### MESLEK YÜKSEK OKULU

**MOBİL TEKNOLOJİ BÖLÜMÜ**



**UNİTY**

**RABBİT JUMP**

**BİTİRME PROJESİ**

**METEHAN GÜMÜŞ**

**Danışman**

**Ogr. Gor. Nilgün İncereis**

**Mayıs 2020**

**Program: Mobil Teknolojileri**

**UNİTY**

**RABBİT JUMP**

**BİTİRME PROJESİ**

**METEHAN GÜMÜŞ**

**tarafından OKAN ÜNİVERSİTESİ**

**Mobil Teknoloji Bölümüne Önlisans**

**derecesi şartını sağlamak için sunulmuştur.**

**Onaylayan:**

**Danışman**

**Ogr. Gor. Nilgün İncereis**

**Üye Üye**

**Üye Üye**

**Mayıs 2020**

**Program: Mobil Teknoloji**

**ÖNSÖZ**

“Web Tabanlı Oto Galeri Otomasyon Sistemi” konulu bu çalışma, Okan Üniversitesi, Mobil Teknoloji Bölümü’nde “Bitirme Projesi” olarak hazırlanmıştır.

Metehan Gümüş

**İÇİNDEKİLER**

1. [GENEL BİLGİLER 6](#_bookmark0)
   1. [Unity (OBJECT RELATİONAL MAPPİNG) 6](#_bookmark1)
      1. [UNİTY PLATFORM PRENSİBİ 6](#_bookmark2)
      2. [UNİTY AVANTAJLARI 6](#_bookmark3)
      3. [UNİTY PAZARLAMA 7](#_bookmark4)
      4. [UNİTY HAKKINDA 7](#_bookmark4)
   2. [UNİTY 3D 8](#_bookmark7)
   3. [UNİTY ARAYÜZÜNÜ TANIYALIM 8](#_bookmark8)
      1. [GAME OBJECT OLUŞTURMAK 9](#_bookmark9)
      2. [SCENE PANELİNDE GEZME 9](#_bookmark10)
      3. [GAME(OYUN) PANELİ 9](#_bookmark11)
      4. [ASSETLERİN KULLANIMI 9](#_bookmark12)
         1. [KLONLAMA(DUPLİCATE) 10](#_bookmark13)
         2. [UNİTY DE SCRİPTLEME 10](#_bookmark14)
         3. [DEĞİŞKENLERE(VARİABLE) DEĞER ATAMAK 13](#_bookmark15)
         4. [COMPONENTLERE ERİŞMEK 14](#_bookmark16)
         5. [OBJE OLUŞTURMAK(INSTANTİATE) 15](#_bookmark17)
            1. [DEBUG İŞLEMİ 16](#_bookmark18)
            2. [YAYGIN OLARAK KULLANILAN FONKSİYONLAR 17](#_bookmark21)
2. [PROJE BİLGİLERİ 18](#_bookmark56)
   1. [GENEL BİLGİLER 18](#_bookmark57)
      1. [OYUNUN AMACI 18](#_bookmark58)
         1. [HEDEF KİTLE 18](#_2.1.1.1_HEDEF_KİTLE)
   2. [PROJE YAPILARI 18](#_bookmark60)
      1. [OYUN TÜRÜ 18](#_bookmark61)
      2. [OYUN PLATFORMU 18](#_bookmark62)
      3. [OYUN NASIL OYNANIR 18](#_OYUN_NASIL_OYNANIR)
      4. [OYUN İÇİN KULLANILAN RESİMLER 19](#_OYUN_İÇİN_KULLANILAN)
   3. [KODLAR 20](#_bookmark66)
3. [KAYNAKÇA 24](#_KAYNAKÇA)
4. [GİTHUB LİNK 24](#_GİTHUB_LİNK)

# GENEL BİLGİLER

## Unity (Oyun Motoru)

**Unity**, öncelikli olarak [bilgisayarlar](https://tr.wikipedia.org/wiki/Bilgisayar), [konsollar](https://tr.wikipedia.org/wiki/Video_oyunu_konsolu) ve [mobil cihazlar](https://tr.wikipedia.org/wiki/Mobil_cihaz) için [video oyunları](https://tr.wikipedia.org/wiki/Video_oyunu) ve [simülasyonları](https://tr.wikipedia.org/wiki/Sim%C3%BClasyon) geliştirmek için kullanılan ve [Unity Technologies](https://tr.wikipedia.org/w/index.php?title=Unity_Technologies&action=edit&redlink=1) tarafından geliştirilen [çapraz platform](https://tr.wikipedia.org/wiki/%C3%87apraz_platform_yaz%C4%B1l%C4%B1mlar%C4%B1) bir [oyun motorudur](https://tr.wikipedia.org/wiki/Oyun_motoru). İlk kez yalnızca Apple'ın 2005'teki [Worldwide Developers Conference](https://tr.wikipedia.org/wiki/Apple_Worldwide_Developers_Conference)'da [OS X](https://tr.wikipedia.org/wiki/MacOS) için ilan edildi, bu tarihten itibaren 27 platformu hedeflemek üzere genişletildi.

*Unity*'nin altı ana sürümü yayınlandı. 2006 [WWDC](https://tr.wikipedia.org/wiki/Apple_Worldwide_Developers_Conference) gösterisinde Apple, *Unity*'yi En İyi Mac OS X Grafik Kullanımı kategorisinde birinci olarak seçti.

Özet

*Unity*; [2B](https://tr.wikipedia.org/wiki/%C4%B0ki_boyutlu_uzay) ve [3B](https://tr.wikipedia.org/wiki/3B) grafikleri, sürükleyip bırakma işlevselliğini ve [C#](https://tr.wikipedia.org/wiki/C) ile komut dosyası yazmayı destekleyen çok yönlü bir oyunmotorudur.İkiprogramlama dili desteklenmektedir: *Unity* 5'in yayınlanmasındansonra kullanımı tavsiye edilmeyen Boo ve Unity 2017.1'in piyasaya sürülmesinden sonra Ağustos 2017'de kullanımı

tavsiye edilmeyen UnityScript. UnityScript, [sözdizimsel](https://tr.wikipedia.org/wiki/S%C3%B6zdizim) olarak [JavaScript](https://tr.wikipedia.org/wiki/JavaScript)'e benzeyen tescilli bir [betik dilidir](https://tr.wikipedia.org/wiki/Betik_dili).

### UNİTY PLATFORM PRENSİBİ

*Unity*, oyunları birden fazla platformda hedefleyebilme özelliğiyle dikkat çekmektedir. Şu anda desteklenen platformlar [Android](https://tr.wikipedia.org/wiki/Android_(i%C5%9Fletim_sistemi)), [Android TV](https://tr.wikipedia.org/wiki/Android_TV), [Facebook Gameroom](https://tr.wikipedia.org/w/index.php?title=Facebook_Gameroom&action=edit&redlink=1), [Fire OS](https://tr.wikipedia.org/w/index.php?title=Fire_OS&action=edit&redlink=1), [GearVR](https://tr.wikipedia.org/w/index.php?title=Gear_VR&action=edit&redlink=1), [GoogleCardboard](https://tr.wikipedia.org/wiki/Google_Cardboard), [GoogleDaydream](https://tr.wikipedia.org/wiki/Google_Daydream), [HTCVive](https://tr.wikipedia.org/wiki/HTC_Vive), [iOS](https://tr.wikipedia.org/wiki/%C4%B0OS), [Linux](https://tr.wikipedia.org/wiki/Linux), [macOS](https://tr.wikipedia.org/wiki/MacOS), [Microsoft,HoloLens](https://tr.wikipedia.org/wiki/Microsoft_HoloLens), [Nintendo3DS](https://tr.wikipedia.org/wiki/Nintendo_3DS) ailesi,[[11]](https://tr.wikipedia.org/wiki/Unity_(oyun_motoru)#cite_note-UnityTech-announces-NN3DSsupport-11)[[12]](https://tr.wikipedia.org/wiki/Unity_(oyun_motoru)#cite_note-IGNcitesUnityNN3DSsupport-12)[[13]](https://tr.wikipedia.org/wiki/Unity_(oyun_motoru)#cite_note-unity-europe-announces-new-3ds-details-13) [NintendoSwitch](https://tr.wikipedia.org/wiki/Nintendo_Switch), [OculusRift](https://tr.wikipedia.org/wiki/Oculus_Rift),

[PlayStation4](https://tr.wikipedia.org/wiki/PlayStation_4), [PlayStationVita](https://tr.wikipedia.org/wiki/PlayStation_Vita), [PlayStationVR](https://tr.wikipedia.org/wiki/PlayStation_VR), [SamsungSmartTV](https://tr.wikipedia.org/w/index.php?title=Samsung_Smart_TV&action=edit&redlink=1), [Tizen](https://tr.wikipedia.org/wiki/Tizen), [tvOS](https://tr.wikipedia.org/wiki/TvOS), [WebGL](https://tr.wikipedia.org/wiki/WebGL), [Wii U](https://tr.wikipedia.org/wiki/Wii_U), [Windows](https://tr.wikipedia.org/wiki/Microsoft_Windows), [Windows Phone](https://tr.wikipedia.org/wiki/Windows_Phone), [Windows Store](https://tr.wikipedia.org/wiki/Windows_Store) ve [Xbox One](https://tr.wikipedia.org/wiki/Xbox_One). *Unity* daha önce kendi Unity Web Player'ını da içeren 7 platformu destekledi. Unity Web Player, yalnızca Windows ve OS X'de desteklenen, WebGL çıktıktan sonra kullanımı sonlandırılan bir tarayıcı eklentisiydi.

### UNİTY AVANTAJLARI

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

### UNİTY PAZARLAMA

Unity Technologies Japonya, 16 Aralık 2013'te Unity-chan (ユニティちゃん *Yuniti-chan*) adlı gerçek adı Kohaku Ōtori (大鳥 こはく *Ōtori Kohaku*) olan (Asuka Kakumoto tarafından seslendirilen) resmi bir maskot karakterinin yeni bir ekran görüntüleri ortaya koydu. Karakter, [Tokyo Big Sight](https://tr.wikipedia.org/wiki/Tokyo_Big_Sight)'ta Comic Market 85 etkinliğinde temalı eşyaların dağıtılacağı ve ses sanatçısının yer alacağı 29 Aralık - 31 Aralık tarihleri ​​arasındaki sergilendi. Karakterin ilişkili oyun verileri 2014 baharında çıkacaktı. Bu karakter, Unity Technologies Japonya tasarımcısı "ntny" tarafından açık kaynak kodlu bir kadın kahraman karakter olarak tasarlandı. Şirket, belirli lisanslar altında ikincil projelerde Unity-chan ve ilgili karakterlerin kullanılmasına izin vermektedir. Örneğin, Unity-chan, [*Runbow*](https://tr.wikipedia.org/w/index.php?title=Runbow&action=edit&redlink=1)'da oynanabilir bir karakter olarak görülebilir. Karakterin popülaritesi, [VOCALOID](https://tr.wikipedia.org/wiki/Vocaloid) için kendi ses kütüphanesi ve VOCALOID ile Unity adlı Unity Motoru 5.0 sürümüyle çalışmak üzere tasarlanmış VOCALOID'in özel bir uyarlaması da dahil olmak üzere VOCALOID uyarlamalarında ortaya çıkmasını sağladı.

### UNİTY HAKKINDA

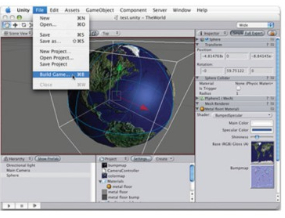
**Unity** oyun yapımında kullanılan ve bir çok platforma oyun yapmanıza olanak sağlayan bir oyun motorudur. 2D ve 3D oyunlar yapabileceğiniz **unity** motorunu isterseniz ücretsiz sürümünü isterseniz de daha fazla olanak sağlayan ücretli pro sürümü kullanabilirsiniz.

## UNİTY 3D

Unity 3D, bir oyun motorudur. Oyun ve bilişim dünyasına getirdiği en önemli yenilik, gelişmiş özelliklere sahip 3 boyutlu oyunların bilgisayara kurulmadan oynanmasını sağlamak olmuştur.  
Unity 3D motorunu kullanan oyunlar, Unity Web Player eklentisi sayesinde hiçbir kurulum işlemi olmadan web tarayıcı üzerinden çalışabilmektedir.  
Bu yönü sayesinde oyunlar bilgisayara daha az yüklenirken bir oyuncuyu yeni bilgisayar alma zahmetinden kurtarmış olur.  
Öte yandan oyun üreticileri, oyunu kendi sunucularından oynattıkları için korsan oyuna çare bulmuş olur. Unity 3D motoru ile yazılmış oyunlara örnek olarak Battlestar Galactica Online verilebilir..

Unity'nin oyun yapımcılarına sağladığı başka bir kolaylık da Unity ile geliştirilen bir oyunun herhangi bir altyapı değişikliğine gerek olmadan farklı platformlara (PC, Mac, Web, iOS, Android, Windows Phone) uygun olarak derlenebilmesidir.

## UNİTY ARAYÜZÜNÜ TANIYALIM



**Tutorialin amacı**

Eğer Unity’e yeni başlıyorsanız bu, ilk

okuduğunuz tutorial olmalı. Bu tutorial Unity

**1.** Tutorialin amacı

arayüzünün önemli parçalarını anlatacak ve

**2.** Arayüz düzeni (Layout)

nasıl bir GameObject oluşturup onu 3 boyutlu

**3.** GameObject’lerin bulunması

uzayda konumlandıracağınızı gösterecek.

**4.** GameObject oluşturmak

Unity’nin arayüzünü tanımak için basit bir

**5.** Scene panelinde gezinmek

sahne (scene) oluşturacağız ve içerisinde bir

**6.** GameObject’leri oynatmak

karakterin yürüyebilmesini sağlayacağız.

**Arayüz düzeni (Layout)**

1. **Scene (Sahne) Paneli**- GameObject’leri (oyun

objeleri) konumlandırmak için kullanılır.

2. **Game (Oyun) Paneli**- oyunun çalıştırıldığında

nasıl görüneceğine buradan bakılır.

3. **Hierarchy (Hiyerarşi) Paneli**- o anki sahnede

yer alan tüm objelerin listesini tutar.

4. **Project (Proje) Paneli**- Kullanıma hazır tüm

assetleri barındırır. (tıpkı bir palet gibi)

5. **Inspector Paneli**- O an seçili olan

GameObject’in özelliklerini gösterir

### GAME OBJECT OLUŞTURMAK

- Game Object->Create Other->Plane yolunu izleyin, böylece sahnenizde bir plane objesi (2

boyutlu yüzey) oluşacak. Bu yüzeyin üzerinde karakterinizi yürütebilirsiniz.

- Game Object->Create Other->Cube yolunu izleyerek bir küp oluşturun.

- Game Object->Create Other->Point Light yolunu izleyerek noktasal bir ışık kaynağı

oluşturun.

### SCENE PANELİNDE GEZME

boyutlu objelerinize istediğiniz bir açıdan şu yöntemlerle bakabilirsiniz:

1. **Rotating**(Kamerayı Döndürmek) (ALT+Sol Mouse Tuşu)

2. **Panning**(Sahnede Gezinmek)(ALT+Orta Mouse Tuşu (Basılı tutun, yuvarlamayın.))

3. **Zooming**(Yakınlaştırmak)(ALT+Sağ Mouse Tuşu)

### GAME (OYUN) PANELİ

Game Paneli oyununuzun çalıştırıldığında nasıl görüneceğini görmenizi sağlar (tüm

kaplamalar, ışıklar vb. gösterilir.). Eğer oyun dünyanızı Game panelinde göremiyorsanız

bunun sebebi kameranızın doğru yöne bakmıyor oluşudur.

### Assetlerin Kullanımı

Bilgisayar oyunları 3 boyutlu modeller, kaplamalar, ses dosyaları, oyun kodları gibi pek çok

parçadan oluşurlar ve Unity’de bunlara oyunun **asset**’leri denir. Unity pek çok hazır standart

asset barındırmaktadır. (Pro versiyonunda daha fazlasını da barındırır.)

Şimdi Unity’nin hazır assetlerini kullanarak oyunumuza karakterimizi ekleyelim. Bunun için

Proje panelinden Standard Assets klasörünü genişletin, ardından içerisindeki Character

Controllers klasörünü genişletin. Bir **prefab**, belli bir düzene göre bir araya getirilmiş olan ve

tekrar tekrar kullanılabilen objeler koleksiyonuna denir. Örneğin bir mermi objesini FPS

oyunlarında bolca kullanacağınız için onu bir prefab olarak ayarlamanız oldukça mantıklıdır.

Böylece o kurşun objesini defalarca kullanabilirsiniz. Konumuza geri dönelim, şimdi Prefabs

klasörünün içerisinde First Person Controller (FPS Controller) prefabını görüyor olmalısınız

### KLONLAMA(DUPLİCATE)

Bir GameObject’i klonlamak Unity’nin en güçlü özelliklerinden biridir. Bir obje klonlanırken

aynı zamanda sahip olduğu tüm component’lerde kopyalanır. Oldukça karmaşık sahneleri

çok hızlı bir şekilde oluştururken bu yöntem vazgeçilmezdir. Şimdi sahnemize daha fazla

*FizikselKup* objesi ekleyeceğiz

### UNİTY DE SCRİPTLEME

Script yazmak (Scriptleme) Unity’nin en önemli parçalarından biridir çünkü

oyununuzu şekillenmesinde doğrudan etkilidir. Bu eğitim dosyası Unity’de JavaScript kullanarak (UnityScript) scriptlemenin temellerini gösterecektir.

Script yazarken işinizin kolay olması için syntax yapılarını renkli olarak kümelere ayıran bir script editörüne sahip olmanız iyi olacaktır. Buna örnek olarak **SubEthaEdit** editörü verilebilir.

**İsimlerdeki gramer kuralları**

Başlamadan önce, Unity’de script yazarken uyulması gereken bazı gramer kurallarına

değinmek iyi olacaktır.

**Değişkenler (Variable)**- küçük harfle başlarlar. Değişkenler oyun sırasında herhangi bir

veriyi depolamak için kullanılırlar.

**Fonksiyonlar (Function)**- büyük harfle başlarlar. Fonksiyonlar birer kod kümeleridirler ve

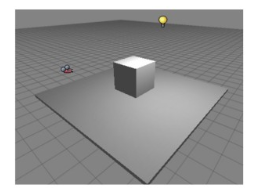
bir kere yazıldıktan sonra defalarca kullanılabilirler.

**Sınıflar (Class)**- büyük harfle başlarlar. Birden fazla fonksiyonun bir araya gelmiş hâli olarak

tanımlanabilir

### GİRDİ ALMAK (INPUT)

Unity’i başlatın.

Önce oyuncunun üzerinde yürüyeceği zemini

oluşturalım. Bizim kullanacağımız zemin

boyutlarıyla oynanmış bir küp olacak.

- Bir küp oluşturun ve Inspector panelinden

boyut ayarlarını (**Scale**) sırayla **5** - **0.1** - **5**

yapın. Artık küpümüz bir zemin gibi duracaktır.

Şimdi küpün adını Hiyerarşi panelinden ‘*Zemin*’

olarak değiştirin.

- İkinci bir küp oluşturup onu zeminin ortasına taşıyın. Eğer objelerinizi Game panelinde

göremiyorsanız Main Camera objesini objelerinizi görecek şekilde konumlandırın. Yeni

oluşturduğumuz küpü ‘*Kup1*’ olarak adlandırın.

- Bir point light oluşturup diğer objelerin biraz üzerine taşıyın, böylece objeleriniz daha rahat

gözükecek.

İlk scriptimiz

Şimdi oyunumuzu programlamaya hazırız. Main Camera objesinin konumunun oyuncudan

gelen girdilere göre değişerek oyuncunun mekanda gezinmesini sağlayacağız. Bunun için

önce girdiyi okuyan bir script yazacağız ve bu scripti Main Camera objesine atayacağız

Unity’de bir GameObject’i hareket ettirmek için onun transform’unun (konumunun) mevcut

pozisyonunu ayarlamalıyız. Bunu ise transform sınıfından **Translate** fonksiyonuyla yaparız.

Translate fonksiyonu 3 parametreye sahiptir; **x**,**y** ve **z** eksenlerinde hareket miktarı. Biz de

Main Camera objesinin ok tuşlarıyla hareket etmesini istediğimiz için yapmamız gereken şey

ilgili eksenlerin olduğu yere o yönle ilgili ok tuşunun basılıp basılmadığını kontrol eden bir kod

yazmaktır



Buradaki Input.GetAxis() fonksiyonu klavyenin sol ok tuşuna basılınca -1, sağ ok tuşuna

basılınca 1 değerini döndürür. Herhangi birine basılmamışsa 0 değerini döndürür.

Fonksiyonun y ekseni için olan 2. parametresinin değerinin 0 olduğuna dikkat edin, bunun

sebebi kamerayı dikey eksende oynatmayacağımızdır. (*x: Sağ-Sol , y: Yukarı-Aşağı , z: İleri-*

*Geri hareket yönlerini temsil eder. (Çevirmen Eklemesi)*) Horizontal ve Vertical butonların

neler olduğu Input Settings’de (Girdi Özellikleri) önceden belirlenmiştir ve Edit->Project

Settings->Input yolunu izleyerek kolayca değiştirilebilir.

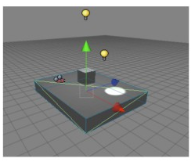
Scriptin objeye atanması

Şimdi ilk scriptimiz hazır olduğuna göre Unity’e hangi objenin bu scripte göre davranış

göstereceğini söylemeliyiz. Bunun için bu scripti istediğimiz GameObject’e atamamız yeterli

### DEĞİŞKENLERE (VARİABLE) DEĞER ATAMAK

Değişkenlerin değerlerini Unity’nin arayüzünü kullanarak değiştirmek Unity’nin en güçlü özelliklerinden biridir. Bu, değişkenlerin değerlerinin scriptlerin içerisinde atanması yerine **sürükle-bırak**yöntemiyle Unity’nin arayüzünden atanmasını sağlar. Böylece oldukça hızlı bir şekilde çalışabiliriz. Bunu yaparken tek bilmemiz gereken, değerinin arayüzden değiştirileceği değişkenlerin birer Görünür Değişken olmasıdır. Şimdi bu yönteme, oyun boyunca karaktere (MainCamera) doğru bakacak bir spot ışık oluşturarak daha yakından bakalım.



Şimdi ne yapacağımıza bakalım. Yapmak istediğimiz şey spot ışığın kamera neredeyse oraya doğru bakması. Bunu yapmak içinse Unity’nin kendi içerisinde hazır bir fonksiyon mevcut; **transform.LookAt()**. Eğer bu fonksiyonu duymadan önce ‘Bunu nasıl yapacağız?’ diye düşünüp karmaşık bir kod bekliyorduysanız size sürekli Unity **API**’yı kontrol etmenizi öneririm. Böylece yapmak istediğiniz bir şeyi yapan hâlihazırda mevcut bir fonksiyon varsa onu bulabilirsiniz. Spotlight objesinin konumuyla ilgili bir eylem yapacağımızdan dolayı da Unity API’nın ‘transform’ sekmesine bakmamız, bu fonksiyonu kolayca bulmamızı sağladı.

Script dosyasını spotlight objesine atayın, spot ışığın Inspector panelinde *TakipEt*

Component’inin olduğu yerde “*hedef*” değişkeninin görünür olduğuna dikkat edin. Spot ışık hâlâ seçiliyken Hiyerarşi panelinden Main Camera objesini tutup Inspector panelindeki “hedef” değişkeninin üzerine sürükleyin. Bu, “hedef” değişkenine Main Camera objesini atamamıza (**assign**) yarayacaktır. Yani artık spot ışık Main Camera objesini takip edecektir. Eğer spotlight’ın başka bir objeyi takip etmesini isteseydik ilgili objeyi sürüklememiz yeterli olurdu.

.

### COMPONENTLERE ERİŞMEK

Bir GameObject birden çok script veya component’e sahip olabileceği için genellikle bu

component’lerin birbirlerinin fonksiyonlarına ve değişkenlerine erişmesi söz konusu olur.

Bunun için Unity’nin **GetComponent()** adında bir fonksiyonu bulunmaktadır.

Şimdi spot ışığı objesine, ne zaman zıplama tuşuna basılırsa (Spacebar) *Kup1* objesine

bakmasını sağlayan yeni bir script atayacağız. Yani yapmak istediğimiz şey şu:

**1.** Zıplama tuşuna basılıp basılmadığını kontrol et.

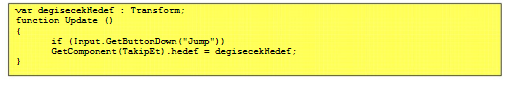
**2.** Zıplama tuşuna basıldığında spotlight’ın *Kup1* objesine bakmasını sağla.

*TakipEt* scripti “*hedef*” adında, takip edilecek objenin depolandığı bir değişkene sahipti. Şimdi

bu değişkene, zıplama tuşuna basıldığında yeni bir obje (*Kup1*) atamalıyız. Bunu kodlarla

uzun yoldan yapabiliriz (Bir sonraki ‘Bu işlemi kodlarla yapmak’ kısmına bakın.), ancak

biliyoruz ki bir değişkene arayüz vasıtasıyla değer atamak bunu yapmak için daha ideal.

- *Degistir* adında yeni bir Javascript oluşturun. Scriptin içerisine şu kodu yazın

Buradaki GetComponent() fonksiyonunda *TakipEt* scriptini parametre olarak kullandığımız

için artık onun içerisindeki “*hedef*” değişkenine ulaşabiliriz.

- *Degistir* scriptini spotlight’a ekleyin ve *Kup1* objesini *degisecekHedef* değişkenine, Inspector

paneli vasıtasıyla atayın.

- Oyunu çalıştırın. Etrafta gezinin ve ardından spacebar tuşuna basın. Spotlight artık *Kup1*

objesine bakıyor olmalı

### OBJE OLUŞTURMAK (INSTANTİATE)

Oyun sırasında sık sık yeni yeni objeler oluşturmak (**create**) zorunda kalırsınız. Bunun için

**Instantiate()** fonksiyonu bulunmaktadır.

Şimdi oyuncunun ateş etme tuşuna (Sol mouse tuşu veya sol CTRL tuşu) her bastığında yeni

bir objenin oluşmasını sağlayan bir kod yazarak bu konuyu anlayalım.

Yani ne yapmak istiyoruz? Oyuncunun etrafta normal bir şekilde dolanmasını, ateş etme

tuşuna basınca da yeni bir obje oluşmasını (instantiate) istiyoruz. Önemli olan hususlar:

**1.** Hangi objeyi oluşturmak istiyoruz?

**2.** Nerede oluşturmak istiyoruz?

Hangi objeyi oluşturacağımız hususunda, yapılabilecek en iyi şey Görünür bir değişken

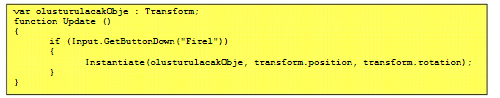
oluşturmaktır. Bu sayede oluşturulacak objeyi arayüz vasıtasıyla her an değiştirebiliriz.

Nerede oluşturacağımıza gelince, şimdilik sadece kamera objesinin olduğu konumda

oluşturacağız.

Instantiate() fonksiyonunda 3 parametre vardır; (1) oluşturacağımız obje, (2) objenin

oluşacağı 3 boyutlu konum ve (3) objenin oluşunca sahip olacağı eğim (rotation).

*Olustur* adında bir Javascript oluşturup içine şu kodu yazın:

Buradaki **transform.position** ve **transform.rotation** ikilisi, scriptin atandığı objenin o anki

konumunu ve eğimini ifade etmek için kullanılır. Bizim durumumuzda bu obje Main Camera

objesi olmakta. Genelde bir obje oluştururken oluşturulacak obje bir **prefab** olur. Bunun için

şimdi *Kup1* objesini bir prefab’a çevireceğiz.

### DEBUG İŞLEMİ (HATA AYIKLAMA)

Debug işlemi, kodlardaki insansı hataları bulup gidermeye denir. Unity, Debug sınıfıyla

(class) bize epey yardımcı olmaktadır. Şimdi beraber **Debug.Log()** fonksiyonuna bakacağız.

Log (Çıktı)

Log() fonksiyonu Unity’nin Console (Konsol) paneline bir mesaj çıktısı yollamaya yarar. Bunu

yapmayı istememiz için olası sebepler:

**1.** Bir kodun belli bir kısmının tamamlandığından emin olmak.

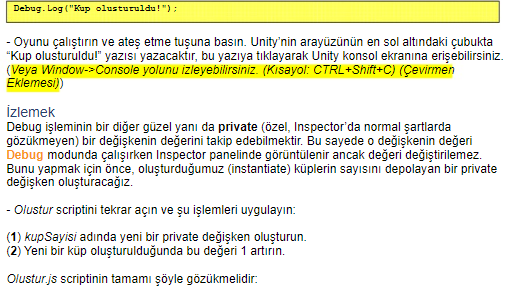
**2.** Bir değişkenin durumundan sürekli haberdar olmak.

Şimdi Log() fonksiyonunu kullanarak oyuncu ateş etme tuşuna her bastığında Unity’nin

konsol ekranına mesaj yollayacağız.

- *Olustur* scriptini açın ve ‘if’ bloğunun içerisinde, ‘Instantiate’ fonksiyonundan hemen sonraki

satıra şu kodu ekleyin:



### YAYGIN OLARAK KULLANILAN FONKSİYONLAR

Her yeni Javascript dosyası varsayılan olarak bir Update() fonksiyonuyla gelir. Bu bölümde

diğer yaygın olarak kullanılan fonksiyonlar listelenecek, yapmanız gereken Update() ismini

alttakilerden biriyle değiştirmek.

FixedUpdate()

Bu fonksiyona yazılan kodlar belli aralıklarla (düzenlenmiş framerate) gerçekleştirilir. Bu

fonksiyon genelde Rigidbody’e güç (force) uygularken kullanılır.

Awake()

Buraya yazılan kod, script çalıştırılmadan hemen önce gerçekleştirilir.

Start()

Buraya yazılan kodlar Update()’den önce ama Awake()’den sonra çalıştırılır. Start() ve

Awake() fonksiyonlarının birbirinden farkı; Start() fonksiyonu sadece eğer ki script açıksa

(enabled) çalıştırılırken (Yani Inspector panelinde scriptin solundaki kutucuk işaretliyse)

Awake() fonksiyonunda bu zorunlu değildir.

OnCollisionEnter()

Buraya yazılan kod, kodun yazıldığı GameObject başka bir GameObject’le temasa geçince

gerçekleştirilir.

OnMouseDown()

Buraya yazılan kod, GUIElement veya Collider component’lerinden herhangi birine sahip bir

objeye mouse ile sol tıklandığında gerçekleştirilir.

OnMouseOver()

Buraya yazılan kod, GUIElement veya Collider component’lerinden herhangi birine sahip bir

objenin üzerinde mouse bekletilince gerçekleştirilir.

# PROJE BİLGİLERİ

## GENEL BİLGİLER

Rabbit Jump, Tavşan karakteri yönettiğimiz yön tuşları ile oynanabilen zıplama oyunu. En yüksek puanı yapmaya dayalı bu oyunda bir çok engel ile karşılaşıyoruz.

### OYUNUN AMACI

### Rabbit Jump, Tavşan karakteri yönettiğimiz yön tuşları ile oynanabilen zıplama oyunu. En yüksek puanı yapmaya dayalı bu oyunda bir çok engel ile karşılaşıyoruz. Bu engeller kimi zaman bir yılan , kimi zaman bir kartal ve bununla da sınırlı değil. Tırmandığımız platform saydam olabiliyor , bazıları hareket ediyor . Ekstra puan kazanmak için Havuç nesnesi toplayarak en yüksek puanı yapmaya çalışıyorsunuz.

### 2.1.1.1 HEDEF KİTLE

Hedef kitlesi 7-18 yaş arası telefon ve bilgisayarda zaman geçirebileceğiniz eğlenceli bir oyundur.

## PROJE YAPILARI

### OYUN TÜRÜ

Tek kişilik oynanabilen 2D ile yapılmış eğlenceli platform oyunudur.

### OYUN PLATFORMU

Mobil ve bilgisayar platform da oynanabilen bir oyundur.

### OYUN NASIL OYNANIR

### Yön tuşları ile oynanır.

### OYUN İÇİN KULLANILAN RESİMLER

**ARKA PLAN**



**KARAKTER DÜŞMANLAR**



**NESNELER EKSTRA PUAN**

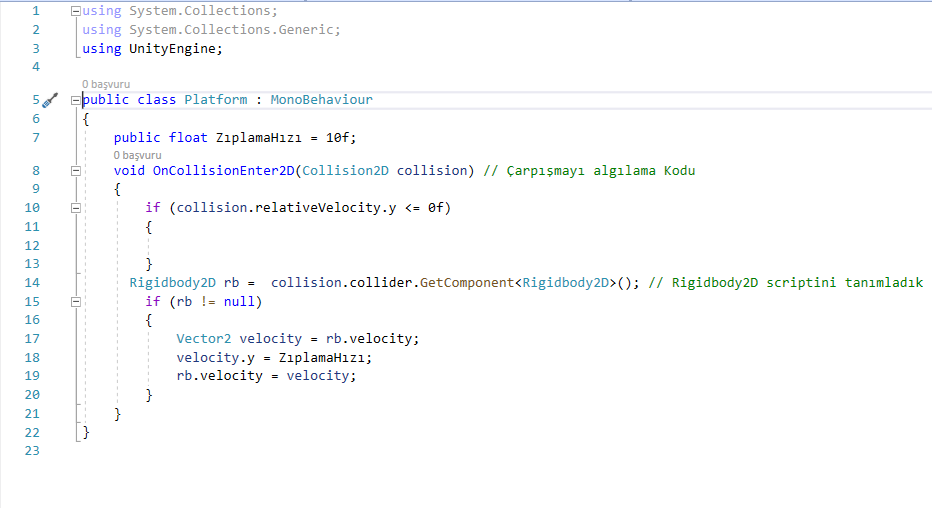
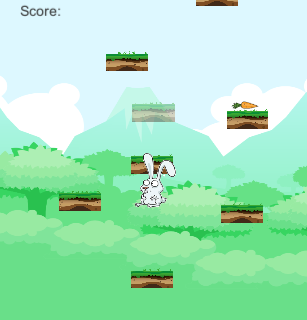


## KODLAR

## KameraTakip C# KOD

## 

## 

**PLATFORM C# KOD**

**PLAYER C# KOD**





# PLAYER C# KOD

# 

# OZELLİK C# KOD

# 

# KAYNAKÇA

<https://tr.wikipedia.org/wiki/Unity_(oyun_motoru)>

<https://www.hataverdi.com/unity-3d-nedir-nasil-indirilir-ne-ise-yarar/2014/03/>

<http://ustaderslik.com/pdf/unity-2/>

<http://www.techno-records.com/unity-nedir-unity-hakkinda-her-sey/>

# GİTHUB LİNK