TilemapShadowCaster.Runtime.TilemapShadowCaster2D>().ReinitializeShapes();
        }
    }
}

```

Şekil 72 Kod

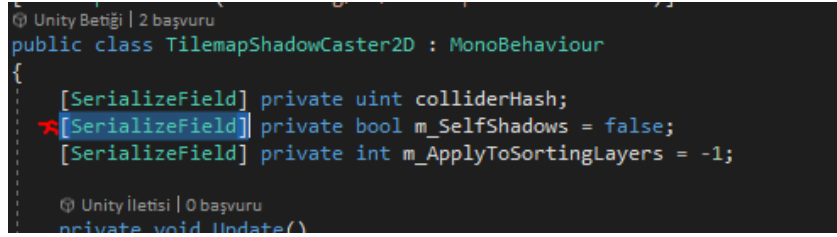
Burada manuel olarak kullanılan gölge oluşturma komutunun kendi gölgelerini aç isimli boolean değer fenerin açık kapalı olduğunun bilgisine göre false veya true olarak ayarlanıyor.

3.26.5.1 DUVAR YANSIMA KONTROL SCRIPT AÇIKLAMASI

İlk olarak “globalisikobjesi, KeyCode fenertusulokal” isimli değişkenler uygun nesneleri temsil etmesi için tanımlanıyor. Daha sonra public olarak tanımlanan fener_kontrol.cs ‘teki fenertusu değişkeni fenertusulokal değişkenine aktarılıyor. Globalisik objesi bilindik bir yöntemle tekrardan temsili olarak alınıyor. Kodlarda gözüktüğü üzere Update() fonksiyonu içerisinde yazarak sürekli kodların aktif olarak çalışması sağlanıyor ve bir adet if-else bloğu ile global ışığın açık mı kapalı mı olduğunu, bunun yanı sıra fener tuşuna basılıp basılmadığını kontrol eden bir koşul bloğu oluşturuluyor. Burada dikkat edilmesi gereken husus SelfShadows nesnesinin aktifliği değiştirildikten sonra ReinititalizeShapes() fonksiyonu ile değer değiştirilen fonksiyon tekrar çağrılıyor. Çünkü bu kod ExecuteInEdit anahtar komutuyla kodlanmış olup editör ekranında değişiklik yaşanmadığı sürece tekrardan gölgeleri oluşturmuyor, dolayısıyla geliştirici içeriden bir değişiklik yaptığında yapıyı bozmamak adına ExecuteInEdit modunu değiştirmiyor. Fakat değişikliklerin aktif olması için fonksiyonu Runtime içerisinde tekrar çağırarak oyun esnasında gölgelerin

devamlı olarak ışık kapatılıp açıldığında oluşmasını sağlıyor. Bu fark gözle görülebilir bir fark olmasa da programcı olarak gölgelerin sürekli bir build işlemi içerisinde olduğu fark ediliyor.

3.26.5.2 SERIALIZEFIELD KULLANIMI



```
Unity Betiği | 2 başvuru
public class TilemapShadowCaster2D : MonoBehaviour
{
    [SerializeField] private uint colliderHash;
    [SerializeField] private bool m_SelfShadows = false;
    [SerializeField] private int m_ApplyToSortingLayers = -1;

    Unity İletisi | 0 başvuru
    private void Update()
    {
    }
}
```

Şekil 73 Kod

Burada [SerializeField] yazılı komut bu nesnenin Inspector Panelinden ulaşılabilirliğini aktif hale getiriyor. Fakat temel olarak Unity zaten public değerleri Inspector paneli üzerinden değişimine imkân sunuyor. Dolayısıyla buradaki SerializeField alanı öznel olarak kaldırılabilir. Önemli olan nokta burada m_SelfShadows değişkenini public olarak tanımlayarak ulaşılabilir hale getirmektir

3.26.6 KARAKTERİN KOŞU VE YAVAŞ YÜRÜME FONKSİYONLARININ KAZANDIRILMASI

Çoğu oyun projesinde olduğu gibi karakterin hız değişkenini kontrol eden bir script dosyayı karaktere tanımlanıyor. Bu projede hali hazırda oyuncunun bastığı bir tuş algılama kod scripti var olduğundan karışıklık olmaması adına o koda eklemeler yapılıyor.

```

// Update is called once per frame
@ Unity İletisi / @ başvuru
void Update()
{
    if (Input.GetKeyDown(KeyCode.LeftShift)) // basıldığı anda (1 kez çalışır)
    {
        this.GetComponent<karakterinyuruyusu>().speed = this.GetComponent<karakterinyuruyusu>().speed + hizlandirmakatsayisi;
    }

    if (Input.GetKeyUp(KeyCode.LeftShift))// cektigi saniye (1 kez çalışır)
    {
        this.GetComponent<karakterinyuruyusu>().speed = this.GetComponent<karakterinyuruyusu>().speed - hizlandirmakatsayisi;
    }

    if (Input.GetKeyDown(KeyCode.LeftControl)) // basıldığı anda (1 kez çalışır)
    {
        this.GetComponent<karakterinyuruyusu>().speed = this.GetComponent<karakterinyuruyusu>().speed - yavaslatmakatsayisi;
    }

    if (Input.GetKeyUp(KeyCode.LeftControl))// cektigi saniye (1 kez çalışır)
    {
        this.GetComponent<karakterinyuruyusu>().speed = this.GetComponent<karakterinyuruyusu>().speed + yavaslatmakatsayisi;
    }
}

```

Şekil 74 Kod

Toplamda 4 adet if bloğu ile LeftShift ve LeftControl Tuşlarını (şimdilik) değiştirilemez olarak belirlenmiş oluyor ve bu tuşlara basıldığında (**GetKeyDown**) ve Tuş basılması bırakıldığında (**GetKeyUp**) Unity fonksiyonları ile kontrol ediliyor.

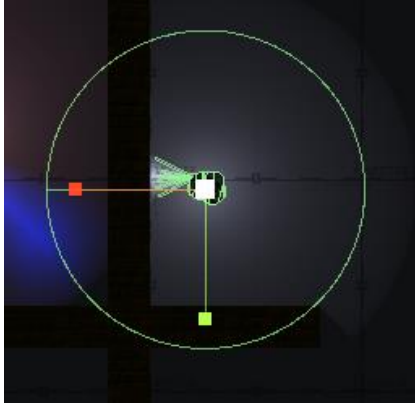
LCTRL → basıldığında karakterinyuruyusu isimli kodun public speed değişkenini hızlandırma katsayısı kadar arttır.

Kaldırıldığında karakterinyuruyusu isimli kodun public speed değişkenini hızlandırma katsayısı kadar azalt.

LSHIFT → üstteki işlemlerin tam tersini yavaşlatma katsayısı ile tekrarla.

3.26.7 KARAKTERE SES ÇEMBERİ OBJESİNİN EKLENMESİ

Bu aşamada ana karakterin ses çemberini kodlayarak gardiyanların ana karakteri duymasını sağlanıyor. Duyduktan sonra ne tarafa hareket edeceğini yapay zeka'nın ses algılama kabiliyeti ile birlikte işleniyor.



Şekil 75 Circle Collider

3.26.7.1 SES ÇEMBERİ SCRIPTİNİN KARAKTERE EKLENMESİ

```

35 // Update is called once per frame
36 @ Unity İletisi / 0 basvuru
37 void FixedUpdate()
38 {
39     if (anarakteriyurumekodutemsili.haraketedyormuyum == true && carpisiyor == false) //carpisiyor bir ise yar
40     {
41         sescember.enabled = true;
42         if (hizdegiskeni.speed < baslangichiz && sescember.radius != yavasradius) // Debug.Log("Yavas Yuruyorum")
43         {
44             sescember.radius = yavasradius;
45         }
46         else if (hizdegiskeni.speed > baslangichiz && sescember.radius != hizliradius) // Debug.Log("Kosuyorum")
47         {
48             sescember.radius = hizliradius;
49         }
50         else if (hizdegiskeni.speed == baslangichiz && sescember.radius != baslangicradius) //Debug.Log("Normal Y
51         {
52             sescember.radius = baslangicradius;
53         }
54     }
55     else if (anarakteriyurumekodutemsili.haraketedyormuyum == false && carpisiyor == false) //Debug.Log("Duruyo
56     {
57         sescember.enabled = false;
58     }
59 }
60

```

Şekil 76 Kod

3.26.7.2 SES ÇEMBERİ SCRIPT AÇIKLAMASI

Bu aşamada çemberin büyüklüğü if blokları ile kontrollü değiştirilerek sesin çok veya az çıkması sağlanıyor. Fakat arada duvar olup olmadığını kontrol ederek bu nesnenin aktif olup olmadığı değiştirileceği için değişkenler public tanımlanıyor.

```

5 public class Karakterinyuruyusu : MonoBehaviour
6 {
7
8     public Rigidbody2D rb; // Fizik yasalarını uygulamak için 2d Fizik motoru
9     Vector2 move; // hareket etmeyi sağlayan vektör
10    public float speed;
11    private string nereyebakıyor; //eski kodlar için duruyor silme*
12
13    [HideInInspector] public bool haraketediyormuyum = false;
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Şekil 77 Kod

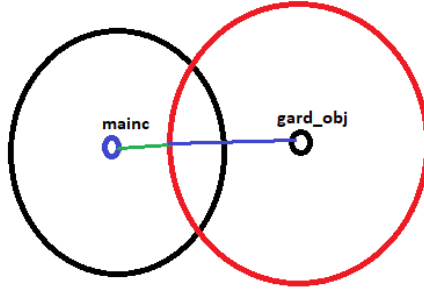
HideInInspector: Bu kod sayesinde public olarak atanılan fonksiyon geliştirici tarafından normalde editör panelinde erişilebilir olurken, bu kod etiketini alınan değişken editör panelinden gizleniyor.

3.26.7.3 PROJEDE MANTIKSAL HATA ÇÖZÜMÜ

Unity yazılımında bir objenin sahip olduğu çarpıştırıcı tek bir çarpışma olayına gidebiliyor. Programcı bir objedeki birden fazla çarpışmayı kontrol edebilmek için farklı yollara gidiyor veya çarpıştırıcıları parçalamak zorunda kalıyor. Alternatif yollar üretmek oyun geliştiricisinin bir mecburi görevi haline geliyor.

3.26.7.4 SES CEMBERİNİN FARKLI ODAYA SES İLETME SORUNU ÇÖZÜMÜ

Bu projedeki alternatif çözüm üretilmesi gereken konu, sesin duvarın arkasında bile olsa iletilmesi olarak düşünülebilir. Şekil1 de gözüktüğü gibi ses kodlarının 4 farklı durumu oluşabiliyor ve Çarpışma fonksiyonunun çıkış fonksiyonları bu sorunu çözmek için yeterli kalmıyor.



Şekil 78 Anlatım Paneli

1. durum çizgi hiçbirine değmiyor
- 2.durum çizgi daireye değiyor gardiyana değmiyor
- 3.durum çizgi daireye değiyor gardiyana değiyor
- 4.durum çizgi daireye değmiyor garddiyana değiyor (imkansız)

```
1. durum  
ayni_odada=true / false  
  
2.durum |  
ayni_odada= true/false  
  
3.durum  
ayni_odada=true
```

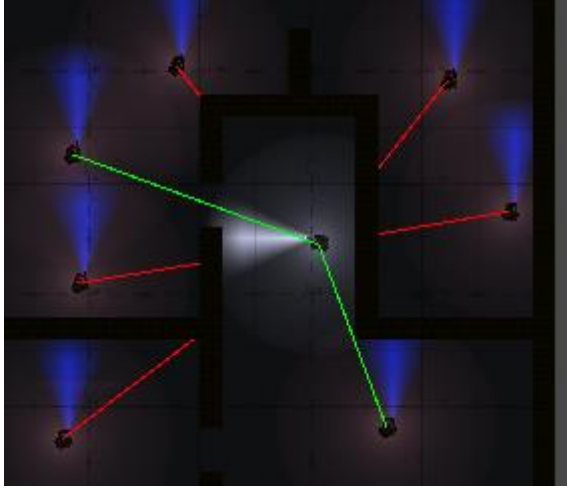
Şekil 79 Kodlama mantığı

Şekil 2 de örnek bir durum analizi işlenmiş ve Unity içerisindeki çarpışma fonksiyonunun çözüm için yeterli kalmadığı tespit edilerek farklı bir bileşen kullanılma yoluna gidilmiştir.

3.26.8 RAYCAST NEDİR

Raycast kavramın Unity yazılımında olduğu gibi birçok oyun geliştirme motorunda var olan bir kavramdır. Raycast bir objeden belirlenen yönde ve uzunlukta bir görünmeyen ışın yollayarak yollandığı konuma ulaşp ulaşamadığını ulaştığında hangi objeye çarptığını sisteme geri döndüren bir Unity taban fonksiyonudur.

3.26.8.1 RAYCAST OLUŞTURMA ve DUVAR KONTROL SCRIPTİNİ ANA KARAKTERE EKLEME



Şekil 80 Oyun içi Görüntü

Şekilde gözüktüğü gibi Raycast sistemi eklendiğinde arada duvar varken ve iki obje birbirine ulaşabiliyorken dönüş değeri doğru, ulaşamıyorken dönüş değeri yanlış oluyor ve buna göre yeşil veya kırmızı rengini alıyor.

3.26.8.2 RAYCAST SCRIPT AÇIKLAMASI

```
5 public class ray_duvarkontrol : MonoBehaviour
6 {
7     Collider2D çevre_0;
8     GameObject anakarakter_temsili, gardiyan_temsili;
9     public LayerMask lm;
10    public float isin_mesafesi;
11
12
13
14    // Start is called before the first frame update
15    @ Unity İletisi | 0 başvuru
16    void Start()
17    {
18        çevre_0 = GameObject.FindGameObjectWithTag("cevre_0").GetComponent<Collider2D>();
19        anakarakter_temsili = GameObject.FindGameObjectWithTag("anakaraktertag");
20        gardiyan_temsili = this.transform.parent.parent.gameObject;
21    }
22 }
```

Şekil 81 Kod

```

void Update()
{
    float mesafe = Vector2.Distance(gardiyan_temsili.transform.position , anakarakter_temsili.transform.position);
    Vector3 farkvektoru = anakarakter_temsili.transform.position - gardiyan_temsili.transform.position;
    Physics2D.queriesHitTriggers = false;
    Physics2D.queriesStartInColliders = false;
    RaycastHit2D hit = Physics2D.Raycast(gardiyan_temsili.transform.position, farkvektoru, mesafe, 1m);
    if (hit.collider!=null)
    {
        // Debug.DrawRay(transform.position, farkvektoru * mesafe, Color.blue);
        Gardiyan_ses_duyuma_kod_temsili.arada_duvar_var = true;
    }
    else
    {
        Gardiyan_ses_duyuma_kod_temsili.arada_duvar_var = false;
    }
}

```

Şekil 82 Kod

Şekil 1 ve 2 de gözüktüğü gibi raycast sistemi Fizik kütüphanesinden eklenerek karaktere entegre ediliyor. Burada oyun geliştiricinin dikkat etmesi gereken husus daha önce belirtildiği gibi ışın yollanılan obje ile çarpışması istenen tüm objelerin aynı layer içerisinde işlenmesi oluyor.

Dolayısıyla bu kodların aktif çalışabilmesi için ‘1m’ layerini duvarlara ve Raycast objesine verilmesi gerekiyor.

Buradan döndürülen değer ses duyma kodlarının gerekli koşul bloğuna ekleniyor. (Ekler içerisinde detaylıca gösterilmiştir.)

3.27 GARDİYAN GENEL İŞLEYİŞ MODELİ

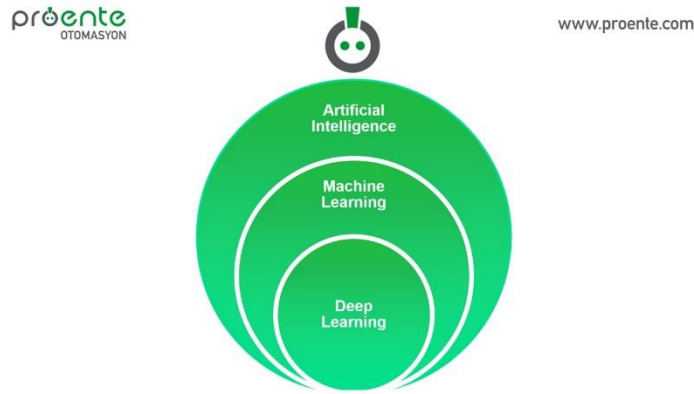
Gardiyan nesnesi, oyun yazılımında çokça bilinen NPC karakteri temsil ediyor. Burada garidyan nesnesinin yapacağı her hareket, tepki ve olaylar yapay zekâ kavramı tarafından işleniyor.

3.28 YAPAY ZEKA NEDİR?

“Yapay zekâ ve oyun etkileşiminde yapay zekanın temel amacı ve kalitesi bilgisayarın veya akıllı cihazın oyuncuya yetenekli ve akıllı bir şekilde tepki vermesi, oyun boyunca planlar yapması, hamleler geliştirmesi ve onu yenebilmesi üzerine kuruludur denilebilir. Ancak yapay zekâ reaksiyonlarının da dengede tutulması gerekiyor.

İnsandan daha hızlı tepki verebilen ve üst düzey taktikler yapabilen yapay zekâ oyunun çekiciliğini düşürebiliyor. Bu noktada oyuna seviyeler ekleniyor ve oyun oynayan kişi karşısındaki bilgisayarın zekilik derecesini kendi seçebiliyor.” [Kaynakça :<https://abainnolab.com/yapay-zeka-ve-oyunlari>]]”

Yapay zeka günümüz teknolojisinde çok farklı sektörlerde çeşitli amaçlarla kullanılabilir. Yoğun kullanım alanlarında yapay zeka makine öğrenmesi ve derin öğrenme ile birlikte işlevsellik gösterse de bunların haricinde de yapay zeka kullanım alanları bulunuyor. Yapay zekanın oyunlardaki kullanımı makine öğrenimi ve derin öğrenmeden ziyade koşul tabanlı öğretme stratejisi ile bilgisayar destekli cihazların, objelerin, karakterlerin, nesnelerin yapacağı hamleleri kimi zaman tahmin edebiliyor, kimi zaman otomatik reaksiyon veriyor.



Şekil 83 Yapay zeka inceleme Modeli

Şekil’de görüldüğü üzere yapay zekânın makine öğrenimi ve derin öğrenmeyi kapsadığı fakat onların haricinde de bir alanda varlık gösterdiği gösterilmiştir.

3.28.1 UNITY OYUN MOTORU İLE YAPAY ZEKÂ

Unity yazılımının hâlihazırda aktif bir yapay zekâ kütüphanesi bulunuyor. Oyun geliştiricisi bu yapay zeka kütüphanesini kullanmak istediğinde bu kütüphaneyi “using AI;” kodu ile script içerisine dahil edebiliyor. Bunun yanı sıra editör panelinden de Unity Asset Store içerisinde var olan yapay zeka paketlerini projeye dahil edebiliyor.

3.28.2 PROJE İÇERİSİNDE YAPAY ZEKÂ KULLANIM AMACI

Bu projede yapay zekâ iki temelde işlenmiştir.

Birinci temelde koşul tabanlı NPC öğrenim sistemi baz alınmıştır.

İkinci temelde ise yol bulma algoritmalarından A* algoritması kullanılarak gardiyan objesinin ana karakter objesine ulaşması planlanmıştır.



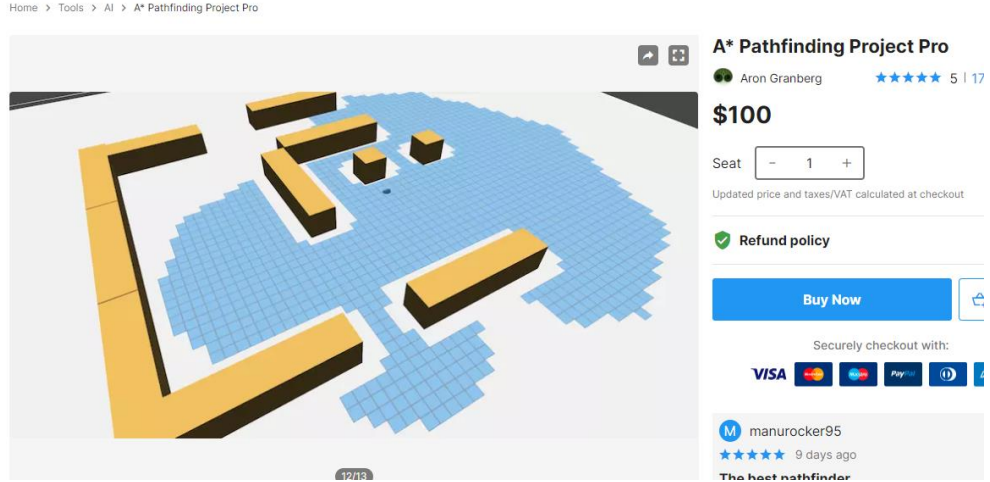
Şekil 84 Yapay zeka Şeması

Birinci temeli incelemek gerekirse;

Gardiyanın koşula bağlı yapay zekâsı şekilde gösterilmiştir.

Burada koşul büyüklüğü sırası öncelikle ses duymasına, daha sonra karakteri görmesine, daha sonra hafızasında bir ses duyup duymamasına bağlı olacak şekilde oluşturulması hedeflenmiştir.

A* algoritması güncel olarak tek pakette toplanmış olsa da hala sahibi olan “Aaron Granberg” isimli geliştirici tarafından 2013 yılında markete sürülmüştür.



Şekil 85 A* Algoritması

Şekilde asset store içerisinde yol bulma algoritması paketi gözüküyor.

3.28.3 A* ALGORTİMASI NEDİR

“

Bilgisayar bilimlerinde en kısa yol bulmak için kullanılan algoritmalarından birisidir. Örneğin seyyar tüccar problemi (travelling salesman problem, TSP) gibi bir problemin çözümünde kullanılabilir. Benzer şekilde oyun programlamada, oyunda bulunan oyuncuların en kısa yolu bularak hedefe gitmeleri için de sıklıkla kullanılan algoritmadır. Kısaca bir düğümden (node) hedef bir düğüme (target node) en kısa hangi düğümler üzerinden gidileceğini bulmaya yarayan “en iyi yerleştirme (best fit)” algoritmasıdır.”

(Prof.Dr. Şadi Evren Şeker, <http://bilgisayarkavramlari.com/2009/03/02/a-yildiz-arama-algoritmasi-a-star-search-algorithm-a/>)



Şekil 86 A* Algoritması Unity

Unity içerisinde ise basitçe düzlemi taban olarak engelleri geçilemez yüzey olarak kabul edip iki konum arasını birim karelere ayırarak bu karelerdeki iki farklı konum arasındaki en kısa yol a* algoritması ile hesaplanıyor.

3.29 GARDİYAN OBJESİNE HAREKET KAZANDIRILMASI

Proje içerisinde gardiyan nesnesi sakın olduğu süreçte rastgele hareket edecek ve sakın durumunun dışındaki süreçte belirlenen rota ile takip algoritmasını işleyecektir.

3.29.1 RASTGELE HAREKET ETME KODLARI

```
44 |  
45 |  
46 |  
47 |  
48 |  
49 |  
50 |  
51 |  
52 |  
53 |  
54 |  
55 |  
56 |  
57 |  
58 |  
59 |  
60 |  
61 |  
62 |  
63 |  
64 |  
65 |  
66 |  
67 |  
68 |  
69 |  
70 |  
71 |  
72 |  
73 |  
74 |  
75 |  
76 |  
77 |  
78 |  
79 |  
80 |  
81 |  
82 |  
83 |  
84 |  
85 |  
86 |  
87 |  
88 |  
89 |  
90 |  
91 |  
92 |  
93 |  
94 |  
95 |  
96 |  
97 |  
98 |  
99 |  
100 |  
101 |  
102 |  
103 |  
104 |  
105 |  
106 |  
107 |  
108 |  
109 |  
110 |  
111 |  
112 |  
113 |  
114 |  
115 |  
116 |  
117 |  
118 |  
119 |  
120 |  
121 |  
122 |  
123 |  
124 |  
125 |  
126 |  
127 |  
128 |  
129 |  
130 |  
131 |  
132 |  
133 |  
134 |  
135 |  
136 |  
137 |  
138 |  
139 |  
140 |  
141 |  
142 |  
143 |  
144 |  
145 |  
146 |  
147 |  
148 |  
149 |  
150 |  
151 |  
152 |  
153 |  
154 |  
155 |  
156 |  
157 |  
158 |  
159 |  
160 |  
161 |  
162 |  
163 |  
164 |  
165 |  
166 |  
167 |  
168 |  
169 |  
170 |  
171 |  
172 |  
173 |  
174 |  
175 |  
176 |  
177 |  
178 |  
179 |  
180 |  
181 |  
182 |  
183 |  
184 |  
185 |  
186 |  
187 |  
188 |  
189 |  
190 |  
191 |  
192 |  
193 |  
194 |  
195 |  
196 |  
197 |  
198 |  
199 |  
200 |  
201 |  
202 |  
203 |  
204 |  
205 |  
206 |  
207 |  
208 |  
209 |  
210 |  
211 |  
212 |  
213 |  
214 |  
215 |  
216 |  
217 |  
218 |  
219 |  
220 |  
221 |  
222 |  
223 |  
224 |  
225 |  
226 |  
227 |  
228 |  
229 |  
230 |  
231 |  
232 |  
233 |  
234 |  
235 |  
236 |  
237 |  
238 |  
239 |  
240 |  
241 |  
242 |  
243 |  
244 |  
245 |  
246 |  
247 |  
248 |  
249 |  
250 |  
251 |  
252 |  
253 |  
254 |  
255 |  
256 |  
257 |  
258 |  
259 |  
260 |  
261 |  
262 |  
263 |  
264 |  
265 |  
266 |  
267 |  
268 |  
269 |  
270 |  
271 |  
272 |  
273 |  
274 |  
275 |  
276 |  
277 |  
278 |  
279 |  
280 |  
281 |  
282 |  
283 |  
284 |  
285 |  
286 |  
287 |  
288 |  
289 |  
290 |  
291 |  
292 |  
293 |  
294 |  
295 |  
296 |  
297 |  
298 |  
299 |  
300 |  
301 |  
302 |  
303 |  
304 |  
305 |  
306 |  
307 |  
308 |  
309 |  
310 |  
311 |  
312 |  
313 |  
314 |  
315 |  
316 |  
317 |  
318 |  
319 |  
320 |  
321 |  
322 |  
323 |  
324 |  
325 |  
326 |  
327 |  
328 |  
329 |  
330 |  
331 |  
332 |  
333 |  
334 |  
335 |  
336 |  
337 |  
338 |  
339 |  
340 |  
341 |  
342 |  
343 |  
344 |  
345 |  
346 |  
347 |  
348 |  
349 |  
350 |  
351 |  
352 |  
353 |  
354 |  
355 |  
356 |  
357 |  
358 |  
359 |  
360 |  
361 |  
362 |  
363 |  
364 |  
365 |  
366 |  
367 |  
368 |  
369 |  
370 |  
371 |  
372 |  
373 |  
374 |  
375 |  
376 |  
377 |  
378 |  
379 |  
380 |  
381 |  
382 |  
383 |  
384 |  
385 |  
386 |  
387 |  
388 |  
389 |  
390 |  
391 |  
392 |  
393 |  
394 |  
395 |  
396 |  
397 |  
398 |  
399 |  
400 |  
401 |  
402 |  
403 |  
404 |  
405 |  
406 |  
407 |  
408 |  
409 |  
410 |  
411 |  
412 |  
413 |  
414 |  
415 |  
416 |  
417 |  
418 |  
419 |  
420 |  
421 |  
422 |  
423 |  
424 |  
425 |  
426 |  
427 |  
428 |  
429 |  
430 |  
431 |  
432 |  
433 |  
434 |  
435 |  
436 |  
437 |  
438 |  
439 |  
440 |  
441 |  
442 |  
443 |  
444 |  
445 |  
446 |  
447 |  
448 |  
449 |  
450 |  
451 |  
452 |  
453 |  
454 |  
455 |  
456 |  
457 |  
458 |  
459 |  
460 |  
461 |  
462 |  
463 |  
464 |  
465 |  
466 |  
467 |  
468 |  
469 |  
470 |  
471 |  
472 |  
473 |  
474 |  
475 |  
476 |  
477 |  
478 |  
479 |  
480 |  
481 |  
482 |  
483 |  
484 |  
485 |  
486 |  
487 |  
488 |  
489 |  
490 |  
491 |  
492 |  
493 |  
494 |  
495 |  
496 |  
497 |  
498 |  
499 |  
500 |  
501 |  
502 |  
503 |  
504 |  
505 |  
506 |  
507 |  
508 |  
509 |  
510 |  
511 |  
512 |  
513 |  
514 |  
515 |  
516 |  
517 |  
518 |  
519 |  
520 |  
521 |  
522 |  
523 |  
524 |  
525 |  
526 |  
527 |  
528 |  
529 |  
530 |  
531 |  
532 |  
533 |  
534 |  
535 |  
536 |  
537 |  
538 |  
539 |  
540 |  
541 |  
542 |  
543 |  
544 |  
545 |  
546 |  
547 |  
548 |  
549 |  
550 |  
551 |  
552 |  
553 |  
554 |  
555 |  
556 |  
557 |  
558 |  
559 |  
560 |  
561 |  
562 |  
563 |  
564 |  
565 |  
566 |  
567 |  
568 |  
569 |  
570 |  
571 |  
572 |  
573 |  
574 |  
575 |  
576 |  
577 |  
578 |  
579 |  
580 |  
581 |  
582 |  
583 |  
584 |  
585 |  
586 |  
587 |  
588 |  
589 |  
590 |  
591 |  
592 |  
593 |  
594 |  
595 |  
596 |  
597 |  
598 |  
599 |  
600 |  
601 |  
602 |  
603 |  
604 |  
605 |  
606 |  
607 |  
608 |  
609 |  
610 |  
611 |  
612 |  
613 |  
614 |  
615 |  
616 |  
617 |  
618 |  
619 |  
620 |  
621 |  
622 |  
623 |  
624 |  
625 |  
626 |  
627 |  
628 |  
629 |  
630 |  
631 |  
632 |  
633 |  
634 |  
635 |  
636 |  
637 |  
638 |  
639 |  
640 |  
641 |  
642 |  
643 |  
644 |  
645 |  
646 |  
647 |  
648 |  
649 |  
650 |  
651 |  
652 |  
653 |  
654 |  
655 |  
656 |  
657 |  
658 |  
659 |  
660 |  
661 |  
662 |  
663 |  
664 |  
665 |  
666 |  
667 |  
668 |  
669 |  
670 |  
671 |  
672 |  
673 |  
674 |  
675 |  
676 |  
677 |  
678 |  
679 |  
680 |  
681 |  
682 |  
683 |  
684 |  
685 |  
686 |  
687 |  
688 |  
689 |  
690 |  
691 |  
692 |  
693 |  
694 |  
695 |  
696 |  
697 |  
698 |  
699 |  
700 |  
701 |  
702 |  
703 |  
704 |  
705 |  
706 |  
707 |  
708 |  
709 |  
710 |  
711 |  
712 |  
713 |  
714 |  
715 |  
716 |  
717 |  
718 |  
719 |  
720 |  
721 |  
722 |  
723 |  
724 |  
725 |  
726 |  
727 |  
728 |  
729 |  
730 |  
731 |  
732 |  
733 |  
734 |  
735 |  
736 |  
737 |  
738 |  
739 |  
740 |  
741 |  
742 |  
743 |  
744 |  
745 |  
746 |  
747 |  
748 |  
749 |  
750 |  
751 |  
752 |  
753 |  
754 |  
755 |  
756 |  
757 |  
758 |  
759 |  
760 |  
761 |  
762 |  
763 |  
764 |  
765 |  
766 |  
767 |  
768 |  
769 |  
770 |  
771 |  
772 |  
773 |  
774 |  
775 |  
776 |  
777 |  
778 |  
779 |  
780 |  
781 |  
782 |  
783 |  
784 |  
785 |  
786 |  
787 |  
788 |  
789 |  
790 |  
791 |  
792 |  
793 |  
794 |  
795 |  
796 |  
797 |  
798 |  
799 |  
800 |  
801 |  
802 |  
803 |  
804 |  
805 |  
806 |  
807 |  
808 |  
809 |  
810 |  
811 |  
812 |  
813 |  
814 |  
815 |  
816 |  
817 |  
818 |  
819 |  
820 |  
821 |  
822 |  
823 |  
824 |  
825 |  
826 |  
827 |  
828 |  
829 |  
830 |  
831 |  
832 |  
833 |  
834 |  
835 |  
836 |  
837 |  
838 |  
839 |  
840 |  
841 |  
842 |  
843 |  
844 |  
845 |  
846 |  
847 |  
848 |  
849 |  
850 |  
851 |  
852 |  
853 |  
854 |  
855 |  
856 |  
857 |  
858 |  
859 |  
860 |  
861 |  
862 |  
863 |  
864 |  
865 |  
866 |  
867 |  
868 |  
869 |  
870 |  
871 |  
872 |  
873 |  
874 |  
875 |  
876 |  
877 |  
878 |  
879 |  
880 |  
881 |  
882 |  
883 |  
884 |  
885 |  
886 |  
887 |  
888 |  
889 |  
890 |  
891 |  
892 |  
893 |  
894 |  
895 |  
896 |  
897 |  
898 |  
899 |  
900 |  
901 |  
902 |  
903 |  
904 |  
905 |  
906 |  
907 |  
908 |  
909 |  
910 |  
911 |  
912 |  
913 |  
914 |  
915 |  
916 |  
917 |  
918 |  
919 |  
920 |  
921 |  
922 |  
923 |  
924 |  
925 |  
926 |  
927 |  
928 |  
929 |  
930 |  
931 |  
932 |  
933 |  
934 |  
935 |  
936 |  
937 |  
938 |  
939 |  
940 |  
941 |  
942 |  
943 |  
944 |  
945 |  
946 |  
947 |  
948 |  
949 |  
950 |  
951 |  
952 |  
953 |  
954 |  
955 |  
956 |  
957 |  
958 |  
959 |  
960 |  
961 |  
962 |  
963 |  
964 |  
965 |  
966 |  
967 |  
968 |  
969 |  
970 |  
971 |  
972 |  
973 |  
974 |  
975 |  
976 |  
977 |  
978 |  
979 |  
980 |  
981 |  
982 |  
983 |  
984 |  
985 |  
986 |  
987 |  
988 |  
989 |  
990 |  
991 |  
992 |  
993 |  
994 |  
995 |  
996 |  
997 |  
998 |  
999 |  
1000 |  
1001 |  
1002 |  
1003 |  
1004 |  
1005 |  
1006 |  
1007 |  
1008 |  
1009 |  
1010 |  
1011 |  
1012 |  
1013 |  
1014 |  
1015 |  
1016 |  
1017 |  
1018 |  
1019 |  
1020 |  
1021 |  
1022 |  
1023 |  
1024 |  
1025 |  
1026 |  
1027 |  
1028 |  
1029 |  
1030 |  
1031 |  
1032 |  
1033 |  
1034 |  
1035 |  
1036 |  
1037 |  
1038 |  
1039 |  
1040 |  
1041 |  
1042 |  
1043 |  
1044 |  
1045 |  
1046 |  
1047 |  
1048 |  
1049 |  
1050 |  
1051 |  
1052 |  
1053 |  
1054 |  
1055 |  
1056 |  
1057 |  
1058 |  
1059 |  
1060 |  
1061 |  
1062 |  
1063 |  
1064 |  
1065 |  
1066 |  
1067 |  
1068 |  
1069 |  
1070 |  
1071 |  
1072 |  
1073 |  
1074 |  
1075 |  
1076 |  
1077 |  
1078 |  
1079 |  
1080 |  
1081 |  
1082 |  
1083 |  
1084 |  
1085 |  
1086 |  
1087 |  
1088 |  
1089 |  
1090 |  
1091 |  
1092 |  
1093 |  
1094 |  
1095 |  
1096 |  
1097 |  
1098 |  
1099 |  
1100 |  
1101 |  
1102 |  
1103 |  
1104 |  
1105 |  
1106 |  
1107 |  
1108 |  
1109 |  
1110 |  
1111 |  
1112 |  
1113 |  
1114 |  
1115 |  
1116 |  
1117 |  
1118 |  
1119 |  
1120 |  
1121 |  
1122 |  
1123 |  
1124 |  
1125 |  
1126 |  
1127 |  
1128 |  
1129 |  
1130 |  
1131 |  
1132 |  
1133 |  
1134 |  
1135 |  
1136 |  
1137 |  
1138 |  
1139 |  
1140 |  
1141 |  
1142 |  
1143 |  
1144 |  
1145 |  
1146 |  
1147 |  
1148 |  
1149 |  
1150 |  
1151 |  
1152 |  
1153 |  
1154 |  
1155 |  
1156 |  
1157 |  
1158 |  
1159 |  
1160 |  
1161 |  
1162 |  
1163 |  
1164 |  
1165 |  
1166 |  
1167 |  
1168 |  
1169 |  
1170 |  
1171 |  
1172 |  
1173 |  
1174 |  
1175 |  
1176 |  
1177 |  
1178 |  
1179 |  
1180 |  
1181 |  
1182 |  
1183 |  
1184 |  
1185 |  
1186 |  
1187 |  
1188 |  
1189 |  
1190 |  
1191 |  
1192 |  
1193 |  
1194 |  
1195 |  
1196 |  
1197 |  
1198 |  
1199 |  
1200 |  
1201 |  
1202 |  
1203 |  
1204 |  
1205 |  
1206 |  
1207 |  
1208 |  
1209 |  
1210 |  
1211 |  
1212 |  
1213 |  
1214 |  
1215 |  
1216 |  
1217 |  
1218 |  
1219 |  
1220 |  
1221 |  
1222 |  
1223 |  
1224 |  
1225 |  
1226 |  
1227 |  
1228 |  
1229 |  
1230 |  
1231 |  
1232 |  
1233 |  
1234 |  
1235 |  
1236 |  
1237 |  
1238 |  
1239 |  
1240 |  
1241 |  
1242 |  
1243 |  
1244 |  
1245 |  
1246 |  
1247 |  
1248 |  
1249 |  
1250 |  
1251 |  
1252 |  
1253 |  
1254 |  
1255 |  
1256 |  
1257 |  
1258 |  
1259 |  
1260 |  
1261 |  
1262 |  
1263 |  
1264 |  
1265 |  
1266 |  
1267 |  
1268 |  
1269 |  
1270 |  
1271 |  
1272 |  
1273 |  
1274 |  
1275 |  
1276 |  
1277 |  
1278 |  
1279 |  
1280 |  
1281 |  
1282 |  
1283 |  
1284 |  
1285 |  
1286 |  
1287 |  
1288 |  
1289 |  
1290 |  
1291 |  
1292 |  
1293 |  
1294 |  
1295 |  
1296 |  
1297 |  
1298 |  
1299 |  
1300 |  
1301 |  
1302 |  
1303 |  
1304 |  
1305 |  
1306 |  
1307 |  
1308 |  
1309 |  
1310 |  
1311 |  
1312 |  
1313 |  
1314 |  
1315 |  
1316 |  
1317 |  
1318 |  
1319 |  
1320 |  
1321 |  
1322 |  
1323 |  
1324 |  
1325 |  
1326 |  
1327 |  
1328 |  
1329 |  
1330 |  
1331 |  
1332 |  
1333 |  
1334 |  
1335 |  
1336 |  
1337 |  
1338 |  
1339 |  
1340 |  
1341 |  
1342 |  
1343 |  
1344 |  
1345 |  
1346 |  
1347 |  
1348 |  
1349 |  
1350 |  
1351 |  
1352 |  
1353 |  
1354 |  
1355 |  
1356 |  
1357 |  
1358 |  
1359 |  
1360 |  
1361 |  
1362 |  
1363 |  
1364 |  
1365 |  
1366 |  
1367 |  
1368 |  
1369 |  
1370 |  
1371 |  
1372 |  
1373 |  
1374 |  
1375 |  
1376 |  
1377 |  
1378 |  
1379 |  
1380 |  
1381 |  
1382 |  
1383 |  
1384 |  
1385 |  
1386 |  
1387 |  
1388 |  
1389 |  
1390 |  
1391 |  
1392 |  
1393 |  
1394 |  
1395 |  
1396 |  
1397 |  
1398 |  
1399 |  
1400 |  
1401 |  
1402 |  
1403 |  
1404 |  
1405 |  
1406 |  
1407 |  
1408 |  
1409 |  
1410 |  
1411 |  
1412 |  
1413 |  
1414 |  
1415 |  
1416 |  
1417 |  
1418 |  
1419 |  
1420 |  
1421 |  
1422 |  
1423 |  
1424 |  
1425 |  
1426 |  
1427 |  
1428 |  
1429 |  
1430 |  
1431 |  
1432 |  
1433 |  
1434 |  
1435 |  
1436 |  
1437 |  
1438 |  
1439 |  
1440 |  
1441 |  
1442 |  
1443 |  
1444 |  
1445 |  
1446 |  
1447 |  
1448 |  
1449 |  
1450 |  
1451 |  
1452 |  
1453 |  
1454 |  
1455 |  
1456 |  
1457 |  
1458 |  
1459 |  
1460 |  
1461 |  
1462 |  
1463 |  
1464 |  
1465 |  
1466 |  
1467 |  
1468 |  
1469 |  
1470 |  
1471 |  
1472 |  
1473 |  
1474 |  
1475 |  
1476 |  
1477 |  
1478 |  
1479 |  
1480 |  
1481 |  
1482 |  
1483 |  
1484 |  
1485 |  
1486 |  
1487 |  
1488 |  
1489 |  
1490 |  
1491 |  
1492 |  
1493 |  
1494 |  
1495 |  
1496 |  
1497 |  
1498 |  
1499 |  
1500 |  
1501 |  
1502 |  
1503 |  
1504 |  
1505 |  
1506 |  
1507 |  
1508 |  
1509 |  
1510 |  
1511 |  
1512 |  
1513 |  
1514 |  
1515 |  
1516 |  
1517 |  
1518 |  
1519 |  
1520 |  
1521 |  
1522 |  
1523 |  
1524 |  
1525 |  
1526 |  
1527 |  
1528 |  
1529 |  
1530 |  
1531 |  
1532 |  
1533 |  
1534 |  
1535 |  
1536 |  
1537 |  
1538 |  
1539 |  
1540 |  
1541 |  
1542 |  
1543 |  
1544 |  
1545 |  
1546 |  
1547 |  
1548 |  
1549 |  
1550 |  
1551 |  
1552 |  
1553 |  
1554 |  
1555 |  
1556 |  
1557 |  
1558 |  
1559 |  
1560 |  
1561 |  
1562 |  
1563 |  
1564 |  
1565 |  
1566 |  
1567 |  
1568 |  
1569 |  
1570 |  
1571 |  
1572 |  
1573 |  
1574 |  
1575 |  
1576 |  
1577 |  
1578 |  
1579 |  
1580 |  
1581 |  
1582 |  
1583 |  
1584 |  
1585 |  
1586 |  
1587 |  
1588 |  
1589 |  
1590 |  
1591 |  
1592 |  
1593 |  
1594 |  
1595 |  
1596 |  
1597 |  
1598 |  
1599 |  
1600 |  
1601 |  
1602 |  
1603 |  
1604 |  
1605 |  
1606 |  
1607 |  
1608 |  
1609 |  
1610 |  
1611 |  
1612 |  
1613 |  
1614 |  
1615 |  
1616 |  
1617 |  
1618 |  
1619 |  
1620 |  
1621 |  
1622 |  
1623 |  
1624 |  
1625 |  
1626 |  
1627 |  
1628 |  
1629 |  
1630 |  
1631 |  
1632 |  
1633 |  
1634 |  
1635 |  
1636 |  
1637 |  
1638 |  
1639 |  
1640 |  
1641 |  
1642 |  
1643 |  
1644 |  
1645 |  
1646 |  
1647 |  
1648 |  
1649 |  
1650 |  
1651 |  
1652 |  
1653 |  
1654 |  
1655 |  
1656 |  
1657 |  
1658 |  
1659 |  
1660 |  
1661 |  
1662 |  

```

```

- references
public Vector2 random_yon(Vector2 aktif_yon)
{
    Vector2 randomyon;
    int sayi = Random.Range(1, 9);
    switch (sayi)
    {
        case 1: //bati
            randomyon.x = -1;
            randomyon.y = 0;
            break;
        case 2: //kuzeybati
            randomyon.x = -1;
            randomyon.y = 1;
            break;
        case 3: //kuzey
            randomyon.x = 0;
            randomyon.y = 1;
            break;
        case 4: // kuzeydogu
            randomyon.x = 1;
            randomyon.y = 1;
            break;
        case 5: //dogu
            randomyon.x = 1;
            randomyon.y = 0;
            break;
    }
}

```

Şekil 88 Kod

```

        break;
        case 5: //dogu
            randomyon.x = 1;
            randomyon.y = 0;
            break;
        case 6: //guneydogu
            randomyon.x = 1;
            randomyon.y = -1;
            break;
        case 7: //guney
            randomyon.x = 0;
            randomyon.y = -1;
            break;
        case 8: //guneybati
            randomyon.x = -1;
            randomyon.y = -1;
            break;
        default:
            randomyon = new Vector2(0, 0);
            break;
    }
    if (aktif_yon != randomyon)
        return randomyon;
    else
        return -randomyon;
}

```

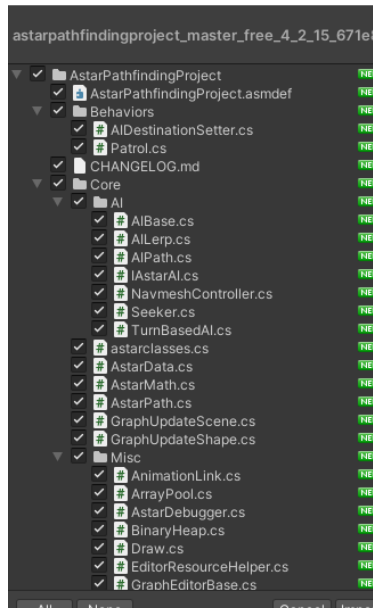
Şekil 89 Kod

Bu kodlar ile gardiyanın sakin durumda rastgele hareket etmesi amaçlanmıştır.

3.29.2 TAKİP ALGORİTMASI KODLARI VE A* PAKETİ KURULUMU

Takip algoritma kodları ekleme sırası proje oluşturulma sürecinde raporlar kaynak alınarak bunları maddeleyerek yapılan işlemlerin sıralanması yöntemiyle bu kısımda açıklanmıştır.

Takip algoritması için öncelikle projeye a* yol bulma paketi eklendi.



Şekil 90 A* paket içeriği

Şekil: eklenecek dosyaların listesi ve paket adı

A* yüzeyi obje olarak projeye eklendi.

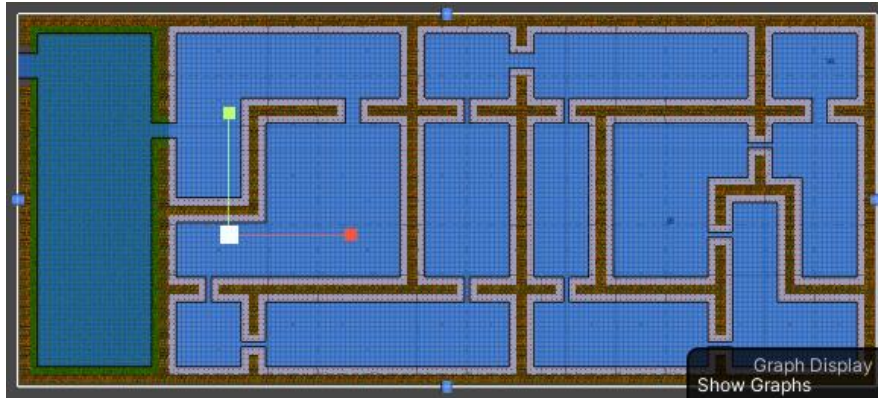
AI kütüphanesinden Pathfinder, A* yüzeyi objesine eklendi.

Katman grafiği Pathfinder bileşeni ile editör panelinden projeye eklendi.

Çeşitli kare-büyükölç ve haritada nesne aralıkları ayarları yapıldı.

Seeker bileşeni gardiyana eklendi.

Takip algoritması kodları gardiyana eklendi.



Şekil 91 A* Yüzeyi Oyun içi Uygulanmış Görüntü

```

if (path == null)
{
    return;
}
if (CurrentWaypoint >= path.vectorPath.Count)
{
    ReachedEndOfPath = true;
    return;
}
else
{
    ReachedEndOfPath = false;
}
// ust taraf astar ile ilgili

if (durumkontrollet_kodu_temsili.hedef_anakarakter)
{
    anakarakteriyakala();
}
else if (durumkontrollet_kodu_temsili.hedef_sonduyulanses)
{
    sesi_arastir();
}
else if (durumkontrollet_kodu_temsili.hedef_rastgele)
{
    randomhareketet();
}

```

Şekil 92 Kod

Kullanılan fonksiyonlar eklerde detaylıca eklenmiştir.

Bu aşamada harita birim karelerde yüzey olarak ayrılmış ve duvarlar farklı bir layer seviyesinde algoritmaya engel olarak tanıtılmıştır.



Şekil 93 A* Sonuç Görüntüsü

3.30 PROJEDEKİ OBJELERE ANİMASYON EKLEME

Unity yazılımında animasyon eklenmesi için geliştiricinin kullanabileceği çift yönlü bir animasyon editörü vardır. Bu yönlerden animatör isimli olan panelde animasyonların birbiri ile olan veri ilişkisi işlenirken, animasyon panelinde grafik olarak fotoğrafların art arda dizilim işlemleri kontrol ediliyor.

3.30.1 ANİMASYON PANELİ KULLANIMI



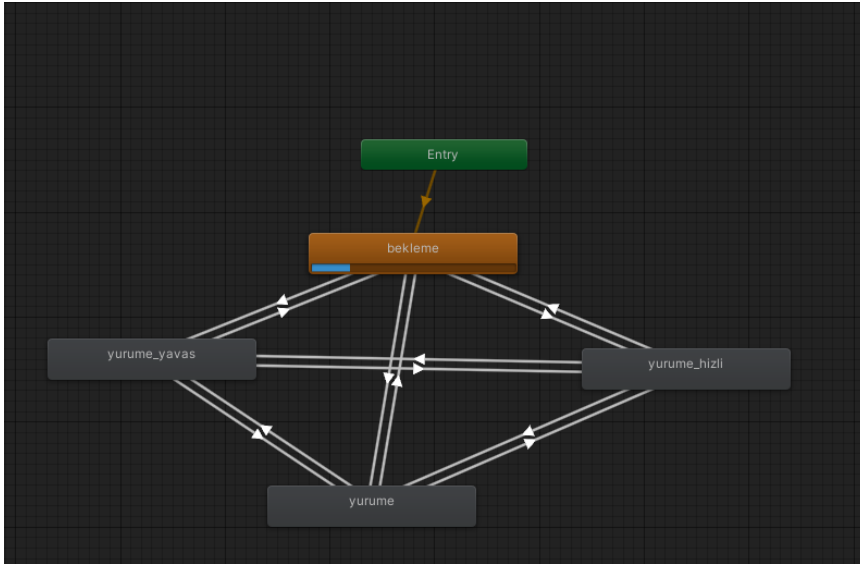
Şekil 94 Animasyon paneli

Bu panelde paketlerde hazır olarak gelen veya el ile oluşturulan görseller art arda eklenerek görsel animasyon kısmı oluşturulup bir obje olarak oyun dosyalarına

ekleniyor. Çeşitli görsel ayarlamalar buradan yapılıyor ve oyun geliştirme ekiplerinin büyük çoğunluğunda bu paneller, grafik tasarımcılarına ayrılarak onların ürettiği dosyalar üzerinde işlemler yapılıyor.

3.30.2 ANİMATÖR PANELİ KULLANIMI

Animatör panelinde ise bir sınıf nesne yapısı ile animasyonların birbirini takip etme ve hangi durumda hangi animasyonların gösterileceği oyun geliştirici tarafından belirleniyor.



Şekil 95 Animator Paneli

Şekilde animatör panelinin örnek kullanımı gösterilmiştir.

Animatör panelinde gözüken her bir blok farklı bir animasyonu temsil eder. Bu animasyonların birbirlerine geçişlerindeki koşullar scriptler ile sağlanır. Şekilde gözüken canlandırıcı panelinde objenin herhangi bir durumda iken bekleme animasyonunu sürekli çalıştırması ve gerekli koşullar gerçekleştiğinde diğer animasyonlar arasında geçiş yapabileceği tasarlanmıştır.

3.30.3 ANİMASYON SCRIPTİ EKLENMESİ

```
// Update is called once per frame
@ Unity İletisi | 0 basyuru
void Update()
{
    if (yurumekodunesnesi.speed == ilkhiz)
    {
        if (yurumekodunesnesi.haraketdiyormuyum == false)
            animator.SetInteger("yurume_durumu", 0); //duruyor
        else
            animator.SetInteger("yurume_durumu", 2); //yuruyor
    }
    else if (yurumekodunesnesi.speed == (ilkhiz + hizlandirmakatsayi))
    {
        animator.SetInteger("yurume_durumu", 3); //kosuyor
    }
    else if (yurumekodunesnesi.speed == (ilkhiz - yavaslatmakatsayi))
    {
        animator.SetInteger("yurume_durumu", 1); //yavas yuruyor
    }
}
```

Şekil 96 Kod

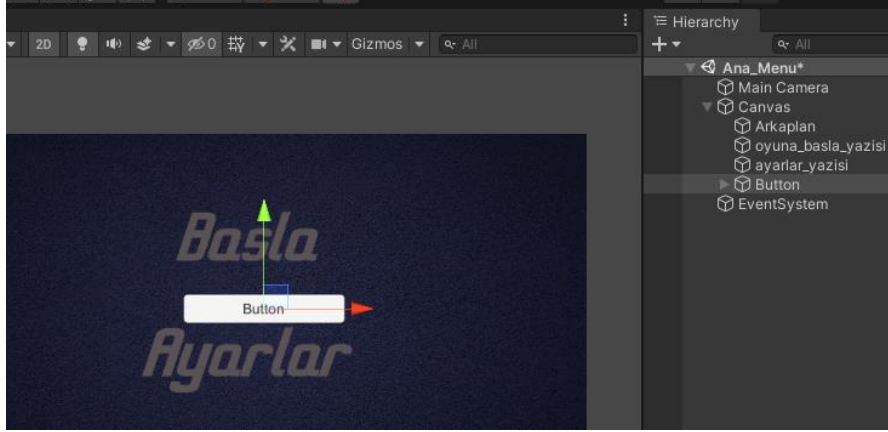
Animatör panelinde oluşturulan her bir durum değişkeni, kodlar ile ilgili objeye atanarak, hangi koşula uyduğu sürekli bir döngüde belirlenerek değiştirilir. Değiştirilen durum değişkeni canlandırıcı tarafından sürekli bir döngüde okunur ve durum değişkeni değiştiğinde canlandırıcı panelindeki okların izin verdiği animasyonlara geçişler sağlanır.

3.31 PROJEYE ANA MENU EKLEME

Her yazılımın bir karşılama ekranı vardır. Web sitelerinin giriş formları veya kullanılan Unity yazılımının ek paketi olan HUB yazılımı bir proje karşılama ekranı olarak örnek verilebilir.

Oyun geliştirme sürecinde, oyun geliştiricinin oluşturduğu proje kadar oyuncuya optimize edilmiş bir proje sunumunu yapması da önemlidir. Bu sebeple oyunların bir başlangıç ve bitiş senaryosu vardır.

Unity yazılımında karşılama ekranı oluşturmak için veya kullanıcı arayüzü ile iletişim yapabilmek için Canvas objesi kullanılır. Canvas nesnesi üzerinde butonlar veya benzeri form elemanları kullanılarak menü tasarımı yapılır.



Şekil 97 Editör Paneli

3.31.1 ANA MENU KODLAMASI

```
}  
    }  
    if(Input.GetKeyDown(KeyCode.Escape))  
    {  
        SceneManager.LoadScene(0);  
    }  
}
```

Şekil 98 Kod

Ana menü kodlaması sahne yönetimi isimli kütüphanenin sahne yükleme komutu ile kodlara eklenir ve ilgili objeye atanır.

4.SONUÇ VE ÖNERİLER

Türkiye’de yazılım sektörü dünya geneline oranla oldukça geri seviyede kalmış durumda. Bunun doğurduğu sebeplerle birlikte Türk oyun yazılım sektöründe kullanılabilecek literatür ve proje sayısı, oyun yazılım sektörünü geliştirmek için yetersiz kalıyor. Bu projenin çalışma sonucunda bir oyun geliştiricisinin bir oyun tasarımıyla sıfırdan son aşamaya kadar gelinebilecek seviyeler hakkında bilgi sahibi olunmuştur. Proje temel alınarak çeşitli oyunlar ve yazılımlar geliştirme imkânı bulunmaktadır. İki boyutlu oyun yazılım sektöründe öncü bir proje temeli oluşturulmuş ve geliştirmeye açıktır. Yapay zekânın oyun sektöründeki yeri ve kullanım amacı detaylıca açıklanmış olup, yazılan oyun projelerinin gerçeğe olabildiğince yakın olması için gerekli düşünce temelleri oluşturulmuştur.

Projenin geliştirilmesi kapsamında çeşitli grafikler eklenerek, harita açık dünya haritasına çevrilerek , bir sunucuya bağlanarak birden fazla kişinin farklı ortamlarda olsa bile aynı oyunda buluşması ve oynaması mümkün hale getirilebilir.

Bir başka bakış açısıyla projenin açık dünya haritasında hikaye tabanlı bir oyun yazılımına dönüştürülerek de senaryo diyalogları ile geniş kapsamlı bir oyun projesi olarak sürülmesi mümkündür.

5.KAYNAKÇA

URL-1. Android hakkında genel bilgi, https://www.android.com/intl/tr_tr/Erişim Tarihi: 2021

URL-2. Android hakkında genel bilgi, [https://tr.wikipedia.org/wiki/Android_\(i%C5%9Fletim_sistemi\)](https://tr.wikipedia.org/wiki/Android_(i%C5%9Fletim_sistemi))/Erişim Tarihi: 2021

URL-3. Unity indirilmesi, <https://unity.com/>Erişim Tarihi: 2021

URL-4. Android studio uygulaması indirildi, <https://developer.android.com/studio/>Erişim Tarihi: 2021

URL-5. Unity hakkında genel bilgi, <https://www.javatpoint.com/unity-ui/>Erişim Tarihi: 2021

URL-6. Unity Programlama hakkında genel bilgi, <https://www.cs.cornell.edu/courses/cs3152/2013sp/lectures/15-Perspective.pdf>/Erişim Tarihi: 2021

Thesis CENTRIA UNIVERSITY OF APPLIED SCIENCES Degree Programme in Information Technology December 2017

ÖZGEÇMİŞ

Adı Soyadı : METEHAN BULUT
Doğum Yeri ve Yılı : ESKİŞEHİR / 1998
Medeni Hali : BEKAR
Yabancı Dili : İNGİLİZCE



Eğitim Durumu

Lise : Ufuk Arslan Anadolu Lisesi
Lisans : Kastamonu Üniversitesi Bilgisayar Mühendisliği / 2021

1998 yılında Eskişehir’de doğdu; ilk ve orta öğrenimini Ankara’da tamamladı; Ufuk Arslan Anadolu Lisesi’nden mezun olduktan sonra 2017 yılında Kastamonu Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü’ne girdi.

ADRES BİLGİLERİ

Adres: 3.Etap Eryaman
13. Caddesi, No. 33, D: 5
Etimesgut / Ankara

Tel: (538) 626 0658
E-posta: metehan.bulut@hotmail.com