



T.C.
BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

UNİTY3D'DE OPENCV İLE OYUN GELİŞTİRME

HÜSEYİN MERT YAVAŞ
BİTİRME ÇALIŞMASI

DANIŞMANI : Prof. Dr. CİHAN KARAKUZU

**BİLECİK
9 Eylül 2021**



T.C.
BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

UNİTY3D'DE OPENCV İLE OYUN GELİŞTİRME
HÜSEYİN MERT YAVAŞ
BİTİRME ÇALIŞMASI

DANIŞMANI : Prof. Dr. CİHAN KARAKUZU

BİLECİK
9 Eylül 2021

BİLDİRİM

Bu kitaptaki bütün bilgilerin etik davranışı ve akademik kurallar çerçevesinde elde edildiğini ve yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

İmza

HÜSEYİN MERT YAVAŞ

Tarih: 9 Eylül 2021

ÖZET

BİTİRME ÇALIŞMASI

UNITY3D'DE OPENCV İLE OYUN GELİŞTİRME

HÜSEYİN MERT YAVAŞ

Bilecik Şeyh Edebali Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Danışman: Prof. Dr. CİHAN KARAKUZU

2021, 49 Sayfa

Jüri Üyeleri

.....
.....
.....

İmza

Bu bitirme çalışmasında OpenCV kütüphanesi ve Unity 3D oyun motoru kullanılarak oyun geliştirme hakkında incelenmiştir. Unity 3D oyun motoru kullanılarak OpenCV kütüphanesi sayesinde bir oyun geliştirilmiştir.

Anahtar Kelimeler: Unity3D, Oyun Geliştirme, OpenCV, Oyun Programlama

ABSTRACT

THESIS

GAME DEVELOPMENT WITH OPENCV IN UNITY3D

HÜSEYİN MERT YAVAŞ

**Bilecik Şeyh Edebali University
Engineering Faculty
Department of Computer Engineering**

Advisor: Prof. Dr. CİHAN KARAKUZU

2021, 49 Pages

Jury

.....
.....
.....

Sign

.....
.....
.....

In this thesis, OpenCV library and Unity 3D game engine development has been examined. A game has been developed with OpenCV library using Unity 3D game engine.

Keywords: Game Programming, OpenCV, Unity3D, Game Development

ÖNSÖZ

Bitirme çalışmamın başından sonuna kadar emeği geçen ve beni bu konuya yönlendiren saygı değer hocam ve danışmanım Sayın CİHAN KARAKUZU'ya tüm katkılarından ve hiç eksiltmediği desteginden dolayı teşekkür ederim.

HÜSEYİN MERT YAVAŞ

9 Eylül 2021

İÇİNDEKİLER

ÖNSÖZ

v

ŞEKİLLER TABLOSU

ix

1 GİRİŞ

1

2 1.BÖLÜM OYUNLAR HAKKINDAKİ KAVRAMLARIN İNCELEMELERİ

2

2.1 Oyun Sektörü	2
2.1.1 İlk Bilgisayar Oyunu	3
2.1.2 Hypercasual Oyunlar	4
2.1.3 Arcade Oyunlar	4
2.2 OpenCV Kütüphanesi	5
2.2.1 OpenCV kütüphanesi ile Geliştirilen Oyun Projeleri	6
2.3 Unity3D Oyun Motoru	8
2.4 Bosca Ceoil - Müzik Oluşturma Uygulaması	9
2.5 Aseprite - Pixel çizim uygulaması	9

3 2.BÖLÜM OYUN GELİŞTİRME

11

3.1 Aseprite Programıyla 2D Grafik Oluşturma	11
3.2 Bosca Ceoil	12
3.3 Unity3D Oyun Motoru ile Oyun Geliştirme	13
3.3.1 Ana Menü Sahnesinin Oluşturulması	15
3.3.2 Oyun Sahnesinin Oluşturulması	18
3.3.3 Bitiş Sahnesin Oluşturulması	26

4 ARAŞTIRMA BULGULARI VE TARTIŞMA

31

5 SONUÇLAR VE ÖNERİLER

32

6 EKLER

33

KAYNAKLAR

38

ŞEKİLLER TABLOSU

1	Oyun Turleri istatistiği [8]	3
2	Hangman Oyunu Ekran Görüntüsü [2]	6
3	Mosquito Killer Oyunu Ekran Görüntüsü [3]	7
4	Template Matching kaynak toplama aracı Ekran Görüntüsü [4]	7
5	Bosca Ceoil Uygulaması Ekran Görüntüsü	9
6	Aseprite Uygulaması Ekran Görüntüsü	10
7	Aseprite Uygulamasının Frog Ekran Görüntüsü	11
8	Aseprite Uygulamasının Player Ekran Görüntüsü	12
9	Aseprite Uygulamasının Arkaplan Ekran Görüntüsü	12
10	Bosca Ceoil Uygulamasının Tema Müziği Oluşturma Ekranının Ekran Görüntüsü	13
11	Oyunun İçerdiği Paketlerin Ekran Görüntüsü	14
12	Oyun Asset dosyalarının Ekran Görüntüsü	14
13	Ana Menu Sahnesinin Ekran Görüntüsü	15
14	Ana Menünün Yakından Ekran Görüntüsü	16
15	Ana Menünün Hiyerarşik Yapısının Ekran Görüntüsü	16
16	MenuMannager objesinin Ekran Görüntüsü	17
17	Ana Menünün Butonunun Ayarlarının Ekran Görüntüsü	17
18	Ana Menüde Bulunan Oyun Adı Ayarının Ekran Görüntüsü	18
19	Oyun Sahnesi Ekran Görüntüsü	18
20	Oyun Sahnesinin Hiyerarşisinin Ekran Görüntüsü	19
21	Oyun Asset dosyalarının Ekran Görüntüsü	20
22	Sprite Editorünün Ekran Görüntüsü	20
23	Player Animasyonu Oluşturulması	21
24	Player Animator ekranı Idle Bağlantısı	21
25	Kurbağa animasyonunun ve animatorünün Ekran Görüntüsü	22
26	Oyun Kamerası Birleşeninin Ekran Görüntüsü	23
27	Oyun Kamerasının Görüntüsü ve Konumu	24
28	Oyun Sahnesinin GameMannager Objesi	24

29	Oyun Sahnesinin MenuMannager Objesi	25
30	Player Objesine Fizik Motorunun Eklentimesi	25
31	Player Objesinin Kuyruk Efekti Ayarları	26
32	Bitiş Sahnesinin Ekran Görüntüsü	27
33	Bitiş Ekranının Hiyerarşik yapısının Görüntüsü	28
34	Bitiş Sahnesi Menumannager objesi Ayarları	29
35	Bitiş Ekranının Butonlarının Çalışma Ayarı	29
36	Bitiş Sahnesindeki Game Over Yazısının Ayarı	30

1 GİRİŞ

Bu bitirme çalışmasında OpenCV kütüphanesi incelenmiştir. İncelenen kütüphanenin verdiği avantajları oyun sektörü içine nasıl entegre edilebildiği anlatılmıştır.

Bu bitirme çalışması iki farklı bölümden oluşmakta olup; ilk bölüm açıklamalar ve incelemelerle, ikinci bölüm ise incelemelerin uygulanarak oluşturulan oyunun yapım aşamalarıyla alakalı olacaktır.

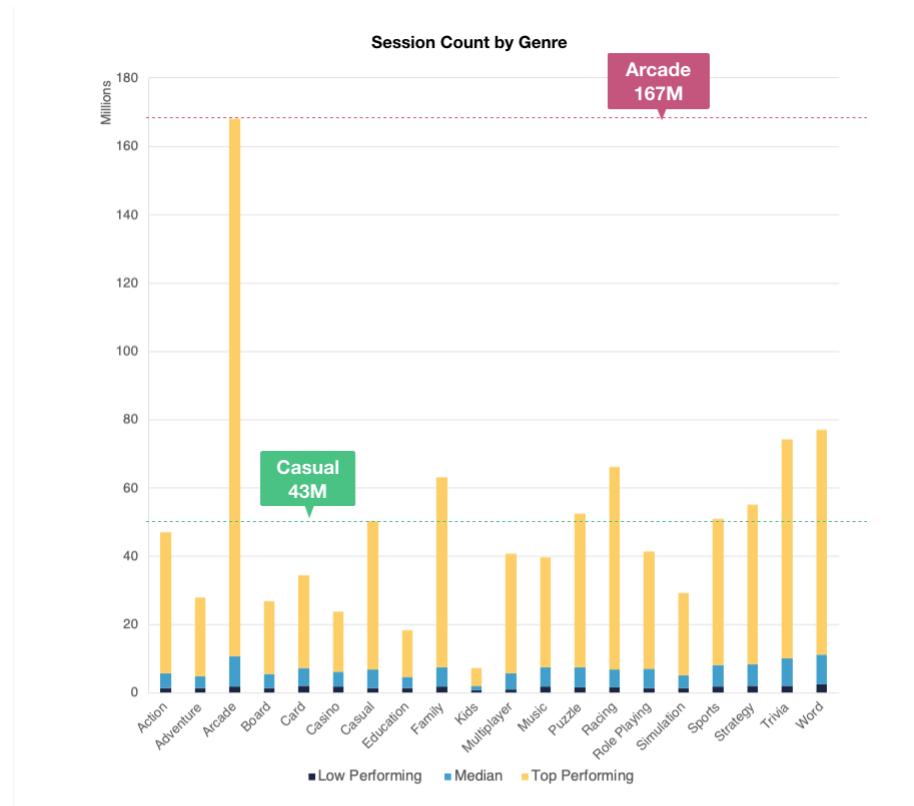
Oyun sektörüne genel bir bakış atılıp, sektör ilerleyişi hakkında incelemelere yer verilmiştir. Bu bitirme çalışmasında gerçekleşen oyun rehabilitasyon oyunları kapsamında değerlendirilebilecek yapıya evrilebilir. Bu konu öneriler kısmında ele alınmıştır.

2 1.BÖLÜM OYUNLAR HAKKINDAKİ KAVRAMLA-RİN İNCELEMELERİ

Bu bölümde genel tanımlamalar ve incelemeler mevcuttur. OpenCV, Unity3D, Oyun Sektörü alanları, aseprite, inceleneciktir.

2.1 Oyun Sektörü

Oyun sektörünün büyüklüğünü anlatmak için bazı başlıklar altında türlerinden bahsedilmiştir. Bunun amacı oyun sektörünün yerini anlatmaktadır. Şekil 1'de gösterilen grafik sektör büyülüğünü ayrıca anlatmıştır.



Şekil 1: Oyun Turleri istatistiği [8]

2.1.1 İlk Bilgisayar Oyunu

IBM ilk kişisel masaüstü bilgisayar IBM5150’yi duyurduğunda kullanıcılarına Donkey.bas sürprizi ile geldi. Microsoft Basic yazılımı altında çalışan IBM5150 de sadece bir gecede geliştirilen Donkey.Bas isimli oyun haricinde bilgisayarlar için geliştirilen ilk ticari oyun olma ünvanına sahip Adventure isimli Microsoft tabanlı bir oyun da bulunuyordu.

Neil Kozen'in de yardımıyla gelişen oyunda bitiş çizgisine gelene kadar yolunuza çıkan eşeklere çarpmamaya çalışmakla ilgili bir oyundur.

Donkey.bas günümüz yarış oyunlarının mimarisinde öncü olmuştur. Oyun içerisinde 4

farklı renk kullanılmaktaydı. Turkuaz, beyaz, gri ve koyu gri olmak üzere 4 farklı renk kullanılan oyunun sektörün büyümesi ve gelişmesindeki payı büyük. Oyun o zamanın teknolojisindeki bilgisayarlar için PC DOS sistemi üzerinden 8 bit olarak tasarlanmıştı.[5]

2.1.2 Hypercasual Oyunlar

Hyper casual, mantığı çok basit bir veya iki oyun mekanığıne dayanan, bu basitliğe rağmen kendini sürekli oynattırmayı amaçlayan, sade oyunların temsil ettiği türdür. Hyper casual, “aşırı basit” anlamına gelen bir terimdir. Bu türdeki oyunlar da tipki ismin belirttiği gibi aşırı basittir. Telefonda oynanılan, tek veya iki basit oyun mekanığıne dayanan fakat bu basitliğiyle oyuncuların oynamasını devam ettirmeyi amaçlayan bir türdür. Bu oyunlarda oynanış ve grafikler oldukça basit olurken oyuncunun oyunu oynamaya devam etmesi hedeflenmektedir. [6]

2.1.3 Arcade Oyunlar

Arcade oyunu, genellikle restoranlar, barlar ve atari salonları gibi halka açık işletmelerde kurulu jetonla çalışan bir oyun makinesi ve bu makinelerde oynanan oyunlardır. Arcade oyunlarının çoğu video oyunları, langırt makineleri, elektromekanik oyunlar, itfa oyunları veya satıcılardır.

Arcade oyunlarında oynanış tarzı bakımından eylem ön plandadır. Kaydetme ve tekrar yükleme seçenekleri yoktur.

İlk arcade benzeri makineler, ödeyenin geleceğini tahmin eden veya mekanik müzik çalan jetonla çalışan makinelerdi. Jetonla çalışan ilk langırt makineleri 1930'larda piyasaya çıktı. Elektronik langırt makinelerinin daha yaygın hale geldiği 1970'lerin sonlarına kadar mekanikti. Batı dünyasında video oyunu konsollarının yaygınlaşmasıyla arcade oyunlar gözden düştüse de günümüzde halen Uzak Doğu'da güçlü bir arcade endüstrisi bulunmaktadır.

Bu tür oyunlar aynı zamanda internet üzerinde muhtelif web sitelerinde de sunulmaktadır.

Flash yazılım teknolojisi ile geliştirilen Arcade oyunlarının bazıları skor tahtalı, bazıları ise sadece online olarak oynanabilirler ve oyuncuların eğlenceli vakit geçirmelerini sağlar. Bu tür oyunlar içinde aynı zamanda geliştirici ve eğitici oyunlar da bulunur. Özellikle gelişim çağındaki çocuklar için düşünülebilecek olan bu tür oyunlar muhtelif sitelerde bulunabilir. [7]

2.2 OpenCV Kütüphanesi

OpenCV kütüphanesi, OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmıştır ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğiniz anlamına gelmektedir. OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir.

2016-05-27 tarihli güncelleme, OpenCV geliştirici Itseez firması Intel tarafından satın alındı. OpenCV geliştirmesine Intel çatısı altından devam edeceğini duyurdu.

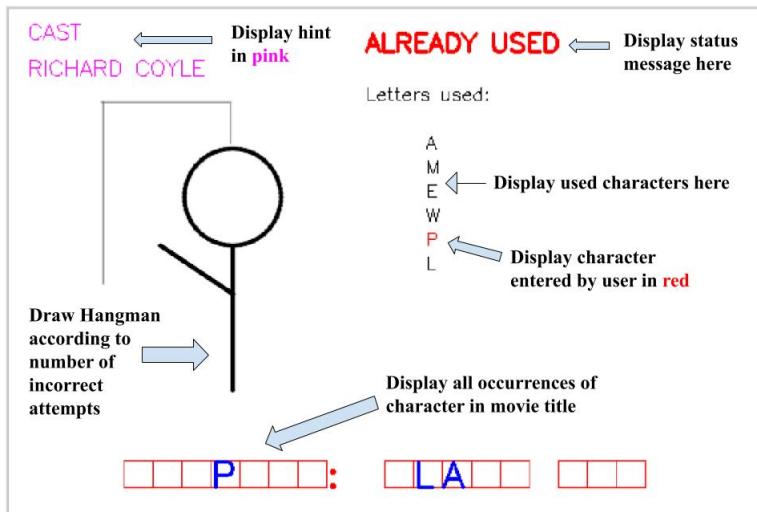
OpenCV kütüphanesi içerisinde görüntü işlemeye (image processing) ve makine öğrenmesine (machine learning) yönelik 2500'den fazla algoritma bulunmaktadır. Bu algoritmalar ile yüz tanıma, nesneleri ayırt etme, insan hareketlerini tespit edebilme, nesne sınıflandırma, plaka tanıma, üç boyutlu görüntü üzerinde işlem yapabilme, görüntü karşılaştırma, optik karakter tanımlama OCR (Optical Character Recognition) gibi işlemler rahatlıkla yapılabilmektedir.[1]

2.2.1 OpenCV kütüphanesi ile Geliştirilen Oyun Projeleri

OpenCv kütüphanesi kullanılarak birçok oyun geliştirilmiştir. Bu oyunlar kontrol etmek için ekstra oyun kumandası gibi aparatları hayatımızdan çıkartmaya yardımcı olacaktır. OpenCV ile yapılan bazı oyunların listesi aşağıda verilmiştir.

- **Hangman**

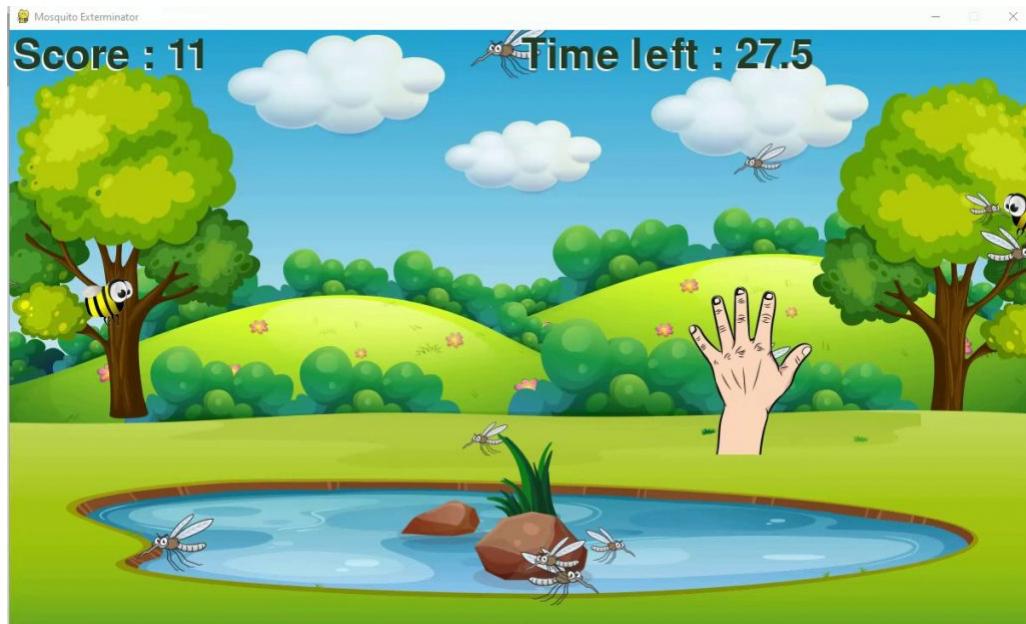
Not: Gösterilen oyun adam asmaca oyununun görüntü işlemeyle yapılmış halidir.



Şekil 2: Hangman Oyunu Ekran Görüntüsü [2]

- **Mosquito Killer**

Not: Gösterilen oyun elimizi algılayarak controller işlemi gören bir oyundur. Oyun-daki böcekelri el hareketimizde öldürebiliriz.



Şekil 3: Mosquito Killer Oyunu Ekran Görüntüsü [3]

- Template Matching

Not: Template Matching. Hazır bir oyunun kaynak toplama işlemine yardım etmesi için geliştirilen bir araçtır. Büyüyen veya spawn olan kaynakları otomatik toplayabiliyoruz.



Şekil 4: Template Matching kaynak toplama aracı Ekran Görüntüsü [4]

Görsterilen oyunlar ve yardımcı araçlar OpenCV kütüphanesiyle yapılmıştır. Geliştirilen oyunlar, oyuncuları oyunların içine dahil edebilmek adına büyük adımlar atmışlardır.

2.3 Unity3D Oyun Motoru

Unity ile Oyun Geliştirme: Unity, Unity Technologies tarafından geliştirilen ve öncelikle bilgisayarlar, konsollar ve mobil cihazlar için video oyunları ve simülasyonlar geliştirmek için kullanılan bir çapraz platform oyun motorudur.

İlk olarak 2005 yılında Apple’ın Dünya Çapında Geliştiriciler Konferansı’nda yalnızca OS X için duyurulmuş, o zamandan beri 27 platformu hedefleyecek şekilde genişletilmiştir.

Unity, 2D ve 3D grafikleri, sürükle ve bırak işlevini ve C# aracılığıyla komut dosyası oluşturmayı destekleyen çok amaçlı bir oyun motorudur.

Unity, özellikle mobil oyun geliştirme için popülerdir ve mobil platformlara odaklanmıştır. Unity3D ‘nin 2D ardışık düzeni, motora daha yeni bir eklemedir ve 3D pipeline hattından daha az olgunlaşmıştır. Buna rağmen Unity, diğer özel 2D motorlarla karşılaşıldığında bile 2D oyunlar geliştirmek için yeterli bir platformdur. Özellikle oyunu birden fazla mobil cihazda yayınılamayı planlıyorsanız.

VR (Sanal Gerçeklik) şu anda çok küçük bir pazar olmasına rağmen Unity, VR geliştirme için de iyi bir seçimdir. Mobil ve PSVR pazarları, sanal gerçeklikte en büyüğüdür ve Unity, oyunları PS4 ve PC gibi birçok platforma veya birçok farklı mobil pazara taşımak için halihazırda iyi bir konuma sahiptir.

Oyun Motoru bu grafik API’lerini hedefler: Windows ve Xbox One’da Direct3D; Linux, macOS ve Windows’ta OpenGL; Android ve iOS’ta OpenGL ES; Web üzerinde WebGL; ve video oyun konsollarında tescilli API’ler.

Ek olarak, Unity, iOS ve macOS’ta düşük seviyeli API’leri, Android, Linux ve Windows’ta Vulkan’ı ve Windows ve Xbox One’da Direct3D 12’yi destekler. 2D oyunlar içinde Unity, sprite ve gelişmiş bir 2D oluşturucusunun içe aktarılmasına izin verir. [9]

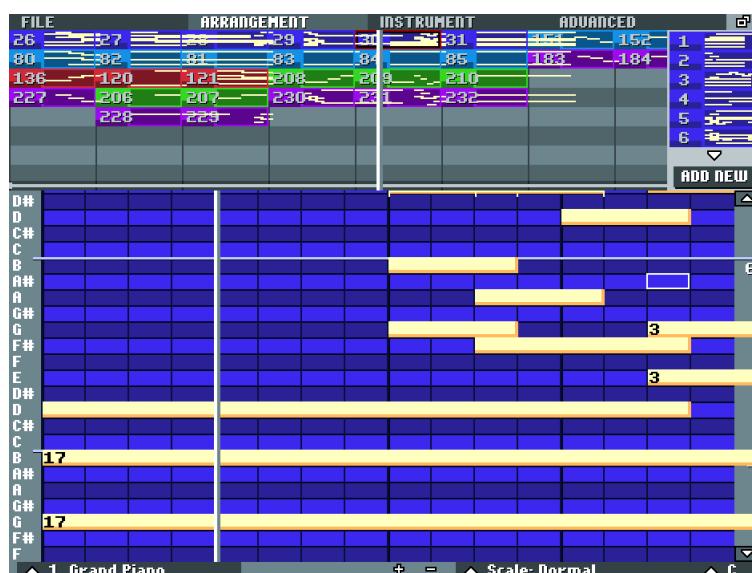
- inside
- Cuphead

- Hearthstone: Heroes of Warcraft

- Assassin's Creed: Identity

2.4 Bosca Ceoil - Müzik Oluşturma Uygulaması

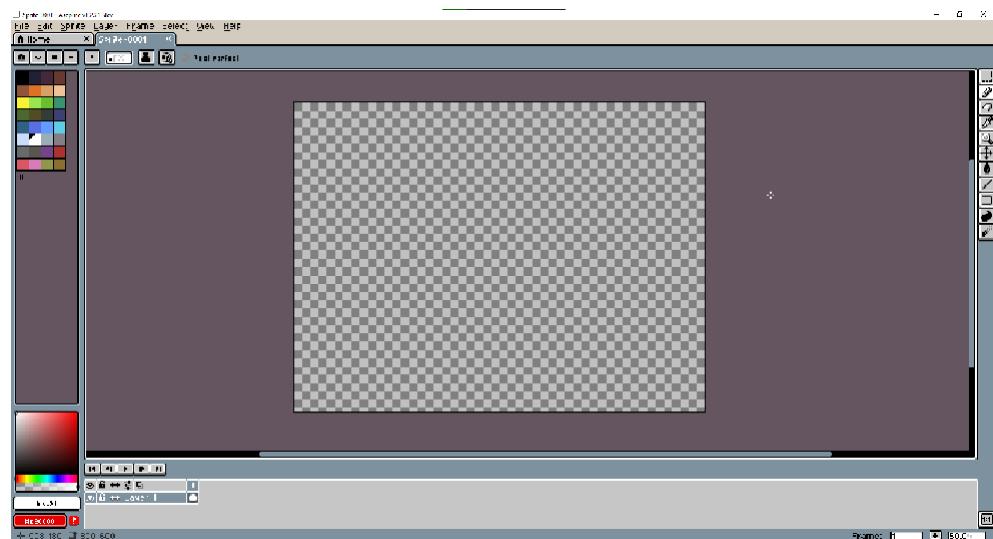
Bu uygulama ücretsiz olarak kullanabileceğimiz bir uygulamadır. Basit arayüzü sayesinde kolaylıkla müzik oluşturulabilektedir. 2. Bölümde anlatılan oyunda kullanılan müzik bu uygulama tarafından üretildi.



Şekil 5: Bosca Ceoil Uygulaması Ekran Görüntüsü

2.5 Aseprite - Pixel çizim uygulaması

Pixel art konusunda kolaylık sağlayan, pixel grafikler oluşturabilmeyi sağlayan bir uygulamadır. 2. Bölümde anlatılan oyunda kullanılan grafikler bu uygulama tarafından üretildi.

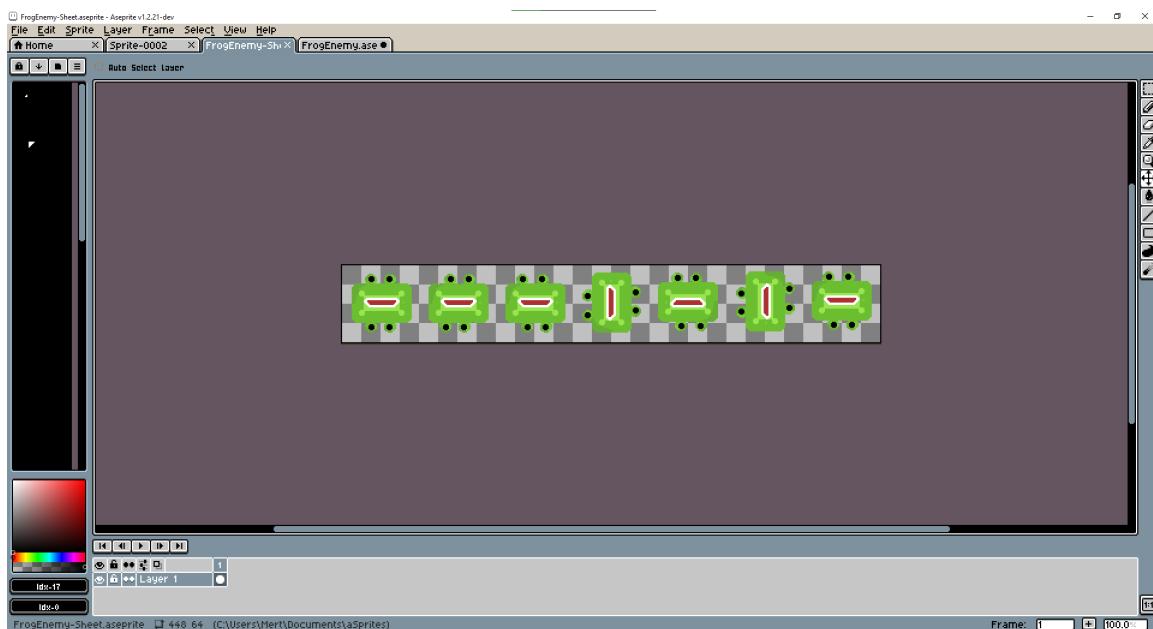


Şekil 6: Aseprite Uygulaması Ekran Görüntüsü

3 2.BÖLÜM OYUN GELİŞTİRME

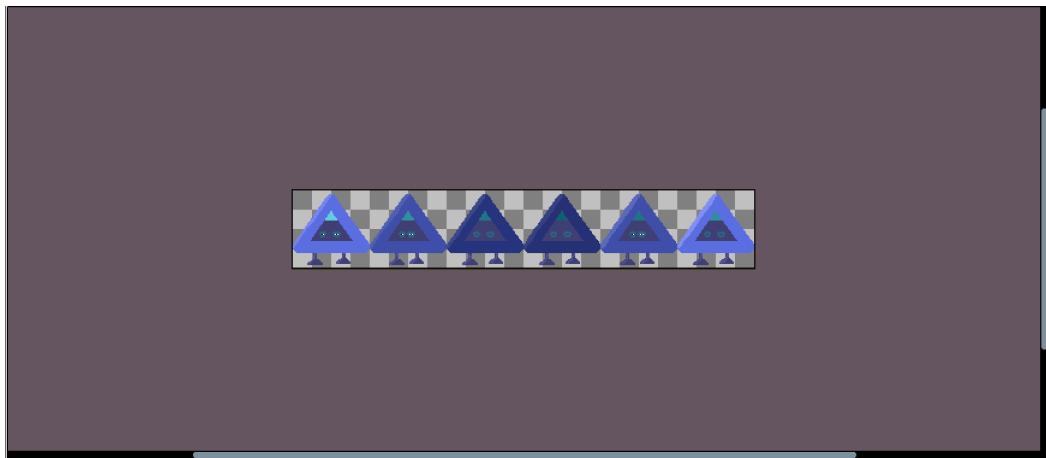
3.1 Aseprite Programıyla 2D Grafik Oluşturma

Aseprite programı kullanılarak 2D grafikler oluşturuldu. Grafiklerin oluşturma aşaması oyun sektörü için ayrı bir meslek kolu olarak geçmektedir. Oluşturulan grafikler istenildiği gibi olduğunda oyuna eklendi. Aşağıda gösterilen şekiller sprite olarak kullanılan karakterlerimizdir.



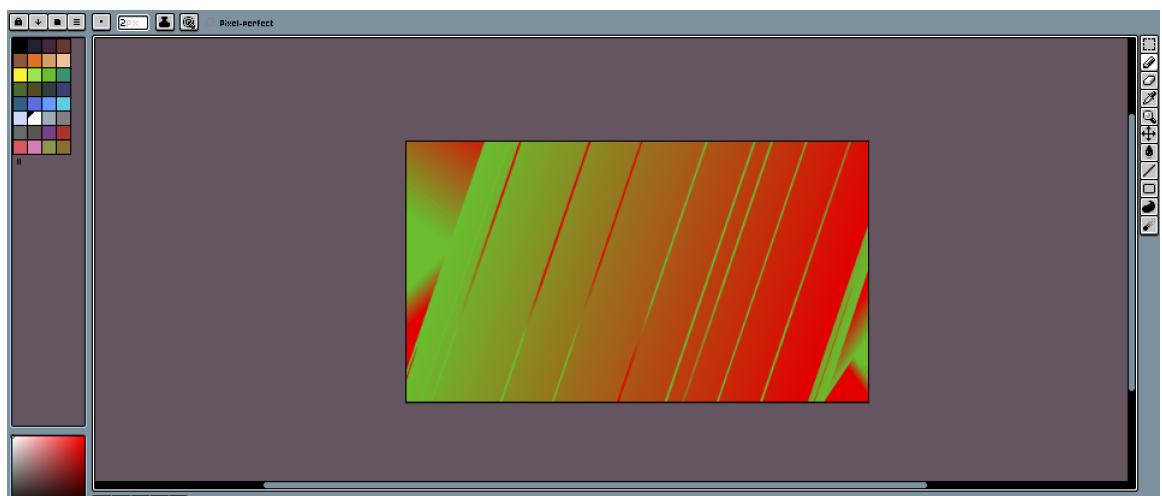
Şekil 7: Aseprite Uygulamasının Frog Ekran Görüntüsü

Kurbağa olarak düşünülen düşman karakteri için şekil 8'deki grafikler oluşturuldu.



Şekil 8: Aseprite Uygulamasının Player Ekran Görüntüsü

Fosforlu üçgen olarak oyuncu karakteri için şekil 9'daki grafikler oluşturuldu.

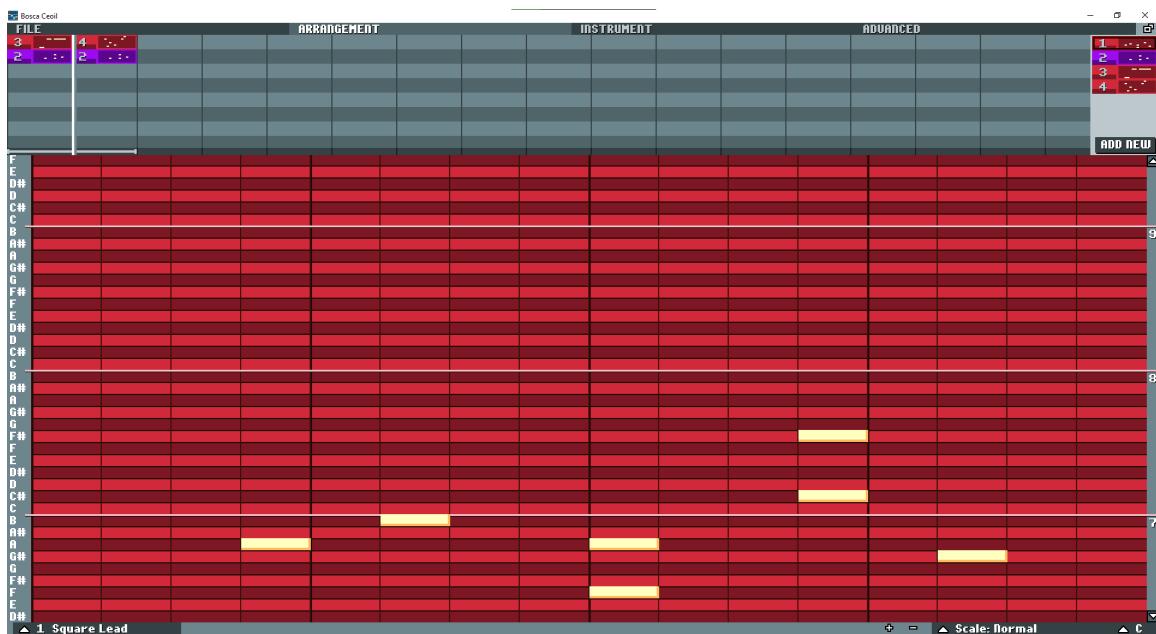


Şekil 9: Aseprite Uygulamasının Arkaplan Ekran Görüntüsü

Arkaplan için şekil 9'da görünen grafik yapıldı. Bu grafik oyunun içinde, başlangıç ve gameover ekranlarında görünen arkaplandır.

3.2 Bosca Ceoil

Müzik oluşturmak için kullanılan programda 2 müzik aleti kullanılmıştır. "Square lead" ve "Calm Bell" isimli enstrümanlar kullanıldı.

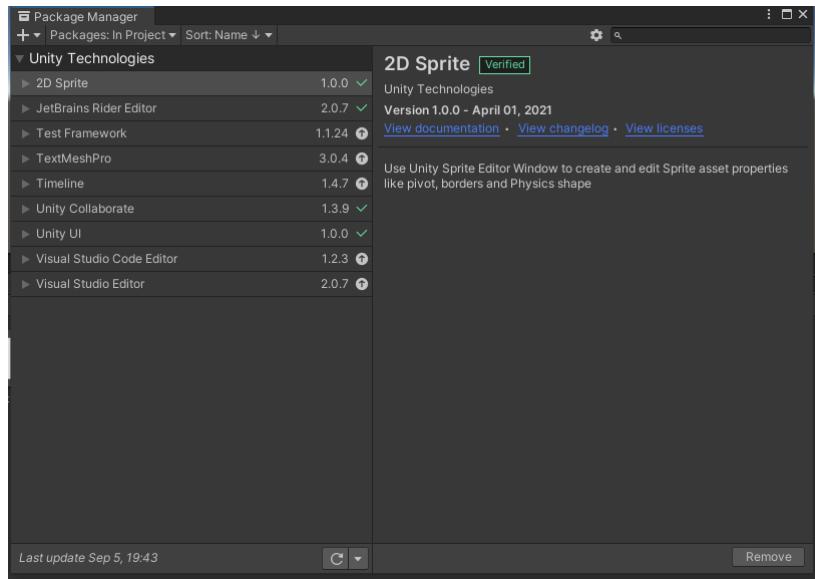


Şekil 10: Bosca Ceoil Uygulamasının Tema Müziği Oluşturma Ekranının Ekran Görüntüsü

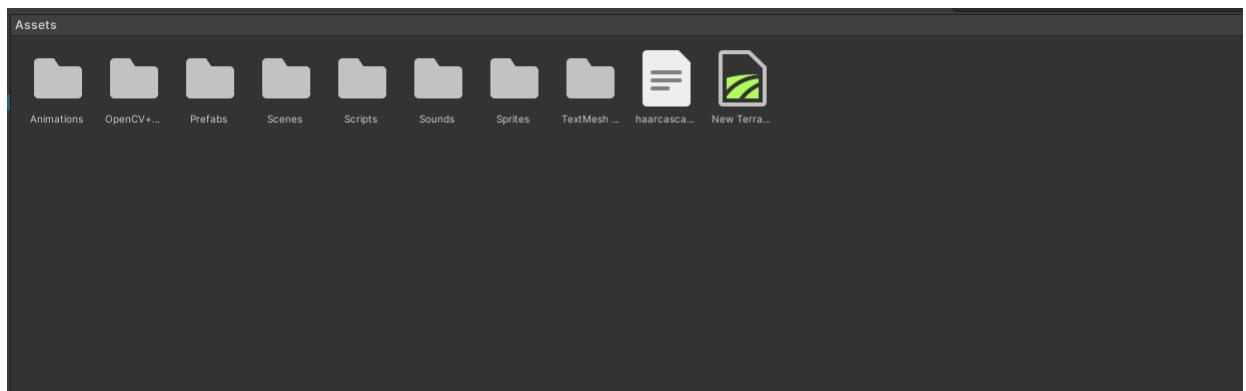
Şekil 10'da tema müziğinin bir aranjmanı gösterilmektedir. Bu programda küçük melodiler oluşturulup peş peşe eklenebilmektedir. Bu özellikler kullanarak çalışmadaki oyuna müzik oluşturuldu.

3.3 Unity3D Oyun Motoru ile Oyun Geliştirme

Bölüm 1'de yapılan incelemelerden yararlanarak OpenCV kütüphanesi kullanılarak oyun oluşturuldu. 3 aşamadan oluşan oyun sırasıyla Ana Menü, Oyun Sahnesi, Bitiş Sahnesidir. Oyuna bazı paketler eklenmiştir. Bu paketleri şekil 11'de gösterilmiştir.



Şekil 11: Oyunun İçerdiği Paketlerin Ekran Görüntüsü



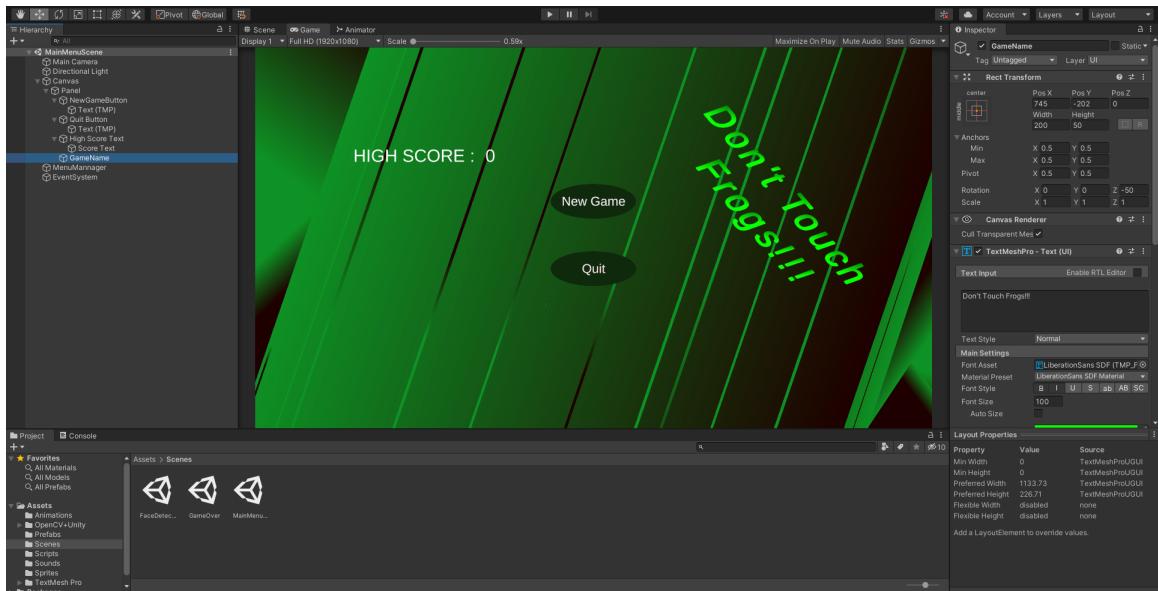
Şekil 12: Oyun Asset dosyalarının Ekran Görüntüsü

Şekil 12'de gösterilen görüntü karmaşıklığı ortadan kaldırmak için kullanılan dosya sistemidir. Oyunun büyük bütçeli olmasına göre çok fazla öğe girebileceğinden karmaşıklığa sebep olmaktadır. Bu karmaşıklığı ortadan kaldırmak için klasör klasör ayırmak önemlidir.

Oyunun yapım aşamaları ise aşağıda sırasıyla anlatılmıştır.

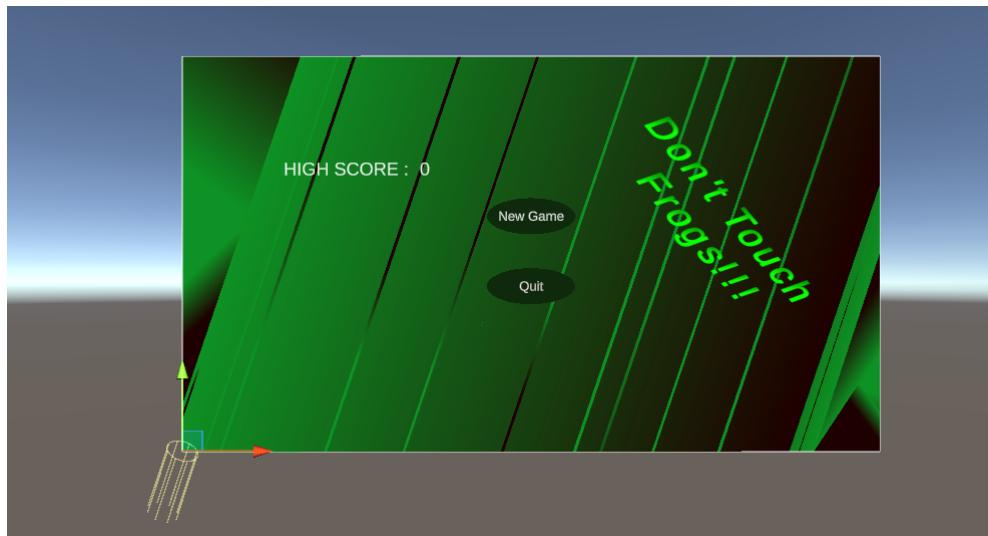
3.3.1 Ana Menü Sahnesinin Oluşturulması

Ana menü sahnesi oyunun ilk açılan sahnesidir. Bu sahnede oyuna giriş yapılmaktı ve oyundan çıkış yapılmaktadır. En yüksek puan ekranda yazdırılmıştır. Oyun ismi script yardımıyla büyütüp küçülmektedir.



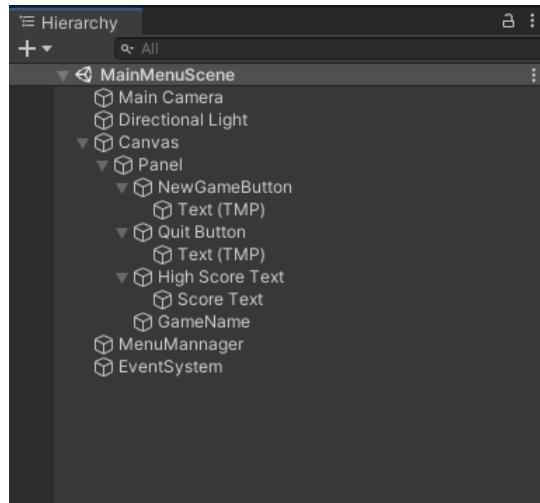
Şekil 13: Ana Menu Sahnesinin Ekran Görüntüsü

Şekil 13'te ana menü sahnesi geniş olarak gösterilmektedir. hiyerarşik yapısı ve sahne sistemlerini görebilmekteyiz. Hiyerarşik yapıdaki oyun objelerini sahneye uygun şekilde entegre ederek elde edilen görüntü oluşmuştur.



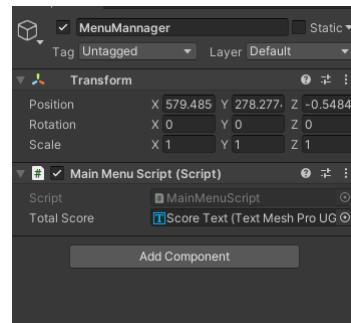
Şekil 14: Ana Menünün Yakından Ekran Görüntüsü

Şekil 14'te görülmekte olan görüntü sahnenin yapısını göstermektedir.



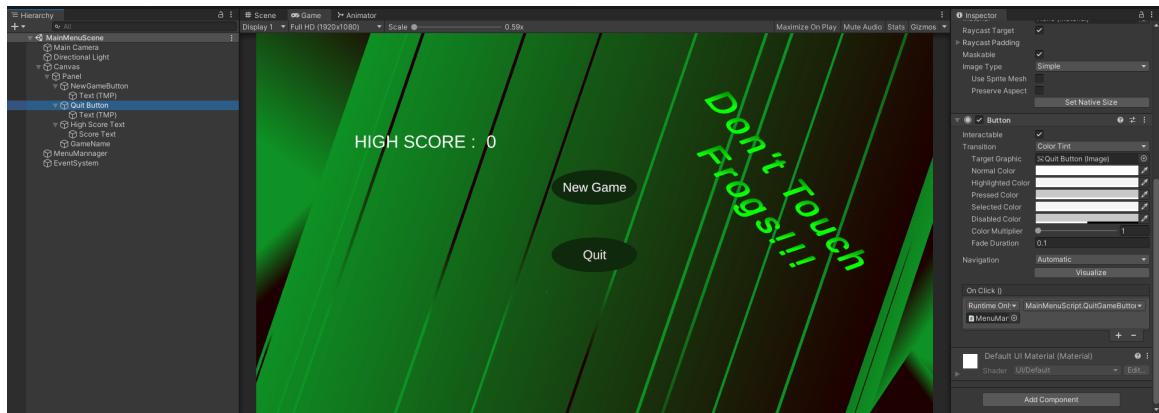
Şekil 15: Ana Menünün Hiyerarşik Yapısının Ekran Görüntüsü

Şekil 15'te gösterilen hiyerarşik yapı sayesinde menü oluşturmaktadır. Hiyerarşinin içinde kamera,Işıklandırma,Canvas,MenuMannager objesi,EventSystem olmak üzere 5 tane ana obje vardır. Bu objelerin altlarına child olarak bazı objeler eklenebilmektedir. Canvas objesinin altında panel objesi, panel objesinin altında button objeleri ve yazı objeleri bulunmaktadır. Canvas, arayüz objelerinden biridir. Childları da arayüz objeleri olup birleşimi arayüz oluşturmuştur. EventSystem objesi ise buttonların aktif olarak çalışmasını sağlamaktadır.



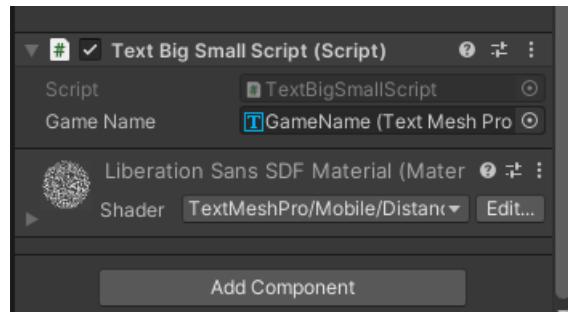
Şekil 16: MenuMannager objesinin Ekran Görüntüsü

Şekil 16'da gösterilen obje sayesinde butonların komutları verilmektedir. Menuscript içinde yazılı olan kodlar ve foksiyonlar buttonlara bu şekilde aktarım sağlamamız için kullanıldı. MenuMannager adı altında obje oluşturuldu. Bu objeye MainMenuScript adındaki script dosyası birleşen olarak eklendi.



Şekil 17: Ana Menünün Butonunun Ayarlarının Ekran Görüntüsü

Şekil 17'de buttonun MenuMannager objesinden aldığı scriptin fonksiyonunu çağırarak çalışması ayarlandı. Button birleşeninin içinde olan OnClick kısmına MenuMannager objesi atıldı. Ardından MainMenuScript içindeki foksiyon atanarak ayarların yapılması sağlandı.



Şekil 18: Ana Menüde Bulunan Oyun Adı Ayarının Ekran Görüntüsü

Oyun adının büyüp küçülebilmesi için Şekil 18'de TextBigSamallScript isminde bir script oluşturuldu. Bu script Ana menu sahnesinin hiyerarşisinde bulunan GameName text objesine eklenmiştir. Bu sayede script içeriğindeki kodlar artık gamename yazısına aktarılmış oldu.

Ana menü sahnesi bu adımlarla oluşturuldu.

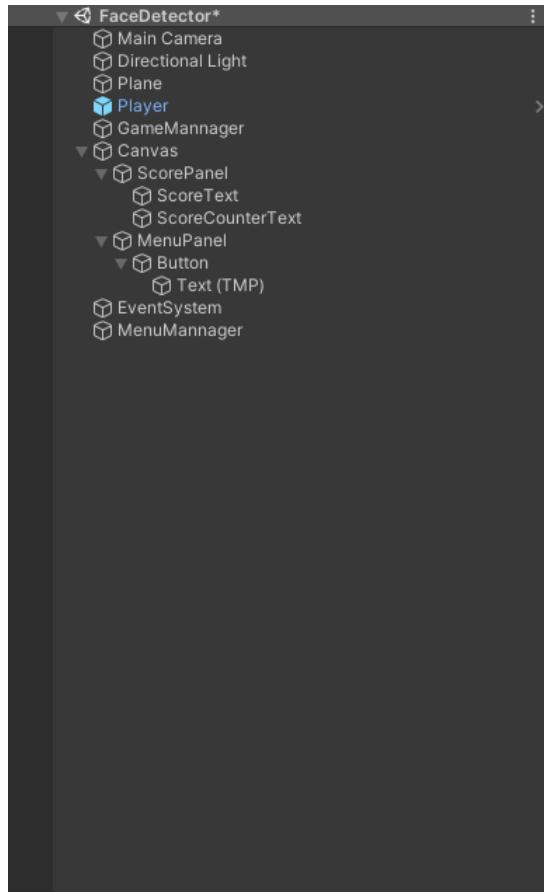
3.3.2 Oyun Sahnesinin Oluşturulması

Oyun Sahnesinin Arayüz görüntüsü şekil 19'daki gibidir.



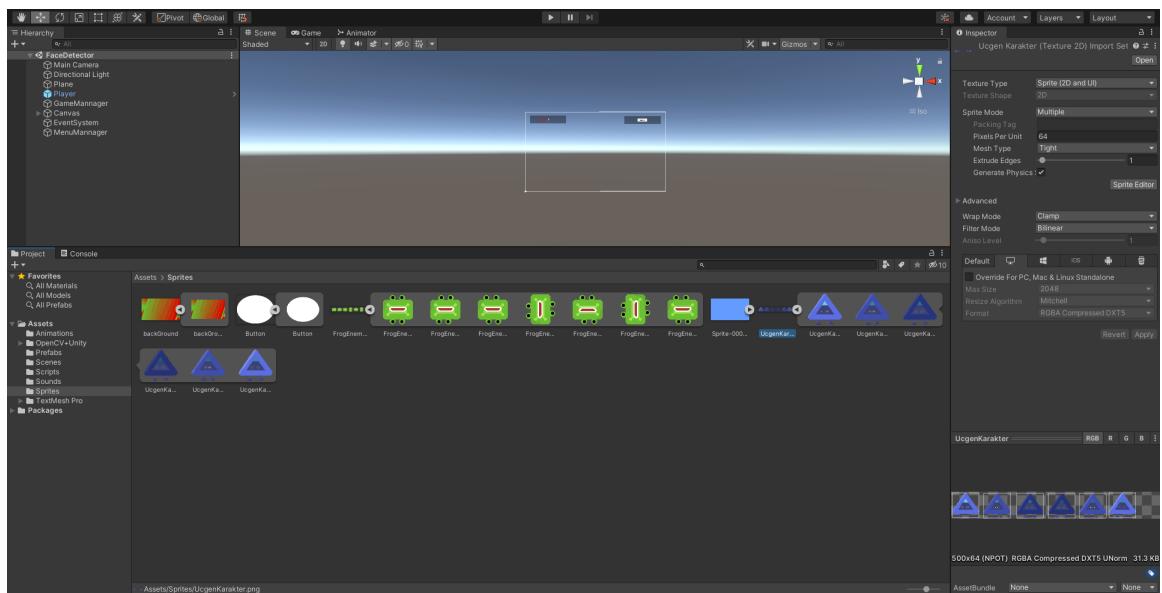
Şekil 19: Oyun Sahnesi Ekran Görüntüsü

Şekil 19'da oyunun arayüz kısmı görülmektedir. Sağ üstte menu butonu ve sol üstte dinamik olarak değişen score testi görülmektedir.



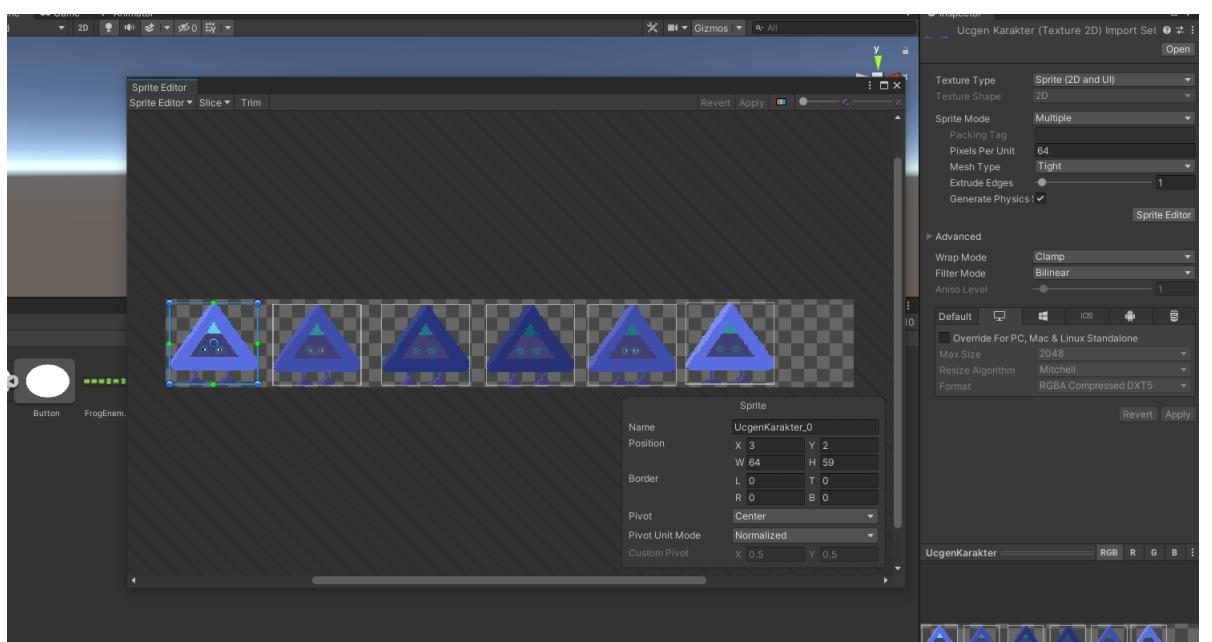
Şekil 20: Oyun Sahnesinin Hiyerarşisinin Ekran Görüntüsü

Şekil 20'de gösterilen hierarşik yapı, oyun sahnesinin oluşması için gerekli olan hierarşik yapıdır. Plane, Player, GameMannager, Canvas, EventSystem, MenuMannager olmak üzere 6 tane ana oyun objesi bulunmaktadır. Player, oyuncunun kontrol ettiği karakterdir. Plane, kamera görüntümüzün ekranda görünmesi için oluşturulmuş 3 boyutlu bir objedir. GameMannager oyun içindeki scriptleri kontrol etmek için üretilen bir objedir. MenuMannager menü butonu için üretilen objedir. Canvas arayüz objelerini çalıştırabilme için üretilen arayüz objesidir.



Şekil 21: Oyun Asset dosyalarının Ekran Görüntüsü

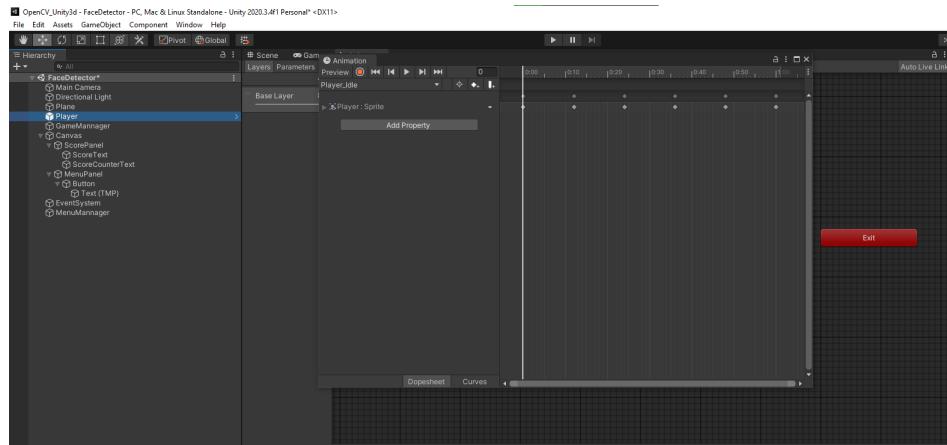
Şekil 21’de gösterilen görüntü, spritelerin bütünüdür. Bu spritelar oluşturulurken sprite editor kullanılmıştır. Sprite birleşenlerinden Sprite mode, multiple olarak ayarlandı. Bu sayede spriteların çoklu olarak bölme özelliği açılmış oldu.



Şekil 22: Sprite Editorünün Ekran Görüntüsü

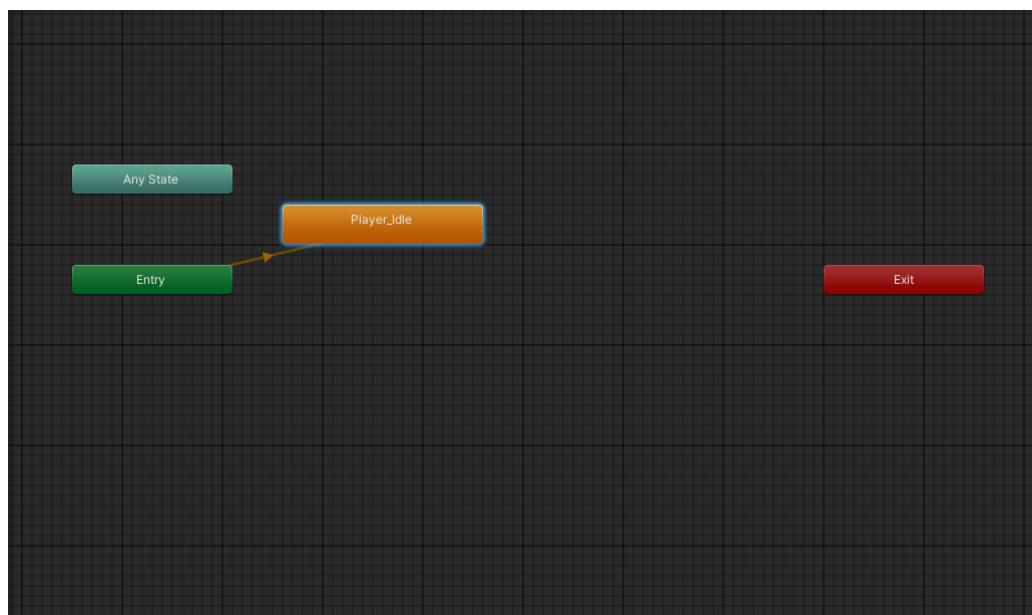
Sprite editor yardımıyla şekil 22’de spritelar çoklu olarak seçildi ve animasyon oluşturma

rulmadan önceki son işlemler yapıldı. Bu işlemle birlikte artık elimizdeki sprite tek bir sprite olarak görünmekten çıktı parçalı olarak birkaç sprite olarak bölündü. Animasyon oluşturmak için bu spriteları ard arda birleştirmek yeterli oldu.



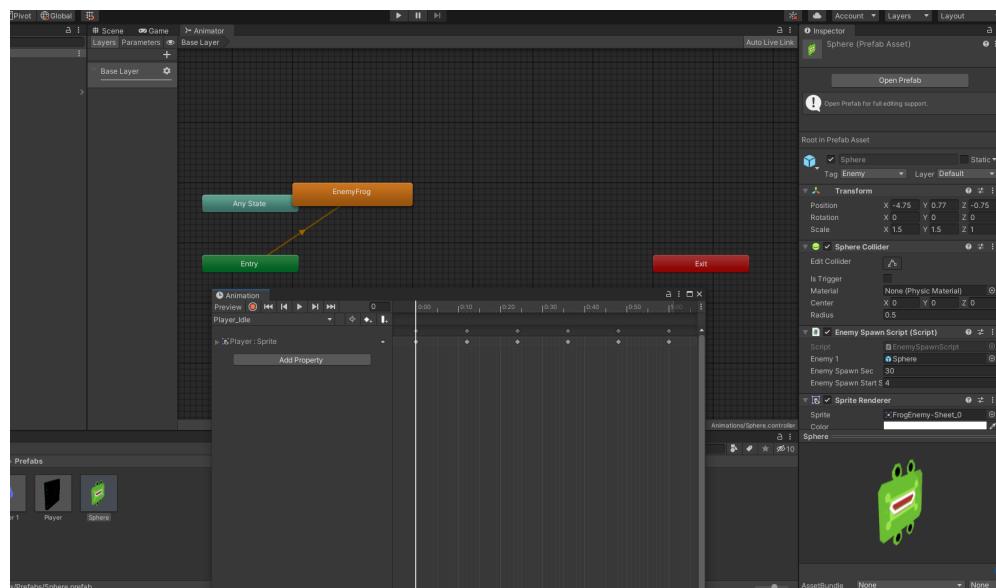
Şekil 23: Player Animasyonu Oluşturulması

Animasyon oluşturmak için çoklu olarak kullanılan player spriteları animasyon bölümüne sürüklerek bırakıldı ve sprite geçiş hızı sürüklerek bırakılan spriteların bitiş noktasından animation üzerinden sağ tarafa çekilerek ayarlandı. Animasyon ayarlarının görüntüsü Şekil 23'te gösterilmiştir.



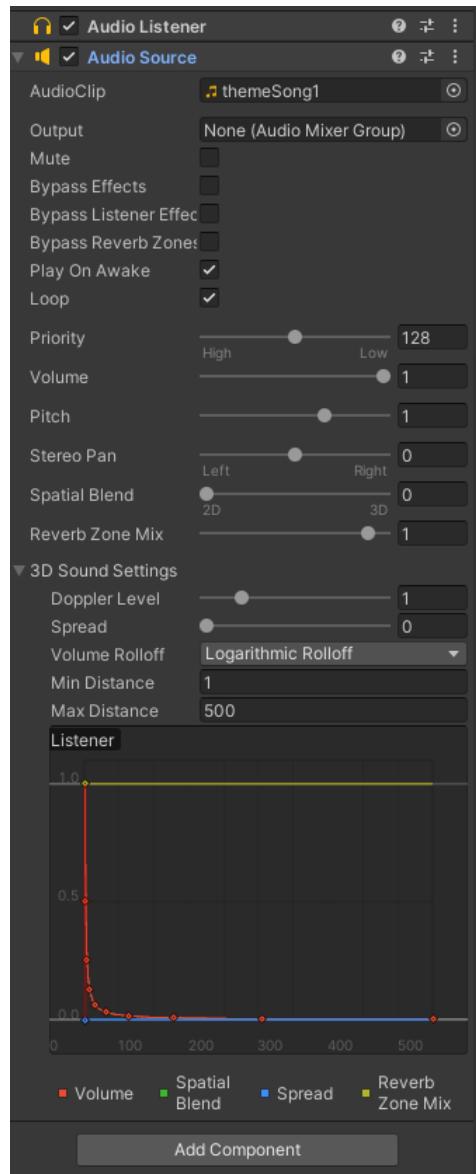
Şekil 24: Player Animator ekranı Idle Bağlantısı

Player animasyonunun kullanıcı tarafından görünmesi için animatorde bağlanması gerekmektedir. Ok direkt olarak Player idle bağlıdır. Başka bir animasyon olmadığı için oyun başlar başlamaz player idle bağlantısı oldu ve sonsuz döngüde kaldı. Bu bağlantı şekil 24'te gösterildi.



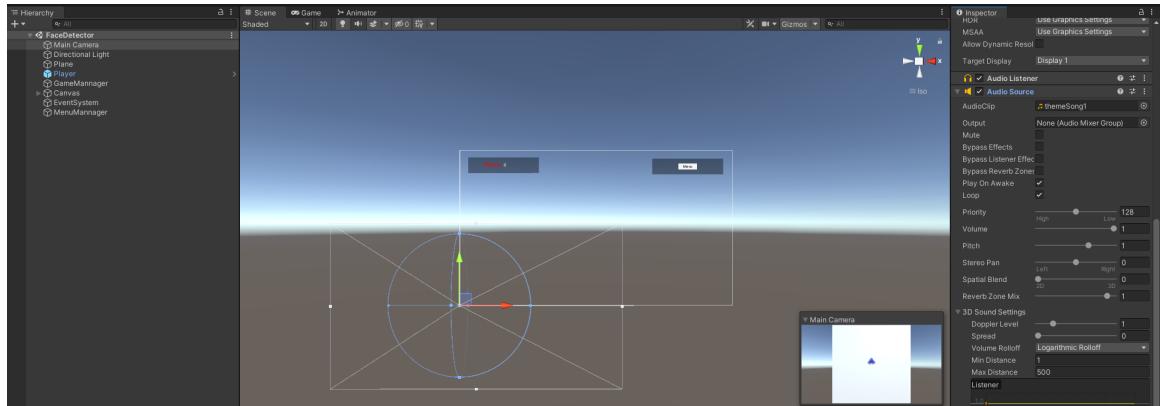
Şekil 25: Kurbağa animasyonunun ve animatorünün Ekran Görüntüsü

Şekil 25'te düşman olan kurbağaların animasyon bağlantılarının yapılmış olduğu şekilde gösterilmiştir. Bu bağlantı player objesinde yaptığımız bağlantıyla aynı şekilde yapılmaktadır. Sırasıyla spriteler animation içine sürüklendi ve sprite geçiş hızları ayarlandı. Ok direkt olarak EnemyFrog'a bağlıdır. Başka bir animasyon olmadığı için oyun başlar başlamaz EnemyFrog bağlantısı oldu ve sonsuz döngüde kaldı.



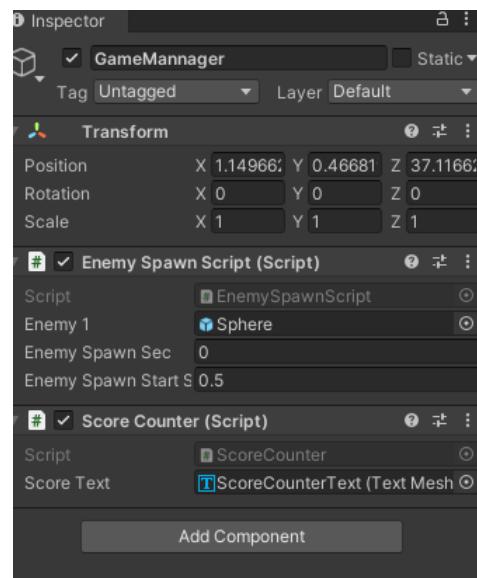
Şekil 26: Oyun Kamerası Birleşeninin Ekran Görüntüsü

Şekil 26'da gösterilen şekilde oyunun kamera birleşeni gösterildi. Oyun kamerasının birleşenine müzik eklendi. Böylelikle müzik sesi oyuna eklendi. Ön ayar gelen veriler kullanıldı. Gelen veriler editör üzerinden ya da kod ile değiştirilebilir.



Şekil 27: Oyun Kamerasının Görüntüsü ve Konumu

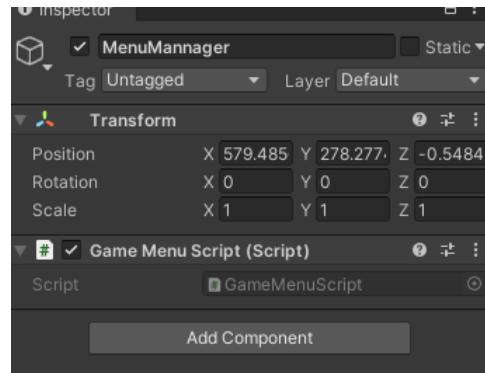
Oyun kamerasının konum ve görüntüsü şekil 27'de verilmiştir. Kamera yarımiyla kullanıcı oyunun sahnesinden dışarı çıkmamıştır.



Şekil 28: Oyun Sahnesinin GameMannager Objesi

EnemySpawnScript ve ScoreCounter Scriptleri Şekil 28'deki gibi GameMannager objesine aktarıldı. Bu obje yardımıyla Düşmanların oluşma sürelerini ve score verilerinin oyun sahnesine eklenmesi sağlandı. EnemySpawnScript Enemy1 adındaki alana düşman olarak olmasını istenilen oyun objesi eklendi. Bu sayede scriptte yazan kodlara göre ekranda düşman objeler oluşturuldu. EnemySpawnSec ve EnemySpawnStartSec isimli alanlar ise kaç saniyede ekrana düşmanın geleceğini ve oyun başladıkten kaç saniye sonra düşmanların oluşacağını belirlemek için public olarak ayarlanan değişkenlerdir. ScoreCounter

scripti ise oyun ekranında score kısmının dinamikleşmesi için yazılan koddur. Score text alanı ekranda dinamik olarak çalışması için gerekli olan text objesidir.



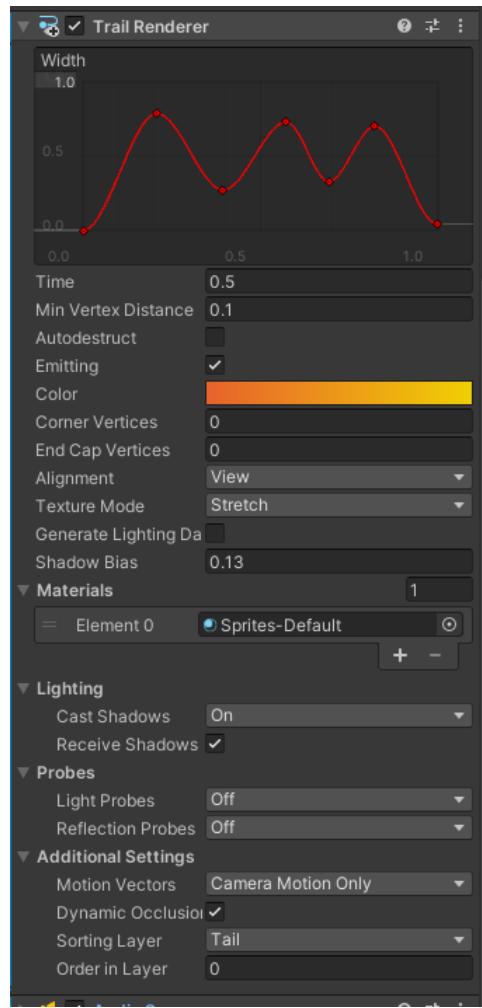
Şekil 29: Oyun Sahnesinin MenuMannager Objesi

Oyun sahnesinden ana menüye gidebilmek için şekil 29'da gösterilen aracı olarak kullanılan bir objedir. Butonlar için gereken fonksiyonlar bu objeden çekildi.



Şekil 30: Player Objesine Fizik Motorunun Eklenmesi

Player objesine fizik motoru eklenebilmesi için şekil 30'daki gibi bileşenleri içine rigidbody birleşeni eklendi. Döndürme özellikleri engellenmiştir. Bu sayede karaktere uygulanın kuvvet karakteri Y,Z eksenlerinde döndüremeyecektir. Eğer döndürmeleri engellememiş olsaydı karakter uygulanan bir kuvvette y ve z eksenleri etrafında dönmeye başlayacaktı. Bu da oyun içinde kötü ve amatörce görüntülerin hatta bugların oluşmasına sebep olurdu.

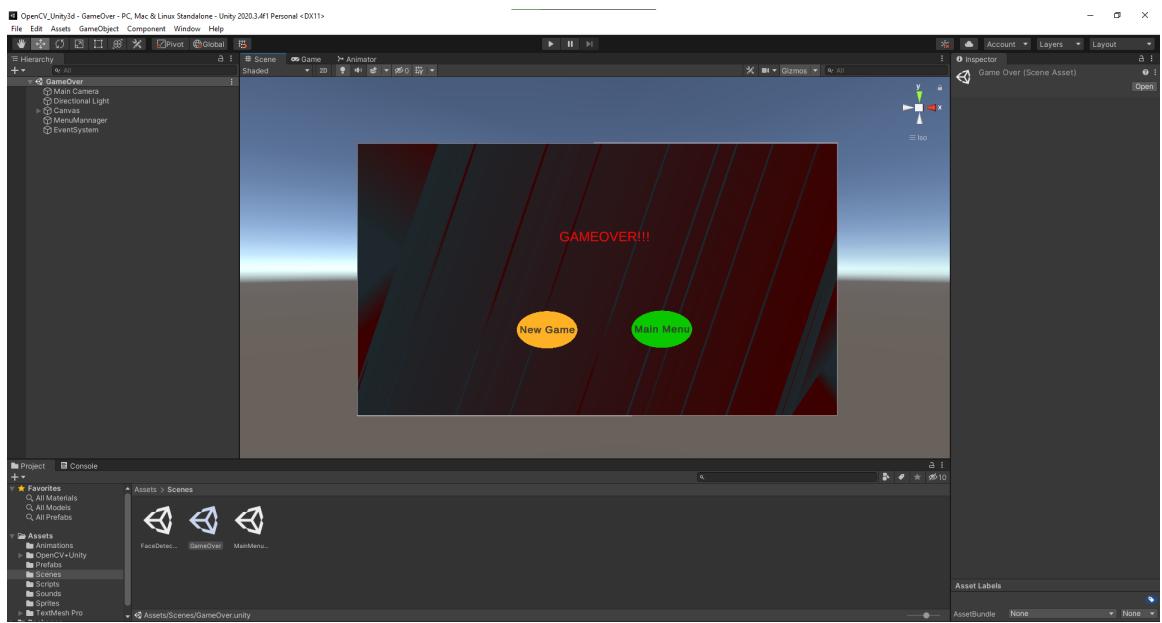


Şekil 31: Player Objesinin Kuyruk Efekti Ayarları

Player objesinin kuyruk efekti oluşabilmesi için şekil 31'de gösterilen ayarlar yapıldı. Width kısmındaki çizelge dalgalandırıldı. Color turuncu renkten sarıya doğru olacak şekilde değiştirildi. Time ve shadow bias kısımları da Şekil 31'de görünen float değerlerle değiştirildi.

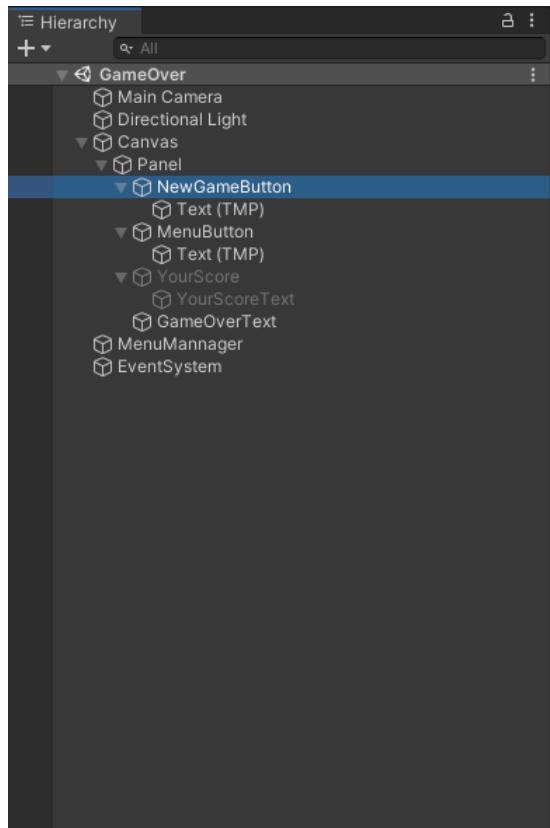
3.3.3 Bitiş Sahnesin Oluşturulması

Bu sahne son sahne olup bitiş ekranı sahnesidir. Oyuncu düşmanlara deðdiðinde veya belirlenen sınırların dışına karakterini götürdüğünde bu sahne devreye girer.



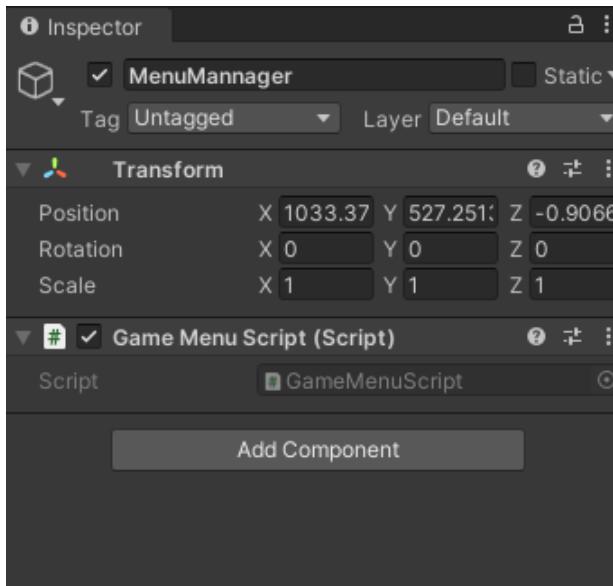
Şekil 32: Bitiş Sahnesinin Ekran Görüntüsü

Şekil 32'de gösterilen ekran Bitiş ekranıdır. Görüntüde hiyerarşik yapıdaki objelerin sahneye nasıl yerleştirildiği görüntülenmektedir. Butonların, yazının ve arkaplanın yerleşimi görüldüğü gibidir.



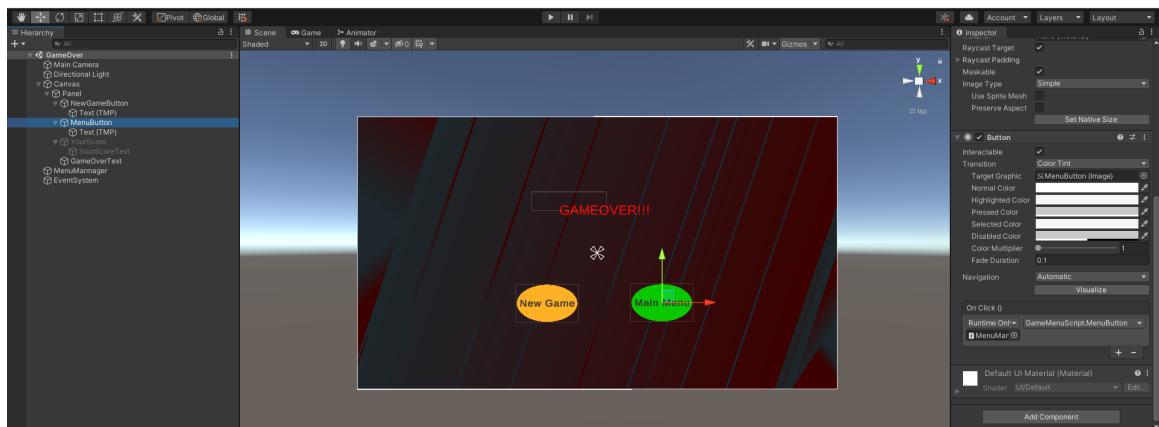
Şekil 33: Bitiş Ekranının Hiyerarşik yapısının Görüntüsü

Şekil 33'te gösterilen görüntü bitiş sahnesinin oluşturulması için kullanılan hiperarşik yapıdır. MenuMannager ve EventSystem menünün olduğu her yerde bulunması gereken iki objedir. MenuMannager objesi yerine script başka bir objeye birleşen olarak eklenerek kullanılabilir fakat bu durum proje büyükçe karmaşıklaşmasına neden olacağı için gereksiz bir hareket olacaktır. Canvas arayüz için gerekli olan objedir. Altında child olarak tanımlanan diğer arayüz objeleri bulunmaktadır.



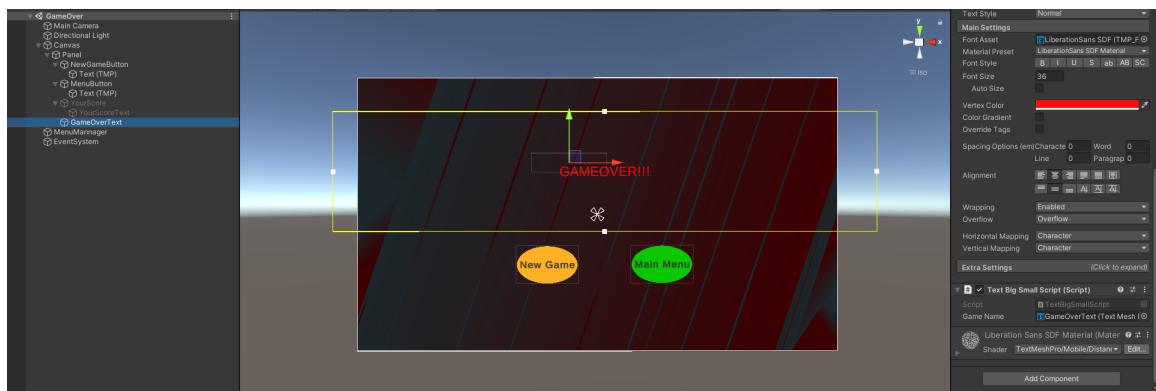
Şekil 34: Bitiş Sahnesi Menumannager objesi Ayarları

Şekil 34’te gösterilen görüntü bitiş ekranının menü ayarlarının yapılabilmesi için gereken objenin ayarlarıdır. Pozisyon değerlerinin hiç bir anlamı yoktur çünkü bu obje boş bir objedir ve sadece script aktarımı yapmak için kullanıldı. Oyunun herhangi bir yerinde olması sorun teşkil etmemektedir. Tek birleşeni ise scripttir.



Şekil 35: Bitiş Ekranının Butonlarının Çalışma Ayarı

Şekil 35’te bitiş ekranı butonlarının çalışma yapısı gösterilmiştir. Oluşturulan oyunda buton ayarları gösterildiği şekilde dir. MenuMannager objesi Onclick içinde bulunan boş alana sürüklendi ardından butonlar için gerekli olan fonksiyonlar seçilerek butonların çalışması sağlandı.



Şekil 36: Bitiş Sahnesindeki Game Over Yazısının Ayarı

Şekil 36'da gösterilen görüntü, game over yazısının ekranda büyüp küçülmesi için text objesinin bileşenlerinin arasına eklenmesi gereken script gösterilmiştir. Aynı script ilk ekranda da eklendi ve aynı işlemi yapması için ayarlandı. TextBigSmallScript adındaki script herhangi bir text objesinin bileşenlerinin içine eklenirse, eklenen text objesi scriptin içinde yazılı olan kodlara göre hareket edecektir. Game over yazısının yaptığı hareketlerin ve boyutunun büyüp küçülmesinin aynısını yapacaktır.

4 ARAŞTIRMA BULGULARI VE TARTIŞMA

Unity 3D oyun motoru oyun geliştiriciler için büyük bir imkan sağlamaktadır. Belirli bir kazancı geçtiği takdirde ücret istemektedir. Bu avantajından dolayı oyun geliştirmek isteyen her yaşta insan bu konular üzerine çalışabilir ve oyun geliştirebilir. Oyun geliştirmek tek başına yapılacak kadar ya da uzaktan bakılarak basit olarak görülecek bir konumda olmamalıdır. İstikrar ve disiplinle çalışmanın ve vazgeçmemek gereklidir. Oyun sektörüne bakacak olursak firmalar bir oyunu senelerce geliştirirler ve hala yetersiz bulunan kısımları ve çokça bugları bulunabilir. Bu kısımdan çıkartmamız gereken oyunun sadece yapımı değil çıktıktan sonrasında da geliştirmeye devam edilmesi gerekmektedir. Oyunlara yamalar çıkararak düzenli olarak teknolojinin gerisinde kalmadan güncel tutmak lazımdır. Güncellenen ve desteklenen oyunlar günümüzde hala oynanmaktadır. Buna örnek olarak Minecraft oyununu örnek verebiliriz. Gelen güncellemelerle oyuncuları kendilerine bağlı tutabilmektedirler.

Oyunları kaçırılmamak için güncel teknolojinin gelişimi ve yeni özellikleri sürekli olarak piyasaya sürmek önemlidir. Bunu yapmayan firmalar bir sonraki oyunları çıkana kadar ekonomik olarak kötü duruma düşebilir hatta firmalarını kapatmaya kadar gidebilirler.

5 SONUÇLAR VE ÖNERİLER

Oyun sektörünün büyülüğu ve finansal olarak dönen miktarların fazlalığı geliştiricileri bu sektörde çekmeye başlamıştır. Basit mekanikler ve sürükleyici oyunlar oluşturmak son zamanda revaçta olan türlerden biridir. Unity3D oyun motoru yardımıyla hypercasual türünde oyunlar kolaylıkla üretilmektedir.

OpenCV ve Unity3D birleşiminden ortaya çıkan oyunlar rehabilitasyon oyunları oluşturmamıza imkan sağlayabilmektedir. Gerçek hayatı fiziksel olarak hareket etmeye dayalı oyunlar insanların hareketsizliklerine çözüm olabilir. Hatta mola olarak kontrollü oyunlar oynayabilirler. Bu sayede sürekli olan hareketsizlikleri eğlenceli bir şekilde çözülmüş olacaktır.

Salgın hastalıkların son derece çoğaldığı bir dönemdeyiz. Bundan dolayı kumandaları elimize almak yerine kameralar karşısında hareketlerimize göre kontrol edebileceğimiz oyunların olması elden ele gezen oyun kumandalarını ortadan kaldırabilir. Bu sayede hijyen bir üst seviyede tutulabilir.

6 EKLER

Github link: https://github.com/HMertYAVAS/OpenCV_Unity3d

using System.Collections; using System.Collections.Generic; using UnityEngine; using UnityEngine.SceneManagement;

```
public class PlayerController : MonoBehaviour
```

```
//kamera açısından dolayı y yerine z boyutunu hareket ettiriyorum. //public AudioClip die;  
public AudioSource audioSource; FaceDetector faceDetector; private GameMenuScript  
gameMenuScript; //float lastY=0; float speed = 5f; float Xlast; float Xfirst; float Ylast;  
float Yfirst; public static bool gameOver=false;
```

```
//Start is called before the first frame update void Start() gameMenuScript = GameObject.  
Find("MenuMannager").GetComponent<GameMenuScript>(); faceDetector = (Fa-  
ceDetector)FindObjectOfType(typeof(FaceDetector)); // audioSource = gameObject.AddComponent<  
// audioSource.clip = die;
```

```
// Update is called once per frame void Update() CharacterMoveAlg(); GameArea();  
if(gameOver) SceneManager.LoadScene(2);
```

```
void CharacterMoveAlg() if(Xfirst == 0) Xfirst=faceDetector.FaceX; Yfirst=faceDetector.FaceY;
```

```
Xlast = faceDetector.FaceX - Xfirst; Ylast = faceDetector.FaceY - Yfirst;
```

```
if(Xlast>0) Xlast=1; if(Xlast<0) Xlast=-1;
```

```
if(Ylast>0) Ylast=1; if(Ylast<=0) Ylast=-1;
```

```
float step = speed * Time.deltaTime; //float normX = Mathf.Clamp(faceDetector.FaceX -  
lastX, -1, 1); //float normY = Mathf.Clamp(faceDetector.FaceY-lastY,-1,1);
```

```
transform.position = Vector3.MoveTowards(transform.position,new Vector3 (transform.position.x  
+ Xlast, transform.position.y - Ylast, transform.position.z ),step); void GameArea() if(this.gameObject
```

```

== "Player" && transform.position.x > 10.0f || transform.position.x < -10.0f || trans-
form.position.y > 10.0f || transform.position.y < -10.0f) gameOver=true; void OnCollisi-
onEnter(Collision other) if(other.gameObject.tag == "Enemy") // Debug.Log("enemy");
gameOver=true; GameMenuScript.pauseStatus = true;

using System.Collections; using System.Collections.Generic; using UnityEngine; using
TMPro;

public class TextBigSmallScript : MonoBehaviour public TMPro.TextMeshProUGUI ga-
meName; private float sec = 2.0f; private float max = 1.5f; private float anothermax =
0.75f; // Start is called before the first frame update void Start()

// Update is called once per frame void Update() sec -= Time.deltaTime;

if(sec <= max && sec > anothermax) gameName.fontSize=100; else if(sec <= another-
max && sec > 0) gameName.fontSize=120; else sec=max;

using System.Collections; using System.Collections.Generic; using UnityEngine; using
UnityEngine.SceneManagement; using TMPro;

public class MainMenuItem : MonoBehaviour public TMPro.TextMeshProUGUI to-
talScore; // Start is called before the first frame update void Start()

// Update is called once per frame void Update() totalScore.text = PlayerPrefs.GetInt("totalScore").To

public void NewGameButton() SceneManager.LoadScene(1); PlayerController.gameOver=false;
GameMenuScript.pauseStatus = false; Time.timeScale=1;

public void QuitGameButton() Application.Quit();

using System.Collections; using System.Collections.Generic; using UnityEngine; using
UnityEngine.SceneManagement;

public class GameMenuScript : MonoBehaviour public static bool pauseStatus; // Start is
called before the first frame update void Start()

```

```

// Update is called once per frame void Update()

private void FixedUpdate()

public void MenuButton() SceneManager.LoadScene(0); public void NewGameButton()
SceneManager.LoadScene(1); PlayerController.gameOver=false;

using System.Collections; using System.Collections.Generic; using UnityEngine; using
UnityEngine.UI; using TMPro;

public class ScoreCounter : MonoBehaviour public TMPro.TextMeshProUGUI score-
Text;

float sec;

// Start is called before the first frame update void Start()

// Update is called once per frame void Update() if (PlayerController.gameOver == false)
sec += Time.deltaTime;

scoreText.text = Mathf.Round(sec).ToString(); if (sec > PlayerPrefs.GetInt("totalScore"))
PlayerPrefs.SetInt("totalScore", (int)Mathf.Round(sec));

using System.Collections; using System.Collections.Generic; using UnityEngine;

public class EnemySpawnScript : MonoBehaviour public GameObject enemy1; public
float enemySpawnSec = 20.0f; public float enemySpawnStartSec = 10.0f; float enemyS-
peed; float enemyLifeTime =10.0f; float enemyRestartLifeTime= 20.0f; Vector3 enemyS-
pawnPos; // Start is called before the first frame update void Start()

InvokeRepeating("SpawnEnemy", enemySpawnStartSec, enemySpawnSec);

// Update is called once per frame void Update() //Enemy lifeTime if (enemyLifeTime
> 0) enemyLifeTime -= Time.deltaTime; else enemyLifeTime = enemyRestartLifeTime;
Destroy(enemy1.gameObject);

void SpawnEnemy()

```

```

Vector3 enemySpawnPos = transform.position; enemySpawnPos.x = (Random.Range (-10, 10)); enemySpawnPos.y = (Random.Range (-10, 10)); enemySpawnPos.z = -0.50f;

Instantiate(enemy1, enemySpawnPos, Quaternion.identity);

//this part just for spawn enemy area. its not important

// if(enemySpawnPos.x < 0 && enemySpawnPos.y < 0) // Instantiate(enemy1, enemySpawnPos, Quaternion.identity); //

// if(enemySpawnPos.x < 0 && enemySpawnPos.y < 0) // Instantiate(enemy1, enemySpawnPos, Quaternion.identity); //

using System.Collections; using System.Collections.Generic; using UnityEngine; using OpenCvSharp;

public class FaceDetector : MonoBehaviour WebCamTexture _webCamTexture; CascadeClassifier cascade; OpenCvSharp.Rect MyFace;

public float FaceY; public float FaceX; // Start is called before the first frame update void Start() WebCamDevice[] devices = WebCamTexture.devices; _webCamTexture = new WebCamTexture(devices[0].name); _webCamTexture.Play(); cascade = new CascadeClassifier(Application.dataPath + @"/haarcascade_frontalface_default.xml");

// Update is called once per frame void Update() //GetComponent<Renderer>().material.mainTexture = _webCamTexture; Mat frame = OpenCvSharp.Unity.TextureToMat(_webCamTexture);

FindNewFace(frame); Display(frame); if(PlayerController.gameOver) _webCamTexture.Stop();

void FindNewFace(Mat frame) var faces = cascade.DetectMultiScale(frame, 1.1, 2, HaarDetectionType.ScaleImage);

if (faces.Length >= 1) //Debug.Log(faces[0].Location);

MyFace = faces[0]; FaceY = faces[0].Y; FaceX= faces[0].X;

```

```
void Display(Mat frame) if (MyFace != null) frame.Rectangle(MyFace,new Scalar(250,0,0),2);
```

```
Texture newtexture = OpenCvSharp.Unity.MatToTexture(frame); GetComponent<Renderer>().material  
= newtexture;
```

KAYNAKLAR

- [1] <https://mesutpiskin.com/blog/opencv-nedir.html> [05.09.2021]
- [2] <https://learnopencv.com/hangman-creating-games-in-opencv/>
[05.09.2021]
- [3] <https://pysource.com/2021/08/24/how-i-built-a-computer-vision-game/>
[05.09.2021]
- [4] <https://github.com/ClarityCoders/ComputerVision-OpenCV>
[05.09.2021]
- [5] <https://www.digitalx.com/2019/08/24/dunyanin-ilk-bilgisayar-oyunu-donkey-bas/> [05.09.2021]
- [6] <https://www.savebutonu.com/hyper-casual-nedir-22466>
[05.09.2021]
- [7] <https://tr.wikipedia.org/wiki/Arcade> [05.09.2021]
- [8] <https://gameworkdobserver.com/2019/07/02/hyper-casual-167m-sessions-a-year> [05.09.2021]
- [9] https://www.3dmadmax.com/oyun_motor/unity-ile-oyun-gelistirme/

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : HÜSEYİN MERT YAVAŞ

Uyruğu : T.C.

Doğum Yeri ve Tarihi: Sakarya / 07.05.1996

Adres : Adapazarı / SAKARYA

Telefon : 05310851395

e-mail : hmertyavas@hotmail.com

EĞİTİM DURUMU

Lisans Öğrenimi : BŞEÜ Bilgisayar Mühendisliği Bölümü

Bitirme Yılı : 2021

Lise : Akyazı Anadolu Öğretmen Lisesi

İŞ DENEYİMLERİ

Yıl : -

Kurum : -

Stajlar : BMC Otomativ San./2021, TRA Bilişim/2021

İLGİ ALANLARI: Yazılım, Gezi, Spor, Kitap, Oyun, Kamp

YABANCI DİLLER:

İngilizce