T.C.
ERCİYES ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

UNITY ILE MULTI PLATFORM KORKU OYUNU

Hazırlayan

Sergen Pekşen
1030516766

Danışman
Dr. Fehim Köylü

Bilgisayar Mühendisliği
Bitirme Ödevi

Haziran 2020
KAYSERİ

"Unity ile Multi Platform Korku Oyunu" adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Bilgisayar Mühendisliği Bölümünde Bitirme Ödevi olarak kabul edilmiştir.

.... / .... / .....

JÜRİ :

Danışman : Dr. Fehim Köylü

Üye :

Üye :

ONAY :

Yukarıdaki imzaların, adı geçen öğretim elemanlarına ait olduğunu onaylarım.

.... / .... /20...
Prof. Dr. Veysel ASLANTAŞ
Bilgisayar Müh. Bölüm
Başkanı

ÖNSÖZ / TEŞEKKÜR

Bu projede bizlere desteğini ve yardımını esirgemeyen başta, danışman hocamız Dr.Öğr.Üyesi Fehim KÖYLÜ hocamıza ve proje yapım aşamalarında bize olan moral, motivasyon ve desteklerinden dolayı kıymetli ailelerimize ve arkadaşlarımıza teşekkürlerimizi ve sevgilerimizi sunarız.

UNITY ILE MULTI PLATFORM KORKU OYUNU

Sergen Pekşen
Erciyes Üniversitesi, Bilgisayar Mühendisliği
Danışman : Dr. Fehim Köylü

ÖZET

Bu çalışmada, Unity Oyun Geliştirme Platformu kullanılarak Mobil(Android, IOS)/PC(MacOs, Linux, Windows) platformlarında çalışan bir korku oyunu oluşturulmuştur. Bizler, yazılan bu oyunun bir temel olarak kullanılarak üstüne yeni seviyeler eklenmesini kolaylaştırmayı ve kod değişiklerinin olabildiğince minimal seviyede olmasını amaçlamaktayız. Bu şekilde, yazılan bir başka oyunun, bu temeli kullanarak, geliştirme süresinin olabildiğince azaltılmasını sağlamaktayız.

Anahtar Kelimeler: Yazılım Mühendisliği, Bilgisayar Grafik, Mobil Geliştirme

MULTIPLATFORM HORROR GAME DEVELOPMENT USING UNITY

Sergen Pekşen
Erciyes University, Computer Engineering
Supervisor: Dr. Fehim Köylü

ABSTRACT

İn this study, We have created a Mobile(Android, IOS)/PC(MacOS, Linux, Windows) horror game using Unity Game Development Platform. Our study aims to become a base for further similiar games. Further games using this study's codebase will only need minimal changes in its code, this will make it easier to add new features to those games. Using this study, we aim to reduce the development time in further games that uses this study's codebase.

Keywords: Software Engineering, Computer Graphics, Mobile Development

İÇİNDEKİLER

UNITY ILE MULTI PLATFORM KORKU OYUNU

ŞEKİLLER LİSTESİ

KISALTMALAR

UnityScriptingAPI	: Unitynin bütün kodlarını içerisinde bulunduran ve geliştiriciye sunduğu "Scriptler" bütünü.
Script	: C# kodu.
MonoBehaviour	: Unity objelerinin ana sınıfıdır. Bütün objeler bu sınıftan kalıtım alırlar.
Component	: Objelerin nasıl davranmasını gerektiğine karar veren parçalardır, her obje bileşenlerden oluşur.
ScriptableObject	: Eğer bir objeye bileşen atanmak istenilmiyorsa kullanılması gereken sınıf, bu sınıf genellikle bilgi saklamak için kullanılır.
Collider	: Unitynin yerleşik bir bileşenidir. Collider bileşeni bulunan objeler birbirlerinin içinden geçemezler(özellikle belirtilmediği sürece(Trigger değil ise)).
Raycast	: Unitynin yerleşik bir "Script"idir. Raycast, ona verilen kameradan bir çizgi çizerek "Collider" bulunan bileşenlere çarptığı zaman bilgi döndermesi görevini taşır. Bu çizgi oyuncu için görünebilir değildir.
Oyuncu	: Oyuncunun kontrolünde bulunan objedir.
Interactable	: First Person Controller tarafından objelerin değişime uğrayabilmesini sağlayan özel "Script" dir.
Scene (*Harita*)	: Oyunun bölümlerine verilen addır. Objelerin tutulduğu yerdir.
Panel	: Oyuncuya oyunun şu anki durumuyla ilgili bilgi vermek için oluşturulan ve içerisinde "Text", "Button" vb. elemanlar bulunduran UI objesidir.
Canvas	: Paneller bütünüdür. Panellerin konumunu ekranın çözünürlüğüne bağlı olarak ayarlamakla yükümlüdür.
UI	: Canvaslar bütünüdür. UI mobil kısımda ayrıca hareketten sorumludur.
Item	: "ScriptableObject"ler kullanılarak oluşturulmuş Envanter elemanlarıdır.
Inventory	: Oyuncunun "Item"ları tuttuğu çantadır.
Diary	: Oyuncunun oyun içerisinde yaptığı şeyleri tutan bir paneldir.
GameObject	: Oyunda görülen bütün nesnelere obje adı verilir.
Transform	: Objenin harita uzayındaki konumunu temsil eder.

Jump Scare	: Oyuncuyu korkutmaya yönelik çevredeki anlık değişimlerdir, bunlar resim, ses, canavar vb. olabilir.
Update	: Obje var olduğu sürece çalışan metottur, her "Frame" sonucu çağırılır.
Start	: Obje oluştuğunda çağırılan metottur, obje aktif olduğu zaman bir kez çağırılır, başka bir zaman çağırılmaz.
Awake	: Start metodundan önce çağırılan bir metottur, sınıfın Constructor metodu gibi düşünülebilir, Unityde constructor yerine kullanılır. Bir kez çağırılır.
Joystick	: UI daki kontrol elemanı. Oyuncu objesini hareket ettirmekten sorumludur. Mobil cihazlarda aktifleşir.
TouchField	: UI daki başka bir kontrol elemanı, mobil cihazlarda aktifleşir ve üzerine dokunulduğu zaman kameranın rotasyonunu belirler.
Frame	: Unitynin bütün çizim işlemlerini gerçekleştirip ekrana çıkardığı resim.
FPS	: Saniyedeki "Frame" sayısı.
Particle System	: Unitynin sağladığı bir bileşen. Parçacık efektlerini (örnek olarak yağmur) kontrol etmekle yükümlüdür.
Shader	: Modelde her pikselin görünümünden sorumlu matematiksel hesaplamalardır.
Texture	: Objenin materyalini sarmalayan dokudur.
Material	: Texture ve Shaderden oluşan bileşendir. Objenin görünümünü belirler.
Model	: Materyallere sahip objeler.
Event	: Oyuncunun "Collider" bileşeni ile çarpıştığı zaman aktive olan objelerdir. Bu objeler genellikle başka objeleri etkinleştirmek veya etkinliğini sonlandırmakla yükümlüdür.
UITracker	: Oyundaki bazı Panel elemanlarını takip etmekle yükümlü "Script".
PlayerTracker	: Oyuncuyu takip etmekle yükümlü "Script"
Safe	:Oyunun belirli bir "Transform" da bulunan kasa objesi.
Renderer	: Objelerin her Frame başı çizilmesinden sorumlu bileşendir.
Occlusion Culling	: Oyunda aktif kameranın görmediği statik objelerin "Renderer" bileşenini kapatmayı sağlar.
Flashback	: Oyuncunun gündüz zamanına transfer olup bazı eşyaların yerini hatırlamaya çalışması durumudur.
InventorySlot	: Itemları tutan ve UI üzerindeki objelerde bileşen olarak bulunan script.

GİRİŞ

Oyun sektörünün dünya çapında 150 milyar doların üzerinde bir cirosu olduğu göze çarpmaktadır. Bu rakamın 2022 yılında 320 milyar dolar olacağı öngörülmektedir [1]. Türkiye'nin bu sektörden pay sahibi olabilmesi için yeni gelişen teknolojiler aracılığıyla birden çok platforma yönelik oyunların geliştirilmesi, en çok market payına sahip olma şansını yanında getirir.

Türkiye'de ki oyun sektörünün dünyaya göre daha düşük değerde sahip olmasının nedenleri olarak: "Oyun tasarımı ve geliştiriminin yüksek maliyetli olması, bu konuda deneyimli işgücünü gerektirmesi, oyun yazılımı konusunda kamusal teşvik politikalarının olmaması, yayın, dağıtım ve pazarlama etkinliklerinin yetersizliği yerli oyun endüstrisinin gelişimini engelleyen en önemli nedenlerdendir" [2]

Unity Technologies, 100.000 $ doların altında ciroya sahip oyunlar için motorun kullanımını ücretsiz yapmaktadır [3]. Bunların yanında verdiği eğitimler ile kullanımını kolaylaştırmaktadır. Unity kullanılarak bu sektörde iyi bir başlangıç yapılabilir.

Unity, konsollar, bilgisayarlar, ve mobil cihazlar için oyunları ve simülasyonları geliştirmek için kullanılan ve çapraz platform bir oyun motorudur.

Motor, aşağıdaki grafik API'lerini hedeflemektedir:

- Direct3D
- OpenGL, OpenGL ES
- WebGL
- Video oyun konsolları üzerindeki sahipli API'ler

Unity'nin oyun yapımcılarına sağladığı bir kolaylık Unity ile geliştirilen bir oyunun herhangi bir altyapı değişikliğine gerek olmadan farklı platformlara (PC, Mac, Web, Android vb.) uygun olarak derlenebilmesidir. Bu sayede PC için hazırlanan bir oyun tek tıklamayla Android içinde çalışır hale getirilebilir.

Bu esneklikten yararlanılarak, yazılacak oyunun sadece "UI" kısmının ve bu UI'yı platforma göre adapte etmenin yolu bulunarak, oyunun istenilen her türlü platformda çalışmasını sağlanacaktır.

Literatür Özeti

Günümüzde Unity ile oyun geliştirme sektörüne olan ilginin yoğun olarak artmasından dolayı geliştirmiş olduğumuz oyun uygulamasına paralellik gösteren projelere denk gelmek elbette mümkündür. Bizim bu projedeki asıl amacımız daha öncede belirttiğimiz üzere uygulamamızı daha optimal ve her kesime hitap eden bir biçime sokabilmek ve en önemlisi de bu sektöre küçükte olsa bir katkıda bulunabilmekti. Geliştirmiş olduğumuz uygulamamızın en önemli özelliği oynayan kişilerin bilinç altına zarar vermeksizin sadece zevk alabileceği bir oyun platformu sunmasıdır. Bilinç altına zarar vermekten kastımız ise oyunun küçük yaştaki çocuklara hitap ederken onların ileriye dönük yaşamında iz bırakacak durumların oyun içerisinde gösterilmemesi durumudur. Bu fikirler doğrultusunda bizim projemize benzerlik gösteren diğer projelerdeki gerekli gördüğümüz ve görmediğimiz özellikleri ortaya koyarak Unity ile geliştirdiğimiz oyunumuzu bu literatür taramalarına dayandırarak gerçekleştirmiş olduk. Projemiz önceden alınmış diğer tescillere oranla ise, nihai çıktı patent gibi fikri/Sınai mülkiyet hakkı oldukça yüksektir.

Çalışmanın Amacı

Bu çalışmasının amaçları şu şekilde sıralanabilir:

- Olabildiğince nesne yönelimli programlama metodolojileri kullanılarak, yazılan kodun tekrar kullanılabilirliğinin artırılması.
- Bu oyunun kod temelini baz alan diğer oyunlar için geliştirme kolaylığının

sağlanması.

- Unity'nin sahip olduğu "Framework"ları OOP temellerine oturtarak maksimum verimlilik ile kullanmak.
- Yazılan UI sistemini diğer multiplatform oyun tarzlarında da çalışmasını sağlamak

Tezin Organizasyonu

Bu çalışmanın düzenlemesi şu şekildedir.

1. Bölüm, yapılan projenin teknik ve yazılımsal ihtiyaçlarına ayrılmış olup bu ihtiyaçların analizlerini içerir.

2. Bölüm, 1.Bölümde analiz edilen ihtiyaçların gerçekleştirilmesi anlatılmaktadır.

1. BÖLÜM

Gereksinim Aşaması

Bu bölümde oyunun nasıl yapılacağı ayrıntılarıyla tartışılmıştır. Oyunun nelerden oluşacağı, performansı, oyunun işlevsel gereksinimleri ele alınmıştır. Çıkarılan sonuç itibari ile oyunun gerçekleştirme aşamasına geçilmiştir.

1.1. Ürün ve Teknik Gereksinimler

Bu başlık altında şu soruların cevapları aranmıştır:

1. Uygulama hangi platformda çalışacak?
2. Uygulama çalıştığı platformun hangi sürümlerini destekleyecek?
3. Uygulama tekli veya çoklu şahıs oynanışını destekleyecek mi?
4. Uygulama nasıl gerçekleşecektir? Hikaye Nedir?
5. Uygulamanın model gereksinimleri nelerdir?
6. Uygulama kontrolleri nasıl olacak?

Soruların cevapları şu şekildedir:

1. Uygulama platformu PC (Mac, Windows, Linux) ve Mobile(Iphone, Android) olarak seçilmiştir.
2. Uygulamanın "Mobil" tarafında modern sistemlere yönelik geliştirilmesi kararlaştırılmıştır (Android 7.0+). Bu kararın sebebi uygulama içerisinde kullanılacak modellerin çokluğu ve ışıklandırmanın gerçek zamanlı yapılması

zorunluluğu sonucunda ortaya çıkacak olan sistem gereksinimleridir. PC tarafında ise herhangi bir sürüm belirlenmemiştir. Performans açısından "Occlusion Culling" tekniği kullanılması GPU daki yükü azalttığı için ayrıca tercih edilmiştir.

3. Oyun bir mobil korku oyunu olacağı için, sadece tekli şahıs oynanışı desteklenecektir.

4. Karakter kendisini, kendi evinin odasında bulur, uykuya dalar ve uyanır. Uyandığı zaman kendi odasının kapısının kitli olduğunu görür. Kendi odasının kapısının anahtarını masasının üstünde bulur. Bu anahtarla kapıyı açar, dışarı çıkar. Anne-Babasının odasına gider ve bu odanın kapısının da kitli olduğunu görür, karakter "Flashback" evresi yaşar, anahtarın nerde olduğunu görür. Merdivenlerden aşağı iner, inerken canavar aniden önünden geçer. Karakter bazı olaylar ile kapıların anahtarlarını bulur, bazen "Jumpscare" yaşar, en sonunda Anne-Babasının odasının kapısından içeriye girer, içerideki kasayı bulduğu ipuçlarıyla açar, kasanın içerisinden çıkan garaj kapısının anahtarını bulur, garajı açar, arabaya biner. Garaj kapısı açılırken canavar kapının arkasından belirir, arabaya zıplar ve oyun biter.

5. Model gereksinimleri, hikayeye göre belirlenmiş olup, öncelikle bir ev, evin eşyaları(lambalar, yatak, gardırop vb.), fener, canavar, ağaçlar, yol, garaj, anahtar olarak belirlenmiştir.

6. Uygulama PC tarafında klavye ve fare ile kontrol edilecektir. Mobil tarafında ise UI ile kontroller sağlanacaktır.

1.2. Obje Gereksinimleri

Objeden kasıt, Unity içerisindeki Obje(GameObject) lerdir. Bu objelerin birbirleriyle nasıl bir uyum içerisinde olması gerektiği, objelerin ortak yönleri, hangi objelerin ne gibi görevlerden sorumlu olduğu bu başlık altında tartışılacaktır.

1.2.1. First Person Controller(Oyuncu)

Oyun için özel bir tekil şahıs kontrolörü yazılmasına karar verildi. Bu özel kontrolör 2 parçadan oluşacak şekilde tasarlandı. İlk parça olarak oyundaki objelerle etkileşime geçebilmek amaçlı bir modül, ikinci parça olarak da tekil şahısın yürütülmesi, etrafa bakabilmesi vb. hareket fonksiyonlarını gerçekleştirecek bir modül. İlk parça"InteractObjects" olarak adlandırıldı. 2.Parçadan sorumlu 2 "Script" çıkarıldı, sırasıyla "CustomController" ve "PlayerMovement". Oyuncuya aynı zamanda hareketlerine bağlı olarak tepki sağlayacak bir kamera verildi. Bu kamera aynı zamanda "Raycast" den sorumlu olmak üzere "InteractObjects" "Script"ine referans olarak verilmesi planlandı.

1.2.2. Obje "Script"leri ve "Bu Script"ler Arası Etkileşimler

1.2.2.1. Interactable Objeler

Interactable objelerin yani Oyuncu ile etkileşime girebilen objelerin, "Raycast" kullanılarak tespit edilerek, eğer oyuncu "Interact" butonuna basıyor ise (PC için varsayılan E, Mobil için sadece UI daki kontrolör) "Interactable" sınıfından "Interact" methodu çağırılarak, bu obje ile etkileşime geçildiğinde ne olması gerektiğine karar verildi.

1.2.2.2. Pickupable Objeler

"Pickupable" objeler "MonoBehaviour"a nazaran "ScriptableObject" sınıfını kalıtım alan "Item" sınıfını içlerinde bulundururlar. Bu objeler için geliştirilen kodun, "Interactable" sınıfından kalıtım alarak "Interact" fonksiyonunu değiştirip kendi gereksinimlerine uydurarak kullanması planlanmıştır. Bu fonksiyon, oyuncu "Interact" butonuna bastığı takdirde kendi içerisinde bulunan "Item" sınıfını "Inventory" sınıfına ekleyerek, bağlı bulunduğu oyun objesini yok eder.

1.2.2.3. Item

"Item"lar, bu proje kapsamında 2 tane alt sınıftan oluşmaktadır, "Flashlight" ve "Key". "Item" sınıfı "Inventory" singleton sınıfında tutulan bileşenlerdir. "Item" sınıflarının "Use" metodu, bu "Item" sınıflarının "Inventory" UI ı üzerinde görünen resimlerinin üzerine tıklandığında çağırılır. Her "Item" in bir "Inventory" iconu ve adı vardır.

1.2.2.4. Key

"Key" "Item" ı belirli bir Id ye sahip olan bir "ScriptableObject" dir. Bu "Key" "Item"ını kullandığımız takdirde, oyuncunun bu "Item" ı eline alması(Objenin kameranın önüne getirilerek tutuluyormuş gibi gösterilmesi) kararlaştırılmıştır. Bu "Key" elde tutulduğu takdirde bir kapıyla çarpışması ile "Key" Id si, çarpışan kapıyla eşleşir ve bu kapıyı açar veya açmaz.

1.2.2.5. Door

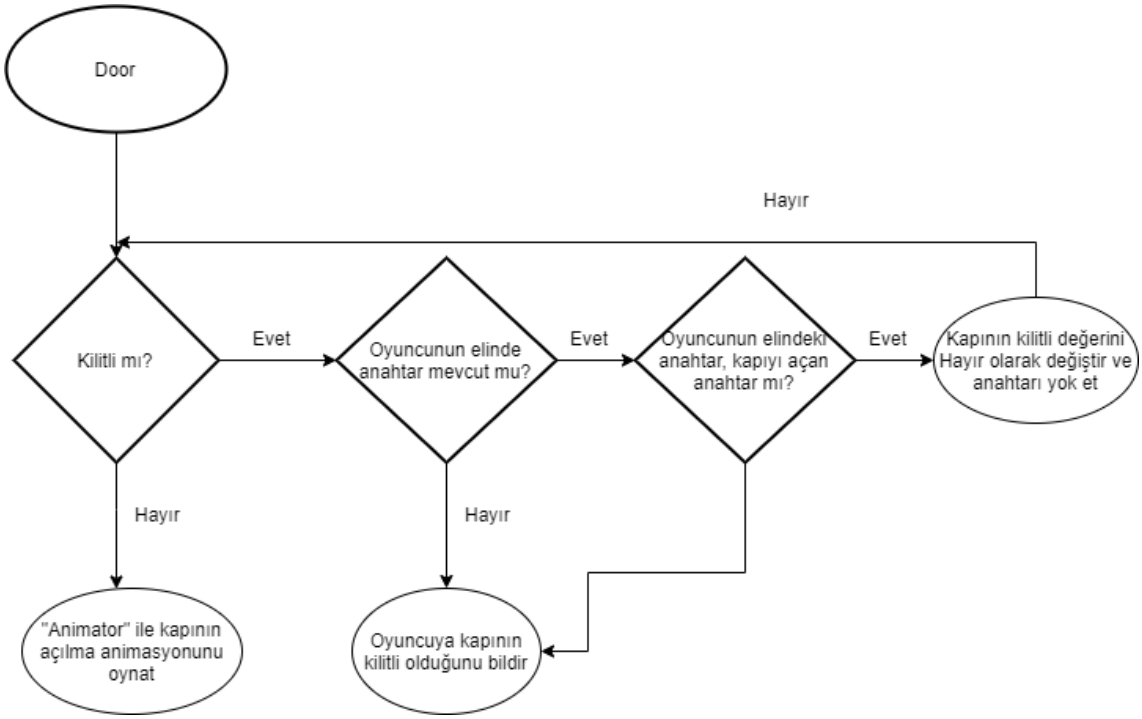
"Interactable" sınıfından kalıtım alan bir "Script"dir. "Door" "Script"i kendi içerisinde bir "Animator" fonksiyonu bulundurur, kendisi ile "Interact" edildiği takdirde, kilitli değilse bu "Animator" kapının açılması animasyonunu oynatır, kapı kilitli "UITracker" ile ekrana kapının kilitli olduğu bilgisi yazılır. Kilitli kapılar "Key" "Item"larının kullanılması ile açılabilir, eğer "Key" "Item"ının Idsi, kilitli kapıyı açacak "Key"in Id si ile aynı ise kapı açılır, "Key" yok edilir. Akış diyagramı Şekil 1.1 deki gibidir.

1.2.2.6. Inventory

"Inventory" sınıfı "Item" tutucu "Singleton" bir sınıftır. Bu sınıf için basit olarak bir "Item" listesi denilebilir.

1.2.2.7. Diary

Oyuncunun oyun içerisinde ilerlerken ortaya çıkan bazı bilgilerin tutulduğu paneldir.



Şekil 1.1. Door Akış Diyagramı

Bu panele ihtiyaç duyulmasının sebebi oyuncunun oyun içerisinde kaybolmamasını sağlamaktır.

1.2.2.8. Event

Oyuncu için belirlenen bazı olayların gerçekleşmesini sağlayan "Script" in var olması kararlaştırılmıştır. Bu olaylar, oyuncu belirli bir "Transform" noktasına eriştiği zaman ateşlenir. Örneğin oyuncu merdivenlerden aşağı inerken, merdivenin ilk adımına geldiği vakit, canavarın bir anlığına karaktere gözükmesi gibi.

1.2.2.9. Flashback

Oyuncunun oyun içerisinde bazı şeyleri hatırlaması mekaniğinden sorumlu bir "Script" yazılmak istenmiştir. Bu "Script" oyunu sabah veya akşam yapıp, "Flashback" objelerinin etkinleştirilmesi veya etkinliğinin sonlandırmasında görevlidir.

1.2.2.10. Safe

Oyuncunun bilmece çözmesine yönelik bir obje yapılması istenmiştir, bunun üzerine elektronik bir kasa ortaya çıkmıştır. Elektronik kasa, oyun içerisinde bulunan ipuçları ile şifresi girilerek açılır. İçerisinde bir anahtar barındırır.

1.2.2.11. Obje Tasarımları

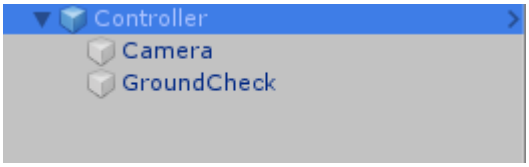
Objelerin tasarlanma kısmında, Google Sketchup kullanılmıştır. Bu program ile birlikte bir ev modellenmiştir. Evin içerdiği bazı malzemeler Google Sketchupın "3d Warehouse" kısmından temin edilmiştir. Bazı diğer malzemeler ise Unity asset store ve archive3d.net isimli siteden elde edilmiştir. "Texture"lar için ise freepbr.com tercih edilmiştir.

2. BÖLÜM

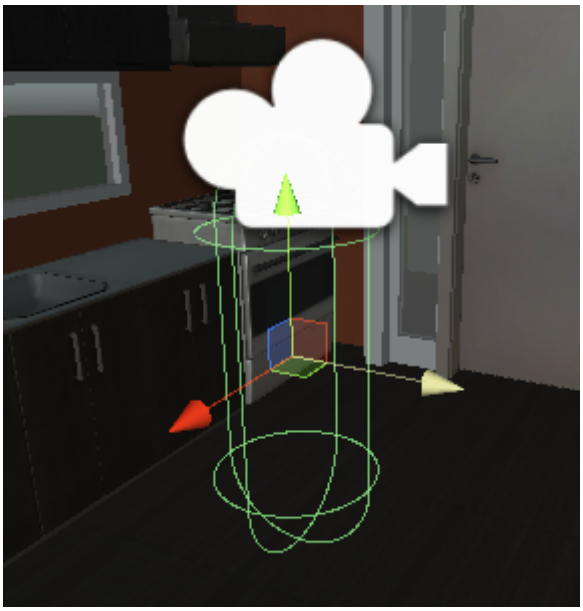
Gerçekleştirme Aşaması

2.1. Oyuncu (First Person Controller)

First Person Controllerin gerçekleştirilmesi için obje Şekil 2.1 ve Şekil 2.2 de görüldüğü gibi tasarlanmıştır. "Camera" bileşeni, oyuna bakış açımızı belirler,



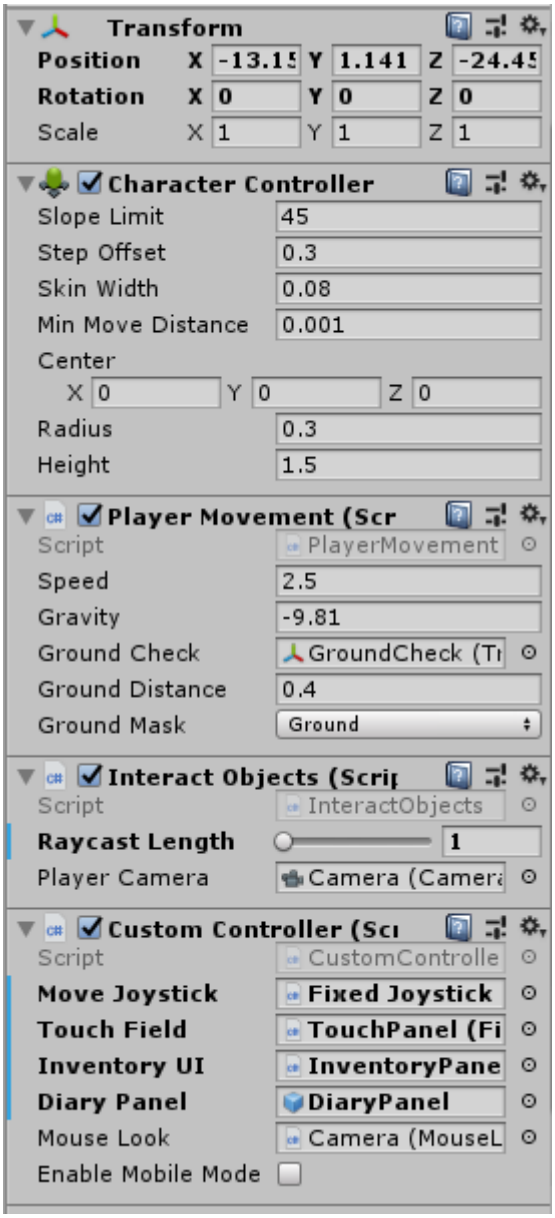
Şekil 2.1. Oyuncu Objesi Hiyerarşisi



Şekil 2.2. Oyuncu Objesi Oyun İçin Görünümü

objeleri cihaza görünür kılar. "GroundCheck" boş(sadece "Transform" bileşenine sahip) bir obje olup oyuncumuzun "Ground" da olup olmadığı test eder, böylelikle

zıplama gerçekleşebilir. Controllerimizin sahip olduğu bileşenler Şekil 2.3 de görüldüğü gibidir. Şekil 2.3 de bahsedilecek önemli "Script"ler “Custom Controller”,



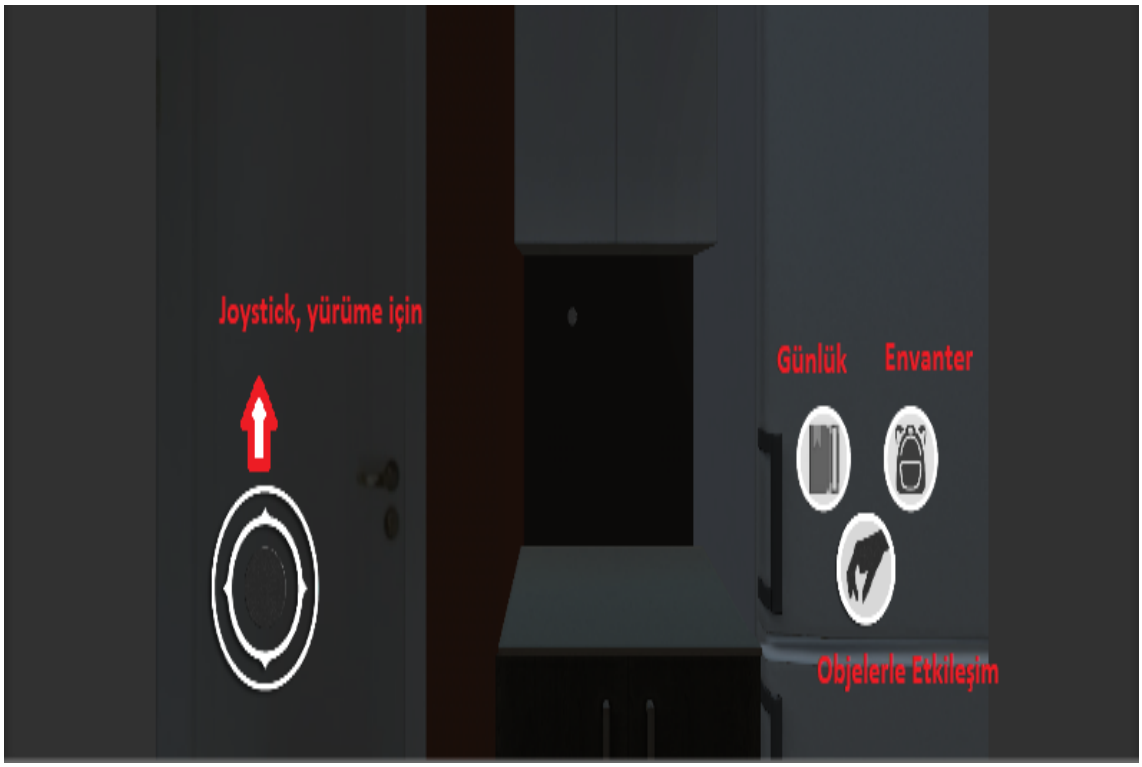
Şekil 2.3. Oyuncu Objesi Bileşenleri

“Interact Objects”, “Player Movement” dir. Unity bize “Character Controller” bileşenini kendisi sağlamıştır, içerisinde var olan komutlar yazdığımız özel kontrol scriptlerinde bize kolaylık sağlar.

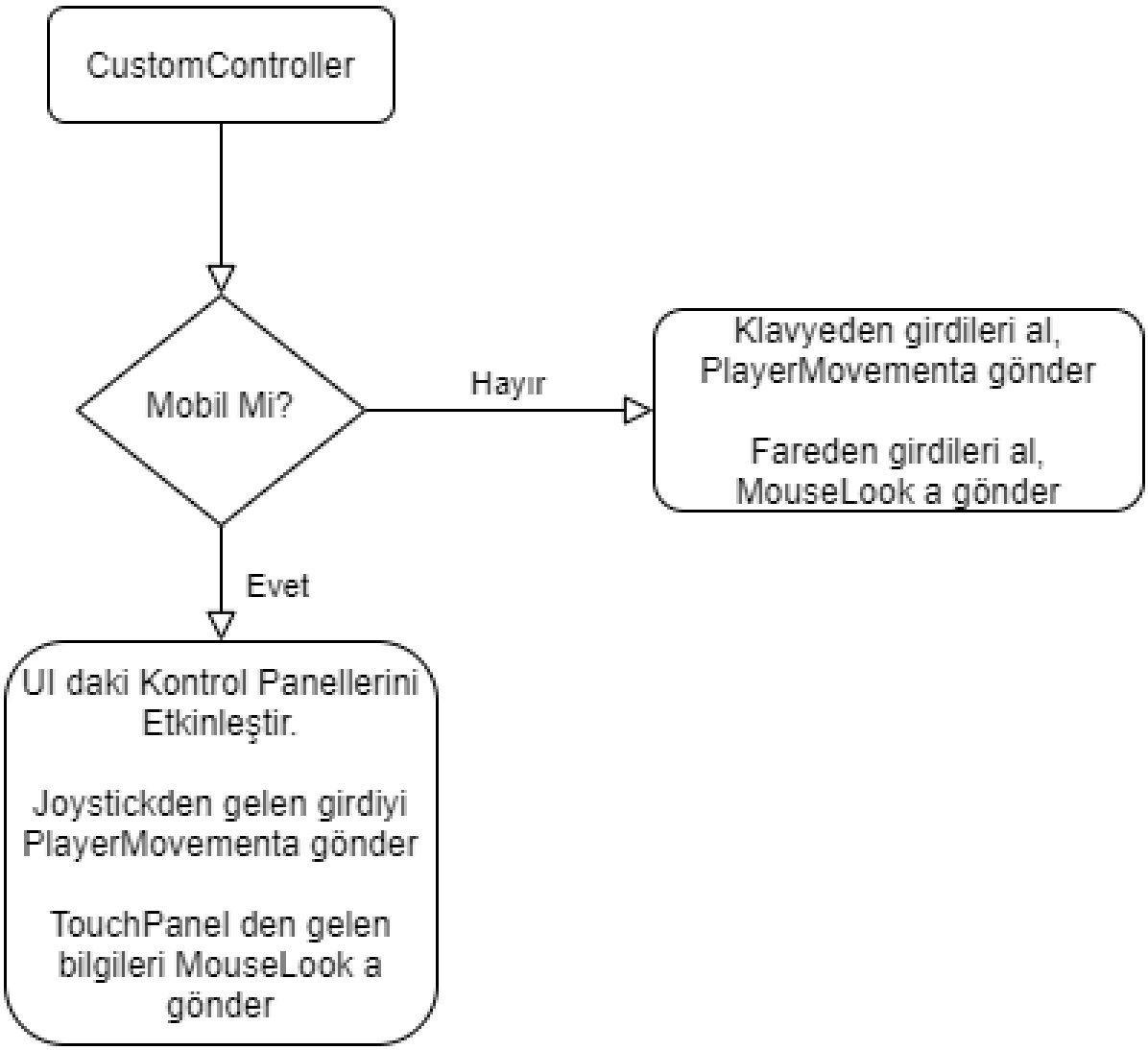
2.1.1. Custom Controller

Custom controllerin görevi oyunun çalıştığı platformun tespitini yaparak buna göre kullanıcının girdilerinin nasıl olacağını belirlemektir. Kullanıcı mobil bir cihazda oyunu oynuyor ise custom controller UI daki Controls kısmını aktive eder, böylelikle kullanıcıdan girdiler alınırken(hareket et, objeyi al, envanteri aç, günlüğü aç, kameranın açısını değiştir) UI kısmının kullanılmasını sağlar. Mobil kontrol UI'ı Şekil 2.4 de görüldüğü gibidir.

Custom Controller, PlayerMovement ve MouseLook Scriptlerine gelen girdiyi, geldiği yere göre yönlendirir. Joystick değerleri [-1, 1] aralığındadır, aynı şekilde klavyeden gelen değerler de bu şekildedir. Joystickde, x ve y bileşenleri bu aralıkta olan bir vektör değeri döndürülür. Bu şekilde RunAxis vektörüne değerler atanır ve PlayerMovement scriptinde bu değerler işlenir. MouseLook Scriptinde işlenen değer, mobil girdiyse bu girdiye özel ayrı bir panel olan TouchField paneli ile alınır, değil ise PC faresinden gelen değerlerdir. Akış diyagramı Şekil 2.5 deki gibidir.



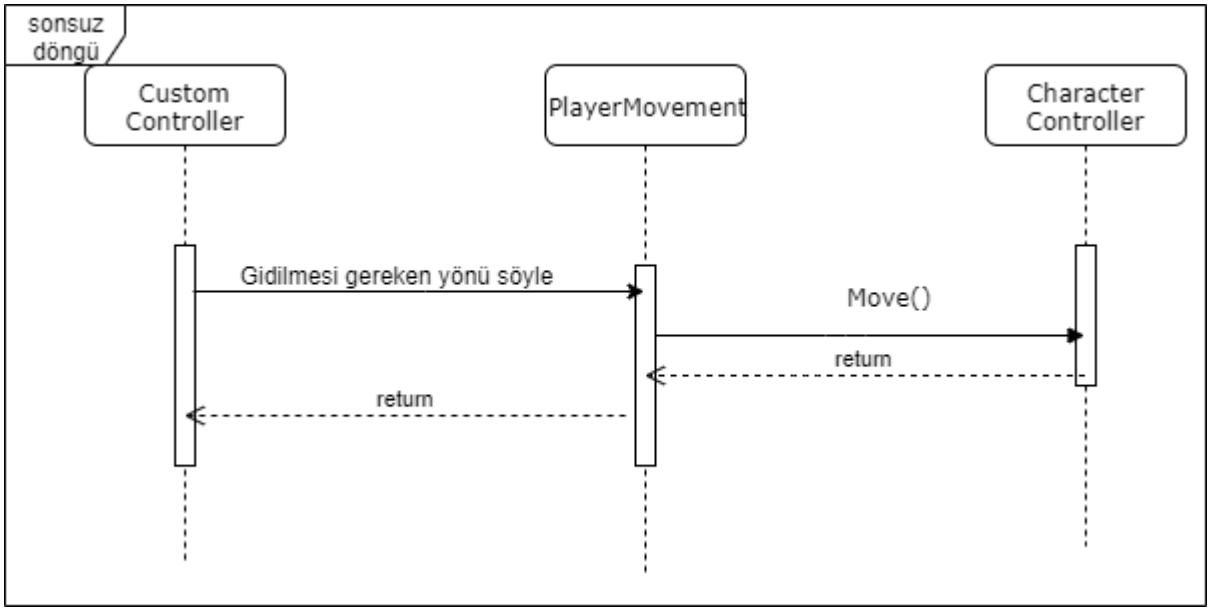
Şekil 2.4. Mobil UI



Şekil 2.5. Custom Controller Akış Diyagramı

2.1.2. PlayerMovement

Bu Script First Person Controllerin "Transform" bileşenini değiştirmekle yükümlüdür. PlayerMovement, Unitynin sağladığı CharacterController bileşenin "Move" metodu kullanılarak "Transform"un verilen RunAxis vektörü doğrultusunda yürümesini sağlamaktadır. Bu vektörün değerleri CustomController sınıfından alınmaktadır.

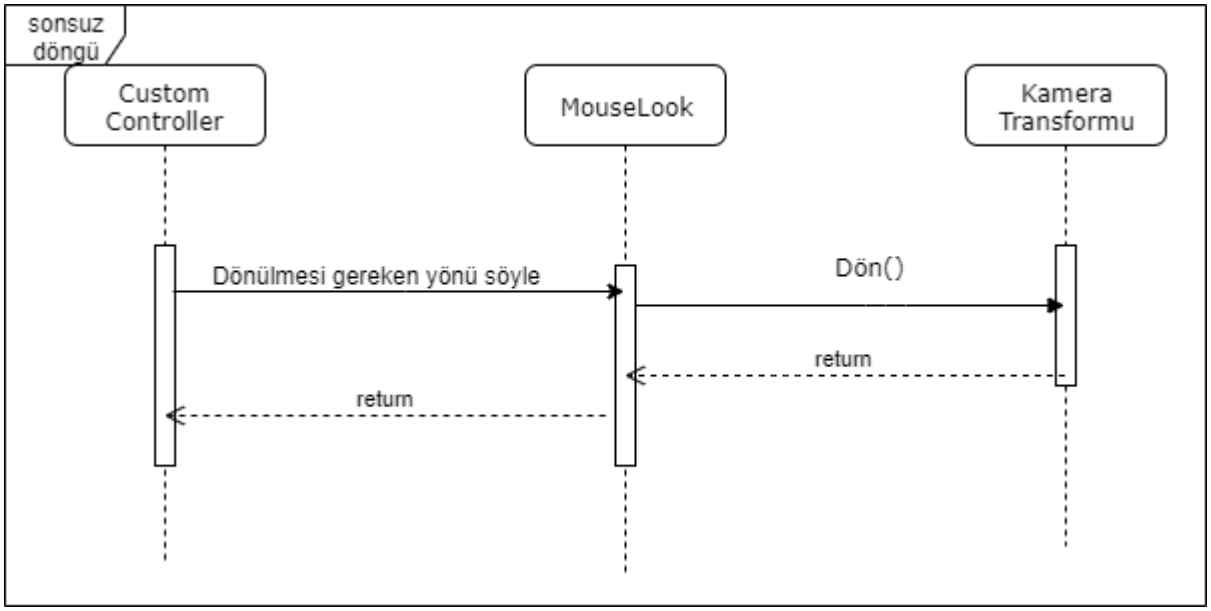


Şekil 2.6. PlayerMovement Sequence Diyagramı

2.1.3. MouseLook

Kameranın rotasyonundan sorumlu olan "Script"dir. Aynı zamanda oyuncu objesinin rotasyonunu da belirler. Kamera objesine bileşen olarak atanmıştır.

Kamera objesi, "Controller" objesinin bir içerisinde bulunduğu için, her zaman "Controller"in baktığı yöne bakar. Bu yön ise "LookAxis" den gelen x (sağ sol) değerine bağlıdır. Vector3.up (0,1,0) değerlerine sahip bir vektördür, "MouseX" bu vektörle çarpıldığı için, "Rotate" fonksiyonu First Person Controlleri sadece Y ekseninde döndürür. Kameranın X rotasyonu yukarı ve aşağı bakılmasından sorumludur. First Person Controller in insan dışı hareketler gerçekleştirmemesi için, bu X rotasyonu [-90, 90] aralığına sıkıştırılmıştır. Bu rotasyon "LookAxis" den gelen y (yukarı aşağı) değerlerine bağlıdır.



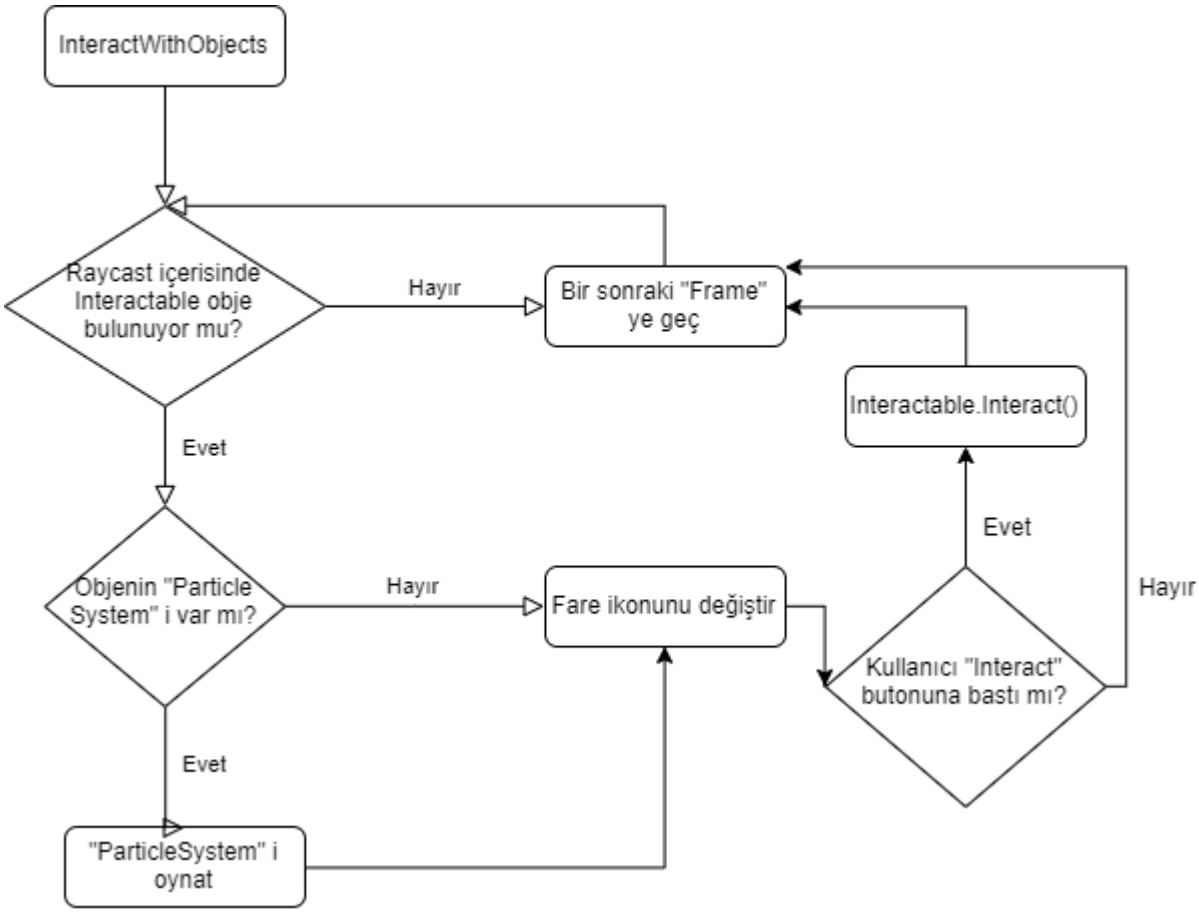
Şekil 2.7. MouseLook Sequence Diyagramı

2.1.4. InteractObjects

Objelerle etkileşimi sağlayan “Script”dir. First Person Controller kamerasından bir “Raycast” yaparak o anki “Raycast”e değen nesnenin bilgisini kaydeder. Aynı zamanda oyuncunun Envanterdeki “Item” ların kullanmasıyla oluşacak objeyi elinde tutma olayını da kontrol eder.

“Raycast” kullanılıp bulunan obje “hit” değişkenine atanır. Bu “Update” metodu içinde olduğu için bu işlem her “Frame” de gerçekleşir. Böylelikle “hit” değişkeni “Raycast” bir şeylere çarpmıyor iken “null” değerini alır.

Akış diyagramı Şekil 2.8 de verilmiştir.



Şekil 2.8. InteractObjects Akış Diyagramı

2.2. Objeler

Bu bölümde iki obje örneği üzerinden gerçeklenim anlatılmaktadır. Bu gerçeklenim her obje için aynı şekilde olduğundan böyle bir anlatıma başvurulmuştur(Objelerin aynılık durumu aşağıda yazılan “Script”leri bir bileşen olarak tutmasıdır, objelerin diğer bileşenleri farklılık gösterebilir).

2.2.1. Glowable

Her Interactable obje “Glowable” sınıfının bir alt sınıfıdır. “Glowable” sınıfı gereksinimlerde bahsedilen "Particle System” efektini oynatmadan sorumlu sınıftır.

Eğer objede “ParticleSystem” var ise, onu 1 saniyeliğine oynatıp, ardından durdurmaya yarayan bir “Script” implemente edilmiştir.

2.2.2. Interactable

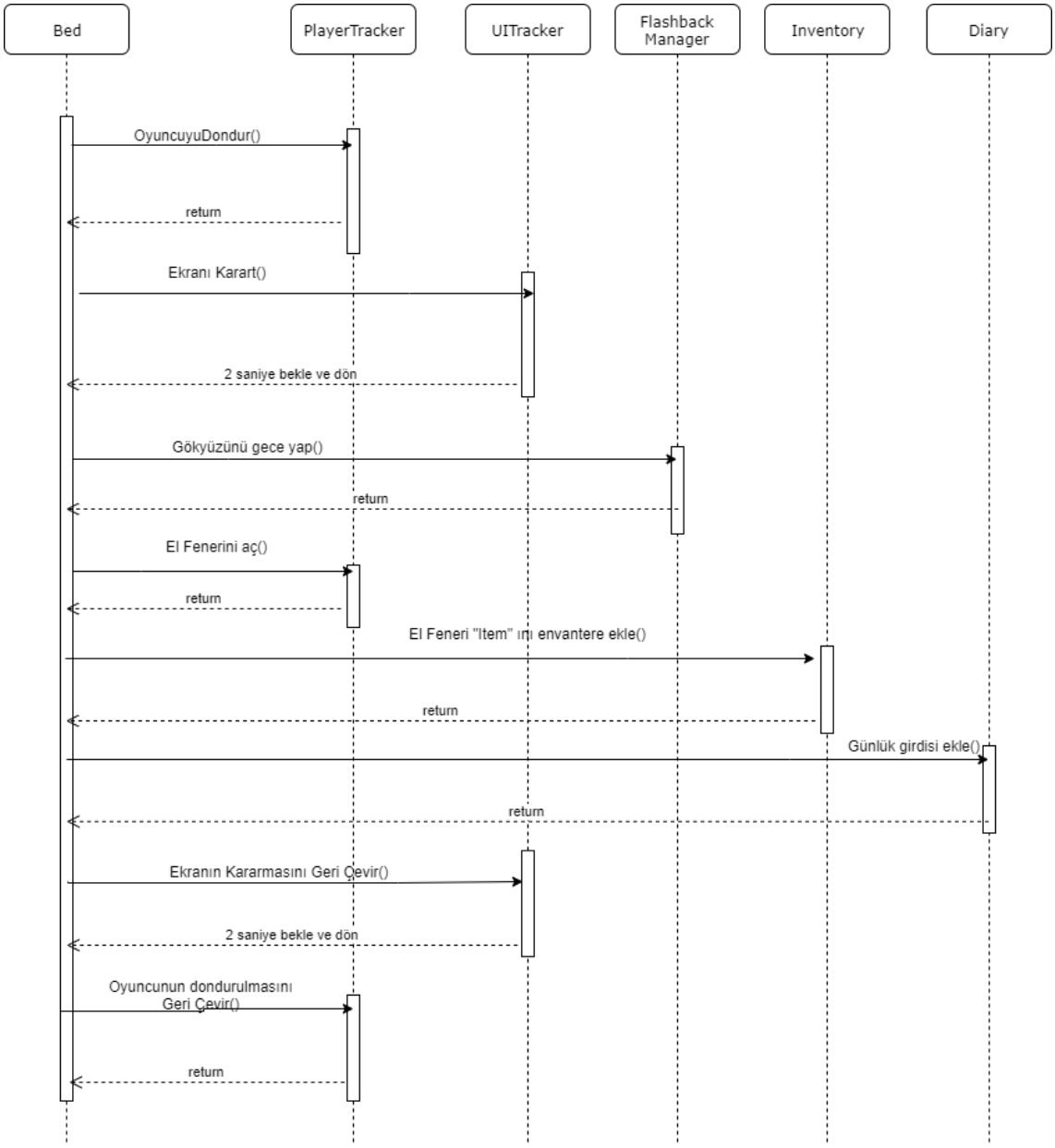
Etkileşime geçilebilir objeler içerisinde “Interactable” soyut sınıfını implemente eden sınıfları bileşen olarak barındırırlar. Şekil 2.9 da Bu sınıfı kalıtım alan "Bed" Sınıfı için Sequence diyagramı verilmiştir.

2.2.3. Pickupable

Interactable bir obje türüdür. Objenin envantere alınmasını sağlar.

2.2.4. Inventory

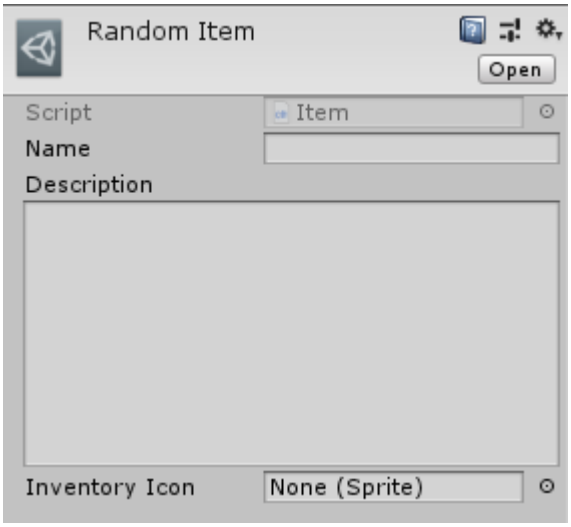
Envanter “Item”ların tutulduğu yerdir. Bu sınıf bir Singleton olarak var olabilir. Bu sınıfa ait herhangi bir değişim olduğunda, yani bir “Item” eklenildiğinde veya çıkarıldığında, bu durumdan haberdar olmak amaçlı bir delegate metodu yazılmıştır. Bunun sebebi gerekli UI güncellemelerini yapmak amaçlı olmuştur. “Inventory” sınıfı aslında bir “Item” listesinden ibarettir.



Şekil 2.9. Bed Sınıfı Sequence Diyagramı

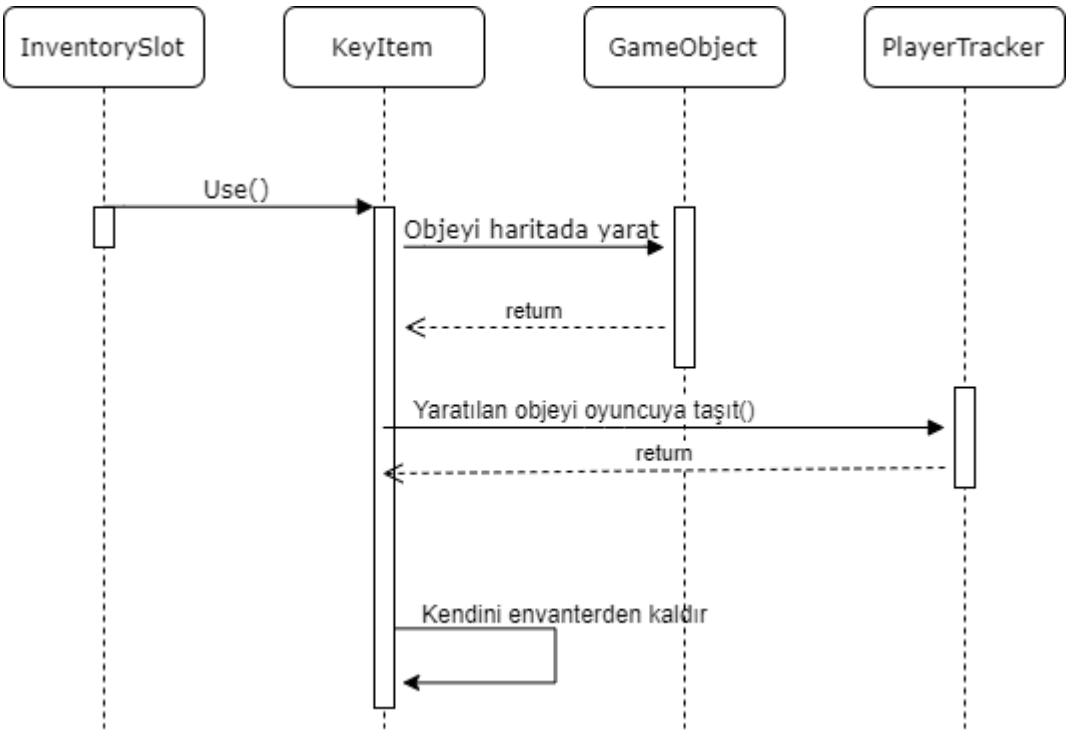
2.2.4.1. Item

“Item” oluşturulabilen bir “ScriptableObject”(Şekil 2.10) dir. Her “Item” bir "InventorySlot” yerine denk gelir.



Şekil 2.10. Scriptable Object

Bu sınıftan üretilen "Key" objesinin Sequence Diyagramı Şekil 2.11 de verilmiştir.



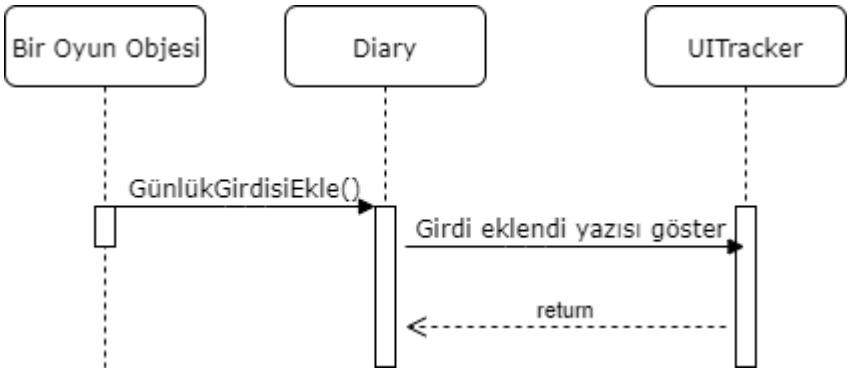
Şekil 2.11. Key Sequence Diyagramı

2.2.5. InventorySlot

InventorySlot Scripti içerisinde Item ve Icon barındırır. Bu obje UI daki butonların çağırdığı objedir. InventoryUI ı basit olarak butonlardan oluşan bir panel olarak düşünülebilir. Bu butonlara tıklandığı zaman, InventorySlot Scriptindeki "UseItem" metodu çağırılır.

2.2.6. Diary

Günlük Envanter mantığı ile aynı mantıkla tasarlanmıştır. "Item" yerine "DiaryEntry" tutarlar. Farkları "Item" bir "ScriptableObject" olup "DiaryEntry" sadece bir sınıftır. Şekil 2.12 de akış diyagramı verilmiştir.



Şekil 2.12. Diary Sequence Diyagramı

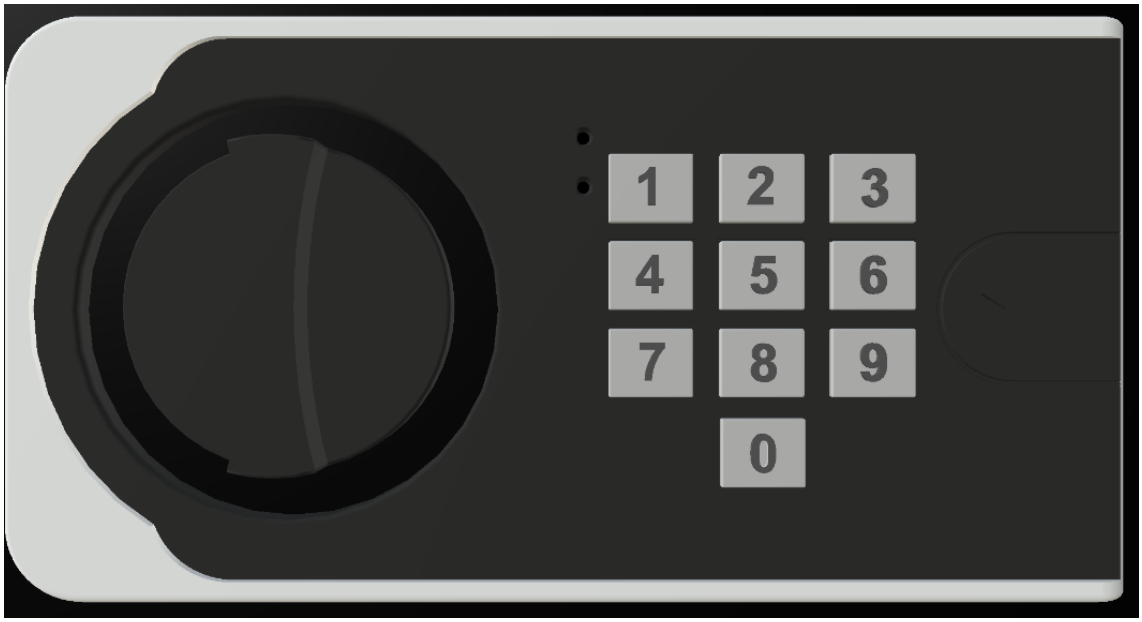
2.2.7. Event

Eventler Unitynin "Collider" bileşeni kullanarak gerçekleştirilmiştir. "Event" bileşeni görünmez bir küp olarak düşünülebilir. Oyuncu ne zaman bu küpün içerisini girerse, "Event" ateşlenir ve kendi görevini tamamladıktan birkaç saniye sonra imha olur. Eventler bir kez çalışırlar. Bu "Event"lere "OnEnterEvent" ismi verilmiştir. Bu sınıftan kalıtım alan "CrawlerJumpScare" eventinin akış diyagramı Şekil 2.13 de verilmiştir.

Şekil 2.13. CrawlerJumpScare Akış Diyagramı

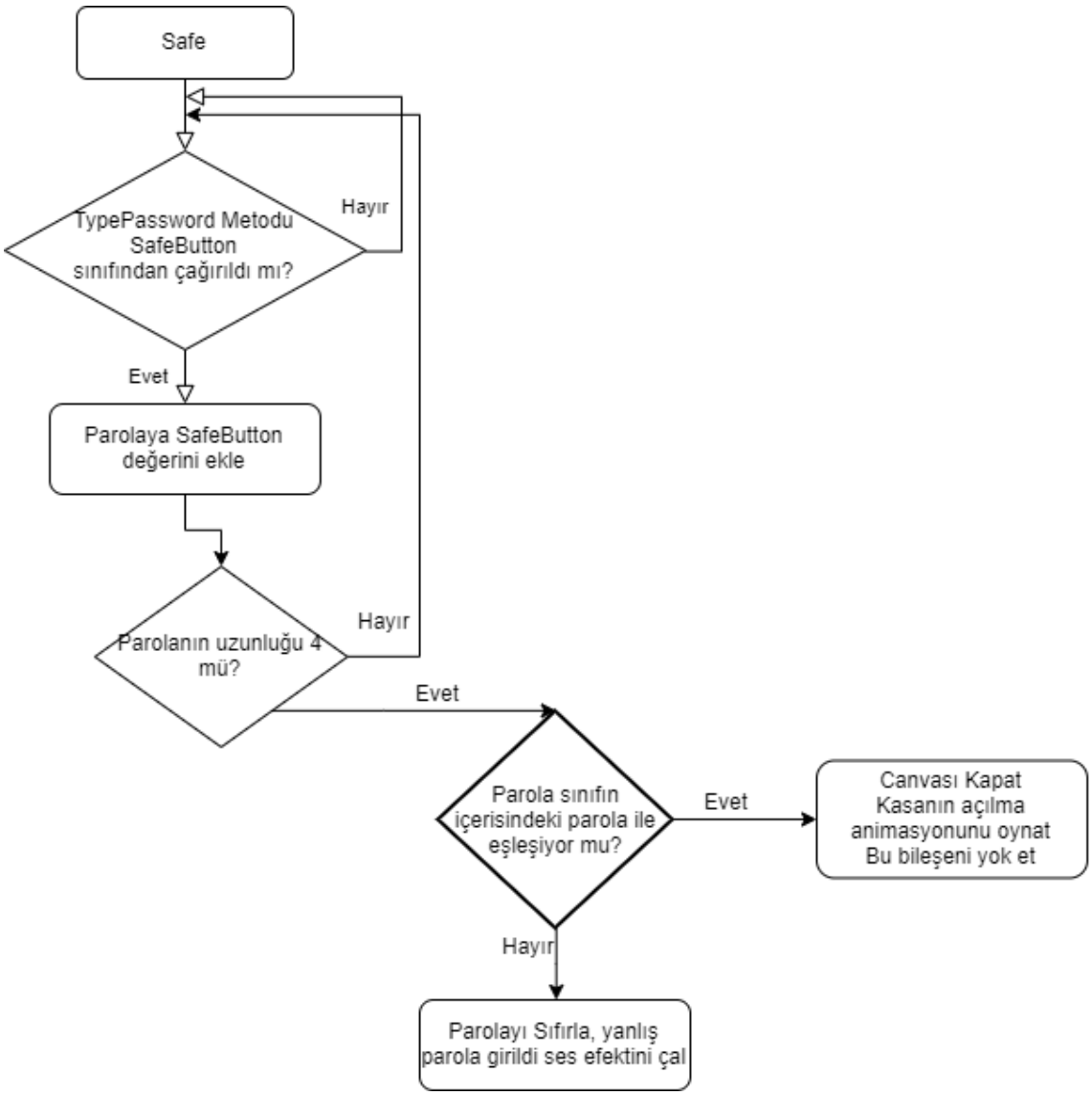
2.2.8. Safe

Kasa objesinin gerçekleştirilmesi şu şekilde olmuştur; Kasa objesi bir Interactable olup, Interact edildiği zaman, bir UI "Canvas" oluşturur. Bu Canvas Şekil 2.14 deki resmi içerir.



Şekil 2.14. Kasa

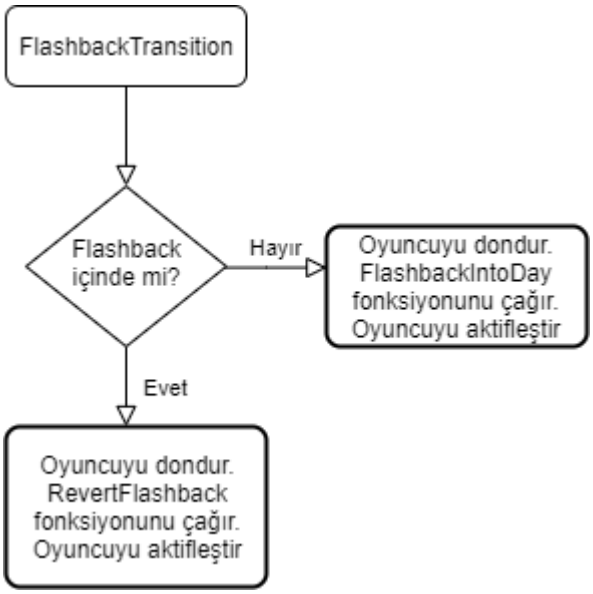
Şekil 2.14 deki resimin üzerine, resimdeki butonların olduğu pozisyonlara, Unity UI Button objesi eklenmiştir. Bu buton objeleri "Safe" scriptiyle iletişim kurarak kasanın parolasını girer. Parola doğru ise "Canvas" kapatılır, kasanın açılma animasyonu oynatılır. Akış diyagramı Şekil 2.15 de verilmiştir.



Şekil 2.15. Kasa Akış Diyagramı

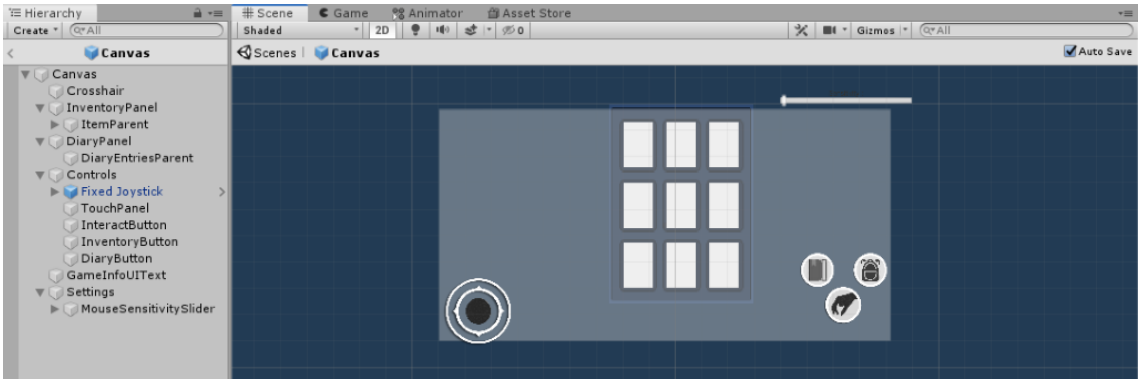
2.2.9. FlashbackTransition

Oyunun Gece/Gündüz arası geçişlerinden sorumlu bileşen. Şekil 2.16 da akış diyagramı verilmiştir.



Şekil 2.16. FlashbackTransition Akış Diyagramı

2.3. UI



Şekil 2.17. UI Implementasyonu

UI oyun başlangıcında tamamen kapalıdır. UI platform belirlendikten sonra buna göre adapte edilir. Eğer platform mobil ise, "Controls" objesi aktive edilir ve ekrana mobil kontroller gelir. Eğer platform PC ise, ekranda bir şey gözükmez

(CustomController UI ı kontrol etmekle yükümlüdür, bu işlemleri Start metodunda gerçekleştirir). Şekil 2.17 de UI verilmiştir.

2.3.1. InventoryUI

Bu "Script" Envanter "Singleton"u güncellendiği zaman kullanıcı ara yüzünün güncellenmesini sağlar.

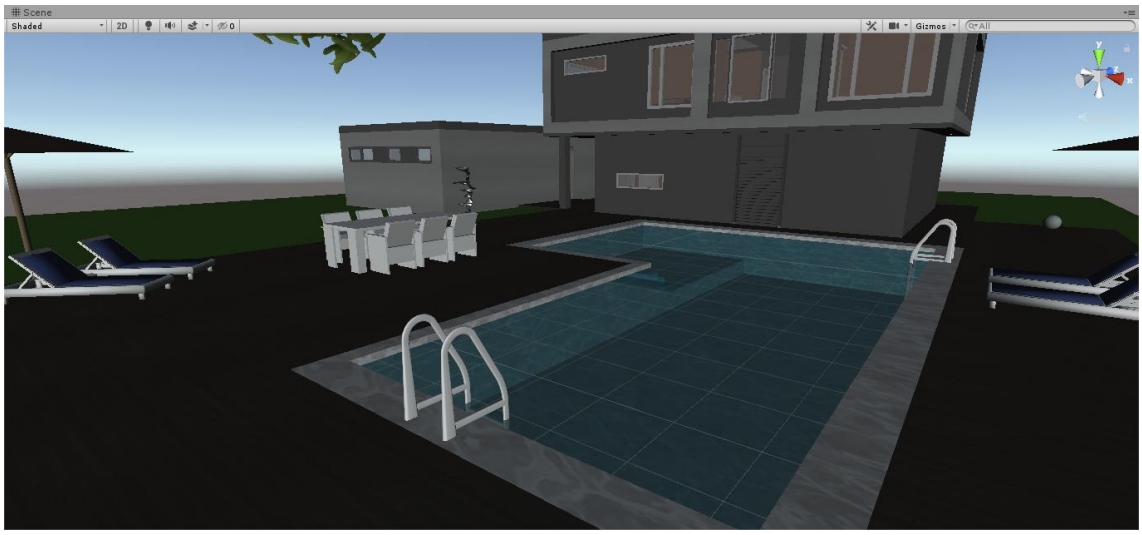
Bu script, Start metodunda, "Inventory" singletonun delege metoduna abone olur. Bu, Inventory singletonu değiştiği an(Item eklendiği veya çıkarıldığı zaman), UpdateUI metodunun çağırılmasını sağlar. UpdateUI metodu, "Inventory" Panelinde gerekli değişimleri yaparak, panelde bulunan "Button" ların buna göre aktif veya deaktif olmasını sağlar. Örnek olarak bir "Key" "Item" ı Inventory ye alınmışsa, UpdateUI çalışarak, panelde "Key" için bir resim oluşturup bu resime tıklanıldığı zaman, "Key" in use metodunun çağırılmasını sağlar.

2.3.2. DiaryUI

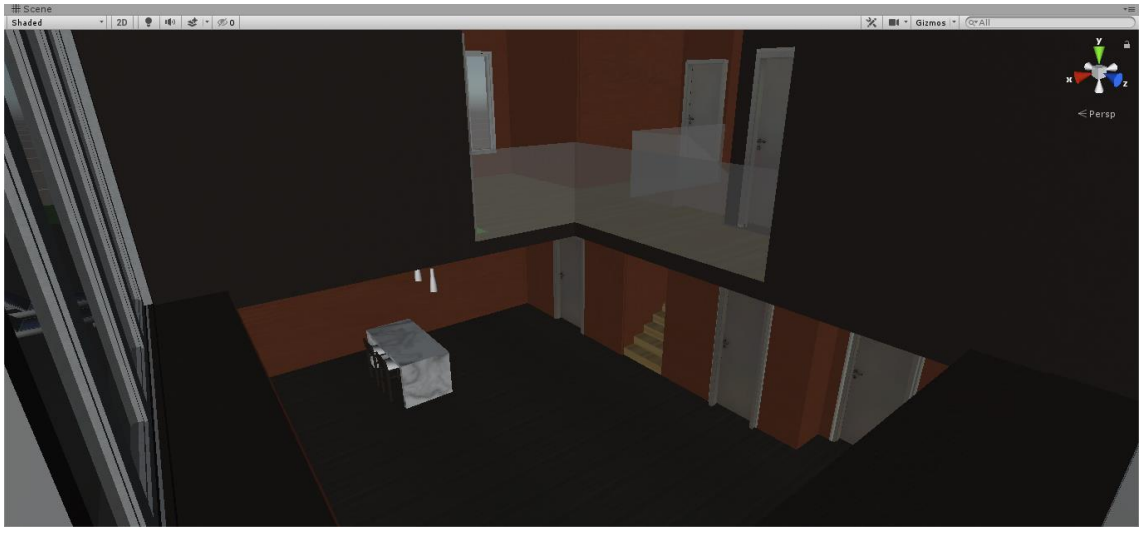
"InventoryUI" sınıfıyla çok benzer işlemleri görür."Diary" de bulunan en son girdiyi alıp, "Text" olarak "UI"a yazar.

2.4. Harita(Scene) İmplementasyonu

Haritanın tasarımı aşağıda şekil olarak verilmiştir

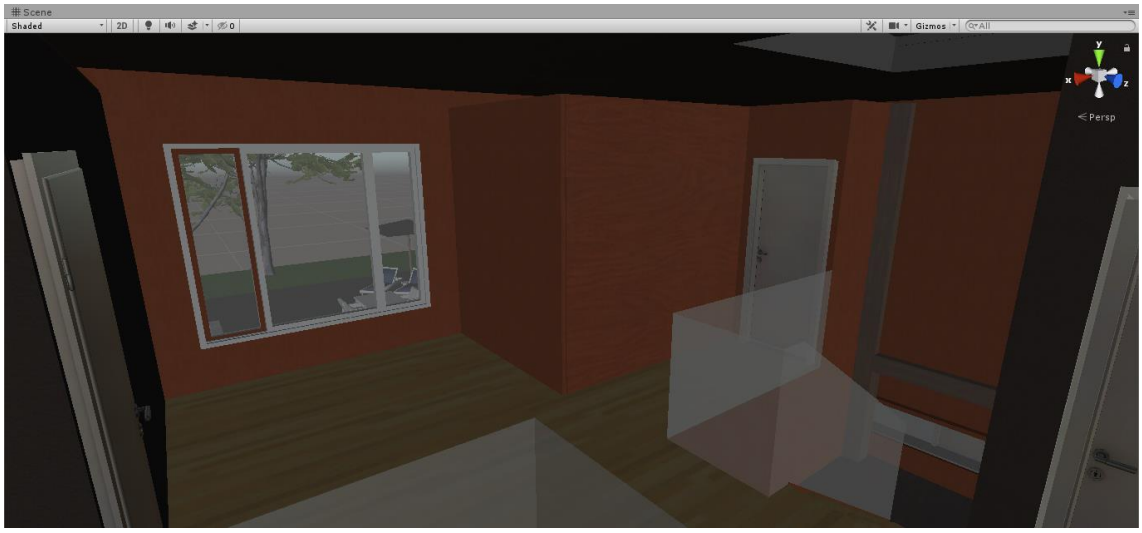


Şekil 2.18. Ev 1

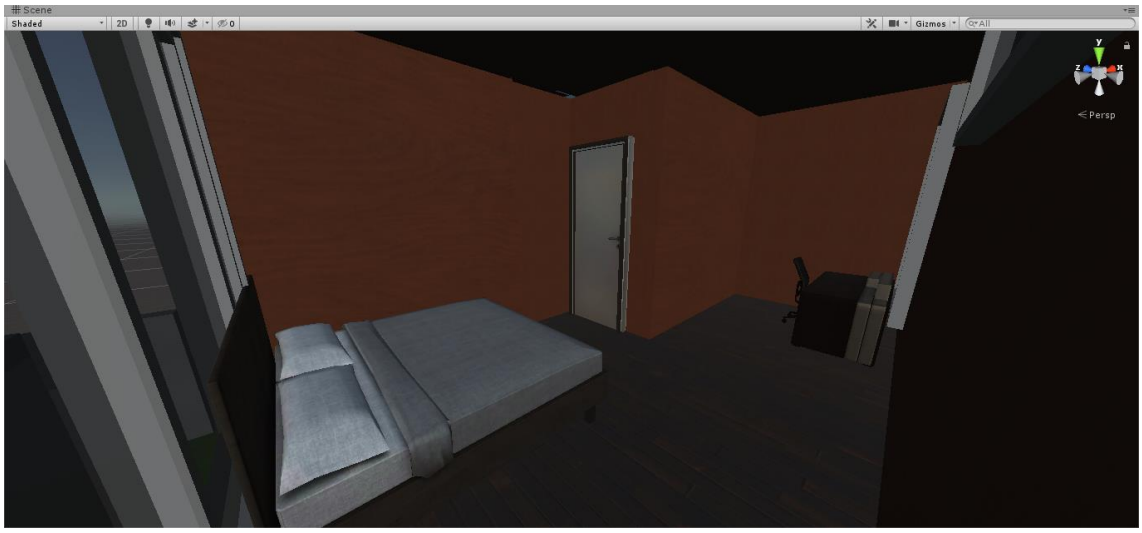


Şekil 2.19. Ev 2

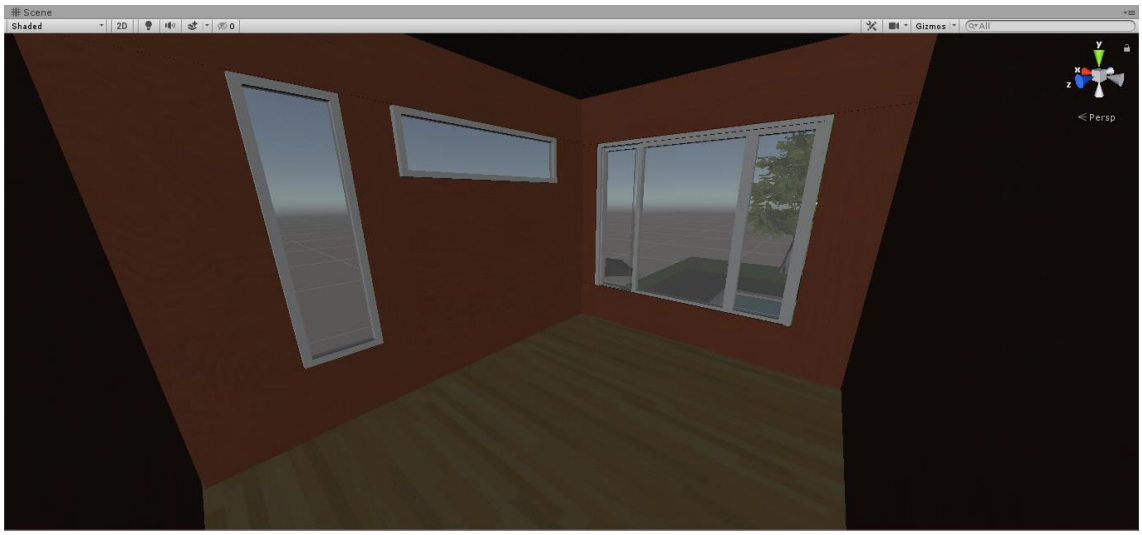
Şekil 2.20. Ev 3



Şekil 2.21. Ev 4



Şekil 2.22. Ev 5



Şekil 2.23. Ev 6

Şekil 2.24. Ev 7

3. BÖLÜM

TARTIŞMA, SONUÇ ve ÖNERİLER

3.1. Tartışma, Sonuç ve Öneriler

Oyunun yapılmasındaki en büyük engelin, hayal edilen oyun için gerekli modellerin ve modellerin animasyonlarının tam olarak mevcut olmayışı olmuştur. Böyle modellerin mevcut olmaması nedeniyle, proje için modellerin özel olarak oluşturulması gerekli görülmüştür. Fakat bu gerek projenin süresini uzatma gibi bir risk taşıdığı için, hazır modeller kullanılmıştır. Bu nedenle hayal edilen senaryo, bu senaryoya uyumlu bir ortam hazırlanmakta güçlük çekilmiştir.

Oyun için aşılması gereken bir başka problem ise performans problemidir. Oyundaki kullanılan ışıklar gerçek zamanlı ışıklar olduğu için, bu ışıkların hesaplanmasının zorluğu, mobil cihazlarda performans kayıplarına yol açmıştır, bu nedenle oyunun daha düşük sistem özelliklerine sahip platformlarda çalışamama problemiyle karşılaşmıştır. Bu problem, Android için 7.0 ve sonraki sürümleri desteklemek zorunda bırakmıştır. Bu nedenle Googleın yayınladığı Android sürümleri kullanım oranlarına bakarak(Şekil 3.1), oyunun yaklaşık olarak piyasada bulunan cihazların %42 sinde çok düşük bir performansla çalışacağı görülmüştür.

Problem uygulama ticari bir amaçla yazılmadığı için göz ardı edilebilir düzeydedir. Oyunun geldiği aşama 2 aylık çalışma sonucu kabul edilebilir düzeydedir. Oyunun sahip olduğu mekanikler bir diğer FPS projesi içinde kullanılmaya müsait şekilde yazılmış olarak görülmektedir.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Şekil 3.1. Android Sürümleri Kullanım Oranları

KAYNAKLAR

1. Digi-capital, 2018. Games software/hardware $165B+ in 2018, $230B+ in 5 years, record $2B+ investment last year.

2. Karahisar, T. Türkiye'de Dijital Oyun Sektörünün Durumu.

3. Unity Personal Usage.