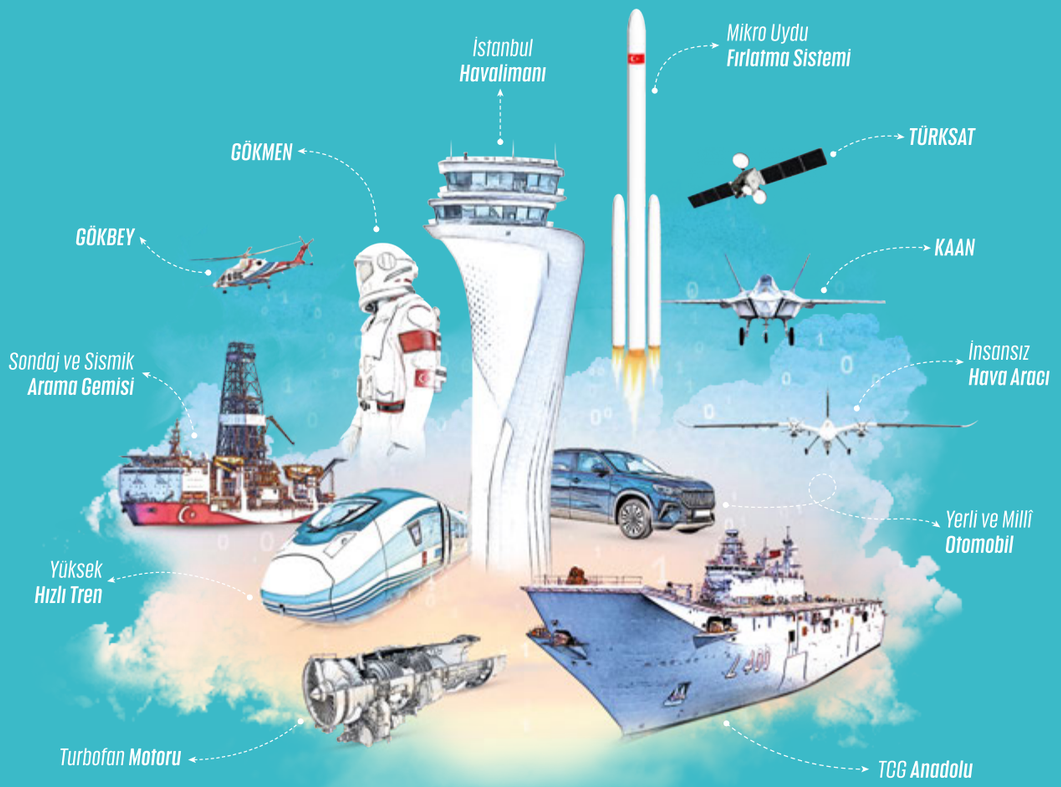
# Bu kitaba sığmayan daha neler var!

eba
www.eba.gov.tr

Karekodu okut, bu kitapla ilgili EBA içeriklerine ulaş!

## Teknoloji Yolculuğumuz




**BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA ÜCRETSİZ OLARAK VERİLMİŞTİR.**
**PARA İLE SATILAMAZ.**

**ISBN: 978-975-11-7977-7**

*Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmeliğin Beşinci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.*

BİLİŞİM TEKNOLOJİLERİ ALANI

OYUN PROGRAMLAMA (KOD TABANLI)

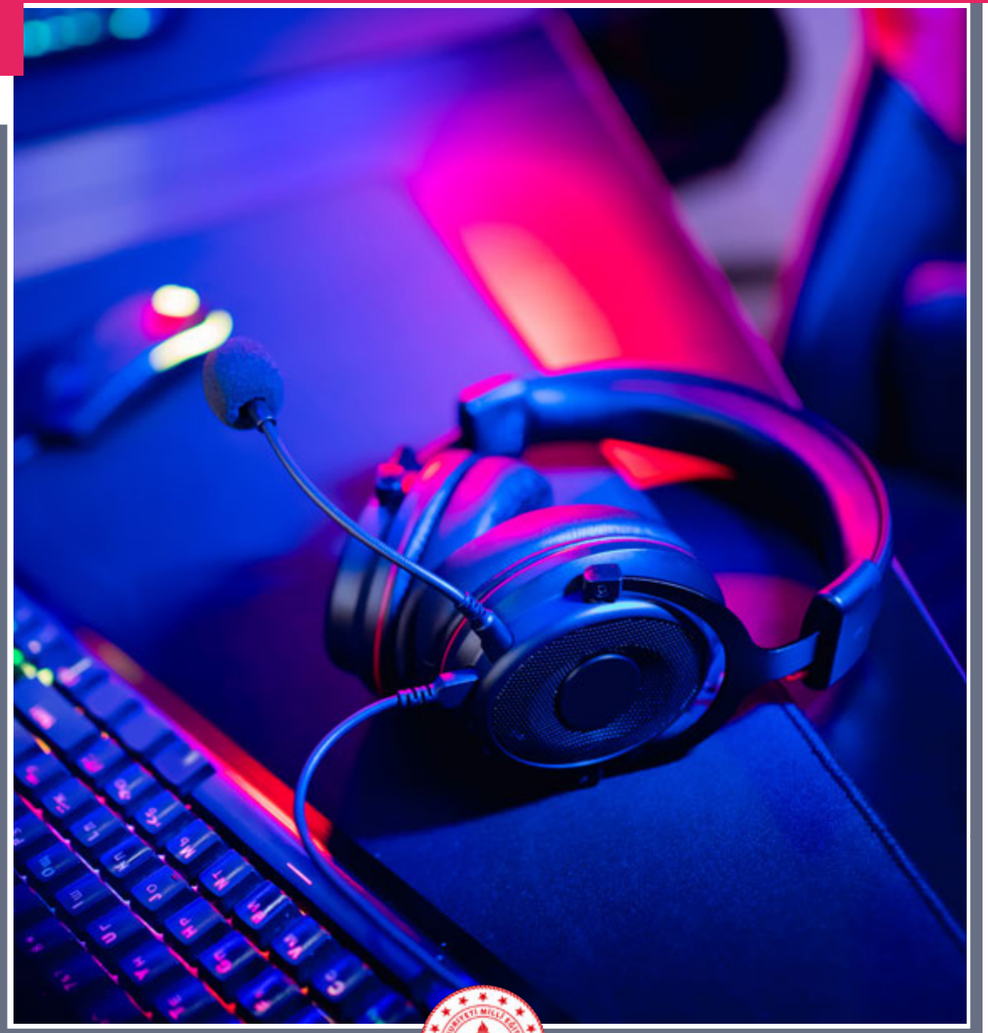
11-12 DERS MATERYALİ

MESLEKİ VE TEKNİK ANADOLU LİSESİ

## BİLİŞİM TEKNOLOJİLERİ ALANI

11-12 DERS MATERYALİ

# OYUN PROGRAMLAMA (KOD TABANLI)



MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

# OYUN PROGRAMLAMA (KOD TABANLI)

# 11-12

## DERS MATERYALİ

**Yazarlar**

Dr. Ömer YILMAZ
Ahmet Emin KOYDEMİR
Coşkun CENGİZ
Tarık BAĞRIYANIK

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI .....: 9453
YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ .....: 3113

Her hakkı saklıdır ve Millî Eğitim Bakanlığına aittir. Ders materyalinin metin, soru ve şekilleri kısmen de olsa hiçbir surette alınıp yayımlanamaz.

**HAZIRLAYANLAR**

**Dil Uzmanı**
Gülhan ÇİĞDEM

**Program Geliştirme Uzmanı**
Erkan AKGÜN

**Ölçme ve Değerlendirme Uzmanı**
Günay DURUCAN

**Rehberlik Uzmanı**
Filiz KILIÇ

**Grafik Tasarım Uzmanı**
Feyza Gedik

ISBN: 978-975-11-7977-7

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Meslekî ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.



# İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

**Mehmet Âkif Ersoy**

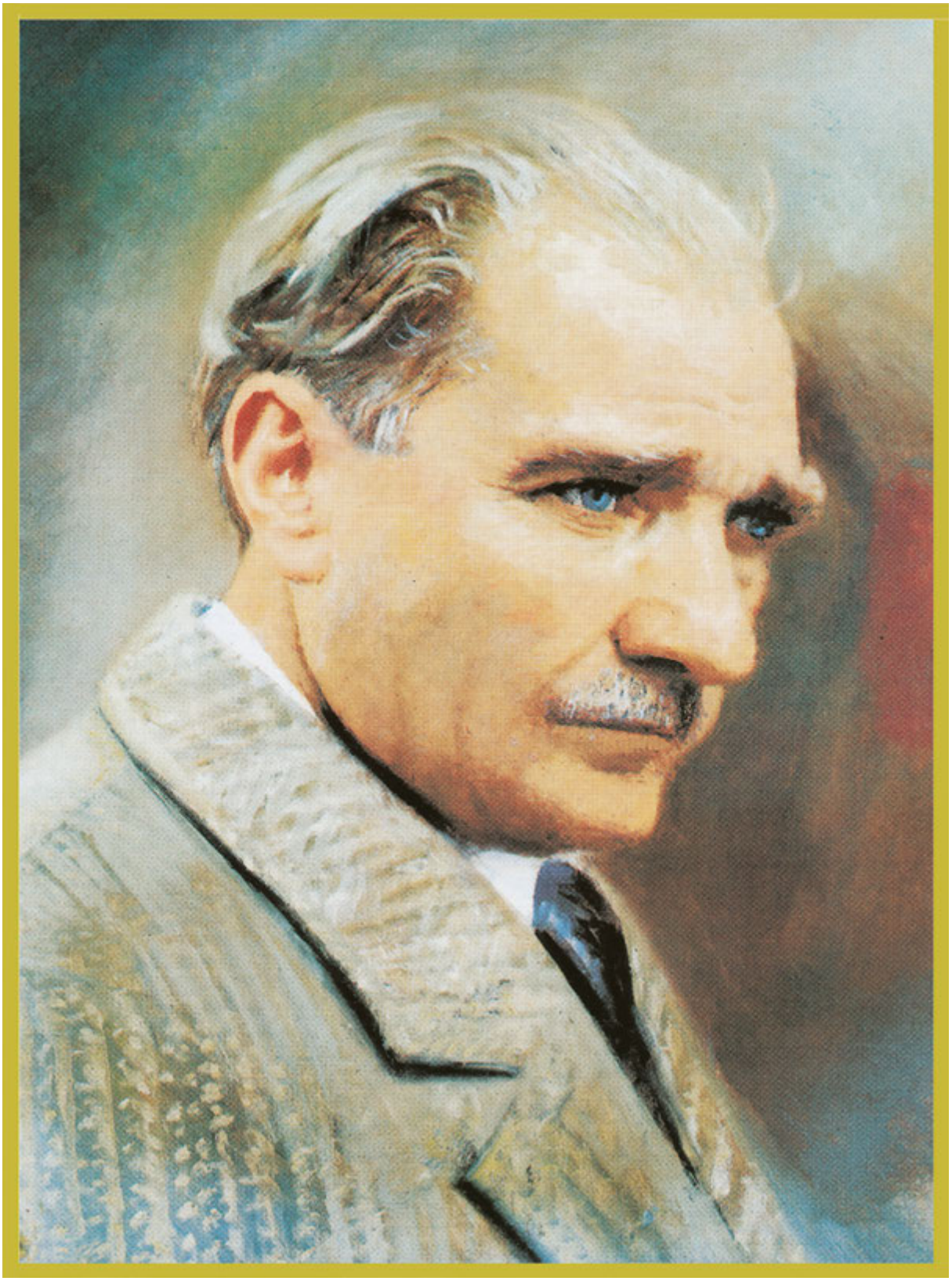
# GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk



MUSTAFA KEMAL ATATÜRK

# İÇİNDEKİLER

DERS MATERYALİNİN TANITIMI ..... 13

## 1. ÖĞRENME BİRİMİ: OYUN PROGRAMLAMA TEMELLERİ

**1.1. OYUN GELİŞTİRME UYGULAMASI TEMEL ARAYÜZÜNÜN TANITILMASI ..... 20**

1.1.1. Uygulama Kurulumu ..... 20

1.1.2. Proje Oluşturma ..... 22

1.1.3. Uygulama Arayüzü ..... 23

1.1.4. Oyun Nesneleri ..... 24

1.1.5. Nesne Dönüşümleri ..... 27

1.1.6. Model Ekleme ve Düzenleme ..... 33

1.1.7. Materyaller, Dokular ve Gölgelendiriciler ..... 40

1.1.8. Oyun Zemini ..... 46

**1.2. OYUN UYGULAMASINDA TEMEL KODLAMA ..... 53**

1.2.1. Script Dosyaları Oluşturma ..... 53

1.2.2. Değişkenler ..... 57

1.2.3. Operatörler ..... 62

1.2.4. Karar Yapıları ..... 66

1.2.5. Döngüler ..... 68

1.2.6. Metotlar ..... 71

**1.3. KULLANICI ETKİLEŞİM İŞLEMLERİ ..... 73**

1.3.1. Kullanıcı Girişleri ..... 74

1.3.2. Oyun Motorunda Fizik ..... 82

1.3.3. Sahne İşlemleri ..... 93

1.3.4. Kullanıcı Arayüzü İşlemleri ..... 94

1.3.5. JSON Veri İşlemleri ..... 105

1.3.6. Zamanlanmış İşlemler ..... 110

**ÖLÇME VE DEĞERLENDİRME ..... 114**

## 2. ÖĞRENME BİRİMİ: KARAKTER VE ÇEVRE DÜZENLEME

**2.1. SENARYO OLUŞTURMA ..... 118**

2.1.1. Oyun Senaryosu İçin Plan Taslağı Hazırlama ..... 118

2.1.2. Oyun Senaryosunda Kullanılan Görsel-İşitsel İçerikler ..... 120

**2.2. RESİM DÜZENLEME ..... 121**

2.2.1. Resim Türleri Dönüşümleri ..... 121

2.2.2. Saydam Arka Planlı Resim Oluşturma ..... 123

2.2.3. Hareketli Resimler Oluşturma ..... 126

2.2.4. Doku Hazırlama ..... 129

**2.3. ÜÇ BOYUTLU MODELLEME ..... 138**

2.3.1. Üç Boyutlu Modelleme Programı ..... 138

2.3.2. Düşük Poligon Modelleme ..... 140

2.3.3. Üç Boyutlu Hazır Karakter ve Animasyon Platformu ..... 151

**2.4. OYUN PROGRAMINA AKTARMA ..... 154**

2.4.1. Projeye Dosya Ekleme ve Düzenleme ..... 154

2.4.2. Tekrar Kullanılabilir Nesne (Prefab) Oluşturma ..... 157

**ÖLÇME VE DEĞERLENDİRME ..... 162**

## 3. ÖĞRENME BİRİMİ: OYUNDA ANİMASYON


**3.1. OYUNDA ANİMASYON ..... 166**

3.1.1. Sinematik Ekleme ..... 166

3.1.2. Tekrarlı Olan Animasyonları Ekleme ..... 171

3.1.3. Kodlama İle Animasyonu Kontrol Etme ..... 176

3.1.4. Karakterin Hareketlendirilmesi ve Etkileşime Girmesi ..... 178

3.1.5. Nesnelerin Hareketlendirilmesi ..... 184

**3.2. SİMÜLASYON İŞLEMLERİ ..... 188**

3.2.1. Çeşitli Etkileri İçeren Uygulamalar ..... 189

**3.3. KARAKTER VE ÇEVRE OLUŞTURMA ..... 206**

3.3.1. Güneş, Işık Türleri ve Etkileri ..... 207

3.3.2. Birden Fazla Kamera Kullanma ..... 210

3.3.3. Kullanıcı Arayüzüne Kuş Bakışı Harita Ekleme ..... 214

3.3.4. Gerçek Zamanlı Yansımalar Ekleme ..... 218

3.3.5. Bitki, Hayvan ve Diğer Doğa Nesnelerini Ekleme ..... 220

3.3.6. Diğer Oyuncular İle İletişim ve Diyalog Pencereleri Yapımı ..... 224

3.3.7. Gece ve Gündüz Gibi Zaman İlerlemesi ..... 230

3.3.8. Şehir Nesnelerini Ekleme ..... 232

3.3.9. Hava, Kara ve Su Araçlarını Kullanma ..... 236

**3.4. SES EKLEME İŞLEMLERİ ..... 240**

3.4.1. Ses Dosyaları Türleri ve Dönüşümleri ..... 240

3.4.2. Etkileşimde Ses Etkileri ..... 241

3.4.3. Arka Plana Müzik Ekleme ..... 243

**ÖLÇME VE DEĞERLENDİRME ..... 246**

## 4. ÖĞRENME BİRİMİ: OYUN OLUŞTURMA VE YAYINLAMA


**4.1. YAPAY ZEKÂ VE ÇOKLU OYUNCU DESTEĞİ ..... 250**

4.1.1. Oyuncu Olmayan Karakterlerin (NPC) Yapay Zekâ ile Etkileşimleri ..... 250

4.1.2. Yerel Ağda Birlikte Oynanan Oyun Yapımı ..... 256

4.1.3. İnternet Üzerinden Oynanan (Online) Oyun Yapımı ..... 263

4.1.4. Alışveriş ve Nesne Yapma Sistemleri ..... 278

**4.2. TEST ETME VE İYİLEŞTİRME İŞLEMLERİ ..... 283**

4.2.1. Oyunun Oynanışını Test Etme ..... 283

4.2.2. Oyun İçi Reklam ve Analiz Desteği ..... 284

4.2.3. Oyun İçi ve Çevrim İçi Yardım Ekleme ..... 292

4.2.4. Oyunun Bellek, Grafik ve Oynanış Olarak İyileştirilmesi ..... 300

**4.3. DERLEME VE YAYINLAMA İŞLEMLERİ ..... 308**

4.3.1. Derlenebilen Platform Türleri ..... 308

4.3.2. Artırılmış (AR), Sanal (VR) ve Karma (MR) Gerçeklik Türleri ..... 310

4.3.3. Uygulama Marketlerine Uygulama Ekleme ..... 315

4.3.4. Güncelleme (Update) ve Yama (Patch) Ekleme ..... 320

**ÖLÇME VE DEĞERLENDİRME ..... 322**

**CEVAP ANAHTARLARI ..... 324**

**KAYNAKÇA ..... 325**

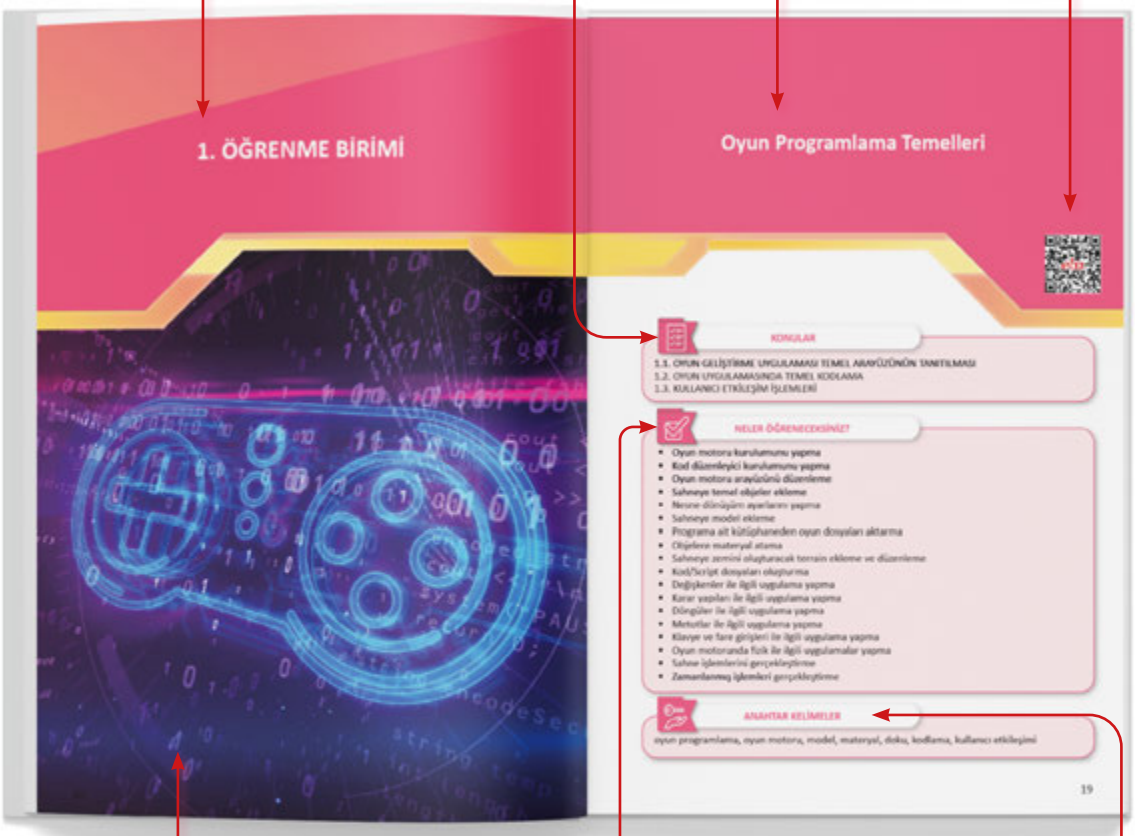
# DERS MATERYALİNİN TANITIMI

Öğrenme biriminin numarasını gösterir.

Öğrenme biriminde yer alan konu başlıklarını gösterir.

Öğrenme biriminin adını gösterir.

Öğrenme biriminin karekodunu gösterir.




Öğrenme birimine uygun görseli içerir.

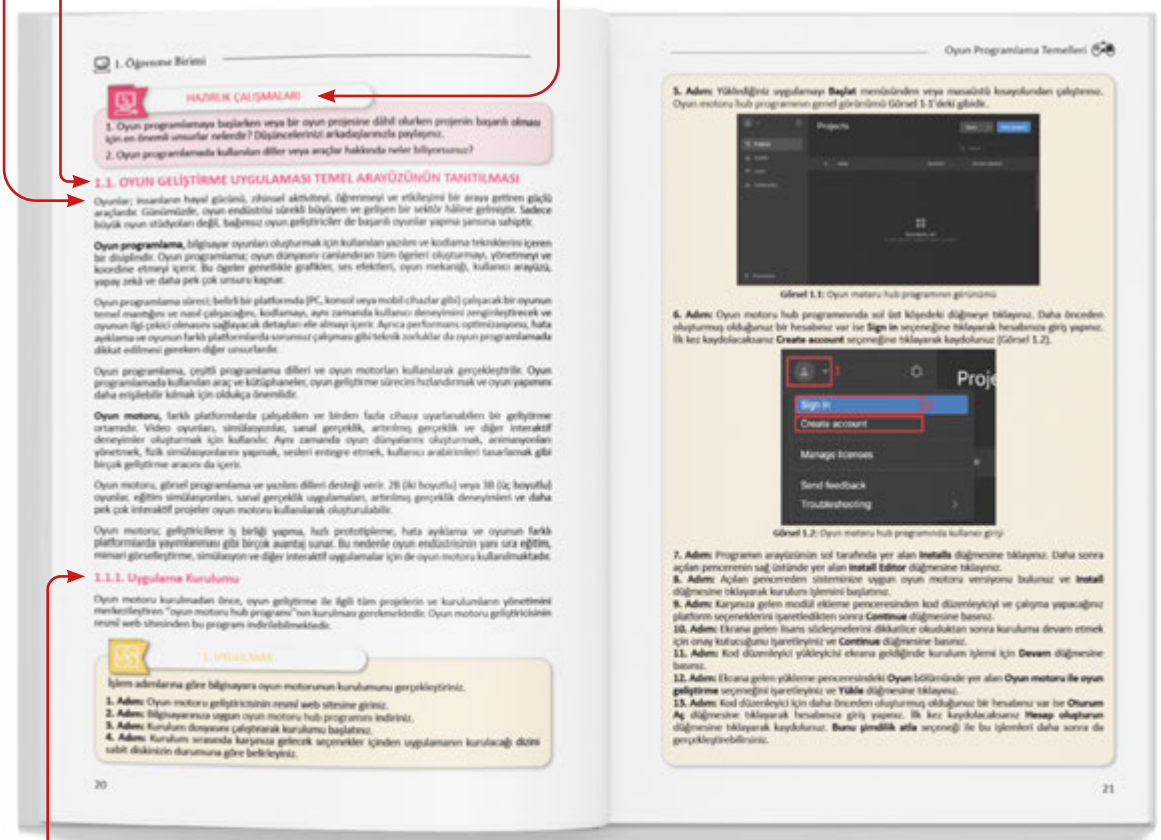
Öğrenme biriminde neler öğrenileceğini gösterir.

Öğrenme birimi ile ilgili öğrenilmesi gereken önemli kavramları gösterir.

Konu anlatımını gösterir.

Konu başlığını gösterir.

Hazırlık çalışmalarını gösterir.



1. Öğrenme Birimi

HAZIRLIK ÇALIŞMALARI

1. Oyun programlamaya başlarken veya bir oyun projesine dâhil olurken projenin başarılı olması için en önemli unsurlar nelerdir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Oyun programlamada kullanılan diller veya araçlar hakkında neler biliyorsunuz?

## 1.1. OYUN GELİŞTİRME UYGULAMASI TEMEL ARAYÜZÜNÜN TANITILMASI

Oyunlar; insanların hayal gücünü, zihinsel aktivitesi, öğrenmeyi ve etkileşimi bir araya getiren güçlü araçlardır. Günümüzde, oyun endüstrisi sürekli büyüyen ve gelişen bir sektör hâline gelmiştir. Sadece büyük oyun stüdyoları değil, bağımsız oyun geliştiriciler de başarılı oyunlar yapma şansına sahiptir.

**Oyun programlama,** bilgisayar oyunları oluşturmak için kullanılan yazılım ve kodlama tekniklerini içeren bir disiplindir. Oyun programlama; oyun dünyasını canlandıran tüm öğeleri oluşturmayı, yönetmeyi ve koordine etmeyi içerir. Bu öğeler genellikle grafikler, ses efektleri, oyun mekaniği, kullanıcı arayüzü, yapay zekâ ve daha pek çok unsuru kapsar.

Oyun programlama süreci; belirli bir platformda (PC, konsol veya mobil cihazlar gibi) çalışacak bir oyunun temel mantığını ve nasıl çalışacağını, kodlamayı, aynı zamanda kullanıcı deneyimini zenginleştirecek ve oyunun ilgi çekici olmasını sağlayacak detayları ele almayı içerir. Ayrıca performans optimizasyonu, hata ayıklama ve oyunun farklı platformlarda sorunsuz çalışması gibi teknik zorluklar da oyun programlamada dikkat edilmesi gereken diğer unsurlardır.

Oyun programlama, çeşitli programlama dilleri ve oyun motorları kullanılarak gerçekleştirilir. Oyun programlamada kullanılan araç ve kütüphaneler, oyun geliştirme sürecini hızlandırmak ve oyun yapımını daha erişilebilir kılmak için oldukça önemlidir.

**Oyun motoru,** farklı platformlarda çalışabilen ve birden fazla cihaza uyarlanabilen bir geliştirme ortamıdır. Video oyunları, simülasyonlar, sanal gerçeklik, artırılmış gerçeklik ve diğer interaktif deneyimler oluşturmak için kullanılır. Aynı zamanda oyun dünyalarını oluşturmak, animasyonları yönetmek, fizik simülasyonlarını yapmak, sesleri entegre etmek, kullanıcı arabirimleri tasarlamak gibi birçok geliştirme aracını da içerir.

Oyun motoru, görsel programlama ve yazılım dilleri desteği verir. 2B (iki boyutlu) veya 3B (üç boyutlu) oyunlar, eğitim simülasyonları, sanal gerçeklik uygulamaları, artırılmış gerçeklik deneyimleri ve daha pek çok interaktif projeler oyun motoru kullanılarak oluşturulabilir.

Oyun motoru; geliştiricilere iş birliği yapma, hızlı prototipleme, hata ayıklama ve oyunun farklı platformlarda yayımlanması gibi birçok avantaj sunar. Bu nedenle oyun endüstrisinin yanı sıra eğitim, mimari görselleştirme, simülasyon ve diğer interaktif uygulamalar için de oyun motoru kullanılmaktadır.

### 1.1.1. Uygulama Kurulumu

Oyun motoru kurulmadan önce, oyun geliştirme ile ilgili tüm projelerin ve kurulumların yönetimini merkezileştiren "oyun motoru hub programı"nın kurulması gerekmektedir. Oyun motoru geliştiricisinin resmî web sitesinden bu program indirilebilmektedir.

1. UYGULAMA

İşlem adımlarına göre bilgisayara oyun motorunun kurulumunu gerçekleştiriniz.

1. **Adım:** Oyun motoru geliştiricisinin resmî web sitesine giriniz.
2. **Adım:** Bilgisayarınıza uygun oyun motoru hub programını indiriniz.
3. **Adım:** Kurulum dosyasını çalıştırarak kurulumu başlatınız.
4. **Adım:** Kurulum sırasında karşınıza gelecek seçenekler içinden uygulamanın kurulacağı dizini sabit diskinizin durumuna göre belirleyiniz.

20

Oyun Programlama Temelleri

5. **Adım:** Yüklediğiniz uygulamayı **Başlat** menüsünden veya masaüstü kısayolundan çalıştırınız. Oyun motoru hub programının genel görünümü Görsel 1.1'deki gibidir.




**Görsel 1.1:** Oyun motoru hub programının görünümü

6. **Adım:** Oyun motoru hub programında sol üst köşedeki düğmeye tıklayınız. Daha önceden oluşturmuş olduğunuz bir hesabınız var ise **Sign in** seçeneğine tıklayarak hesabınıza giriş yapınız. İlk kez kaydolacaksanız **Create account** seçeneğine tıklayarak kaydolunuz (Görsel 1.2).




**Görsel 1.2:** Oyun motoru hub programında kullanıcı girişi

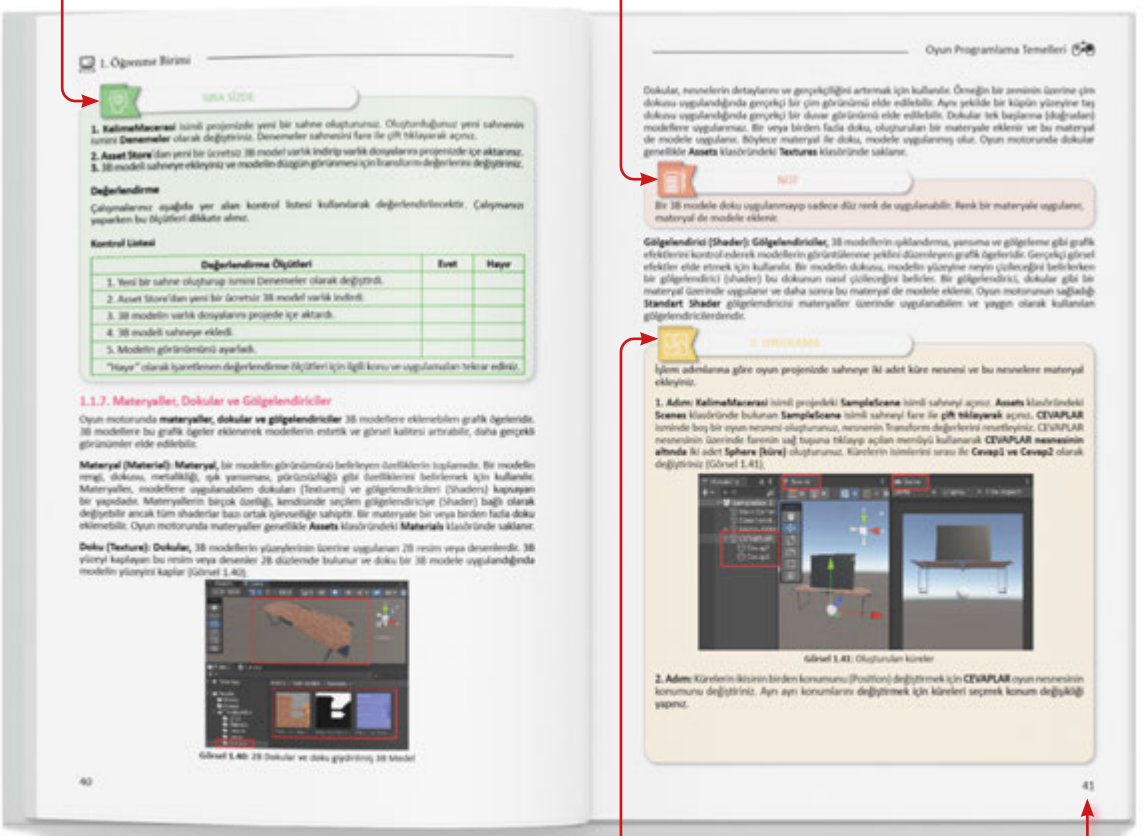
7. **Adım:** Programın arayüzünün sol tarafında yer alan **Installs** düğmesine tıklayınız. Daha sonra açılan pencerenin sağ üstünde yer alan **Install Editor** düğmesine tıklayınız.
8. **Adım:** Açılan pencereden sisteminize uygun oyun motoru versiyonu bulunuz ve **Install** düğmesine tıklayarak kurulum işlemini başlatınız.
9. **Adım:** Karşınıza gelen modül ekleme penceresinden kod düzenleyiciyi ve çalışma yapacağınız platform seçeneklerini işaretledikten sonra **Continue** düğmesine basınız.
10. **Adım:** Ekrana gelen lisans sözleşmelerini dikkatlice okuduktan sonra kuruluma devam etmek için onay kutucuğunu işaretleyiniz ve **Continue** düğmesine basınız.
11. **Adım:** Kod düzenleyici yükleyicisi ekrana geldiğinde kurulum işlemi için **Devam** düğmesine basınız.
12. **Adım:** Ekrana gelen yükleme penceresindeki **Oyun** bölümünde yer alan **Oyun motoru ile oyun geliştirme** seçeneğini işaretleyiniz ve **Yükle** düğmesine tıklayınız.
13. **Adım:** Kod düzenleyici için daha önceden oluşturmuş olduğunuz bir hesabınız var ise **Oturum Aç** düğmesine tıklayarak hesabınıza giriş yapınız. İlk kez kaydolacaksanız **Hesap oluşturun** düğmesine tıklayarak kaydolunuz. **Bunu şimdilik atla** seçeneği ile bu işlemleri daha sonra da gerçekleştirebilirsiniz.

21

Alt konu başlığını gösterir.

Öğrenilen konu ile ilgili öğrencilerin sınıf ortamında uygulama yapmasının istendiği bölümü gösterir.

Konu ve uygulamalara ilişkin ek bilgileri gösterir.



1. Öğrenme Birimi

SIRA SİZDE

**1. KelimeMacerası** isimli projenizde yeni bir sahne oluşturunuz. Oluşturduğunuz yeni sahnenin ismini **Denemeler** olarak değiştiriniz. Denemeler sahnesini fare ile çift tıklayarak açınız.

**2. Asset Store**'dan yeni bir ücretsiz 3B model varlık indirip varlık dosyalarını projenizde içe aktarınız.

**3.** 3B modeli sahneye ekleyiniz ve modelin düzgün görünmesi için Transform değerlerini değiştiriniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Yeni bir sahne oluşturup ismini Denemeler olarak değiştirdi. | | |
| 2. Asset Store'dan yeni bir ücretsiz 3B model varlık indirdi. | | |
| 3. 3B modelin varlık dosyalarını projede içe aktardı. | | |
| 4. 3B modeli sahneye ekledi. | | |
| 5. Modelin görünümünü ayarladı. | | |

"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.

### 1.1.7. Materyaller, Dokular ve Gölgelendiriciler

Oyun motorunda **materyaller, dokular ve gölgelendiriciler** 3B modellere eklenebilen grafik öğelerdir. 3B modellere bu grafik öğeler eklenerek modellerin estetik ve görsel kalitesi artırılabilir, daha gerçekçi görünümler elde edilebilir.

**Materyal (Material): Materyal,** bir modelin görünümünü belirleyen özelliklerin toplamıdır. Bir modelin rengi, dokusu, metalliği, ışık yansıması, pürüzsüzlüğü gibi özelliklerini belirlemek için kullanılır. Materyaller, modellere uygulanabilen dokuları (Texturers) ve gölgelendiricileri (Shaders) kapsayan bir yapıdadır. Materyallerin birçok özelliği, kendisinde seçilen gölgelendiriciye (Shader) bağlı olarak değişebilir ancak tüm shaderlar bazı ortak işlevselliğe sahiptir. Bir materyale bir veya birden fazla doku eklenebilir. Oyun motorunda materyaller genellikle **Assets** klasöründeki **Materials** klasöründe saklanır.

**Doku (Texture): Dokular,** 3B modellerin yüzeylerinin üzerine uygulanan 2B resim veya desenlerdir. 3B yüzeyi kaplayan bu resim veya desenler 2B düzlemde bulunur ve doku bir 3B modele uygulandığında modelin yüzeyini kaplar (Görsel 1.40).



**Görsel 1.40:** 2B Dokular ve doku giydirilmiş 3B Model

40

Oyun Programlama Temelleri

Dokular, nesnelerin detaylarını ve gerçekçiliğini artırmak için kullanılır. Örneğin bir zeminin üzerine çim dokusu uygulandığında gerçekçi bir çim görünümü elde edilebilir. Aynı şekilde bir küpün yüzeyine taş dokusu uygulandığında gerçekçi bir duvar görünümü elde edilebilir. Dokular tek başlarına (düğradan) modellere uygulanmaz. Bir veya birden fazla doku, oluşturulan bir materyale eklenir ve bu materyal de modele uygulanır. Böylece materyal ile doku, modele uygulanmış olur. Oyun motorunda dokular genellikle **Assets** klasöründeki **Textures** klasöründe saklanır.

NOT

Bir 3B modele doku uygulanmayıp sadece düz renk de uygulanabilir. Renk bir materyale uygulanır, materyal de modele eklenir.

**Gölgelendirici (Shader): Gölgelendiriciler,** 3B modellerin aydınlatma, yansıma ve gölgeleme gibi grafik efektlerini kontrol ederek modellerin görüntülenme şeklini düzenleyen grafik öğeleridir. Gerçekçi görsel efektler elde etmek için kullanılır. Bir modelin dokusu, modelin yüzeyine neyin çizileceğini belirlerken bir gölgelendirici (shader) bu dokunun nasıl çizileceğini belirler. Bir gölgelendirici, dokular gibi bir materyal üzerinde uygulanır ve daha sonra bu materyal de modele eklenir. Oyun motorunun sağladığı **Standart Shader** gölgelendiricisi materyaller üzerinde uygulanabilen ve yaygın olarak kullanılan gölgelendiricilerdendir.

UYGULAMA

İşlem adımlarına göre oyun projenizde sahneye iki adet küre nesnesi ve bu nesnelere materyal ekleyiniz.

**1. Adım: KelimeMacerası** isimli projedeki **SampleScene** isimli sahneyi açınız. **Assets** klasöründeki **Scenes** klasöründe bulunan **SampleScene** isimli sahneyi fare ile **çift tıklayarak** açınız. **CEVAPLAR** isminde boş bir oyun nesnesi oluşturunuz, nesnenin Transform değerlerini sıfırlayınız. CEVAPLAR nesnesinin üzerinde farenin sağ tuşuna tıklayıp açılan menüyü kullanarak **CEVAPLAR nesnesinin altında** iki adet **Sphere (küre)** oluşturunuz. Kürelerin isimlerini sırası ile **Cevap1 ve Cevap2** olarak değiştiriniz (Görsel 1.41).



**Görsel 1.41:** Oluşturulan küreler

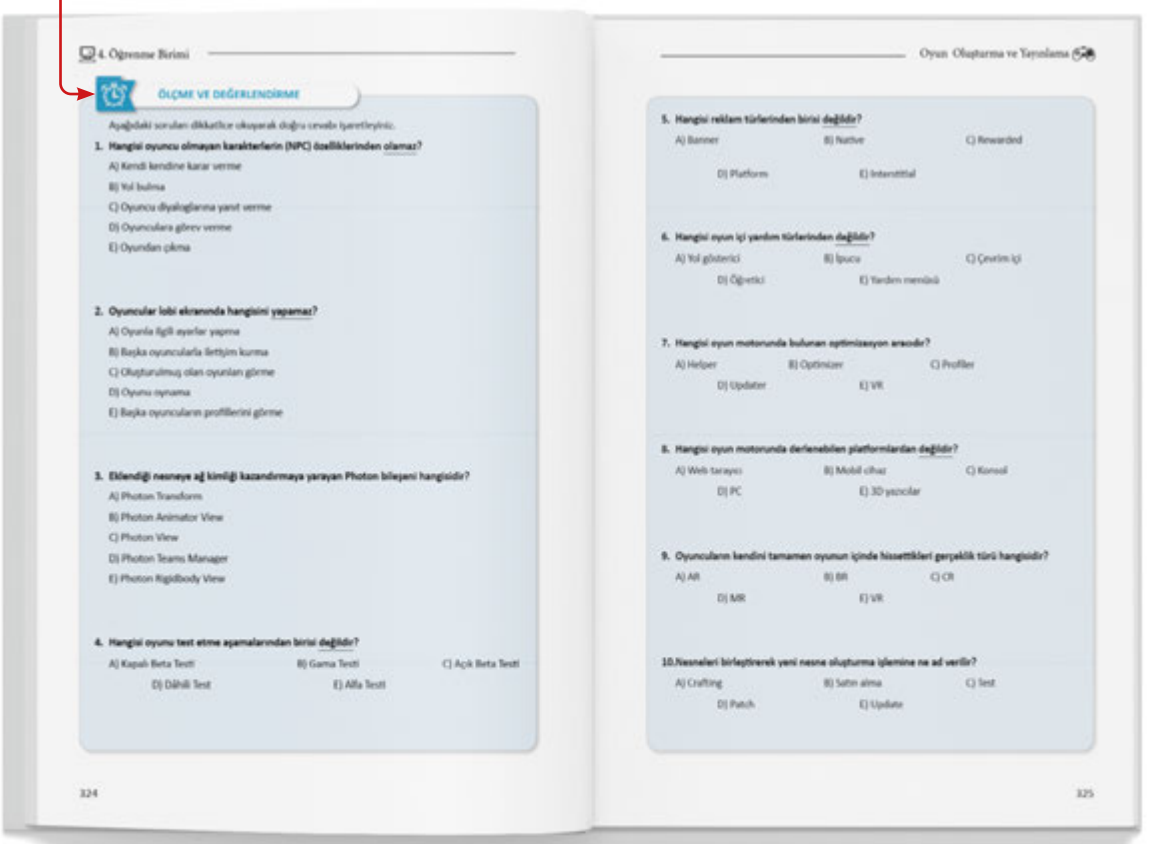
**2. Adım:** Kürelerin ikisinin birden konumunu (Position) değiştirmek için **CEVAPLAR** oyun nesnesinin konumunu değiştiriniz. Aynı aynı konumlarını değiştirmek için küreleri seçerek konum değişikliği yapınız.

41

Öğrenme birimindeki konulara dikkat çekmek ve öğrencilerde merak duygusunu geliştirmek amacıyla yapılacak çalışmaları gösterir.

Sayfa numarasını gösterir.

Kazanılan bilgi ve becerilerin her öğrenme birimi sonunda ölçüldüğü soruları gösterir.



4. Öğrenme Birimi

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru cevabı işaretleyiniz.

**1. Hangisi oyuncu olmayan karakterlerin (NPC) özelliklerinden olamaz?**

A) Kendi kendine karar verme
B) Yol bulma
C) Oyuncu diyaloglarına yanıt verme
D) Oyunculara görev verme
E) Oyundan çıkma

**2. Oyuncular lobi ekranında hangisini yapamaz?**

A) Oyunla ilgili ayarlar yapma
B) Başka oyuncularla iletişim kurma
C) Oluşturulmuş olan oyunları görme
D) Oyunu oynama
E) Başka oyuncuların profillerini görme

**3. Eklendiği nesneye ağ kimliği kazandırmaya yarayan Photon bileşeni hangisidir?**

A) Photon Transform
B) Photon Animator View
C) Photon View
D) Photon Teams Manager
E) Photon Rigidbody View

**4. Hangisi oyunu test etme aşamalarından birisi değildir?**

A) Kapalı Beta Testi
B) Gama Testi
C) Açık Beta Testi
D) Dâhili Test
E) Alfa Testi

324

Oyun Oluşturma ve Yayınlama

**5. Hangisi reklam türlerinden birisi değildir?**

A) Banner
B) Native
C) Rewarded
D) Platform
E) Interstitial

**6. Hangisi oyun içi yardım türlerinden değildir?**

A) Yol gösterici
B) İpucu
C) Çevrim içi
D) Öğretici
E) Yardım menüsü

**7. Hangisi oyun motorunda bulunan optimizasyon aracıdır?**

A) Helper
B) Optimizer
C) Profiler
D) Updater
E) VR

**8. Hangisi oyun motorunda derlenebilen platformlardan değildir?**

A) Web tarayıcı
B) Mobil cihaz
C) Konsol
D) PC
E) 3D yazıcılar

**9. Oyuncuların kendini tamamen oyunun içinde hissettikleri gerçeklik türü hangisidir?**

A) AR
B) BR
C) CR
D) MR
E) VR

**10. Nesneleri birleştirerek yeni nesne oluşturma işlemine ne ad verilir?**

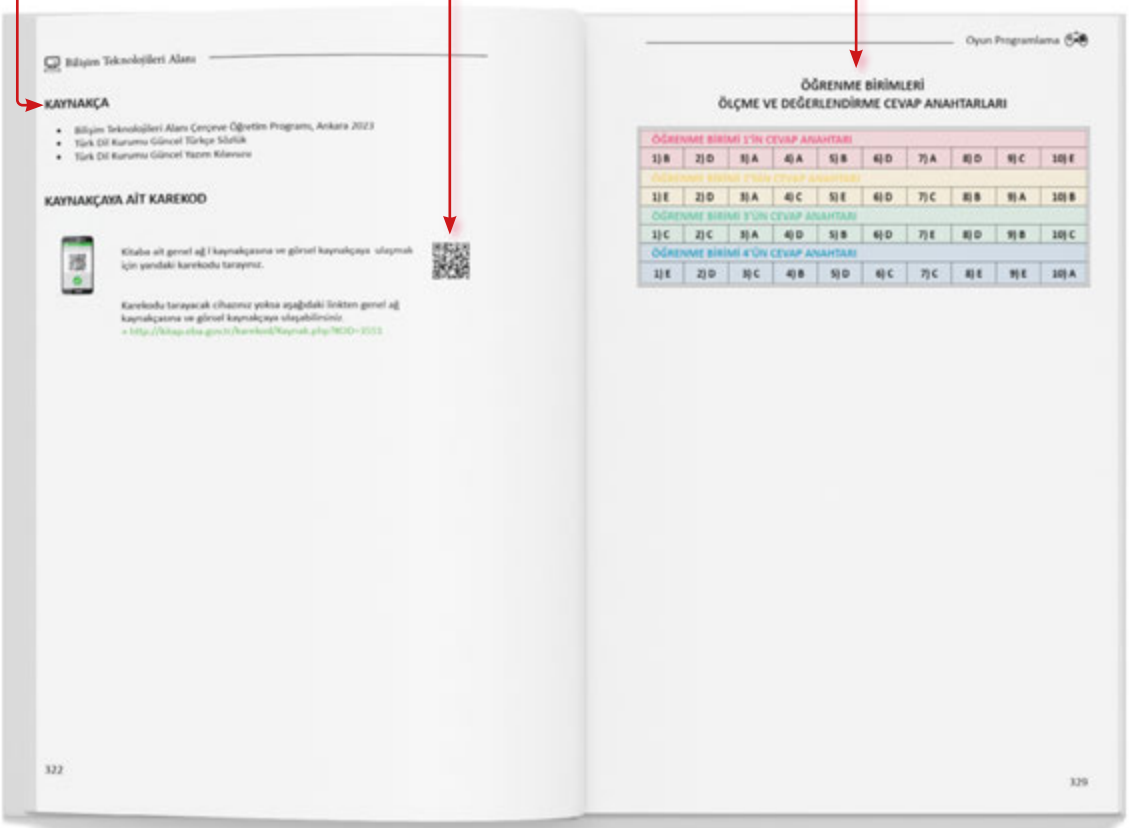
A) Crafting
B) Satın alma
C) Test
D) Patch
E) Update

325

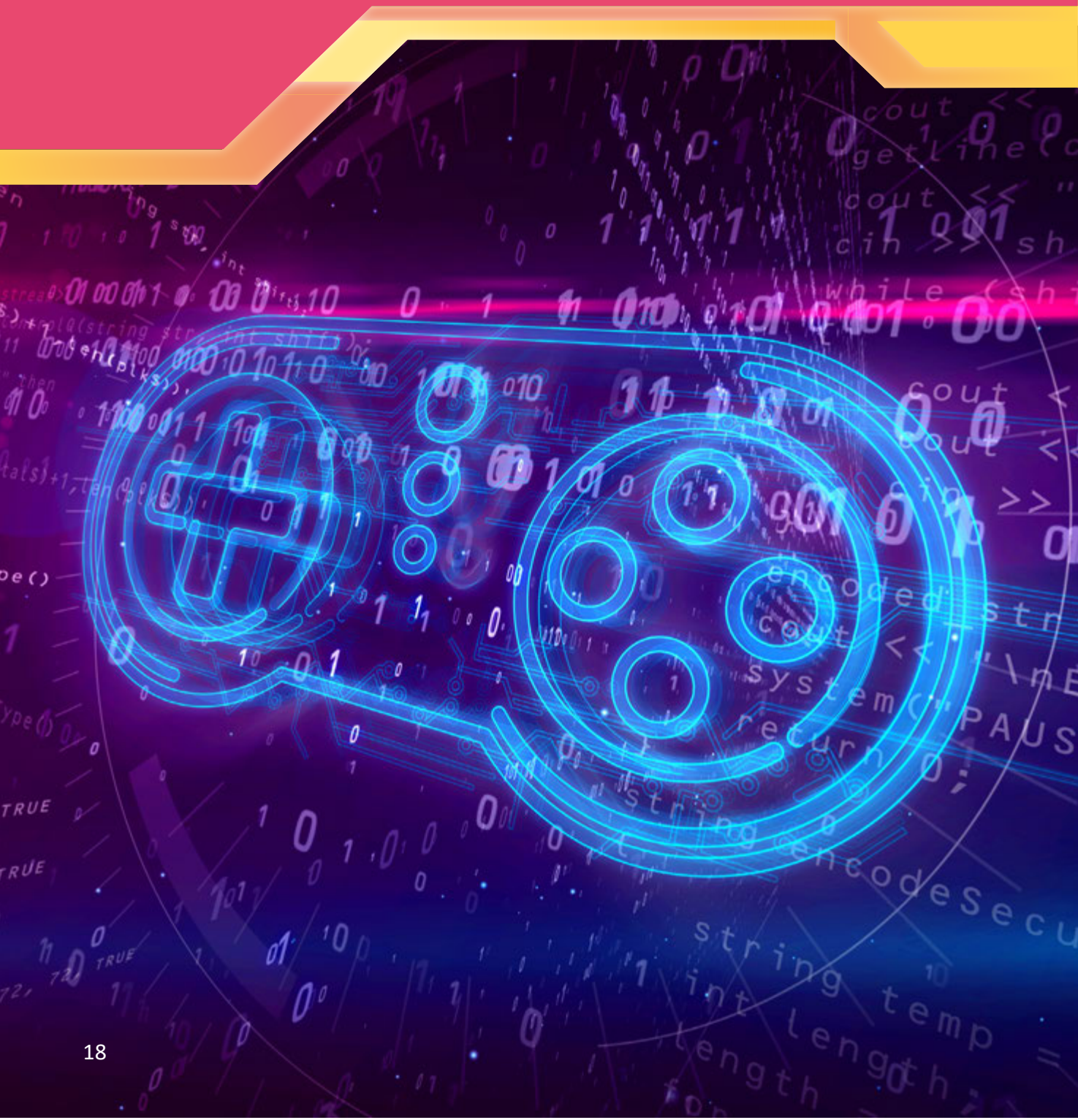
Kaynakçayı gösterir.

Karekod, görsel kaynakçayı gösterir.

Ölçme ve değerlendirme sorularına ait cevap anahtarlarını gösterir.



# 1. ÖĞRENME BİRİMİ



# Oyun Programlama Temelleri

## KONULAR

1.1. OYUN GELİŞTİRME UYGULAMASI TEMEL ARAYÜZÜNÜN TANITILMASI
1.2. OYUN UYGULAMASINDA TEMEL KODLAMA
1.3. KULLANICI ETKİLEŞİM İŞLEMLERİ

## NELER ÖĞRENECEKSİNİZ?

- Oyun motoru kurulumunu yapma
- Kod düzenleyici kurulumunu yapma
- Oyun motoru arayüzünü düzenleme
- Sahneye temel objeler ekleme
- Nesne dönüşüm ayarlarını yapma
- Sahneye model ekleme
- Programa ait kütüphaneden oyun dosyaları aktarma
- Objelere materyal atama
- Sahneye zemini oluşturacak terrain ekleme ve düzenleme
- Kod/Script dosyaları oluşturma
- Değişkenler ile ilgili uygulama yapma
- Karar yapıları ile ilgili uygulama yapma
- Döngüler ile ilgili uygulama yapma
- Metotlar ile ilgili uygulama yapma
- Klavye ve fare girişleri ile ilgili uygulama yapma
- Oyun motorunda fizik ile ilgili uygulamalar yapma
- Sahne işlemlerini gerçekleştirme
- Zamanlanmış işlemleri gerçekleştirme

## ANAHTAR KELİMELER

oyun programlama, oyun motoru, model, materyal, doku, kodlama, kullanıcı etkileşimi

HAZIRLIK ÇALIŞMALARI

1. Oyun programlamada kullanılan diller veya araçlar hakkında neler biliyorsunuz? Arkadaşlarınızla paylaşınız.
2. Bir oyun projesinin başarılı olabilmesi için önemli unsurlar nelerdir? Araştırınız.

## 1.1. OYUN GELİŞTİRME UYGULAMASI TEMEL ARAYÜZÜNÜN TANITILMASI

Oyunlar; insanların hayal gücünü, zihinsel aktiviteyi, öğrenmeyi ve etkileşimi bir araya getiren güçlü araçlardır. Günümüzde oyun endüstrisi sürekli büyüyen ve gelişen bir sektör hâline gelmiştir. Sadece büyük oyun stüdyoları değil, bağımsız oyun geliştiriciler de başarılı oyunlar yapma olanağına sahiptir.

**Oyun programlama,** bilgisayar oyunları oluşturmak için kullanılan yazılım ve kodlama tekniklerini içeren bir disiplindir. Oyun programlama; oyun dünyasını canlandıran tüm ögeleri oluşturmayı, yönetmeyi ve koordine etmeyi içerir. Bu ögeler genellikle grafikler, ses efektleri, oyun mekaniği, kullanıcı arayüzü, yapay zekâ ve daha pek çok unsuru kapsar.

Oyun programlama süreci; belirli bir platformda (PC, konsol veya mobil cihazlar gibi) çalışacak bir oyunun temel mantığını ve nasıl çalışacağını, kodlamayı, aynı zamanda kullanıcı deneyimini zenginleştirecek ve oyunun ilgi çekici olmasını sağlayacak detayları ele almayı içerir. Ayrıca performans optimizasyonu, hata ayıklama ve oyunun farklı platformlarda sorunsuz çalışması gibi teknik zorluklar da oyun programlamada dikkat edilmesi gereken diğer unsurlardır.

Oyun programlama, çeşitli programlama dilleri ve oyun motorları kullanılarak gerçekleştirilir. Oyun programlamada kullanılan araç ve kütüphaneler, oyun geliştirme sürecini hızlandırmak ve oyun yapımını daha erişilebilir kılmak için oldukça önemlidir.

**Oyun motoru,** farklı platformlarda çalışabilen ve birden fazla cihaza uyarlanabilen bir geliştirme ortamıdır. Video oyunları, simülasyonlar, sanal gerçeklik, artırılmış gerçeklik ve diğer interaktif deneyimler oluşturmak için kullanılır. Aynı zamanda oyun dünyalarını oluşturmak, animasyonları yönetmek, fizik simülasyonlarını yapmak, sesleri entegre etmek, kullanıcı arabirimleri tasarlamak gibi birçok geliştirme aracını da içerir.

Oyun motoru, görsel programlama ve yazılım dilleri desteği verir. 2B (iki boyutlu) veya 3B (üç boyutlu) oyunlar, eğitim simülasyonları, sanal gerçeklik uygulamaları, artırılmış gerçeklik deneyimleri ve daha pek çok interaktif projeler oyun motoru kullanılarak oluşturulabilir.

Oyun motoru; geliştiricilere iş birliği yapma, hızlı prototipleme, hata ayıklama ve oyunun farklı platformlarda yayımlanması gibi birçok avantaj sunar. Bu nedenle oyun endüstrisinin yanı sıra eğitim, mimari görselleştirme, simülasyon ve diğer interaktif uygulamalar için de oyun motoru kullanılmaktadır.

### 1.1.1. Uygulama Kurulumu

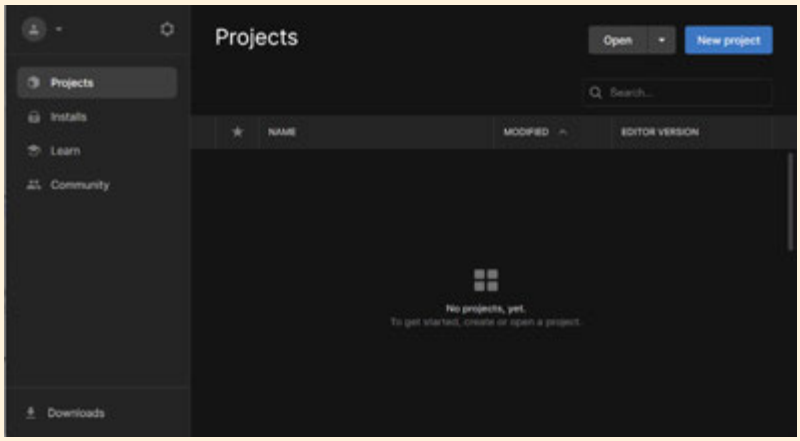
Oyun motoru kurulmadan önce, oyun geliştirme ile ilgili tüm projelerin ve kurulumların yönetimini merkezileştiren oyun motoru hub programının kurulması gerekmektedir. Oyun motoru geliştiricisinin resmî web sitesinden bu program indirilebilmektedir.

1. UYGULAMA

İşlem adımlarına göre bilgisayara oyun motorunun kurulumunu gerçekleştiriniz.

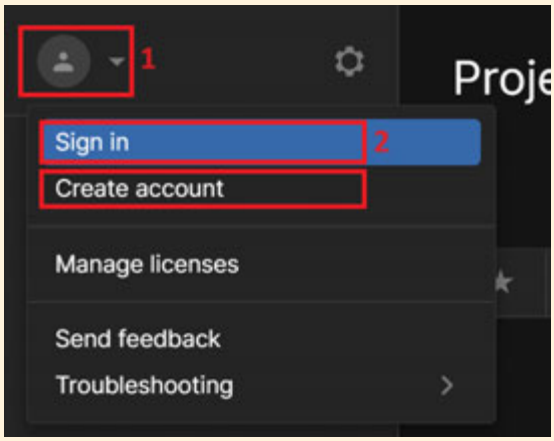
**1. Adım:** Oyun motoru geliştiricisinin resmî web sitesine giriniz.
**2. Adım:** Bilgisayarınıza uygun oyun motoru hub programını indiriniz.
**3. Adım:** Kurulum dosyasını çalıştırarak kurulumu başlatınız.
**4. Adım:** Kurulum sırasında karşınıza gelecek seçenekler içinden uygulamanın kurulacağı dizini sabit diskinizin durumuna göre belirleyiniz.

**5. Adım:** Yüklediğiniz uygulamayı **Başlat** menüsünden veya masaüstü kısayolundan çalıştırınız. Oyun motoru hub programının genel görünümü Görsel 1.1'deki gibidir.




**Görsel 1.1:** Oyun motoru hub programının görünümü

**6. Adım:** Oyun motoru hub programında sol üst köşedeki düğmeye tıklayınız. Daha önceden oluşturmuş olduğunuz bir hesabınız var ise **Sign in** seçeneğine tıklayarak hesabınıza giriş yapınız. İlk kez kaydolacaksanız **Create account** seçeneğine tıklayarak kaydolunuz (Görsel 1.2).




**Görsel 1.2:** Oyun motoru hub programında kullanıcı girişi

**7. Adım:** Programın arayüzünün sol tarafında yer alan **Installs** düğmesine tıklayınız. Daha sonra açılan pencerenin sağ üstünde yer alan **Install Editor** düğmesine tıklayınız.
**8. Adım:** Açılan pencerede oyun motorunun sisteminize uygun güncel LTS (Long Term Support, Uzun Süreli Destek) sürümünü bulunuz ve **Install** düğmesine tıklayarak kurulum işlemini başlatınız.
**9. Adım:** Karşınıza gelen modül ekleme penceresinden kod düzenleyiciyi ve çalışma yapacağınız platform seçeneklerini işaretledikten sonra **Continue** düğmesine basınız.
**10. Adım:** Ekrana gelen lisans sözleşmelerini dikkatlice okuduktan sonra kuruluma devam etmek için onay kutucuğunu işaretleyiniz ve **Continue** düğmesine basınız.
**11. Adım:** Kod düzenleyici yükleyicisi ekrana geldiğinde kurulum işlemi için **Devam** düğmesine basınız.
**12. Adım:** Ekrana gelen yükleme penceresindeki **Oyun** bölümünde yer alan **Oyun motoru ile oyun geliştirme** seçeneğini işaretleyiniz ve **Yükle** düğmesine tıklayınız.
**13. Adım:** Kod düzenleyici için daha önceden oluşturmuş olduğunuz bir hesabınız var ise **Oturum Aç** düğmesine tıklayarak hesabınıza giriş yapınız. İlk kez kaydolacaksanız, **Hesap oluşturun** düğmesine tıklayarak kaydolunuz. **Bunu şimdilik atla** seçeneği ile bu işlemleri daha sonra da gerçekleştirebilirsiniz.

SIRA SİZDE

Oyun motorunun bilgisayarınıza uyumlu farklı bir versiyonunu kurunuz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Oyun motoru hub yazılımında Installs bölümüne girdi. | | |
| 2. Install Editör düğmesine tıkladı. | | |
| 3. Oyun motoru versiyonunu seçti. | | |
| 4. Kurulum için gerekli modülleri seçti. | | |
| 5. Kurulumu başlattı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.1.2. Proje Oluşturma

**Oyun motoru hub programı;** oyun projelerinde ve kurulumlarında gezinme, indirme ve yönetme şeklini kolaylaştıran bağımsız bir uygulamadır. Görsel 1.1'de görüldüğü gibi uygulama ekranında yer alan bağlantılar şunlardır:

1. **Projects:** Yeni bir proje oluşturmak, mevcut projeleri yönetmek veya oyun motorunda bir proje açmak için bu sekme kullanılabilir.
2. **Installs:** Oyun motorunun modüllerini ve sürümlerini yönetme, indirme ve kurmayı sağlar.
3. **Learn:** Bu bölümde, projeler için oyun motorunun nasıl kullanılacağını öğreten ücretsiz öğrenme içeriklerine erişilebilir. Bu içerikler farklı oyun türleri için dersleri ve hazır projeleri kapsar.
4. **Community:** Bu sekme, diğer oyun motoru kullanıcılarıyla etkileşim kurmayı sağlar. Ayrıca bu sekme; oyun motoru geliştirici yetkilileri ile iletişim kurmak, onlara geri bildirimde bulunmak veya onlardan yardım almak için de kullanılabilir.

2. UYGULAMA

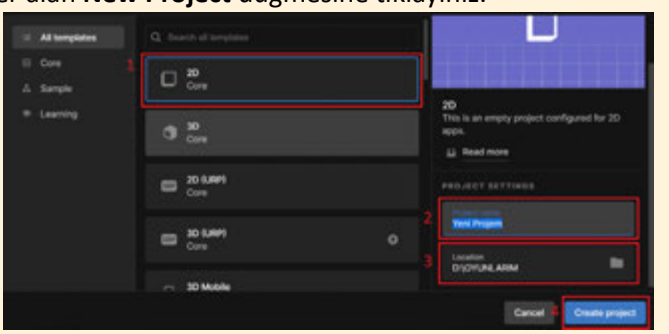
İşlem adımlarına göre yeni bir 2D oyun projesi oluşturunuz.

**1. Adım:** Oyun motoru hub programını **Başlat** menüsünden veya masaüstü kısayolundan çalıştırınız.

**2. Adım:** Programın arayüzünün sol tarafında yer alan **Projects** düğmesine tıklayınız.

**3. Adım:** Açılan pencerenin sağ üstünde yer alan **New Project** düğmesine tıklayınız.

**4. Adım:** Açılan pencerenin orta kısmında yer alan 2D (Core) seçeneğine tıklayınız. Pencerenin sağ tarafında yer alan proje ayarları bölümünde projeye bir isim veriniz ve projeyi oluşturacağınız konumu belirleyip pencerenin sağ alt tarafında yer alan Create project düğmesine tıklayarak projenizi oluşturunuz (Görsel 1.3).




**Görsel 1.3:** Oyun motoru hub programı ile yeni proje oluşturulması

SIRA SİZDE

Bilgisayarınızın masaüstünde **Oyun** isimli bir klasör oluşturunuz. Bu klasör içinde **ilkOyunum** isminde yeni bir 3D oyun projesi oluşturunuz.

**Değerlendirme**

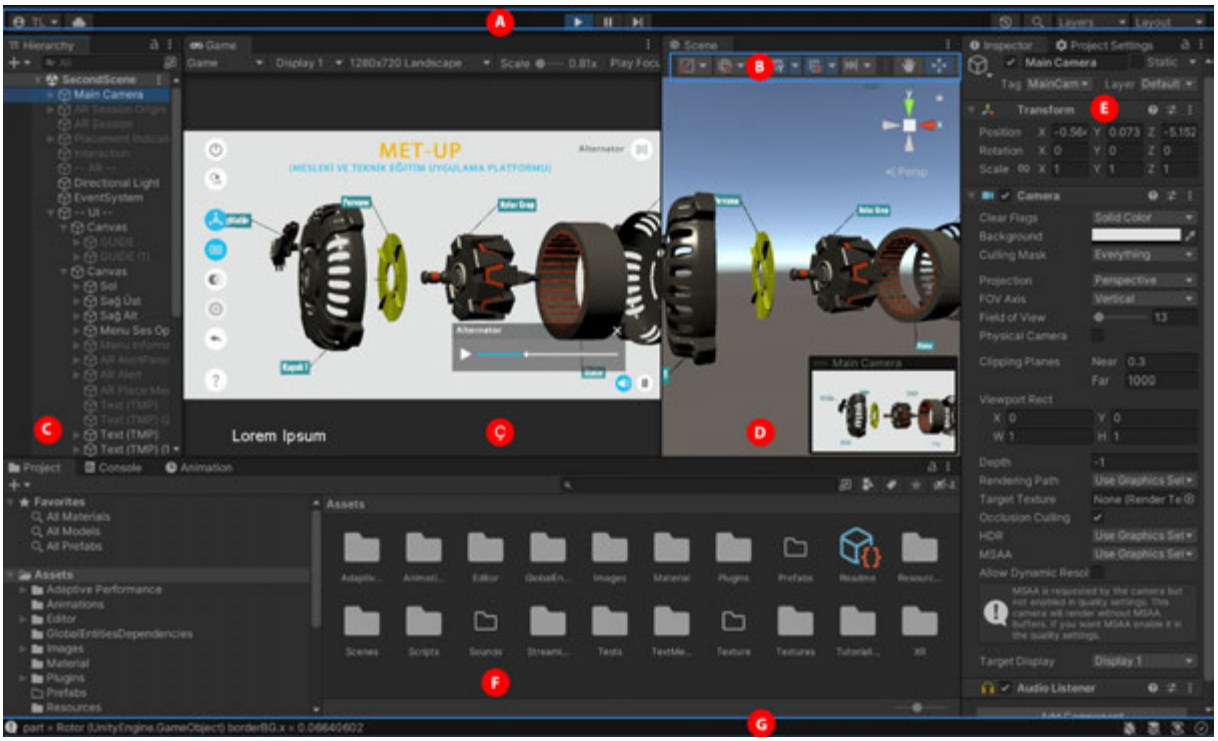
Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Oyun motoru hub programında projeler bölümüne girdi. | | |
| 2. 3D proje türünü seçti. | | |
| 3. Proje ayarları bölümünden projeye isim verdi. | | |
| 4. Proje ayarları bölümünden projenin oluşturulacağı dizini oluşturdu. | | |
| 5. Create Project düğmesine basarak projeyi ilgili klasörde oluşturdu. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.1.3. Uygulama Arayüzü

Oyun motorunun çalışma alanında kullanıcılara varsayılan olarak sunulan bir dizi panel bulunmaktadır (Görsel 1.4). Bu panellerin düzeni, kullanıcıların tercihlerine göre özelleştirilebilir. Bazı paneller kapatılabilirken kapalı olanlar daha sonra tekrar açılabilir ve sürükle bırak yöntemi ile panellerin yerleri değiştirilebilir. Kullanıcılar, çalışma düzenlerini kişisel ihtiyaçlarına göre ayarlayarak daha rahat bir şekilde çalışabilirler.




**Görsel 1.4:** Oyun motorunun arayüzü

Görsel 1.4'te harfler ile belirtilen paneller şunlardır:

**A) Toolbar:** Bu alandan hesaba ve bulut hizmetlerine erişim sağlanabilir. Ayrıca oynatma modu için kontrol düğmeleri, geçmişi geri alma düğmesi, arama düğmesi, katmanların görünüm ayarları ve panel düzenleyici ayarları bu alan üzerinde yer almaktadır.

**B) Overlays:** Sahne görünümünü ve sahnenin içindeki nesneleri değiştirmek için temel araçları içerir. Bu alandan, iş akışının iyileştirilmesi için özel katmanlar da alana eklenebilir.

**C) Hierarchy:** Proje içerisinde kullanılan oyun nesnelerinin hiyerarşik yapıda ve metinsel olarak gösterildiği paneldir. Sahne içerisinde oluşturulan bir nesne veya sahneye sürükle bırak ile yerleştirilen bir asset (varlık), bu alana otomatik olarak eklenecektir. Hierarchy paneli, nesnelerin birbirine nasıl bağlandığının yapısını göstermektedir.

**Ç) Game:** Oyun görünümü, oyunun son hâlinin sahne kameraları aracılığıyla nasıl görüneceğini simüle eder. Oyunun görüntüsü, çeşitli çözünürlük seçenekleri ile buradan kontrol edilebilir. Oynat düğmesine tıklandığında proje çalıştırılır ve simülasyon başlar.

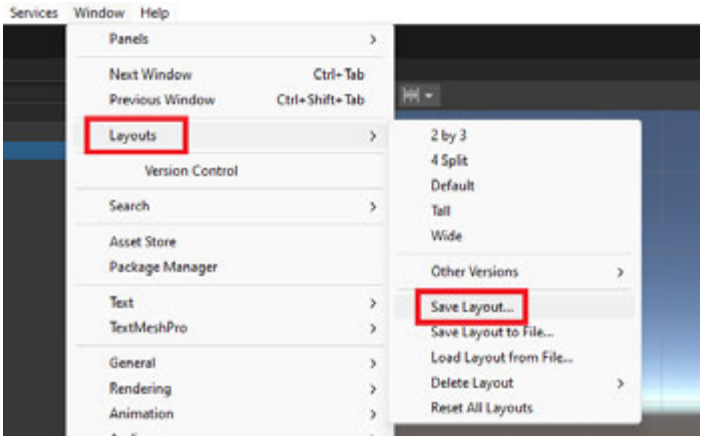
**D) Scene:** Sahne görünümü, sahnede görsel olarak gezinmeye ve düzenleme yapmaya olanak sağlar. Üzerinde çalışılan proje türüne bağlı olarak 2B veya 3B perspektif görüntüleme yapılabilir.

**E) Inspector:** Seçili olan nesnenin tüm bileşenlerini ve özelliklerini görüntülemeyi ve düzenlemeyi sağlar. Farklı nesne türlerinin farklı özellik kümeleri olduğundan, farklı bir nesne seçildiğinde **Inspector** panelinin düzeni ve içeriği değişir.

**F) Project:** Projede kullanılabilen varlık kitaplığını görüntüler. Varlıklar projeye aktarıldığında ilk olarak burada görünür. Daha sonra varlıklar sahneye veya **Hierarchy** alanına sürüklenip kullanılabilir.

**G) Status Bar:** Bu alanda, çeşitli işlemler hakkında bildirimler görüntülenir. Ayrıca ilgili araçlara ve ayarlara hızlı erişim sağlanabilir.

Çalışma alanında yapılan değişiklikler Görsel 1.5'te görüldüğü gibi Window>Layouts>Save Layout ya da Toolbar üzerinde sağ tarafta yer alan Layout>Save Layout komutu kullanılarak daha sonra kullanılmak üzere kaydedilebilir. Layout menüsü altında uygulama tarafından hazır olarak sunulan görünüm türleri de kullanılabilir.




**Görsel 1.5:** Oyun motoru görünüm ayarları

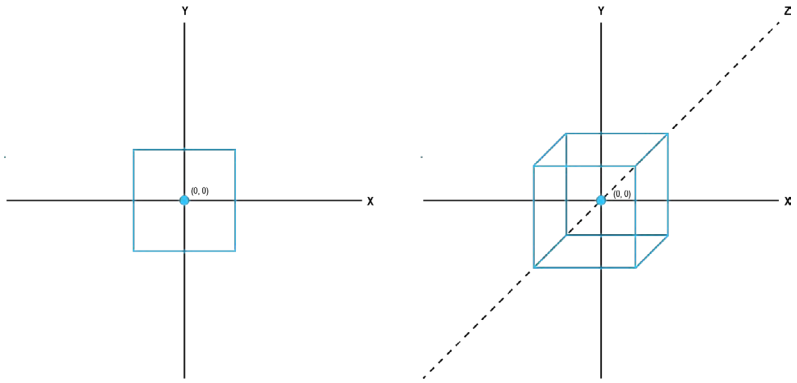
## 1.1.4. Oyun Nesneleri

Bir oyun projesinin temel bileşenleridir. Oyun geliştirme sürecinde, oyun dünyasını oluşturan ve etkileşimde bulunulan her öge **Oyun Nesneleri (Game Objects)** olarak adlandırılır. Oyun motoru içindeki **GameObject** sınıfıyla temsil edilen bu nesneler, bir oyunun yapı taşlarını oluşturur ve çeşitli bileşenlerle bir araya gelerek oyun dünyasını canlandırır. Oyun motorunda oyun nesneleriyle çalışmaya başlanmadan önce 2B ve 3B koordinat sistemleri hakkında bilgi sahibi olunması gerekir.

**Boyutlar ve Koordinat Sistemi**

Oyun geliştiricilerin oyun motorunda sahneleri oluştururken sahneye nesneleri yerleştirmeleri, konumlandırmaları ve düzenlemeleri için temel bir kavramdır.

Oyun motorunda hem 2B hem de 3B oyun geliştirilebilir. 2B oyunlar için yalnızca yatay (X) ve dikey (Y) eksenler kullanılır, derinlik (Z) ekseni kullanılmaz. 3B oyunlar 2B oyunlara benzer fakat Z eksenini de içerir (Görsel 1.6).




**Görsel 1.6:** 2 boyutlu bir kare ve 3 boyutlu bir küp

Oyun motorunun koordinat sisteminde X ekseni sağa-sola, Y ekseni yukarıya-aşağıya ve Z ekseni ileri-geri yönlere karşılık gelmektedir. X, Y ve Z değerleri bir nesnenin oyun dünyasındaki konumunu ifade eder. Örneğin bir nesnenin konumu **5, 7, 1 ise** bu nesnenin X ekseninde **5 birim sağda,** Y ekseninde **7 birim yukarıda** ve Z ekseninde **1 birim ileride** konumlandığı anlamına gelir. Başka bir ifadeyle bu nesnenin yatayda **5 birim sağda,** dikeyde **7 birim yukarıda** ve derinlikte **1 birim ileride** konumlandığı söylenebilir.

NOT

Oyun motorunda global (dünya) koordinat sistemi ve local (yerel) koordinat sistemi kullanılmaktadır. Bunlar nesnelerin konumlarını belirtmek için kullanılan iki farklı referans sistemidir. Bir oyun nesnesinin dünya koordinatları, sahnenin merkezine göre belirlenir. Yerel koordinatları ise oyun nesnesinin kendi merkezine göre belirlenir.

Oyun motorunda **oyun nesneleri,** oyunu oluşturan temel yapı taşlarıdır. Oyun motorundaki resim, şekil, ışık, kamera, ses, video nesnelerinin her biri birer oyun nesnesidir. Oyun sahnesine oyun nesnesi eklemek için **GameObject** menüsü veya **Hierarchy** panelinde farenin sağ tuşuna tıklandığında gelen menü kullanılır. Oyun motorunda **Component (Bileşen),** oyun nesnelerine özellik veya davranış eklemek için kullanılan yapıdır. Bir oyun nesnesine birden çok Component eklenebilir. Componentleri eklemek ve yönetmek için **Inspector** paneli kullanılır.

3. UYGULAMA

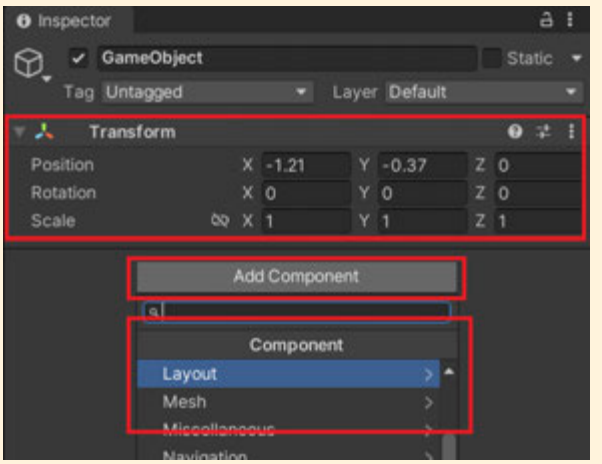
İşlem adımlarına göre sahneye bir adet boş oyun nesnesi ve bir adet Capsule nesnesi ekleyiniz.

**1. Adım:** Yeni bir 3B oyun projesi oluşturunuz veya mevcut bir projeniz varsa bu projeyi açınız.
**2. Adım:** GameObject menüsünden Create Empty'i seçerek boş bir oyun nesnesi ekleyiniz (Görsel 1.7).



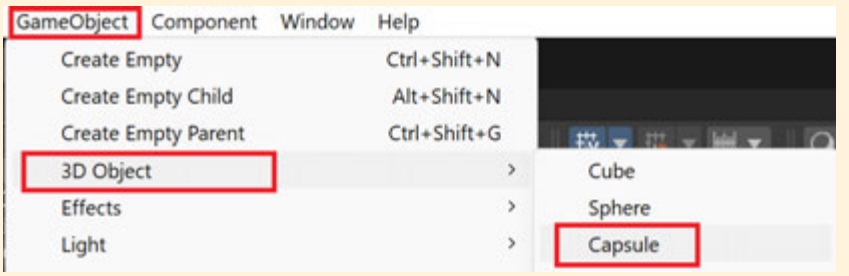

**Görsel 1.7:** Boş nesne eklenmesi

**Hierarchy** panelinde GameObject isminde bir nesne oluşacaktır. Bu nesne seçildiğinde **Inspector** panelinde bu nesneye ait sadece **Transform** bileşeni görülecektir. Hierarchy panelinde **Add Component** düğmesine tıklandığında oyun nesnesine eklenebilecek bütün bileşenler görülebilmektedir (Görsel 1.8).



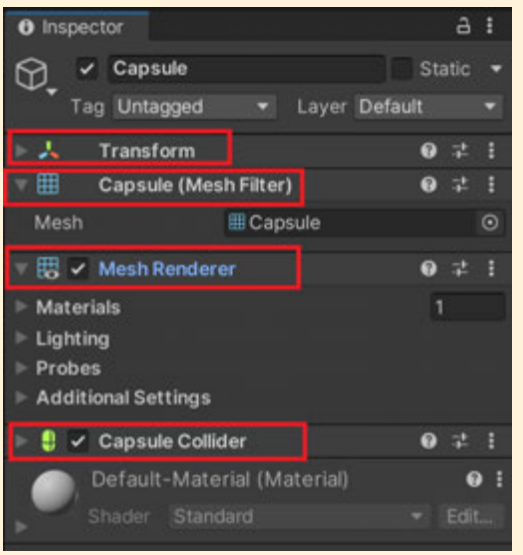

**Görsel 1.8:** Oyun nesnesine ait Inspector paneli

**3. Adım: GameObject>3D Object** menüsünden **Capsule**'ü seçerek bir kapsül ekleyiniz (Görsel 1.9).




**Görsel 1.9:** Capsule nesnesinin eklenmesi

Capsule oyun nesnesinin boş oyun nesnesi gibi Transform bileşeni olmakla birlikte Mesh Filter, Mesh Renderer, Capsule Collider gibi bileşenlerinin de olduğu görülecektir (Görsel 1.10).




**Görsel 1.10:** Capsule nesnesinin bileşenleri

NOT

Fare; sahnenin üzerindeyken klavyeden **Alt tuşuna** basılı tutup, **farenin sol tuşuna** basıp fareyi sürüklediğinizde sahnedeki oyun nesnelerini farklı açılardan görebilirsiniz. Sadece **farenin sağ tuşuna** basıp fareyi sürüklediğinizde imlecin konumuna göre sahneyi farklı açılardan görebilirsiniz. Farenin **döndürme (scroll) tuşuna** basarak fareyi sürüklediğinizde sahne görünümünü hareket ettirirsiniz. Farenin döndürme tuşunu döndürdüğünüzde ise sahneyi yaklaştırabilir ve uzaklaştırabilirsiniz.

SIRA SİZDE

Oyun sahnesine küp ve silindir ekleyiniz. Fare ve klavyeyi kullanarak sahne görünüm işlemlerini yapınız.

**Değerlendirme**

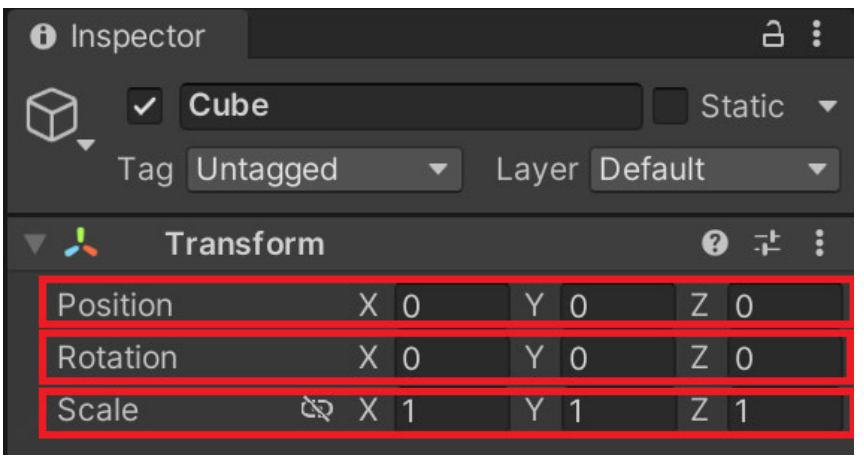
Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Oyun sahnesine küp nesnesi ekledi. | | |
| 2. Oyun sahnesine silindir nesnesi ekledi. | | |
| 3. Fare ve klavyeyi kullanarak sahne görünüm işlemlerini gerçekleştirdi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.1.5. Nesne Dönüşümleri

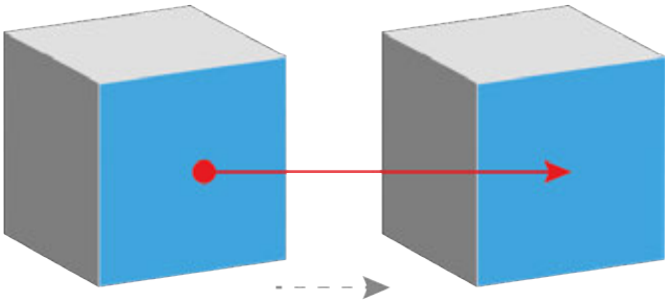
Oyun motorunda, sahnedeki tüm oyun nesnelerinin koordinat sistemine göre bir konumu (Position), rotasyonu (Rotation) ve ölçeği (Scale) vardır. Oyun nesnelerinin **Transform bileşeni** ile oyun nesnelerinin konumunu, rotasyonunu ve ölçeğini yönetebilirsiniz ve bunların değerlerini değiştirebilirsiniz (Görsel 1.11). Bu işlemlerin genel ismine **nesne dönüşümleri** denir.




**Görsel 1.11:** Nesne dönüşüm değerleri

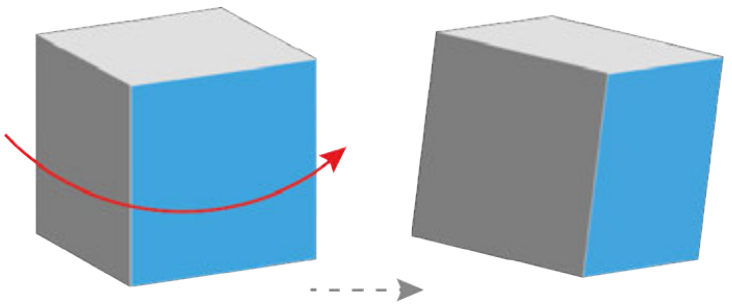
Bu dönüşümlerin genel özellikleri şunlardır:

**Position:** Oyun nesnesinin dünya koordinatlarındaki konumunu belirtir. Üç eksendeki konum değerlerini ifade eden X, Y ve Z değerleri vardır. Nesnenin sahnede belirlenen bir konuma taşınması (translate) için kullanılır (Görsel 1.12).



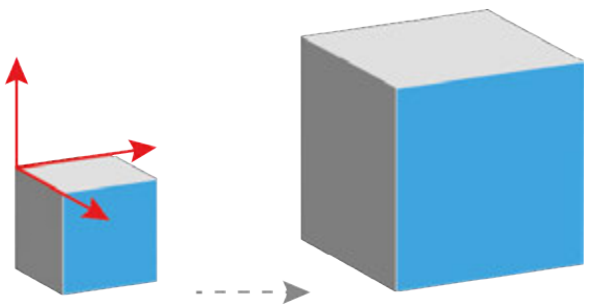
**Görsel 1.12:** Nesnenin taşınması

**Rotation:** Oyun nesnesinin dünya koordinat sistemine göre rotasyonunu (dönüşünü) belirtir. Üç eksendeki rotasyon değerlerini ifade eden X, Y ve Z değerleri vardır. Nesneyi belirlenen eksenlerde döndürmek için kullanılır (Görsel 1.13).



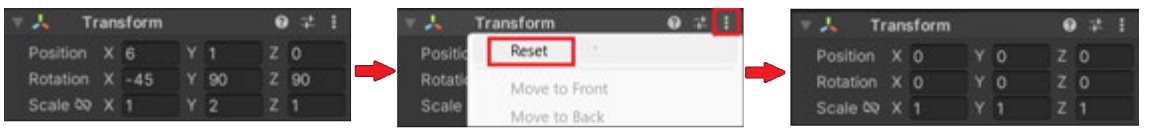
**Görsel 1.13:** Nesnenin döndürülmesi

**Scale:** Oyun nesnesinin ölçeğini belirtir ve nesnenin boyutunu kontrol eder. Üç eksendeki Scale değerlerini ifade eden X, Y ve Z değerleri vardır. Nesnenin boyutunu belirlenen eksenlerde büyütmek için kullanılır (Görsel 1.14).



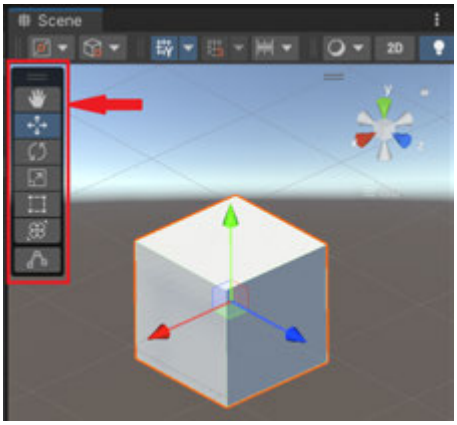
**Görsel 1.14:** Nesnenin büyütülmesi

Oyun nesnesinin Transform bileşinin en sağ üstünde bulunan üst üste üç noktaya tıklanıp, açılan menüde Reset düğmesine tıklandığında nesnenin bütün Transform değerleri sıfırlanır (Görsel 1.15).




**Görsel 1.15:** Transform değerlerinin sıfırlanması

Bir oyun nesnesinin Transform değerleri Transform araç kutusundaki araçlarla da değiştirilebilir (Görsel 1.16).

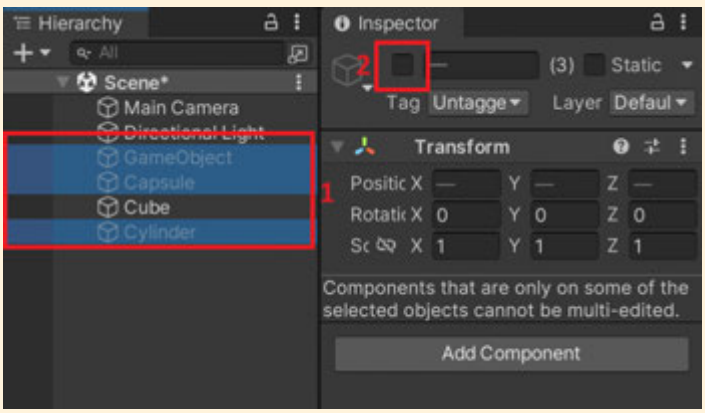



**Görsel 1.16:** Transform araç kutusu

## 4. UYGULAMA

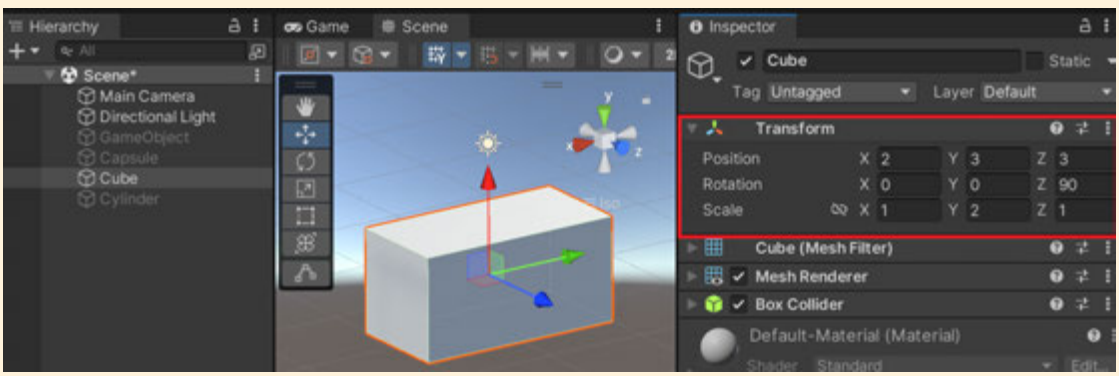
İşlem adımlarına göre Cube nesnesinin dönüşüm değerlerini değiştiriniz.

**1. Adım:** Oyun motorunda daha önceden oluşturduğunuz projenizi açınız. Hierarchy panelinde Cube nesnesi haricinde eklediğiniz diğer nesneleri klavyeden **Ctrl** tuşuna basarak fare ile seçiniz ve Inspector panelinde nesne aktiflik seçeneğini iptal ediniz. Seçtiğiniz nesneler pasif olacaktır (Görsel 1.17).



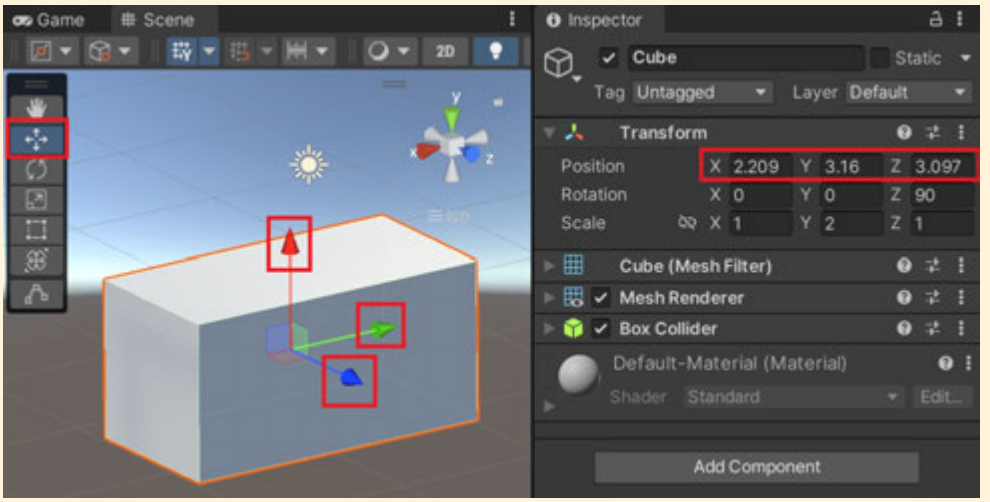

**Görsel 1.17:** Nesnelerin pasif yapılması

**2. Adım:** Cube nesnesini seçiniz ve bu nesnenin Transform değerlerini sıfırlayınız. Bu nesneye ait Transform bileşenindeki **Position, Rotation** ve **Scale**'in X, Y ve Z değerlerini klavyeden değer girerek değiştiriniz. Her değeri girdikten sonra nesnenin konumu, rotasyonu ve ölçeğindeki değişiklikleri gözlemleyiniz (Görsel 1.18).



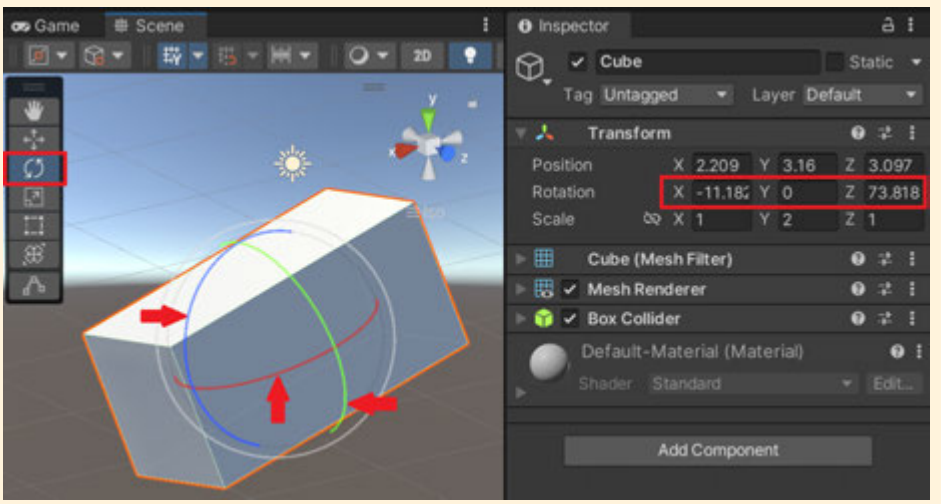

**Görsel 1.18:** Küpün Transform değerlerinin değiştirilmesi

**3. Adım: Cube** nesnesini seçiniz. Nesnenin Transform değerlerini resetleyiniz. Transform araç kutusundaki **Move Tool** aracına tıklayınız. Nesnenin üzerinde görünen kırmızı (X ekseni), yeşil (Y ekseni) ve mavi (Z ekseni) oklara sırasıyla farenin sol tuşu ile basılı tutarak fareyi farklı yönlerde sürükleyiniz. Bu işlemler sonucunda nesnenin ilgili eksenlerde hareket ettiğini ve Transform bileşeninde ilgili **Position** değerlerinin değiştiğini gözlemleyiniz (Görsel 1.19).




**Görsel 1.19:** Move Tool aracının kullanımı

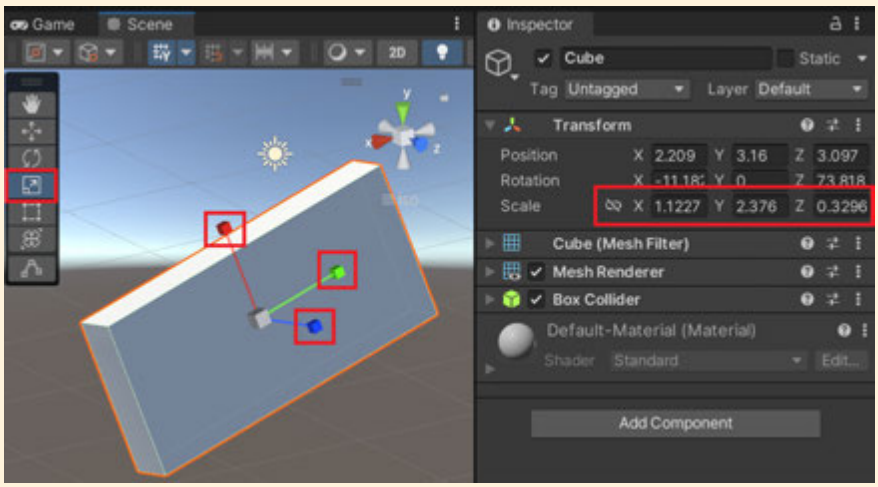
**4. Adım:** Cube nesnesini seçiniz. Transform araç kutusundaki **Rotate Tool** aracına tıklayınız. Nesnenin üzerinde görünen kırmızı (X ekseni), yeşil (Y ekseni) ve mavi (Z ekseni) yaylara sırasıyla farenin sol tuşu ile basılı tutarak fareyi farklı yönlerde sürükleyiniz. Bu işlemler sonucunda nesnenin ilgili eksenlerde döndüğünü ve Transform bileşeninde ilgili Rotation değerlerinin değiştiğini gözlemleyiniz (Görsel 1.20).




**Görsel 1.20:** Rotate Tool aracının kullanımı

**5. Adım:** Cube nesnesini seçiniz. Transform araç kutusundaki **Scale Tool** aracına tıklayınız. Nesnenin üzerinde görünen kırmızı (X ekseni), yeşil (Y ekseni) ve mavi (Z ekseni) çizgilere sırasıyla farenin sol tuşu ile basılı tutarak fareyi farklı yönlerde sürükleyiniz.

Bu işlemler sonucunda nesnenin boyutunun ilgili eksenlerde değiştiğini ve Transform bileşeninde ilgili Scale değerlerinin değiştiğini gözlemleyiniz (Görsel 1.21).




**Görsel 1.21:** Scale Tool aracının kullanımı

SIRA SİZDE

**1.** Sahnedeki Cube nesnesini pasif yapınız.

**2.** Sahneye bir Cylinder (silindir) nesnesi ekleyiniz.

**3.** Bu nesnenin **Transform** değerlerini ilk önce **Inspector panelini** sonra **Transform araç kutusunu** kullanarak değiştiriniz.

**4.** Nesnenin Transform değerlerini sıfırlayınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

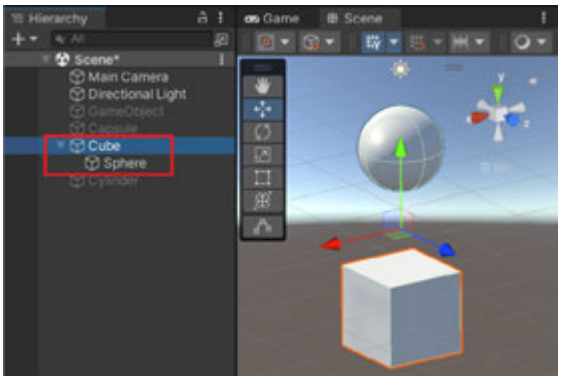
**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Cube (küp) nesnesini pasif yaptı. | | |
| 2. Sahneye bir Cylinder (silindir) nesnesi ekledi. | | |
| 3. Nesnenin Transform değerlerini Inspector panelinde değiştirdi. | | |
| 4. Nesnenin Transform değerlerini Transform araç kutusunu kullanarak değiştirdi. | | |
| 5. Nesnenin Transform değerlerini sıfırladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

NOT

Hierarchy panelindeki bir oyun nesnesinin üzerinde farenin sağ tuşuna tıklanıp gelen menüdeki **Rename** komutuyla oyun nesnesinin ismi değiştirilebilir. Bir veya birden fazla nesne seçilip bu nesne veya nesneler sırasıyla **Ctrl+X, Ctrl+C, Ctrl+V** klavye tuşları ile kesilebilir, kopyalanabilir ve yapıştırılabilir. **Ctrl+D** tuşlarına basılarak da çoğaltılabilir.

Hierarchy panelinde bulunan bir oyun nesnesi fare ile sürüklenip başka bir oyun nesnesinin üstüne bırakılarak içine yerleştirilebilir. En üst düzeydeki nesne **ebeveyn (parent)** olarak isimlendirilir. Ebeveyn içine yerleştirilen nesne ise **çocuk (child)** olarak isimlendirilir. Ayrıca **Hierarchy** panelinde bulunan bir oyun nesnesi üzerinde farenin sağ tuşu ile tıklanıp açılan menüden herhangi bir oyun nesnesi seçildiğinde, seçilen oyun nesnesi ilgili nesnenin içine çocuk nesnesi olarak eklenir (Görsel 1.22).




**Görsel 1.22:** Ebeveyn (parent) ve çocuk (child) nesneleri

Ebeveyn nesnesi taşınabilir, döndürülebilir ve ölçeklendirilebilir. Çocuk nesnesi de bu dönüşüm işlemlerinden etkilenir. Çocuk nesnesi, ebeveyni ile beraber taşınır, döner ve nesnenin boyutları değişir fakat bu durumda ebeveyn nesnesinin Transform değerleri değişirken çocuk nesnesinin bu değerleri değişmez. Bunun nedeni çocuk nesnesinin Transform değerlerinin dünya koordinatlarına değil, kendi ebeveyninin koordinat sistemine göre belirlenmesidir.

NOT

Sahneye eklenen oyun nesneleri çoğaldıkça bazen sahnedeki oyun nesnelerini bulmak ve yakından görmek zor olabilir. Bu durumda hangi nesne sahnede yakından görülmek isteniyorsa Hierarchy panelinde bu nesneye **çift tıklandığında** veya bu nesne seçilip fare sahne üzerine getirildikten sonra klavyeden **F tuşuna** basıldığında ilgili nesne sahnede daha yakından görülebilir.

SIRA SİZDE

**1.** Sahneye boş bir oyun nesnesi ekleyiniz ve nesnenin ismini ebeveyn olarak değiştiriniz.

**2.** Bu nesnenin içine bir silindir nesnesi ekleyiniz.

**3.** Silindir nesnesini seçip klavyeden **Ctrl+D** tuşlarına basınız.

**4.** Ebeveyn isimli nesnenin Transform değerlerini değiştiriniz, ebeveyn ve silindir nesnelerinin Transform değerlerindeki değişiklikleri gözlemleyiniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

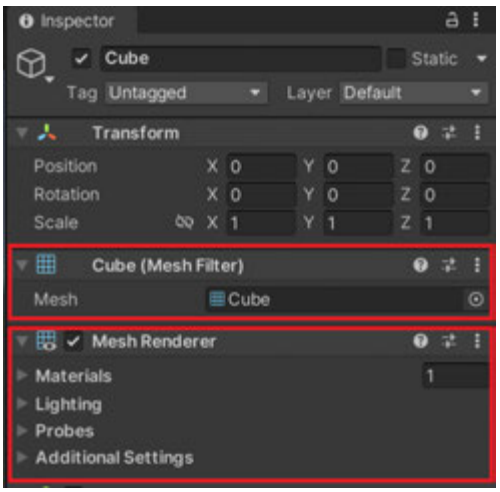
**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Sahneye boş bir oyun nesnesi ekleyip ismini ebeveyn olarak değiştirdi. | | |
| 2. Boş oyun nesnesinin içine bir silindir nesnesi ekledi. | | |
| 3. Silindir nesnesini çoğaltı. | | |
| 4. Ebeveyn nesnesinin Transform değerlerini değiştirip değişiklikleri gözlemledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

### 1.1.6. Model Ekleme ve Düzenleme

Oyun programlamada bir model, genellikle bir 3B nesneyi temsil etmekte ve **3B model** olarak da ifade edilmektedir. **3B model** üç boyutlu bir nesnenin dijital temsilidir. 3B modeller, genellikle üç boyutlu bir modelleme yazılımı kullanılarak oluşturulmakta ve bu modeller oyun geliştirme, animasyon, simülasyon ve benzeri uygulamalarda kullanılmaktadır. Oyun motorunda modellerin kullanılması, oyun geliştirme sürecinde önemli bir rol oynamaktadır.

Bir 3B modelin görünüşünü ve şeklini tanımlayan yapıya **Mesh (ağ, örgü)** denir. Mesh, üçgenlerden ve noktalardan oluşan bir ağdır. Bu üçgenler noktaların birleşmesinden oluşur. Her üçgen, 3B modelin belirli bir bölgesini temsil eder ve bu üçgenler bir araya gelerek nesnenin genel formunu (şeklini) oluşturur. Oyun motorunda, 3B modeller genellikle oyun nesnelerinin üzerine eklenen **Mesh Filter** ve **Mesh Renderer** bileşeni ile görsel olarak temsil edilir (Görsel 1.23).




**Görsel 1.23**: Mesh Filter ve Mesh Renderer bileşeni

Mesh Filter bileşeni, bir 3B nesnenin şeklini tanımlayan mesh bilgisini içerir. Mesh Renderer bileşeni ise modelin ışığa karşı nasıl tepki vereceğini, malzeme özelliklerini ve görünen diğer özellikleri kontrol ederek sahnede nasıl görüneceğini belirler.

Oyun motorunda küp, küre, kapsül, silindir, düzlem gibi temel 3B nesneler sahneye eklenip kullanılabilir. Bu temel şekiller tek olarak kullanabileceği gibi bir ebeveyn nesnenin içine bu şekillerden bir veya birden fazla eklenerek de farklı 3B nesneler oluşturabilir. Bununla birlikte bir 3B modelleme yazılımı ile oluşturulmuş 3B modeller de oyun motorunda içe aktarılıp kullanılabilir. Oyun motoru fbx, obj, dae, 3ds, dxf formatlarındaki 3B modellerini desteklemektedir.

3B modelleme yazılımı ile oluşturulan bir modelin oyun programına eklenmesi için model dosyası, **Assets (Varlıklar)** klasörüne veya bu klasör altındaki başka bir klasöre yerleştirilir. Daha sonra bu model sürüklenerek **Hierarchy** paneline veya sahneye bırakıldığında model içe aktarılmış olur.

5. UYGULAMA

İşlem adımlarına göre 3B modelleme yazılımı ile oluşturulan bir modeli oyun projesinde içe aktarınız ve gerekli düzenlemeleri yapınız.

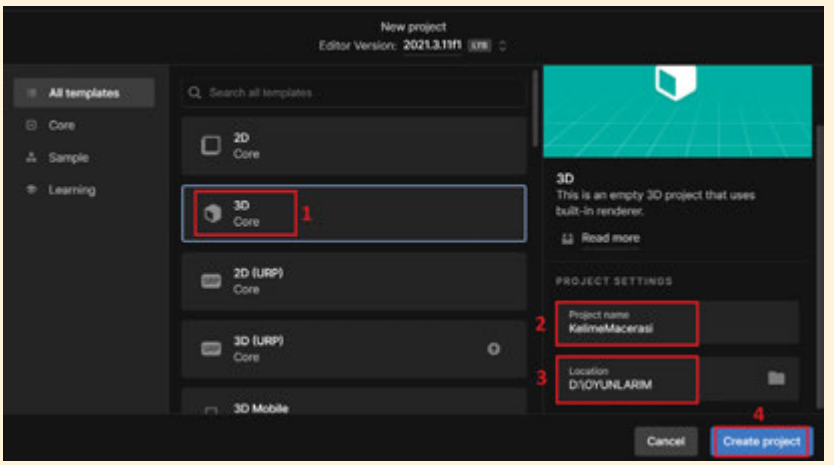
NOT

Kitabın ilerleyen konularında yapılacak uygulamalar bu uygulama adımında oluşturulacak oyun projesinin devamı niteliğinde olacağından bu uygulamada oluşturacağınız oyun projesinin saklanması önemlidir.

**1. Adım:** Yeni bir 3B oyun projesi oluşturunuz ve projenizin ismini **KelimeMacerasi** olarak belirleyiniz (Görsel 1.24).

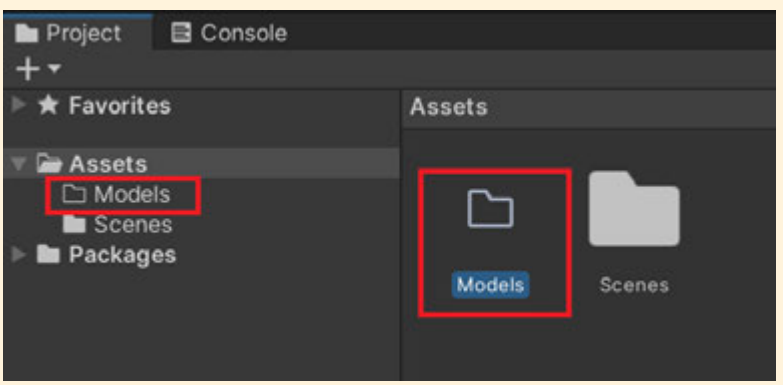
NOT

**KelimeMacerasi** projesinde oyun başlatıldığında oyun içinde kullanıcıya ekranda bir kelime gösterilecektir. Kullanıcı, oyun karakterini doğru veya yanlış kürelerine temas ettirerek bu kelimenin yazımının doğru veya yanlış olduğunu bulmaya çalışacaktır. Kullanıcının seçtiği zorluk seviyesine göre kelime sayısı belirlenecek ve bu kelimeler kullanıcıya sırayla gösterilecektir. Kullanıcı karakter tipini seçebilecektir. Doğru cevaplarda +10 puan, yanlış cevaplarda -2 puan verilecektir ve oyunun oynanması için belirli bir süre olacaktır.




**Görsel 1.24:** Yeni bir 3B proje oluşturulması

**2. Adım:** Oyun motorunda **Project** panelinde **Assets** klasörü üzerinde farenin sağ tuşuna tıklayınız. Açılan menüde **Create>Folder** seçeneği ile **Assets** klasörü altında **Models** isminde bir klasör oluşturunuz (Görsel 1.25).

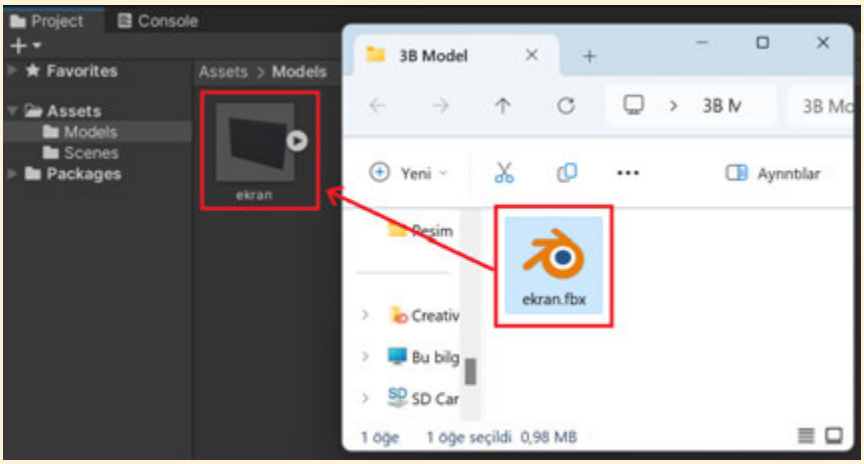



**Görsel 1.25:** Models klasörünün oluşturulması

NOT

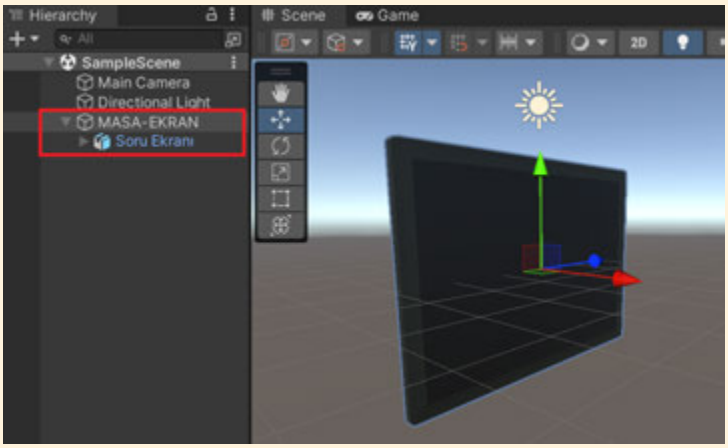
Bir sonraki işlem adımında kullanılacak ekran.fbx dosyasına ulaşmak için yandaki karekodu tarayınız.

**3. Adım: ekran.fbx** dosyasının bulunduğu pencereyi açınız ve bu dosyayı sürükleyerek Models klasörü içine yerleştiriniz (Görsel 1.26).



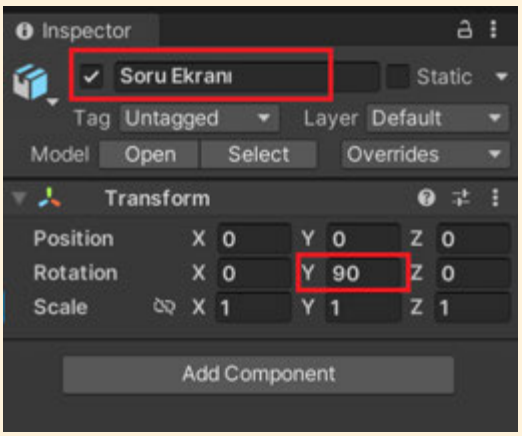

**Görsel 1.26:** 3B Modelin projeye eklenmesi

**4. Adım: Hieararchy** panelinde **MASA-EKRAN** isminde boş bir oyun nesnesi oluşturunuz.
**5. Adım: Models** klasöründeki ekran modelini sürükleyerek **MASA-EKRAN** oyun nesnesinin üstüne bırakıp içine yerleştiriniz. İsmini **Soru Ekranı** olarak değiştiriniz (Görsel 1.27).



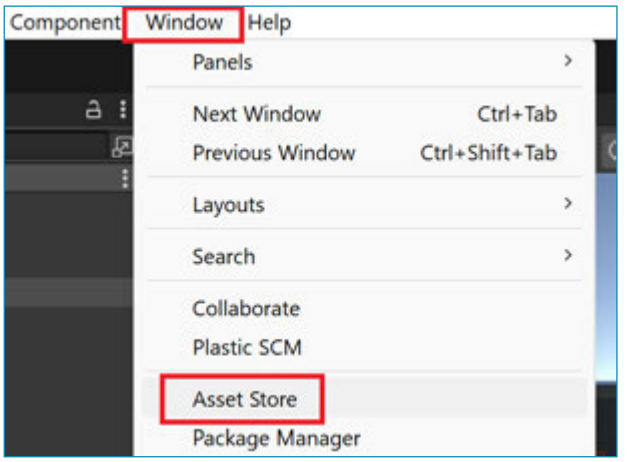

**Görsel 1.27:** Modelin sahneye eklenmesi

**6. Adım:** Soru Ekranı nesnesini seçip Rotation Y eksen değerini **90** yapınız (Görsel 1.28).




**Görsel 1.28**: Modelin Y ekseninde döndürülmesi

Oyun motorunun varlık mağazası **Asset Store**'dan, oyun motorunda kullanılmak üzere modeller indirilebilir. Bu modeller ücretli veya ücretsiz olabilir. Asset Store'u web tarayıcısında açmak için **Window** menüsünden **Asset Store** seçeneğine tıklanır (Görsel 1.29).

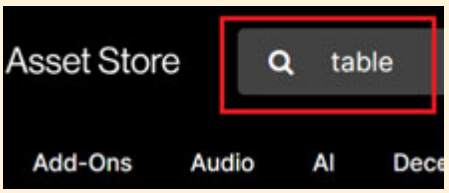



**Görsel 1.29:** Asset Store seçeneği

## 6. UYGULAMA

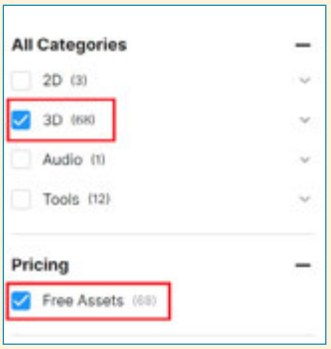
İşlem adımlarına göre **Asset Store**'dan bir masa modelini indiriniz ve bu modeli oyun projesinde içe aktarıp gerekli düzenlemeleri yapınız.

**1. Adım:** KelimeMacerasi isimli projenizde **Window** menüsünden **Asset Store** seçeneğine tıklayınız. Web tarayıcınızda Asset Store web sitesi açılacaktır. Bu varlık mağazasından oyun motorunda kullanmak üzere kendi varlık listenize varlık (model, hazır çözümler, vb.) ekleyebilmek için bu platforma oyun motoru hesap bilgilerinizle giriş yapmanız gerekmektedir.
**2. Adım:** Projeye bir masa modeli eklemek için Arama kutusuna **table** yazıp **Enter**'a basınız (Görsel 1.30).



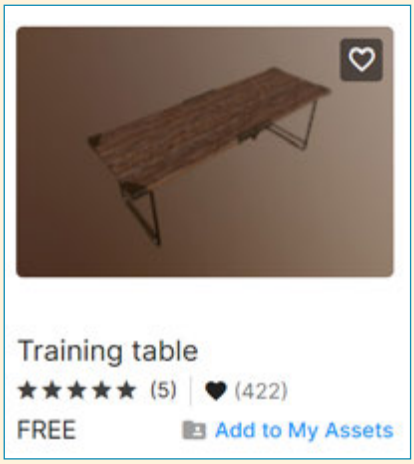

**Görsel 1.30:** Asset Store'da arama yapılması

**3. Adım:** Sağ taraftaki kategorilerden **3D**'yi, ücretsiz model için de **Free Assets**'i seçiniz (Görsel 1.31).



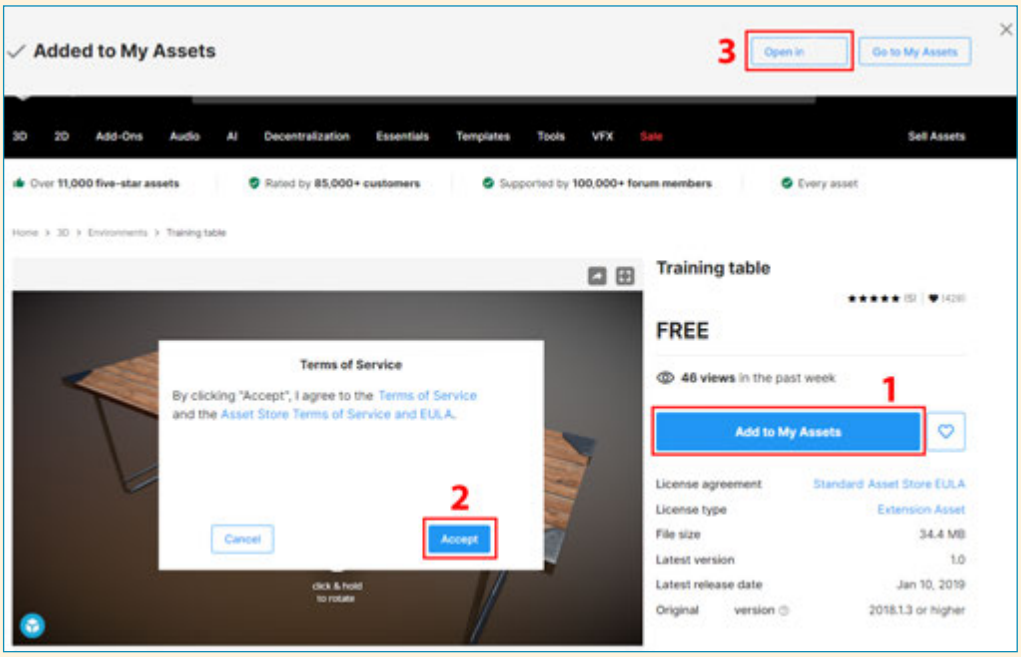

**Görsel 1.31:** Filtreleme yapılması

**4. Adım:** Çıkan sonuçlardan bir masa modeline tıklayınız (Görsel 1.32).




**Görsel 1.32:** Masa modeli varlığı

**5. Adım:** Açılan sayfanın sağ tarafındaki **Add to My Assets** düğmesine tıklayınız. Sayfanın ortasında açılan pencerede **Accept** düğmesine tıklayınız. Sayfanın üstünde açılan alanda **Oyun Motorunda Aç düğmesine** tıklayınız (Görsel 1.33).



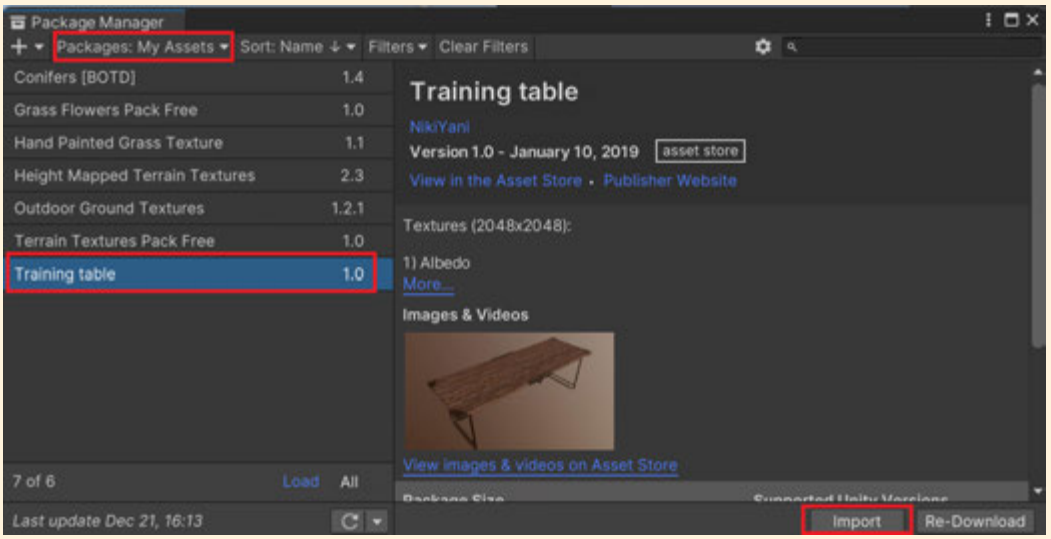

**Görsel 1.33:** Varlığın oyun motorunda açılması

**KelimeMacerasi** isimli projenizde **Package Manager (Paket Yöneticisi)** açılacaktır. Oyun motoru (Hub) üzerinden oyun motoru hesap bilgilerinizle giriş yapmanız gerektiğini unutmayınız.
**6. Adım:** Package Manager penceresinde **Packages: My Assets** ifadesi ile bu pencerede varlıklarınızın listelendiği belirtilmektedir. Bu listede varlık mağazasından eklediğiniz **Training table** isimli varlığın olduğunu gözlemleyiniz.

NOT

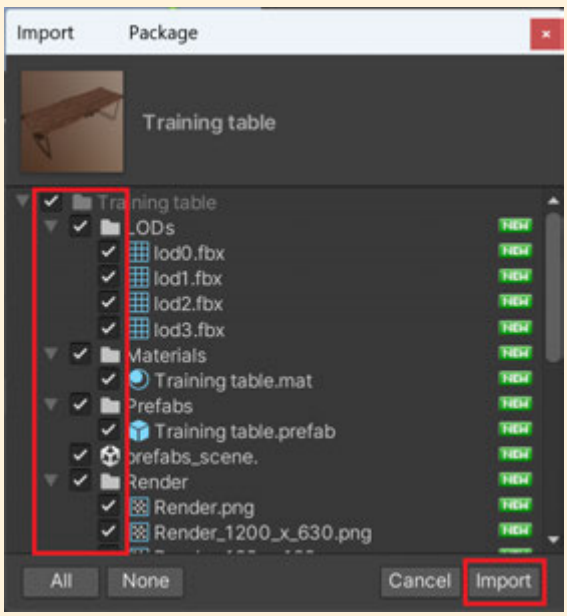
**Asset Store**'da seçtiğiniz modele göre varlık ismi farklılık gösterebilir.

Bu varlığı projenize eklemek için Import düğmesine basınız (Görsel 1.34).

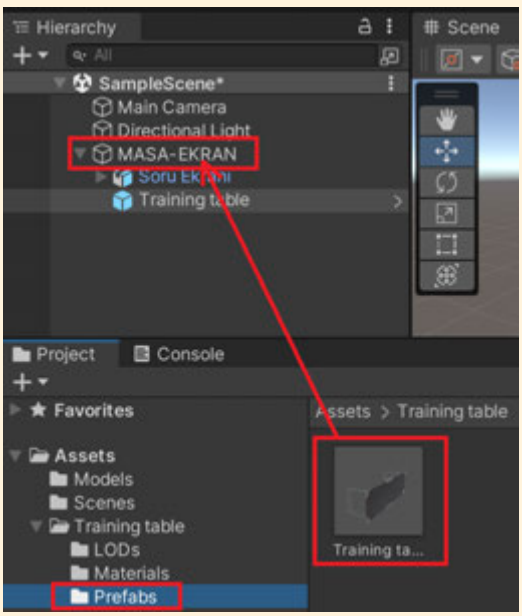



**Görsel 1.34:** Packages Manager penceresi

**7. Adım:** Açılan pencerede varlığın bütün dosyaları seçili hâldeyken Import düğmesine basınız (Görsel 1.35).
**8. Adım:** Varlık dosyaları, **Project** panelinde **Assets** klasörünün altındaki **Training table** klasörü altında görünecektir. Bu klasörü genişletip **Prefabs** klasörüne tıklayınız. Sağ tarafta oyun projenizde kullanıma hazır hâle getirilmiş ve masa modelini içeren **Training table** isimli prefabı **MASA-EKRAN** oyun nesnesinin üzerine sürükleyip bırakınız (Görsel 1.36).




**Görsel 1.35:** Varlık dosyalarının oyun motoruna eklenmesi



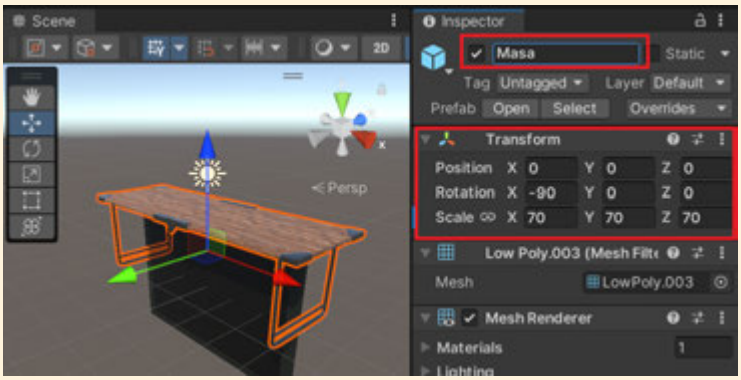

**Görsel 1.36:** Modelin sahneye eklenmesi

**9. Adım:** Hierarchy panelinde **MASA-EKRAN** oyun nesnesinin altında **Training table** isimli oyun nesneniz görünecektir. Bu nesnenin ismini **Masa** olarak değiştiriniz.

**10. Adım:** Bu nesnenin dönüşümlerinden **Position, Rotation, Scale** değerlerini güncelleyerek masanın sahnedeki görünümünü değiştiriniz (Görsel 1.37).

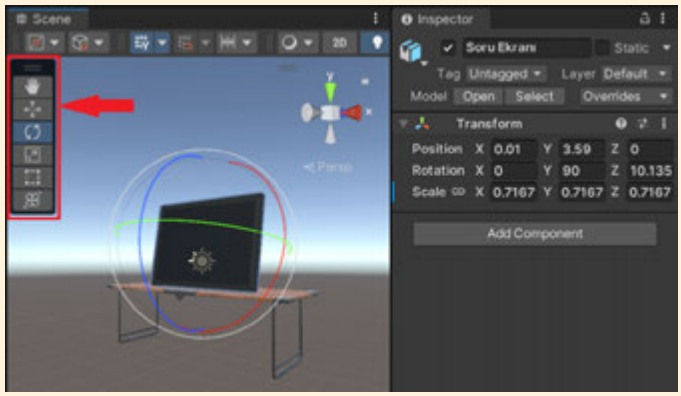
NOT

**Asset Store**'da seçtiğiniz modele göre nesnenin dönüşüm değerleri farklı olabilir.




**Görsel 1.37:** Transform değerlerinin değiştirilmesi

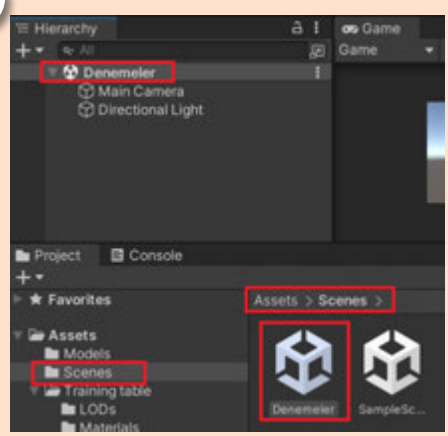
**11. Adım: Transfrom** araç kutusunu kullanarak **Soru Ekranı** nesnesinin Position, Rotation, Scale değerlerini değiştirerek Görsel 1.38'deki görünümü elde ediniz.




**Görsel 1.38:** Transform araç kutusunun kullanılması

NOT

Oyun motorunda projeye birden fazla sahne eklenebilir. Bu sahnelerde diğer sahnelerden bağımsız denemeler yapılabilir. Her uygulama için yeni bir proje oluşturmak yerine yeni sahne oluşturulup uygulama bu sahnelerde yapılabilir. Sahneler genellikle **Assets** klasörünün içinde **Scenes** isimli klasörde saklanır. Yeni bir sahne oluşturulması için **Project** panelinde belirlenen bir klasörün üstünde farenin sağ tuşu tıklanır ve gelen menüde **Create>Scene** seçeneğine tıklanır. Belirlediğiniz klasör seçilip, **Asset** menüsünden **Create>Scene** seçeneğine tıklanarak da yeni bir sahne oluşturulabilir. Yeni oluşturulan sahneye bir isim verilir ve sahneyi açmak için bu sahneye çift tıklanır (Görsel 1.39).




**Görsel 1.39:**
Yeni bir sahne oluşturulup açılması

SIRA SİZDE

1. **KelimeMacerasi** isimli projenizde yeni bir sahne oluşturunuz. Oluşturduğunuz yeni sahnenin ismini **Denemeler** olarak değiştiriniz. Denemeler sahnesini fare ile çift tıklayarak açınız.
2. **Asset Store**'dan yeni bir ücretsiz 3B model varlık indirip varlık dosyalarını projenizde içe aktarınız.
3. 3B modeli sahneye ekleyiniz ve modelin düzgün görünmesi için Transform değerlerini değiştiriniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

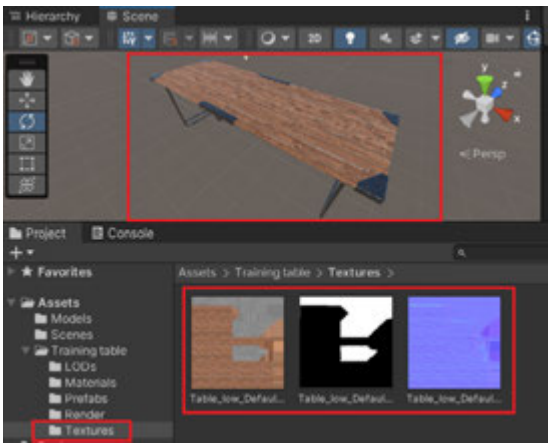
| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Yeni bir sahne oluşturup ismini Denemeler olarak değiştirdi. | | |
| 2. Asset Store'dan yeni bir ücretsiz 3B model varlık indirdi. | | |
| 3. 3B modelin varlık dosyalarını projede içe aktardı. | | |
| 4. 3B modeli sahneye ekledi. | | |
| 5. Modelin görünümünü ayarladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.1.7. Materyaller, Dokular ve Gölgelendiriciler

Oyun motorunda 3B modellere eklenebilen grafik ögeleridir. 3B modellere bu grafik ögeler eklenerek modellerin estetik ve görsel kalitesi artırabilir, modellerde daha gerçekçi görünümler elde edilebilir.

**Materyal (Material):** Bir modelin görünümünü belirleyen özelliklerin toplamıdır. Bir modelin rengi, dokusu, metalikliği, ışık yansıması, pürüzsüzlüğü gibi özelliklerini belirlemek için kullanılır. Materyaller, modellere uygulanabilen dokuları (Textures) ve gölgelendiricileri (Shaders) kapsayan bir yapıdadır. Materyallerin birçok özelliği, kendisinde seçilen gölgelendiriciye (Shader) bağlı olarak değişebilir ancak tüm shaderlar bazı ortak işlevselliğe sahiptir. Bir materyale bir veya birden fazla doku eklenebilir. Oyun motorunda materyaller genellikle **Assets** klasöründeki **Materials** klasöründe saklanır.

**Doku (Texture):** 3B modellerin yüzeylerinin üzerine uygulanan 2B resim veya desenlerdir. 3B yüzeyi kaplayan bu resim veya desenler 2B düzlemde bulunur ve doku bir 3B modele uygulandığında modelin yüzeyini kaplar (Görsel 1.40).




**Görsel 1.40:** 2B Dokular ve doku giydirilmiş 3B Model

Dokular, nesnelerin detaylarını ve gerçekçiliğini artırmak için kullanılır. Örneğin bir zeminin üzerine çim dokusu uygulandığında gerçekçi bir çim görünümü elde edilebilir. Aynı şekilde bir küpün yüzeyine taş duvar dokusu uygulandığında gerçekçi bir duvar görünümü elde edilebilir. Dokular tek başlarına (doğrudan) modellere uygulanmaz. Bir veya birden fazla doku, oluşturulan bir materyale eklenir ve bu materyal de modele uygulanır. Böylece materyal ile doku, modele uygulanmış olur. Oyun motorunda dokular genellikle **Assets** klasöründeki **Textures** klasöründe saklanır.

NOT

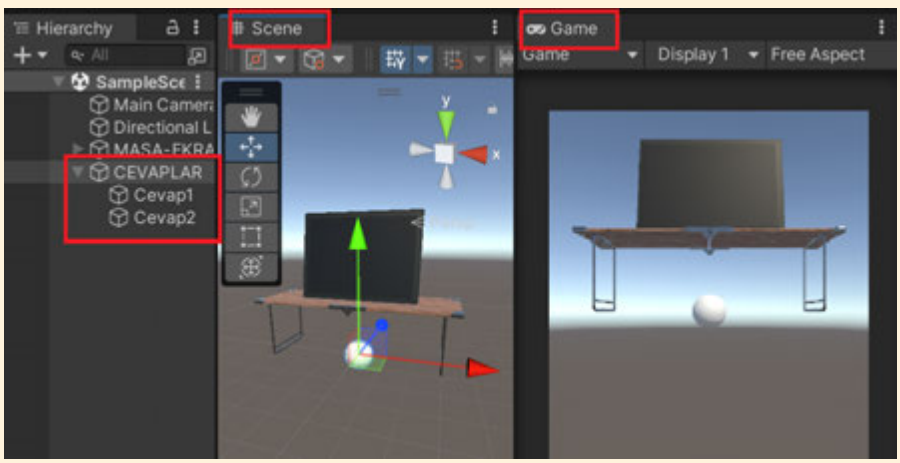
Bir 3B modele doku uygulanmayıp sadece düz renk de uygulanabilir. Renk bir materyale uygulanır, materyal de modele eklenir.

**Gölgelendirici (Shader):** 3B modellerin ışıklandırma, yansıma ve gölgeleme gibi grafik efektlerini kontrol ederek modellerin görüntülenme şeklini düzenleyen grafik ögeleridir. Gerçekçi görsel efektler elde edilmesi için kullanılır. Bir modelin dokusu, modelin yüzeyine neyin çizileceğini belirlerken bir gölgelendirici (shader) bu dokunun nasıl çizileceğini belirler. Bir gölgelendirici, bir materyal üzerinde dokular gibi uygulanır ve daha sonra bu materyal de modele eklenir. Oyun motorunun sağladığı **Standart Shader** gölgelendiricisi, materyaller üzerinde uygulanabilen ve yaygın olarak kullanılan gölgelendiricilerdendir.

7. UYGULAMA

İşlem adımlarına göre oyun projenizde sahneye iki adet küre nesnesi ve bu nesnelere materyal ekleyiniz.

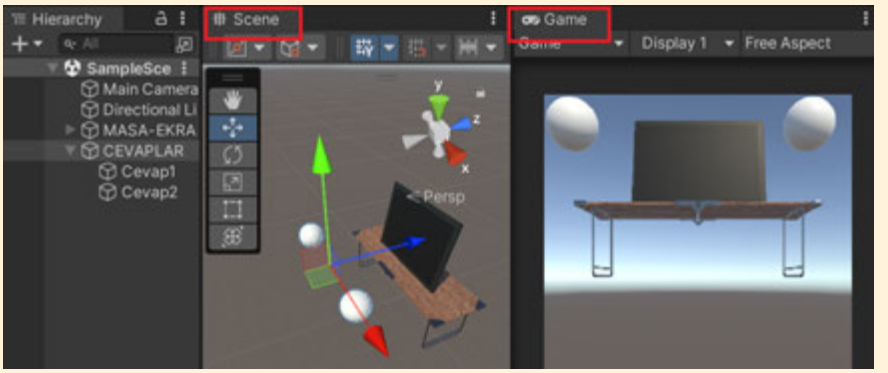
**1. Adım: KelimeMacerasi** isimli projedeki **SampleScene** isimli sahneyi açınız. **Assets** klasöründeki **Scenes** klasöründe bulunan **SampleScene** isimli sahneyi fare ile **çift tıklayarak** açınız. **CEVAPLAR** isminde boş bir oyun nesnesi oluşturunuz, nesnenin Transform değerlerini resetleyiniz. CEVAPLAR nesnesinin üzerinde farenin sağ tuşuna tıklayıp açılan menüyü kullanarak CEVAPLAR nesnesinin altında iki adet **Sphere (küre)** oluşturunuz. Kürelerin isimlerini sırası ile **Cevap1** ve **Cevap2** olarak değiştiriniz (Görsel 1.41).




**Görsel 1.41:** Oluşturulan küreler

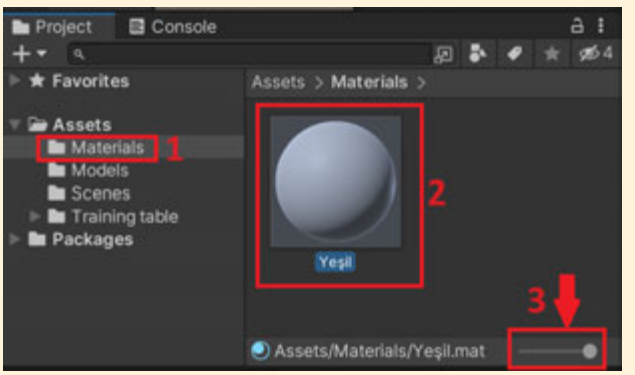
**2. Adım:** Kürelerin ikisinin birden konumunu (Position) değiştirmek için **CEVAPLAR** oyun nesnesinin konumunu değiştiriniz. Ayrı ayrı konumlarını değiştirmek için küreleri seçerek konum değişikliği yapınız.

**Scene** ve **Game** panelleri yan yana getirilerek **Game** panelinde nesnelerin oyun çalışma zamanındaki görünümleri de görülebilir (Görsel 1.42).



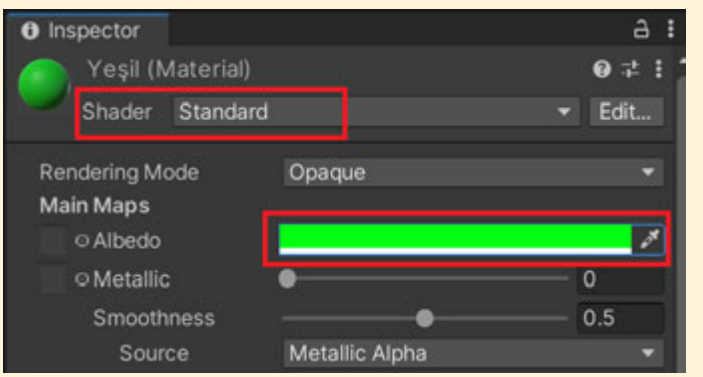

**Görsel 1.42:** Nesnelerin konumlarının değiştirilmesi

**3. Adım: Assets** klasörü altında **Materials** isminde bir klasör oluşturunuz. Bu klasörü seçip sağ taraftaki alanda farenin sağ tuşuna tıklayınız. Açılan menüde **Create>Material** seçip yeni bir materyal oluşturunuz ve ismini **Yeşil** olarak değiştiriniz. Materyalin bulunduğu panelin sağ alt tarafındaki sürgüyü sürükleyip materyallerin simgelerini büyütünüz (Görsel 1.43).




**Görsel 1.43:** Materyal oluşturulması

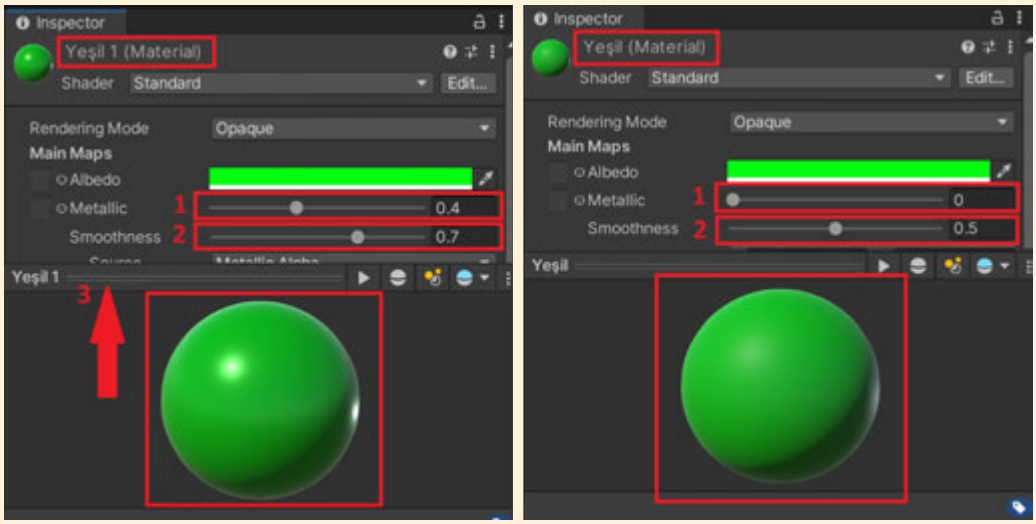
**4. Adım:** Yeşil materyalini seçip **Inspector** panelinde shader'ın (gölgelendirici) **Standard** gölgelendirici olduğuna dikkat ediniz. **Standard** gölgelendiriciye göre **Main Maps** altındaki **Albedo**'nun yanındaki renk kutusundan yeşil rengini seçiniz. Böylece materyalin renk özelliğini belirlemiş olacaksınız. Materyale şu anda doku eklemediniz ama materyalin rengini belirlediniz (Görsel 1.44).




**Görsel 1.44:** Materyalin renginin belirlenmesi

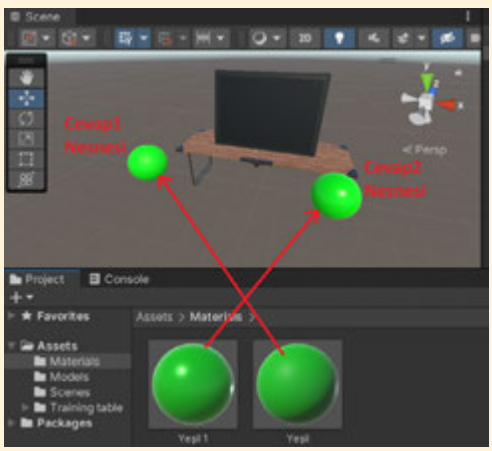
**5. Adım:** Yeşil isimli materyali seçiniz ve klavyeden **Ctrl+D** (Dublicate/Çoğaltma) tuşlarına basınız. Yeşil isimli materyal ile aynı özelliklere sahip **Yeşil 1** isimli materyal oluşacaktır.

**6. Adım: Yeşil 1** materyalini seçiniz. Inspector panelinde **Standard** gölgelendirici özelliklerinden **Metallic** değerini **0.4, Smoothness** değerini **0.7** yapınız. **Yeşil** ve **Yeşil 1** isimli materyalleri sırayla seçiniz, Inspector panelinin en altında materyalin isminin yanındaki çift çizgiye tıklayınız ve açılan bölmede iki materyalin görünümlerindeki farkı gözlemleyiniz (Görsel 1.45).




**Görsel 1.45:** Özellikleri farklı materyaller

**7. Adım: Yeşil** isimli materyali sahnedeki **Cevap1** isimli küpün üzerine sürükleyip bırakınız. **Yeşil 1** isimli materyali de sahnedeki **Cevap2** isimli küpün üzerine sürükleyip bırakınız. Böylece oluşturduğunuz materyalleri 3B nesnelere eklemiş olacaksınız. Eklediğiniz bu materyaller **Inspector** panelinde ilgili oyun nesnelerinin bileşeni olarak da görünecektir (Görsel 1.46).




**Görsel 1.46:** Materyallerin nesnelere eklenmesi

NOT

**Metallic ve Smoothness,** ışığın nesnenin yüzeyinde nasıl yansıdığı ve yüzeyin nasıl göründüğünü belirleyen gölgelendirici (shader) özellikleridir. **Metallic,** bir materyalin ne kadar metalik olduğunu belirleyen bir özelliktir. **0-1** arasında değer alır. 0 değeri metalik özelliğini kaldırır, 1 değeri tamamen metalik özellik oluşturur. Metalik yüzeyler genellikle daha yansıyan ve parlak bir görünüme sahiptir. **Smoothness,** bir materyalin ne kadar pürüzsüz olduğunu belirleyen bir özelliktir. **0-1** arasında bir değeri alır, 0 tamamen pürüzlü bir yüzeyi, 1 ise tamamen pürüzsüz bir yüzeyi temsil eder. Yüksek smoothness değerleri nesnenin yüzeyinde daha belirgin bir yansıma etkisi oluşturur.

SIRA SİZDE

1. **KelimeMacerasi** isimli projede **SampleScene** sahnesini açınız. **Kırmızı** isimli ve rengi **kırmızı** olan bir materyal oluşturunuz.
2. Materyalin **Metallic** ve **Smoothness** değerlerini değiştiriniz.
3. Bu materyali **Cevap2** nesnesine ekleyiniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

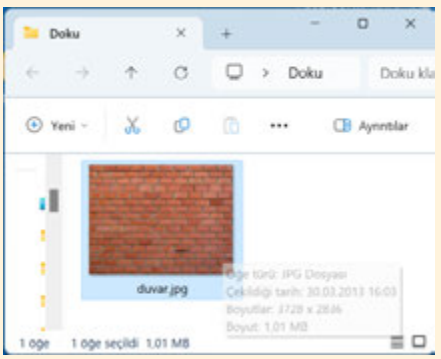
**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Kırmızı isimli ve rengi kırmızı olan bir materyal oluşturdu. | | |
| 2. Materyalin Metallic ve Smoothness değerlerini değiştirdi. | | |
| 4. Materyali Cevap2 nesnesine ekledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

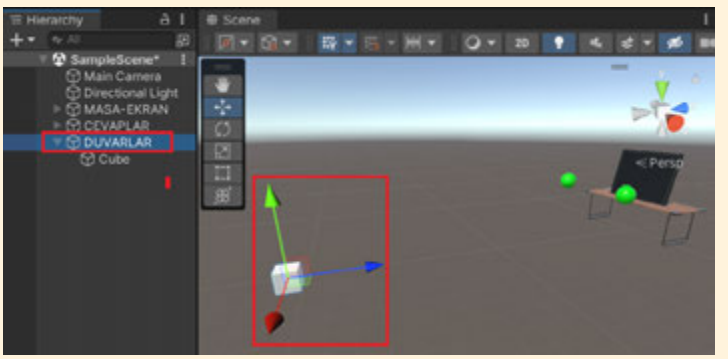
8. UYGULAMA

İşlem adımlarına göre bir dokuyu oyun nesnesine ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projede **SampleScene** sahnesini açınız. Web tarayıcınızdan arama motoruna **brick texture free** yazarak png veya jpg türünde ücretsiz bir duvar dokusu bulup indiriniz, dokunun ismini **duvar** olarak değiştiriniz (Görsel 1.47).
**2. Adım:** Oyun motorunda **Assets** klasörü altında **Textures** isimli bir klasör oluşturunuz. Bu klasörün içine **duvar.jpg** resim dosyasını sürükleyerek yerleştiriniz.
**3. Adım: DUVARLAR** isminde bir boş oyun nesnesi oluşturunuz ve bu nesnenin Transform değerlerini resetleyiniz.
**4. Adım: DUVARLAR** oyun nesnesinin içinde bir **küp** oluşturunuz. **DUVARLAR** nesnesinin konumunu Görsel 1.48'de görüldüğü gibi değiştiriniz.

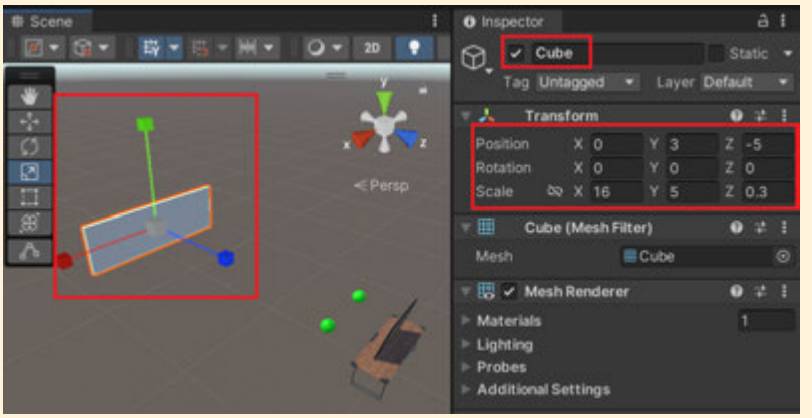



**Görsel 1.47:** Duvar dokusu



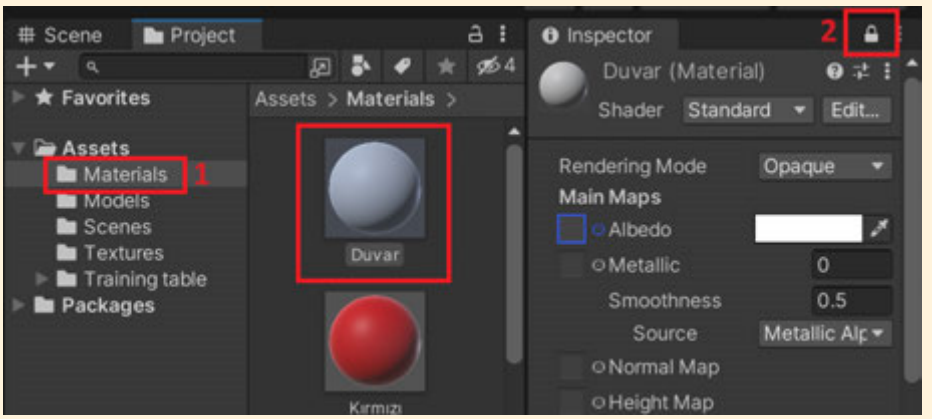

**Görsel 1.48:** DUVARLAR'ın konumunun değiştirilmesi

**5. Adım: Cube** nesnesinin Transform değerlerini değiştirerek Görsel 1.49'daki görünümü elde ediniz. Transform değerlerini Inspector panelinde Transform bileşenindeki değerleri değiştirerek veya Transform araç kutusundaki araçları kullanarak değiştirebilirsiniz.



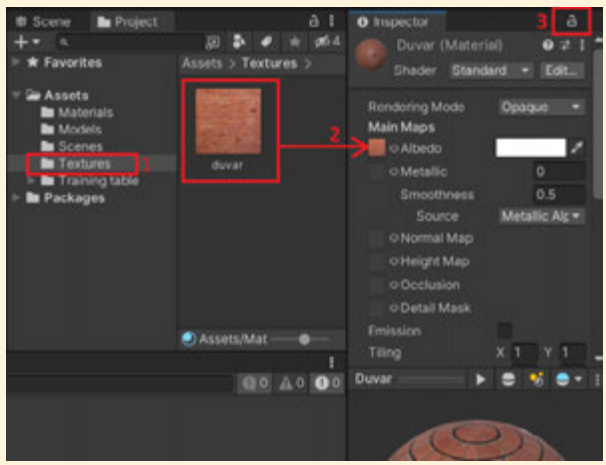

**Görsel 1.49:** Cube'un dönüşüm değerlerinin ayarlanması

**6. Adım:** Materyaller klasöründe **Duvar** isminde bir materyal oluşturunuz. **Duvar** materyalini seçip **Inspector** panelindeki kilit düğmesine tıklayınız (Görsel 1.50).



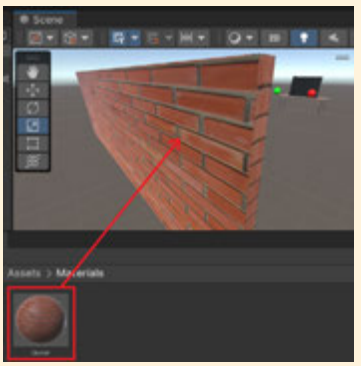

**Görsel 1.50:** Duvar isimli materyalin oluşturulması

**7. Adım: Textures** klasöründeki **duvar** isimli dokuyu materyalin **Albedo** özelliğinin sol tarafındaki küçük karenin içine sürükleyerek dokuyu **Duvar** materyaline ekleyiniz. Kilit butonuna tekrar tıklayıp Inspector panelindeki kilidi açınız (Görsel 1.51).




**Görsel 1.51:** Duvar materyaline duvar dokusunun (texture) eklenmesi

**8. Adım:** Doku eklenmiş **Duvar** isimli materyali sürükleyerek sahnedeki **Cube** oyun nesnesinin üzerine bırakınız (Görsel 1.52).



**Görsel 1.52:** Duvar materyalinin nesneye eklenmesi

SIRA SİZDE

**1. KelimeMacerasi** isimli projedeki **Denemeler** sahnesini açınız ve sahneye bir küp nesnesi ekleyiniz.
**2.** Nesnenin Scale değerlerini değiştirip nesneyi zemin hâline getiriniz.
**3.** Web tarayıcınızdan arama motoruna **grass texture free** yazarak ücretsiz bir png veya jpg türünde bir çimen dokusu bulup indiriniz.
**4.** Bu dokuyu küp nesnesine ekleyiniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Denemeler sahnesini açıp bir küp nesnesi ekledi. | | |
| 2. Küp nesnesini zemin hâline getirdi. | | |
| 3. Çimen dokusu indirdi. | | |
| 4. Çimen dokusunu küp nesnesine ekledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.1.8. Oyun Zemini

Oyun dünyasının temelini oluşturan, üzerinde hareket edilen, etkileşimde bulunulan ve oyunun oynandığı alanı tanımlayan, oyun tasarımının önemli bir unsurudur. Oyunculara daha etkileyici bir oyun deneyimi sunmak, oyun dünyasını daha gerçekçi hâle getirmek, oyunun genel kalitesini ve oynanabilirliğini artırmak için zemin tasarımına önem verilmesi gerekir.

Oyun motorunda zemin oluşturmak için genellikle **Plane (Düzlem)** ve **Terrain (Arazi)** nesneleri kullanılmaktadır. Oyunda zemin oluşturmak için hangi nesnenin kullanacağı oyunun ihtiyaçlarına ve hedeflerine bağlıdır.

**Plane (Düzlem):** Plane, temel bir 3B nesne türüdür ve düz zeminleri ve yüzeyleri oluşturmak için kullanılır. Genellikle düz zemin, duvar ve tavan gibi yapılar oluşturmak için tercih edilir. Temelde bir düzlem olduğu için hızlı bir şekilde oluşturulabilir ve düzenlenebilir. Plane nesnesine, materyal eklenerek bir doku veya düz renk uyarlanabilir. Plane nesnesi GameObject menüsünden **3D Object>Plane** seçeneği ile sahneye eklenebilir.

**Terrain (Arazi):** Oyun dünyasında detaylı, özelleştirilebilir ve gerçekçi arazi ve çevre oluşturmak için kullanılan bir oyun nesnesidir. Bu nesne, oyun geliştiricilere detaylı ve gerçekçi oyun dünyaları oluşturmak için çeşitli araçlar sunar. Çeşitli yükseklikler, düzlükler, dağlar, tepeler, vadiler, ovalar, göletler içeren araziler oluşturmayı sağlar. Ayrıca Terrain araçları ile arazi üzerine çimen, toprak, kum gibi farklı zemin türlerine uygun dokular eklenebilir. Terrain nesnesi GameObject Menüsünden **3D Object>Terrain** seçeneği ile sahneye eklenebilir.

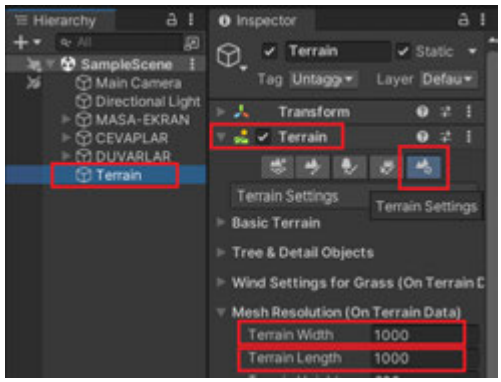
**Zemin Oluşturma ve Zemin Dokusunu Ekleme**

Oyuna zemin oluşturmak için sahneye bir Terrain eklendiğinde bu Terrain'in çok büyük olduğu görülecektir. İlk olarak oyunun yapısına göre Terrain'in hem boyutu hem de konumu ile ilgili özelliklerin değiştirilmesi gerekebilecektir. Eklenen Terrain'in boyutunu küçültmek için çözünürlük değerleri değiştirilmelidir. Bunun için Terrain seçilip, Inspector panelinden **Terrain** bileşeni altındaki **Terrain Settings** düğmesine tıklanıp açılan özelliklerden **Mesh Resolution** başlığı altındaki **Terrain Width** ve **Terrain Length** değerleri değiştirilmelidir.

**Terrain Settings Düğmesi:** Terrain'in çeşitli özelliklerini yönetmeyi sağlayan alanı açar.

**Terrain Width:** Terrain'in X eksenindeki boyutunu belirtir.

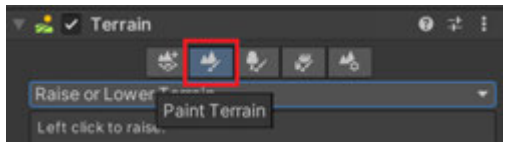
**Terrain Length:** Terrain'in Z eksenindeki boyutunu belirtir (Görsel 1.53).




**Görsel 1.53:** Terrain çözünürlük değerleri

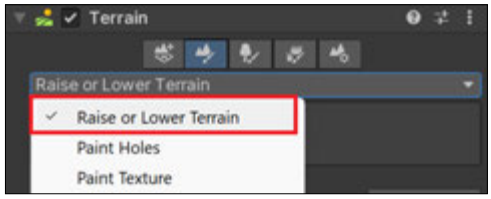
Sahnedeki Terrain üzerinde fare ile çeşitli arazi yükselti efektleri oluşturmak ve Terrain'e çimen, kum, toprak gibi dokular eklemek için yine **Terrain** bileşeni altındaki şu araçlar kullanılır:

**Paint Terrain Düğmesi:** Çeşitli dağlar, tepeler, ovalar gibi arazi görünümleri oluşturmayı ve Terrain'e çeşitli gerçekçi dokular eklemeyi sağlayan araçları barındıran alanı açan düğmedir (Görsel 1.54).



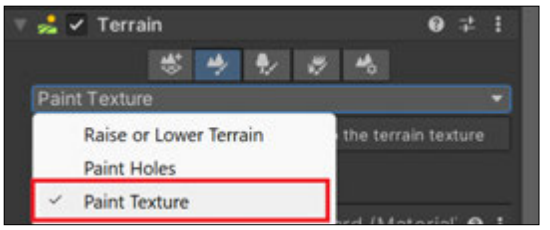

**Görsel 1.54:** Paint Terrain Düğmesi

**Raise or Lower Terrain Seçeneği:** Paint Terrain düğmesi ile açılan alandaki açılır listede bulunan bir seçenektir. Terrain üzerinde fırça ile dağlar, tepeler, ovalar gibi çeşitli **yükselti efektleri** oluşturmayı sağlar (Görsel 1.55).



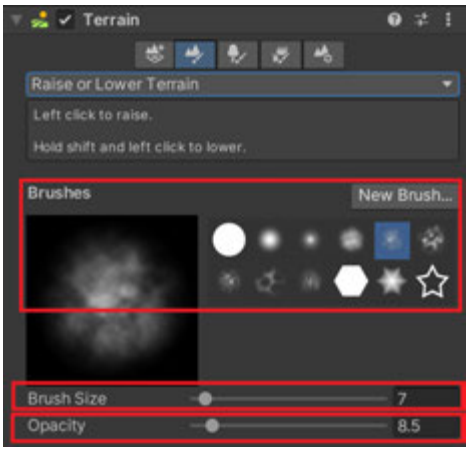

**Görsel 1.55:** Raise or Lower Terrain Seçeneği

**Paint Texture Seçeneği:** Paint Terrain Düğmesi ile açılan alandaki açılır listede bulunan bir seçenektir. Terrain üzerinde fırça ile arazi dokuları oluşturmayı sağlar (Görsel 1.56).




**Görsel 1.56:** Paint Texture Seçeneği

**Brushes:** Terrain üzerinde yükselti efektlerini veya dokuyu oluşturmak için kullanılacak fırçanın stillerini belirtir (Görsel 1.57).
**Brush Size:** Yükselti efekti veya doku fırçasının boyutunu belirtir (Görsel 1.57).
**Opacity:** Yükselti efekti veya doku fırçasının gücü belirlenir (Görsel 1.57).

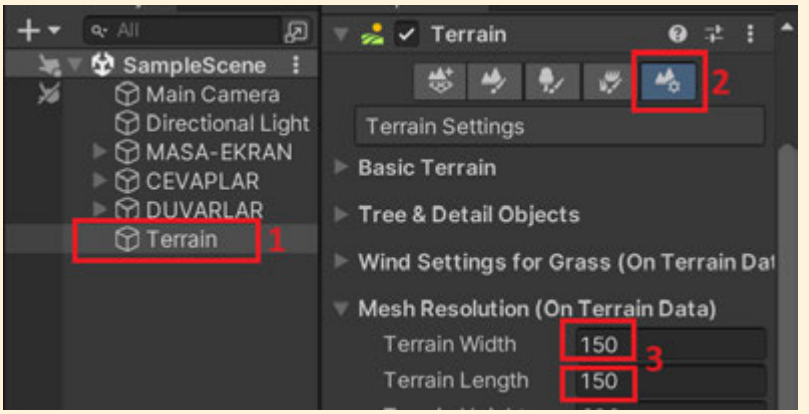



**Görsel 1.57:** Brushes, Brush Size ve Opacity

## 9. UYGULAMA

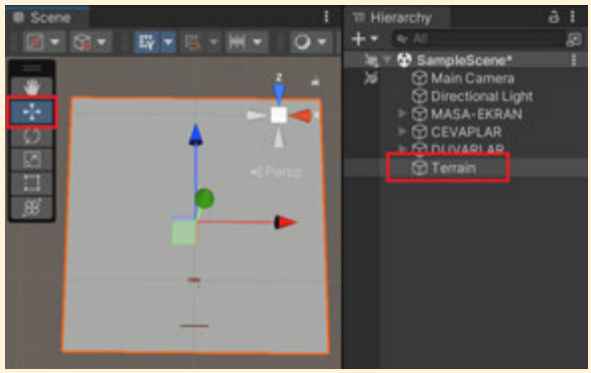
İşlem adımlarına göre sahneye **Terrain** ekleyiniz ve arazi şekilleri oluşturma işlemlerini yapınız.

**1. Adım: KelimeMacerasi** isimli projede **SampleScene** sahnesini açınız. GameObject menüsünden **3D Object>Terrain seçeneği ile** sahnenize bir **Terrain** ekleyiniz.
**2. Adım:** Terrain'i seçip Inspector panelinden **Terrain** bileşeni altındaki **Terrain Settings** düğmesine tıklayınız. Burada **Terrain Width** ve **Terrain Length** değerlerinin ikisini de **150** olarak değiştiriniz (Görsel 1.58).




**Görsel 1.58:** Terrain'in boyutlarının değiştirilmesi

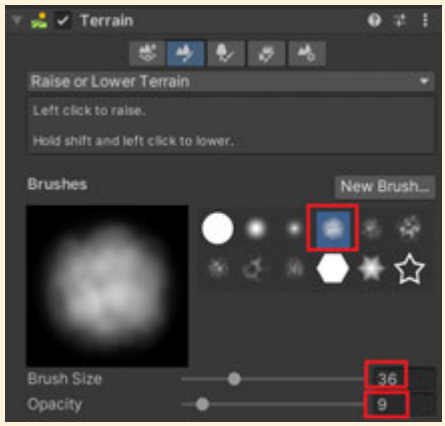
**3. Adım:** Terrain'i seçip sahne (scene) görünümde Transform araçlarından **Move Tool** aracını seçiniz ve Terrain'in konumunu sahnedeki oyun nesnelerinize göre orantılı olarak konumlandırınız. Bunu sahnenizdeki oyun nesnelerinize yukarıdan bakarak yapınız (Görsel 1.59).




**Görsel 1.59:** Terrain'in konumunun ayarlanması

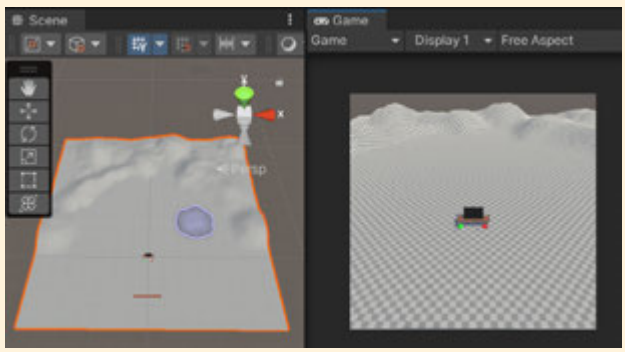
**4. Adım: Hierarchy panelinde** Terrain'i seçip Inspector panelinde **Terrain** bileşeni altındaki **Paint Terrain** düğmesine tıklayınız. Bu düğmenin hemen altındaki listede **Raise or Lower Terrain** seçeneğini seçiniz.

**5. Adım:** Görsel 1.60'ta görüldüğü gibi **Brushes**'ın altındaki fırça stilini seçiniz. **Brush Size** (yükselti efekti boyutu) değerini **36, Opacity** (efektin gücü) değerini **9** olarak değiştiriniz. Bu fırça stilini ve değerleri, oluşturmak istediğiniz arazi yükselti şekline göre değiştirebilirsiniz.




**Görsel 1.60:** Fırça özelliklerinin belirlenmesi

**6. Adım:** Fırça ayarlarını yaptıktan sonra sahnedeki Terrain üzerinde fare ile tıklamalar yaparak Görsel 1.61'deki görüntüyü elde ediniz. Hatalı efekt (yükselti) eklediğinizde bunu düzeltmek veya silmek için klavyeden **Shift** tuşunu basılı tutarak bu yükseltinin üzerine fare ile tıklayınız. Fırça stilini ve değerlerini değiştirerek Terrain'e farklı efektler ekleyebilirsiniz.




**Görsel 1.61:** Terrain üzerinde arazi şekillerinin oluşturulması

SIRA SİZDE

**KelimeMacerasi** isimli projedeki **Denemeler** sahnesini açınız ve sahneye bir terrain ekleyiniz. Bu terrain üzerinde farklı stil ve boyutlarda fırçalar ile çeşitli arazi şekilleri oluşturunuz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

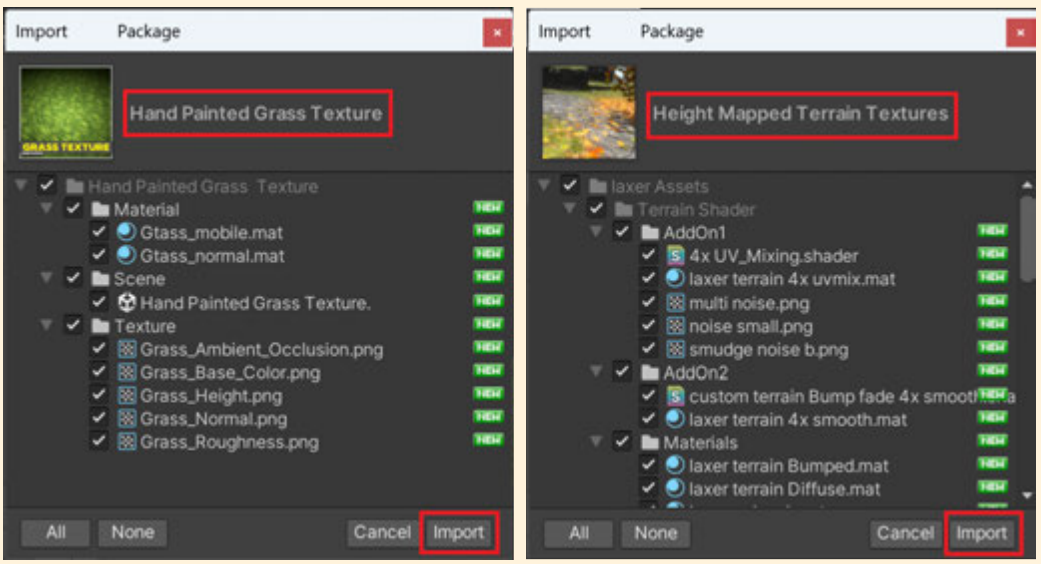
**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Denemeler sahnesine terrain nesnesi ekledi. | | |
| 2. Raise or Lower Terrain seçeneğini seçti. | | |
| 3. Fırça stilini belirledi. | | |
| 4. Fırça özelliklerini belirledi. | | |
| 5. Terrain nesnesi üzerinde çeşitli arazi şekilleri oluşturdu. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

10. UYGULAMA

İşlem adımlarına göre Terrain nesnesine doku eklemelerini yapınız.

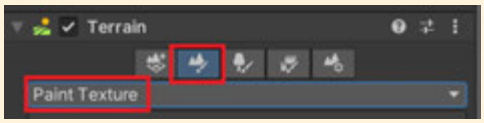
**1. Adım:** KelimeMacerasi isimli projede SampleScene sahnesini açınız. Varlık mağazasından (Asset Store) **Hand Painted Grass Texture** ve **Height Mapped Terrain Textures** isimli çeşitli dokular içeren varlıkları elde edip projenizde içe aktarınız (Görsel 1.62). Terrain üzerine farklı dokular eklemek isterseniz varlık mağazasından farklı dokular indirebilirsiniz.




**Görsel 1.62:** Asset Store'dan arazi doku varlıklarının indirilmesi

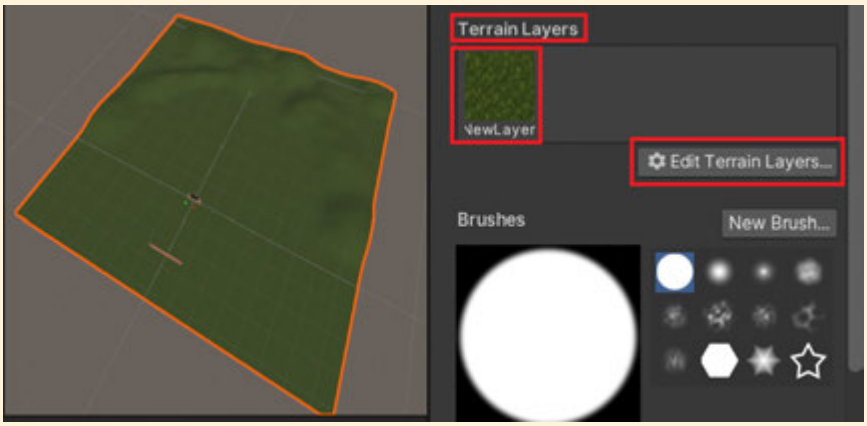
**2. Adım:** Sahneye daha önce eklediğiniz Terrain'i seçiniz. Inspector panelinde Terrain bileşeni altındaki **Paint Terrain** düğmesine tıklayınız.

Bu düğmenin hemen altındaki listede **Paint Texture** seçeneğini seçiniz (Görsel 1.63).




**Görsel 1.63:** Terrain doku ekleme seçeneği

**3. Adım: Edit Terrain Layers** düğmesine tıklayınız. Açılan listeden **Create Layer** seçeneğine tıklayınız. Açılan pencereden **Grass_Base_Color** isimli dokuyu bulup seçiniz. Seçim yaptıktan sonra bu arazi dokusu **Terrain Layers** bölümünün altında yeni bir katman (NewLayer) olarak görünecek ve Terrain'e eklenecektir (Görsel 1.64).

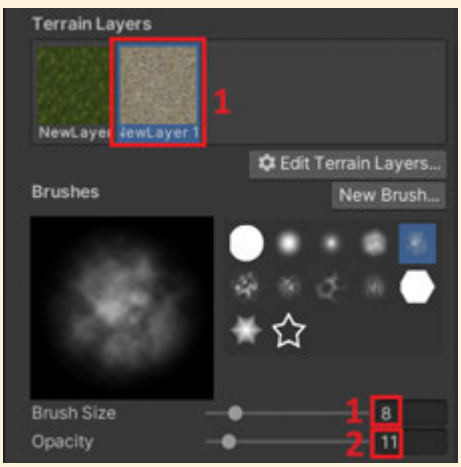



**Görsel 1.64:** Terrain'e doku eklenmesi

**4. Adım:** Tekrar Edit Terrain Layers düğmesine tıklayınız. Açılan listeden Create Layer seçeneğine tıklayınız. Açılan pencereden **sand alpha** isimli dokuyu bulup seçiniz. Seçim yaptıktan sonra bu arazi dokusu **Terrain Layers** bölümünün altında yeni bir katman (NewLayer 1) olarak görünecektir.

**5. Adım:** Terrain Layers bölümünün altındaki yeni eklediğiniz NewLayer 1'i seçip aktif yapınız. Görsel 1.65'te görüldüğü gibi dokunuzun fırça özelliklerini ayarlayınız. Fırça özellikleri isteğe göre değiştirilebilir.

**6. Adım:** Görsel 1.66'daki görünümü elde etmek için oluşturduğunuz yeni doku fırçası ile sahnedeki Terrain üzerinde boyama yapınız. Bu fırçayı kullanmak için Terrain üzerinde ihtiyaca göre fare ile tıklayınız veya fareyi sürükleyiniz. Sahne görünümünü gerektiğinde yakınlaştırıp büyütünüz. Çizim hatası yaptığınızda Terrain Layers altındaki NewLayer'ı seçip fare ile hatalı yeri boyayınız. Tekrar NewLayer 1'i seçip doku boyama işlemine devam ediniz.

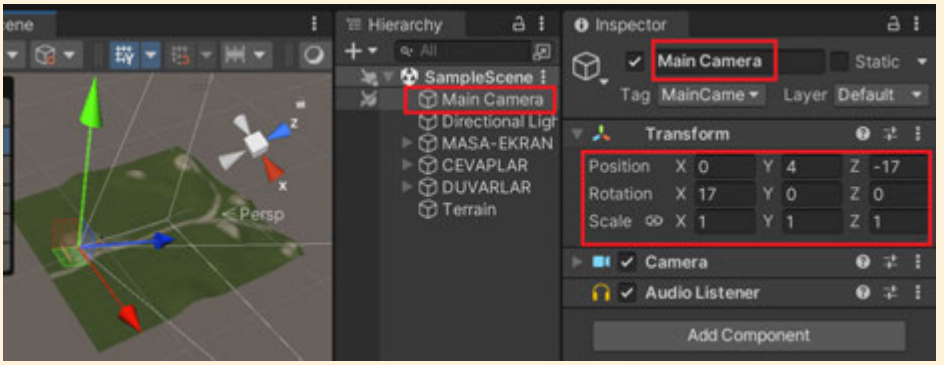



**Görsel 1.65:** Yeni dokunun fırça özellikleri



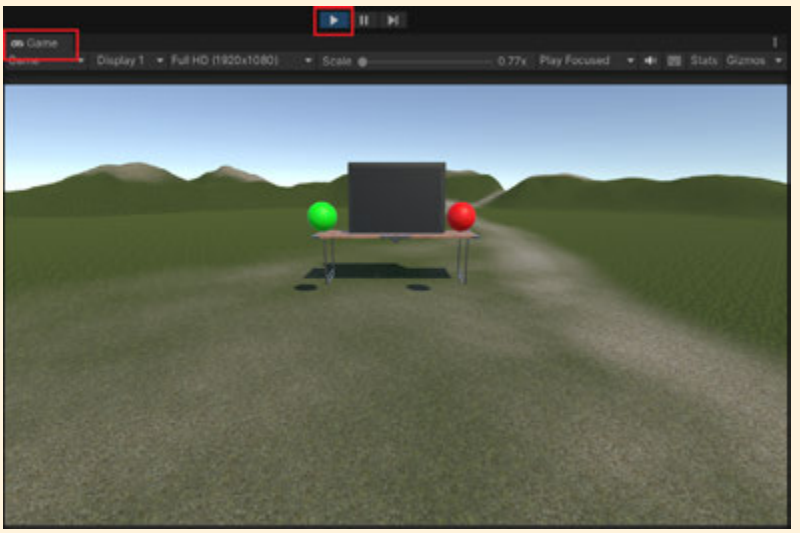

**Görsel 1.66: T**errain'in doku fırçası ile boyanması

**7. Adım: Main Camera** nesnesinin konumunu ve rotasyonunu değiştiriniz (Görsel 1.67).




**Görsel 1.67:** Main Camera'nın dönüşümlerinin değiştirilmesi

**8. Adım: Oynat (Çalıştır)** düğmesine basarak oyun dünyasını **Game** panelinde görüntüleyiniz (Görsel 1.68).



**Görsel 1.68:** Projenin çalıştırılması

SIRA SİZDE

Kar dokusu temin edip dağların yüksek yerlerine bu dokuyu uygulayınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Kar dokusu temin etti. | | |
| 2. Kar dokusunu Terrain Layers alanına ekledi. | | |
| 3. Dağların yüksek yerlerine kar dokusunu uyguladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

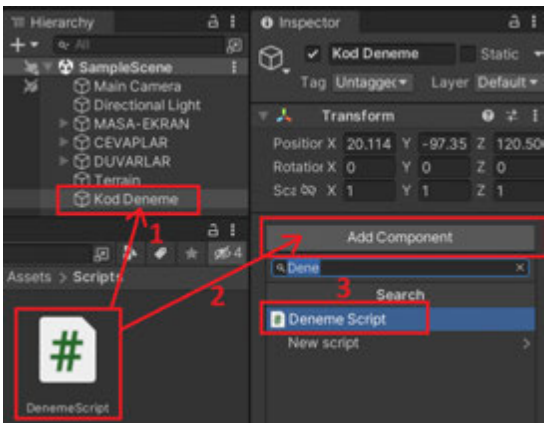
## 1.2. OYUN UYGULAMASINDA TEMEL KODLAMA

**Kodlama,** bilgisayarın belirli görevleri yerine getirmesini sağlayan programları oluşturma veya mevcut programları düzenleme sürecidir. Kodlama, oyun uygulamasında (oyun motorunda) oyun geliştirmenin temel taşlarından biridir. Oyun motorunda oyun nesnelerinin davranışlarını, etkileşimlerini, hareketlerini, oyunun kurallarını ve işleyişini oluşturmak ve kontrol etmek için kodlama kullanılır.

### 1.2.1. Script Dosyaları Oluşturma

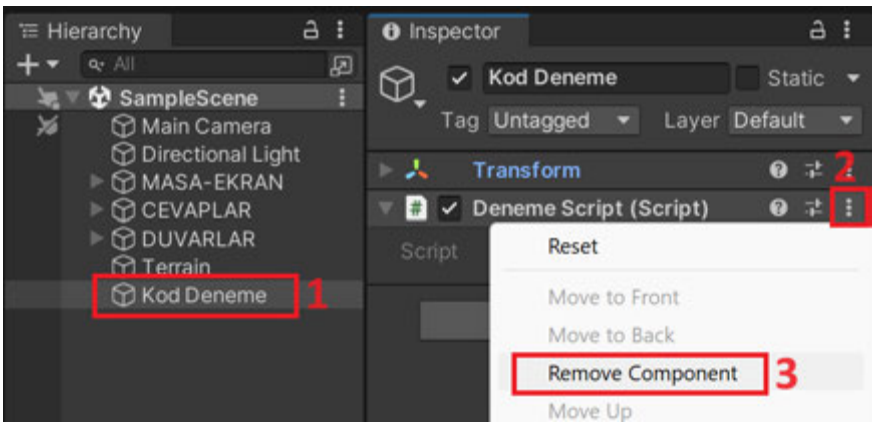
Oyun motorunda **script dosyaları,** oyun nesneleri için davranışlar ve özellikler tanımlayıp bunları kontrol etmek için kullanılan kod dosyalarıdır. Script dosyaları da birer asset'tir (varlıktır) ve oyun motorunda genellikle **Assets** klasörü altında oluşturulan **Scripts** isimli klasörün içinde oluşturulur. Bir script dosyasındaki kodların çalışması için bu dosyanın bir oyun nesnesine eklenmesi gerekmektedir ve bu işlem birden fazla yöntemle yapılabilir. Bu yöntemler şunlardır:

1. Script dosyası sürüklenerek istenilen oyun nesnesinin üzerine bırakılır.
2. Oyun nesnesi seçilir ve script dosyası sürüklenerek Inspector panelinde en alttaki **Add Component** düğmesinin üzerine bırakılır.
3. Oyun nesnesi seçilir ve Inspector panelinde en alttaki **Add Component** düğmesine tıklanır, açılan arama kutusuna script dosyasının adı yazılır ve bu dosya seçilir (Görsel 1.69).




**Görsel 1.69:** Script dosyasının nesneye eklenme yöntemleri

Bir oyun nesnesine eklenen scripti kaldırmak için ilgili oyun nesnesi seçilir. Inspector panelinde eklenmiş olan script bileşeninin sağ üstünde bulunan üst üste üç noktaya tıklanır, açılan menüde **Remove Component** seçeneği seçilir. Bu işlem Inspector panelindeki diğer bileşenler için de uygulanabilir (Görsel 1.70).




**Görsel 1.70:** Oyun nesnesinden bileşenin kaldırılması

İlk defa oluşturulan bir script dosyasının yapısı Tablo 1.1'de görüldüğü gibidir.

**Tablo 1.1:** Script Dosyasının Genel Yapısı

| | |
|---|---|
| **A** | `using System.Collections;`<br>`using System.Collections.Generic;`<br>`using UnityEngine;` |
| **B** | `public class DenemeScript : MonoBehaviour`<br>`{` |
| **C** | `// Start is called before the first frame update`<br>`void Start()`<br>`{`<br><br>`}` |
| **Ç** | `// Update is called once per frame`<br>`void Update()`<br>`{`<br><br>`}` |
| | `}` |

**A:** Bu bölüm **using bildirimleri** bölümüdür. Kodlama yapılırken belirli sınıf ve fonksiyonlara erişim sağlayan namespace'ler burada belirtilir.

**B: Sınıf bildirimi** bölümüdür. Oluşturulan her yeni script dosyasında bir sınıf vardır ve bu sınıfın ismi script dosyasının ismi ile aynıdır. Bu bölümdeki **DenemeScript : MonoBehaviour** ifadesi, **DenemeScript** sınıfının **MonoBehaviour** sınıfından türetildiğini belirtir. MonoBehaviour sınıfı, oyun motorunda oyun nesnelerine özellik ve davranış eklenmesi için kullanılan temel bir sınıftır. DenemeScript sınıfının açma ve kapama şeklinde { } iki süslü parantezi vardır ve bu parantezler içine yazılan bütün kodlar bu sınıfa, aynı zamanda DenemeScript script dosyasına ait olacaktır.

**C: Start** (başlangıç) metodudur. Start metodu, oyun nesnesi oluşturulduğunda çağrılan ve başlangıçta yapılması gereken ayarlamaları içeren metottur.

**Ç: Update** (güncelleme) metodudur. Her frame'de (karede) çalıştırılıp devamlı güncellenmesi gereken kodlar bu metot içine yazılır.

NOT

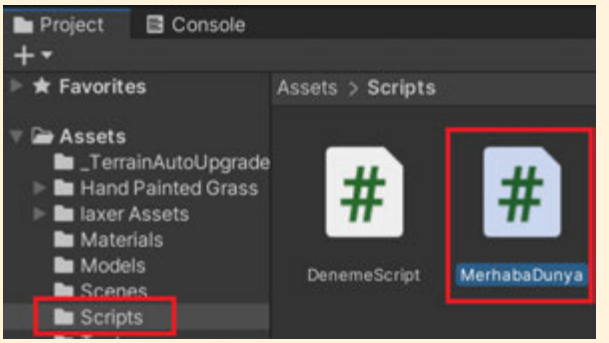
Bir oyun ekranındaki her bir görüntü veya kare, bir frame'i temsil eder. Bir oyunda saniyede belirli bir sayıda frame (kare) gösterilir. Bu sayıya **kare hızı (frame rate)** denir ve kare hızının birimi FPS (frame per second, saniyedeki kare sayısı)'dir. Kare hızı, oyunun çalıştığı cihaza göre değişebilir. Bir oyunda Update () metodu, her kare (frame)'de bir kez çağrılıp çalıştırılır. Örneğin 60 FPS kare hızına sahip bir cihazda Update() metodu saniyede 60 defa çalıştırılır.

11. UYGULAMA

İşlem adımlarına göre bir script dosyası oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **Denemeler** sahnesini açınız. Assets klasörü altında Scripts isimli bir klasör oluşturunuz. Bu klasörün üzerinde farenin sağ tuşuna tıklayıp açılan menüde **Create>C# Script** komutunu seçiniz veya Scripts klasörünü seçip Assets menüsünden **Create>C#** komutunu seçiniz.

**2. Adım:** Scripts klasörü içinde oluşturulan script dosyasının ismi **düzenlenebilir hâlde iken ilk harfi büyük olacak şekilde** script dosyasının ismini (Ör: MerhabaDunya) değiştiriniz ve klavyeden Enter'a basınız (Görsel 1.71). Script dosyasının isminin sonradan değiştirilmesi hatalara neden olabilir.



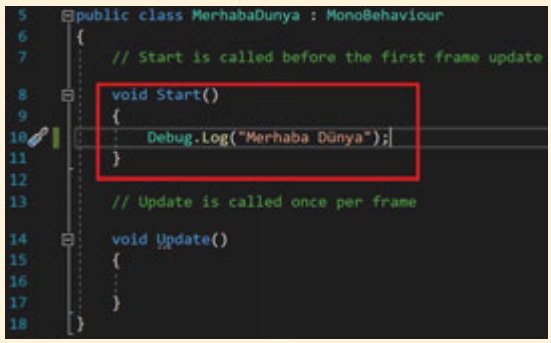

**Görsel 1.71:** Yeni oluşturulan script dosyası

**3. Adım:** MerhabaDunya script dosyasını editörde (kod düzenleyici) açmak için üzerinde fare ile çift tıklayınız. Dosya varsayılan düzenleyicide açılacaktır.

NOT

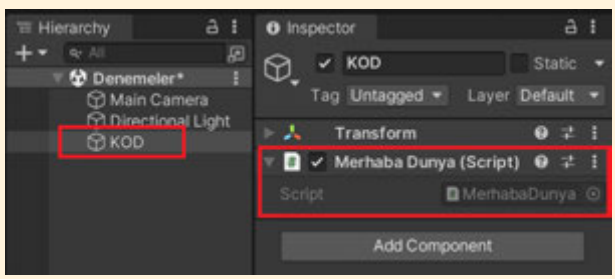
Oyun motorunda kod editörünü kullanabilmek için kod editörünün oyun motoruna tanıtılması gerekmektedir. Bunu yapmak için **Edit** menüsünden **Preferences** komutuna tıklanır. Açılan Preferences penceresinde sol taraftan **External Tools** seçilir. Sağ tarafta **External Script Editor** özelliğindeki listeden Browse düğmesine tıklanır ve ilgili kod editörünün çalışma dosyası bulunup seçilir. Artık varsayılan kod editörü seçilen kod editörü olur.

**4. Adım:** Start metodundaki { } işaretleri arasına Debug.Log("Merhaba Dünya"); kod satırını yazınız ve klavyeden Ctrl+S tuşlarına tıklayarak veya Dosya>Kaydet komutuyla değişikliği kaydediniz (Görsel 1.72).



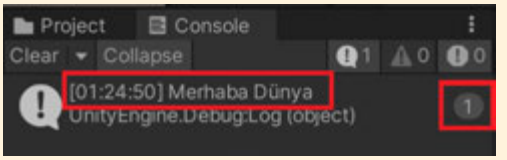

**Görsel 1.72:** Kod yazımı

**5. Adım:** Boş bir oyun nesnesi oluşturunuz ve ismini **KOD** olarak değiştiriniz. MerhabaDunya script dosyasını sürükleyerek KOD isimli oyun nesnesinin üzerine dosyayı bırakınız (Görsel 1.73).




**Görsel 1.73:** Nesneye script dosyasının eklenmesi

**6. Adım:** Oyunu çalıştırınız. **Merhaba Dünya** yazısının **Console** panelinde bir kere yazıldığını gözlemleyiniz (Görsel 1.74).




**Görsel 1.74:** Yazılan kodun çıktısı

SIRA SİZDE

**1.** MerhabaDunya2 isimli bir script dosyası oluşturunuz.

**2.** Update metodu içine, Console panelinde görünmesini istediğiniz bir mesajı yazdıran kodu yazınız.

**3.** Denemeler sahnesine bir küp nesnesi ekleyiniz ve oluşturduğunuz script dosyasını bu küp nesnesine ekleyiniz.

**Değerlendirme**

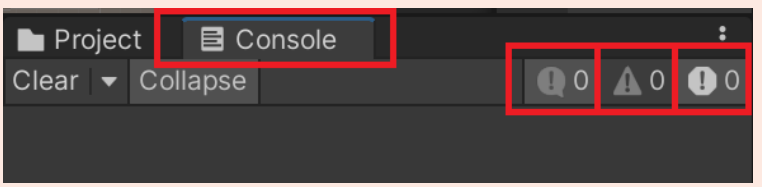
Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. MerhabaDunya2 isimli bir script dosyası oluşturdu. | | |
| 2. Bu script dosyasındaki Update metodu içine mesaj yazdıran kodu yazdı. | | |
| 3. Denemeler sahnesine bir küp nesnesi ekledi. | | |
| 4. MerhabaDunya2 script dosyasını küp nesnesine ekledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

NOT

Oyun programında **Console** paneli; geliştirici tarafından eklenen bilgi, uyarı ve hata mesajlarını gösterir. Üst menüde **Window>General>Console** ile aktif edilebilir. Console panelinde bilgi, uyarı veya hata mesajlarını görebilmek için bu panelin sağ üst tarafındaki ilgili düğmelerin aktif edilmesi gerekmektedir (Görsel 1.75).




**Görsel 1.75:** Console paneli ve mesaj düğmeleri

Debug.Log("Yazılacak Mesaj"); komutu ile oyun çalışırken belirli bilgiler veya mesajlar konsol ekranına yazdırılabilir. Debug.Log komutu kullanılarak oyun çalışma anında istenilen değişken değerleri, olaylar veya herhangi bir bilgi konsol ekranına yazdırılarak takip edilebilir. **Metin ifadeleri " " (çift tırnak) içinde yazılır.** Metin ifadesi ile bir değişken değerini beraber yazdırmak için **aralarına + işareti** yazılır.

## 1.2.2. Değişkenler

Kodlamada kullanılan, programın çalışma anında verileri saklamayı ve bu verileri tekrar kullanmayı sağlayan veri tutucularıdır. Her değişkenin veri türü ve bir ismi vardır. Değişkenlerin kodlamada kullanabilmesi için değişkenlerin tanımlanması gerekmektedir.

### Değişken Veri Türleri

Birden fazla değişken veri türü bulunmaktadır. Her bir veri türü farklı veri türlerini temsil eder ve oyun geliştirme sürecinde çeşitli görevlerde kullanılır. Oyun programlamada en çok kullanılan temel değişken veri türleri ve bu türlerin açıklamaları Tablo 1.2'de verilmiştir.

**Tablo 1.2:** Değişken Veri Türleri

| **Değişken Veri Türü** | **Açıklama** |
|---|---|
| int | Pozitif ve Negatif tam sayıları temsil eder. |
| float | Ondalık sayıları temsil eder (Örnek: 6.3). |
| string | Metin veya karakter dizilerini temsil eder. |
| bool | Mantıksal (doğru/yanlış) değerleri temsil eder. |
| char | Tek bir karakteri temsil eder. |

Oyun motorunda temel değişken veri türleri dışında kullanılan veri (nesne) türleri de bulunmaktadır. Bu veri türlerinden en çok kullanılanlar ve bunların açıklamaları Tablo 1.3'te verilmiştir.

**Tablo 1.3:** Diğer Veri Türleri

| **Veri Türü** | **Açıklama** |
|---|---|
| Vector2 | 2B vektörleri temsil eder. |
| Vector3 | 3B vektörleri temsil eder. |
| Color | Renkleri temsil eder. |
| GameObject | Oyun nesnelerini temsil eder. |
| Transform | Oyun nesnelerinin dönüşümlerinden konum, rotasyon ve ölçek değerlerini temsil eder. |
| TextMeshProUGUI | Gelişmiş metin nesnesini temsil eder. |

### Değişken Tanımlama

Bir değişken tanımlanırken değişkenin veri türü ve ismi belirtilir. Değişken tanımlamada aşağıdaki söz dizimi kullanılır:

| **Veri Türü** | **Değişken İsmi** |
|---|---|

**Tam sayı** veri türünde **sayi1** isimli bir değişken aşağıdaki gibi tanımlanır:

```
int sayi1;
```

Aynı veri türünde olan değişkenler, aynı satırda ve değişken isimlerinin aralarına virgül konularak tanımlanabilir.

**Örnek:**

```
    int oyunSuresi, skor;
    string karakterAdi, baslamaMesaji, bitisMesaji;
```

Bir script dosyasındaki sınıf ve metotlarda kullanılan { } küme parantezleri arasında bulunan alan ilgili sınıfa veya ilgili metoda aittir. Bu alan **blok** olarak isimlendirilir ve bir blok arasında tanımlanan değişkenler sadece o blok arasında kullanılabilir. Örneğin Start metodu içinde tanımlanan bir değişken, Update metodu içinde kullanılamaz ama sınıf içinde tanımlanan bir değişken o sınıf içinde bulunan Start ve Update gibi metotların hepsinde kullanılabilir çünkü Start, Update gibi metotlar sınıfın bloğu içinde bulunur.

**Örnek:**

```
public class DegiskenOrnek : MonoBehaviour
{
    private int skor = 0;
    void Start()
    {
        string mesaj = "Hoş Geldiniz";
        skor = 10;
    }
}
```

Yukarıdaki örnekte skor değişkeni, sınıfın içinde her yerde kullanılabilir ama mesaj değişkeni sadece Start metodu içinde kullanılabilir.

Oyun motorunda değişkenler tanımlanırken **private** ve **public** erişim belirleyicileri kullanılır. Bu erişim belirleyicileri sınıf düzeyinde tanımlanan değişkenlerde kullanılır. Private ifadesi ile tanımlanan değişkenler sadece tanımlandığı sınıf içinde kullanılabilir ve Inspector panelinde de görünmez. Public ifadesi ile tanımlanan değişkenler ise başka sınıflar tarafından da kullanılabilir ve Inspector panelinde de görünür.

**Örnek:**

```
public class OyunKontrolu : MonoBehaviour
{
    // Bu değişkene diğer sınıflardan da ulaşılabilir ve Inspector
    // panelinde görünür.
    public int arabaHizi;
    // Bu değişken sadece bu sınıfda kullanılabilir ve Inspector
    // panelinde görünmez.
    private int oyuncuSkoru;
    void Start()
    {
    }
}
```

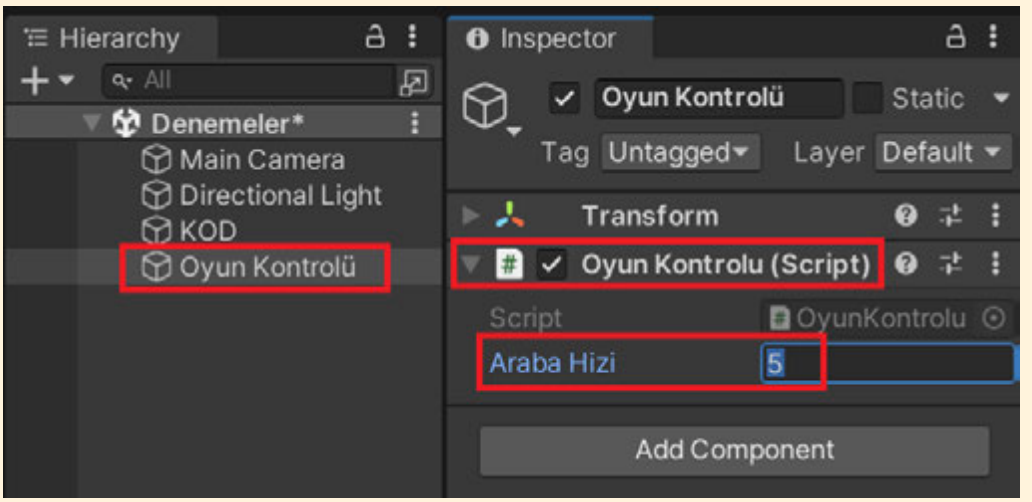
## 12. UYGULAMA

İşlem adımlarına göre erişim belirleyicileri ile ilgili script dosyasını oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **Denemeler** sahnesini açınız. Scripts klasöründe **Oyun-Kontrolu** isminde bir script dosyası oluşturunuz.
**2. Adım:** Script dosyasını kod düzenleyicide açınız ve aşağıdaki kodları yazınız.

```
public class OyunKontrolu: MonoBehaviour
{
      public int arabaHizi;
      private int oyuncuSkoru;
      void Update()
      {
          Debug.Log("Arabanın Hızı = " + arabaHizi);
      }
}
```

**3. Adım: Oyun Kontrolü** isminde boş bir oyun nesnesi oluşturunuz ve OyunKontrolu script dosyasını bu oyun nesnesine ekleyiniz.
**4. Adım:** Oyun Kontrolü oyun nesnesini seçip Inspector panelinde, OyunKontrolu script bileşeni altında arabaHizi değişkeninin görüldüğünü ve bunun düzenlenebildiğini gözlemleyiniz. arabaHizi değişkenin değerini Inspector panelinde 5 yapınız (Görsel 1.76).




**Görsel 1.76:** public değişkenin Inspector panelinde yönetilmesi

**5. Adım:** Oyunu çalıştırınız. Console panelinde **Arabanın Hızı = 5** yazısı görünecektir. Oyunun çalışma anında Inspector panelindeki arabaHizi değişkeninin değerini **7** yapınız ve Console panelinde yazılan Arabanın Hızı değerinin değiştiğini gözlemleyiniz.

SIRA SİZDE

1. OyunKontrolu script dosyası içinde **public** erişim belirleyicisi ile **string** veri tipinde **oyuncuAdi** isminde bir değişken tanımlayınız.
2. Update metodu içinde bu değişkenin değerini mesaj olarak yazdırınız.
3. Inspector panelinden bu değişkenin değerini giriniz.
4. Sahneyi çalıştırıp, oyunun çalışma anında da bu değişkenin değerini değiştirip Console panelinde değişikliği gözlemleyiniz.

**Değerlendirme**

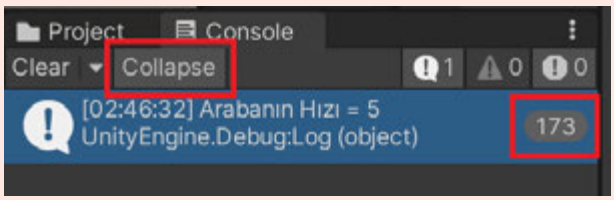
Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. public erişimli string türünde oyuncuAdi isimli bir değişken tanımladı. | | |
| 2. Update metodunda oyuncuAdi değişkeninin değerini mesaj olarak yazdırdı. | | |
| 3. Inspector panelinde oyuncuAdi değişkeninin değerini değiştirdi. | | |
| 4. Oyunun çalışma anında da oyuncuAdi değişkeninin değerini değiştirdi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

NOT

Console panelinde alt alta tekrarlanarak yazılan aynı mesajları tek bir mesaj hâlinde yazmayı sağlamak için **Collapse** seçeneği aktif edilir. Tekrarlanan mesaj sayısı bu tek mesajın yanında yazılır ve Console panelindeki mesajlar daha okunabilir hâle gelir (Görsel 1.77).




**Görsel 1.77:** Console panelinde Collapse seçeneği

Oyun motorunda, çok sayıda aynı veri türünde veriyi depolamak ve işlemek için genellikle **dizi** (array) ve **liste** (list) veri yapıları kullanılır.

**Örnek:**

```
// Integer veri tipinde 5 elemanlı bir dizi tanımlanmıştır.
public int[] sayilar = new int[5];
// Integer veri tipinde bir liste tanımlanmıştır.
public List<int> sayiListesi = new();
```

### Değişkenlere Değer Atama

Bir değişkenin değeri değişken tanımlanırken veya değişken tanımlandıktan sonra atanabilir. Oyun programlamada temel değişken veri türleri ve bu veri türlerinden tanımlanan değişkenlere değer atama örnekleri Tablo 1.4'te verilmiştir. Değer atamaları değişken tanımlama anında yapılmıştır.

**Tablo 1.4:** Değişkenlere Değer Atama

| Değişken Veri Türü | Değer Atama |
|---|---|
| int | int skor = 10; |
| float | float hiz = 5.5f; |
| string | string isim = "Ahmet"; |
| bool | bool dogruMu = true; |
| char | char harf = 'A'; |

13. UYGULAMA

İşlem adımlarına göre değişkenlerle ile ilgili bir script dosyası oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **Denemeler** sahnesini açınız. Scripts klasöründe bir script dosyası oluşturunuz ve bu script dosyasını bir oyun nesnesine ekleyiniz.
**2. Adım:** Aşağıdaki kodları Start metodu içine yazınız.

```
void Start()
{
int sayi1 = 77; // sayi1 değişkeni tanımlanmış ve değeri hemen
                // atanmıştır.
int sayi2; // sayi2 değişkeni tanımlanmıştır.
sayi2 = 60; // Değeri sonradan atanmıştır.
// iller dizisi tanımlanmış ve değerleri hemen atanmıştır.
string[] iller = new string[] {"Yalova", "Tokat", "Tekirdağ" };
// sayilar dizisi tanımlanmış ve değerleri sonradan atanmıştır.
int[] sayilar = new int[3];
sayilar[0] = 9;
sayilar[1] = 8;
sayilar[2] = 3;
// sayilar2 listesi tanımlanmış ve değerleri hemen atanmıştır.
List<int> sayilar2 = new() { 1, 2, 3 };
// iller listesi tanımlanmış ve değerleri sonradan atanmıştır.
List<string> iller2 = new();
iller2.Add("İstanbul");
iller2.Add("Bursa");
iller2.Add("Konya ");
Debug.Log("sayi2 = " + sayi2);
Debug.Log("iller dizisinin 2.Elemanı = " + iller[1]);
Debug.Log("iller2 listesinin 3.Elemanı = " + iller2[2]);
}
```

**3. Adım:** Projeyi çalıştırıp Console panelindeki mesajları kontrol ediniz.

SIRA SİZDE

**1. KelimeMacerasi** isimli projedeki **Denemeler** sahnesini açınız. Scripts klasöründe **OyunKontrolu2** isminde bir script dosyası oluşturunuz ve bu script dosyasını bir oyun nesnesine ekleyiniz.

**2.** Bu script dosyasını kod düzenleyicide açınız ve Tablo 1.5'te verilen değişkenleri sınıf seviyesinde (sınıfın içinde) oluşturunuz ve gerekli değer atamalarını yapınız.

**Tablo 1.5:** Tanımlanacak Değişkenler

| Erişim Belirleyici | Veri Türü | Değişken İsmi | Değeri | Değer Atama Zamanı |
|---|---|---|---|---|
| public | int | oyuncuSkoru; | 100 | Tanımlanırken |
| public | float | oyuncuHizi; | 10.71 | Tanımlanırken |
| private | string | mesaj; | Hoş Geldiniz | Sonradan |
| public | bool | oyunBitti; | false | Tanımlanırken |
| public | int (3 elemanlı dizi) | sayilar | 14, 53, 23 | Sonradan |

**3.** Tanımladığınız değişken değerlerini Console panelinde bir kere yazdırınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Scripts klasöründe OyunKontrolu2 isminde bir script dosyası oluşturdu. | | |
| 2. Denemeler sahnesindeki bir oyun nesnesine OyunKontrolu2 script dosyasını ekledi. | | |
| 3. OyunKontrolu2 script dosyasında değişkenleri tanımlayıp değer atamalarını yaptı. | | |
| 4. Değişken değerlerini Console panelinde bir kere yazdırdı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.2.3. Operatörler

Bir programlama dilinde değişkenler ve değerler üzerinde çeşitli işlemleri gerçekleştirmek için kullanılan sembollerdir. Operatörler; işlevlerine göre genellikle aritmetiksel operatörler, karşılaştırma operatörleri, mantıksal operatörler, atama operatörleri ve artırma operatörleri olmak üzere beş kısma ayrılır.

### Aritmetiksel Operatörler

Sayılar arasında matematiksel işlemleri gerçekleştirmek için kullanılan operatörlerdir (Tablo 1.6).

**Tablo 1.6:** Aritmetiksel Operatörler

| Sembolü | Operatör İsmi | Açıklama |
|---|---|---|
| + | Toplama | İki sayıyı toplar. |
| - | Çıkarma | İki sayıyı birbirinden çıkarır. |
| * | Çarpma | İki sayıyı çarpar. |
| / | Bölme | Bir sayıyı diğerine böler. |
| % | Mod alma | Bir sayının diğerine bölünmesinden kalanı verir. |

### Karşılaştırma Operatörleri

İki değeri karşılaştıran ve bu karşılaştırmanın sonucunda true (doğru) veya false (yanlış) mantıksal (boolean) değeri döndüren operatörlerdir (Tablo 1.7).

**Tablo 1.7:** Karşılaştırma Operatörleri

| Sembolü | Operatör İsmi | Açıklama |
|---|---|---|
| == | Eşittir | İki değer birbirine eşitse true değerini, değilse false değerini döndürür. |
| >, < | Büyüktür, Küçüktür | Bir değerin diğerinden büyük veya küçük olup olmadığını kontrol eder. Doğru ise true değerini, değilse false değerini döndürür. |
| >=, <= | Büyük Eşittir, Küçük Eşittir | Bir değerin diğerinden büyük eşit veya küçük eşit olup olmadığını kontrol eder. Doğru ise true değerini, değilse false değerini döndürür. |
| != | Eşit Değil | İki değer birbirine eşit değilse true değerini, eşitse false değerini döndürür. |

### Mantıksal Operatörler

Birden fazla mantıksal ifadeyi (boolean, true/false) karşılaştırıp değerlendirmek ve kontrol etmek için kullanılan operatörlerdir (Tablo 1.8).

**Tablo 1.8:** Mantıksal Operatörler

| Sembolü | Operatör İsmi | Açıklama |
|---|---|---|
| && | Ve (And) | İki mantıksal ifadeden her ikisi de true ise true değerini döndürür. İki ifadeden biri veya ikisi de false ise false değerini döndürür. |
| \|\| | Veya (Or) | İki mantıksal ifadeden en az biri veya ikisi de true ise true değerini döndürür. Her iki ifade de false ise false değerini döndürür. |
| ! | Değil (Not) | Bir mantıksal ifadeyi tersine çevirir. Mantıksal ifade true ise false değerini, false ise true değerini döndürür. |

### Atama Operatörleri

Bir değişkenin değerini belirlemek veya güncellemek için kullanılan operatörlerdir. Bu operatörlerle değişkene sağ taraftaki değer atanır (Tablo 1.9).

**Tablo 1.9:** Atama Operatörleri

| Sembolü | Operatör İsmi | Açıklama |
|---|---|---|
| = | Atama | Sağ tarafındaki değeri sol tarafındaki değişkene atar. |
| += | Toplama ve Atama | Sol tarafındaki değişkenin mevcut değerine sağ tarafındaki değeri ekler ve sonucu sol taraftaki değişkene atar. |
| -= | Çıkarma ve Atama | Sol tarafındaki değişkenin mevcut değerinden sağ tarafındaki değeri çıkarır ve sonucu sol taraftaki değişkene atar. |
| *= | Çarpma ve Atama | Sol tarafındaki değişkenin mevcut değerini sağ tarafındaki değerle çarpar ve sonucu sol tarafındaki değişkene atar. |
| /= | Bölme ve Atama | Sol tarafındaki değişkenin mevcut değerini sağ tarafındaki değere böler ve sonucu sol tarafındaki değişkene atar. |

### Artırma operatörleri

Bir değişkenin değerini bir birim artırmak veya azaltmak için kullanılır (Tablo 1.10).

**Tablo 1.10:** Artırma Operatörleri

| Sembolü | Operatör İsmi | Açıklama |
|---|---|---|
| ++ | Artırma | Bir değişkenin değerini bir birim artırır. |
| -- | Azaltma | Bir değişkenin değerini bir birim azaltır. |



14. UYGULAMA

İşlem adımlarına göre operatörler ile ilgili bir script dosyası oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **Denemeler** sahnesini açınız. Scripts klasöründe bir script dosyası oluşturunuz ve bu script dosyasını bir oyun nesnesine ekleyiniz.

**2. Adım:** Aşağıdaki kodları oluşturduğunuz script dosyasındaki Start metodu içine yazınız.

```
void Start()
{
  // ARİTMETİKSEL OPERATÖRLER.
  int sayi1 = 60, sayi2 = 77, sayi3 = 77;
  int toplam = 0, fark = 0, carpim=0, mod = 0;
  float bolum = 0;
  toplam = sayi1 + sayi2;
  fark = sayi1 - 10;
  carpim = sayi1 * sayi2;
  bolum = sayi2 / sayi1;
  mod = 10 % 3;
  // KARŞILAŞTIRMA OPERATÖRLERİ.
  sayi1 = 2; sayi2 = 3;
  Debug.Log(sayi1 == sayi2); // false yazdırır.
  Debug.Log(sayi1 > sayi2);  // false yazdırır.
  Debug.Log(sayi1 < sayi2);  // true yazdırır.
  Debug.Log(sayi1 >= sayi2); // false yazdırır.
  Debug.Log(sayi1 <= sayi2); // true yazdırır.
```

```
    Debug.Log(sayi1 != sayi2); // true yazdırır.
    // MANTIKSAL OPERATÖRLER.
    sayi1 = 2; sayi2 = 3;
    Debug.Log((sayi1 < sayi2) && (sayi1 > 0)); // true yazdırır.
    Debug.Log((sayi1 > sayi2) && (sayi1 > 0 )); // false yazdırır.
    Debug.Log((sayi1 > sayi2) || (sayi1 > 0)); // true yazdırır.
    Debug.Log(!(sayi1 > sayi2)); // true yazdırır.
    Debug.Log(!(sayi1 > 0)); // false yazdırır.
    // ATAMA OPERATÖRLERİ.
    sayi3 = sayi1; // sayi1 değişkeninin değerini sayi3 değişkenine
                   // atar.
    sayi1 += 1;
    Debug.Log(sayi1); // 3 yazdırır.
    sayi2 -= 1;
    Debug.Log(sayi2); // 2 yazdırır.
    sayi2 *= 2;
    Debug.Log(sayi2); // 4 yazdırır.
    sayi2 /= 2;
    Debug.Log(sayi2); // 2 yazdırır.
    // ARTIRMA OPERATÖRLERİ.
    sayi1 = 2; sayi2 = 3; sayi3 = 0;
    sayi1++;
    Debug.Log(sayi1); // 3 yazdırır.
    sayi2--;
    Debug.Log(sayi2); // 2 yazdırır.
    sayi3 = ++sayi1; // sayi1'i 1 artırır, sonra sayi1'i sayi3'e
                     // atar.
    Debug.Log(sayi3); // 4 yazdırır.
    sayi3 = sayi1++;  // sayi1'i sayi3'e atar, sonra sayi1'i 1
                      // artırır.
    Debug.Log(sayi3); // 4 yazdırır.
}
```

**3. Adım:** Projeyi çalıştırıp Console panelindeki mesajları kontrol ediniz.



SIRA SİZDE

**1.** int veri türünde üç adet değişken tanımlayınız ve bu değişkenlere ilk değerlerini atayınız.

**2.** Bu değişkenler üzerinde farklı operatörler kullanarak on adet işlem yaptırınız ve sonuçları ekrana yazdırınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. int türünde üç adet değişken tanımladı. | | |
| 2. Değişkenlere değer atadı. | | |
| 3. Değişkenler üzerinde farklı operatörler ile on adet işlem yaptıran kodları Start metodu içinde yazdı. | | |
| 4. İşlem sonuçlarını Console panelinde bir kere yazdıran kodları Start metodu içinde yazdı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.2.4. Karar Yapıları

Oyun motorunda karar yapıları; kullanıcı girişlerini, oyun nesneleri arasındaki etkileşimleri, oyun içi durumları ve birçok senaryoyu kontrol etmek ve oyun mantığını yönetmek için kullanılır. Örneğin oyun süresinin bitip bitmediği, kullanıcının bir engele değip değmediği, oyun seviyesinin bitirilip bitirilmediği, kullanıcının klavyeden yön tuşlarına basıp basmadığı gibi durumlar karar yapılarıyla tespit edilebilir. Oyun motorunda kullanılan temel karar yapıları; **if, if / else** ve **if / else if** ifadeleri ile oluşturulan koşullu ifadelerdir.

**if Yapısı**

"if"in anlamı "eğer"dir. Belirli bir şartın doğru olduğu durumda belirli bir kod bloğunu çalıştırmak için kullanılır. Bu yapıda şartın değeri true ise ilgili kod bloğundaki kodlar çalıştırılır, false ise hiçbir şey yapılmaz.

```
if (şart)
{
   // Şart doğru ise bu bloktaki kodlar çalışır.
}
```

**if / else Yapısı**

"else"in anlamı "değilse"dir. if ile belirtilen şart doğru ise if bloğundaki kodlar doğru değilse else bloğundaki kodlar çalıştırılır.

```
if (şart)
{
    // Şart doğru ise bu bloktaki kodlar çalışır.
}
else
{
    // Şart doğru değilse bu bloktaki kodlar çalışır.
}
```

**if / else if Yapısı**

Bu ifade ile birden fazla şart sırayla kontrol edilir. Sıradaki şart doğru ise sonraki şartlar kontrol edilmez, sıradaki şart doğru değilse "else"den sonraki şart kontrol edilir.

```
if (şart)
{
    // Şart doğru ise bu bloktaki kodlar çalışır.
}
else if (şart)
{
   // Önceki şart doğru değilse ve else if'teki şart doğru ise bu
   // bloktaki kodlar çalışır.
}
.
. // Varsa diğer else if ifadeleri yazılır.
.
else
{
    // Hiçbir şart doğru ise bu bloktaki kodlar çalışır.
}
```

## 15. UYGULAMA

İşlem adımlarına göre karar yapıları ile ilgili bir script dosyası oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım:** Scripts klasöründe bir script dosyası oluşturunuz ve bu script dosyasını bir oyun nesnesine ekleyiniz.
**2. Adım:** Aşağıdaki kodları Start metodu içine yazınız. Bu kodlar ile 0-100 arasında rastgele sayı üretilecektir ve bu sayının kontrolü üç farklı if yapısı ile kontrol edilecektir.

```
void Start()
 {
   // [0,100] arasında random (rastgele) bir sayı üretir.
   int sayi = Random.Range(0, 101);
   Debug.Log("Sayı: " + sayi);
   // Sadece if kullanımı.
   if (sayi > 50)
   {
       Debug.Log("Sayı 50 ile 100 arasında!");
   }
   // if / else kullanımı.
   if (sayi >= 50)
   {
       Debug.Log("Sayı 50 ile 100 arasında!");
   }
   else
   {
       Debug.Log("Sayı 0 ile 50 arasında!");
   }
   // if / else if kullanımı.
   if (sayi >= 75)
   {
       Debug.Log("Sayı 75 ile 100 arasında!");
   }
   else if (sayi >= 50)
   {
       Debug.Log("Sayı 50 ile 75 arasında!");
   }
   else
   {
       Debug.Log("Sayı 0 ile 50 arasında!");
   }
 }
```

**3. Adım:** Projeyi çalıştırıp Console panelindeki mesajları kontrol ediniz.

SIRA SİZDE

**1.** int türünde **not** isimli bir değişken tanımlayınız.

**2.** Bu değişkene 0 ile 100 arasından rastgele bir sayı ataması yapınız.

**3.** not değişkeninin değerini if / else if yapısı ile kontrol ettiriniz. Aşağıdaki aralıklara göre ilgili ifadeyi ekrana yazdırınız.

85-100 = Pekiyi
70-84 = İyi
60-69 = Orta
50-59 = Geçer
0-49 = Geçmez

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. int türünde not isimli bir değişken tanımladı. | | |
| 2. not değişkenine 0 ile 100 arasından rastgele bir sayı ataması yaptı. | | |
| 3. Değişkenin değerini kontrol ettirip aralıklara göre ilgili ifadeyi ekrana yazdırdı. | | |
| 4. Değişken değerlerini Console panelinde bir kere yazdırdı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.2.5. Döngüler

Programlamada döngüler, belirli bir kod bloğunu tekrar etmek için kullanılır. Oyun geliştirmede döngülerle oyun nesnelerini kontrol etmek, bir dizi nesneyi kontrol etmek, belirli bir işlemi belirli bir koşula kadar tekrarlamak, bir animasyonu yönetmek gibi işlemleri yapmak mümkündür. Oyun programında kullanılan temel döngüler for, while ve foreach döngüleridir.

**For Döngüsü**

Genellikle belirli bir kod bloğunu belirli bir sayıda tekrar çalıştırmak için kullanılan döngüdür. For döngüsü ifadesinde üç kısım bulunmaktadır. Bu kısımların arasında **;** işareti yazılır. Birinci kısımda döngünün çalışması için bir sayaç değişkeni tanımlanır ve bu sayacın başlangıç değeri belirtilir. İkinci kısımda döngünün sayacının kontrol edildiği şart ifadesi yazılır. Üçüncü kısımda da sayaç değişkenin artırılması veya azaltılması sağlanır.

```
for (sayaç=başlangıç değeri; şart ifadesi; sayacı artır veya azalt)
{
      // Çalıştırılacak kod veya kodlar.
}
```

Sonraki sayfada verilen for döngüsü örneğinde döngünün birinci adımında i sayacının başlangıç değeri 0'dır. i sayacının değeri 10'dan küçük olduğu sürece i sayacının değeri **i++** ifadesi ile 1 artırılıp döngü tekrar ettirilir ve döngü içindeki kodlar çalıştırılır. Buna göre **i sayacı sırasıyla 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 değerlerini alır** ve en son 1 artırılıp 10 olduğunda döngü artık durur ve çalışmaz. Böylece döngü içindeki kodlar 10 kez çalıştırılmış olur.

```
for (int i=0; i<10; i++)
{
      // Çalıştırılacak kod veya kodlar.
}
```

NOT

Döngünün i sayaç değişkeni 0'dan değil de 1'den başlatılırsa ve döngünün yine 10 kez çalışması istenilirse şart ifadesinin i<=10 veya i<11 şeklinde değiştirilmesi gerekir.

**While Döngüsü**

Belirli bir şart sağlandığı sürece belirli bir kod bloğunun tekrar çalıştırılması için kullanılan döngüdür. Genel yapısı şu şekildedir:

```
while (Şart İfadesi)
{
      // Çalıştırılacak kodlar.
}
```

While döngüsü döngü sayaç değişkeni döngü dışında tanımlanır ve bu değişkenin başlangıç değeri de döngü dışında atanır. Sayaç değişenin değeri döngünün içinde artırılır veya azaltılır.

```
int i = 0;
while (i<10)
{
      // Çalıştırılacak kodlar.
      i++;
}
```

Yukarıdaki while döngüsü örneğinde i sayaç değişkeni döngünün dışında tanımlanmıştır ve bu değişkenin başlangıç değeri 0 olarak belirlenmiştir. i<10 döngü şartına göre i sayacının değeri 10'dan küçük olduğu sürece döngü çalışmaya devam eder. Döngünün her adımında i sayacının değeri 1 artırılır ve i sayacı sırasıyla döngünün her adımında 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 değerlerini alır. i sayacı en son 1 artırılıp 10 olduğunda döngüye girilmez ve döngü çalışmaz. Böylece döngü içindeki kodlar 10 kez çalıştırılmış olur.

NOT

While döngüsü içinde sayaç değişkeni döngünün sonlanmasını sağlayacak şekilde artırılmaz veya azaltılmazsa döngünün içinden çıkılmaz. Buna dikkat edilmelidir.

**Foreach Döngüsü**

Bir dizi veya bir listenin elemanlarına tek tek erişilmesi için kullanılan bir döngüdür. Foreach döngüsü, dizi veya bir listenin boyutu ve elemanlarına erişme işlemleriyle ilgilenmek zorunda kalmadan kolay bir şekilde elemanlara erişmeyi sağlar. Bu döngü ile her döngü adımında dizi veya listedeki her bir eleman sırasıyla belirtilen bir değişkene atanır ve bu değişken sayesinde elemanlar ile ilgili istenilen işlemler yapılabilir. Genel kullanım şu şekildedir:

```
foreach (veriTürü elemanDeğişkeni in dizi/liste)
{
    // eleman üzerinde işlemler yapılır.
}
```

- **veriTürü:** Dizi veya listenin elemanlarının veri türüdür. int, string, gameObject, Transform gibi bir veri türü olabilir.
- **elemanDeğişkeni:** Her bir döngü adımında dizi veya listenin bir elemanını temsil eden değişkendir.
- **dizi/liste:** Elemanları üzerinde dolaşılacak olan dizi veya listedir.

**Örnek:**

```
void Start()
{
      string[] sehirler = { "Bursa", "Çorlu", "Tokat", "Yalova" };
      foreach (string sehir in sehirler)
      {
            Debug.Log(sehir);
      }
}
```

## 16. UYGULAMA

İşlem adımlarına göre döngüler ile ilgili bir script dosyası oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım:** Scripts klasöründe bir script dosyası oluşturunuz ve bu script dosyasını bir oyun nesnesine ekleyiniz.

**2. Adım:** Aşağıdaki kodları Start metodu içine yazınız. Bu kodlar for ve while döngüsü kullanılarak 1'den 10 kadar olan sayıların toplamını bulup ekrana yazdırır.

```
void Start()
{
  int toplam1 = 0;
  // 1'den 10'a kadar olan sayıları toplar.
  for (int i = 1; i <= 10; i++)
  {
      toplam1 += i;
  }
  Debug.Log("Toplam (for): " + toplam1);
  int toplam2 = 0;
  int sayac = 1;
  // 1'den 10'a kadar olan sayıları toplar.
  while (sayac <= 10)
  {
      toplam2 += sayac;
      sayac++;
  }
  Debug.Log("Toplam (while): " + sayac);
}
```

**3. Adım:** Projeyi çalıştırıp **Console** panelindeki mesajları kontrol ediniz.

SIRA SİZDE

For ve while döngüsü kullanıp, bu döngülerle ayrı ayrı 1'den 100'e kadar olan tek sayıları toplayıp ekrana yazdıran kodları yazınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. For döngüsü ile 1'den 100'e kadar olan tek sayıların toplamını hesaplattı. | | |
| 2. While döngüsü ile 1'den 100'e kadar olan tek sayıların toplamını hesaplattı. | | |
| 3. Toplam değerlerini ekrana yazdırdı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.2.6. Metotlar

Programlamada belirli bir görevi yerine getiren ve çağrıldığında çalışan kod bloklarıdır. Metotlar; kodları bölümlere ayırmayı, düzenlemeyi ve daha yönetilebilir, okunabilir ve sürdürülebilir hâle getirmeyi sağlar.

Metotların genel yapısı şu şekildedir:

```
<Erişim Belirleyici>  <Dönüş Türü>  MetotİsmI ( < Parametre Listesi> )
{
    // Metot Gövdesi (Kod Bloğu)
    <return ifadesi>
}
```

- **Erişim Belirleyici:** Bu belirleyici, metoda nerelerden erişebileceğini belirtir. Bir metodun erişim belirleyicisi **public** yapılırsa bu metoda herhangi bir kod bloğundan (kendi sınıfından veya başka bir sınıftan) erişilebilir. **private** yapılırsa sadece tanımlandığı sınıf içinde erişilebilir olacaktır.
- **Dönüş Türü:** Metot çağrılıp çalıştırıldığında metodun döndüreceği değerin türünü (int, string, gameObject, Transform gibi) belirtir. Eğer metot herhangi bir değer döndürmeyecekse dönüş türüne **void** ifadesi yazılır.
- **return İfadesi:** Metotta bir dönüş türü belirtilmişse ve metot bir değer döndürüyorsa return ifadesi kullanılarak dönüş değeri belirtilir. Metot bir değer döndürmüyorsa değer döndürmek için return ifadesi kullanılmaz.
- **MetotİsmI:** Metodun ismini belirtir. Metodun çağrılmasında kullanılacak isimdir. Yazım kuralı olarak metot isimlerinin ilk harfi büyük harf olmalıdır.
- **Parametre Listesi:** Metot çağrılırken kullanılacak olan giriş değerlerini tanımlayan listedir. Metotta, bu giriş değerleri belirli bir işlevi gerçekleştirmek için kullanılabilir. Bir metodun özelliğine göre metodun parametre listesi bulunabilir veya bulunamaz.
- **Metot Gövdesi:** Bu kısım, { } süslü parantezleri arasında bulunur ve metot çağrıldığında çalıştırılacak olan kodları içerir.

**Örnekler**

| Metotlar | Açıklama | Kullanımı |
|---|---|---|
| void Topla()<br>{<br>int a = 30, b = 3, sonuc = 0;<br>sonuc = a + b;<br>Debug.Log("Toplam = " + sonuc);<br>} | Parametre almayan ve geri dönüş değeri olmayan bir metottur. | void Start()<br>{<br>Topla();<br>} |
| void Fark(int a, int b)<br>{<br>int sonuc = 0;<br>sonuc = a + b;<br>Debug.Log("Fark = " + sonuc);<br>} | Parametre alan ve geri dönüş değeri olmayan bir metottur. | void Start()<br>{<br>Fark(5,4);<br>} |
| int Carp()<br>{<br>int a = 30, b = 3, sonuc = 0;<br>sonuc = a * b;<br>return sonuc;<br>} | Parametre almayan ve int türünde geri dönüş değeri olan bir metottur. | void Start()<br>{<br>int sonucCarp = Carp();<br><br>Debug.Log("Çarpım= "+ sonucCarp);<br>} |
| float Bol(int a, int b)<br>{<br>float sonuc = 0f;<br>sonuc = a / b;<br>return sonuc;<br>} | Parametre alan ve float türünde geri dönüş değeri olan bir metottur. | void Start()<br>{<br>float sonucBol = Bol(99,3);<br><br>Debug.Log("Bölüm = " + sonucBol);<br>} |

Bir metot çağrılırken ismi yazıldıktan sonra ( ) parantez açma ve kapama sembolleri kullanılır. Metot parametre alıyorsa parametreler parantez içine yazılır, almıyorsa parantez içi boş bırakılır.

### 17. UYGULAMA

İşlem adımlarına göre metotlar ile ilgili bir script dosyası oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım:** Scripts klasöründe bir script dosyası oluşturunuz ve bu script dosyasını bir oyun nesnesine ekleyiniz.
**2. Adım:** Aşağıdaki kodları script dosyası içine yazınız. Bu kodlarda üç adet aynı isimde metot tanımlanmıştır. Bunlardan biri hiç parametre almaz, biri bir adet parametre alır, biri de iki adet paremetre alır. Bu şekilde, aynı isimde fakat parametre sayıları farklı olan metot tanımlanmaya **Metot Aşırı Yüklemeleri (Method Overloads)** denmektedir.

```
public class DenemeScript : MonoBehaviour
{
  void Start()
  {
      Debug.Log("Parametresiz metot = " + Topla());
      Debug.Log("Bir parametreli Metot = " + Topla(96));
      Debug.Log("İki parametreli Metot = " + Topla(1000,1));
  }
```

```
    public int Topla()
    {
        int sayi1 = 30, sayi2 = 3;
        return (sayi1 + sayi2);
    }
    public int Topla(int sayi1)
    {
        int sayi2 = 3;
        return (sayi1 + sayi2);
    }
    public int Topla(int sayi1, int sayi2)
    {
        return (sayi1 + sayi2);
    }
}
```

**3. Adım:** Projeyi çalıştırıp Console panelindeki mesajları kontrol ediniz.

SIRA SİZDE

Yukarıdaki örneği bölme işlemi için gerçekleştiriniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Parametre almayan ve bölme işlemi yapan metodu oluşturdu. | | |
| 2. Bir parametre alan ve bölme işlemi yapan metodu oluşturdu. | | |
| 3. İki parametre alan ve bölme işlemi yapan metodu oluşturdu. | | |
| 4. Oluşturduğu metotları Start metodu içinde kullandı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.3. KULLANICI ETKİLEŞİM İŞLEMLERİ

Oyun motorunda kullanıcı etkileşimi, kullanıcıların oyun dünyasıyla etkileşimde bulunmalarını ifade eder. Oyun karakterini hareket ettirme, bir oyun nesnesi ile temasın olması, oyun arayüzünde seçim yapma gibi işlemler kullanıcı etkileşimleri ile gerçekleştirilir.

Oyun motorunda kullanıcı etkileşim işlemlerini gerçekleştirmek için kullanılan temel kavramlar ve yöntemler şunlardır:

- Kullanıcı Girişleri
- Oyun Motorunda Fizik
- Kullanıcı Arayüzü (User Interface, UI) İşlemleri
- Sahne (Scene) İşlemleri
- JSON Verileri
- Zamanlanmış İşlemler (Coroutine)

## 1.3.1. Kullanıcı Girişleri

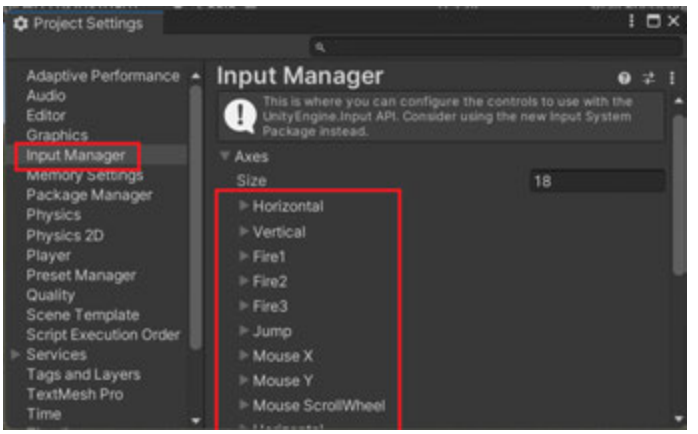
Kullanıcı girişleri, oyun motorunda kullanıcıların oyun dünyasıyla etkileşimde bulunmalarını sağlamak için kullanılır. Oyunun çalışma anında oyun karakterinin hareket ettirilmesi, oyunun görünümünün değiştirilmesi, nesnelerin seçilmesi, nesnelerin taşınması, yakınlaştırma ve uzaklaştırma yapılması, oyun arayüzünde seçim yapılması gibi işlemler kullanıcı girişleri ile gerçekleştirilebilir. Bu kullanıcı girişleri de klavye, fare, oyun kumandası gibi giriş birimleri ile elde edilir.

**Input Manager (Giriş Yöneticisi)**

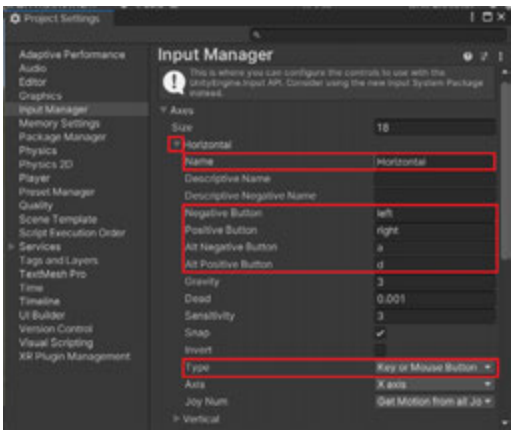
Oyun motorunda klavye, fare veya kumanda kolu (joystick) gibi giriş birimleriyle kontrol edilebilen sanal eksenler sunan ve yeni sanal eksenler tanımlamayı sağlayan bir sistemdir. Bir eksenin float (ondalıklı) veri tipinde bir değeri vardır. Bu değer, oyunun çalışma zamanında bir giriş biriminden gelen komuta göre [-1..1] arasında ondalıklı olarak değişir. Örneğin klavyeden sol yön tuşuna basıldığında değeri -1'e doğru değişen, sağ yön tuşuna basıldığında değeri 1'e doğru değişen bir eksen tanımlanabilir. Sol veya sağ yön tuşuna basılmadığında bu eksen nötr durumdadır ve bu eksenin değeri 0'dır. Klavyeden sağ yön tuşuna basılmaya başlandığında eksenin değeri 1'e doğru ondalıklı olarak artmaya başlar ve sağ yön tuşuna sürekli basılırsa eksenin değeri 1 olur. Sol tuşuna basıldığında da tam tersi gerçekleşir. Bu eksen, kod içinde kullanılarak sağ ve sol yön tuşları ile bir oyun nesnesi sağa ve sola hareket ettirilebilir. Fare ile kontrol edilebilen ve fare sola doğru hareket ettirildiğinde değeri -1'e doğru değişen, fare sağa doğru hareket ettirildiğinde değeri 1'e doğru değişen bir eksen de tanımlanabilir.

Input Manager'ı açmak için **Edit** menüsünden **Project Settings** komutuna tıklanır ve gelen Project Settings penceresinde **Input Manager** sekmesi seçilir. Input Manager'da hazır eksenler ve oyun geliştirici tarafından tanımlanan eksenler **Axes** grubunun altında bulunur (Görsel 1.78).

Bu pencerede yeni eksenler tanımlanabilir veya varsayılan eksenler üzerinde değişiklikler yapılabilir. Her bir eksenin bir ismi vardır ve kod içinde klavye veya fare ile gerçekleşen kullanıcı girişlerine erişmek için bu eksen isimleri kullanılır. Eksenler genişletilerek eksen özelliklerinin bulunduğu alan açılır. Bu alanda eksen isimleri belirlenebilir. Bir ekseni kontrol edecek giriş cihazı türü (klavye, fare, oyun kumandası gibi) ve eksenin değerinin değişmesini sağlayacak butonlar (yön tuşları gibi) da bu alanda belirlenebilir (Görsel 1.79).




**Görsel 1.78:** Input Manager ve eksenler




**Görsel 1.79:** Bir eksenin özellikleri

NOT

Kod içinde, klavye gibi bir giriş birimindeki belirli tuşlarla bir oyun nesnesini hareket ettirmek şeklinde bir eylemi gerçekleştirmek için birden fazla satır kod yazılması gerekebilir. Kod içinde eksenler kullanılarak bu ve benzeri işlemler daha az kod ile gerçekleştirilmektedir.

Eksen özellikleri ve açıklamaları Tablo 1.11'de görülmektedir.

**Tablo 1.11:** Eksen Özellikleri ve Açıklamaları

| Özellik | Açıklama |
|---|---|
| **Name** | Eksenin ismidir. Kod içinde eksene erişmek için kullanılır. |
| **Negative Button/Positive Button** | Eksenin değerini negatif ve pozitif yönde değiştirmeyi sağlayan kontrollerdir. Bunlar klavyedeki tuşlar veya joystick veya faredeki düğmeler olabilir. Ekseni, klavyeden sağ ve yön tuşları ile eşleştirmek için left/right yazılabilir. |
| **Alt Negative Button/Alt Positive Button** | Eksenin değerini negatif ve pozitif yönde değiştirmeyi sağlayan alternatif kontrollerdir. Ekseni, klavyeden A ve D tuşları ile eşleştirmek için A/D yazılabilir. |
| **Gravity** | İlgili tuşların basımı bırakıldığında eksenin nötr durumu olan 0 değerine dönme hızıdır. |
| **Dead** | Kullanıcının joystick gibi bir giriş cihazını ne kadar hareket ettirmesi gerektiğini belirler. Bu değerden küçük herhangi bir joystick hareket girişi geçersiz sayılır. |
| **Sensitivity** | Eksenin hedef değere doğru hareket edeceği saniyedeki birim hızdır. Eksenin giriş biriminden gelen komuta ne kadar hızlı yanıt vereceğini belirler. |
| **Snap** | İşaretlendiğinde, ters yöne karşılık gelen bir düğmeye basıldığında eksen değeri sıfırlanır. |
| **Invert** | İşaretlendiğinde kontrolleri tersine çevirir. |
| **Type** | Ekseni kontrol eden giriş türüdür. Aşağıdakilerden biri seçilir:<br>- Key or Mouse button (Klavye veya fare tuşu)<br>- Mouse Movement (Fare hareketi)<br>- Joystick Axis (Oyun kumanadası ekseni) |

NOT

Tek bir klavye tuşu ile veya tek bir fare tuşu ile eşleştirilen ve kontrol edilen eksenler de tanımlanabilir. Bunun için sadece Positive Button özelliğine ilgili girişin ismi yazılır. Positive Button'a sadece **space** yazılırsa bu eksen boşluk tuşuyla kontrol edilir; **mouse 0** yazılırsa farenin sol tuşuyla, **mouse 1** yazılırsa farenin sağ tuşuyla ve **mouse 2** yazılırsa farenin orta tuşuyla kontrol edilir.

**Klavye Girişleri**

Geliştirilen oyun veya uygulamada kullanıcının klavye üzerinden gerçekleştirdiği tuş basımlarını ifade eder. Oyun motorunda klavye ve fare girişleri, Input sınıfı aracılığıyla elde edilir. Bu sınıf, çeşitli giriş birimlerinden (klavye, fare, vb.) kullanıcı giriş bilgilerini elde etmek için kullanılır. Kullanıcı, klavyedeki tuşlara basarak çeşitli komutları tetikleyebilir ve bu komutlarla çeşitli etkileşimleri gerçekleştirebilir. Klavye girişlerini tespit etmek için kullanılan temel fonksiyonlar şunlardır:

- **Input.GetAxis:** Input Manager'de, klavye tuşları (genellikle yön tuşları) ile kontrol edilecek şekilde tanımlanmış bir sanal eksenin float türündeki değerini döndürür. Input Manager'de klavye girişleri ile ilgili varsayılan eksenlerden **Horizontal, Vertical** gibi eksenlerin veya sonradan tanımlanan klavye girişleri ile ilgili eksenlerin float türündeki değerlerini döndürmeyi sağlar. Döndürülen değer [-1..1] arasında ondalık bir sayıdır. Hangi eksenin değeri döndürülecekse o eksenin ismi çift tırnak içinde yazılır.

```
// Horizontal ekseninin değeri
float sagaSolaDegeri = Input.GetAxis("Horizontal");
```

```
// Vertical ekseninin değeri
float ileriGeriDegeri = Input.GetAxis("Vertical");
```

```
// SagaSola ekseninin değeri
float sagaSolaDegeri = Input.GetAxis("SagaSola");
```

- **Input.GetButton:** Input Manager'de, klavye tuşları ile kontrol edilecek şekilde tanımlanmış bir sanal ekseni kontrol eden tuşlardan birine **basıldığı sürece** true boolean değerini döndürür.

```
// Jump ekseninin tuşuna/tuşlarına basılıyor ise
if (Input.GetButton("Jump"))
      { Debug.Log("Zıplama Tuşuna Basılıyor"); }
```

- **Input.GetButtonDown:** Input Manager'de, klavye tuşları ile kontrol edilecek şekilde tanımlanmış bir sanal ekseni kontrol eden tuşlardan birine **basıldığı anda** true boolean değerini döndürür.

```
// Jump ekseninin tuşuna/tuşlarına basıldı ise
if (Input.GetButtonDown("Jump"))
      { Debug.Log("Zıplama Tuşuna Basıldı"); }
```

- **Input.GetButtonUp:** Input Manager'de, klavye tuşları ile kontrol edilecek şekilde tanımlanmış bir sanal ekseni kontrol eden tuşlardan birine **basılıp bırakıldığı anda** true boolean değerini döndürür.

```
//Jump ekseninin tuşuna/tuşlarına basılıp bırakıldı ise
if (Input.GetButtonUp("Jump"))
      { Debug.Log("Zıplama Tuşuna Basılıp Bırakıldı"); }
```

- **Input.GetKey:** Belirtilen klavye tuşuna **basıldığı sürece** true boolean değerini döndürür.

```
if (Input.GetKey(KeyCode.Space))
      { Debug.Log("Space Tuşuna Basılıyor"); }
```

- **Input.GetKeyDown:** Belirtilen klavye tuşuna **basıldığı anda** true boolean değerini döndürür.

```
if (Input.GetKeyDown(Keycode.Backspace))
      { Debug.Log("Geri Tuşuna Basıldı"); }
```

- **Input.GetKeyUp:** Belirtilen klavye tuşuna **basılıp bırakıldığı anda** true boolean değerini döndürür.

```
if (Input.GetKeyUp(KeyCode.A))
      { Debug.Log("A Tuşuna Basılıp Bırakıldı"); }
```

## 18. UYGULAMA

İşlem adımlarına göre klavye girişleri ile ilgili bir script dosyası oluşturunuz ve bu dosyayı bir oyun nesnesine ekleyiniz.

**1. Adım:** Scripts klasöründe bir script dosyası oluşturunuz ve bu script dosyasını bir oyun nesnesine ekleyiniz.
**2. Adım:** Aşağıdaki kodları Update metodu içine yazınız.

```
void Update()
{
  // Klavyeden sağ-sol yön tuşları veya A-D tuşlarına basıldığında
  // değişen Horizontal eksenin değeri, sagaSolaDegeri değişkenine
  // atanır.
  float sagaSolaDegeri = Input.GetAxis("Horizontal");
  Debug.Log("Yataydaki Sağ Sol Değeri =" + sagaSolaDegeri);
  if (Input.GetKey(KeyCode.Space))
    { Debug.Log("Space Tuşuna Basılıyor"); }
  if (Input.GetKeyDown(KeyCode.Backspace))
    { Debug.Log("Geri Tuşuna Basıldı"); }
}
```

**3. Adım:** Projeyi çalıştırınız ve ilgili tuşlara basıp Console panelindeki çıktıları kontrol ediniz.

## SIRA SİZDE

**1.** Vertical ekseninin değerini, ileriGeriDegeri isimli bir değişkene atayarak ekrana yazdıran kodları yazınız.
**2.** Geri tuşuna basılıyorken geri tuşuna basıldığını ekrana yazdıran kodları yazınız.
**3.** A tuşuna basıldığı anda A tuşuna basıldığını ekrana yazdıran kodları yazınız.
**4.** Vertical ekseninin hangi klavye tuşları ile ilişkilendirildiğini Input Manager penceresinden bulunuz.
**5.** Projeyi çalıştırınız ve ilgili tuşlara basıp Console panelindeki çıktıları kontrol ediniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

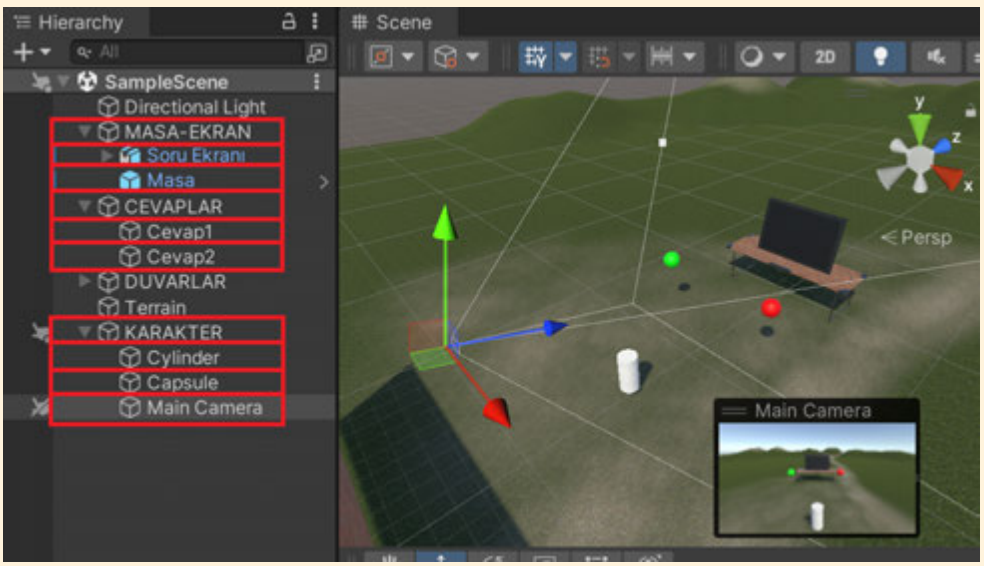
**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Vertical ekseninin değerini, ileriGeriDegeri isimli bir değişkene atayarak ekrana yazdıran kodları yazdı. | | |
| 2. Geri tuşuna basılıyorken geri tuşuna basıldığını ekrana yazdıran kodları yazdı. | | |
| 3. A tuşuna basıldığı anda A tuşuna basıldığını ekrana yazdıran kodları yazdı. | | |
| 4. Vertical ekseninin hangi klavye tuşları ile ilişkilendirildiğini Input Manager penceresinden buldu. | | |
| 5. Projeyi çalıştırıp Console panelindeki çıktıları kontrol etti. | | |
| "Hayır"olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 19. UYGULAMA

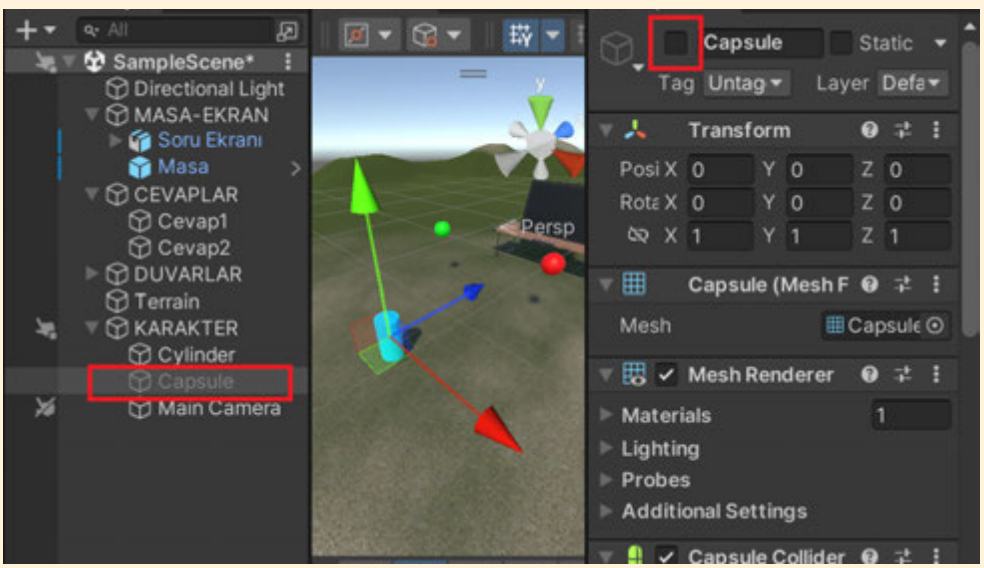
İşlem adımlarına göre KelimeMacerasi isimli projede klavyedeki yön tuşları ile oyun karakterinin sağa ve sola hareket etmesini sağlayınız.

**1. Adım: KelimeMacerasi** isimli projedeki **SampleScene** isimli sahneyi açınız. **KARAKTER** isimli boş bir oyun nesnesi oluşturunuz ve Transform değerlerini sıfırlayınız.
**2. Adım:** KARAKTER isimli oyun nesnesinin altında bir adet silindir, bir adet de kapsül nesnesi ekleyiniz.
**3. Adım:** Main Camera nesnesini sürükleyip KARAKTER oyun nesnesinin üstüne bırakınız.
**4. Adım:** KARAKTER, CEVAPLAR, MASA-EKRAN ebeveyn nesnelerinin ve bunların altındaki Cevap1, Cevap2, Cylindir, Capsule, Main Camera, Soru Ekranı, Masa isimli çocuk nesnelerinin konumlarını (Position) ve boyutlarını (Scale) düzenleyerek Görsel 1.80'deki görünümün benzerini elde ediniz.




**Görsel 1.80:** Oyun nesnelerinin konumlarının düzenlenmesi

**5. Adım:** Cylinder nesnesine mavi, Capsule nesnesine yeşil renkli materyal ekleyiniz ve Capsule nesnesinin görünürlüğünü iptal ediniz (Görsel 1.81).




**Görsel 1.81:** Nesne görünürlüğünün iptal edilmesi

**6. Adım:** Scripts klasörünün altında **KarakterHareketi** isimli bir script dosyası oluşturunuz ve bu script dosyasının içine aşağıdaki kodları yazınız.

```
public class KarakterHareketi : MonoBehaviour
{
    public float hareketHizi = 5f;
    void Update()
    {
        // Horizontal sanal ekseninde tanımlanan sağ-sol yön
        // tuşlarına veya A-D tuşlarına kullanıcı bastığında
        // değişen Horizontal eksen değeri alınır.
        float sagaSolaDegeri = Input.GetAxis("Horizontal");

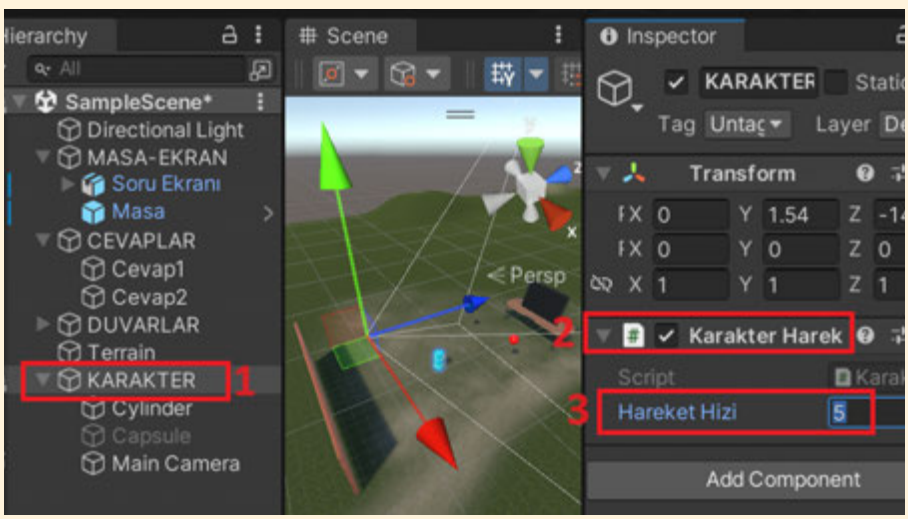
        // Kullanıcının klavye girişine göre değişen Horizontal
        // sanal eksen değeri ile hareketHizi ve Time.deltaTime
        // değeri çarpılarak X (yatay) eksenindeki hareket miktarı
        // hesaplanır.
        float xHareket = sagaSolaDegeri * hareketHizi * Time.deltaTime;

        // Bu komut KarakterHareketi script dosyasının ekleneceği
        // oyun nesnesini X (yatay) ekseninde xHareket değeri kadar
        // hareket ettirir.
        transform.Translate(new Vector3(xHareket, 0, 0));
    }
}
```

NOT

**Time.deltaTime** ifadesi son kare (frame)'den geçerli kareye kadar geçen saniye cinsinden aralık değerini ifade eder. Bu değer oyunun çalıştığı saniyedeki kare sayısı (FPS) değerine göre değişir. Bu ifadenin çarpan olarak kullanılması saniyedeki kare sayısı (FPS) değeri farklı olan cihazlarda dahi hareketin hep aynı hızda gerçekleşmesini sağlar.

**7. Adım:** KarakterHareketi isimli script dosyasını KARAKTER oyun nesnesinin üzerine ekleyiniz. Hareket hızını, proje çalışmıyorken veya çalışma zamanında hız denemesi için Inspector panelinde değiştirebilirsiniz (Görsel 1.82). Projeyi çalıştırınız ve klavyeden sağ-sol yön tuşlarına veya A-D tuşlarına basarak karakterin yatay eksendeki hareketini sağlayınız.




**Görsel 1.82:** Script dosyasının oyun nesnesine eklenmesi

SIRA SİZDE

**KarakterHareketi** isimli script dosyası içinde, klavyeden ileri-geri veya W-S tuşları ile karakteri ileri geri hareket ettirmek için gerekli olan kodları ekleyiniz. İleri geri hareketi sağlamak için Vertical isimli sanal ekseni kullanınız. Elde ettiğiniz eksen değerini **Transform.Translate(new Vector3(xHareket, 0, 0))** komutunda Z ekseni değerine (ikinci 0 değerine) yazınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Vertical isimli sanal eksenin değerini bir değişkene atadı. | | |
| 2. Z (derinlik) eksenindeki ileri geri hareket miktarını hesaplattı. | | |
| 3. Karakterin ilgili tuşlarla ileri geri hareket etmesini sağladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

**Fare Girişleri**

Geliştirilen oyun veya uygulamada kullanıcının fare ile gerçekleştirdiği çeşitli eylemleri ifade eder. Bu eylemler temel olarak farenin sol veya sağ tuşuna tıklanması, farenin tekerleğinin döndürülmesi ve farenin hareket ettirilmesi gibi işlemlerdir. Oyun motorunda fare girişleri, oyun geliştiricilerinin oyuncunun fare ile yaptığı etkileşimleri kontrol etmesini ve işlemesini sağlar. Oyun motorunda fare girişleri de klavye girişleri gibi **Input** sınıfı aracılığıyla elde edilir.

Kullanıcı, faredeki tuşlara basarak veya fareyi hareket ettirerek çeşitli komutları tetikleyebilir ve bu komutlarla çeşitli etkileşimleri gerçekleştirebilir. Fare girişlerini tespit etmek için kullanılan temel fonksiyonlar şunlardır:

- **Input.GetAxis:** Input Manager'de fare girişleri ile ilgili varsayılan eksenlerden **Mouse X, Mouse Y** ve **Mouse ScrollWheel** eksenlerinin float türündeki değerlerini döndürmeyi sağlar. Bu eksenlerin **Type** özelliği **Mouse Movement (Fare Hareketi)** olarak belirlenmiştir. Bu eksenlerin değişim değerleri **-1 ile 1 arasında** değildir, bu değerlerin dışına da çıkabilir. Mouse X ekseni, fare hareket ettirildiğinde fare imlecinin x eksenindeki değişim değerini tutar. Mouse Y ekseni, fare hareket ettirildiğinde fare imlecinin y eksenindeki değişimi değerini tutar. Mouse ScrollWheel ekseni, fare tekerleği döndürüldüğünde fare tekerleğinin değişimi değerini tutar.

```
float fareXdegisimi = Input.GetAxis("Mouse X");
float fareYdegisimi = Input.GetAxis("Mouse Y");
float fareTekerlegiDegisimi =Input.GetAxis("Mouse ScrollWheel");
```

- **Input.GetMouseButton:** Belirtilen fare tuşu **basılı tutulduğu sürece** true boolean değerini döndürür. **0** farenin sol tuşunu, **1** farenin sağ tuşunu ve **2** farenin kaydırma (scroll) tuşunu temsil eder.

```
if (Input.GetMouseButton(0))
     { Debug.Log("Farenin Sol Tuşu Basılı Tutuluyor"); }
```

- **Input.GetMouseButtonDown:** Belirtilen fare tuşuna **tıklandığı anda** true boolean değerini döndürür.

```
if (Input.GetMouseButtonDown(1))
     { Debug.Log("Farenin Sağ Tuşuna Tıklandı"); }
```

- **Input.GetMouseButtonUp:** Belirtilen fare tuşuna **tıklanıp bırakıldığı anda** true boolean değerini döndürür.

```
if (Input.GetMouseButtonUp(2))
   { Debug.Log("Farenin Orta (Scroll) Tuşuna Tıklanıp Bırakıldı"); }
```

- **Input.mousePosition:** Fare imlecinin ekran üzerindeki pozisyonununun değerini Vector3 türünden döndürür.

```
Vector3 mousePosition = Input.mousePosition;
Debug.Log("Konum =" + mousePosition);
```

## 20. UYGULAMA

İşlem adımlarına göre KelimeMacerasi isimli projede farenin sol tuşuna basılı tutulup fare sağa sola hareket ettirildiğinde oyun karakterinin Y ekseninde dönmesini sağlayınız.

**1. Adım: KelimeMacerasi** isimli projedeki **SampleScene** isimli sahneyi açınız.
**2. Adım:** KarakterHareketi isimli script dosyasını açınız ve aşağıdaki kodları bu scrip dosyasına ekleyiniz.

```
public class KarakterHareketi : MonoBehaviour
{
  ..... (19.Uygulamada oluşturulan kodlar)
  public float donmeHizi = 250f;
  void Update()
  {
    ..... (19.Uygulamada oluşturulan kodlar)
    // Farenin sol tuşuna basılı tutuluyor ise
    if (Input.GetMouseButton(0))
    {
    // Fare hareket ettirildiğine Mouse X isimli eksen ile farenin X
    // (yatay) eksendeki değişim değerini alır.
    float yataydaFareHareketi = Input.GetAxis("Mouse X");
    // Dönme miktarı hesaplanır
    float donmeMiktari = yataydaFareHareketi * donmeHizi * Time. deltaTime;
    // Script dosyasının eklendiği karakter nesnesi Y ekseninde
     // donmeMiktari kadar döndürülür.
    transform.Rotate(Vector3.up, donmeMiktari);
    }
  }
}
```

### NOT

Transform.Rotate(Vector3.up, donmeMiktari) komutu ilgili nesneyi belirtilen eksende belirtilen miktar kadar döndürür. **Vector3.up ifadesi ile Y ekseni, Vector3.right ifadesi ile X ekseni, Vector3.forward ifadesi ile Z ekseni etrafında döndürme** sağlanır.

**3. Adım:** Projeyi çalıştırıp farenin sol tuşuna basılı tutarak **KARAKTER** isimli oyun nesnesinin çocuk nesneleriyle beraber Y ekseninde dönmesini sağlayınız.

SIRA SİZDE

Farenin sağ tuşuna veya orta tuşuna basılı tutulup fare ileri geri hareket ettirildiğinde, KARAKTER nesnesinin X ekseninde dönmesini sağlayan kodları yazınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Farenin sağ tuşuna basılı tutulup tutulmadığını tespit ettirdi. | | |
| 2. Farenin orta tuşuna basılı tutulup tutulmadığını tespit ettirdi. | | |
| 3. Fare ileri geri hareket ettirildiğinde KARAKTER nesnesinin X ekseninde dönmesini sağladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.3.2. Oyun Motorunda Fizik

Oyun motoru, oyun nesnelerine gerçekçi fiziksel özellikler ve davranışlar kazandırmayı sağlar. Oyun nesneleri bu özellikler ve davranışlar ile yer çekimi, çarpışma, sürtünme, hızlanma gibi fizik kurallarına uyar; çarpışmalara, yer çekimine ve çeşitli diğer kuvvetlere doğru bir şekilde tepki verir. Bu sayede daha etkileyici ve gerçekçi oyunlar geliştirilebilir.

Oyun motorunda fizik ile ilgili temel bileşenler ve yöntemler şunlardır:

- Collider'lar (Çarpıştırıcılar)
- Rigidbody (Katı Gövde)
- Collision and Trigger (Çarpışma ve Tetikleyici)

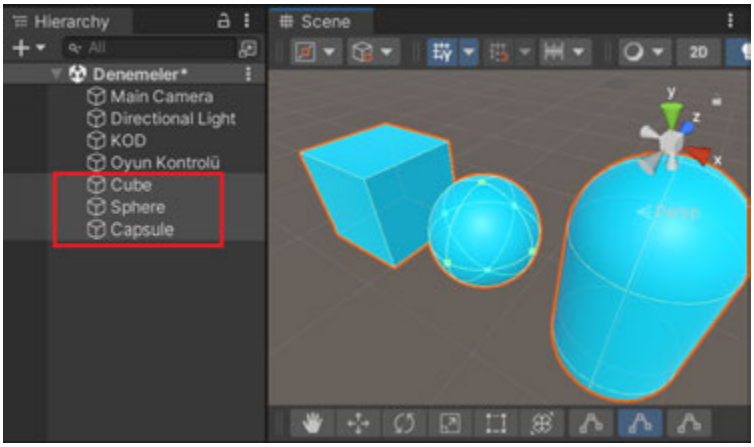
**Collider'lar (Çarpıştırıcılar)**

Bir oyun nesnesinin fiziksel (çarpışma) alanını tanımlayan ve oyun nesneleri arasındaki çarpışmaların tespit edilmesini sağlayan bileşenlerdir. Oyun programlamada yaygın olarak kullanılan Collider türleri; Box Collider, Sphere Collider, Capsule Collider ve Mesh Collider'dır.

- **Box Collider (Kutu Çarpıştırıcı):** Bir oyun nesnesini kutu şeklinde çevreleyen bir Collider'dır.
- **Sphere Collider (Küre Çarpıştırıcı):** Bir oyun nesnesini küre şeklinde çevreleyen bir Collider'dır.
- **Capsule Collider (Kapsül Çarpıştırıcı):** Bir oyun nesnesini bir kapsül (silindirin iki ucunun yuvarlatılmış hâli) şeklinde çevreleyen bir Collider'dır.
- **Mesh Collider (Ağ Çarpıştırıcı):** Bir 3B model oyun nesnesinin yapısına göre çarpışma alanını tanımlar. Mesh Collider bileşeni, modelin mesh verileri kullanılarak nesneye eklenir. Karmaşık şekillerdeki nesneler için uygundur ancak bu tür çarpıştırıcılar performans açısından daha fazla hesaplama gücü gerektirebilir.

Oyun motorunun kullanım için sunduğu Cube (Küp), Sheper (Küre), Capsule (Kapsül) gibi temel 3B nesneler veya tasarlanmış bir 3B bir model sahneye eklendiğinde bu nesnelerde varsayılan Collider'lar bulunmaktadır.

Görsel 1.83'te oyun nesneleri ve bu nesnelerin üzerlerindeki yeşil renkteki Collider'ları görülmektedir.




**Görsel 1.83:** Nesnelerin Collider'larının görünümü

Bir oyun nesnesinin üzerindeki Collider'ın (Mesh Collider hariç) büyüklüğü değiştirilebilmektedir ve bir oyun nesnesine birden fazla Collider eklenebilir. Collider'ın büyüklüğü ve sayısı oyundaki ihtiyaçlara göre belirlenmektedir. Örneğin bir oyun karakterinin üzerine eklenen Capsul Collider'ın boyutları karakter nesnesinin boyutlarından bir miktar büyük yapılarak, duvar gibi nesnelerle çarpışma olduğunda karakterin duvarın içine girmesi engellenebilir.

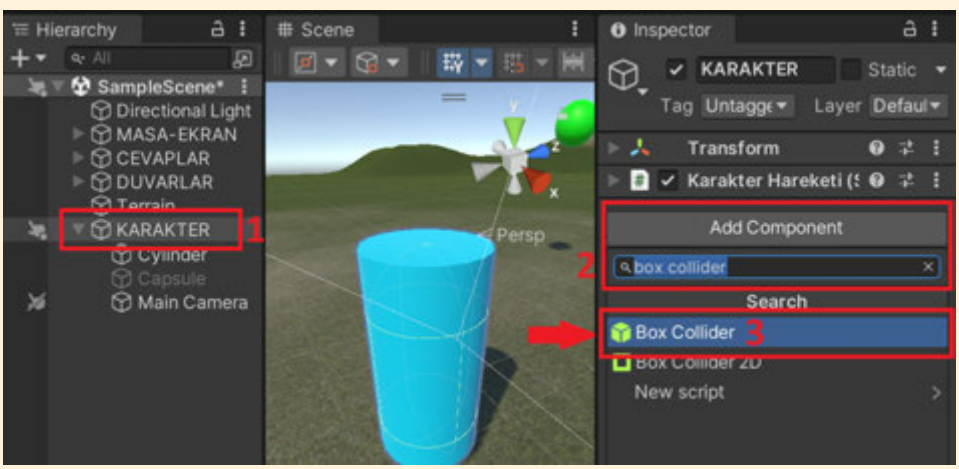
NOT

Boş bir oyun nesnesine de Collider eklenebilir. Görünmeyen bir oyun sınırı oluşturulacaksa bu kullanışlı olacaktır.

21. UYGULAMA

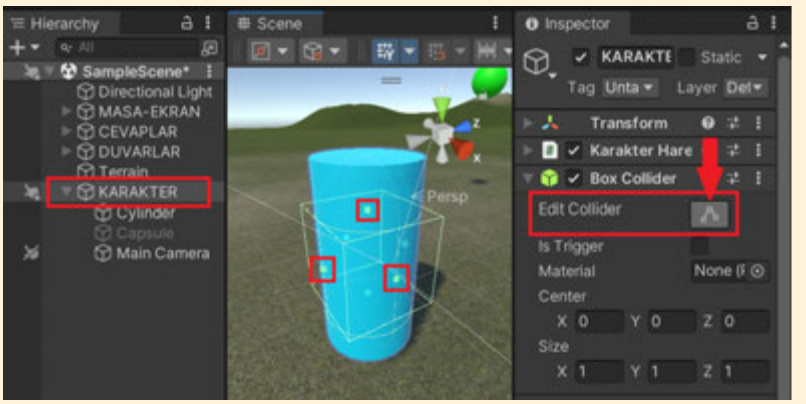
İşlem adımlarına göre **KelimeMacerasi** isimli projedeki KARAKTER nesnesine Collider ekleyiniz ve bu collider ile ilgili işlemleri yapınız.

**1. Adım: KelimeMacerasi** isimli projedeki **SampleScene** isimli sahneyi açınız.
**2. Adım:** KARAKTER nesnesini seçiniz ve Inspector panelinde **Add Component** düğmesine tıklayınız. Açılan alanda arama kutusuna **Box Collider** yazınız ve alt tarafta görünen Box Collider bileşenini seçiniz. Box Collider bileşeni KARAKTER isimli oyun nesnesine eklenmiş olacaktır (Görsel 1.84).



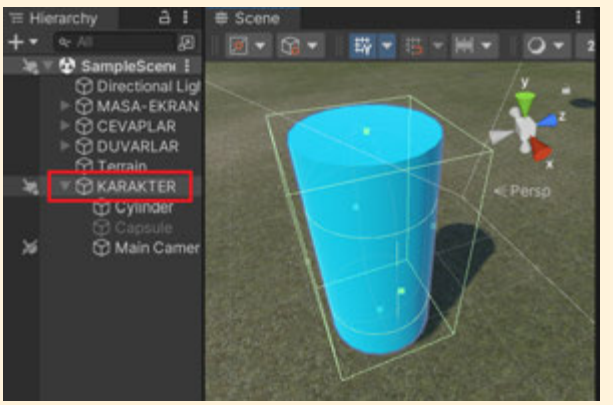

**Görsel 1.84:** Oyun nesnesine Collider eklenmesi

**3. Adım:** Collider KARAKTER nesnesine eklendikten sonra nesneyi seçiniz ve Box Collider bileşeni altındaki Edit Collider düğmesine tıklayınız. Bu, eklenen Collider'ın büyüklüğünü collider üzerindeki tutamaçlar ile ayarlamanızı sağlayacaktır (Görsel 1.85).




**Görsel 1.85:** Edit Collider düğmesi ve tutamaçlar

**4. Adım:** Collider'ın ilgili tutamaçlarını fare ile tutup sürükleyerek Box Collider'ın büyüklüğünü Görsel 1.86'daki gibi ayarlayınız. Collider'ın boyutunu simetrik olarak büyütmek için tutamaca tıkladıktan sonra klavyeden Alt tuşuna basarak tutamacı sürükleyiniz.




**Görsel 1.86:** Collider'ın boyutlarının ayarlanması

SIRA SİZDE

Farklı bir sahneye boş bir nesne ekleyip bu nesneye Capsule Collider ekleyiniz ve Collider'ın boyutlarını ayarlayınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Boş bir nesne ekledi. | | |
| 2. Boş nesneye Capsule Collider ekledi. | | |
| 3. Capsule Collider'ın boyutlarını ayarladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

**Rigidbody (Katı Gövde)**

Nesnelere gerçek dünyadaki gibi fiziksel özellikler ve davranışlar kazandırmayı ve bunları kontrol etmeyi sağlayan bir bileşendir. Bu bileşen, nesnelerin fiziksel etkileşimlere tabi olarak hareket etmelerini sağlar. Nesnelerin fizik kurallarına tabi davranışlarının, hareketlerinin ve çarpışmalarının yönetilmesi için kullanılır.

Oyun motorunda 3B bir nesneye Rigidbody eklemek için nesne seçilir ve Inspector panelinden Add Component düğmesine tıklanır. Açılan alanda arama kutusuna **Rigidbody** yazılır ve bu alandan Rigidbody seçilir.

Rigidbody bileşeninin özellikleri Tablo 1.12'de görülmektedir.

**Tablo 1.12:** Rigidbody Özellikleri

| Özellik | Açıklama |
|---|---|
| **Mass (Kütle)** | Oyun nesnesinin kütle miktarını kilogram cinsinden tanımlar. Kütle varsayılan olarak 1 olarak ayarlanmıştır. Kütle, bir nesnenin yer çekimine karşı düşüş hızını etkilemez. |
| **Drag (Sürükleme)** | Bir oyun nesnesine bir kuvvet (yer çekimi, itme gibi) uygulandığında nesneye uygulanacak hava direncini veya sürtünme değerini tanımlar. Nesnenin hareket ederken hızının nasıl azalacağını belirler. 0 ile infinity (∞) arasında değer alır. Varsayılan değeri 0'dır ve bu durumda hareket eden nesneye hava direnci hiç uygulanmaz, nesnenin hızı hiç azalmaz ve nesne bir engelle karşılana kadar hareket etmeye devam eder. Değeri yükseldikçe uygulanan hava direnci artar ve hareket eden nesne bir engelle karşılaşmasa bile bir süre sonra durur. Gerçek dünyadaki hafif nesneleri simüle etmek için değeri yükseltilir. **infinity** değeri verilirse nesne hareket etmeden duracaktır. |
| **Angular Drag (Açısal Sürükleme)** | Oyun nesnesi dönerken nesneye uygulanacak hava direncini belirler. Nesnenin dönme hızının nasıl azalacağını belirler. Varsayılan değeri 0.05'tir. 0 değeri verildiğinde nesneye hiçbir direnç uygulanmaz ve nesneye dönme kuvveti uygulandığında nesne devamlı döner. Değer yükseldikçe dönen nesneye uygulanan hava direnci artar ve nesnenin durması belirli bir sürede sağlanır. |
| **Use Gravity (Yer Çekimini Kullan)** | Oyun nesnesi üzerinde yer çekimi etkilerini açmayı ve kapatmayı sağlar. True veya false değeri alır ve varsayılan değeri true'dur. Etkinleştirildiğinde oyun nesnesine yer çekimi yönünde (Y eksenin aşağı olarak) hareket ettirecek yer çekimi kuvveti uygulanır. Yer çekimi kuvvetinin nesneye olan etkisi, nesnenin Drag (Sürükleme) değerine bağlı olarak fazla veya az olabilir. |
| **Is Kinematic (Kinematik mi)** | Bu özellik etkinleştirildiğinde oyun nesnesi fizik kurallarından etkilenmez ve nesneyi hareket ettirmek veya döndürmek için nesneye fizik kuvvetleri uygulanamaz. Varsayılan değeri false'tur. |
| **Interpolate (Düzeltme/ İnterpolasyon)** | Bu özellik oyunun çalışma zamanında oyun nesnesi, Rigidbody ile hareketi ettirilirken titriyormuş gibi görünüyorsa, bu görünümü düzelterek, yumuşatarak nesnenin pürüzsüz bir şekilde hareket etmesini sağlar. Üç tür değer alır. Bunlar:<br>**None:** Düzeltme işlemi yapılmaz.<br>**Interpolate:** Rigidbody'nin bir önceki karedeki poz (görünüm) ve hız gibi fizik özelliklerine göre düzeltme işlemi yapılır.<br>**Extrapolate:** Rigidbody'nin sonraki karedeki fizik güncellemesini tahmin etmek için mevcut poz kullanılarak düzeltme işlemi yapılır. |

| | |
|---|---|
| **Collision Detection (Çarpışma Algılama)** | Oyun nesneleri arasındaki çarpışmaların nasıl algılanacağını belirleyen özelliktir. Varsayılan değeri Discrete'dir ve 4 farklı seçeneği şunlardır:<br>**Discrete (Sürekli Olmayan):** Discrete seçeneği hızlı hareket eden nesnelerle çarpışma olmadığı durumlarda seçilir. Hızlı hareket eden nesnelerdeki çarpışmaları kaçırabilir.<br>**Continuous (Sürekli):** Nesneler arasındaki çok hızlı çarpışmaların daha hassas tespiti için kullanılır. Yüksek hızlarda dahi daha güvenilir çarpışma algılama sağlar.<br>**Continuous Dynamic (Sürekli Dinamik):** Çarpışma algılaması discrete olan nesnelerle gerçekleşen çarpışmalarda discrete algılama kullanılır. Diğer continuous nesnelerle gerçekleşen çarpışmalarda continuous algılama kullanılır.<br>**Continuous Speculative (Sürekli Spekülatif):** Yüksek hızlı çarpışmalarda nesneler arasındaki oluşabilecek potansiyel çarpışma önceden tahmin edilmeye çalışılır. Bazı durumlarda tam doğruluk sağlamayabilir. Çarpışma hassasiyeti önemli değilse Continuous Speculative seçilebilir. |
| **Constraints (Kısıtlamalar)** | Oyun nesnesinin hareketi ve rotasyonu (dönmesi) üzerinde kısıtlamalar uygular.<br>**Freeze Position:** Nesnenin seçili olan X, Y ve Z eksenlerinde hareket etmesini engeller.<br>**Freeze Rotation:** Nesnenin seçili olan X, Y ve Z eksenlerinde dönmesini engeller. |

NOT

Bir oyun nesnesinin Collider'ı (çarpıştırıcısı) olup da Rigidbody'si yok ise bu nesnedeki Collider **Statik Collider** olarak isimlendirilir. Nesnenin Collider'ı ile beraber Rigidbody'si de varsa bu Collider **Dinamik Collider** olarak isimlendirilir. Statik çarpıştırıcılar, dinamik çarpıştırıcılarla etkileşime girebilir ancak çarpışmalardan sonra hareket etmez, devrilmez çünkü bir RigidBody'e sahip değildir.

Kod içinde, Rigidbody eklenen bir oyun nesnesinin Rigidbody'sini yönetmek için bu nesnenin Rigidbody bileşenini elde etmek gerekir. Bunun için **GetComponent** fonksiyonu kullanılır. Bu fonksiyon ile bir oyun nesnesinin diğer bileşenleri de elde edilebilmektedir.

```
Rigidbody rb
rb = gameObject.GetComponent<Rigidbody>()
```

Yukarıdaki kodlarda Rigidbody sınıfı türünden tanımlanana rb değişkenine ilgili oyun nesnesinin Rigidbody bileşeni atanmıştır. Bu rb değişkeni ile nesnenin Rigidbody'si yönetilebilecektir. Örneğin rb değişkeni kullanılarak bu Rigidbody'ye bir kuvvet uygulanabilir veya Use Gravity, Is Kinematic gibi bazı özellikleri değiştirilebilir. Rb gibi bir değişken tanımlamadan kod içinde doğrudan **gameObject.GetComponent<Rigidbody>()** komutu ile de Rigidbody bileşeni kontrol edilebilir.

22. UYGULAMA

İşlem adımlarına göre KelimeMacerasi isimli projedeki KARAKTER oyun nesnesine Rigidbody ve zıplama davranışı ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **SampleScene** isimli sahneyi açınız.
**2. Adım: KARAKTER** isimli oyun nesnesini seçiniz. Inspector panelinden **Add Component** düğmesine tıklayınız. Açılan alanda arama kutusuna **Rigidbody** yazınız ve bu alandan Rigidbody'yi seçiniz.
**3. Adım: KarakterHareketi** isimli script dosyasını açınız ve aşağıdaki kodları bu script dosyasına ekleyiniz.

```
public class KarakterHareketi : MonoBehaviour
{
  ..... (20.Uygulamada oluşturulan kodlar)
```

```
  public float ziplamaGucu = 7f;
  Rigidbody rb;
  void Start()
  {
     rb = gameObject.GetComponent<Rigidbody>();
  }
  void Update()
  {
    ..................  (20.Uygulamada oluşturulan kodlar)
    // Jump sanal ekseninde tanımlanan klavyedeki space (boşluk)
    // tuşuna kullanıcı bastığı anda Ziplama metodu çağrılır.
    if (Input.GetButtonDown("Jump"))
    {
        Ziplama(ziplamaGucu);
    }
  }
  void Ziplama(float ziplamaGucu)
  {
    // Nesnenin Rigidbody'sine AddForce metodu ile Up (Y) ekseni
    // yönünde bir kuvvet uygulanır. Bu nesnenin yukarı doğru
    // zıplamasını sağlar.
      rb.AddForce(Vector3.up * ziplamaGucu, ForceMode.Impulse);
  }
}
```

NOT

**ForceMode.Impulse parametresi,** AddForce metodu ile uygulanacak kuvvetin türünü belirten bir parametredir. Impulse ifadesi ile nesneye bir anlık kuvvet uygulanması sağlanır.

**4. Adım:** Projeyi çalıştırınız ve ilgili oyun nesnesinin zıplama davranışını kontrol ediniz.

SIRA SİZDE

1. KARAKTER nesnesini seçiniz, Inspector panelinde KarakterHareketi script bileşenindeki Ziplama Gücü değerini artırınız ve projeyi çalıştırıp zıplama davranışındaki değişikliği kontrol ediniz.
2. KARAKTER nesnesinin Inspector panelinde Rigidbody bileşenin sırasıyla Mass, Drag, Use Gravity, Is Kinematic özelliklerini değiştiriniz ve nesnenin zıplamasıyla ile ilgili oluşan farklılıkları kontrol ediniz.

NOT

Inspector panelindeki bu değişiklikleri projeyi çalıştırdıktan sonra da yapabilirsiniz, projenin çalışmasını sonlandırdığınızda değişiklikler iptal olacaktır.

**Değerlendirme**

Çalışmalarınız diğer sayfada yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Ziplama Gücü değerini artırıp zıplama davranışındaki değişikliği kontrol etti. | | |
| 2. Rigidbody'nin Mass, Drag, Use Gravity, Is Kinematic özelliklerini değiştirdi. | | |
| 3. Nesnenin zıplamasıyla ile ilgili oluşan farklılıkları kontrol etti. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

### Collision (Çarpışma)

İki nesnenin birbirine temas ettiği durum olan çarpışmayı ifade eder. İki nesne arasında bir çarpışmanın gerçekleşmesi ve tespit edilmesi için iki nesneden en az birinin kinematik olmayan (Is Kinematic özelliği seçili olmayan) bir Rigidbody'e sahip olması gerekmektedir. Oyun nesneleri arasında çarpışma gerçekleştiğinde, ilgili nesnelere eklenmiş olan scriptlerde (betiklerde) belirli isimlere sahip metotlar çağırılır. Çarpışma gerçekleştiği anda istenilen kodlar bu metotların içine yazılabilir. Örneğin oyun karakteri bir engele çarptığında bir ses çalınabilir veya oyuncunun puanı azaltılabilir. Bu metotlar şunlardır:

- **OnCollisionEnter:** Bu metot, iki nesne arasında bir çarpışma (temas) başladığı anda çağrılır.

```
void OnCollisionEnter(Collision col)
    {
        // Çarpışma başladığı anda yapılacak işlemler
    }
```

- **OnCollisionStay:** Bu metot, iki nesne arasındaki çarpışma devam ettiği sürece sürekli olarak çağrılır.

```
void OnCollisionStay(Collision col)
    {
        // Çarpışma devam ettiği sürece yapılacak işlemler
    }
```

- **OnCollisionExit:** Bu metot, iki nesne arasındaki çarpışma sona erdiği anda çağrılır.

```
void OnCollisionExit(Collision col)
    {
        // Çarpışma sona erdiğinde yapılacak işlemler
    }
```



NOT

Çarpışma ile ilgili OnCollisionEnter, OnCollisionStay, ve OnCollisionExit metodlarının Collision türünde **col** isminde bir parametreleri vardır. Bu parametre, çarpışma ile ilgili bilgileri içerir. Örneğin çarpışan diğer nesnenin ismi ve etiketi gibi bilgilere col parametresi ile erişilir.

### Trigger (Tetikleyici)

Bir oyun nesnesi üzerindeki Collider'ın, **Is Trigger** özelliği aktif yapıldığında bu Collider bir **Trigger (Tetikleyici)** olarak yapılandırılmış olur. Böylece oyun nesnesi katı bir nesne gibi davranmaz, diğer nesnelerin içinden geçmesine izin verir ve fiziksel bir çarpışma algılanmaz. Trigger sadece nesne girişlerini ve çıkışlarını kontrol eder. Bu nesne giriş ve çıkışları gerçekleştiğinde ilgili nesnelere eklenmiş olan scriptlerde belirli isimlere sahip metotlar çağrılır. Nesne giriş çıkışı gerçekleştiği anda istenilen kodlar bu metotların içine yazılabilir. Örneğin oyun karakteri belirli bir ödül nesnesinin (altın yıldız gibi) içinden geçtiğinde oyuncunun puanı artırılabilir. Bu metotlar şunlardır:

- **OnTriggerEnter:** Bu metot, bir nesne bir tetikleyiciye girdiği anda çağrılır.

```
void OnTriggerEnter(Collider other)
    {
        // Tetikleyiciye giriş olduğu anda yapılacak işlemler
    }
```

- **OnTriggerStay:** Bu metot, bir tetikleyici içinde bir nesne kaldığı sürece çağrılır.

```
void OnTriggerStay(Collider other)
    {
        // Tetikleyici içinde kalındığı sürece yapılacak işlemler
    }
```

- **OnTriggerExit:** Bu metot, tetikleyiciden bir nesne çıktığı anda çağrılır.

```
void OnTriggerExit(Collider other)
    {
        // Tetikleyiciden çıkıldığı anda yapılacak işlemler
    }
```



NOT

OnTriggerEnter, OnTriggerStay ve OnTriggerExit tetikleme metotlarının Collider türünde **other** isimli bir parametreleri vardır. Bu parametre ile tetiklemeyi gerçekleştiren diğer nesnenin ismi ve etiketi gibi bilgilere erişilir.

İki oyun nesnesi arasında bir tetiklemenin gerçekleşmesi için en az bir nesnenin Rigidbody'si olması ve en az bir nesnenin Collider'ının is Trigger özelliğinin aktif olması gerekir. Ayrıca oyun nesnelerinden birisinin Collider'ının is Trigger özelliği aktif ise fiziksel çarpışma (Collision) olmaz.

İki nesne birbirine temas ettiğinde çarpışma (Collision) veya tetikleme (Trigger) olaylarının gerçekleşmesi ve buna bağlı olarak ilgili metotların çağırılması, ilgili nesnelerin Collider'larının ve Rigidbody'lerinin yapılandırmalarına bağlıdır. Tablo 1.13'te nesnelere eklenen Rigidbody bileşeninin özelliklerine göre hangi olayın (çarpışma veya tetikleme) gerçekleşeceği gösterilmektedir.

**Tablo 1.13:** Collider ve Rigidbody Özelliklerine Göre Gerçekleşen Çarpışmalar ve Tetiklemeler

| | **Static Collider** | **Rigidbody Collider** | **Kinematic Rigidbody Collider** | **Static Trigger Collider** | **Rigidbody Trigger Collider** | **Kinematic Rigidbody Trigger Collider** |
|---|---|---|---|---|---|---|
| **Static Collider** | | Collision | | | Trigger | Trigger |
| **Rigidbody Collider** | Collision | Collision | Collision | Trigger | Trigger | Trigger |
| **Kinematic Rigidbody Collider** | | Collision | | Trigger | Trigger | Trigger |
| **Static Trigger Collider** | | Trigger | Trigger | | Trigger | Trigger |
| **Rigidbody Trigger Collider** | Trigger | Trigger | Trigger | Trigger | Trigger | Trigger |
| **Kinematic Rigidbody Trigger Collider** | Trigger | Trigger | Trigger | Trigger | Trigger | Trigger |

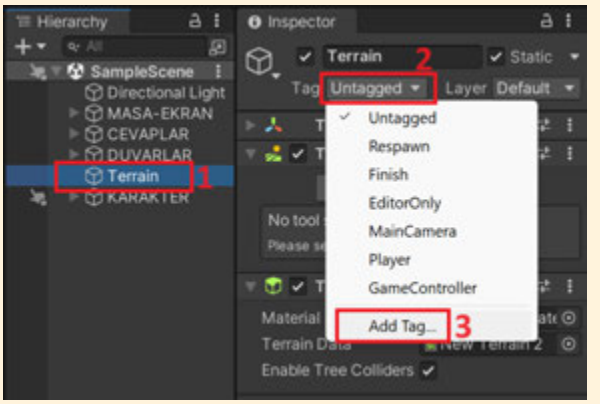
**KelimeMacerasi** isimli projedeki KARAKTER nesnesi, klavyeden boşluk tuşuna basıldığında Rigidbody bileşenine uygulanan kuvvet ile zıplamaktadır ama bu zıplama eyleminde şöyle bir sorun bulunmaktadır: KARAKTER nesnesi havadayken de klavyeden boşluk tuşuna basıldığında nesne tekrar zıplamaktadır. Bu sorunu düzeltmek için nesne, Terrain nesnesine temas ettiğinde zıplama (boşluk) tuşunu aktif yapmak ve zıplama gerçekleştikten sonra nesne tekrar zemine temas edene kadar zıplama tuşunu pasif yapmak gerekmektedir. Bu işlemler **Collision (Çarpışma)** metotları kullanılarak yapılabilir.

## 23. UYGULAMA

İşlem adımlarına göre **KelimeMacerasi** isimli projedeki KARAKTER oyun nesnesinin zıplama davranışını düzeltiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **SampleScene** isimli sahneyi açınız.

**2. Adım:** Terrain nesnesini seçiniz. Inspector panelinde Tag listesini açınız ve **Add Tag** düğmesine tıklayınız (Görsel 1.87).

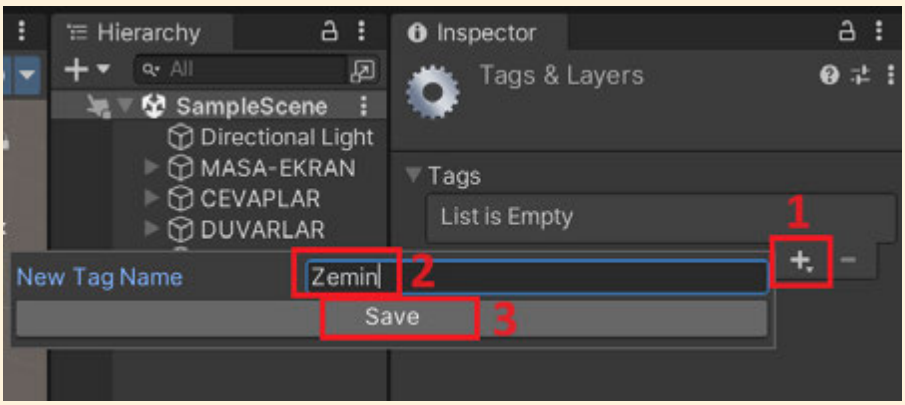



**Görsel 1.87:** Etiket eklenmesi

### NOT

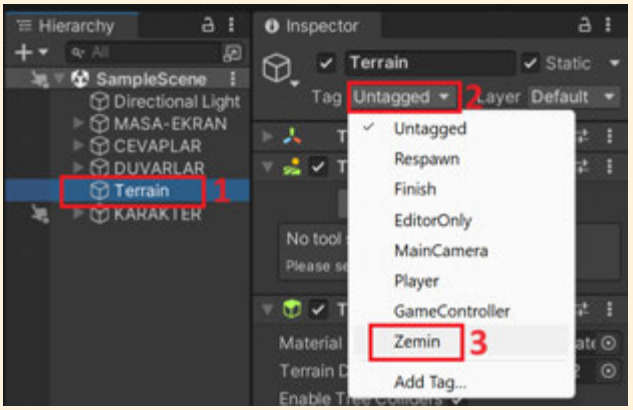
Tag'lar (etiketler), oyun nesnelerini gruplandırmak, tanımlamak ve oyun içinde belirli işlemleri gerçekleştirmek için kullanılır. Oyun nesnelerine etiket atanarak nesne çarpışmaları, Trigger tetiklemeleri, nesneler arası etkileşim gibi durumlarda ilgili nesneler etiketlerine göre kolay bir şekilde tespit edilebilir.

**3. Adım:** Add Tag düğmesine basıldıktan sonra Inspector panelinde açılan Tags & Layer alanında **+** düğmesine tıklayınız. Açılan alanda yazı kutusuna **Zemin** yazıp Save düğmesine tıklayınız. Zemin isimli etiket tanımlanmış olacaktır (Görsel 1.88).




**Görsel 1.88:** Etiket oluşturulması

**4. Adım:** Terrain nesnesini tekrar seçiniz. Inspector panelinde Tag listesini açınız ve Zemin tag'ını (etiketini) seçiniz. Terrain oyun nesnesinin etiketi Zemin olacaktır (Görsel 1.89).



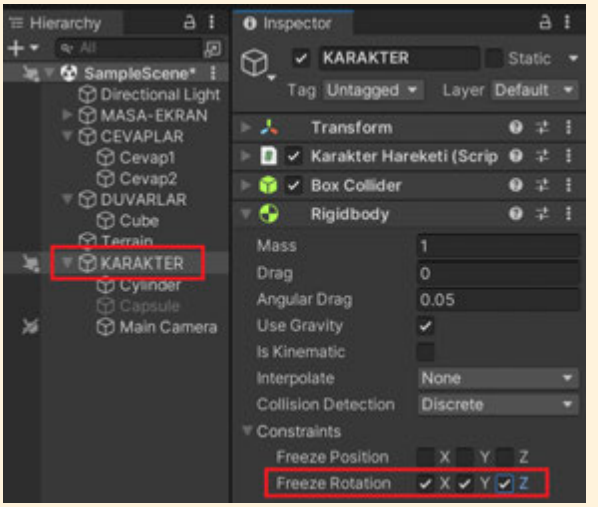

**Görsel 1.89:** Terrain nesnesine etiket atanması

**5. Adım: KarakterHareketi** isimli script dosyasını açınız ve aşağıdaki kodları ve değişiklikleri bu script dosyasına ekleyiniz.

```
public class KarakterHareketi : MonoBehaviour
{
    ..... (22.Uygulamada oluşturulan kodlar)
    private bool ziplamaYapabilir = true;
    void Start()
    {
    ..... (22.Uygulamada oluşturulan kodlar)
    }

    void Update()
    {
     ..... (22.Uygulamada oluşturulan kodlar)
        // Bu if yapısına "ziplamaYapabilir &&" ifadesi eklendi.
        if (ziplamaYapabilir && Input.GetButtonDown("Jump"))
        {
            Ziplama(ziplamaGucu);
        }
    }
    void Ziplama(float ziplamaGucu)
    {
     ..... (22.Uygulamada oluşturulan kodlar)
      ziplamaYapabilir = false;
    }
    void OnCollisionEnter(Collision collision)
    {
      // KARAKTER nesnesi ile çarpışan nesnenin etiketi Zemin ise
      // başka bir ifadeyle KARAKTER nesnesi Terrain ile temas
      // etti ise tekrar zıplamaya izin verilir.
      if (collision.gameObject.CompareTag("Zemin"))
        {
            ziplamaYapabilir = true;
        }
    }
}
```

**6. Adım:** KARAKTER nesnesi zıplarken bunun devrilmesi söz konusu olabilir. Bunu engellemek için KARAKTER nesnesini seçiniz. Rigidbody bileşeninde Constraints (Kısıtlamalar) kısmında Frezee Rotation'da X Y Z eksenlerini işaretleyiniz. Bu; nesnenin fiziksel olarak X, Y ve Z eksenlerindeki dönme hareketini durdurup devrilmesini engelleyecektir (Görsel 1.90).




**Görsel 1.90:** Nesnenin devrilmesinin engellenmesi

**7. Adım:** Projeyi çalıştırıp zıplama eylemini kontrol ediniz.

## SIRA SİZDE

1. **Engel** isimli etiket oluşturup bu etiketi DUVARLAR nesnesi içindeki Cube oyun nesnesine atayınız.
2. **Dogru** isimli etiket oluşturup bu etiketi CEVAPLAR nesnesi içindeki Cevap1 oyun nesnesine atayınız.
3. **Yanlis** isimli etiket oluşturup bu etiketi CEVAPLAR nesnesi içindeki Cevap2 oyun nesnesine atayınız.
4. KARAKTER nesnesi, Cube nesnesine temas ettiğinde Console panelinde **Engele Çarptınız!** mesajını, Cevap1 nesnesine temas ettiğinde Console panelinde **Doğru Cevap** mesajını, Cevap2 nesnesine temas ettiğinde Console panelinde **Yanlış Cevap** mesajını yazdırınız.

### Değerlendirme

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

### Kontrol Listesi

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Engel isimli etiket oluşturup bu etiketi DUVARLAR nesnesi içindeki Cube oyun nesnesine atadı. | | |
| 2. Dogru isimli etiket oluşturup bu etiketi CEVAPLAR nesnesi içindeki Cevap1 oyun nesnesine atadı. | | |
| 3. Yanlis isimli etiket oluşturup bu etiketi CEVAPLAR nesnesi içindeki Cevap2 oyun nesnesine atadı. | | |
| 4. KARAKTER nesnesi ilgili nesnelere çarptığında çarpma ile ilgili mesajları ekrana yazdırdı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

### 1.3.3. Sahne İşlemleri

Oyun motorunda sahneler (Scene), oyunların bir bölümünü temsil eder ve oyundaki o bölümle ilgili bütün oyun nesnelerini barındırır. Temsil ettikleri; bir ana menü, oyundaki farklı seviyeler, oyunun bitiş ekranı gibi bölümler olabilir. Geliştirilen oyunda birden çok sahne kullanılabilir, bu sahneler yönetilerek sahneler arasında geçişler yapılabilir.

Sahneler genellikle Assets klasörü altındaki **Scenes** klasörü içinde oluşturulur. Yeni bir sahne oluşturmak için Assets menüsünden Create>Scene komutuna tıklanır veya Project panelindeki Scenes klasörü üzerinde farenin sağ tuşu tıklanıp açılan menüde Create>Scene komutuna da tıklanır.

Sahne yönetimi için kod içinde **SceneManager** sınıfı kullanılır. **SceneManager.LoadScene("Sahne Adı")** metodu ile mevcut sahne kapanır ve belirtilen sahne yüklenir.

Bir sahneden başka bir sahneye geçiş yapıldığında bir önceki sahnedeki bazı verilerin yeni sahnede de kullanılması gerekebilir. **PlayerPrefs** sınıfı bir önceki sahnedeki verilerin, yeni sahnede veya başka sahnelerde de kullanımını sağlamaktadır. PlayerPrefs sınıfı, oyun motorunda küçük boyutlardaki verileri depolamak ve bu verilere erişmek için kullanılır. PlayerPrefs; genellikle oyuncu tercihleri, ilerlemesi, puanı gibi verileri saklamak için kullanılır. Bu veriler, bir sahneden başka bir sahneye geçildiğinde veya oyun kapatılıp tekrar başlatıldığında da kullanılabilir. PlayerPrefs'lerin yapısı bir anahtar ve bu anahtarın değeri şeklindedir.

**Örnek:**

```
// String veri türünde, Ahmet verisi OyuncuAdi ismiyle saklanır.
PlayerPrefs.SetString("OyuncuAdi", "Ahmet");
// Integer veri türünde, 100 verisi Puan ismiyle saklanır.
PlayerPrefs.SetInt("Puan", 100);
```

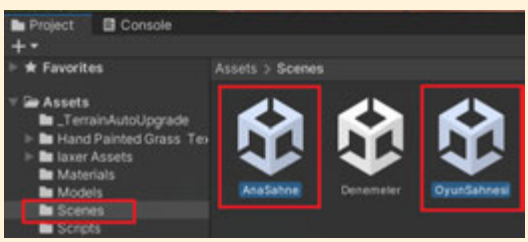
```
// Veriler kalıcı hale getirilir.
PlayerPrefs.Save();
```

```
// Puan isimli veri silinir
PlayerPrefs.DeleteKey("Puan");
// Tüm PlayerPrefs verileri silinir.
PlayerPrefs.DeleteAll();
```

#### 24. UYGULAMA

İşlem adımlarına göre yeni bir sahne ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **SampleScene** isimli sahneyi açınız.
**2. Adım:** Assets klasörü altındaki Secenes klasörü üzerinde farenin sağ tuşuna tıklayıp gelen menüden Create>Scene komutuna tıklayınız. Yeni oluşturulan sahnenin ismini **AnaSahne** olarak değiştiriniz. Var olan SampleScene isimli sahnenin ismini de **OyunSahnesi** olarak değiştiriniz (Görsel 1.91).




**Görsel 1.91:** Sahnenin oluşturulması ve sahne isminin değiştirilmesi

**3. Adım:** AnaSahne sahnesine fare ile çift tıklayarak bu sahneyi açınız.

SIRA SİZDE

Yeni bir sahne oluşturunuz ve bu sahnenin ismini SahneDeneme olarak değiştiriniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Yeni bir sahne oluşturdu. | | |
| 2. Yeni sahnenin ismini SahneDeneme olarak değiştirdi. | | |
| 3. SahneDeneme isimli sahneyi açtı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.3.4. Kullanıcı Arayüzü İşlemleri

Oyun motorunda **Kullanıcı Arayüzü (User Interface, UI);** oyun veya uygulama içinde kullanıcının etkileşimde bulunabileceği ve bilgi alabileceği düğmeler, metin alanları, resimler, paneller gibi ögelerin tamamını ifade eder. Kullanıcı arayüzü ögeleri ile menüler oluşturulabilir, kullanıcılara bilgi vermek için metinler ve resimler görüntülenebilir. Sahneye UI ögeleri eklemek için Game Object menüsünde UI seçeneği ile açılan menü kullanılır. Ayrıca Hierarchy panelinde farenin sağ tuşu tıklanıp açılan menüde UI seçeneği tıklandığında açılan menü de kullanılabilir. Oyun motorunda kullanıcı arayüzünün temel ögeleri şunlardır:

- **Canvas (Tuval):** UI ögelerini kapsayan nesnedir. Canvas, UI elemanlarını içeren ve düzenlenmesini sağlayan bir alanı temsil eder.
- **Panel:** Canvas içindeki diğer UI elemanlarını gruplamak ve düzenlemek için kullanılan nesnedir.
- **Text - TextMeshPro:** Metin görüntülemek için kullanılan nesnedir.
- **Button - TextMeshPro (Düğme):** Kullanıcının tıklayabildiği etkileşimli bir düğmedir.
- **Input Field - TextMeshPro (Giriş Alanı):** Kullanıcının metin girebildiği yazı alanıdır.
- **Dropdown - TextMeshPro (Açılır Menü):** Kullanıcının bir listeden seçim yapmasına izin veren nesnedir.
- **Image (Görsel):** Resim görüntülemek için kullanılan nesnedir.

NOT

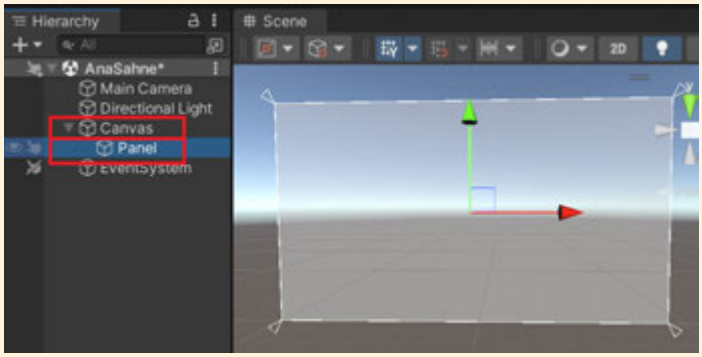
**TextMeshPro,** oyun motorunda metin görüntülemek için kullanılan bir pakettir. İlgili UI nesnelerinde metinle ilgili daha fazla özelleştirme ve düzenleme seçeneği sunar.

25. UYGULAMA

İşlem adımlarına göre **KelimeMacerasi** isimli projede **AnaSahne** isimli sahnede kullanıcı arayüzü nesnelerini kullanarak bir oyun menüsü oluşturunuz.

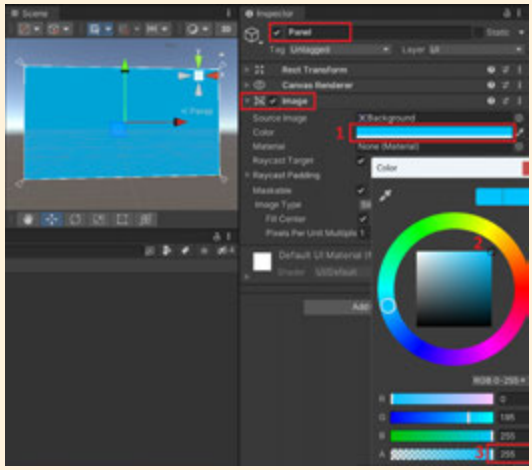
**1. Adım: KelimeMacerasi** isimli projedeki **AnaSahne** isimli sahneyi açınız.
**2. Adım:** GameObject menüsünden **UI>Canvas** komutuyla sahneye bir Canvas ekleyiniz.

**3. Adım:** Yeni oluşturulan Canvas nesnesini seçiniz ve GameObject menüsünden **UI>Panel** komutuyla Canvas nesnesinin altına bir Panel ekleyiniz (Görsel 1.92).




**Görsel 1.92:** Panel nesnesin eklenmesi

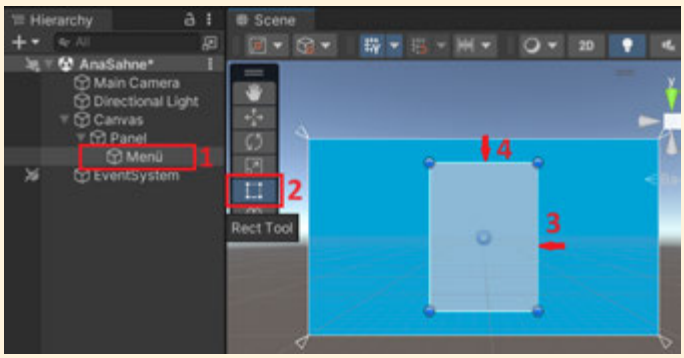
**4. Adım:** Panel nesnesini seçiniz. Panel nesnesinin **Image** bileşininde Color özelliğindeki renk kutusuna tıklayınız. Açılan Color panelinde renk paletinde maviyi seçiniz ve A (Şeffaflık) değerini 255 yapınız (Görsel 1.93).



**Görsel 1.93:** Panelin renginin ayarlanması

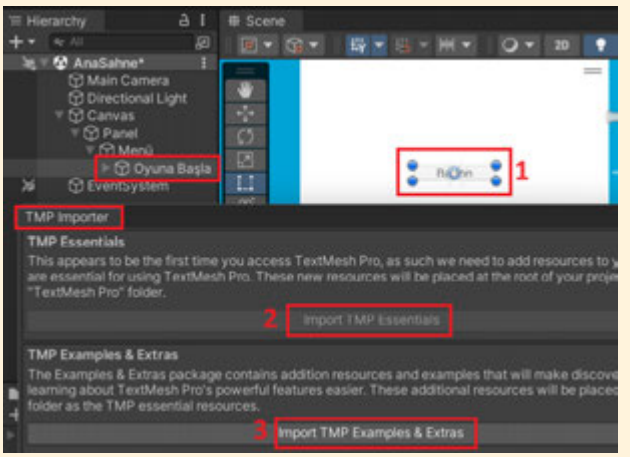
**5. Adım:** Panel isimli nesnenin altında bir tane Panel oluşturunuz ve bu panelin ismini **Menü** olarak değiştiriniz.

**6. Adım:** Menü isimli nesneyi seçiniz. Menü panelinin boyutunu Görsel 1.94'teki gibi ayarlayınız. Bunu yapmak için Transform araç kutusundaki **Rect Tool** aracını seçiniz. Panelin uzun kenarlarından birini fare ile tutup, aynı zamanda klavyeden **Alt** tuşuna basarak fareyi sürükleyiniz. Panelin yataydaki uzunluğu simetrik olarak küçülecektir. Aynı işlemi panelin dikeydeki uzunluğu için de yapınız.



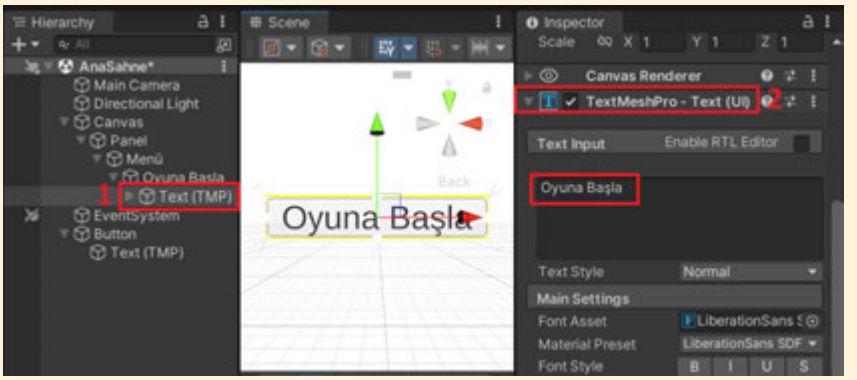

**Görsel 1.94:** Menü panelinin boyutunun değiştirilmesi

**7. Adım:** Menü panelinin rengini beyaz ve A (Şeffaflık) değerini 255 yapınız.
**8. Adım:** Hierarchy panelinde Menü nesnesinin üzerinde farenin sağ tuşuna tıklayıp gelen menüde **UI>Button TextMeshPro** komutuna tıklayarak bir Button nesnesi ekleyiniz. Bunu yaptığınızda TextMeshPro paketini yüklemeniz için **TMP Importer** penceresi açılacaktır. Bu pencerede **Import TMP Essentials** düğmesine tıklayınız. Ardından **Import TMP Examples & Extras** düğmesine tıklayınız. Button nesnesinin ismini Oyuna Başla olarak değiştiriniz (Görsel 1.95).



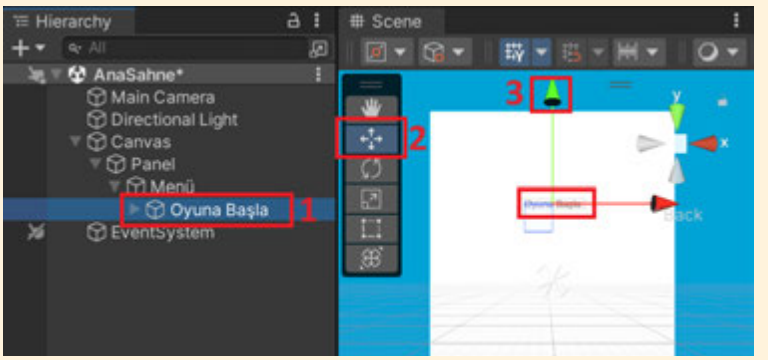

**Görsel 1.95:** Butonun eklemesi ve TextMeshPro'nun yüklenmesi

**9. Adım:** Oyuna Başla butonunun sahne üzerinde görünen yazısını değiştirmek için butonu genişletiniz ve butonun altındaki **Text (TMP)** nesnesini seçiniz. Inspector panelinde **TextMeshPro – Text (UI)** bileşeninde Button ifadesi yerine Oyuna Başla ifadesini yazınız (Görsel 1.96).



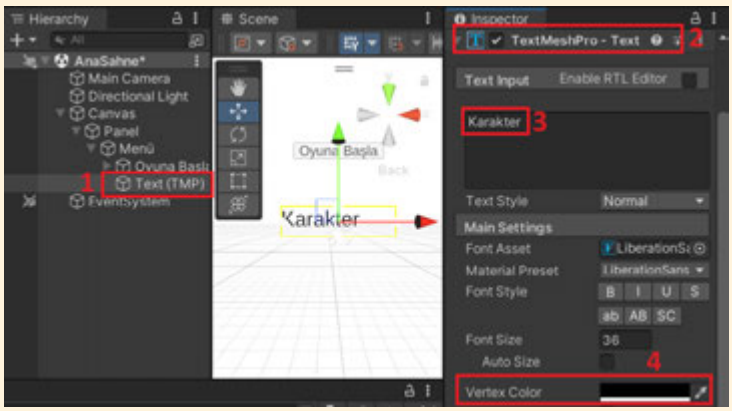

**Görsel 1.96:** Butonun düzenlenmesi

**10. Adım:** Oyuna Başla butonunu seçiniz ve Transform araç kutusundaki Move Tool aracını seçiniz. Sahne üzerinde butonu Y ekseninde bir miktar yukarı doğru taşıyınız (Görsel 1.97).



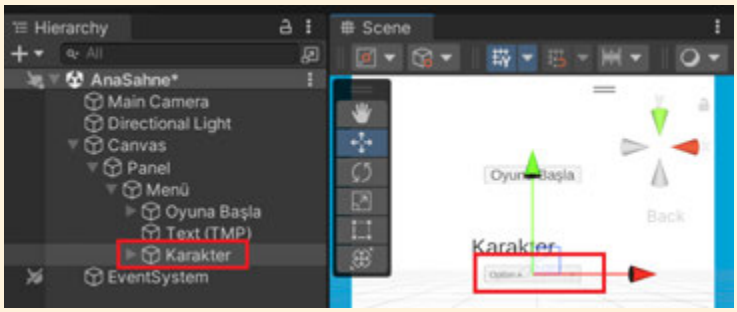

**Görsel 1.97:** Butonun Y eksenindeki konumunun değiştirilmesi

**11. Adım:** Hierarchy panelinde Menü nesnesinin üzerinde farenin sağ tuşuna tıklayıp açılan menüde **UI>Text TextMeshPro** komutuna tıklayarak bir metin nesnesi ekleyiniz.
**12. Adım:** Bu metin nesnesini seçip Inspector panelinde **TextMeshPro – Text (UI)** bileşenindeki New Text ifadesi yerine Karakter ifadesini yazınız. **Vertex Color** özelliğinde de rengini siyah yapınız (Görsel 1.98).



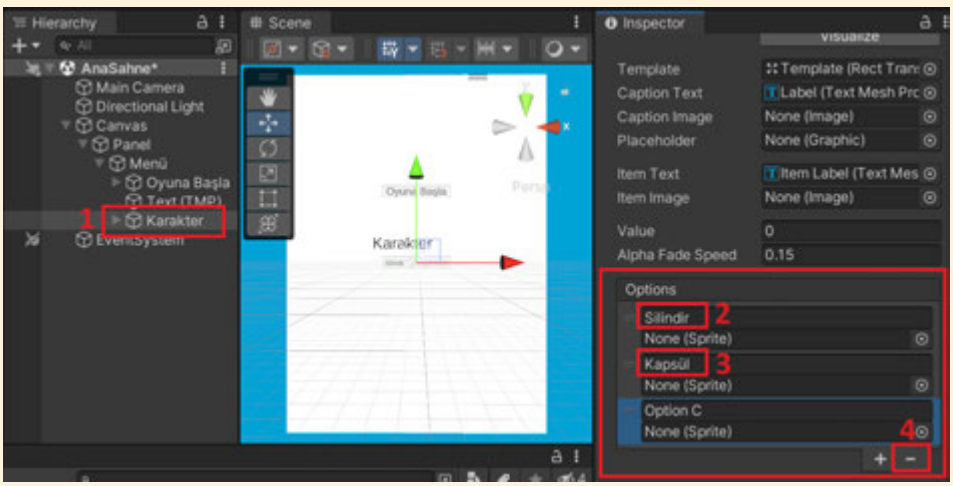

**Görsel 1.98:** Metnin düzenlenmesi

**13. Adım:** Hierarchy panelinde Menü nesnesinin üzerinde farenin sağ tuşuna tıklayıp gelen menüde **UI>Dropdown TextMeshPro** komutuna tıklayarak bir Dropdown (açılır liste) nesnesi ekleyiniz. Nesnenin ismini Karakter olarak değiştiriniz ve konumunu ayarlayınız (Görsel 1.99).



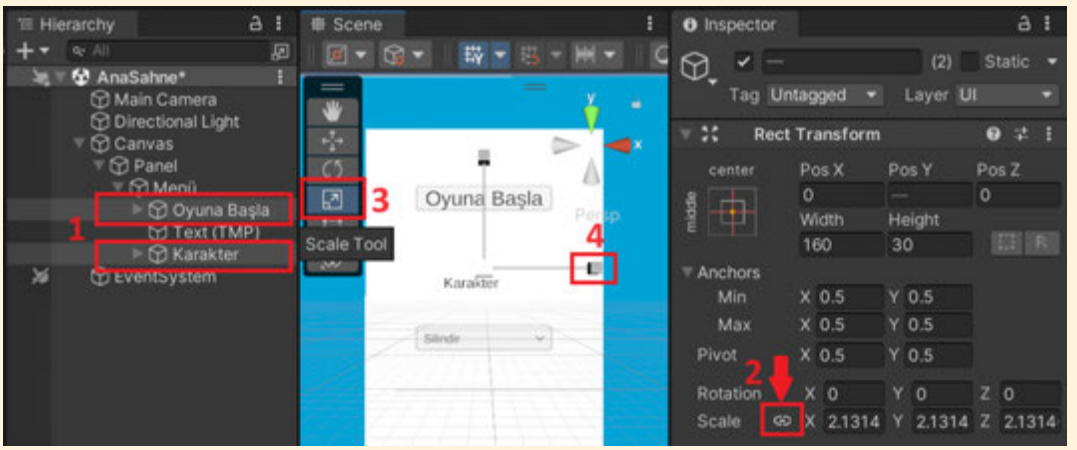

**Görsel 1.99:** Dropdown eklenmesi ve özelliklerinin ayarlanması

**14. Adım:** Karakter Dropdown'u seçiniz. Inspector panelinde Dropdown – TextMeshPro bileşeninde Options kısmında Option A ifadesi yerine Silindir, Option B ifadesi yerine Kapsül yazınız. Option C ifadesinin bulunduğu alanı seçip - düğmesine basıp bu alanı siliniz (Görsel 1.100).



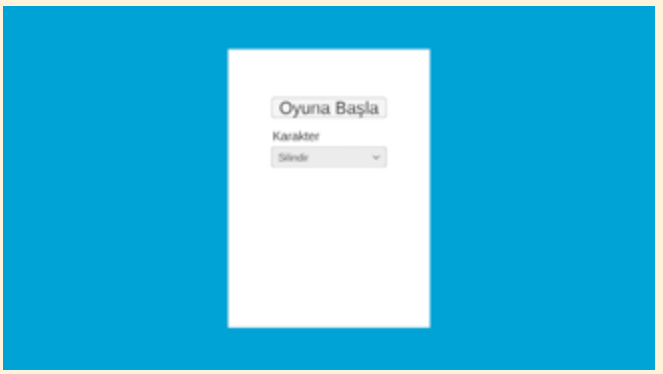

**Görsel 1.100:** Dropdown seçeneklerinin ayarlanması

**15. Adım:** Oyuna Başla butonu ve Karakter açılır listesini, klavyeden Ctrl tuşuna basarak fare ile birlikte seçiniz. Inspector panelinde Scale'ın sağındaki X, Y ve Z değerlerini birlikte değiştirmeyi sağlayan düğmeye tıklayınız. Daha sonra Scale Tool aracını seçiniz ve sahnedeki tutamaçlardan birini kullanarak seçili olan iki nesnenin boyutlarını Görsel 1.101'deki gibi büyütünüz.




**Görsel 1.101:** Butonun ve açılır listenin boyutunun ayarlanması

**16. Adım:** Menü panelindeki üç nesnenin konumunu ve görünümünü Görsel 1.102'deki gibi ayarlayınız.




**Görsel 1.102:** Nesnelerin konumunun ayarlanması

SIRA SİZDE

Menü paneline bir adet metin yazısı, bir adet açılır liste ve bir adet buton ekleyiniz ve Görsel 1.103'te görünen Menü panelinin tasarımını gerçekleştiriniz.




**Görsel 1.103:** Menü panelinin son hâli

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Menü paneline bir adet metin yazısı ekledi. | | |
| 2. Menü paneline bir adet açılır liste ekledi. | | |
| 3. Menü paneline bir adet buton ekledi. | | |
| 4. Menü panelinin tasarımını gerçekleştirdi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 26. UYGULAMA

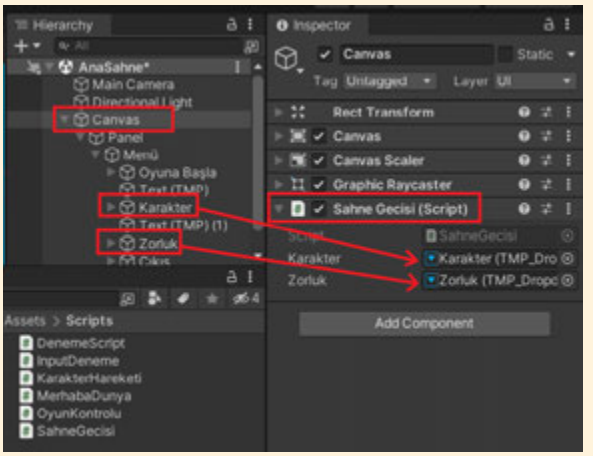
İşlem adımlarına göre **KelimeMacerasi** isimli projede oyunu başlatmak için sahne geçişini ve OyunSahnesi isimli sahnedeki düzenlemeleri gerçekleştiriniz.

**1. Adım: KelimeMacerasi** isimli projedeki **AnaSahne** isimli sahneyi açınız.
**2. Adım:** Scripts klasöründe **SahneGecisi** isimli bir script dosyası oluşturunuz. Bu script dosyası **içine** aşağıdaki kodları yazınız.

```
using System.Collections.Generic;
using UnityEngine;
using TMPro; // TextMeshPro'ya sahip UI nesneleri için gerekli
             // sınıftır.
using UnityEngine.SceneManagement; // Sahne işlemleri için
                                   // gerekli sınıftır.
public class SahneGecisi : MonoBehaviour
{
  // Karakter ve Zorluk açılır listelerini tutacak olan
  // değişkenler.
  public TMP_Dropdown karakter, zorluk;
  void Start()
  {
    // karakterSecenegi isimli bir PlayerPrefs'ın değeri alınır,
    // böyle bir değer    yok ise 0 değeri alınır.
    int saklananKarakterSecenegi = PlayerPrefs.
    GetInt("karakterSecenegi", 0);
    // Karakter açılır listesinin seçeneği
    // saklananKarakterSecenegi değerine eşitlenir. Bu değer 0
    // ise açılır listede Silindir, 1 ise Kapsül ifadesi görünür.
    karakter.value = saklananKarakterSecenegi;
  }
  public void SahneDegistir(string sahneAdi)
  {
```

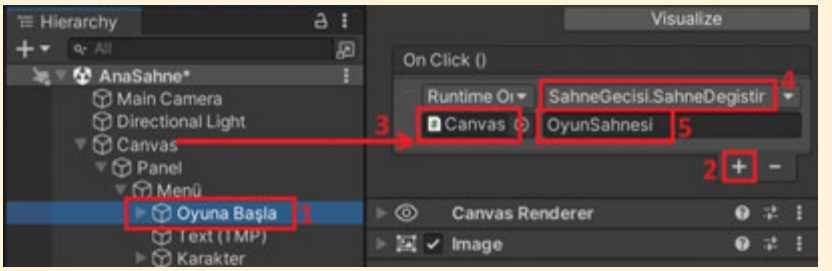
```
    // OyunSahnesi aktif olduğunda Karakter açılır listesinin
    // seçilen değeri (0 veya 1) karakterSecenegi isimli
    // PlayerPrefs olarak belirlenir. OyunSahnesi'nde
    // karakterSecenegi değeri okutulur ve bu değere göre oyun
    // karakteri olarak silindir veya kapsül aktif edilir.
    PlayerPrefs.SetInt("karakterSecenegi", karakter.value);
    // PlayerPrefs değerleri cihaza kaydedilir.
    PlayerPrefs.Save();
    // Belirtilen sahneye geçiş yapılır.
    SceneManager.LoadScene(sahneAdi);
  }
  public void OyunuKapat()
  {
    // Oyun bir cihazda oynanıyorsa oyundan çıkış yapılır.
    Application.Quit();
  }
}
```

**3. Adım:** Canvas nesnesini seçiniz. SahneGecisi script dosyasını Canvas nesnesine ekleyiniz. Inspector panelinde Sahne Gecisi bileşenindeki Karakter referans değişkenine, Karakter açılır listesini sürükleyip ekleyiniz. Zorluk referans değişkenine de Zorluk açılır listesini sürükleyip ekleyiniz (Görsel 1.104).



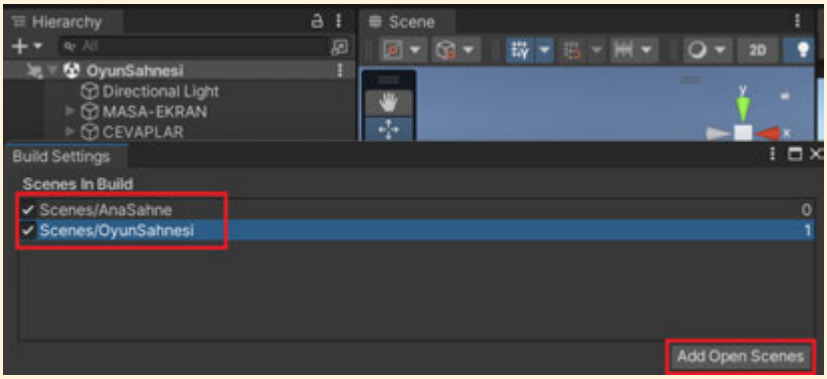

**Görsel 1.104:** Canvas nesnesine SahneGecisi script dosyasının eklenmesi

**4. Adım:** Menü panelinin altındaki Oyuna Başla butonunu seçiniz. Inspector panelinde bu butona ait Button bileşenindeki **On Click ()**'in altındaki **+** düğmesine tıklayınız. Gelen kutulardan None yazan kutuya, Hierarchy panelinden Canvas nesnesini sürükleyip ekleyiniz. No Function yazan listeyi açınız ve bu listeden **SahneGecisi** sınıfını seçiniz. Açılan listeden de **SahneDegistir(String)** metodunu seçiniz. Gelen boş metin kutusuna aktif yapılacak sahnenin ismi olarak **OyunSahnesi** yazınız. Bu işlemler Oyuna Başla butonuna tıklayınca OyunSahne'sinin açılmasını sağlayacaktır (Görsel 1.105).



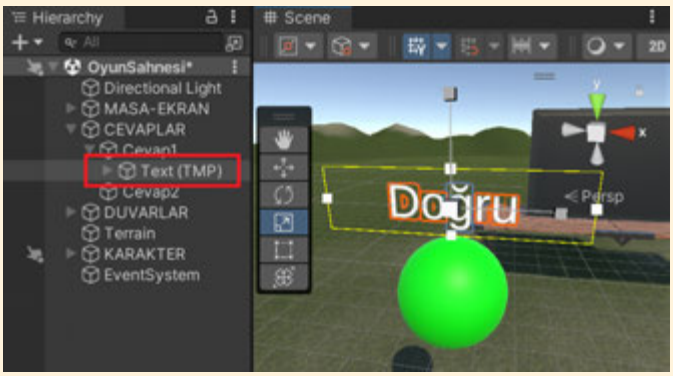

**Görsel 1.105:** Oyuna Başla butonun tıklama olayının ayarlanması

**5. Adım:** Hem AnaSahne hem de OyunSahnesi isimli sahnenin oyun motoru tarafından sahne işlemlerine (sahneyi açma gibi) dâhil edilebilmesi için bu sahnelerin **Build Settings** penceresinde **Scenes In Build** kısmına eklenmeleri gerekmektedir. Bunun için AnaSahne isimli sahne açıkken File menüsünden **Build Settings** komutuna tıklayınız. Gelen pencerede **Add Open Scene** düğmesine tıklayınız. Daha sonra OyunSahnesi isimli sahneyi de açıp aynı işlemi bu sahne için de yapınız (Görsel 1.106).



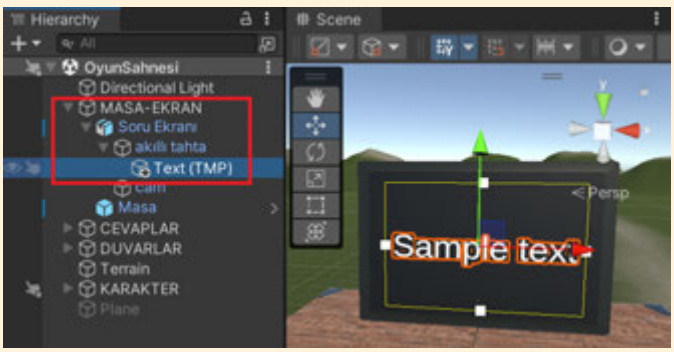

**Görsel 1.106:** Sahnelerin Scenes In Build'e eklenmesi

**6. Adım:** Projeyi çalıştırınız. Oyuna Başla butonuna tıklayıp Oyun Sahnesi'nin açıldığını gözlemleyiniz.
**7. Adım:** Projenin çalışmasını durdurunuz. **OyunSahnesi** isimli sahneyi açınız. Hiearchy panelinde CEVAPLAR nesnesinin altındaki Cevap1 nesnesinin üzerinde farenin sağ tuşuna tıklayınız. Açılan menüde 3D Object>Text TextMeshPro komutuna tıklayıp Cevap1 nesnesinin altında bir 3D metin nesnesi ekleyiniz.
**8. Adım:** Eklediğiniz metin nesnesinin görünümünü ve yazısını Görsel 1.107'deki gibi ayarlayınız.



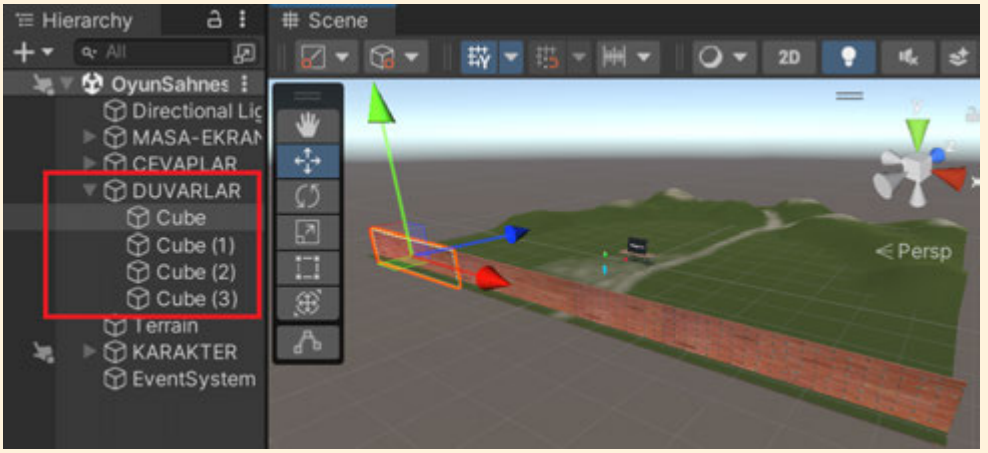

**Görsel 1.107:** 3D Metin yazısının oluşturulması

**9. Adım: akıllı tahta** nesnesinin altında bir 3D metin nesnesi ekleyiniz. Eklediğiniz metin nesnesinin görünümünü ve yazısını Görsel 1.108'deki gibi ayarlayınız.



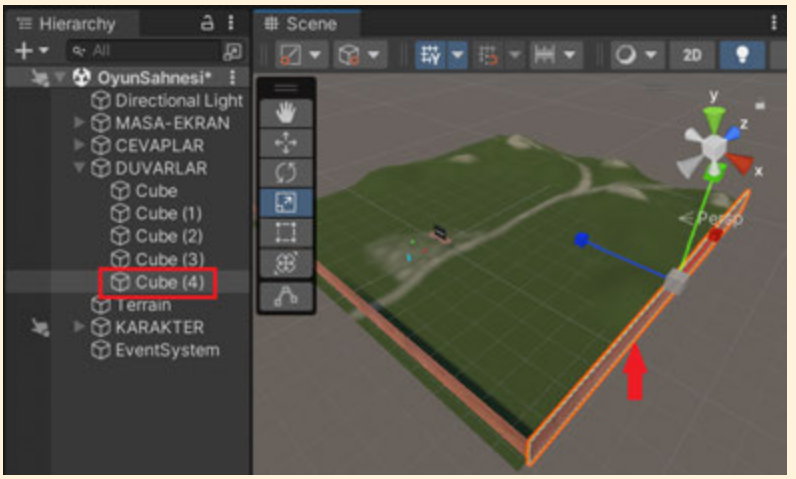

**Görsel 1.108:** Akıllı tahtada 3D metin yazısının oluşturulması

**10. Adım:** DUVARLAR nesnesi altındaki Cube nesnesinden üç adet daha çoğaltıp bu nesnelerin konumlarını ve görünümlerini Görsel 1.109'daki gibi ayarlayınız.



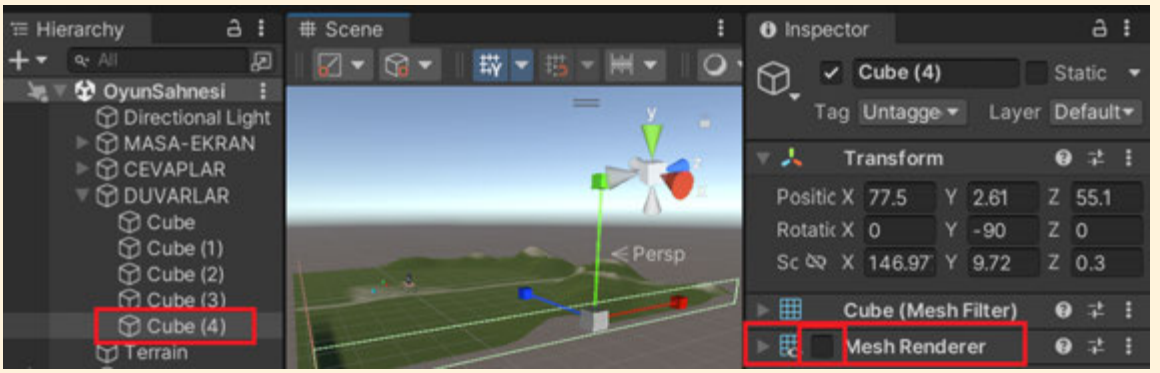

**Görsel 1.109:** Duvar engellerinin oluşturulması

**11. Adım:** DUVARLAR nesnesi altındaki Cube (3) nesnesinden bir adet çoğaltınız ve bu nesnenin konumunu ve görünümünü Görsel 1.110'daki gibi ayarlayınız.



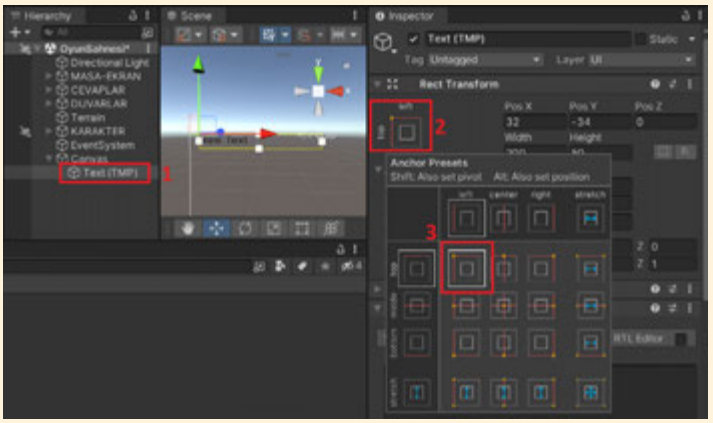

**Görsel 1.110:** Büyük engelin oluşturulması

**12. Adım:** Cube (4) nesnesinin Mesh Renderer bileşenin görünürlüğünü pasif yapınız (Görsel 1.111). Böylece görünmeyen bir engeliniz olacaktır.




**Görsel 1.111:** Görünmeyen engel oluşturulması

**13. Adım:** Sahneye bir Canvas nesnesi ekleyiniz. Canvas nesnesinin altına bir Text – TextMeshPro metin kutusu nesnesi ekleyiniz. Bu metin kutusunu seçiniz ve **Rect Transform** bileşeni altındaki **Anchor Presets** düğmesine basınız. Açılan alanda, klavyden **Shift+Alt** tuşlarına basılı tutarak **sol-üst** düğmesine tıklayınız. Böylece metin kutusu Canvas nesnesinin sol üst tarafını referans alarak konumlanacaktır. Metin kutusunun konumunu, boyutunu ve görünümünü ayarlayınız (Görsel 1.112).



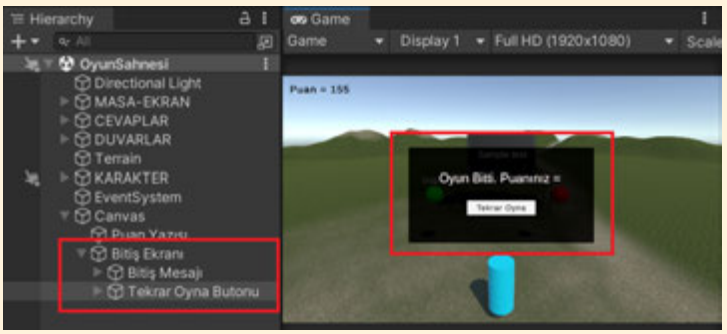
**Görsel 1.112:** Metin kutusunun eklenmesi ve özelliklerinin ayarlanması

**14. Adım:** Metin kutusunun ismini **Puan Yazısı** olarak değiştiriniz ve üzerinde görünen yazıyı **Puan = 0** olarak değiştiriniz (Görsel 1.113).




**Görsel 1.113:** Metin kutusunun isminin ve yazısının değiştirilmesi

**15. Adım:** Canvas nesnesi altına bir panel ekleyiniz. Panel ismini **Bitiş Ekranı** olarak değiştiriniz. Panel nesnesinin içinde bir metin kutusu ve bir buton nesnesi oluşturunuz. Metin kutusunun ismini **Bitiş Mesajı,** butonun ismini **Tekrar Oyna Butonu** olarak değiştiriniz. Panelin ve içindeki nesnelerin görünümünü Görsel 1.114'teki gibi ayarlayınız.

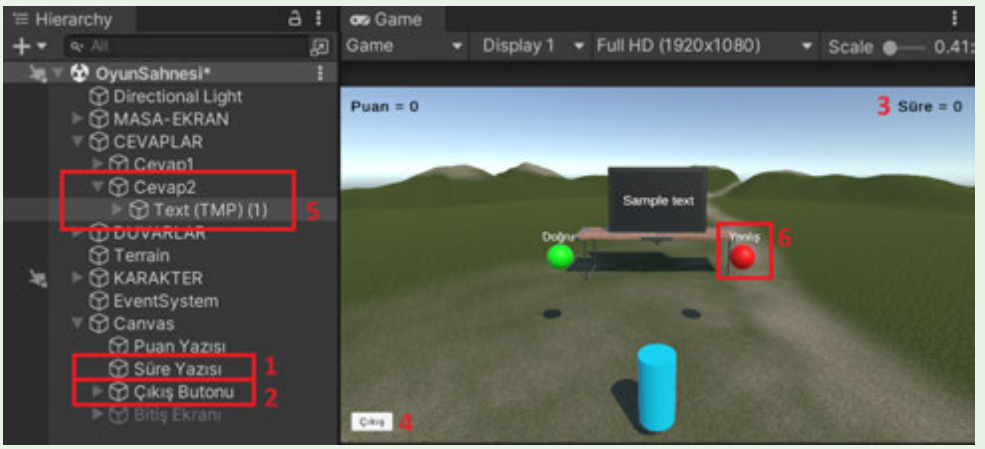



**Görsel 1.114:** Bitiş Ekranı'nın oluşturulması

**16. Adım:** Bitiş Ekranı nesnesinin görünürlüğünü Inspector panelinden pasif yapınız. Bu ekran oyun bittiğinde gösterilecektir.

SIRA SİZDE

1. OyunSahnesi isimli sahnede Canvas nesnesi altına bir metin kutusu ve bir buton nesnesi ekleyiniz.
2. Metin kutusunun ismini **Süre Yazısı,** buton nesnesinin ismini **Çıkış Butonu** olarak değiştiriniz ve bu iki nesnenin görünümünü Görsel 1.115'teki gibi ayarlayınız.
3. CEVAPLAR nesnesinin altındaki Cevap2 nesnesinin altına bir 3D metin kutusu ekleyiniz ve bu metin kutusunun yazısını Yanlış olarak değiştiriniz (Görsel 1.115).




**Görsel 1.115:** Yanlış yazısı, Süre Yazısı ve Çıkış Butonu'nun oluşturulması

4. Çıkış Butonu'na tıklayınca AnaSahne sahnesinin aktif olması için gerekli işlemleri yapınız. Bunun için gerekli olan metodu KARAKTER nesnesine ekli olan KarakterHareketi script dosyası içine yazınız. Bu script dosyasında using bildirimleri bölümünde oyun motorunun **SceneManagement** sınıfını eklemeyi unutmayınız. KARAKTER nesnesini de Çıkış Butonu'nun On Click () alanına ekleyerek ilgili sahne geçişi metodunu seçebilirsiniz.
5. AnaSahne isimli sahneyi açınız. Bu sahnedeki Zorluk isimli açılır listenin değeriyle ilgili PlayerPrefs işlemlerini sağlayan kodları (Karakter isimli açılır listenin değeri ile ilgili kodlar gibi) SahneGecisi isimli script dosyasında yazınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Canvas nesnesi altına arayüz nesnelerini ekledi. | | |
| 2. Eklediği nesnelerin isimlerini değiştirdi. | | |
| 3. Eklediği nesnelerin görünümünü ayarladı. | | |
| 4. Cevap2 nesnesinin altına bir 3D metin kutusu ekleyip metnini değiştirdi. | | |
| 5. Çıkış Butonu'nun On Click() ayarlarını yaptı. | | |
| 6. Zorluk isimli açılır listenin değeriyle ilgili PlayerPrefs işlemlerini sağlayan kodları SahneGecisi isimli script dosyasında yazdı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

### 1.3.5. JSON Veri İşlemleri

**JSON (JavaScript Object Notation),** programlamada veri işlemlerinde kullanılan bir veri formatıdır. JSON, insanlar tarafından okunabilir bir metin formatına sahiptir ve programlar arasında veri alışverişini kolaylaştırmak amacıyla kullanılır. Oyun motorunda geliştirilecek bir oyun veya uygulamada kullanılacak çok satırlı veri setleri JSON dosyalarında saklanabilir. Örneğin bu veri setleri kullanıcıya sorulacak soruları içerebilir. KelimeMacerası isimli projede kullanıcılara gösterilecek kelimeler ve bu kelimelerde yazım yanlışı olup olmadığı bilgisi bir JSON dosyasında tutulacaktır ve oyun içinde bu dosyadan kelimeler ve yazım bilgileri sırayla okunacaktır. Bir JSON dosyasının yapısı şu şekildedir:

```
[
    {"Kelime": "çevrim içi", "DogruMu": true},
    {"Kelime": "çevrimiçi","DogruMu": false},
    {"Kelime": "küsur","DogruMu": true},
    {"Kelime": "küsür", "DogruMu": false},
]
```

Yukarıdaki JSON yapısında 4 elemanlı bir dizi yapısı vardır ve her eleman bir veri setidir. Bu veri setlerinde 2 anahtar ve bunların değerleri vardır. Bunlar Kelime anahtarı ve DogruMu anahtarıdır. Örneğin birinci veri setinde Kelime anahtarının değeri **çevrim içi**'dir, DogruMu anahtarının değeri **true**'dur. Oyunda kullanıcıya gösterilecek olan kelime ve bu kelimede yazım yanlışı olup olmadığı bu anahtarlar ve onların değerleri ile tespit edilebilmektedir.

Böyle bir veri setine sahip JSON dosyasındaki verileri oyun motoru içinde kullanmak için aşağıdaki işlemler gerçekleştirilir:

**1.** Verilerin içinde bulunduğu JSON dosyası (kelimeler.json), Assets klasörü içinde oluşturulan **Resources** klasörü içinde saklanır.

**2.** JSON dosyasındaki bir veri setini kod içinde temsil etmek için bir serileştirilebilen sınıf oluşturulur. Bir sınıfın serileştirilebilmesi, sınıfın ve sınıfın değişkenlerinin JSON gibi bir dosya biçimine dönüştürülebilmesi anlamında gelir.

```
[System.Serializable]
    // JSON dosyasındaki veri setlerine karşılık gelen sınıftır.
    public class Soru     {
        public string Kelime;
        public bool DogruMu;
    }
```

**3.** Resources klasöründe bulunan kelimeler.json dosyasının içeriğini, kod içinde Soru sınıfı türünden bir liste değişkenine aktarmak için aşağıdaki kodlar kullanılır:

```
// Soru sınıfı türünden bir Liste değişkeni tanımlanır.
private List<Soru> sorularListesi;
// Resources klasöründeki kelimeler isimli JSON dosyasının içindeki
// bütün veriler bir string değişkene atanır.
public void VerileriJSONdanOku()
{
   string jsonText = Resources.Load<TextAsset>("kelimeler").text;
   if (!string.IsNullOrEmpty(jsonText))
     {
     // jsonText değişkenindeki veriler, Soru sınıfı türündeki
     // sorularListesi      isimli liste değişkenin içine aktarılır.
     sorularListesi = JsonConvert.
     DeserializeObject<List<Soru>>(jsonText);
   }
}
```

**4.** sorularListesi isimli listedeki her kelime ve yazım bilgisine kod ile ulaşılabilir.

## 27. UYGULAMA

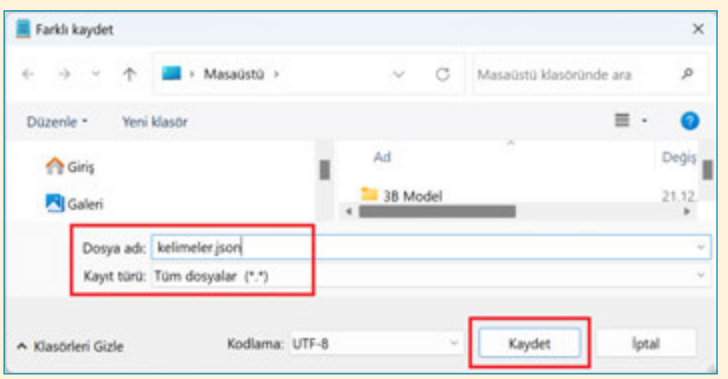
İşlem adımlarına göre **KelimeMacerasi** isimli projede JSON dosyasını ve oyunun temel işlevlerini sağlayan script dosyasını oluşturunuz. Gerekli atamaları yapınız.

**1. Adım: KelimeMacerasi** isimli projedeki **OyunSahnesi** isimli sahneyi açınız.

**2. Adım: Assets** klasörü altında **Resources** isimli bir klasör oluşturunuz. Masaüstünde bir metin belgesi dosyası oluşturunuz ve bu dosyanın içine aşağıdaki JSON verilerini yazınız.

```
[
    {"Kelime": "Yaradan", "DogruMu": true},
    {"Kelime": "Yaratan","DogruMu": false},
    {"Kelime": "yalnız", "DogruMu": true},
    {"Kelime": "yanlız", "DogruMu": false},
    {"Kelime": "çevrim içi", "DogruMu": true},
    {"Kelime": "çevrimiçi","DogruMu": false},
    {"Kelime": "küsur","DogruMu": true},
    {"Kelime": "küsür", "DogruMu": false},
    {"Kelime": "hafriyat", "DogruMu": true},
    {"Kelime": "harfiyat","DogruMu": false},
    {"Kelime": "ünvan", "DogruMu": true},
    {"Kelime": "unvan", "DogruMu": false},
    {"Kelime": "şoför", "DogruMu": true},
    {"Kelime": "şöför", "DogruMu": false},
    {"Kelime": "doküman", "DogruMu": true},
    {"Kelime": "döküman","DogruMu": false},
    {"Kelime": "acayip","DogruMu": true},
    {"Kelime": "acaip", "DogruMu": false},
    {"Kelime": "zarafet", "DogruMu": true},
    {"Kelime": "zerafet", "DogruMu": false}
]
```

**3. Adım:** Bu metin dosyası açıkken Dosya menüsünden Farklı Kaydet komutuna basınız. Gelen pencerede dosya adına **kelimeler.json,** kayıt türünü **Tüm Dosyalar (*.*)** olarak seçiniz ve kaydet düğmesine basınız. Masaüstündeki kelimeler.json dosyasını oyun motorunda oluşturduğunuz Resources klasörü içine taşıyınız (Görsel 1.116).




**Görsel 1.116:** JSON Dosyasının kaydedilmesi

**4. Adım:** Scripts klasöründe **KelimeOyunKontrolu** isimli bir script dosyası oluşturunuz. Bu script dosyasının içine şu kodları yazınız.

NOT

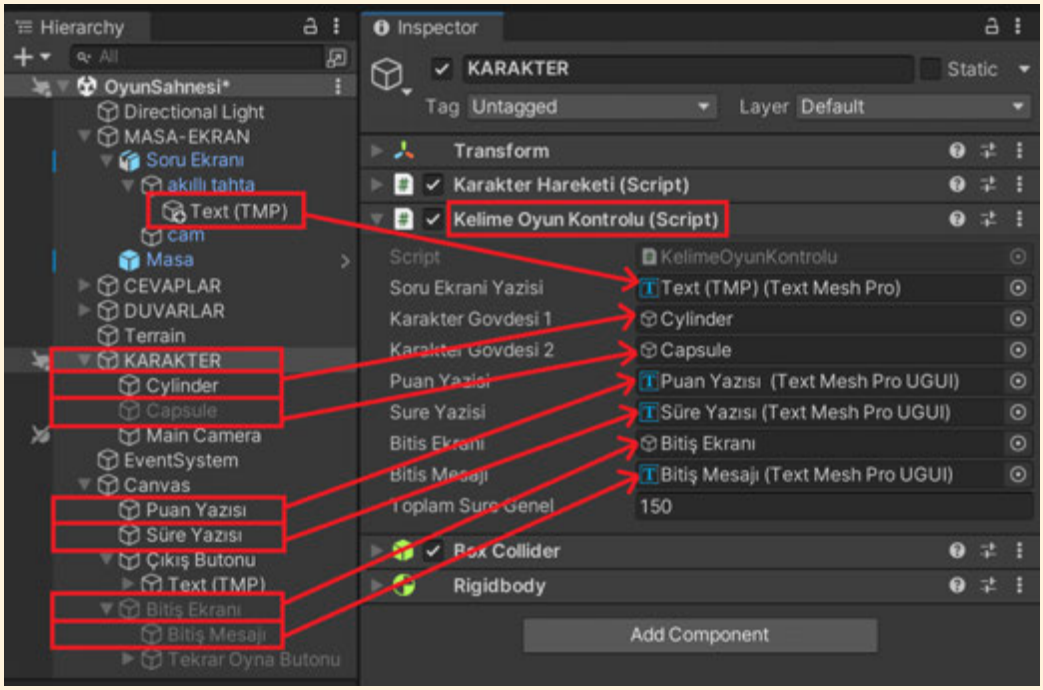
JSON işlemleri için **Newtonsoft.Json** sınıfı kullanılmaktadır. Bu sınıfı kullanabilmek için NewtonSoft Json paketinin oyun motoruna yüklenmesi gerekmektedir. Bu paketi oyun motoruna yüklemek için **https://cdnmeslek.eba.gov.tr/bilisim/oyun/jsn.zip** adresinden sıkıştırılmış paket dosyalarını KelimeMacerasi proje klasörünün içine indiriniz. İndirilen sıkıştırılmış paket dosyalarını bir klasöre çıkarınız. Oyun motorunda **Package Manager** penceresinde sol üstte bulunan **"+"** düğmesine tıklayıp açılan listede **Add package from disk...** seçeneğine tıklayınız. Açılan pencerede, indirdiğiniz paket dosyalarından **package.json** dosyasını bulup seçiniz ve Aç düğmesine tıklayınız.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Newtonsoft.Json; // JSON işlemleri için gerekli sınıftır.
using TMPro;
public class KelimeOyunKontrolu : MonoBehaviour
{
  // Inspector panelinden nesne atamaları yapılacak olan public
  // erişimli değişkenlerdir.
  public TextMeshPro soruEkraniYazisi;
  public GameObject karakterGovdesi1,karakterGovdesi2;
  public TextMeshProUGUI puanYazisi, sureYazisi;
  public GameObject bitisEkrani;
  public TextMeshProUGUI bitisMesaji;
  public int toplamSureGenel = 150; // Oyunun süresi.
  private int toplamSureAzalan; // Kontrol edilecek süre sayacı.
  private int puanDegiskeni = 0; // Oyuncunun puanı.
  private int soruSayisi = 0;  // Kontrol edilecek soru sayısı sayacı.
  private int saklananZorlukSecenegi = 0; // Kullanıcının seçtiği
                                          // zorluk verisi.
  private int saklananKarakterSecenegi=0;// Kullanıcının seçtiği
                                         // karakter verisi.
  private int sorulacakSoruSayisi; // Seçilen zorluğa göre soru
                                   // sayısını tutar.
  private List<Soru> sorularListesi; // kelimeler.json'dan okunan
                                     // verileri tutar.
  private bool oyuncuCevabi; // Kullanıcının değdiği küreye göre
                             // cevabı tutar.
  private bool aktifSoruCevabi; // Ekranda aktif olan sorunun
                                // cevabını tutar.
  private bool soruCevaplandiMi = true; // Bir sonraki soruyu göstermek
    // için kullanılır. Değeri true ise yeni soru ekranda gösterilir.
  [System.Serializable]
  public class Soru  // kelimeler.json'daki verileri temsil eden
                     // sınıftır.
  {
    public string Kelime;
    public bool DogruMu;
  }
  void Start()
```

```
{
    // AnaSahnede seçilen zorluğa göre sorulacak soru sayısı
    // belirlenir.
    saklananZorlukSecenegi = PlayerPrefs.GetInt("zorlukSecenegi",
    0);
    if (saklananZorlukSecenegi == 0)
        sorulacakSoruSayisi = 5;
    else if (saklananZorlukSecenegi == 1)
        sorulacakSoruSayisi = 10;
    // AnaSahnede seçilen karaktere göre Silindir veya Kapsül görünür
    // yapılır.
    saklananKarakterSecenegi = PlayerPrefs.
    GetInt("karakterSecenegi", 0);
        if (saklananKarakterSecenegi == 0)
        {
            karakterGovdesi1.SetActive(true);
            karakterGovdesi2.SetActive(false);
        }
        else if (saklananKarakterSecenegi == 1)
        {
            karakterGovdesi1.SetActive(false);
            karakterGovdesi2.SetActive(true);
        }
    // kelimeler.json dosyasındaki verileri okuyan metot çağrılır.
    // Bu dosyadaki bütün kelimeler ve yazım bilgileri sorularListesi
    // listesine atanır.
    SorulariJSONdanOku();
    // SoruUret metodu çağrılarak ilk soru ekrana yazılır.
    if (soruCevaplandiMi)
        SoruUret();
}
// kelimeler.json dosyasındaki verileri sorularListesi listesine
// aktaran metottur.
public void SorulariJSONdanOku()
{
  string jsonText = Resources.Load<TextAsset>("kelimeler").text;
  if (!string.IsNullOrEmpty(jsonText))
  {
  sorularListesi = JsonConvert.
  DeserializeObject<List<Soru>>(jsonText);
  }
}
// sorularListesi listesindeki sorulardan rastgele 1 tane soru
// seçip, bu soruyu Soru sınıfı türünde döndüren metottur.
public Soru RastgeleSoruOlustur()
{
```

```
        int index = Random.Range(0, sorularListesi.Count);
        Soru secilenSoru = sorularListesi[index];
        sorularListesi.RemoveAt(index);
        return secilenSoru;
    }
    // RastgeleSoruOlustur metodunu kullanarak seçilen sorudaki kelimeyi
    // ekrana yazdıran ve bu kelimenin imlâ bilgisini ilgili değişkene
    // atayan metottur.
    public void SoruUret()
    {
        Soru aktifSoru = RastgeleSoruOlustur();
        string aktifSoruMetni = aktifSoru.Kelime;
        aktifSoruCevabi = aktifSoru.DogruMu;
        soruEkraniYazisi.text = soruSayisi + 1 + " - " + aktifSoruMetni;
        soruCevaplandiMi = false; // Bu değişken false yapılarak
        // kullanıcı soruyu cevaplayana kadar aktif sorunun (kelimenin)
        // ekranda kalması sağlanır. Kullanıcı soruyu cevapladığında
        // bu değişken true yapılarak yeni sorunun ekrana yazılması
        // sağlanır.
    }
}
```

**5. Adım:** KARAKTER isimli oyun nesnesine KelimeOyunKontrolu isimli script dosyasını ekleyiniz ve bu script bileşeninin public değişkenlerine gerekli oyun nesnesi atamalarını yapınız (Görsel 1.117).




**Görsel 1.117:** Değişkenlere nesnelerin atanması

**6. Adım:** AnaSahne'yi açınız ve projeyi çalıştırıp denemeleri yapınız.

SIRA SİZDE

**1.** Resources isimli klasördeki kelimeler.json dosyasına on adet veri seti ekleyiniz. Veri setlerini beş kelimenin doğru ve yanlış yazım bilgilerini içerecek şekilde ekleyiniz.

**2.** kelimeler.json dosyasının ismini sorular.json şeklinde değiştiriniz ve ilgili kodlarda gerekli değişikliği yapınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. kelimeler.json dosyasına 10 adet veri seti ekledi. | | |
| 2. kelimeler.json dosyasının ismini sorular.json şeklinde değiştirdi. | | |
| 3. İlgili kodlarda gerekli değişikliği yaptı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 1.3.6. Zamanlanmış İşlemler

**Coroutine**: Zamanla ilgili işlemleri etkili bir şekilde yönetmek, bir işlemin belirli bir süre çalışmasını beklemek veya bir işlemi belirli aralıklarla tekrarlamak için kullanılan bir yapıdır. Bir Coroutine, **StartCoroutine** fonksiyonu kullanılarak başlatılır. StartCoroutine fonksiyonu ile **IEnumerator** türünden dönüş yapan bir metot çağrılır. Bu metodun içinde **yield return** ifadesi ile **WaitForSeconds** fonksiyonu kullanılarak bekleme süresi saniye cinsinden belirtilir.

**InvokeRepeating:** Bir metodun belirli aralıklarla çalışmasını sağlayan bir fonksiyondur.

```
InvokeRepeating(nameof(MetotİsmI), baslangicBeklemeSuresi, tekrarBeklemeSuresi)
```

- **nameof(MetotİsmI):** Tekrarlanacak olan metodun ismini belirtir. Nameof fonksiyonu, metodun ismini string türünde döndürmeyi sağlar.
- **baslangicBeklemeSuresi:** Metodun ilk defa çalışmaya başlaması için saniye cinsinden geçen süredir.
- **tekrarBeklemeSuresi:** Metodun her çalışması arasında geçecek olan saniye cinsinden süredir.

**Invoke:** Belirli bir metodu belirtilen süre kadar bekledikten sonra çalıştırmak için kullanılan fonksiyondur.

```
Invoke(nameof(MetotİsmI), baslangicBeklemeSuresi)
```

- **nameof(MetotİsmI):** Tekrarlanacak olan metodun ismini belirtir. Nameof fonksiyonu, metodun ismini string türünde döndürmeyi sağlar.
- **baslangicBeklemeSuresi:** Metodun ilk defa çalışmaya başlaması için saniye cinsinden geçen süredir.

## 28. UYGULAMA

İşlem adımlarına göre **KelimeMacerasi** isimli projede oyunun temel işlevlerini sağlayan KelimeOyunKontrolu isimli script dosyasına gerekli kodları ekleyiniz.

**1. Adım: KelimeMacerasi** isimli projedeki **OyunSahnesi** isimli sahneyi açınız.
**2. Adım:** KelimeOyunKontrolu isimli script dosyasına aşağıdaki kodları ekleyiniz:

```
public class KelimeOyunKontrolu : MonoBehaviour

{
   ..... (27.Uygulamada oluşturulan kodlar)

    void Start()
    {
    ..... (27.Uygulamada oluşturulan kodlar)
        toplamSureAzalan = toplamSureGenel; // Süre sayacına toplam
                                            // süre atanır.
        InvokeRepeating(nameof(SureSay), 0f, 1f); // 1sn aralıklarla
        // SureSay metodu çalıştılır.
    }
    ..... (27.Uygulamada oluşturulan kodlar)
// Bu metot sürenin ekranda 1sn azalarak görünmesini sağlar.
private void SureSay()
{
    // Süre sayacı birer birer azaltılır.
    toplamSureAzalan--;
    // Süre sayacının değeri 0 olunca metotun çalışması durdurulur ve
    // bitiş ekranı görünür yapılarak toplam puan bitiş ekranında
    // gösterilir.
    if (toplamSureAzalan <= 0)
    {
      toplamSureAzalan = toplamSureGenel;
      CancelInvoke(nameof(SureSay));
      bitisEkrani.SetActive(true);
      bitisMesaji.text="Süre Bitti, Puanınız: "
                       + (puanDegiskeni + toplamSureAzalan * 1);
    }
    // Süre sayacının değeri ekranda gösterilir.
    sureYazisi.text = "Süre = " + toplamSureAzalan.ToString() + "sn";
}
// Bu metot kullanıcının verdiği cevaba göre işlemler yapar.
private void KulCvbnaGoreIslem(int puan)
{
    // Kendisine gönderilen puan değerini, toplam puana ekleyerek
    // ekranda görünmesini sağlar.
    puanDegiskeni += puan;
    puanYazisi.text = "Puan = " + puanDegiskeni;
    soruSayisi++;
    // Sorulacak soru sayısı bitti ise bitiş ekranını gösterir.
    if (soruSayisi == sorulacakSoruSayisi)
    {
      soruEkraniYazisi.text = puan + " Puan";
      puanYazisi.text = "Puan = "
                        + (puanDegiskeni + toplamSureAzalan * 1);
```

```
        Invoke(nameof(BitisEkrani), 1f);
        bitisMesaji.text = "Tebrikler,Puanınız: " +
                           (puanDegiskeni + toplamSureAzalan * 1);
        toplamSureAzalan = toplamSureGenel;
        CancelInvoke("SureSay");
    }
    // Sorulacak soyu sayısı bitmediyse bir sonraki soruyu üreten
    // metodu çağırır.
    else
    {
        soruEkraniYazisi.text = puan + " Puan";
        // BekleVeSoruUret metodunu başlatır.
        StartCoroutine(BekleVeSoruUret(1f, 1f));
    }
}
// Bu metot belirli süreler bekleme yaptırıp yeni soru üretmeyi sağlar.
IEnumerator BekleVeSoruUret(float saniye1, float saniye)
{
    // saniye1 değeri kadar bekleme yapar.
    yield return new WaitForSeconds(saniye1);
    // Soru ekranında Soru Hazırlanıyor mesajını yazdırır.
    soruEkraniYazisi.text = "Soru Hazırlanıyor";
    // saniye değeri kadar bekleme yapar.
    yield return new WaitForSeconds(saniye);
    // Yeni soru oluşturulup, soru (kelime) soru ekranına yazdırılır.
    SoruUret();
}
// Bu metot ile KULLANICI oyun nesnesinin doğru veya yanlış kürelerine
// çarpıp çarpmadığı kontrol edilir ve buna göre gerekli işlemler
// yapılır.
private void OnCollisionEnter(Collision collision)
{
// Kullanıcı KARAKTER nesnesini doğru küresine temas ettirdi ise ve
// ekranda yeni soru gösterildi ise
    if (collision.gameObject.CompareTag("Dogru") && !soruCevaplandiMi)
    {
    // Ekranda aktif olan kelimenin yazımı da doğru ise +10 puan,
    // yanlış ise -2 puan eklenmesi için KulCvbnaGoreIslem metodu
    // çağrılır.
        oyuncuCevabi = true;
        soruCevaplandiMi = true;
        if (aktifSoruCevabi == oyuncuCevabi)
            KulCvbnaGoreIslem(10);
        else
            KulCvbnaGoreIslem(-2);
    }
    // Kullanıcı KARAKTER nesnesini yanlış küresine temas ettirdi
    // ise ve ekranda yeni soru gösterildi ise
    else if (collision.gameObject.CompareTag("Yanlis") &&
    !soruCevaplandiMi)
    {
    // Ekranda aktif olan kelimenin yazımı da yanlış ise +10 puan,
    // doğru ise -2 puan eklenmesi için KulCvbnaGoreIslem metodu_
    // çağrılır.
```

```
            oyuncuCevabi = false;
            soruCevaplandiMi = true;
            if (aktifSoruCevabi == oyuncuCevabi)
                KulCvbnaGoreIslem(10);
            else
                KulCvbnaGoreIslem(-2);
        }
    }
    // Bu metot bitiş ekranı aktif olduğunda, bu ekrandaki Tekrar Oyna
    // düğmesine tıklayınca oyunun tekrar başlamasını sağlar.
    public void TekrarOyna()
     {
        bitisEkrani.SetActive(false);
        puanDegiskeni = 0;
        sureYazisi.text = "Süre = " + toplamSureAzalan.ToString() + "sn";
        puanYazisi.text = "Puan = 0";
        InvokeRepeating(nameof(SureSay), 0f, 1f);
        toplamSureAzalan = toplamSureGenel;
        soruSayisi = 0;
        if (soruCevaplandiMi)
            SoruUret();
    }
    // Bu metot bitiş ekranını görünür yapmayı sağlar.
    void BitisEkrani()
    {
        soruEkraniYazisi.text = "Oyun Bitti...";
        bitisEkrani.SetActive(true);
    }
}
```

SIRA SİZDE

1. Oyunda Bitiş Ekranı aktif olduğunda, kullanıcı oyunu 10 sn içinde tekrar başlatmaz ise AnaSahneyi aktif yapmayı sağlayan kodları KelimeOyunKontrolu script dosyası içinde Coroutine yapısını kullanarak yazınız.
2. Oyunda Bitiş Ekranı aktif olduğunda, kullanıcı oyunu 10 sn içinde tekrar başlatmaz ise AnaSahneyi aktif yapmayı sağlayan kodları KelimeOyunKontrolu script dosyası içinde Invoke yapısını kullanarak yazınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. 10 sn içinde AnaSahneyi aktif yapmayı sağlayan Coroutine metodunu oluşturdu. | | |
| 2. Coroutine metodunu BitisEkrani metodu içinde kullandı. | | |
| 3. 10 sn içinde AnaSahneyi aktif yapmayı sağlayan ve Invoke yapısı ile kullanılacak metodu oluşturdu. | | |
| 4. İlgili metodu Invoke yapısı ile BitisEkrani metodu içinde kullandı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## ÖLÇME VE DEĞERLENDİRME

**Aşağıdaki soruları dikkatlice okuyarak doğru cevabı işaretleyiniz.**

**1. Oyun motorunda sahne (scene) nedir?**

A) Oyunun grafik tasarımıdır.

B) Oyunun bir bölümüdür.

C) Kodlama arayüzüdür.

D) Oyunun menüsüdür.

E) Oyunun nesneleridir.

**2. Aşağıdakilerden hangisi oyun motorunda oyun nesnelerini hareket ettirmek için kullanılan bileşendir?**

A) Etkileşim B) Movement C) Physics

D) Rigidbody E) Vector

**3. Oyun motorunda UI (Kullanıcı Arayüzü) oluşturmak için aşağıdakilerden hangisi kullanılır?**

A) Canvas B) Collider C) Rigidbody

D) Shader E) Transform

**4. Oyun motorunda hangi bileşen, oyun nesnelerinin diğer nesnelerle çarpışmalarını kontrol eder?**

A) Collider B) Crasher C) Exchange

D) Impactor E) Trigger

**5. Aşağıdakilerden hangisi oyun motorunda modellerin görünüm özelliklerinin tümünü belirlemek ve düzenlemek için kullanılan yapıya verilen isimdir?**

A) Alpha B)Material C) Shader

D) Surface E) Texture

**6. Oyun motorunda hangi terim, oyun nesnelerinin belirli bir bölgeye girmesini algılamak için kullanılır?**

A) Area B) Radar C) Rigidbody

D) Trigger E) Zone

**7. Aşağıdakilerden hangisi oyun motorunda zamanla ilgili işlemleri etkili bir şekilde yönetmek için kullanılan yapıdır?**

A) Coroutine B) Crasher C) Physics

D) Texture E) Vector

**8. Aşağıdakilerden hangisi oyun motorunda detaylı, özelleştirilebilir ve gerçekçi arazi ve çevre oluşturmak için kullanılan bir oyun nesnesidir?**

A) Module B) Plane C) Surface

D) Terrain E) Transform

**9. Bir nesne bir tetikleyiciye girdiği anda çağrılan metot aşağıdakilerden hangisidir?**

A) OnTriggerExit

B) OnTriggerStay

C) OnTriggerEnter

D) OnCollisionEnter

E) OnCollisionStay

**10. Aşağıdaki Rigidbody bileşenlerinden hangisi etkinleştirildiğinde oyun nesnesi fizik kurallarından etkilenmez?**

A) Automic Tensor

B) Constraints

C) Drag

D) Interpolate

E) Is Kinematic

# 2. ÖĞRENME BİRİMİ



# Karakter ve Çevre Düzenleme

## KONULAR

2.1. SENARYO OLUŞTURMA
2.2. RESİM DÜZENLEME
2.3. ÜÇ BOYUTLU MODELLEME
2.4. OYUN PROGRAMINA AKTARMA

## NELER ÖĞRENECEKSİNİZ?

- Oyun senaryosu için bir plan taslağı hazırlama
- Resim türleri dönüşümlerini yapma
- Hareketli resimler tasarlama
- Saydam arka plan hazırlama
- Doku hazırlama, kaplama ve boyama işlemlerini yapma
- Katı modellemede düşük çokgenler ile nesneler yapma
- Araç, doğa, insan gibi çeşitli nesnelerin modellemesini ve renklendirmesini yapma
- Oluşturulan resim ve nesnelerin düzenlemesini yapma
- Proje içine resim ve nesne ekleme
- Tekrar kullanılabilir nesne oluşturma

## ANAHTAR KELİMELER

oyun senaryosu, resim türleri, hareketli resim, doku hazırlama, düşük poligon, tekrar kullanılabilir nesne

HAZIRLIK ÇALIŞMALARI

1. Oyun geliştirme sürecinin genel aşamaları nelerdir? Araştırınız.
2. Üç boyutlu modellemede kullanılan programlar veya araçlar hakkında neler biliyorsunuz? Arkadaşlarınızla paylaşınız.

## 2.1. SENARYO OLUŞTURMA

Oyun geliştirme süreci, birden fazla katmanı ve farklı disiplinleri içeren karmaşık bir yapıya sahiptir. Bu yapı, bir bütün olarak ele alınırken farklı unsurlar ayrı ayrı göz önünde bulundurulur ve uygulanır. Oyunun oluşturulması aşamasında senaryo, oyun motoru, karakter ve mekân tasarımı, arayüz, ses, prototipler ve test aşamaları yer almaktadır.

**Oyunun hikâyesi,** genel bir tema etrafında adım adım ilerleyen bir hikâye biçiminde belirlenir. **Oyun motorları** ise oyunun gerçekçi işlemesi için kodlanmış, tekrar kullanılabilir bileşenler koleksiyonudur. Render, yükleme, animasyon, çarpışma algılama, fizik, girişler, grafik arayüzü ve yapay zekâ, oyun motorunun bileşenlerini oluşturur.

**İçerik tasarımı;** oyunun karakterleri, varlıkları, mekânları, malzemeleri ve diğer tüm nesnelerin planlanma sürecidir. **Ses tasarımı** ise oyun içindeki müzikten nesne seslerine kadar tüm ses içeriğinin düzenlenmesini içerir.

Oyunlar, prototip ve test aşamasında oyuncular tarafından deneyimlenerek eksiklikler ve hatalar belirlenir. Bu aşamada gerekli düzeltmeler ve güncellemeler yapılarak oyunun son hâli oluşturulur.

### 2.1.1. Oyun Senaryosu İçin Plan Taslağı Hazırlama

**Oyun senaryosunun taslağı;** oyunun hikâye, oynanış ögeleri, karakterler, oyun dünyası ve diğer temel unsurlarını tanımlayan planın kısa bir özetidir. Oyun senaryosunun taslağı; oyunun hikâyesinin genel hatlarını, karakterlerin temel özelliklerini ve oyun içi olayların gelişimini içerir. Genellikle oyunun başından sonuna kadar olan genel olay örgüsünü belirtir ve oyunun temel mekaniği, oyun içi etkileşimler veya engeller gibi oynanış unsurlarını açıklar. Ayrıca karakterlerin arka plan hikâyelerini, hedeflerini ve oyun dünyasının temel özelliklerini tanımlayabilir.

Oyun senaryosunun taslağı oyunun genel yapısını görselleştirmeye yardımcı olur. Senaryo taslağı, oyun geliştirme sürecinde yol gösterici olarak kullanılır ve oyunun ilerlemesinde rehberlik eder. Aynı zamanda paydaşlara sunularak oyunun potansiyelini ve yapısını göstermek için kullanılabilir. Bu taslak, genellikle gelişim aşamaları boyunca güncellenir ve yenilenir. Tablo 2.1'de genel bir oyun senaryosu taslağı görülmektedir.

**Tablo 2.1:** Örnek Oyun Senaryosu

| | |
|---|---|
| **Oyun Adı:** | Matematik Macerası |
| **Oyun Türü:** | Eğitici, Bulmaca |
| **Oyun İçeriği:** | Oyun, oyuncuların temel matematik becerilerini geliştirmelerini sağlamak için eğlenceli ve interaktif bir ortam sunar. Oyuncular, harf bulmacaları ve sayısal işlemlerle ilgili görevleri tamamlayarak matematiksel becerilerini geliştirirler. |
| **Oyun Hikâyesi:** | Ana karakter, bir bilim kahramanı veya maceracı olarak tanımlanabilir. Bilgiyle dolu bir kitap dünyasına yolculuk yapar. Kitap dünyası, matematiksel sorular ve bulmacalarla doludur. Karakter, kayıp harfleri bulmak ve sayısal işlemleri çözmek için maceraya atılır. |
| **Ana Karakter:** | Adı: Mateminik<br>Özellikleri: Zeki, meraklı ve matematik sever. |

| | |
|---|---|
| **Oyun Mekaniği:** | • Oyuncu, farklı kitap sayfalarına seyahat eder ve her sayfada farklı matematiksel bulmacalarla karşılaşır.<br>• Harf bulmacaları, basit denklem çözme görevleri, sayısal dizileri tamamlama gibi farklı matematiksel etkinlikler sunulur.<br>• Başarılı tamamlanmış görevler, oyuncunun ilerlemesini ve yeni seviyelere geçmesini sağlar. |
| **Diyalog ve Etkileşim:** | • Oyuncular, kitap karakterleri ve rehberlerle etkileşime geçebilir. Görevleri anlamak ve ipuçları almak için diyalog seçenekleri sunulabilir.<br>• Oyun, oyuncuların kendi seçimlerine göre zorluk seviyelerini ayarlayabilir. |
| **Oyun Bitişi:** | • Oyuncu, kitap dünyasını keşfederken matematiksel becerilerini geliştirir ve harf bulmacalarını tamamlayarak sona erer.<br>• Her tamamlanan bulmaca, oyuncunun puanlarını artırır ve oyuncuya yeni görevlerin kilidini açar. |

## 1. UYGULAMA

İşlem adımlarına göre önceki derslerde oluşturduğunuz "KelimeMacerasi" isimli projeniz için bir oyun senaryosu taslağı oluşturunuz.

**1. Adım:** Oyunun türünü (Bilim kurgu, fantastik, macera vb.) yazınız.
**2. Adım:** Oyunun teması hakkında bilgi veriniz.
**3. Adım:** Oyunun hikâyesini yazınız.
**4. Adım:** Ana/Yardımcı karakterler hakkında bilgi veriniz.
**5. Adım:** Oyuncuların oyun dünyasında etkileşime geçebileceği nesneler, beceriler ve oyun kuralları hakkında bilgi veriniz.
**6. Adım:** Oyuncularla karakterler arasındaki etkileşimleri ve diyalogları belirtiniz.

## SIRA SİZDE

Mevcut oyun türleri hakkında araştırma yaparak kendi oyun senaryonuz için bir plan taslağı hazırlayınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Oyunun türünü belirledi. | | |
| 2. Oyunun içeriğini tanımladı. | | |
| 3. Oyunun içeriğine uygun bir oyun adı seçti. | | |
| 4. Ana/Yardımcı karakterleri özellikleri ile tanımladı. | | |
| 5. Oyunun mekaniğini belirledi. | | |
| 6. Diyalog ve etkileşimleri belirledi. | | |
| 7. Oyunun bitişini belirledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

### 2.1.2. Oyun Senaryosunda Kullanılan Görsel-İşitsel İçerikler

Oyun geliştirme sürecinde kullanılan **görsel ve işitsel içerikler,** oyun deneyimini zenginleştiren ve oyuncuları oyun dünyasının içine çeken temel ögelerdir.

Bu unsurlar, oyunun estetik yapısını ve duygusal boyutunu bir arada şekillendirir. Görsel unsurlar; karakter tasarımları, arka planlar, efektler ve nesneler aracılığıyla oyuncuların görsel algısını yönlendirirken işitsel unsurlar ise kullanılan müzikler, ses efektleri ve karakter seslendirmeleriyle duygusal bir derinlik kazandırır.

Oyunun genel kalitesini ve etkileyiciliğini belirlemede kilit rol oynayan görsel ve işitsel içerikler, oyunlara uygun bir şekilde entegre edildiklerinde, oyuncuların oyun deneyimi daha etkileyici ve unutulmaz hâle gelir. Görsel ve işitsel içeriklerin dengeli bir biçimde kullanılması, oyunun bütünselliğini güçlendirir ve oyuncuların deneyimini derinleştirir.

#### 2. UYGULAMA

İnternet üzerinde araştırma yaparak bulmaca türünde bir oyun belirleyiniz. Seçtiğiniz oyunun incelemesini aşağıdaki işlem adımlarını dikkate alarak yapınız.

**1. Adım:** Oyunu tanımlayınız.
**2. Adım:** Oyunun amacını yazınız.
**3. Adım:** Oyunun hikâyesini yazınız.
**4. Adım:** Oyunun ana/yardımcı karakterlerini belirtiniz.
**5. Adım:** Oyunda kullanılan grafikleri ve sanatsal ögeleri belirtiniz.
**6. Adım:** Oyunda kullanılan ses ve müzikler hakkında bilgi veriniz.

#### SIRA SİZDE

**1.** İnternet üzerinde araştırma yaparak oyunlarda kullanılan görsel ve işitsel içeriklerin, oyunun türü veya oyunun hikâyesi gibi unsurlar ile ilişkisini inceleyiniz.

**2.** Bir oyun türü belirleyerek oyun senaryonuz için bir plan taslağı hazırlayınız. Plan taslağınızda belirttiğiniz tanımlara ve özelliklere uygun olan, oyununuzu daha etkileyici hâle getirecek görsel ve işitsel içerikleri belirleyiniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Oyunun türünü belirledi. | | |
| 2. Oyunun içeriğini tanımladı. | | |
| 3. Oyunun içeriğine uygun bir oyun adı seçti. | | |
| 4. Ana/Yardımcı karakterleri özellikleri ile tanımladı. | | |
| 5. Oyunun mekaniğini belirledi. | | |
| 6. Diyalog ve etkileşimleri belirledi. | | |
| 7. Oyunun bitişini belirledi. | | |
| 8. Oyuna uygun görsel ve işitsel içerik belirledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 2.2. RESİM DÜZENLEME

Oyunun görsel kalitesini ve iyi bir kullanıcı deneyimini sağlamak resim düzenleme becerilerini gerektirir. Resim düzenleme; oyun geliştirme sürecindeki kullanıcı arayüzü tasarımında, karakterlerin ve diğer nesnelerin görünümünü oluşturmada önemli bir rol oynar.

**Resim düzenleme programı,** dünya genelinde yaygın olarak kullanılan profesyonel bir görüntü düzenleme ve grafik tasarım yazılımıdır. Profesyonel fotoğrafçılık, grafikerlik, web tasarım, oyun geliştirme ve görsel medya uzmanlığı başta olmak üzere birçok sektörde kullanılan resim düzenleme programı, geniş özellik yelpazesi ve kullanım kolaylığı ile öne çıkmaktadır.

Hem iki boyutlu oyunlarda doğrudan hem de üç boyutlu oyunlarda modellerin dokularında kullanılan resimler ve bu resimlerin oyun motorunda işlenmeleri oyun geliştirme sürecinin önemli bir parçasıdır. Resim düzenleme programı oyun geliştirme sürecinde şu amaçlar için kullanılır:

**Görsel İçerik Oluşturma:** Resim düzenleme programı, oyun motorunda kullanılmak üzere grafikler, arka planlar ve diğer görsel ögelerin oluşturulmasında yaygın olarak kullanılır. Oyun karakterleri, arayüz ögeleri, efektler ve diğer görseller resim düzenleme programında tasarlandıktan sonra oyun motoruna aktarılır.

**Doku ve Materyal Oluşturma:** Resim düzenleme programı, oyun motorunda kullanılmak üzere doku oluşturmak için sıklıkla tercih edilir. Özellikle üç boyutlu modellerin materyallerinin oluşturulması için resim düzenleme programında dokular hazırlanır ve bu dokular oyun motorunda modeller üzerine uygulanarak materyal oluşturulur.

**Kullanıcı Arayüzü Tasarımı:** Oyun motorunda, oyunların kullanıcı arayüzleri (butonlar, menüler, metinler vb.) genellikle resim düzenleme programında tasarlanır ve ardından oyun motoruna aktarılır.

**Post Prodüksiyon ve Efektler:** Oyunun grafik kalitesini artırmak veya belirli efektleri oyunda oluşturmak için resim düzenleme programı kullanılabilir. Örneğin efektler, renk düzenlemeleri veya özel efektlerin hazırlanması gibi işlemler resim düzenleme programında gerçekleştirilir ve ardından oyun motorunda oyunun parçası hâline getirilir.

### 2.2.1. Resim Türleri Dönüşümleri

Resim türü, bir resim dosyasının formatını veya dosya uzantısını ifade eder. Her bir resim türü farklı görsel ihtiyaçları karşılar ve farklı özelliklere sahiptir. Resim türlerinin başlıcaları: PNG, JPEG (JPG), GIF, BMP, TIFF, PSD, TGA'dır. Bu formatlardan **PNG**, **JPEG (JPG)** ve **GIF** oyun programında en yaygın kullanılan resim türlerindendir.

**PNG:** Bu resim türü, resimlerde saydamlık desteği sunar ve kaliteli görüntüler için kullanılır. Oyun geliştirmede PNG; sprite, UI (kullanıcı arayüzü) ögeleri, simgeler ve efektler için tercih edilmektedir.

**JPEG:** Yüksek kaliteye sahip görüntüleri sıkıştırmak için kullanılır. Resimlerde daha fazla sıkıştırma sağlar ancak dosya boyutları genellikle büyük olabilir.

**GIF:** Oyun geliştirmede basit animasyonlu sprite (2D grafik nesnesi), simgeler ve düşük renkli görüntüler için kullanılır.

**TIFF:** Yüksek kaliteli görüntüler için kullanılan bir formattır. Kayıpsız sıkıştırma ve renk profili desteği sağlar ancak dosya boyutları genellikle büyük olabilir.

Bu resim türleri, oyun motoru içerisinde yaygın olarak kullanılan resim formatlarıdır. Proje gereksinimlerine ve kullanılacak içeriğin türüne bağlı olarak farklı formatlar tercih edilebilir. Örneğin şeffaflık gerektiren durumlarda PNG, düşük boyutlu resimler için JPEG sıklıkla tercih edilir.

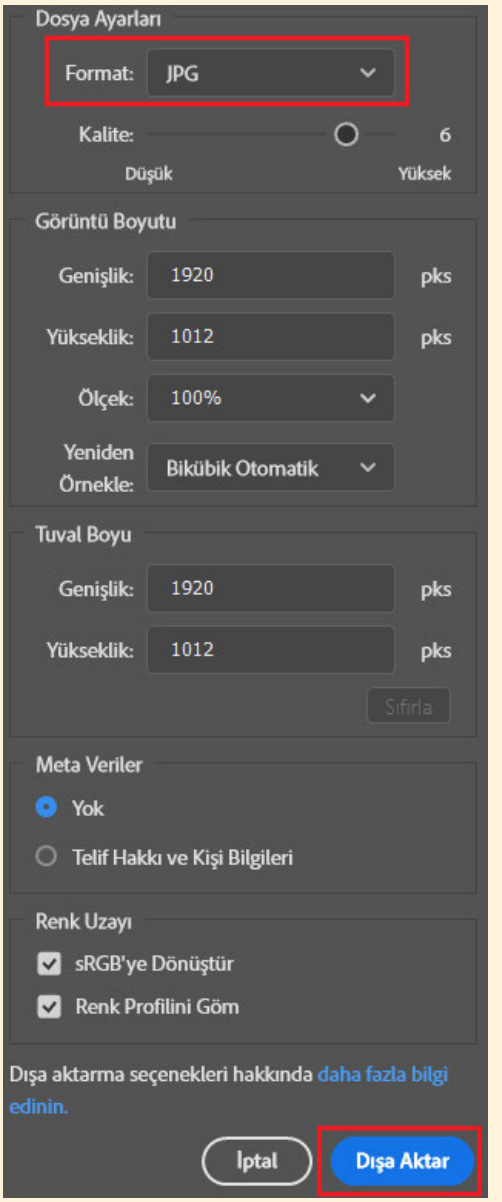
Resim düzenleme programında oluşturulan proje dosyaları, oyun motorunda kullanılabilir formatlara (PNG, JPEG vb.) dönüştürülerek oyun geliştirme sürecine dâhil edilir.

Resim düzenleme programında üzerinde çalışılan bir resim, **Dosya** menüsündeki **Farklı Kaydet** seçeneğiyle farklı resim formatlarında kaydedilebilir ve yine **Dosya** menüsündeki **Dışa Aktar**> **Farklı Dışa Aktar** seçeneğiyle farklı resim türlerinde, farklı boyutlarda ve kalitede dışa aktarılabilir.

## 3. UYGULAMA

Aşağıdaki işlem adımlarına göre resim tür dönüşümlerini gerçekleştiriniz.

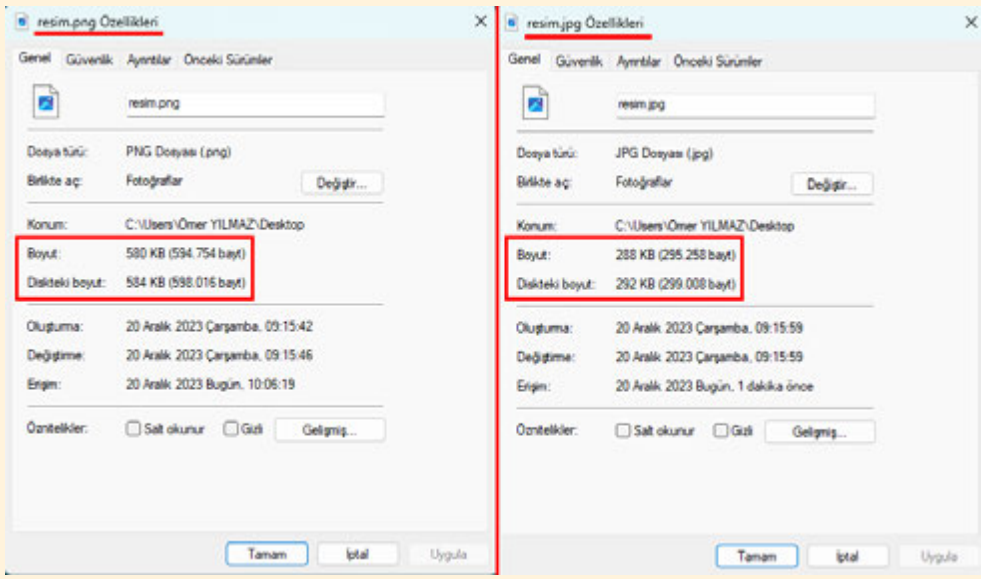
**1. Adım:** Resim düzenleme programı ile bilgisayarınızdan herhangi bir resim dosyası açınız.
**2. Adım: Dosya** menüsündeki **Farklı Kaydet** komutunu seçiniz.
**3. Adım:** Açılan çalışmanızı kaydedeceğiniz konumu belirleyiniz. Dosya adını veriniz ve kayıt türü olarak **PNG** formatını seçerek **Kaydet** düğmesine tıklayınız.
**4. Adım: Dosya**> **Dışa Aktar**> **Farklı Dışa Aktar** yolunu takip ederek **Farklı Dışa Aktar** penceresini açınız.
**5. Adım: Dosya Ayarları** bölümünden **JPG** türünü seçiniz ve **Dışa Aktar** düğmesine tıklayınız (Görsel 2.1).




**Görsel 2.1:** Dosyayı dışa aktarma

**6. Adım:** Açılan pencereden çalışmanızı kaydedeceğiniz konumu seçiniz. Dosya adını yazınız ve **Kaydet** düğmesine tıklayınız.

**7. Adım: JPG** ve **PNG** türlerinde kaydettiğiniz resimlerin özelliklerinden boyutlarına bakınız ve aradaki farkı gözlemleyiniz (Görsel 2.2).




**Görsel 2.2:** PNG ve JPG dosya türleri

SIRA SİZDE

Bilgisayarınızda bulunan herhangi bir **JPG** veya **PNG** türündeki resmi, resim düzenleme programında açarak aynı isimde ve aynı konumda **GIF** dosya türüne dönüştürünüz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Resim düzenleme programı ile kayıtlı resmi açtı. | | |
| 2. Farklı Kaydet/Farklı Dışa Aktar penceresini açtı. | | |
| 3. GIF dosya türünü seçti. | | |
| 4. Dosyaya isim verdi. | | |
| 5. Dosyanın oluşturulacağı dizini seçti. | | |
| 6. Kaydet düğmesine basarak tür dönüşümünü gerçekleştirdi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

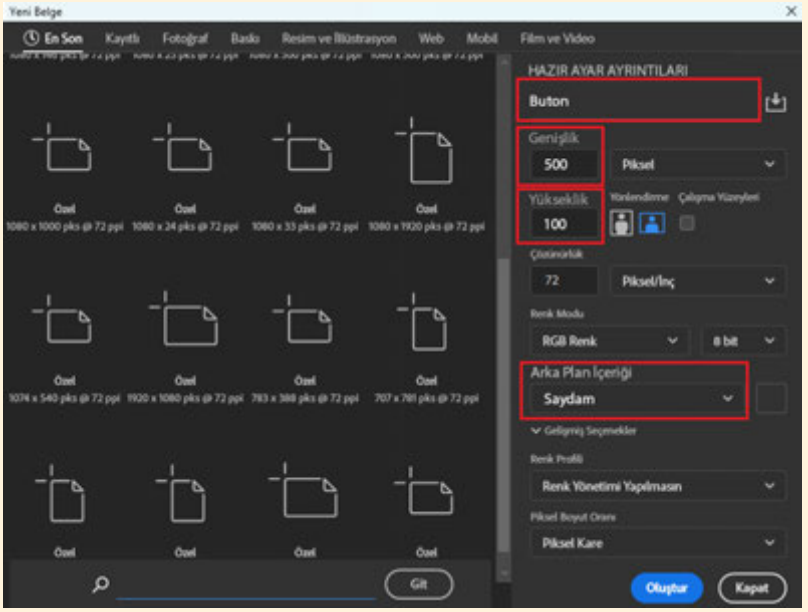
## 2.2.2. Saydam Arka Planlı Resim Oluşturma

Oyun geliştirme sürecinde sahnedeki nesnelerin, karakterlerin ve kullanıcı arayüzü nesnelerindeki buton, text ve image gibi ögelerin gerçekçi ve görsel olarak kaliteli görünmeleri için **saydam arka planlı resimler** yaygın olarak kullanılmaktadır. Resim düzenleme programında saydam arka planlı resimler oluşturulabilir. Bu işlem, üzerinde çalışılan projenin **PNG** resim türünde dışa aktarılmasıyla gerçekleştirilir.

## 4. UYGULAMA

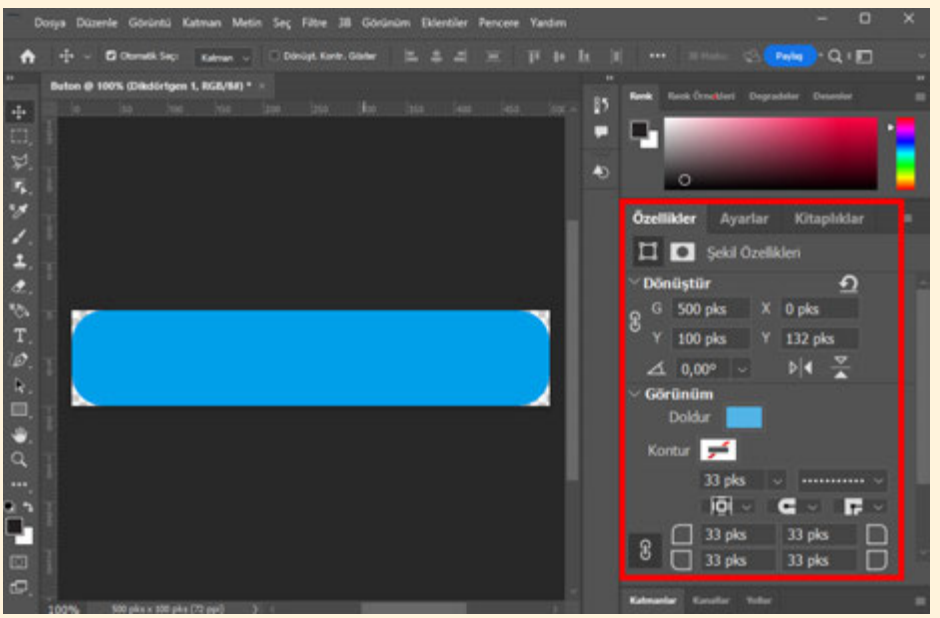
Aşağıdaki iş adımlarına göre saydam arka planlı bir resim oluşturunuz.

**1. Adım:** Resim düzenleme programında 500 piksel genişliğinde, 100 piksel yüksekliğinde, ismi "Buton" olan yeni bir belge oluşturunuz (Görsel 2.3).




**Görsel 2.3:** Resim düzenleme programı yeni belge oluşturma

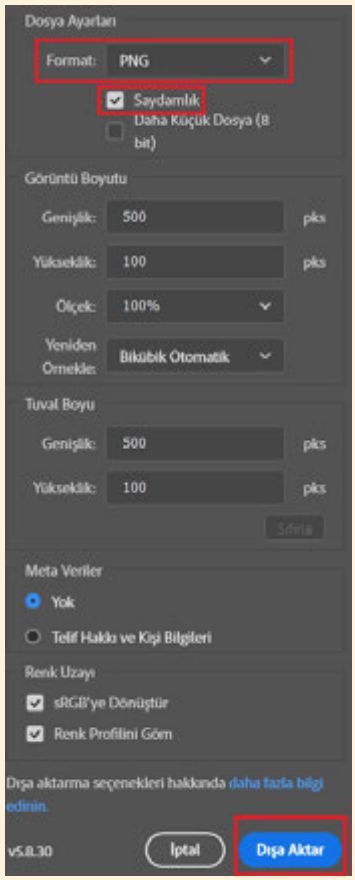
**2. Adım**: **Araçlar** panelindeki **dikdörtgen** aracını seçip farenin sol tuşuna basılı tutarak çalışma alanına bir dikdörtgen çiziniz. Çizdiğiniz dikdörtgeni seçip özellikler panelinden dikdörtgenin **genişliğini** 500 piksel, **yüksekliğini** 100 piksel, **dolgu rengini** mavi, **konturunu** (çerçeve rengini) boş, **köşe kavislerini** de 33 piksel olarak ayarlayınız (Görsel 2.4).




**Görsel 2.4:** Resim düzenleme programı özellikler paneli

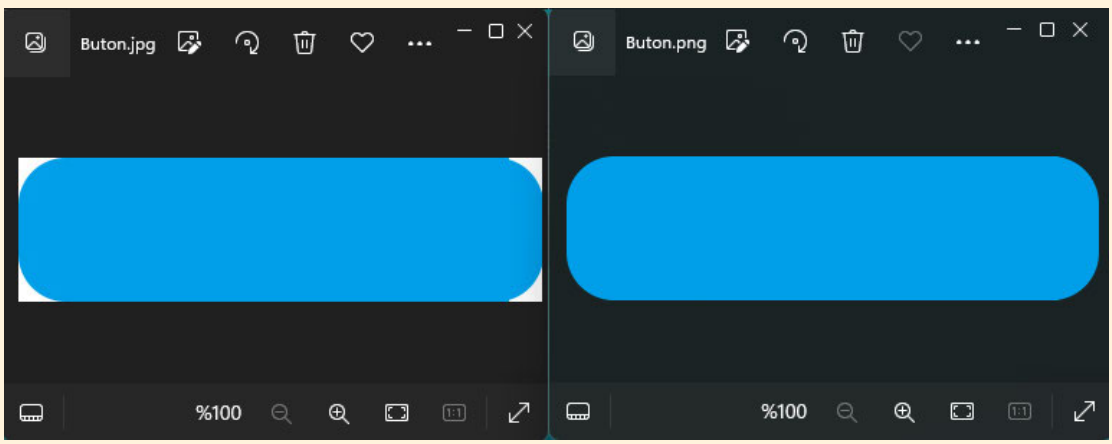
**3. Adım: Dosya**> **Dışa Aktar**> **Farklı Dışa Aktar** yolunu takip ederek **Farklı Dışa Aktar** penceresini açınız.

**4. Adım: Dosya Ayarları** bölümünden **PNG** türünü seçiniz ve **Saydamlık** seçeneğini aktif ederek **Dışa Aktar** düğmesine tıklayınız (Görsel 2.5).




**Görsel 2.5:** PNG türünde kaydetme

**5. Adım:** Açılan pencereden çalışmanızı kaydedeceğiniz konumu seçiniz. Dosya adını yazınız ve **Kaydet** düğmesine tıklayınız.
**6. Adım:** Projenizi üçüncü, dördüncü ve beşinci adımlardaki işlemleri tekrarlayarak **JPG** dosya türünde dışa aktarınız.
**7. Adım: JPG** ve **PNG** türlerinde kaydettiğiniz resimleri inceleyiniz ve iki resim arasındaki farkı gözlemleyiniz (Görsel 2.6).




**Görsel 2.6:** JPG türündeki ve PNG türündeki görüntüler

SIRA SİZDE

Resim düzenleme programında 1024 piksel genişliğinde ve 200 piksel yüksekliğinde yeni bir belge oluşturunuz. Çalışma alanınıza **metin** aracını kullanarak adınızı ve soyadınızı yazınız. Çalışmanızı **saydam** olarak kaydediniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Resim düzenleme programı ile belirtilen ölçülerde yeni belge oluşturdu. | | |
| 2. Metin aracını kullanarak adını ve soyadını yazdı. | | |
| 3. Farklı Dışa Aktar penceresini açtı. | | |
| 4. PNG dosya türünü seçti. | | |
| 5. Saydamlık seçeneğini aktif etti. | | |
| 6. Kaydet düğmesine basarak çalışmayı kaydetti. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 2.2.3. Hareketli Resimler Oluşturma

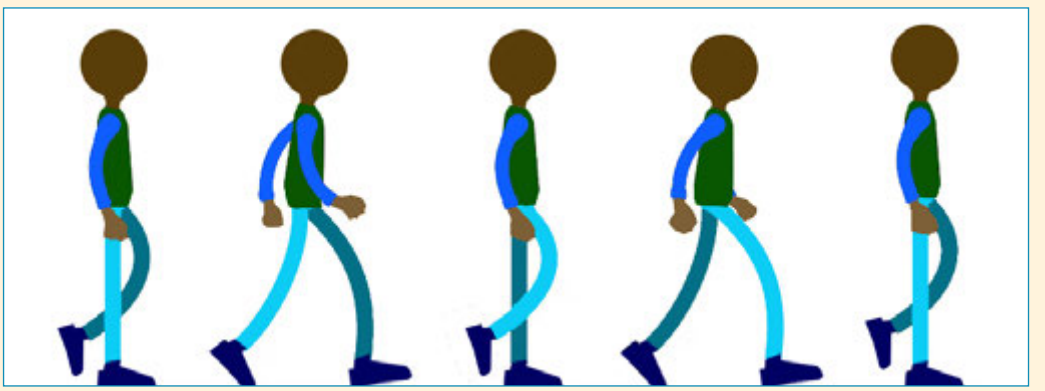
**Hareketli resimler,** birden fazla görüntünün tek bir dosyada sıkıştırılmasıyla oluşturulan bir grafik formatıdır. Hareketli bir resim dosyası, ardışık karelerde farklı görüntüler içerir ve bu kareler hızlı bir şekilde oynatıldığında hareketli bir izlenim oluşturur. Her bir kare, bir önceki ve bir sonraki kare arasında değişiklik gösterir ve bu değişimler genellikle hızlı bir şekilde gerçekleşir.

Hareketli resimler; genellikle kısa ve etkileyici animasyonlar, duygusal ifadeler, eğlenceli içerikler veya basit hareketli resimler gibi çeşitli amaçlar için kullanılır. Hareketli resim formatı **GIF**'tir.

5. UYGULAMA

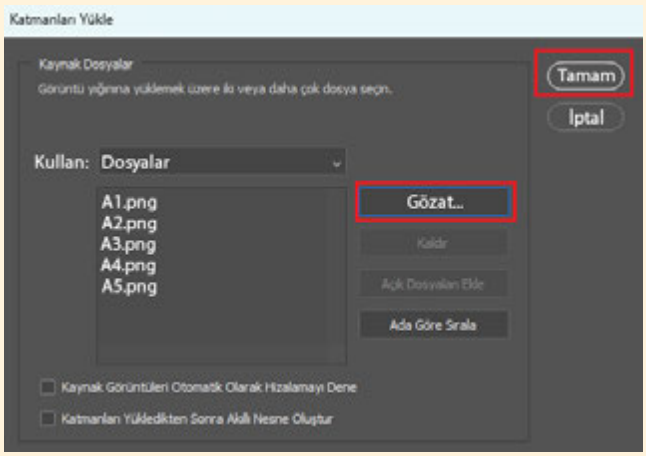
Aşağıdaki işlem adımlarına göre hareketli bir resim oluşturunuz.

**1. Adım:** Resim düzenleme programında Görsel 2.7'dekine benzer yürüyüş döngüsü resmi oluşturunuz veya telifsiz bir görüntüyü resim düzenleme programı ile açınız.



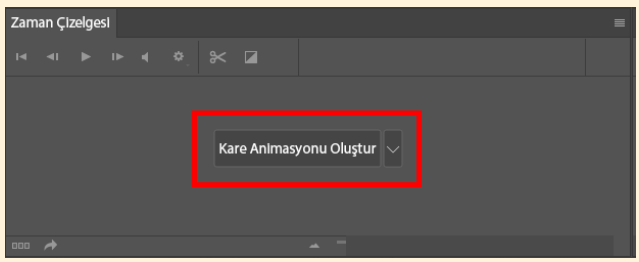
**Görsel 2.7:** Yürüyüş döngüsü

**2. Adım:** Her bir hareketin görüntüsünü ayrı ayrı A1.png, A2.png, A3.png, A4.png, A5.png isimlerinde kaydediniz.
**3. Adım:** Resim düzenleme programında **Dosya**> **Komut Dosyaları**> **Dosyaları Yığına Yükle** komutuna tıklayınız.
**4. Adım:** Açılan pencereden **Gözat** düğmesine tıklayarak, daha önce kaydetmiş olduğunuz resimlerin tümünü seçip **Tamam** düğmesine basınız (Görsel 2.8).



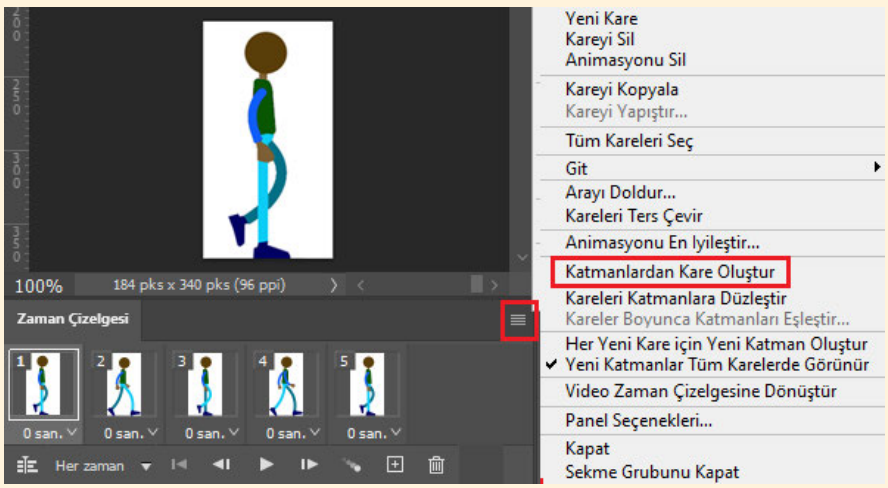

**Görsel 2.8:** Katmanları Yükle penceresi

**5. Adım:** Resim düzenleme programında **Zaman Çizelgesi** penceresinin açık olduğundan emin olunuz. Pencere açık değilse **Pencere**> **Zaman Çizelgesi** komutu ile pencereyi açabilirsiniz.
**6. Adım: Zaman Çizelgesi** üzerinde **Kare Animasyonu Oluştur** düğmesine tıklayınız (Görsel 2.9).



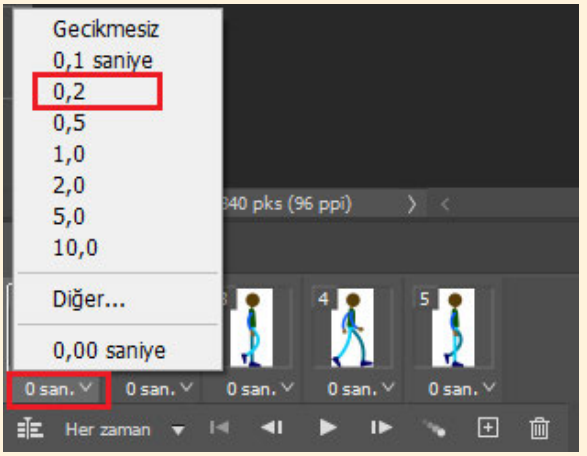

**Görsel 2.9:** Zaman Çizelgesi

**7. Adım: Zaman Çizelgesi**'nin sağ üst köşesinde yer alan açılır menüye tıklayarak **Katmanlardan Kare Oluştur** komutuna tıklayınız (Görsel 2.10).



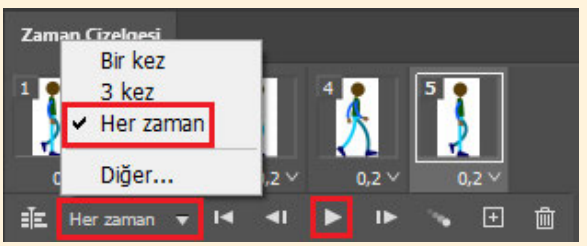

**Görsel 2.10:** Katmanlardan kare oluşturma

**8. Adım:** Görsel 2.11'de görüldüğü gibi her bir karenin gösterilme zamanını 0,2 saniye olarak ayarlayınız.



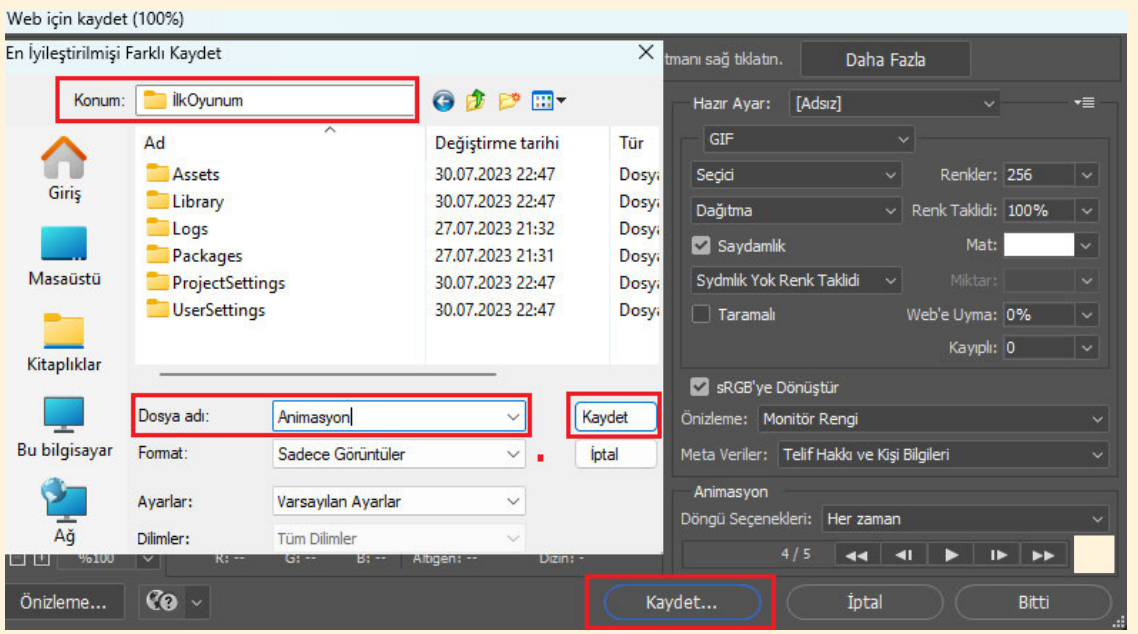

**Görsel 2.11:** Kare gösterim süresi

**9. Adım:** Karelerin kaç kez oynatılacağını Görsel 2.12'de görüldüğü gibi **Her zaman** olarak seçiniz ve **oynatma** düğmesine tıklayınız. Animasyonu ön izleme yaparak gözden geçiriniz.




**Görsel 2.12:** Animasyon döngü değeri

**10. Adım: Dosya**> **Dışa Aktar**> **Web için kaydet** komutuna tıklayınız.
**11. Adım:** Açılan pencereden **Kaydet** düğmesine tıklayınız. Resminizi kaydedeceğiniz konumu seçerek çalışmanıza bir isim veriniz ve **Kaydet** düğmesine tıklayınız (Görsel 2.13).




**Görsel 2.13:** Web için kaydet penceresi

SIRA SİZDE

Resim düzenleme programında zıplayan bir top hareketi içeren bir resim oluşturunuz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Resim düzenleme programında yeni belge oluşturdu. | | |
| 2. Top görüntüsü oluşturdu/Hazır görüntüyü programa aktardı. | | |
| 3. Zaman çizelgesinde kare animasyonu oluşturdu. | | |
| 4. Birinci kare için topu çalışma alanında konumlandırdı. | | |
| 5. Yeni kareler oluşturarak her bir kare için topu konumlandırdı. | | |
| 6. Projeyi hareketli resim olarak kaydetti. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

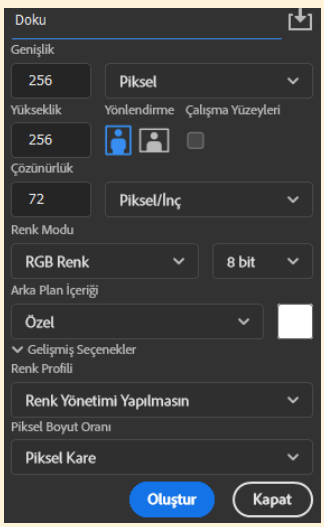
## 2.2.4. Doku Hazırlama

Oyun motorunda **dokular,** malzeme özelliklerini belirleyen önemli unsurlardır. Dokular, bir nesnenin yüzeyinin görünümünü belirleyen resim dosyalarıdır. Dokuları, dış kaynaklardan indirebilir veya resim düzenleme programıyla kendiniz oluşturabilirsiniz. Resim düzenleme programı, dokuları oluşturmak ve düzenlemek için güçlü bir araçtır. Resim düzenleme programında hazırlanan doku, oyun motorunda veya diğer 3B grafik uygulamalarında kullanılmak üzere dışa aktarılabilir.

6. UYGULAMA

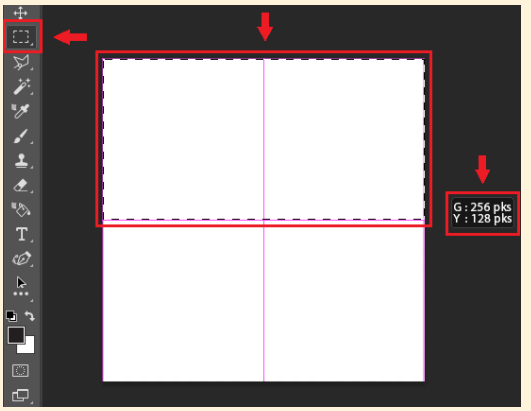
Aşağıdaki işlem adımlarına göre resim düzenleme programında duvar deseni oluşturunuz.

**1. Adım:** Resim düzenleme programını açınız ve Görsel 2.14'te gördüğünüz özelliklerde **yeni bir belge** oluşturunuz.



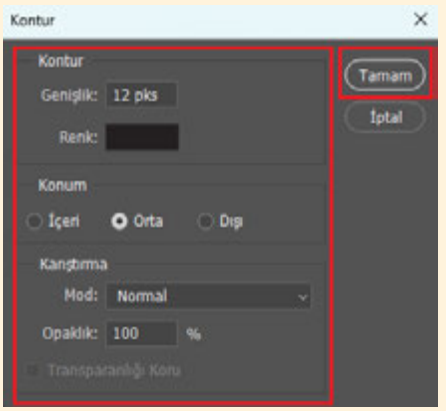

**Görsel 2.14:** Yeni belge ayarları

**2. Adım: Araçlar** panelinden **dikdörtgen seçim çerçevesi** aracını seçiniz. Görsel 2.15'te görüldüğü gibi 256 piksel genişliğinde 128 piksel yüksekliğinde bir seçim alanı oluşturunuz.



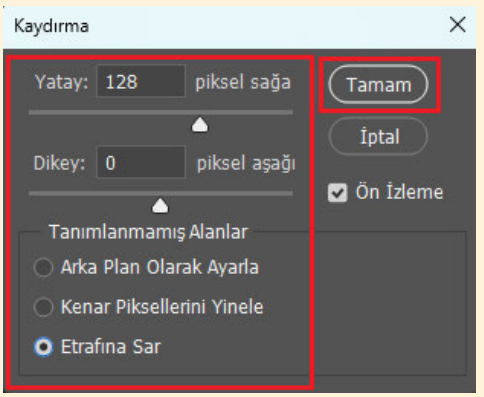

**Görsel 2.15:** Dikdörtgen seçim çerçevesi aracı ile seçim yapma

**3. Adım: Düzenle** menüsünden **Kontur** komutunu seçiniz. Açılan pencereden genişlik değerini 12 piksel ve konum seçeneğini **orta** olarak seçiniz ve **Tamam** düğmesine basınız (Görsel 2.16).




**Görsel 2.16:** Kontur düzenleme

**4. Adım: Seç** menüsünden **Ters** komutunu veriniz.
**5. Adım:** Seçimi iptal etmeden **üçüncü** adımdaki işlemleri tekrar ediniz.
**6. Adım: Filtre**> **Diğer**> **Kaydırma** komutunu tıklayınız. Açılan pencereden Görsel 2.17'de görülen değerleri giriniz ve **Tamam** düğmesine tıklayınız.




**Görsel 2.17:** Kaydırma filtresi

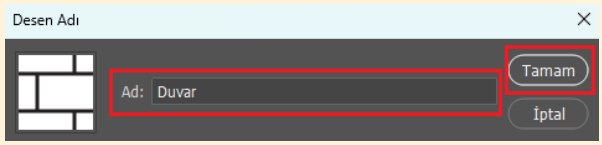
**7. Adım: Seç** menüsünden **Seçimi Kaldır** komutunu tıklayınız.

**8. Adım: Filtre**> **Diğer**> **Kaydırma** komutunu seçiniz. Açılan pencerede Görsel 2.18'deki değerleri giriniz ve **Tamam** düğmesine basınız.



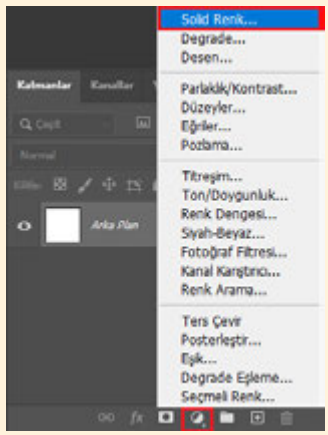

**Görsel 2.18:** Kaydırma filtresi ayarları

**9. Adım: Düzenle** menüsünden **Desen Tanımla** komutunu seçiniz. Açılan pencerede isim olarak "Duvar" ismini veriniz ve **Tamam** düğmesine tıklayınız (Görsel 2.19).



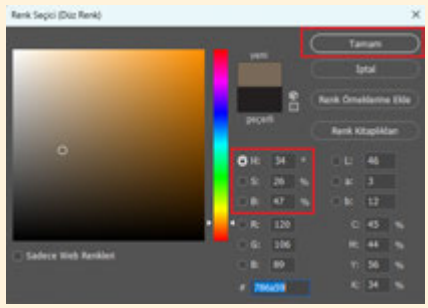

**Görsel 2.19:** Desen tanımlama

**10. Adım:** 1024 piksel genişliğinde 1024 piksel yüksekliğinde yeni bir belge oluşturunuz.
**11. Adım: Katmanlar** panelinde **yeni bir dolgu veya ayar katmanı oluştur** düğmesine tıklayarak açılan seçeneklerden **Solid** rengi seçiniz (Görsel 2.20).



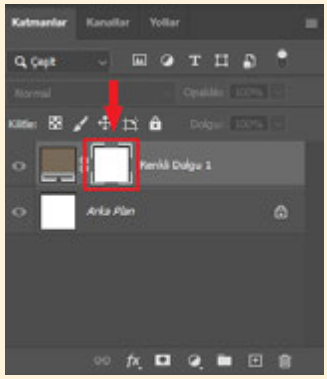

**Görsel 2.20:** Yeni bir dolgu veya ayar katmanı menüsü

**12. Adım: Renk seçici** penceresinde Görsel 2.21'deki değerleri giriniz ve **Tamam** düğmesine tıklayınız.



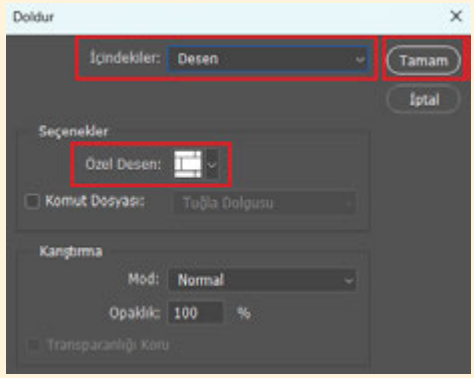
**Görsel 2.21:** Renk Seçici penceresi

**13. Adım: Katmanlar** panelinde katman maskesi resmine klavyeden **Alt** tuşuna basarak fare ile tıklayınız ve **katman maskesi** moduna geçiniz (Görsel 2.22).




**Görsel 2.22:** Katmanlar paneli

**14. Adım: Düzenle**> **Dolgu** komutunu seçiniz. Açılan pencerede, içindekiler alanından **Desen**'i seçiniz. **Özel Desen** olarak daha önce oluşturmuş olduğunuz **Duvar** dokusunu seçiniz ve **Tamam** düğmesine tıklayınız (Görsel 2.23).

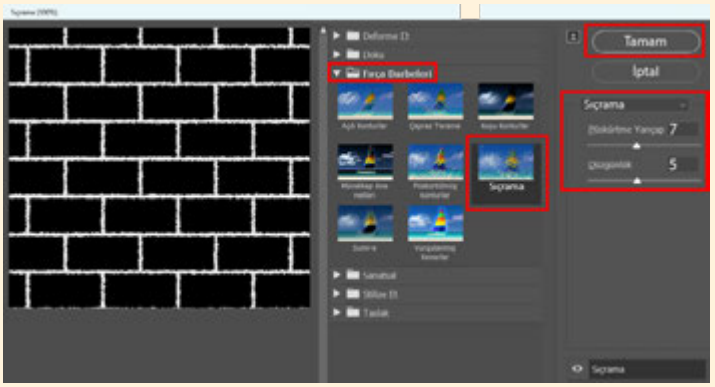



**Görsel 2.23:** Dolgu ayarları

**15. Adım: Görüntü**> **Ayarlamalar**> **Ters Çevir** komutunu seçiniz. **Katmanlar** panelinde, katman maskesi resmine klavyeden **Alt** tuşuna basarak fare ile tıklayınız ve **Duvar** deseni ile doldurma işlemini tamamlayınız.

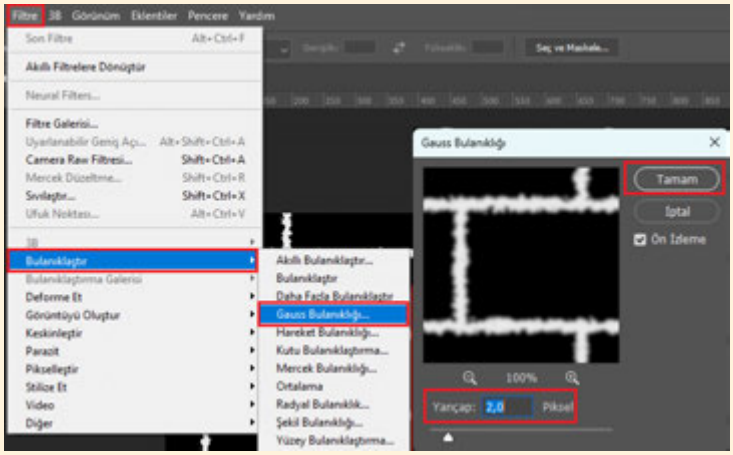
**16. Adım: Katmanlar** panelinde katman maskesi resmine klavyeden **Alt** tuşuna basarak fare ile tıklayınız ve **katman maskesi** moduna geçiniz. **Filtre**> **Filtre Galerisi** komutuna tıklayınız.

**17. Adım:** Açılan pencereden **Fırça Darbeleri** grubundaki **Sıçrama** filtresini seçiniz. Filtre ayarlarından püskürtme yarıçapını 7, düzgünlük değerini 5 olarak seçiniz ve **Tamam** düğmesine tıklayınız (Görsel 2.24).



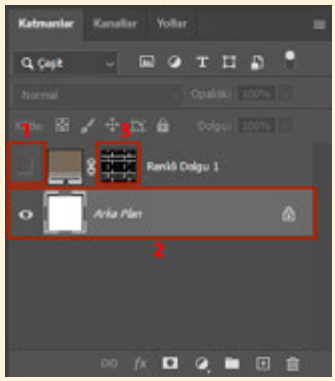

**Görsel 2.24:** Filtre Galerisi

**18. Adım: Filtre**> **Bulanıklaştır**> **Gauss Bulanıklığı** komutunu seçiniz. Açılan pencereden yarıçap değerini 2 piksel olarak giriniz ve **Tamam** düğmesine tıklayınız (Görsel 2.25).



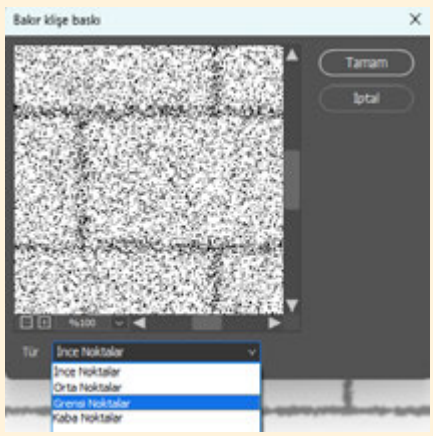

**Görsel 2.25:** Gauss Bulanıklığı filtresi

**19. Adım:** Görsel 2.26'da görüldüğü gibi **Katmanlar** panelinden **Renkli Dolgu 1** katmanını gizleyiniz (1). **Arka Plan** katmanını seçiniz (2). Katman maskesi resmine klavyeden **Ctrl** tuşuna basarak fare ile tıklayınız (3).



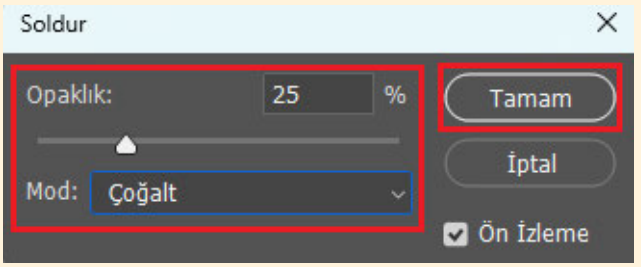

**Görsel 2.26:** Seçim aralığı oluşturma

**20. Adım: Düzenle**> **Dolgu** komutunu seçiniz. Açılan pencerede içindekiler alanından **%50 Gri** seçeneğini seçiniz ve **Tamam** düğmesine tıklayınız. Seçimi iptal etmek için **Ctrl + D** tuşlarına basınız.
**21. Adım: Filtre**> **Pikselleştir**> **Bakır klişe baskı** komutuna tıklayınız. Açılan pencereden **Tür** olarak **Grensi Noktalar**'ı seçerek **Tamam** düğmesine basınız (Görsel 2.27).



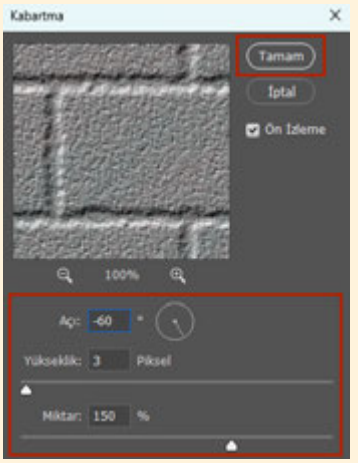

**Görsel 2.27:** Bakır klişe baskı filtresi

**22. Adım: Düzenle**> **Soldur Bakır klişe baskı** komutuna tıklayınız. Açılan **Soldur** penceresinde Görsel 2.28'de gördüğünüz ayarları uygulayınız.



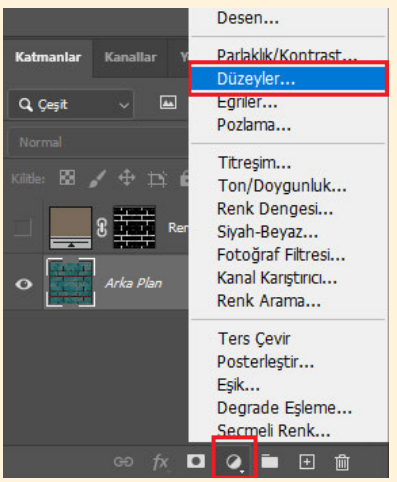

**Görsel 2.28:** Soldurma ayarları

**23. Adım: Filtre**> **Bulanıklaştır**> **Gauss Bulanıklığı** komutunu seçiniz. Açılan pencereden yarıçap değerini 0,5 piksel olarak giriniz ve **Tamam** düğmesine tıklayınız.
**24. Adım: Filtre**> **Stilize et**> **Kabartma** komutunu seçiniz. Açılan **Kabartma** penceresinde Görsel 2.29'da gördüğünüz ayarları uygulayınız.



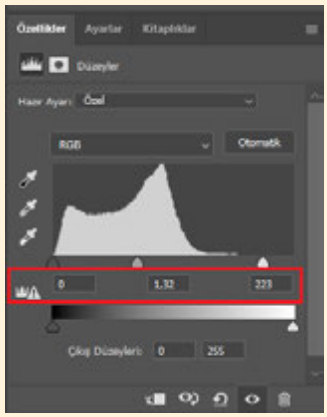

**Görsel 2.29:** Kabartma filtresi ayarları

**25. Adım: Düzenle**> **Soldur Kabartma** komutuna tıklayınız. Açılan pencereden **Mod** olarak **Fark** seçeneğini seçerek **Tamam** düğmesine basınız.
**26. Adım:** Katmanlar panelinde **yeni bir dolgu veya ayar katmanı oluştur** düğmesine tıklayarak açılan seçeneklerden **Düzeyler** seçeneğini seçiniz (Görsel 2.30).



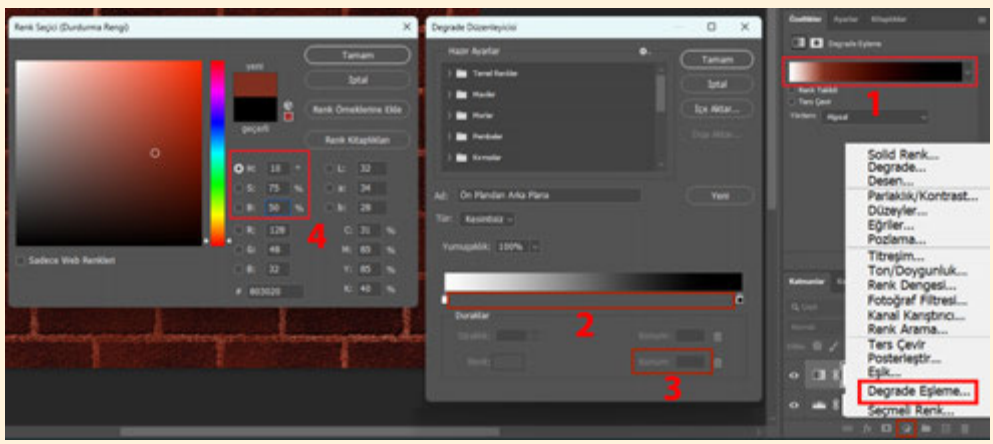

**Görsel 2.30:** Yeni bir dolgu veya ayar katmanı menüsü

**27. Adım: Düzeyler 1** katmanının **Özellikler** penceresinde ilgili alanlara Görsel 2.31'deki değerleri giriniz.



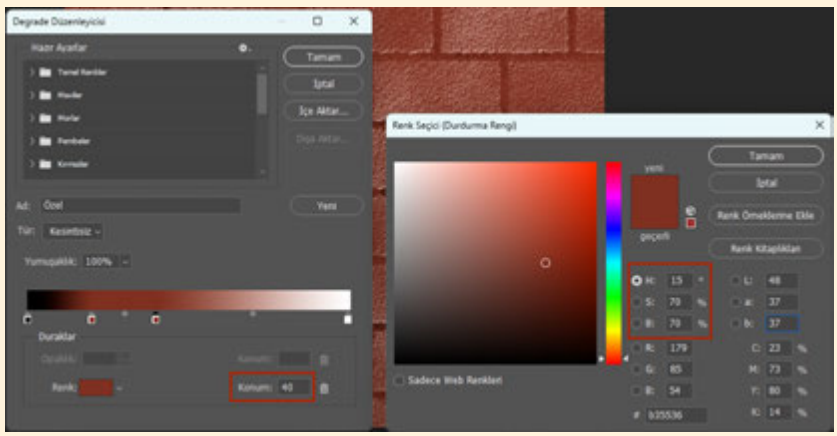

**Görsel 2.31:** Düzey ayarları

**28. Adım: Katmanlar** panelinde **yeni bir dolgu veya ayar katmanı oluştur** düğmesine tıklayarak açılan seçeneklerden **Degrade Eşleme** seçeneğini seçiniz. **Özellikler** penceresinde renk geçişi için Görsel 2.32'de görüldüğü gibi ilgili alana tıklayınız (1). Açılan pencerede **durak** seçmek için boş bir yere tıklayınız (2). Tam konum için 20 değerini giriniz (3). Oluşturduğunuz durak noktasına çift tıklayarak **Renk Seçici** penceresinde renk değerlerini giriniz (4) ve **Tamam** düğmesine basınız.



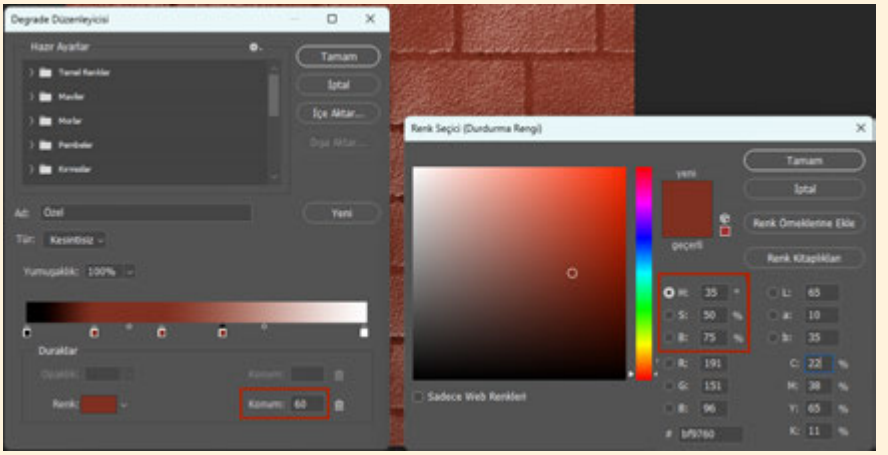

**Görsel 2.32:** Renk geçişi ayarları

**29. Adım:** Bir önceki adımda **Degrade Düzenleyicisi** penceresinde yaptığımız durak ekleme işlemini, Görsel 2.33'ü referans alarak **ikinci durak** eklemek için tekrarlayınız.



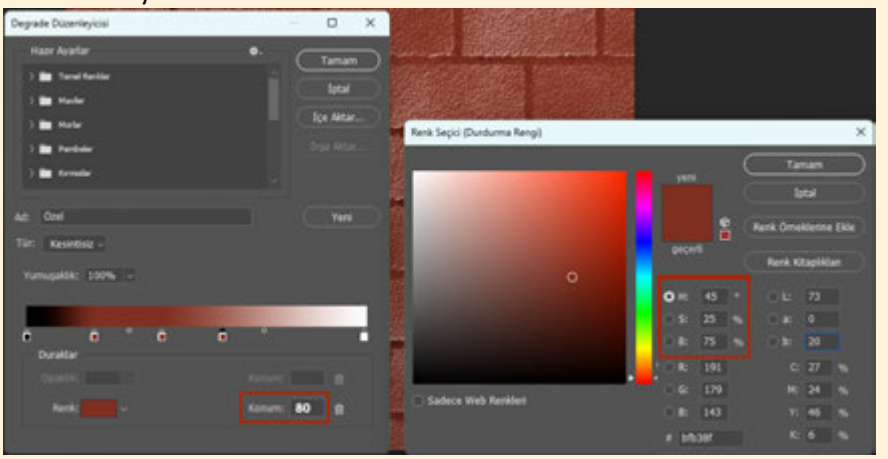

**Görsel 2.33:** Renk geçişine ikinci durak ekleme

**30. Adım: Degrade Düzenleyicisi** penceresinde **üçüncü durak** ekleme işlemini Görsel 2.34'ü referans alarak tekrarlayınız.



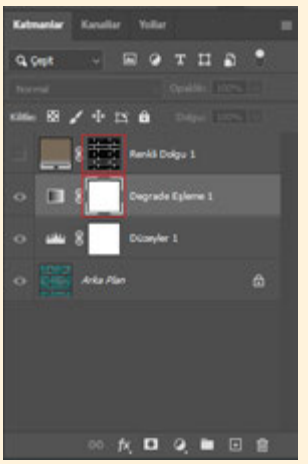

**Görsel 2.34:** Renk geçişine üçüncü durak ekleme

**31. Adım: Degrade Düzenleyicisi** penceresinde **dördüncü durak** ekleme işlemini Görsel 2.35'i referans alarak tekrarlayınız.



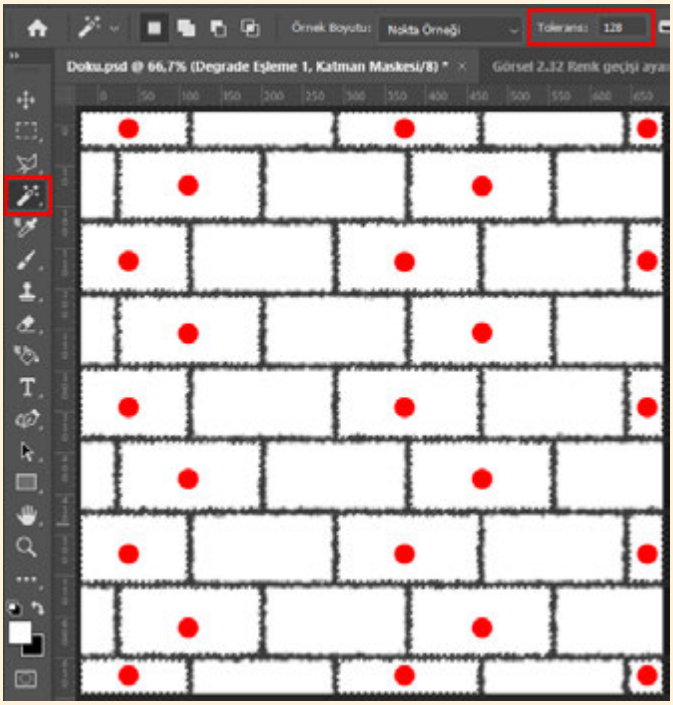

**Görsel 2.35:** Renk geçişine dördüncü durak ekleme

**32. Adım: Katmanlar** panelinde klavyenin **Alt** tuşu ile **Degrade Eşleme 1** katmanının küçük resmine tıklayınız. **Ctrl** tuşu ile **Renkli Dolgu 1** katmanının küçük resmine tıklayınız (Görsel 2.36).



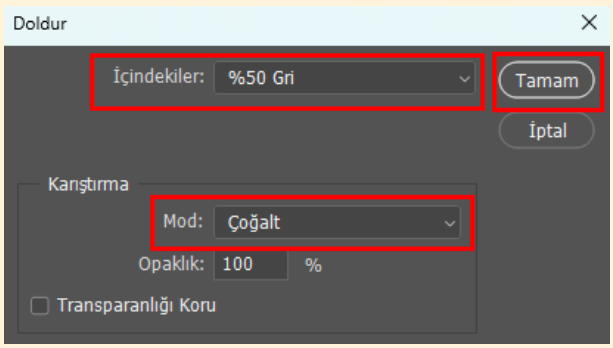

**Görsel 2.36:** Seçim aralığı oluşturma

**33. Adım: Düzenle**> **Dolgu** komutunu seçiniz. Açılan pencerede içindekiler alanından **Siyah** seçeneğini seçiniz ve **Tamam** düğmesine tıklayınız. Seçimi iptal etmek için **Ctrl + D** tuşlarına basınız.
**34. Adım:** Sol tarafta yer alan araçlardan **Sihirli Değnek** aracını seçiniz. **Tolerans** değerini **128** olarak girerek Görsel 2.37'deki işaretli alanları klavyenin **Shift** tuşuna basılı tutarak seçiniz.




**Görsel 2.37:** Sihirli Değnek aracı ile seçim yapma

**35. Adım: Düzenle**> **Dolgu** komutunu seçiniz. Açılan pencerede içindekiler alanından **%50 Gri** seçeneğini, **Mod** olarak **Çoğalt** seçeneğini seçiniz ve **Tamam** düğmesine tıklayınız (Görsel 2.38). Seçimi iptal etmek için **Ctrl + D** tuşlarına basınız.




**Görsel 2.38:** Dolgu ayarları

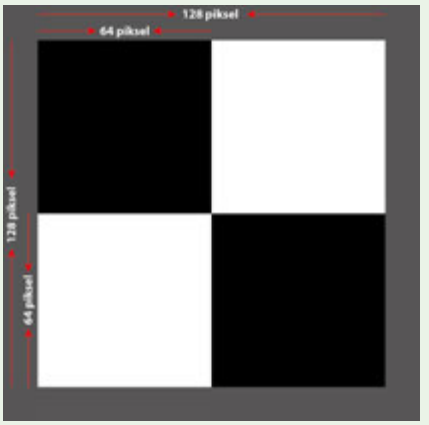
**36. Adım: Katmanlar** panelinde klavyenin **Alt** tuşuna basılı tutarak **Degrade Eşleme 1** katmanının küçük resmine tıklayınız.
**37. Adım: Renkli Dolgu 1** katmanını görünür yapınız. **Dosya**> **Dışa Aktar**> **Farklı Dışa Aktar** komutunu seçiniz. Resminizi **PNG** türünde ve "tuglaDokusu" isminde kaydediniz.

NOT

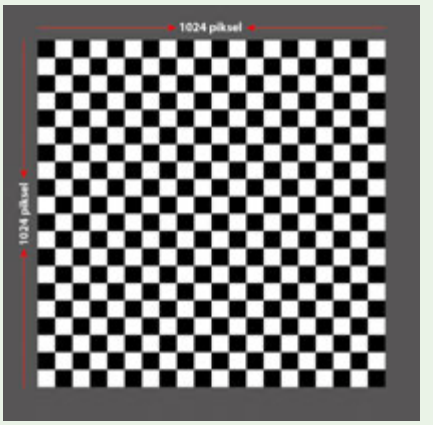
Bu uygulamada oluşturduğunuz **tuglaDokusu.png** resmi sonraki konularda tekrar kullanılacaktır.

SIRA SİZDE

Resim düzenleme programında Görsel 2.39'da görülen deseni oluşturunuz. Bu deseni kullanarak oluşturacağınız yeni belge üzerine dolgu ekleyiniz (Görsel 2.40).




**Görsel 2.39:** Desen oluşturma




**Görsel 2.40:** Desen kaplama

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Resim düzenleme programında belirtilen ölçülerde yeni bir belge oluşturdu. | | |
| 2. Araçları kullanarak istenilen görüntüyü oluşturdu. | | |
| 3. Oluşturduğu görüntüyü desen olarak kaydetti. | | |
| 4. İstenilen ölçülerde ikinci bir belge oluşturdu. | | |
| 5. Belgeyi önceden oluşturduğu desen ile kapladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 2.3. ÜÇ BOYUTLU MODELLEME

Bilgisayar programları kullanılarak bir nesnenin üç boyutlu dijital versiyonunun oluşturulmasını ve tasarlanmasını ifade eden bir süreçtir. Bu süreç, sanal ortamlarda gerçekçi nesnelerin oluşturulmasını sağlar. Nesnelerin boyutu, şekli, dokuları, renkleri ve diğer özellikleri dijital ortamda detaylı olarak modellenebilir. **Üç boyutlu modelleme;** mimari tasarım, oyun geliştirme, film ve animasyon endüstrisi, endüstriyel tasarım, tıp ve eğitim gibi birçok farklı alanda kullanılan çok yönlü bir teknolojidir.

Modelleme süreci, birçok farklı teknikle gerçekleştirilebilir. Poligon modelleme, NURBS modelleme, doku haritalama, ışıklandırma ve animasyon gibi teknikler 3B modellemenin farklı yönlerini kapsar. Sonuç olarak bu teknikler, gerçek hayattaki nesnelerin dijital versiyonlarının oluşturulmasında önemli rol oynar.

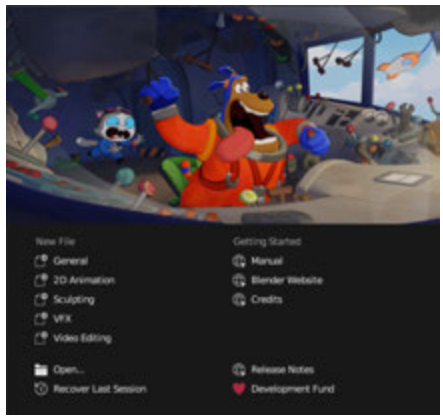
### 2.3.1. Üç Boyutlu Modelleme Programı

**3B modelleme yazılımları,** üç boyutlu nesnelerin oluşturulmasına ve düzenlenmesine olanak tanıyan araçlardır. Bu programlar; farklı sektörlerde, farklı beceri seviyelerindeki kullanıcılar için farklı gereksinimlere hitap eder. Örneğin bir kullanıcının karakter modelleme için kullandığı program ile animasyon için kullandığı program farklı olabilir. Bunun nedeni, programların farklı alanlarda öne çıkan ve gelişmiş özelliklere sahip olmasıdır.

**3B modelleme programı,** 3 boyutlu grafikler oluşturmak için kullanılan açık kaynaklı bir yazılımdır. Özellikle bağımsız geliştiriciler ve küçük stüdyolar için oldukça maliyetli olan diğer seçeneklerle karşılaştırıldığında, benzersiz güçlü özellikleriyle öne çıkar. Diğer birçok açık kaynak programda olduğu gibi 3B modelleme yazılımı da tamamen ücretsizdir. 3B modelleme programı geliştiricisinin resmî web sitesinden bu yazılım indirilebilir.

Program, bilgisayara kurulduğunda masaüstüne programın kısayol simgesi eklenecektir. Bu kısayol simgesi kullanılarak program başlatılabilir.

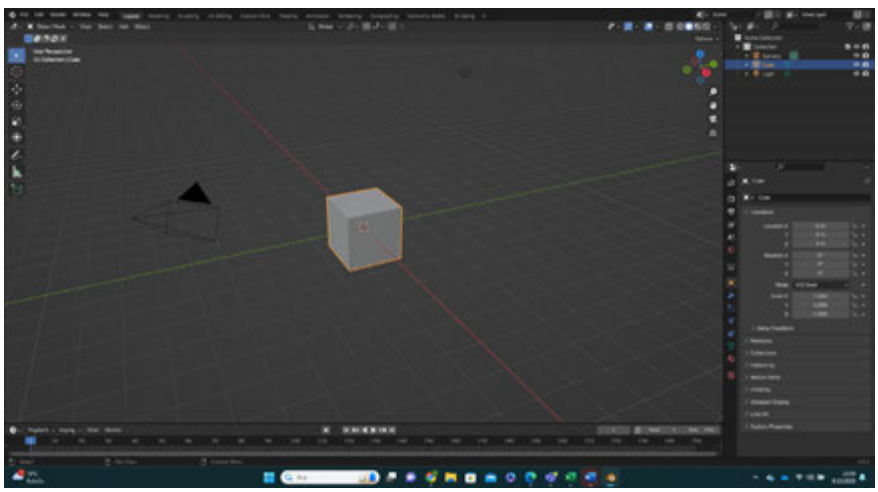
Başlangıç ekranı; çeşitli işlemler için önceden yapılandırılmış yeni bir çalışma dosyası oluşturmayı, son kullanılan dosyalara ve kayıtlı diğer dosyalara kolay erişim imkânını sağlar. Ayrıca 3B modelleme programı ile ilgili detaylı bilgiye bu ekrandan erişilebilir. Herhangi bir sebepten dolayı olumsuz bir durum yaşanırsa çalışmanın kurtarabileceği bir seçenek başlangıç ekranında görülebilir (Görsel 2.41).




**Görsel 2.41:** Başlangıç ekranı

Başlangıç ekranındaki seçenekler kullanılmak istenmiyorsa klavyeden **Esc** tuşuna basılabilir ya da çalışma ekranındaki herhangi boş bir alana fare ile tıklanabilir. Bu ekranın başlangıçta gösterilmesini engellemek için **Edit**> **Preferences**> **Interface** yolu takip edilerek **Splash Screen** kutucuğunun işareti kaldırılabilir.

3B modelleme programının çalışma alanında, kullanıcılara varsayılan olarak sunulan bir dizi panel bulunmaktadır. Bu panellerin düzeni, kullanıcıların tercihlerine göre kişiselleştirilebilir. **Çalışma alanı,** üç boyutlu bir alandır ve oyun motorundaki sahne görünümüne oldukça benzer bir görünüme sahiptir (Görsel 2.42).



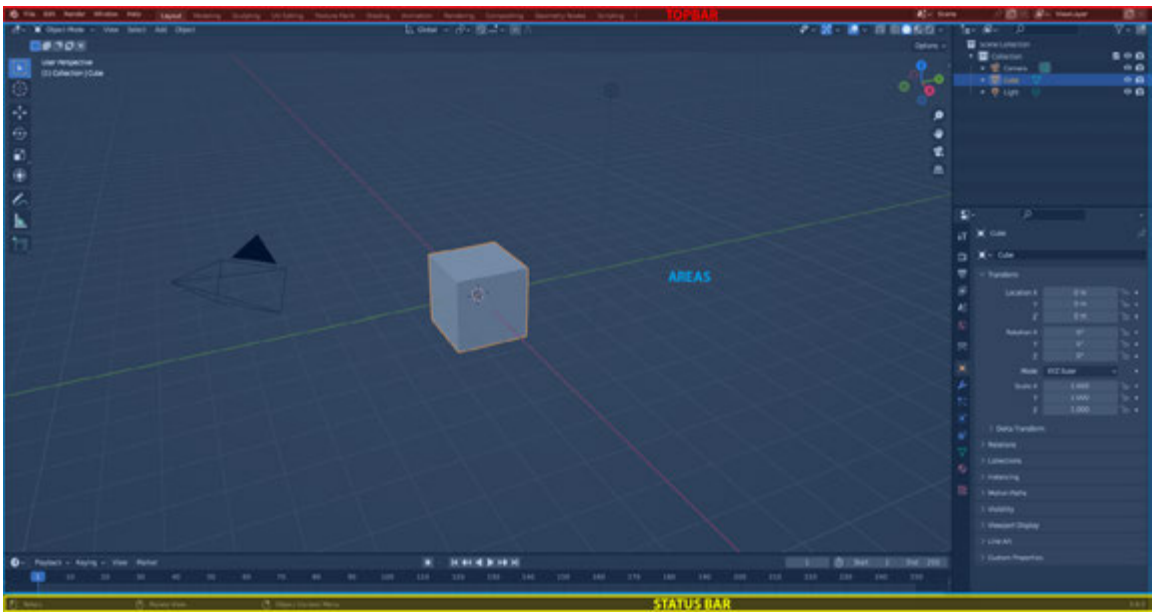
**Görsel 2.42:** Çalışma alanı

Görsel 2.43'te görüldüğü gibi program arayüzü üç ana bölüm hâlinde gruplandırılmaktadır.

**Topbar:** Projenin dosya işlemleri (kaydetme, açma, içe/dışa aktarma vb.), render seçenekleri, dosya düzenleme seçenekleri (kopyalama, yapıştırma vb.), görünüm ayarları, seçilen nesneyle ilgili özellikler (örneğin bir nesnenin ölçeklendirilmesi veya konumuyla ilgili ayarlar) gibi temel işlevleri, çalışma alanları arasında geçiş yapmak için kullanılan önceden tanımlanmış pencere düzenlerini içerir. Bu bölüm, kullanıcıların genel projelerini yönetmelerini ve temel işlevlere hızlıca erişmelerini sağlar.

**Areas:** Bu bölge, kullanıcıların başlıca çalışma alanını oluşturur. Kullanıcıların nesnelerle, sahnelerle ve araçlarla etkileşimde bulunmak için kullandıkları araçların tamamına yakını bu alanda yer alır.

**Status Bar:** Kısayol önerilerini ve ilgili istatistikleri görüntüler.




**Görsel 2.43:** Arayüz bölümleri

## 2.3.2. Düşük Poligon Modelleme

Düşük poligon sayısına sahip 3B modeller oluşturma sürecidir. Bu teknik, 3B grafiklerin oluşturulmasında kullanılır ve özellikle oyun geliştirme gibi alanlarda tercih edilir.

Düşük poligon modelleme, az sayıda poligon kullanarak nesneleri oluşturmayı amaçlar. **Poligonlar,** 3B nesnelerin yüzeylerini oluşturan çokgen şekillerdir. Düşük poligonlu modeller, daha az detay içerir ancak performans açısından avantaj sağlar ve daha küçük dosya boyutlarına sahiptir. Bu da oyun geliştiricileri için oyunların daha akıcı çalışmasını ve oyun geliştiricilerin daha geniş bir oyuncu kitlesine ulaşabilmesini sağlar.

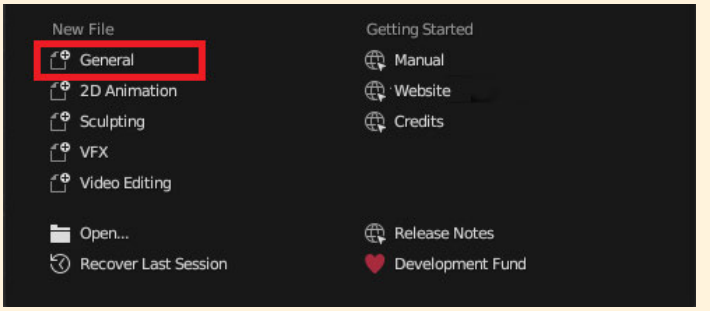
Düşük poligon modelleme yapılırken detayların az olması ve yüzeylerin daha basit geometrik şekillere indirgenmesi önemlidir. Bu teknik; genellikle oyunlarda karakterler, objeler, ortamlar ve arka planlar gibi görsellerin oluşturulmasında tercih edilir. Ayrıca çizgi film veya minimalist sanat tarzlarında da sıklıkla kullanılır.

### 7. UYGULAMA

Aşağıdaki işlem adımlarına göre üç boyutlu modelleme programında 3B araç modelini oluşturunuz.

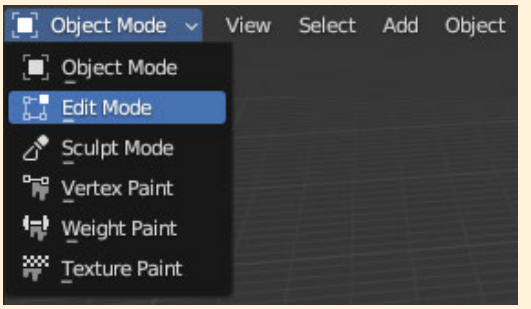
**1. Adım:** 3B modelleme programını çalıştırınız.

Açılış ekranından **General** (Genel) düğmesini tıklayarak yeni bir proje dosyası açınız (Görsel 2.44).



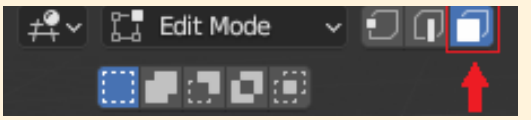

**Görsel 2.44:** Yeni proje oluşturma

**2. Adım:** Nesnede düzenleme yapabilmek için 3D Viewport (3B Görünüm Penceresi) başlığındaki **Object Mode**'u, **Edit Mode** olarak değiştiriniz (Görsel 2.45).



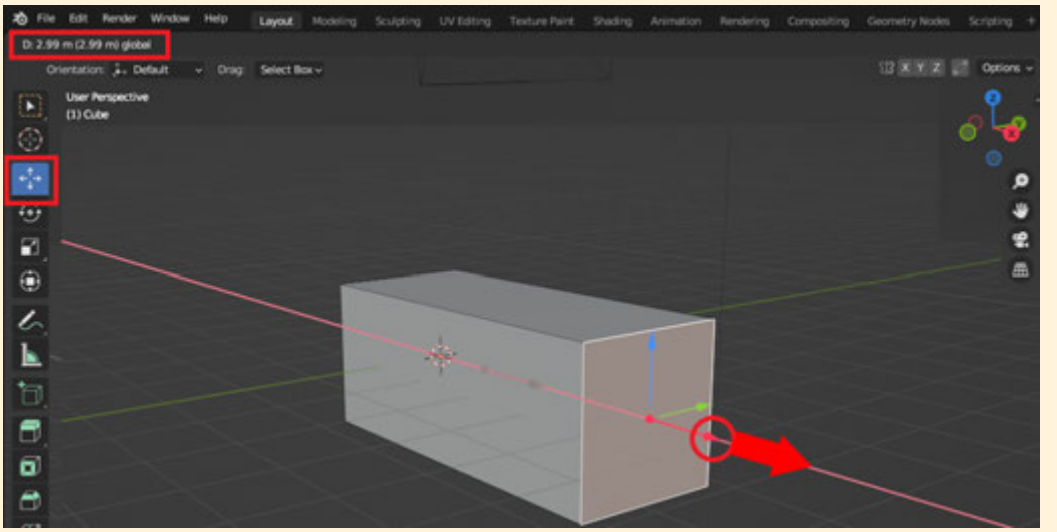

**Görsel 2.45:** Çalışma modu değiştirme

**3. Adım:** Nesnenin yüzeyi ile ilgili işlem yapabilmek için **Face Select** modunu seçiniz (Görsel 2.46).



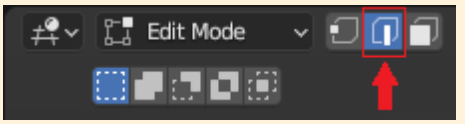

**Görsel 2.46:** Face Select modu

**4. Adım:** Nesnenin bir yüzeyini fare ile seçtikten sonra sol tarafta yer alan araçlardan **Move** aracı ile nesneyi bir yöne doğru uzatınız (Görsel 2.47).



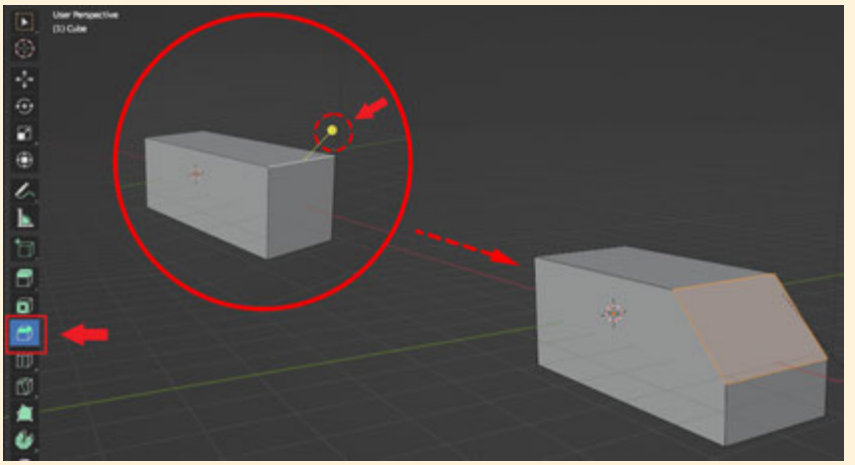
**Görsel 2.47:** Nesne yüzeyini uzatma

**5. Adım:** Nesnenin kenarı ile ilgili işlem yapabilmek için **Edge Select** modunu seçiniz (Görsel 2.48).



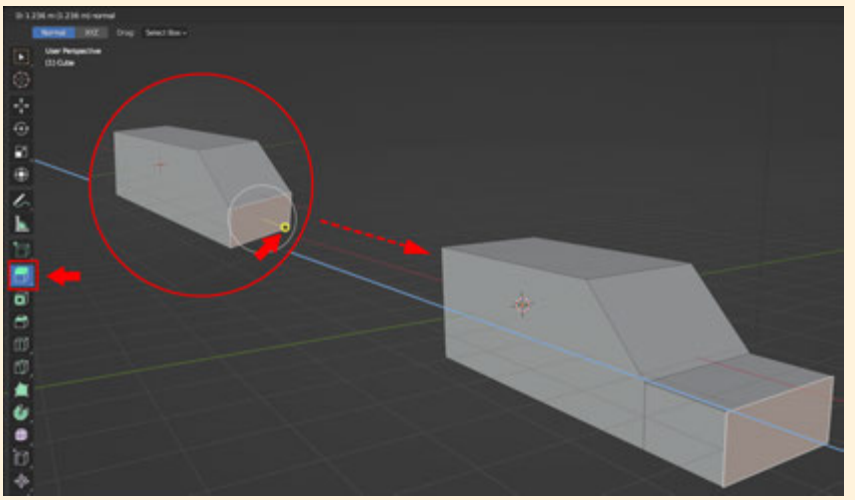

**Görsel 2.48:** Edge Select modu

**6. Adım:** Nesnede eğimli yeni bir yüzey oluşturmak için sol tarafta yer alan araçlardan **Bevel** aracını seçiniz. Nesnenin üst kenarına farenin sol tuşu ile tıklayıp sürükleyerek eğimli yeni bir yüzey oluşturunuz (Görsel 2.49).



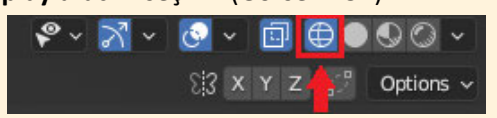
**Görsel 2.49:** Yüzeye eğim verme

**7. Adım:** Nesnenin yüzeyi ile ilgili işlem yapabilmek için **Face Select** modunu seçiniz. Fare ile nesnenin ön yüzeyini seçtikten sonra sol taraftaki araçlardan **Extrude Region** aracını seçiniz ve **+** simgesine fare ile tıklayıp sürükleyerek yüzeyi öne doğru uzatınız (Görsel 2.50).



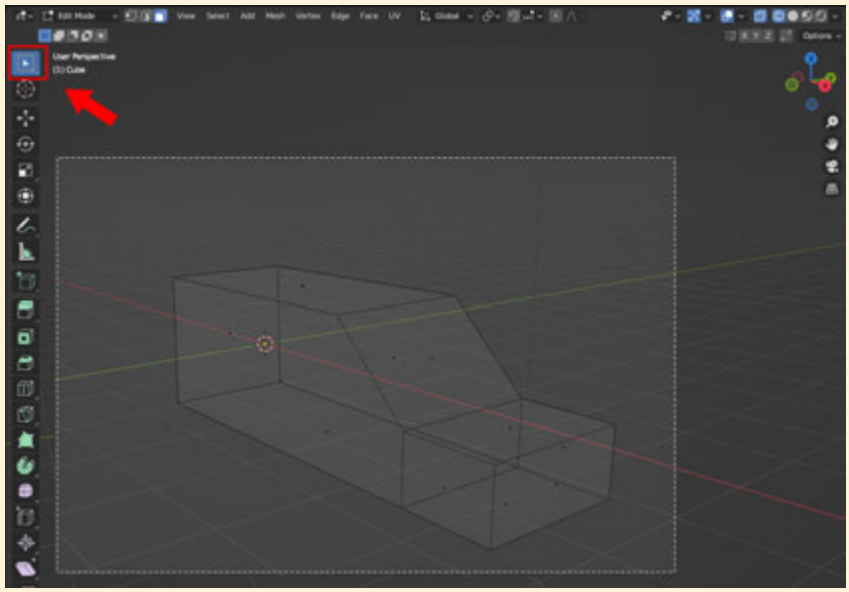
**Görsel 2.50:** Extrude Region aracı

**8. Adım:** Nesnemizin sahip olduğu tüm alanları seçebilmek için sağ üst tarafta yer alan görüntüleme araçlarından **Wireframe Display** aracını seçiniz (Görsel 2.51).



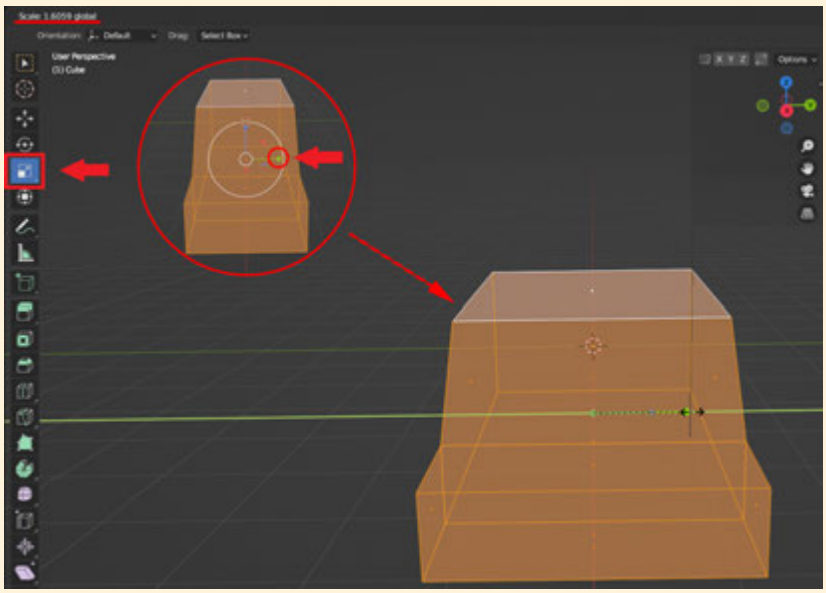

**Görsel 2.51:** Wireframe Display

**9. Adım:** Sol taraftaki **Select Box** aracını seçiniz ve fare yardımıyla nesnemizin tüm alanlarını seçili hâle getiriniz (Görsel 2.52).



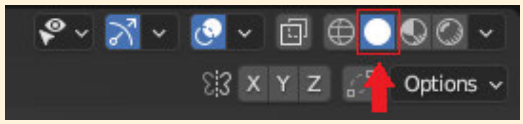
**Görsel 2.52:** Nesnenin tüm alanlarını seçme

**10. Adım:** Sol taraftaki araçlardan **Scale** aracını seçiniz ve nesneyi uygun eksende genişletiniz (Görsel 2.53).



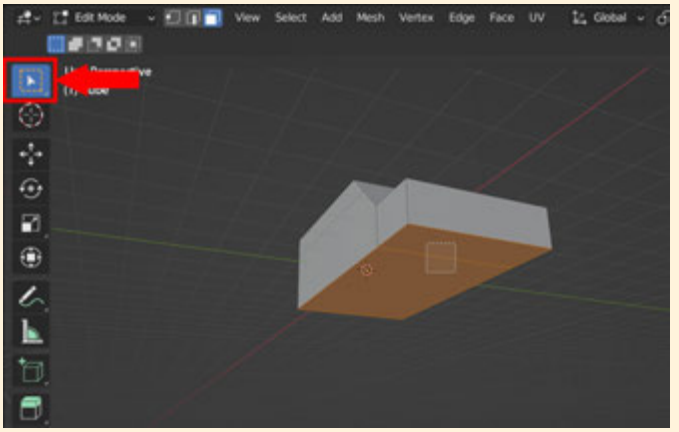
**Görsel 2.53:** Scale aracı ile genişletme

**11. Adım:** Sağ üst tarafta yer alan görüntüleme araçlarından **Solid Display** aracını seçiniz (Görsel 2.54).



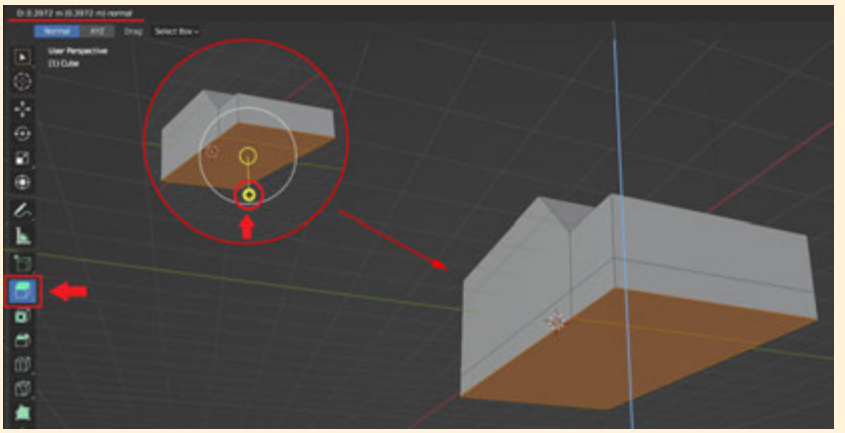

**Görsel 2.54:** Solid Display

**12. Adım:** Farenin **orta tuşunu** kullanarak perspektifi nesnenin alt yüzeylerini görecek konuma getiriniz ve sol taraftaki **Select Box** aracını seçerek nesnenin alt yüzeylerini seçili hâle getiriniz (Görsel 2.55).



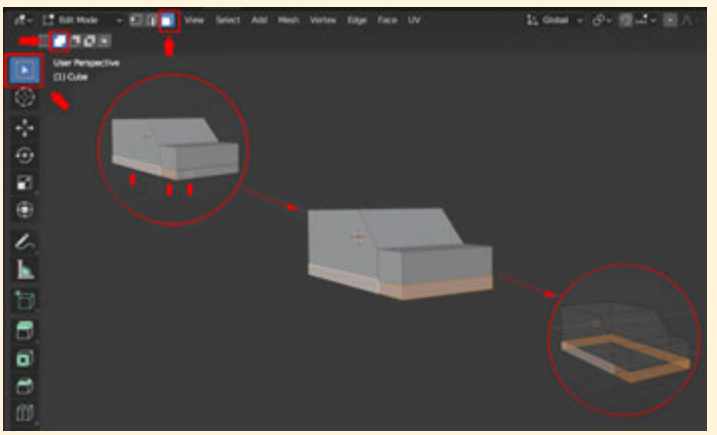
**Görsel 2.55:** Nesnenin alt yüzeylerini seçme

**13. Adım:** Sol tarafta yer alan **Extrude Region** aracını seçiniz ve fare yardımı ile nesneyi uygun eksende genişletiniz (Görsel 2.56).



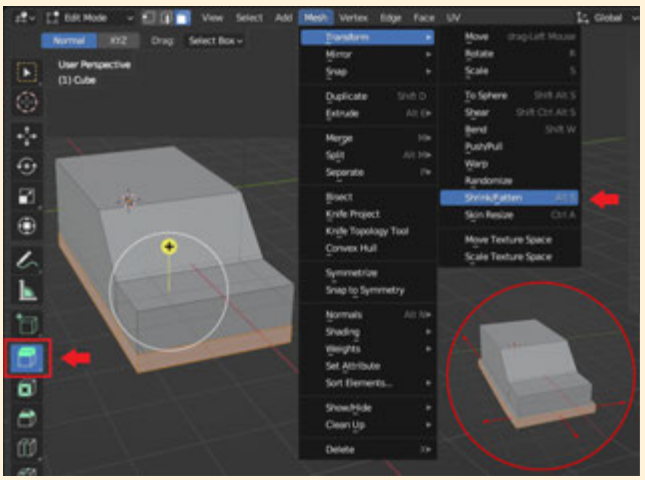
**Görsel 2.56:** Extrude Region aracı ile genişletme

**14. Adım:** Sol tarafta yer alan **Select Box** aracını seçiniz. Yüzey seçimi için **Face Select** ve hemen altında yer alan **Extend** modunu seçiniz. Nesnenin **yan-alt** yüzeylerinin tamamını fare yardımıyla seçiniz (Görsel 2.57).



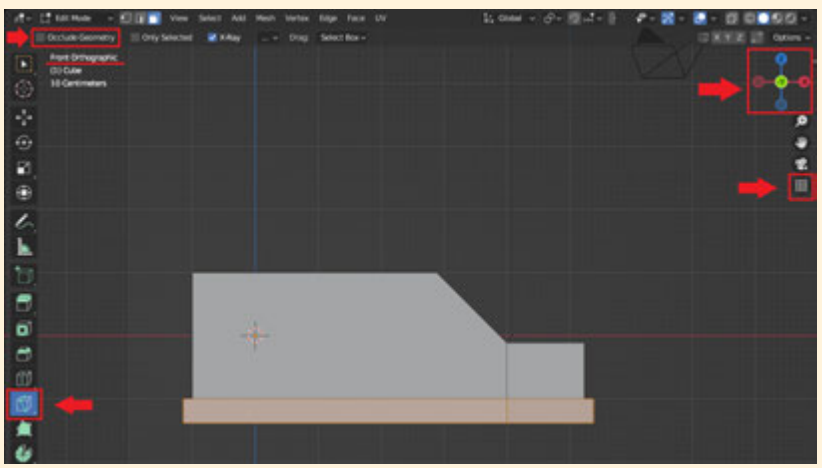
**Görsel 2.57:** Nesnenin belirli yüzeylerini seçili hâle getirme

**15. Adım:** Sol tarafta yer alan **Extrude Region** aracını seçiniz. Yukarıda yer alan menüden **Mesh**> **Transform**> **Shrink/Fatten** seçimini yapınız ve fareyi hiçbir tuşa basmadan, sadece yukarı aşağı oynatarak yan-alt yüzeyleri istediğiniz ölçüde genişletiniz (Görsel 2.58).



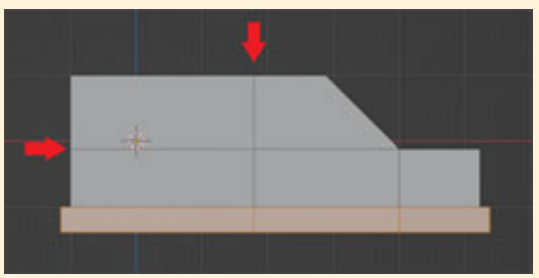
**Görsel 2.58:** Seçili yüzeyleri genişletme

**16. Adım:** Sağ tarafta yer alan sahne görünüm seçeneğini **Orthographic** olarak değiştiriniz ve nesnemizi herhangi bir açıdan net şekilde görecek şekilde bakış açısını ayarlayınız. Sol tarafta yer alan **Knife** aracını seçiniz. Aracı seçtikten sonra sol üst tarafta yer alan **Occlude Geometry** seçimini iptal ediniz (Görsel 2.59).



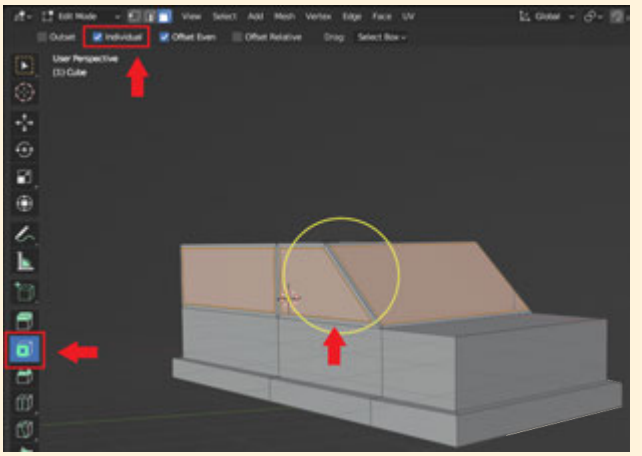
**Görsel 2.59:** Sahne görünümünü değiştirme

**17. Adım: Knife** aracı ile nesnemizin yan yüzeylerini Görsel 2.60'taki gibi kesiniz. Kesme işlemini başlatacağınız yere farenin sol tuşu ile tıkladıktan sonra bitireceğiniz yere farenin sol tuşu ile tekrar tıklayınız ve klavyeden **Enter** tuşuna basınız.



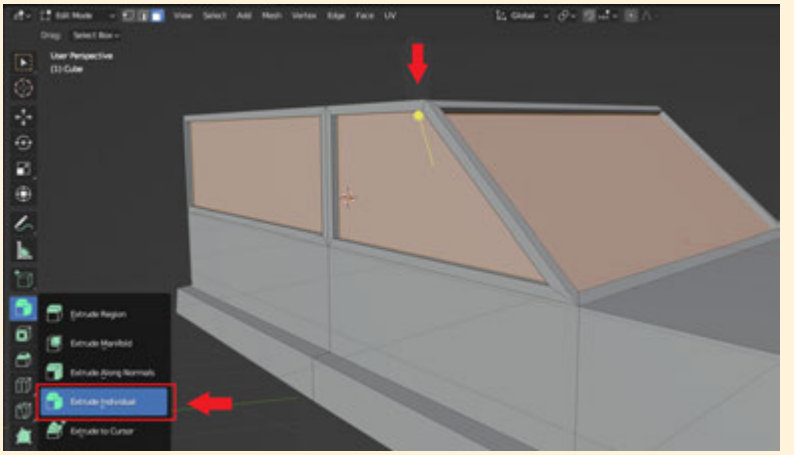
**Görsel 2.60:** Knife aracı ile kesme işlemi

**18. Adım: Select Box** aracı ile aracın camları için uygun parçaları seçiniz. Sol tarafta yer alan **Inset Faces** aracını seçiniz. Nesne üzerinde beliren çember içerisinde farenin sol tuşu ile tıklayınız ve fareyi yukarı-aşağı hareket ettirerek aracın camları için yüzey oluşturunuz (Görsel 2.61).



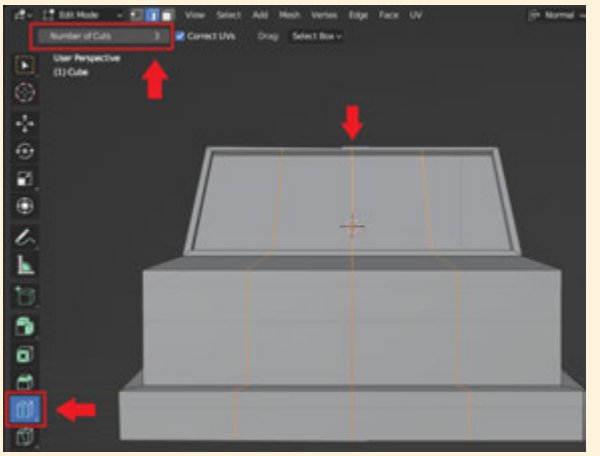
**Görsel 2.61:** Cam yüzeyi oluşturma

**19. Adım:** Tüm camlar seçili durumda iken **Extrude Individual** aracını seçiniz. Farenin sol tuşunu kullanarak ekrandaki kol yardımıyla cam yüzeylerini içeriye doğru daraltınız (Görsel 2.62).



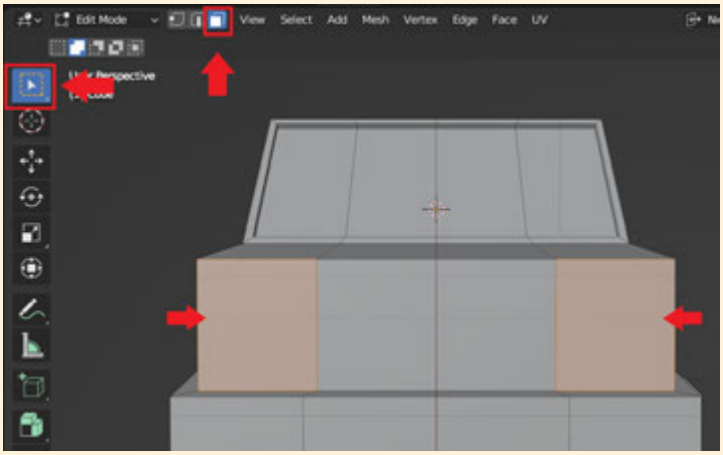
**Görsel 2.62:** Extrude Individual aracı ile yüzeye derinlik verme

**20. Adım: Loop Cut** aracı ile araç modelini Görsel 2.63'te gösterildiği gibi kesiniz.



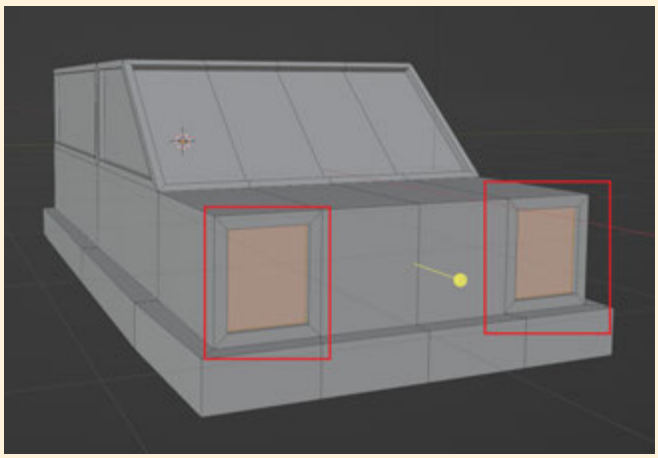
**Görsel 2.63:** Loop Cut aracı ile kesme işlemi

**21. Adım: Select Box** aracını seçiniz ve **Face select** moduna geçiniz. Görsel 2.64'te görüldüğü gibi aracın farları için iki yüzeyi birlikte seçiniz.



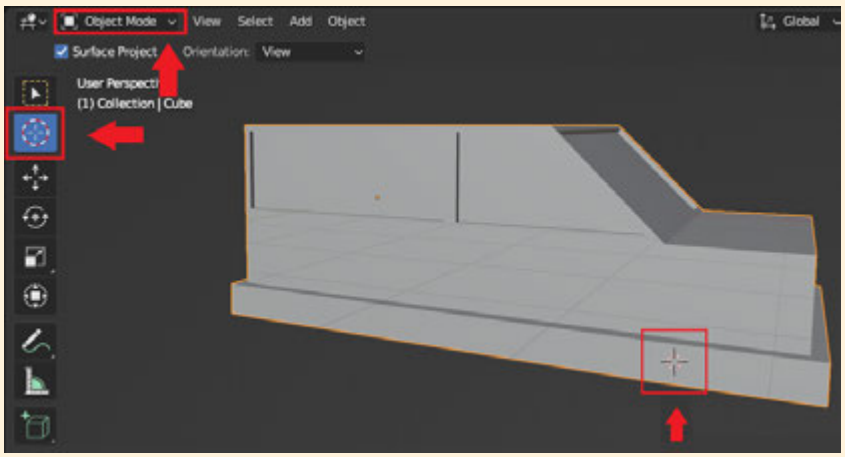

**Görsel 2.64:** Far yüzeylerini seçme

**22. Adım: Inset Faces** aracını kullanarak seçili yüzeylerde farlar için yüzey oluşturunuz. **Extrude Individual** aracı ile far yüzeylerini uygun ölçüde içeri doğru konumlandırınız (Görsel 2.65).



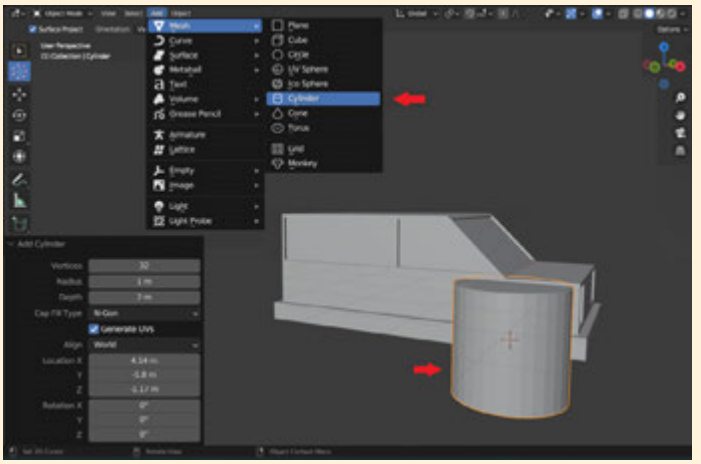
**Görsel 2.65:** Far yüzeyi oluşturma

**23. Adım: 3D Viewport** başlığındaki **Edit Mode**'u **Object Mode** olarak seçiniz. **Cursor** aracını seçiniz ve tekerlekleri ekleyeceğiniz alandaki bir noktaya tıklayarak kursörü taşıyınız (Görsel 2.66).



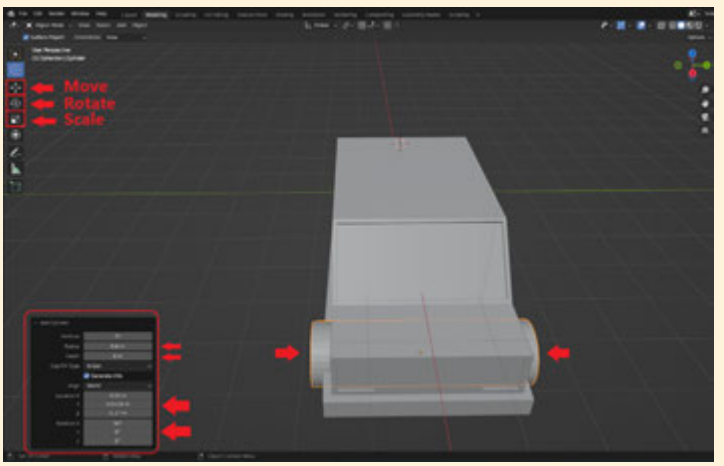

**Görsel 2.66:** Tekerlek konumunu belirleme

**24. Adım: Add**> **Mesh**> **Cylinder** yolunu takip ederek bir silindir ekleyiniz (Görsel 2.67).



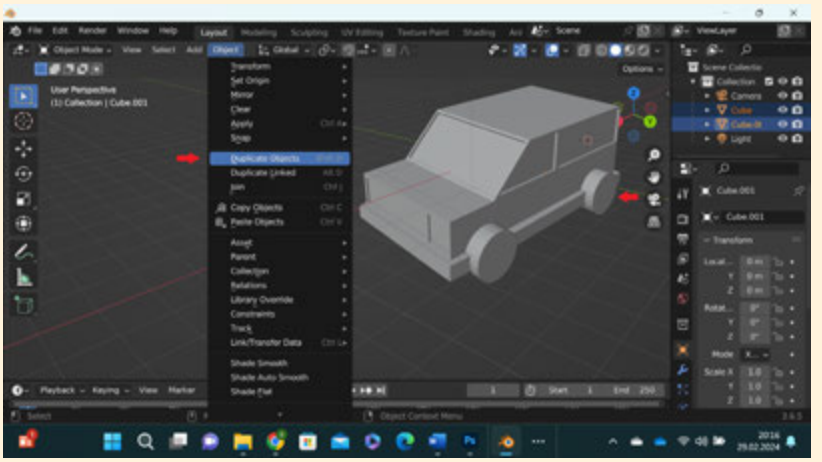
**Görsel 2.67:** Silindir ekleme

**25. Adım:** Eklediğiniz silindiri Görsel 2.68'de sol aşağıda gösterilen küçük pencereden veya sol üstte yer alan **Move**, **Rotate** ve **Scale** araçlarını kullanarak görseldeki hâle getiriniz.



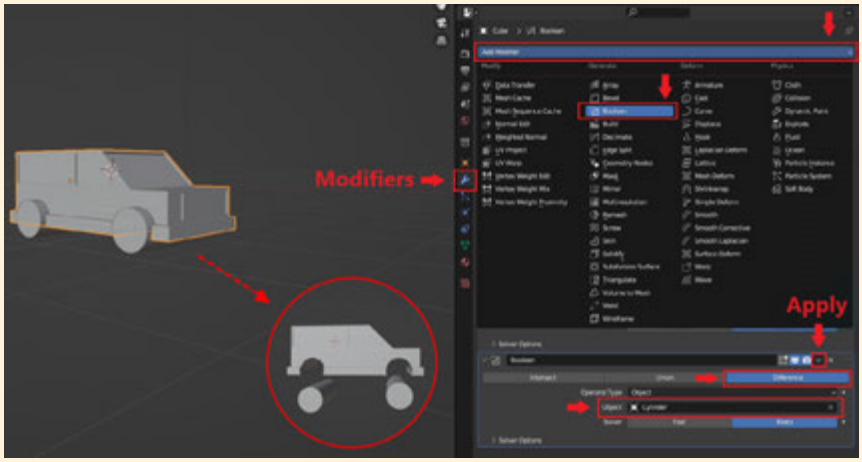

**Görsel 2.68:** Silindiri düzenleme

**26. Adım:** Silindir seçiliyken sol üst bölümde yer alan **Object** menüsünden **Duplicate Objects** komutuna tıklayınız ve klavyeden **Enter** tuşuna basınız. Çoğaltılan silindiri **Move** aracını kullanarak Görsel 2.69'daki gibi konumlandırınız.



**Görsel 2.69:** Silindirleri çoğaltma

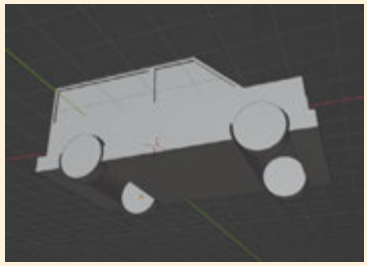
**27. Adım:** Silindir nesnesini araç gövdesi nesnesinden ayırmak için **Select Box** aracı ile aracı oluşturan üst objeyi seçiniz. Sağ bölümde yer alan **Modifiers** düğmesine tıklayınız. Sağ tarafta yer alan **Add Modifier** açılır penceresinden **Boolean** seçeneğini seçiniz. **Object** alanından oluşturduğunuz silindir objelerinin herhangi birini seçiniz. Mesh seçeneklerinde **Difference** seçeneğini seçiniz. İşlemin gerçekleşmesi için **Apply** komutuna tıklayınız veya klavyeden **Ctrl + A** tuşlarına basınız (Görsel 2.70).




**Görsel 2.70:** Tekerlekler için çıkarma işlemi

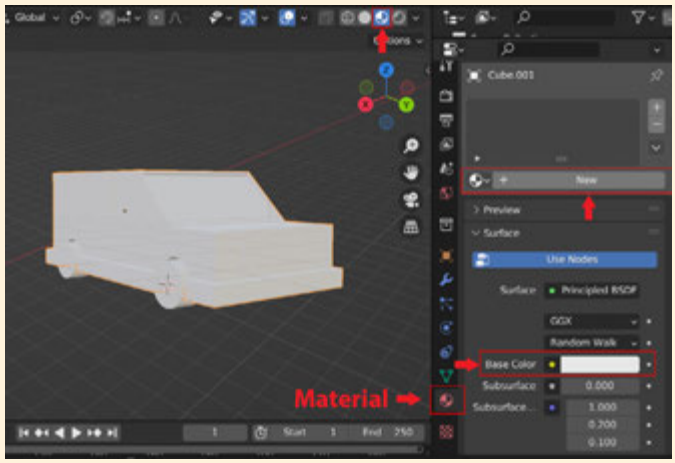
**28. Adım:** 27. Adımda yapılan işlemleri diğer silindir nesnesi için tekrarlayınız.

**29. Adım:** Silindirlerden herhangi birini siliniz. Diğer silindiri **Scale** ve **Move** araçlarını kullanarak araca uygun büyüklükte bir tekerlek ebatlarına getiriniz. **Object** menüsünden **Duplicate objects** komutunu kullanarak diğer üç tekerleği oluşturunuz ve konumlandırınız (Görsel 2.71).



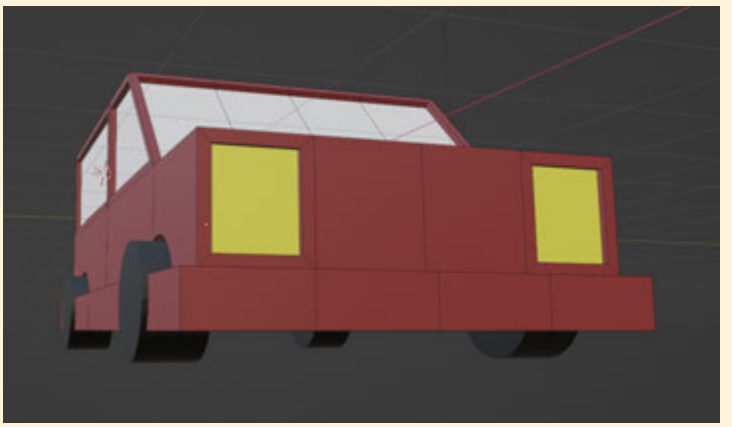
**Görsel 2.71:** Tekerlekleri konumlandırma

**30. Adım: Select Box** ile aracın gövdesini seçiniz. Üst bölümde yer alan görünüm seçeneklerinden **Material Preview** görünümünü seçiniz. Sağ bölümde yer alan **Material** düğmesine tıklayınız. Açılan pencereden **New** düğmesine tıklayınız ve yeni bir materyal oluşturunuz. **Base Color** seçeneği ile araç için bir renk belirleyiniz (Görsel 2.72).




**Görsel 2.72:** Aracı renklendirme

**31. Adım:** Aracın tekerlekleri için renklendirme işlemini 30. Adımdaki işlemlere benzer şekilde yapınız.
**32. Adım:** Aracın camlarını renklendirmek için araç seçiliyken **3D Viewport** başlığındaki **Object Mode**'u **Edit Mode** olarak değiştiriniz. **Select Box** aracı ile camların tümünü seçiniz. Yeni bir materyal ekleyiniz. **Base Color** seçeneği ile camlar için bir renk seçimi yapınız ve **Assign** düğmesine tıklayınız. Aracın farları için aynı işlemleri tekrar ediniz (Görsel 2.73).



**Görsel 2.73:** Renklendirilmiş araç gövdesi ve tekerlekler

SIRA SİZDE

**SIRA SİZDE**

Üç boyutlu modelleme programında bir adet mavi renkli 3B araç modelini oluşturunuz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

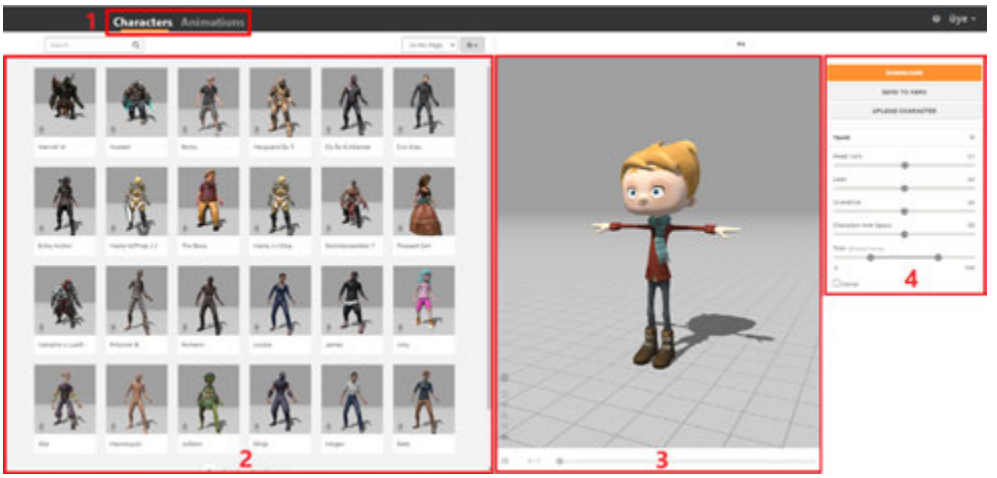
**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Üç boyutlu modelleme programında doğru seçim yaparak yeni bir belge oluşturdu. | | |
| 2. Uygulama araçlarını kullanarak çalışma düzlemine şekilleri ekledi. | | |
| 3. Doğru düzenleme modunu ve görüntüleme aracını seçerek şekilleri boyutlandırdı. | | |
| 4. Doğru düzenleme modunu ve görüntüleme aracını seçerek şekilleri hizaladı. | | |
| 5. Oluşturmuş olduğu modelde ilgili şekilleri seçerek "Material" görünümünde renklendirme işlemini gerçekleştirdi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

### 2.3.3. Üç Boyutlu Hazır Karakter ve Animasyon Platformu

İçerisinde hazır üç boyutlu karakter modelleri ve karakterler için animasyonlar sunan **hazır karakter ve animasyon platformu,** kullanımı basit ve ücretsiz bir çevrim içi kaynaktır. Bu platformda sunulan hazır karakterlere veya dışarıdan bu platforma yüklenilen karakterlere projeye uygun yürüme, oturma vb. önceden oluşturulmuş hazır animasyonlar uygulanabilir. Karakter ve animasyon seçimi yapıldıktan sonra model indirilebilir ve indirildikten sonra üzerinde düzenlemeler yapılabilir. Bu platformun kullanılabilmesi için ücretsiz olarak kaydolunabilen bir hesabın olması gerekir.

Görsel 2.74'te görüldüğü gibi **1** numaralı bölümde yer alan **Karakter**/**Animasyon** sekmeleri kullanılarak her iki grupta yer alan tüm kaynaklar **2** numaralı alanda görüntülenebilir. Projeye uygun karakter veya animasyon seçildikten sonra bunların ön izlemesi **3** numaralı alanda görüntülenebilir. **4** numaralı alanda yer alan araçlarla seçilen animasyon ile ilgili ayarlar yapılabilir, dışarıdan model yüklenebilir ve model bilgisayara indirilebilir.

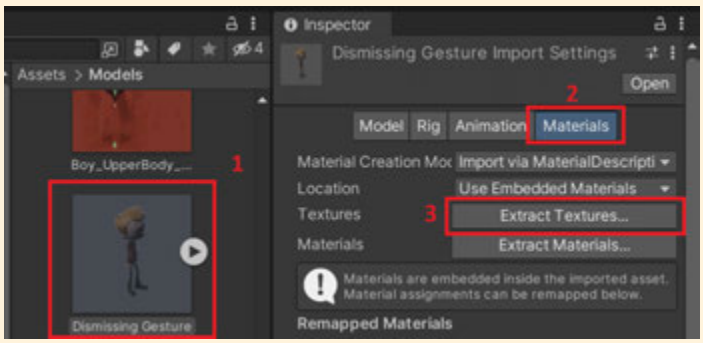



**Görsel 2.74:** Hazır karakter ve animasyon platformu

8. UYGULAMA

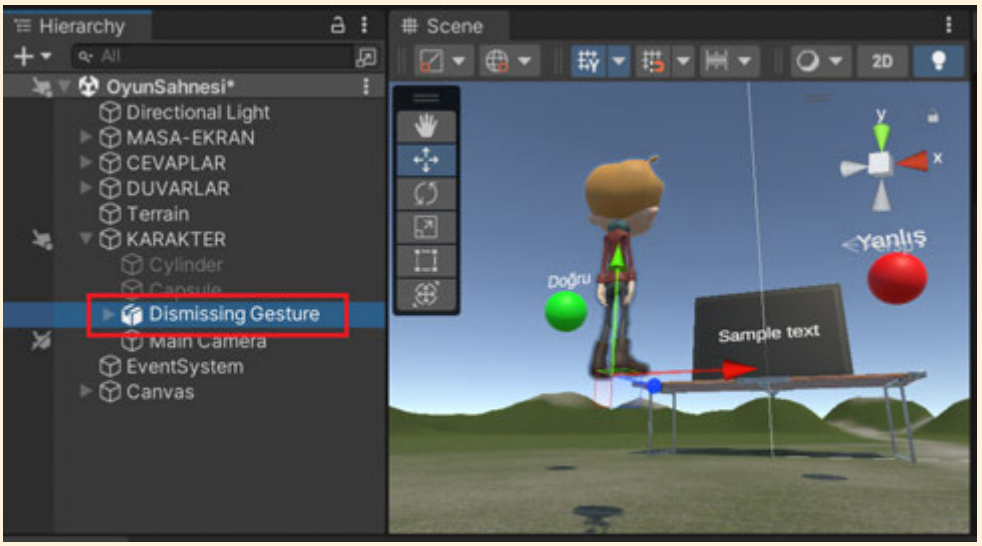
Aşağıdaki işlem adımlarına göre hazır karakter modelini oyun projenize ekleyiniz.

**1. Adım:** Hazır karakter ve animasyon platformunun web adresine gidiniz ve ücretsiz üye olunuz. Daha önceden oluşturmuş olduğunuz bir üyeliğiniz varsa platforma giriş yapabilirsiniz.
**2. Adım:** Projenize uygun karakter seçimini yapınız.
**3. Adım:** Seçtiğiniz karakteri oyun motoruna uygun formatta bilgisayarınıza indiriniz.
**4. Adım:** İndirdiğiniz modeli oyun motorunda **Assets**> **Models** Klasörü içine aktarınız.
**5. Adım:** İçe aktardığınız modeli seçerek **Inspector** panelinde **Materials** sekmesine tıklayınız. **Extract Textures** düğmesine tıklayarak modele ait ek dosyaların hangi klasöre aktarılacağını belirleyiniz (Görsel 2.75).




**Görsel 2.75:** Hazır karakter modeline doku ekleme

**6. Adım: Assets**> **Models** içerisindeki modeli, **Hierarchy** panelindeki **KARAKTER** oyun nesnesinin altına ekleyiniz (Görsel 2.76).



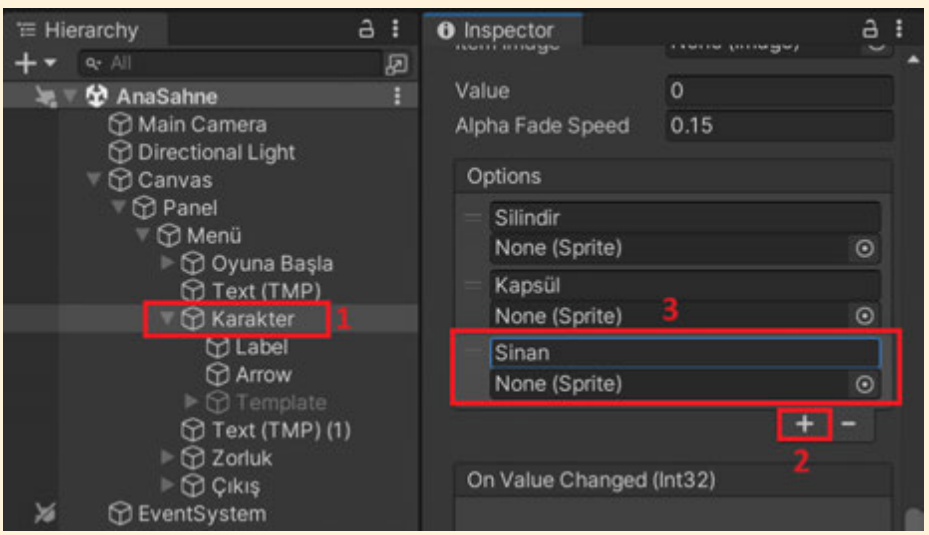

**Görsel 2.76:** Hazır karakter modelini sahneye ekleme

**7. Adım:** Sahneye eklediğiniz modelin **Transform** ayarlarını Görsel 2.77'deki gibi değiştiriniz.




**Görsel 2.77:** Hazır karakter modelinin Transform ayarları

**8. Adım: AnaSahne** isimli sahneyi açınız. **Karakter** isimli açılır listeyi seçerek **Inspector** panelindeki **Options** alanında **+** düğmesine basınız. Yeni eklenen satırda isim değişikliğini yapınız (Görsel 2.78).

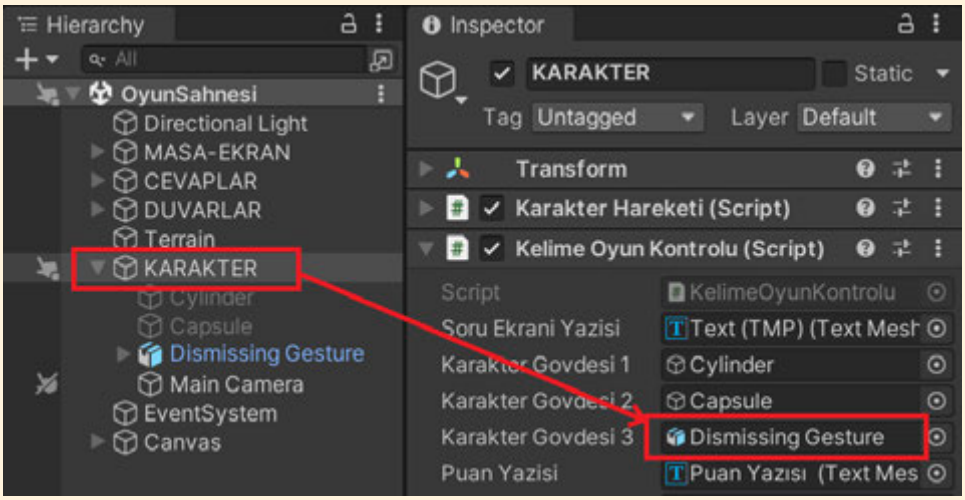



**Görsel 2.78:** Hazır karakter modelini oyunun karakter seçeneklerinin içerisine ekleme

**9. Adım:** "KelimeOyunKontrolu" isimli script dosyasını açarak **karakterGovdesi1**, **karakterGovdesi2** değişkenlerinin yanına yeni karakterimiz için aşağıdaki kırmızı renkli değişken tanımlamasını yapınız.

```
public GameObject karakterGovdesi1,karakterGovdesi2,
karakterGovdesi3;
```

**10. Adım:** "OyunSahnesi" isimli sahneyi açınız. "KARAKTER" isimli oyun nesnesini seçiniz. Inspector panelinde, **Kelime Oyun Kontrolu** script bileşenindeki **Karakter Govdesi 3** referans değişkenine hazır karakter modelini sürükleyiniz (Görsel 2.79).




**Görsel 2.79:** Hazır karakter modelinin referans değişkene atanması

**11. Adım:** "KelimeOyunKontrolu" isimli script dosyasını açarak **Start()** metodu içerisine aşağıdaki kırmızı renkli satırları ekleyiniz.

```
// AnaSahnede seçilen karaktere göre Silindir, Kapsül veya Hazır
// Model  görünür yapılır.
saklananKarakterSecenegi = PlayerPrefs.
GetInt("karakterSecenegi", 0);
if (saklananKarakterSecenegi == 0)
{
    karakterGovdesi1.SetActive(true);
    karakterGovdesi2.SetActive(false);
    karakterGovdesi3.SetActive(false);
}
else if (saklananKarakterSecenegi == 1)
{
    karakterGovdesi1.SetActive(false);
    karakterGovdesi2.SetActive(true);
    karakterGovdesi3.SetActive(false);
}
else if (saklananKarakterSecenegi == 2)
{
    karakterGovdesi1.SetActive(false);
    karakterGovdesi2.SetActive(false);
    karakterGovdesi3.SetActive(true);
}
```

SIRA SİZDE

Hazır karakter ve animasyon platformundan projenize uygun bir karakter seçerek hazır karakter modelini oyun projenize ekleyiniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

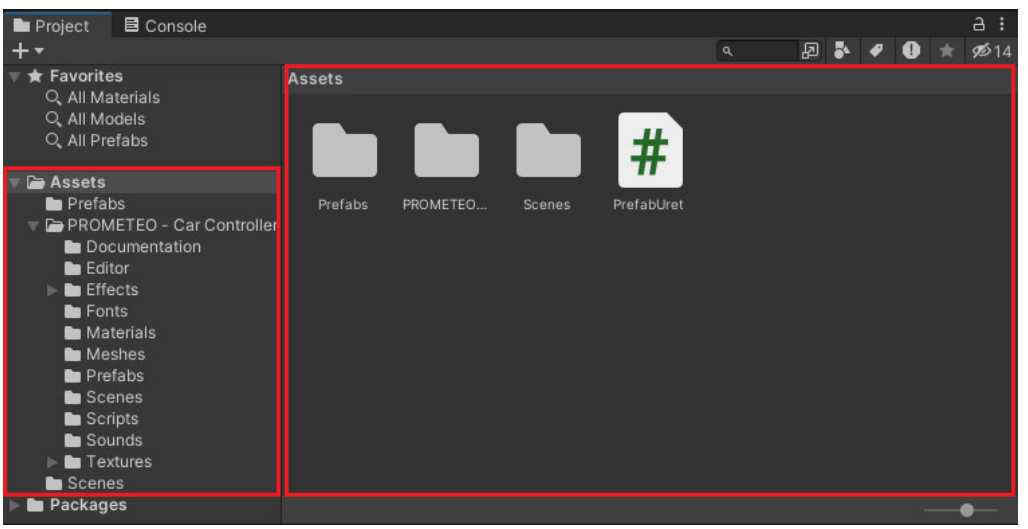
| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Projeye uygun karakter seçimini yaptı. | | |
| 2. Oyun motoruna uygun dosya formatını seçerek karakteri bilgisayarına indirdi. | | |
| 3. İndirdiği karakter modelini oyun motorunda uygun klasör içerisine aktardı. | | |
| 4. İçe aktardığı modeli seçerek modele ait ek dosyaları oyun motorunda uygun klasör içerisine aktardı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 2.4. OYUN PROGRAMINA AKTARMA

Oyun projenizde kullanmayı düşündüğünüz sesler, grafikler, 3B modeller gibi kaynakların tamamını oyun uygulamasında oluşturmak mümkün olmayabilir. Böyle bir durumda farklı programlara veya kullanacağınız dosya türünün hazır olarak sunulduğu platformlara ihtiyaç duyulacaktır. Bilgisayarda farklı programlar kullanılarak oluşturulmuş olan veya hazır kaynaklardan indirilen dosyalar oyun uygulamasına aktarılarak projelerde kullanılabilir. İçe aktarılan bu içerikler oyun uygulamasının sağladığı araçlar ve özellikler sayesinde kolayca yönetilebilir.

### 2.4.1. Projeye Dosya Ekleme ve Düzenleme

Oyun motoru projesinde içe aktarılacak resimler, modeller, metin dosyaları, ses dosyaları veya script dosyaları gibi çeşitli dosya türleri **Assets** klasörüne eklenir ve **Assets** sekmesinde listelenir (Görsel 2.80). Dışarıdan eklenen bu dosyalar türlerine göre, projede farklı amaçlar için kullanılabilir.




**Görsel 2.80:** Assets klasörü ve Assets sekmesi

**Asset Store**

Oyun motorunun sahip olduğu zengin içerikli dijital varlık kütüphanesidir. Bu kütüphane; oyun geliştiricilerin ihtiyaç duyduğu çeşitli varlıkları bulabileceği, indirebileceği ve kullanabileceği oldukça geniş içeriğe sahip bir mağazadır.

Bu mağazada geliştiricilerin projelerine çeşitli varlık türlerini eklemelerine yardımcı olabilecek çeşitli kaynaklar bulunmaktadır. Bu kaynaklar; 3B ve 2B modeller, karakterler, animasyonlar, materyaller, efektler, kod parçacıkları ve daha birçok farklı kategoride sunulmaktadır. Oyun geliştirme sürecini hızlandırmak ve kaliteli içerik konusunda ideal bir kaynaktır.

Asset Store mağazasında varlıklar ücretli veya ücretsiz olarak sunulmaktadır. Ücretli varlıklar genellikle daha yüksek kalitede ve özel özelliklere sahip olabilirken ücretsiz varlıklar genellikle başlangıç seviyesindeki geliştiriciler için faydalı olabilir. Varlıklar genellikle kategorilere ayrılmıştır ve kullanıcılar istedikleri varlık türünü ve özelliklerini filtreleyebilirler. Ayrıca geliştiriciler kendi varlıklarını Asset Store mağazasında yayımlayabilirler.

### 9. UYGULAMA

Aşağıdaki işlem adımlarına göre oyun projenize doku resmi ekleyiniz.
**1. Adım:** Oyun motorunda **Project** panelinde bulunan **Assets** klasörüne tıklayınız.
**2. Adım: Assets** panelinde boş bir alana farenin sağ tuşu ile tıklayarak **Create**> **Folder** seçeneğine tıklayınız ve yeni klasöre "Textures" ismini veriniz.
**3. Adım:** Dosyaları projenize eklemek için **Textures** klasörü üzerine fare ile sağ tıklayınız.
**4. Adım:** Açılan menüden **Import New Asset** seçeneğine tıklayınız.
**5. Adım:** Resim düzenleme programı ile daha önce oluşturmuş olduğunuz **tuglaDokusu.png** dosyasını seçtikten sonra **Import** düğmesine tıklayınız.
**6. Adım:** İçe aktardığınız resmin **Assets**> **Textures** klasörü altında listelendiğini kontrol ediniz.

#### NOT

**Assets** klasörüne dosya eklemenin diğer bir yolu da eklemek istediğiniz dosyayı bulunduğu klasörden seçerek, oyun motorunda **Assets** paneline sürükleyip bırakmaktır.

### 10. UYGULAMA

Aşağıdaki işlem adımlarına göre oyun projenize dışarıdan 3B araç modeli ekleyiniz.
**1. Adım: Window**> **Asset Store** seçeneğine tıklayarak **Asset Store** mağazasını web tarayıcıda açınız.
**2. Adım:** Üst bölümde yer alan arama kutusuna **car** (araba) kelimesini yazınız.
**3. Adım:** Sonuçlar listelendikten sonra sağ bölümde yer alan filtreleme alanından **Free Assets** kutucuğunu işaretleyiniz.
**4. Adım:** Listede yer alan ücretsiz araba varlıklarından herhangi birini seçiniz.
**5. Adım:** Açılan sayfada sağ tarafta yer alan **Add to My Assets** düğmesine tıklayınız.
**6. Adım:** Açılan uyarı penceresinde sözleşmeyi kabul etmek için **Accept** düğmesine tıklayınız.
**7. Adım:** Varlığın listenize eklendiğine dair gösterilen açılır pencerede **Open** düğmesine tıklayınız veya **Go to My Assets** düğmesine tıkladıktan sonra açılan sayfadaki varlıklar listenizden **Open** düğmesine tıklayınız.
**8. Adım:** Oyun motorunda **Package Manager** penceresinde **Asset Store** mağazasından varlıklar listenize eklemiş olduğunuz varlığı seçiniz ve sağ bölümde yer alan **Download** düğmesine basarak varlığı indiriniz.
**9. Adım:** İndirme işlemi bittikten sonra görülecek **Import** düğmesine basınız. Açılan pencerede herhangi bir değişiklik yapmadan paketin tamamını içeri aktarmak için tekrar **Import** düğmesine basınız.

**10. Adım:** İçe aktarma yüklemeleri bittikten sonra varlık paketinin **Assets** klasörü altında listelendiğini kontrol ediniz.
**11. Adım: Assets** klasörü içerisine eklediğiniz pakette 3B araba modelini bulunuz. Modeli sahnenize yerleştiriniz ve inceleyiniz.

SIRA SİZDE

Oyun motorunda **Assets** klasörü altında "Sprites" isimli bir klasör oluşturarak bilgisayarınızda kayıtlı herhangi bir resmi oyun motoruna aktarınız. Sprites klasörü altına eklediğiniz resmi sahneye ekleyiniz.

**Değerlendirme**

Çalışmalarınız diğer sayfada yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Assets klasörü altına Sprites isimli yeni bir klasör oluşturdu. | | |
| 2. Dışarıdaki resmi Sprites klasörü içerisine aktardı. | | |
| 3. Resmi sahneye ekledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

SIRA SİZDE

**Asset Store** mağazasında ücretsiz ağaç modeli paketlerini listeleyeniz. İstediğiniz modeli seçerek bilgisayarınızda kurulu olan oyun motoruna içe aktarınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Asset Store mağazasını tarayıcıda açtı. | | |
| 2. Arama terimini yazarak mağazada arama yaptı. | | |
| 3. Filtreleme alanından ücretsiz varlıklar kutucuğunu seçti. | | |
| 4. Modeli, mağaza hesabındaki varlıklar listesine ekledi. | | |
| 5. Modeli, oyun motoru editöründe açtı. | | |
| 6. Paket yöneticisini kullanarak modeli bilgisayarına indirdi. | | |
| 7. İndirilen modeli içe aktararak Assets klasörüne ekledi. | | |
| 8. Modeli sahneye yerleştirdi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 2.4.2. Tekrar Kullanılabilir Nesne (Prefab) Oluşturma

**Prefab yapıları,** oyun motorunda kullanılan önceden tanımlanmış nesne şablonlarıdır. Bu şablonlar; belirli özelliklere, bileşenlere ve ayarlara sahip olan oyun nesnelerinin klonlanabilir versiyonlarıdır.

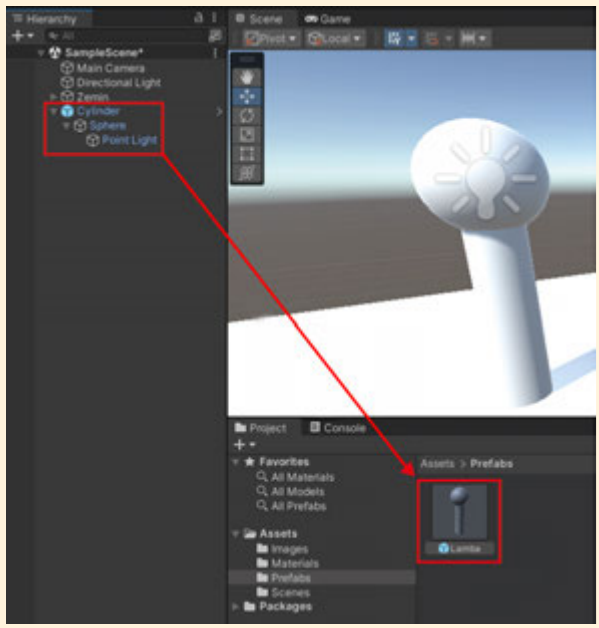
Prefab yapıları, aynı yapıdaki nesnelerin tekrar tekrar oluşturulmasını önler ve değiştirildiğinde tüm bu nesneleri etkiler. Bu özellikleriyle benzer özelliklere sahip birden fazla nesnenin hızlıca oluşturulmasını, düzenlenmesini ve güncellenmesini sağlar. Örneğin bir oyunda aynı özelliklere sahip birden fazla karakter varsa bu karakterleri tek tek oluşturmak yerine bir defa oluşturup bu nesne prefab hâline getirebilir. Daha sonra bu prefab, sahneye sürüklenip bırakılarak kolayca kopyalanabilir ve düzenlenebilir.

Prefab yapıları; oyun geliştirme sürecini daha verimli hâle getirir, tekrar kullanımı sağlar ve benzer nesnelerin yönetimini kolaylaştırır. Ayrıca prefab yapıları üzerinde yapılan değişiklikler diğer kopyaları etkilerken bağımsız olarak da düzenlenebilir, bu da projenin düzenini koruma esnekliği sunar.

### 11. UYGULAMA

Aşağıdaki işlem adımlarına göre oyun projenizde tekrar kullanılabilir nesne oluşturunuz.

**1. Adım: Lambalar** isminde yeni bir 3B oyun projesi oluşturunuz.
**2. Adım: Zemin** isminde bir **Plane** nesnesi ekleyiniz. Konumunu **(X=0, Y=0, Z=0)** olarak ayarlayınız.
**3. Adım:** Sahneye yeni bir **3D Cylinder** nesnesi ve bir **3D Sphere** nesnesi ekleyiniz. Sahnedeki **Directional Light** nesnesini **(X=0, Y=10, Z=0)** konumuna getiriniz.
**4. Adım:** Eklediğiniz silindiri **(X=0,5, Y=1, Z=0,5)** ölçeğiyle **(X=0, Y=1, Z=0)** noktasına konumlandırınız. Silindire **Rigidbody** bileşenini ekleyiniz.
**5. Adım:** Küreyi **(X=1, Y=1, Z=1)** ölçeğiyle **(X=0, Y=2,2, Z=0)** noktasına konumlandırınız. Kürenin üzerine bir **Point Light** bileşeni ekleyiniz.
**6. Adım: Hierarchy** penceresinde küreyi seçerek silindirin üzerine sürükleyiniz.
**7. Adım: Project** penceresinde **Assets** klasörü altında yeni bir klasör oluşturunuz. Yeni klasöre **Prefabs** adını veriniz. **Prefabs** klasörünün üzerine farenin sağ tuşu ile tıklayınız. Açılan menüden **Create**> **Prefab** komutunu seçerek **Lamba** isminde yeni bir **prefab** oluşturunuz.
**8. Adım: Hierarchy** penceresinde silindiri tıklayıp **Project** penceresindeki **Lamba** prefab yapısının üzerine sürükleyiniz (Görsel 2.81). **Hierarchy** penceresindeki silindirin simgesinin maviye döndüğünü fark edeceksiniz. Artık silindiri sahneden silebilirsiniz.



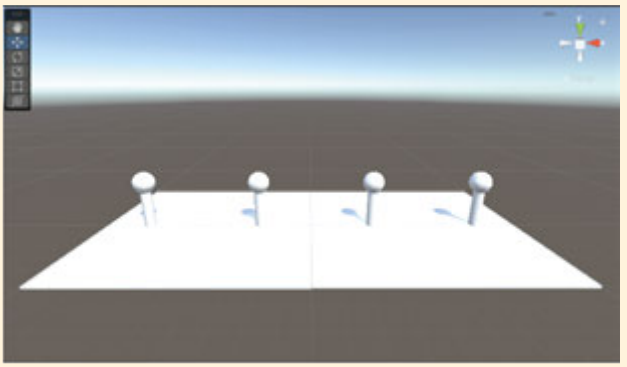

**Görsel 2.81:** Prefab yapısına nesne ekleme

**9. Adım: Lamba** prefab yapısını **Zemin** nesnesinin üzerine sürükleyiniz ve lambanın zeminin ortasında nasıl göründüğüne dikkat ediniz.

NOT

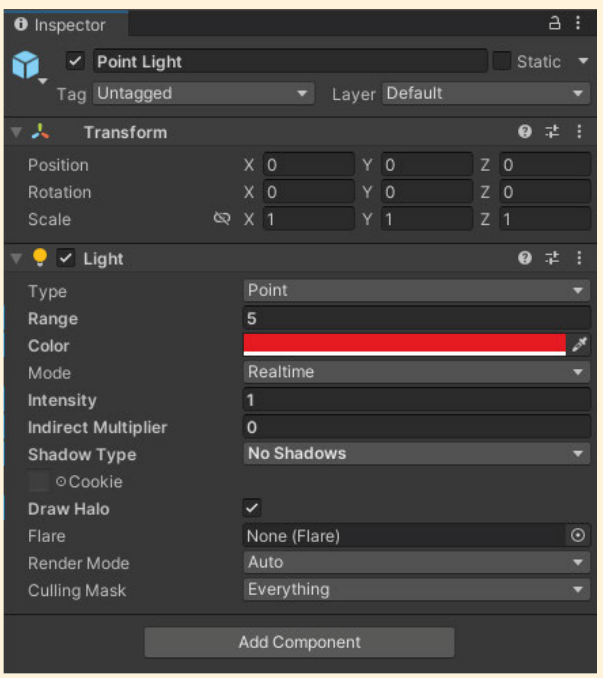
**Prefab** yapısını **Project** penceresinden sahneye sürüklerseniz, sürüklediğiniz yerde konumlanacaktır. **Hierarchy** penceresinin boş bir kısmına sürüklerseniz başlangıç konumu, prefab yapısında ayarlanan konum olacaktır.

**10. Adım: Lamba** prefab yapısından üç adet daha **Zemin** üzerine sürükleyiniz ve lambaları Görsel 2.82'de görüldüğü gibi konumlandırınız.



**Görsel 2.82:** Lambaların sahne üzerindeki yerleşimleri

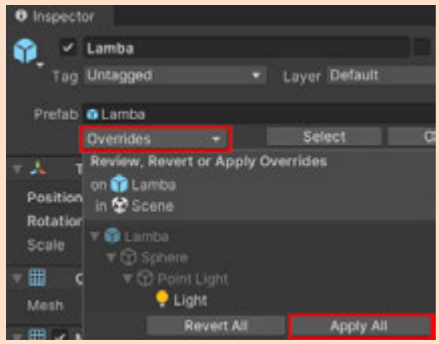
**11. Adım: Hierarchy** penceresinde **Lamba** prefab yapılarından herhangi birinde **Sphere**> **Point Light** alt bileşenini seçiniz. **Inspector** penceresinde ışığın rengini kırmızıyla değiştiriniz ve diğer ayarları Görsel 2.83'te görüldüğü gibi yapınız. Bu adımda diğer lambaların bu işlemden etkilenmediğini gözlemleyiniz.




**Görsel 2.83:** Point Light ayarları

NOT

**Hierachy** penceresinde bir prefab yapısının herhangi bir örneğinde yapılan değişikliğin tüm prefab yapısına uygulanması için **Inspector** penceresinde Görsel 2.84'te görüldüğü gibi **Overrides**> **Apply All** düğmesine tıklanması gereklidir. Aksi takdirde değişiklik sadece üzerinde çalışılan örnek üzerinde geçerli olacaktır ve prefab yapısı bu değişiklikten etkilenmeyecektir. Prefab yapısında genel bir değişiklik yapılarak bu değişikliğin ilgili prefab yapısından türetilen tüm örneklere uygulanması isteniyorsa **Project** penceresinden prefab yapısı seçilerek **Inspector** penceresinde **Open** düğmesine tıklanması veya doğrudan prefab yapısına çift tıklanması sonrası işlem yapılması gereklidir.




**Görsel 2.84:** Değişiklikleri aynı türdeki tüm prefab yapısına uygulama

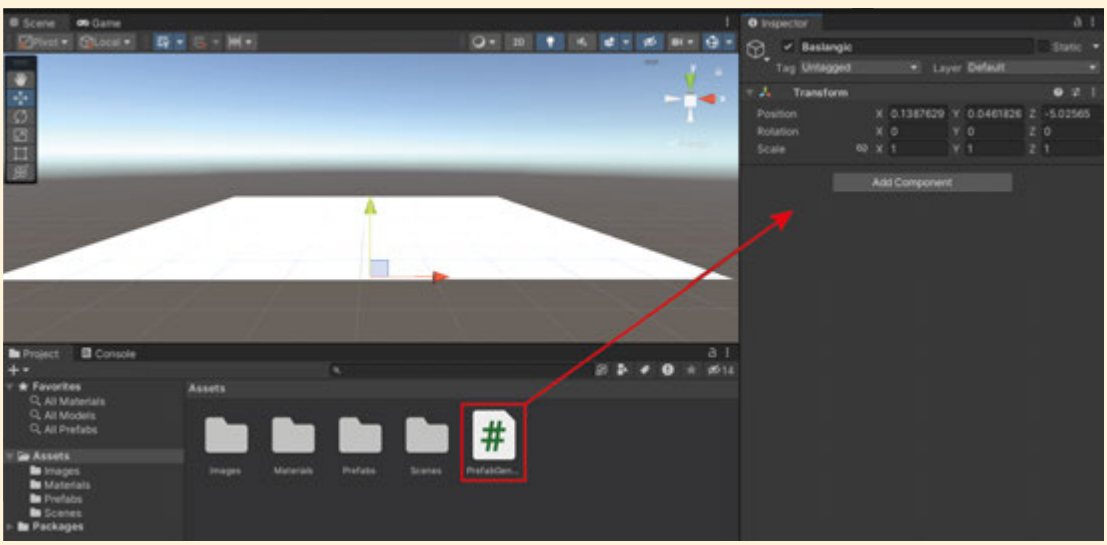
**12. Adım:** Yaptığınız değişikliklerin tüm prefab yapısında geçerli olması için **Inspector** penceresinde **Overrides**> **Apply All** düğmesine tıklayınız. Bu adımdan sonra tüm lambaların kırmızı ışığa dönüştüğünü gözlemleyiniz. Çalışmanızı kaydediniz.

NOT

Prefab yapıları kodla da oluşturulabilir. Bu da çalışma zamanı boyunca neredeyse sonsuz sayıda nesne oluşturulmasına olanak tanır.

**13. Adım:** Sahnedeki tüm prefab yapısı örneklerini siliniz. **Project** penceresinde **Lamba** prefab yapısını seçiniz ve **Inspector** penceresinde **(X= -1, Y=1, Z=-5)** noktasına konumlandırınız.

**14. Adım:** Sahneye boş bir **GameObject** ekleyiniz. Oyun nesnesini **Baslangic** olarak isimlendiriniz ve **(X=1, Y=1, Z=– 5)** konumuna getiriniz.
**15. Adım:** Projenizde **PrefabUret** isminde bir script dosyası oluşturunuz. Komut dosyasını **Baslangic** ismindeki nesneye ekleyiniz (Görsel 2.85).




**Görsel 2.85:** Komut dosyasını nesneye ekleme

**PrefabUret** ismindeki dosyanızı kod editöründe açarak aşağıdaki kodları yazınız.

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

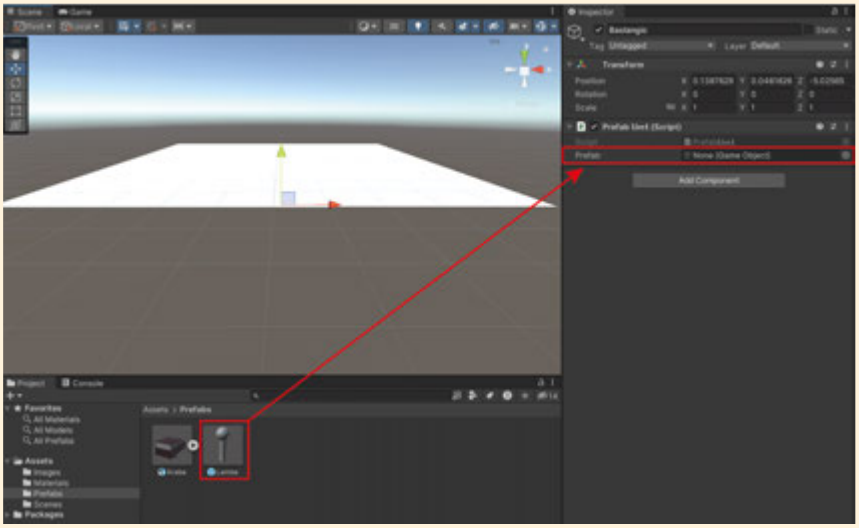
public class PrefabUret : MonoBehaviour
{
  // Oluşturulacak prefab nesnesi
  public GameObject prefab;

  void Start()
  {
    // Başlangıçta bir şey yapmıyoruz, bu yüzden bu metod boş.
  }
  void Update()
  {
    // Prefab'ın oluşturulacağı rastgele bir konum belirleme
    Vector3 pos = new Vector3(Random.Range(-5, 5), 1, Random.Range(-5, 5));

    // «B» tuşuna basıldığında
    if (Input.GetKeyDown(KeyCode.B))
    {
      // Prefab'ın bir kopyasını sahneye ekleme
      Instantiate(prefab);
    }

    // «Space» tuşuna basıldığında
    if (Input.GetKeyDown(KeyCode.Space))
    {
      // Rastgele bir konumda ve mevcut nesnenin rotasyonunda prefab'ın bir
      // kopyasını sahneye ekleme
      Instantiate(prefab, pos, transform.rotation);
    }
  }
}
```

**16. Adım: Baslangic** nesnesi seçiliyken **Project** alanındaki **Lamba** prefab yapısını **Inspector** alanındaki **Prefab Uret** bileşeninin **Prefab** özelliğine sürükleyiniz ve projenizi oynatınız (Görsel 2.86). **B** tuşuna basılarak oluşturulan prefab yapıları ile **boşluk** tuşuna basılarak oluşturulan prefab yapıları arasındaki farkı gözlemleyiniz.



**Görsel 2.86:** Prefab yapısını Prefab Uret bileşeni ile ilişkilendirme

SIRA SİZDE

Üç boyutlu modelleme programında herhangi bir şekil kullanarak bir adet tekrar kullanılabilir nesne oluşturunuz. Oluşturmuş olduğunuz bu nesneyi sahneye eklemek için klavyeden bir tuş tanımlayınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Üç boyutlu modelleme programında doğru seçim yaparak yeni bir belge oluşturdu. | | |
| 2. Sahneye bir "Plane" nesnesi ekleyerek konumunu ayarladı. | | |
| 3. Uygulama araçlarını kullanarak çalışma düzleminde tekrar kullanılabilir nesne için bir şekil oluşturdu. | | |
| 4. Oluşturduğu şekli uygun şekilde ölçeklendirip, konumlandırdı. | | |
| 5. Oluşturduğu şekle "Rigidbody" bileşenini ekledi | | |
| 6. Projesindeki uygun bir klasöre yeni bir "prefab" oluşturarak daha önce oluşturmuş olduğu şekli bu prefab ile ilişkilendirdi. | | |
| 7. Yeni bir script dosyası oluşturarak tekrar kullanılabilir nesnenin klavye ile kontrolü için gerekli kodları yazdı. | | |
| 8. Projeye yeni bir "GameObject" ekledi ve bu nesneyi daha önce oluşturduğu tekrar kullanılabilir nesne ve kod dosyası ile ilişkilendirdi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## ÖLÇME VE DEĞERLENDİRME

**Aşağıdaki soruları dikkatlice okuyarak doğru cevabı işaretleyiniz.**

**1. Oyun senaryosu, oyun geliştirme sürecinde hangi aşamalarda kullanılır?**

A) Oyunun piyasaya sürülmesinden sonra

B) Yalnızca fikir aşamasında

C) Yalnızca karakter tanımlamalarında

D) Yalnızca oyunun test aşamalarında

E) Yaratıcı fikir aşamasından başlayarak geliştirme sürecinin tümünde

**2. Aşağıdaki dosya türlerinden hangisi saydam arka planlı resimler için kullanılır?**

A) BMP B) GIF C) JPG

D) PNG E) TIFF

**3. Hareketli resimler için kullanılan dosya türü aşağıdakilerden hangisidir?**

A) GIF B) PNG C) PSD

D) SVG E) TGA

**4. Resim düzenleme programında birden fazla resmi aynı çalışma dosyasına farklı katmanlar hâlinde yüklemek için aşağıdakilerden hangisi seçilmelidir?**

A) Dosya> Dışa Aktar> Katmanlardan Dosyalara

B) Dosya> İçe Aktar> Notlar

C) Dosya> Komut Dosyaları> Dosyaları Yığına Yükle

D) Dosya> Oluştur> Görüntü Varlıkları

E) Dosya> Otomatikleştir> Toplu İş

**5. Üç boyutlu modelleme programında nesnenin yüzeyi ile ilgili işlem yapabilmek için aşağıdakilerden hangisinin seçilmesi gereklidir?**

A) Edge Select B) Face Select C) Mesh Select

D) Point Select E) Vertex Select

**6. Resim düzenleme programında tasarladığımız görüntüyü Desen olarak tanımlamak için aşağıdakilerden hangisi seçilmelidir?**

A) Dosya> Desen Aktar

B) Dosya> Desenlere Gözat

C) Dosya> Oluştur> Desen

D) Düzenle> Desen Çiz

E) Düzenle> Desen Tanımla

**7. Aşağıdakilerden hangisi resim düzenleme programında görüntünün saydamlığını artırmak veya azaltmak için kullandığımız değerin adıdır?**

A) Akıllı Nesne

B) Bulanıklık

C) Canlılık

D) Dolgu

E) Düzeyler

**8. Performans açısından avantaj sağlayan, az detay içeren ve daha küçük dosya boyutuna sahip modelleme türü aşağıdakilerden hangisidir?**

A) Düşük Poligon

B) Düşük Vertex

C) Düşük Wireframe

D) Yüksek Poligon

E) Yüksek Wireframe

**9. Oyun motorunda dışarıdan içe aktarılan dosyalar hangi klasör altına eklenir?**

A) Assets B) Logs C) Packages

D) Temp E) UserSettings

**10. Oyun motorunda, benzer özelliklere sahip birden fazla nesnenin hızlıca oluşturulmasını, düzenlenmesini ve güncellenmesini sağlayan yapı aşağıdakilerden hangisidir?**

A) Collider B) Prefab C) Rigidbody

D) Scene E) Shader

# 3. ÖĞRENME BİRİMİ



# Oyunda Animasyon

## KONULAR

3.1. OYUNDA ANİMASYON
3.2. SİMÜLASYON İŞLEMLERİ
3.3. KARAKTER VE ÇEVRE OLUŞTURMA
3.4. SES EKLEME İŞLEMLERİ

## NELER ÖĞRENECEKSİNİZ?

- Tanıtım veya hikâye ile ilgili sinematik ekleme işlemini oyun içine yapma
- Tekrarlı olan animasyonları ekleme, kodlama ile animasyonu kontrol etme işlemlerini yapma
- Karakterin ve nesnelerin hareketlendirilmesi işlemlerini yapma
- Çeşitli etkileri içeren uygulamaların tasarlama işlemini yapma
- Çevre oluşturma işlemini yapma, ışık türleri ve etkileri kavramlarını açıklama
- Birden fazla kamera kullanma ve kuş bakışı harita ekleme işlemini yapma
- Gerçek zamanlı yansımalar, bitki, hayvan ve diğer doğa nesnelerini ekleme işlemini yapma
- Diğer oyuncular ile iletişim ve diyalog pencereleri oluşturma işlemini yapma
- Zamanın ilerletilmesi ve şehir nesneleri ekleme işlemini yapma
- Kullanılabilir çeşitli araçların eklenmesi işlemini yapma
- Ses dosyaları türlerini açıklama ve ses dosyalarının dönüşüm işlemini yapma
- Etkileşimde ses etkilerini ve arka plana müzik ekleme işlemini yapma

## ANAHTAR KELİMELER

animasyon, canlandırma, efektler, etkileşim, hareketlendirme, ses etkileri, simülasyon, sinematik

HAZIRLIK ÇALIŞMASI

Daha önce oynamış olduğunuz bir oyundaki ses efektleri ve animasyonlarının oyun tasarımına olan olumlu ve olumsuz etkileri ile ilgili düşüncelerinizi sınıfta paylaşınız.

## 3.1. OYUNDA ANİMASYON

Oyun içinde gereksinim duyulan hareketlendirme işlemleri yapılabilir. Animasyon işlemleri; Cinemachine, Tween, kod yazılarak ve Animator ile oluşturulabilir.

### 3.1.1. Sinematik Ekleme

Oyun veya bir hikâye hakkında bilgi vermek için kullanılan film türüne **sinematik** denir. Sinematik, önceden hazırlanmış bir video veya oyun motoru üzerinde kaydedilmiş bir sahneyi içerir. Hazırlık aşamasında, hikâyenin anlatılacağı bir storyboard (hikâye panosu) oluşturulur. Bu panoda, animasyonla canlandırılacak nesneler kabaca sıralanır ve kameranın odaklanacağı anlar belirlenir.

Gerçek zamanlı animasyonlu hikâye anlatımı denen diğer yöntemle oyun motorunda kaydedilmiş bir sahne sinematik yapılır. Sanal kamera, ışık ve görsel efektlerle zaman çizgisi üzerinde animasyonlar oluşturulur.

Oyun motorunda sinematik oluşturma aracı olan **Cinemachine** ile çekimler oluşturulur. Daha sonra bu çekimler zaman çizelgesinde sıralanır, düzenlenir ve harmanlanır. Son aşamada ise alt yazı, müzik, ses efektleri ve seslendirme çalışmaları tamamlanır. Oyundaki tercihlere göre seçilen dil kullanılarak akışa yazı ve ses eklenir.

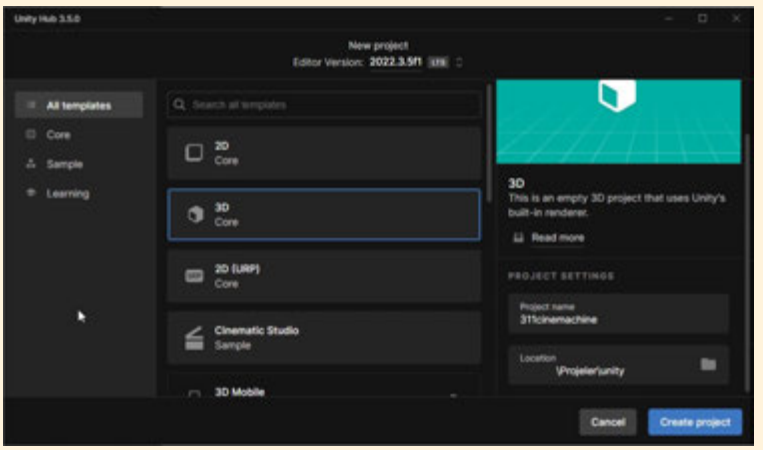
NOT

Oluşturulan **video**, kalitesine bağlı olarak oyunun boyutunu artırabilir. Ayrıca videoyu düzenlemek için bir video editör kullanmak gerekebilir.

1. UYGULAMA

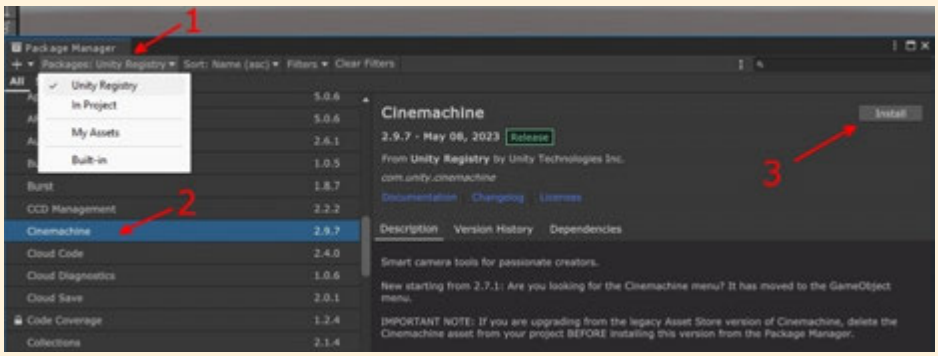
İşlem adımlarına göre Cinemachine eklentisini kullanarak sinematik yapınız.

**1. Adım:** Görsel 3.1'de görüldüğü gibi yeni bir 3D Core projesi oluşturunuz.



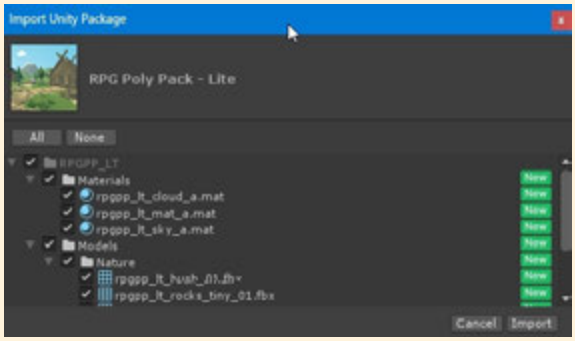

**Görsel 3.1:** Yeni proje oluşturma

**2. Adım:** Görsel 3.2'de görüldüğü gibi Window>Package Manager ile gelen pencerede oyun motorunda kayıtlı olan Cinemachine'i seçiniz ve kurunuz. Kurulum sonunda Component menüsünde Cinemachine seçeneğinin eklenmiş olduğunu görünüz.




**Görsel 3.2:** Package Manager ile eklenti kurulumu

**3. Adım:** Görsel 3.3'te görüldüğü gibi sahnenin boş kalmaması için Asset Store'u kullanarak **RPG Poly Pack - Lite** paketini indiriniz.
**4. Adım:** Görsel 3.4'te görüldüğü gibi **rpgpp_lt_scene_1.0** isimli sahneyi açınız.

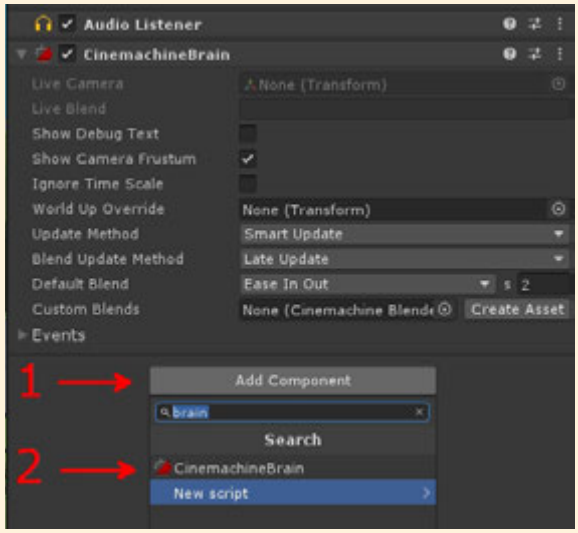



**Görsel 3.3:** Örnek bir sahne kurulması



**Görsel 3.4:** Örnek sahnenin açılması

**5. Adım:** Görsel 3.5'te görüldüğü gibi MainCamera'yı seçiniz ve Add Component>CinemachineBrain bileşenini ekleyiniz.

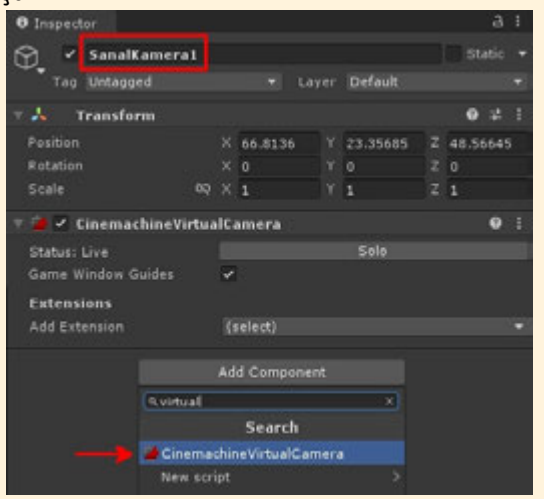



**Görsel 3.5:** CinemachineBrain bileşeninin eklenmesi

NOT

CinemachineBrain bileşeni kameranın bazı özelliklerini kendi kontrolüne alarak **sanal kameraları** yönetir.

**6. Adım:** Görsel 3.6'da görüldüğü gibi Game Object>Create Empty menüsü ile boş bir nesne oluşturunuz ve bu nesnenin içine Cinemachine>Virtual Camera ekleyiniz. Sanal kameranın adını SanalKamera1 olarak değiştiriniz.




**Görsel 3.6:** CinemachineVirtualCamera bileşeninin eklenmesi

NOT

- CinemachineBrain bileşeni otomatik olarak eklenen sanal kameraları takip etmeye başlar. Hangi kamera aktif olursa onun özelliklerini kontrol eder. CinemachineBrain içinde Live Camera özelliğinde hangi kameranın aktif olduğu görülür.
- Ana kameranın Field of View gibi bazı özellikleri değiştirilemez durumdadır. Değiştirilmek istenen özellikler, yeni eklenen nesnedeki CinemachineVirtualCamera bileşenindeki LensVertical FOV bölümünde yapılır.

**7. Adım:** İki kamera arasında yumuşak geçişin (Blend-harmanlama, kaynaştırma) sağlanması için 6. adımda yapılan işlemi tekrarlayarak SanalKamera2 isminde kamera nesnesi oluşturunuz. Görsel 3.7'de görüldüğü gibi SanalKamera2'nin yerini değiştiriniz.



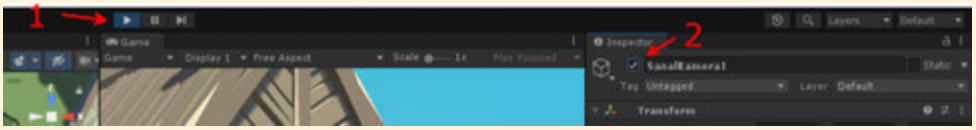

**Görsel 3.7:** SanalKamera2'nin yerini değiştirme

NOT

Önceden oluşturduğunuz bir nesneyi tekrar kullanmak için SanalKamera1'i seçip Ctrl+D (Duplicate) ile çoğaltarak adını değiştirmek diğer bir yöntemdir.

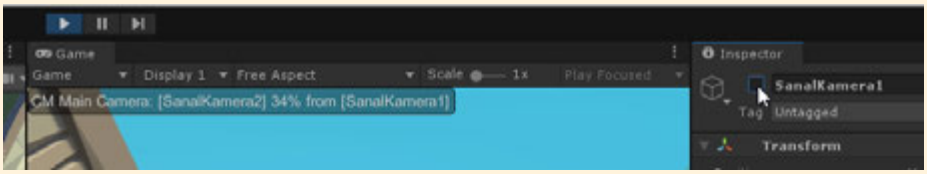
**8. Adım:** Kamera geçişini denemek için Ctrl+P kısa yolu ile oyunu başlatınız. İlk kameramız aktif olacaktır.

Görsel 3.8'de görüldüğü gibi SanalKamera1'in aktifliği kapatılıp açıldığında, diğer kamera ile yumuşak bir geçiş olduğu görülecektir.



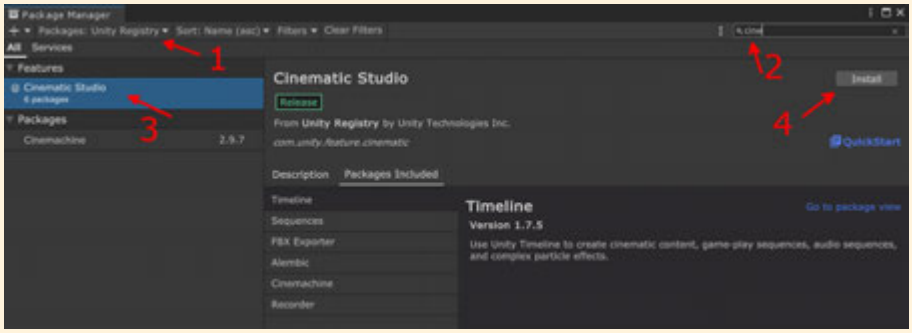

**Görsel 3.8:** SanalKamera1'in aktif özelliği

**9. Adım:** Hangi kamerada olunduğu ve hangisine geçiş yapıldığını görmek için Main Camera'nın CinemachineBrain bileşenindeki Show Debug Text seçeneğini aktif yapınız. Görsel 3.9'da görüldüğü gibi oyun çalıştırılınca üst köşede bilgilendirme metni çıkacaktır.




**Görsel 3.9:** Kamera geçişi ile ilgili bilgilendirme metni

**10. Adım:** Görsel 3.10'da görüldüğü gibi Cinematic Studio'yu seçerek film oluşturmak için gereken tüm paketleri kurunuz.



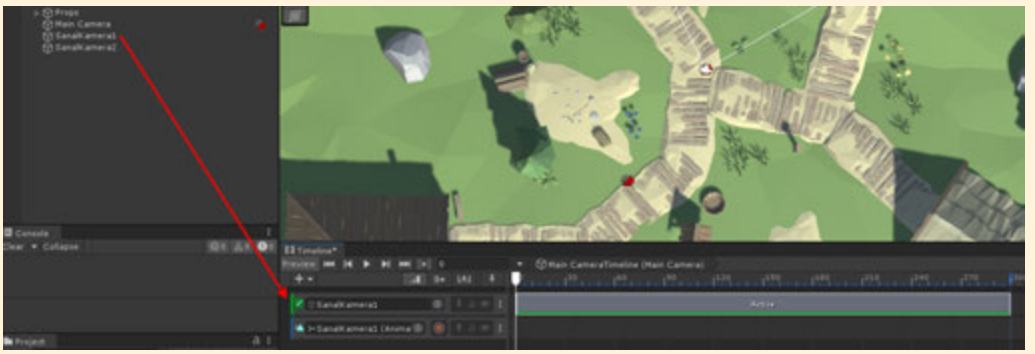

**Görsel 3.10:** Cinematic Studio paketinin kurulumu

**11. Adım:** Görsel 3.11'de görüldüğü gibi Window>Sequencing>Timeline ile zaman çizgisi penceresi açınız. MainCamera seçili iken Create komutu veriniz. Yeni *.playable uzantılı bir animasyon oluşturunuz. Timeline penceresini sahnenin altına taşıyınız.



**Görsel 3.11:** Ana kameraya animasyon yöneticisi ekleme

**12. Adım:** Görsel 3.12'de görüldüğü gibi SanalKamera1'i zaman çizgisine sürükleyerek önce Add Activation Track ve sonrasında SanalKamera1'i tekrar sürükleyerek Add Animation Track ekleyiniz.

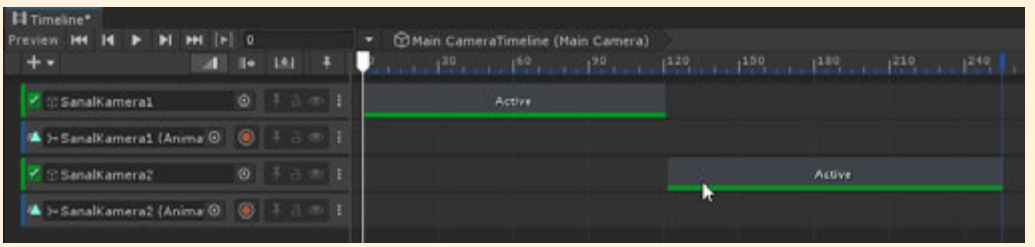


**Görsel 3.12:** Zaman çizgisine iz (track) ekleme

NOT

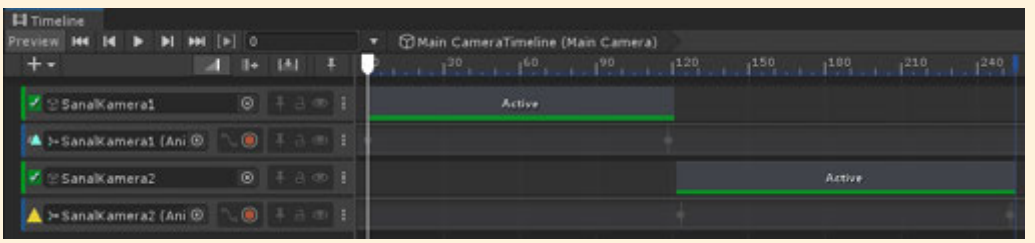
Zaman çizgisinde blok hâlindeki parçaları kenarlarından tutarak kısaltabilir, uzatabilir, kaydırabilirsiniz.

**13. Adım:** Görsel 3.13'te görüldüğü gibi bir önceki adımları tekrar ederek SanalKamera2'yi zaman çizgisine ekleyiniz.



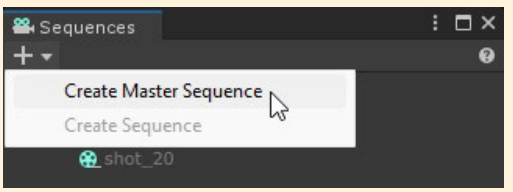

**Görsel 3.13:** Diğer kameranın eklenmesi

**14. Adım:** Görsel 3.14'te görüldüğü gibi Animasyon için eklenen izlerdeki kırmızı renkteki Start recording düğmelerini kullanarak kameraların ilerleme ve dönme gibi hareketlerini anlatılmak istenen duruma göre konumlandırılarak kaydediniz.




**Görsel 3.14:** Kameralara hareket ekleme

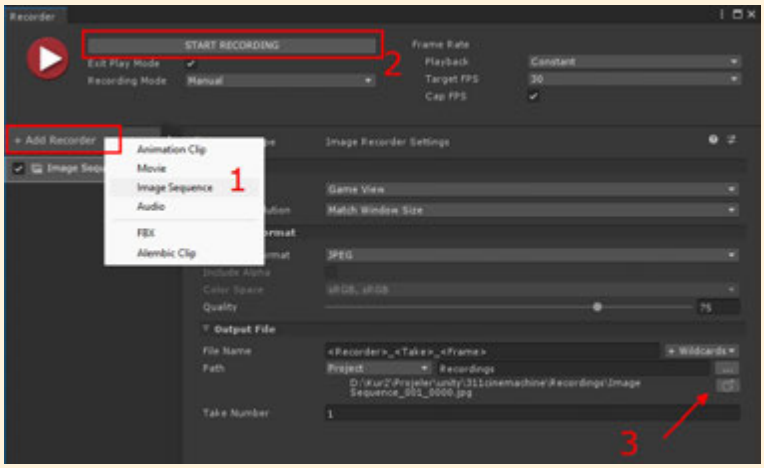
**15. Adım:** Görsel 3.15'te görüldüğü gibi Window>Sequencing>Sequences penceresini açarak Create Master Sequence komutunu veriniz.




**Görsel 3.15:** Ana kesit nesnesi oluşturma

**16. Adım:** Görsel 3.16'da görüldüğü gibi Window>General>Recorder>Recorder Window menüsü ile kaydetme penceresini açınız ve şu adımları yapınız:

- Add Recorder ile açılan menüden Image Sequence'ı seçiniz.
- Görsel 3.16'da gösterilen ayarları yaparak START RECORDING düğmesine basınız.
- Oluşan resimleri proje klasöründe Recordings altında görünüz.




**Görsel 3.16:** Sinematiğin kaydedildiği pencere

**17. Adım:** Örnek olarak oluşturulan yaklaşık 150 kare resim dosyasını bir video düzenleme programı kullanarak bir araya getiriniz. Çalışmaya müzik, alt yazı gibi özellikler ekleyip kullanıma hazır bir film elde ediniz.

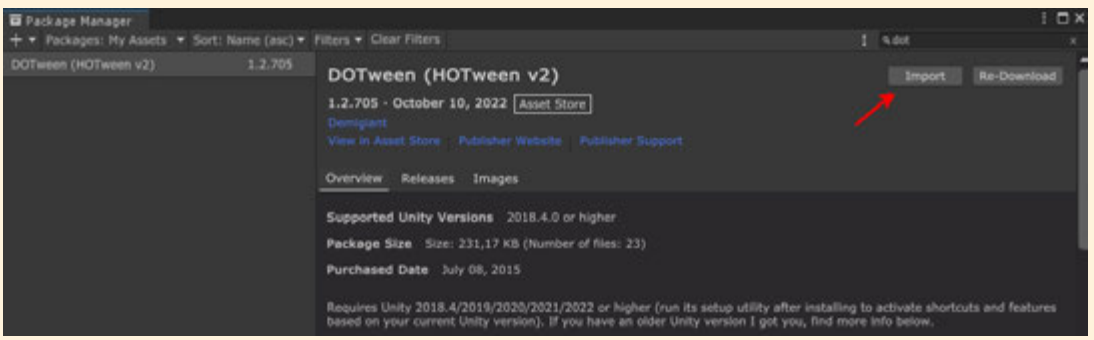
### 3.1.2. Tekrarlı Olan Animasyonları Ekleme

Havada bir bulutun, kuşun hareket etmesi veya bir balonun yukarı-aşağı hareket etmesi gibi nesnelerin bir durumdan diğerine geçişini kolaylaştıran işleme **Tween** denir. Tween; bir oyun nesnesinin konumunu, dönüşünü, boyutunu, rengini veya başka herhangi bir özelliğini canlandırmak için kullanılır.

2. UYGULAMA

İşlem adımlarına göre Tween eklentisini kullanarak animasyon oluşturunuz.

**1. Adım:** Oyun motorunda yeni bir 3D Core uygulaması oluşturunuz.
**2. Adım:** Görsel 3.17'de görüldüğü gibi Asset Store penceresinden DoTween paketini indirerek içeri aktarınız.




**Görsel 3.17:** DOTween paketinin indirilmesi

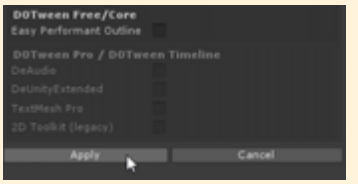
**3. Adım:** Görsel 3.18, Görsel 3.19 ve Görsel 3.20'de görüldüğü gibi Tween paketini indirerek içeri aktarınız.




**Görsel 3.18:** İlk kurulum penceresi




**Görsel 3.19:** Setup DOTween ile kurulum

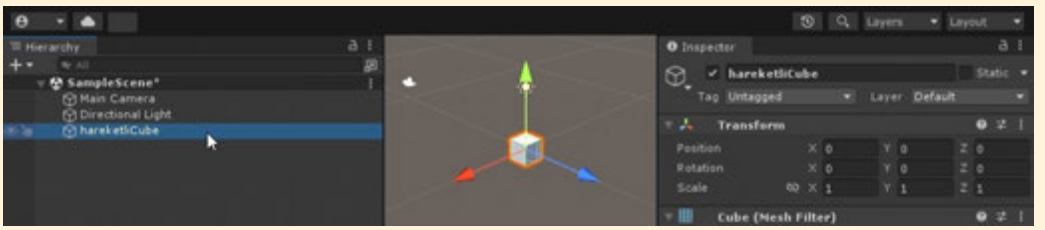



**Görsel 3.20:** Apply düğmesi ile işlemin uygulanması

NOT

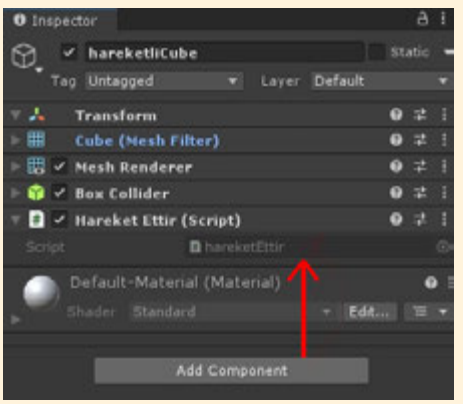
Tween'in yardımcı ayarlar penceresine Tools>Demigiant>DOTween Utility Panel ile ulaşılabilir.

**4. Adım:** Görsel 3.21'de görüldüğü gibi sahneye bir küp ekleyiniz.




**Görsel 3.21:** Küp ekleyip adını hareketliCube olarak değiştiriniz.

**5. Adım:** Görsel 3.22'de görüldüğü gibi Küp üzerine Component ekleme menüsünü kullanarak NewScript seçeneği ile hareketEttir isminde yeni bir script oluşturunuz.




**Görsel 3.22:** Küp nesnesine kod ekleme

```
using UnityEngine;
using DG.Tweening;          //Gerekli kütüphane
public class hareketEttir : MonoBehaviour
{
void Update()
 {
```

```
  if (Input.GetKeyDown(KeyCode.Space))
  {
     transform.DOMove(new Vector3(10, transform.position.y,
transform.position.z), 1);
        }
    }
}
```

**6. Adım:** Boşluk tuşuna basarak küp nesnesini sağa doğru bir saniye süresince hareket ettiriniz.
**7. Adım:** Harekete SetEase() komutu ekleyerek daha yumuşak geçişli animasyon yaptırınız. DOMove komut satırı gibi kodu değiştirerek **hareket sonunda** zıplama etkisi oluşturunuz.

```
transform.DOMove(new Vector3(10, transform.position.y, transform.
position.z), 1).SetEase(Ease.OutBounce);
```

DOTween'de kullanabileceğiniz birçok farklı Ease ayarı vardır:

- **Linear:** Hareketi baştan sona düzgün bir şekilde ivmelendirir.
- **In ile başlayanlar:** Hareketin başlangıcında ivme çeşidi uygular.
- **Out ile başlayanlar:** Hareketin sonunda ivme çeşidi uygular.
- **InOut ile başlayanlar:** Hareketin başlangıcında ve sonunda ivme çeşidi uygular.

Hareket etme dışında sık kullanılan diğer komutlar şunlardır:

- **DOMove:** X, Y ve Z boyutlarında istenen miktarda hareket eder.
- **DOScale:** X, Y ve Z boyutlarında istenen miktarda boyutlandırır.
- **DORotate:** X, Y ve Z açılarında istenen miktarda döndürür.
- **DOLookAt:** Nesnenin ön tarafını istenen yöne doğru çevirir.

**8. Adım:** Animasyonun döngü hâlinde devamlılığı istenir ise SetLoops komutu ekleyiniz.

```
     transform.DOMove(new Vector3(10, transform.position.y,
transform.position.z), 1)
            .SetEase(Ease.OutBounce)
            .SetLoops(-1, LoopType.Yoyo);
```



NOT

SetLoops parametresi -1 ise **sonsuza** kadar aynı hareketi tekrar eder. İhtiyaca göre -1 yerine adet olarak belli bir sayı yazılır, LoopType ile döngü tipi değiştirilir.

```
.SetLoops(2, LoopType.Restart) //İki defa animasyon baştan oynatılır.
```

**9. Adım:** Ek olarak ilk hareket bitince sonuna başka bir animasyon daha yapınız.

Örnek kodda nesne sağa doğru çarpıp zıpladıktan sonra iki katı büyüyecektir:

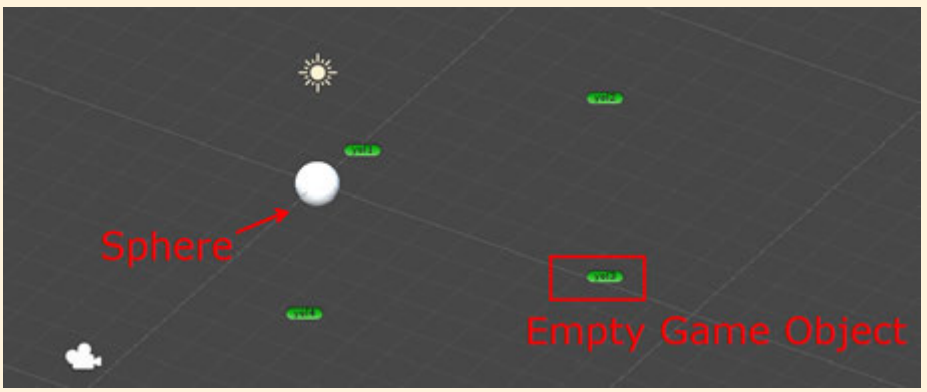
```
     transform.DOMove(new Vector3(10, transform.position.y,
transform.position.z), 1)
     .SetEase(Ease.OutBounce).OnComplete(() =>
            {
                transform.DOScale(2, 1);
            });
```

**10. Adım:** Örnek kodda oluğu gibi istenen zamanda animasyona ara verme ve devam ettirme gibi işlemleri yapınız.

```
using UnityEngine;
using DG.Tweening;              //Gerekli kütüphane

public class hareketEttir : MonoBehaviour
{
    Tween animasyon;
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {//Boşluk tuşu ile animasyon bellekte depolanır.
            animasyon = transform.DOMove(new Vector3(10,
transform.position.y, transform.position.z), 1)
             .SetEase(Ease.OutBounce)
             .SetLoops(-1, LoopType.Yoyo);
        }
        if (Input.GetKeyDown(KeyCode.D))
        {
        animasyon.Pause(); //D tuşu ile animasyona ara verilir.
        }
        if (Input.GetKeyDown(KeyCode.Y))
        {
 animasyon.Play();  //Y tuşu ile animasyona devam edilir.
        }
    }
}
```

**11. Adım:** Görsel 3.23'te görüldüğü gibi küre ve dört adet boş oyun nesnesini ekleyiniz. Boş nesnelerin özellikler penceresinde sol üst köşedeki "Select Icon" simgesi ile gösterim biçimlerini düzenleyiniz.




**Görsel 3.23:** Küre ve dört adet boş oyun nesnesi eklenir.

NOT

Düz bir yolda ilerlemek yerine nesne bir yol (**path** veya **spline**) takip edebilir.

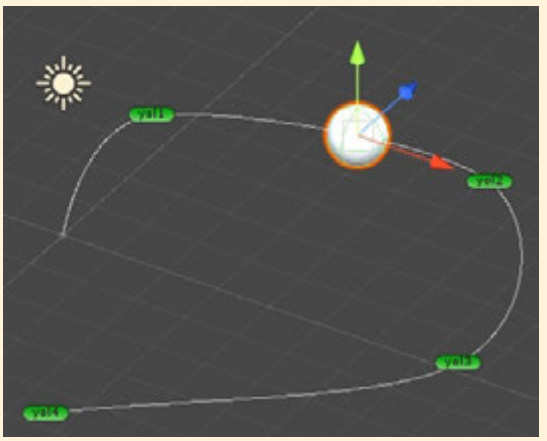
**12. Adım:** Küre üzerine patikaHareketi adında yeni bir kod ekleyiniz.

```
using UnityEngine;
using DG.Tweening;          //Gerekli kütüphane
public class patikaHareketi : MonoBehaviour
{
    public Transform[] yolNesneleri;
    public Ease yontem;
    void Start()
    {
        Vector3[] yollar;
        yollar = new Vector3[yolNesneleri.Length];
        int i = 0;
        foreach (var yolItem in yolNesneleri)
        {
 yollar.SetValue(yolItem.position, i);
            i++;
        }
        transform.DOPath(yollar, 10f, PathType.CatmullRom)
             .SetEase(yontem)
             .SetLoops(-1, LoopType.Yoyo);
    }
}
```

**13. Adım:** Görsel 3.24'te görüldüğü gibi küre nesnesinin kodundaki yolNesneleri içine yol nesnelerini sürükleyiniz.
**14. Adım:** Görsel 3.25'teki gibi proje çalıştırıldığında sıra ile dört adet yol nesnesini takip ederek ilerleyen küre görülecektir.




**Görsel 3.24:** Koddaki özellikleri ayarlama



**Görsel 3.25:** Çalıştırıldığında tasarım ekranında izlenecek yolu belirten çizgiler görünür.

NOT

DOTween tercihlerinde varsayılan olarak Draw Path Gizmos aktiftir. Projeyi çalıştırdığınızda tasarım ekranında yol çizgileri görülür.

Hareket eden nesne küre şeklinde değil de bir araç, hayvan veya insan olsaydı, patikada ilerlerken nesnenin ön tarafının da dönmesi gerekirdi. Bunun için SetLookAt() fonksiyonu eklenir. Ease yöntemi başlangıçta seçilir. PathType değerleri şunlardır:

- **Linear:** Nesne doğrusal bir yol boyunca hareket eder.
- **CatmullRom:** Kontrol noktalı eğri kullanarak daha yumuşak ve akıcı bir hareket sağlar.
- **CubicBezier:** İki kontrol noktası kullanarak eğri üzerinde hareket sağlar.

### 3.1.3. Kodlama İle Animasyonu Kontrol Etme

Oyun motorunda kod penceresinde **Update()** fonksiyonu kullanılarak herhangi bir nesnenin bir özelliğine hareketlilik katılır. Kod satırı ne kadar fazla olursa ve **Update()** komutu kullanan nesne sayısı ne kadar artarsa oyunun performansı bu oranda düşmeye başlar. Kodlama ile animasyon çok fazla esneklik sunsa da **gerçek zamanlı** olarak hesaplanan görevlerin artması oyunda takılmalara neden olur.

Bir nesnenin 10 birim ileriye hareket etmesinin sağlanması için şu kod kullanılır:

```
transform.localPosition += Vector3.forward * 10;
```

Bir nesnenin 90 derece sola dönmesinin sağlanması için şu kod kullanılır:

```
transform.localRotation = Quaternion.Euler(0, 90, 0);
```

Bir nesnenin boyutunun iki kat büyütülmesinin sağlanması için şu kod kullanılır:

```
transform.localScale *= 2;
```

local (yerel) ile başlayan komutlar nesnenin ebeveynine göre hareket etmesini sağlarken transform.position ise nesnenin oyun dünyasının koordinatlarına göre hareket etmesine neden olur.

transform kodları çalıştırılınca nesnenin özelliği **anında** değişir ve ara geçişler hissedilemez. Animasyon özelliğinin belli bir sürede uygulanması için Time sınıfı kullanılır.

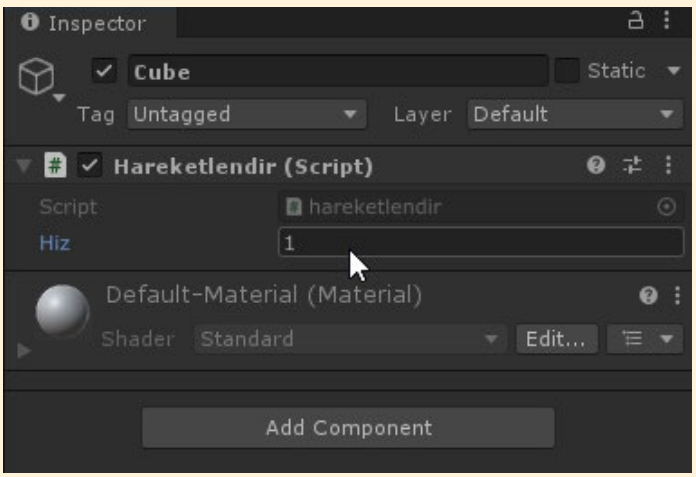
Oyun motorunda Time.deltaTime değişkeni bir oyun döngüsünde son iki kare arasındaki zaman farkını temsil eder. Bu özellik animasyonlar, fizik ve diğer etkileşimler gibi zamana duyarlı görevlerin doğru şekilde zamanlanmasını, oyunun hızını kontrol etmeyi ve nesne hareketinin her zaman **aynı hızda** çalışmasını sağlamak için kullanılır.

3. UYGULAMA

Boş bir proje açarak gerekli nesneleri ekleyip kodları uygulayınız.

**1. Adım:** Sahneye küp nesnesi ve bu nesneye de hareketlendir isminde yeni kod dosyası ekleyiniz.
**2. Adım:**Kodları düzenleyerek projeyi çalıştırınız.

```
using UnityEngine;
public class hareketlendir : MonoBehaviour
{
public float hiz = 1f; //Varsayılan hızı 1'dir.
    void Update()
    {
      //Obje belli bir hızda gider.
      this.transform.Translate(0, 0, hiz * Time.deltaTime);
      //Obje belli bir hızda döner.
      this.transform.Rotate(0, 0, hiz * Time.deltaTime);
      //Obje belli bir hızda büyür.
        this.transform.localScale += Vector3.one * hiz * Time.
deltaTime;
    }
}
```




**Görsel 3.26:** Çalışma zamanında hız değeri değiştirildiğinde transform bilgilerinin değiştiği görülür.

3.26'da görüldüğü gibi hız bilgisi negatif sayı olarak girildiğinde ters yönde hareket eder, sıfır sayı değeri girildiğinde durur. Sınır konulmadığı için nesne her zaman ilerler, döner ve büyür.

**3. Adım:** Başka bir yönteme göre nesnenin hareketlendirme kodunu düzenleyiniz.

```
using UnityEngine;
public class hareketlendir : MonoBehaviour
{
    public Vector3 hedefPozisyon = new Vector3(5f, 0f, 0f);
    public float animasyonHizi = 2f;
    private Vector3 ilkPozisyon;
    private float baslangicZamani;
    void Start()
   {
        ilkPozisyon = transform.position;
        baslangicZamani = Time.time;

    }
    void Update()
    {
```

```
float gecenSure = Time.time - baslangicZamani;
Vector3 yeniPozisyon = Vector3.Lerp(ilkPozisyon, hedefPozisyon,
gecenSure * animasyonHizi);
        transform.position = yeniPozisyon;
        if (gecenSure * animasyonHizi >= 1.0f)
        {
            //Bitince bir işlem yaptırılabilir.
        }
    }
}
```

Son kod örneği ile hareketlendirilen nesnenin **x** koordinatında 5 birim ilerlediği görülür. Hem zamana göre nesnenin hızı belirlenir hem de Lerp komutu ile daha yumuşak bir geçiş sağlanır.

SIRA SİZDE

Nesnelerin kod ile hareket ettirilmesini gerçekleştiriniz. Hareket sırasında oluşan hataları not ediniz.

**Değerlendirme**

Çalışmanız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken **Kontrol Listesi**ndeki değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

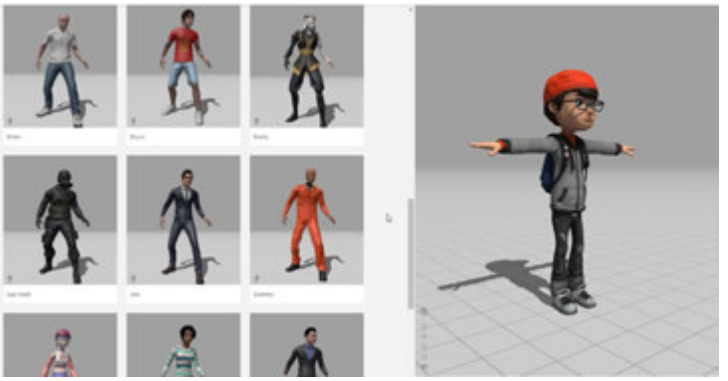
| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Oyun motoru ile boş bir proje açtı. | | |
| 2. Boş sahne içine bir oyun nesnesi ekledi. | | |
| 3. Nesnenin hareketini sağlayan program yazımını gerçekleştirdi. | | |
| 4. Oyunu çalıştırdığında hareketin uygun olup olmadığını test etti. | | |
| 5. Oluşan hataları gidererek projenin çalışmasını sağladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 3.1.4. Karakterin Hareketlendirilmesi ve Etkileşime Girmesi

Oyuncunun kontrol ettiği karakter (**Player**) nesnesinin boşta durma, yürüme, koşma, dönme, eğilme, zıplama gibi eylemlerinin birbirine sorunsuz ve pürüzsüz bir şekilde geçiş yapabilmesi önemlidir. Aynı şekilde oyun dünyasındaki diğer varlıklar, hayvanlar, robotlar ve benzeri karakterlerin de kendi hareketleri sırasında sorun yaşamadan geçiş yapmaları sağlandığında gerçekçi ve doğal bir oyun deneyimi oluşturulur.

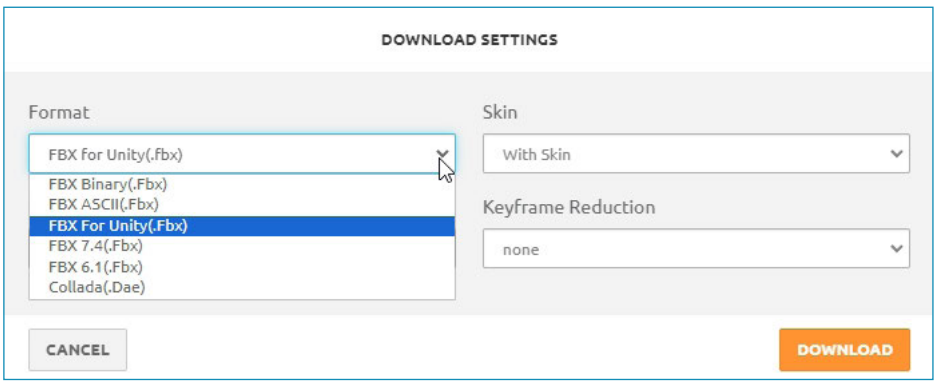
Oyun motorunda, bir karakterin hareketini kontrol etmek ve etkileşime girmesini sağlamak için ilk adım olarak karakterin 3D modeli oluşturulur. Daha sonra karakterin hareketlerini yönetmek için animasyonlar eklenir.

Görsel 3.27'de görüldüğü gibi web tabanlı servis kullanılarak hazırlanmış karakterler ve animasyonlar arasından seçim yapılır.



**Görsel 3.27:** Karakterler ve animasyonlarının listesi

Görsel 3.28'de görüldüğü gibi dosya boyutlarını küçültmek amacıyla ilk indirilen FBX için **Skin>With Skin**, diğer FBX dosyalarda ise **Skin>Without Skin** seçilir. Böylece son indirilen dosyalarda sadece animasyon kısmı inmiş olur.




**Görsel 3.28:** İndirme seçenekleri

## 4. UYGULAMA

Boş bir proje açarak gerekli nesneleri projeye ekleyiniz ve kodları burada uygulayınız.

**1. Adım:** Görsel 3.29'da görüldüğü gibi önce bir karakter ve sonra karaktere ait dört adet animasyon seçerek FBX dosyalarını indiriniz.



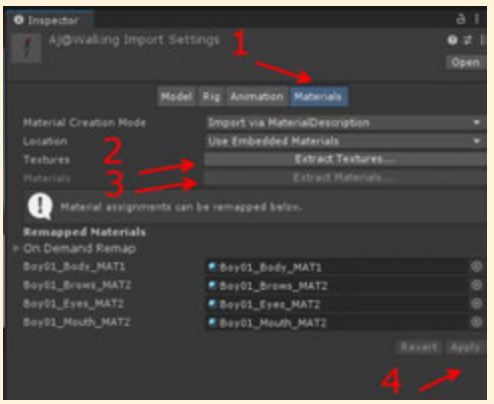

**Görsel 3.29:** Idle, Jumping, Slow Run ve Walking animasyonları indiriniz.

**2. Adım:** Görsel 3.30'da görüldüğü gibi proje içinde bir klasör açarak klasörün içine bu dosyaları kopyalayınız.

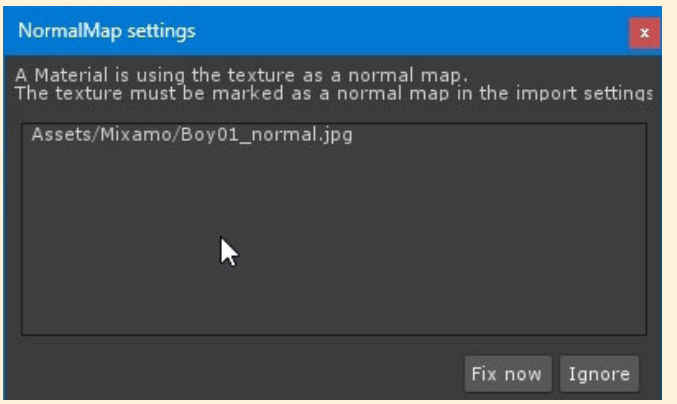



**Görsel 3.30:** Dosyalar sürükle bırak ile proje içine kopyalanır.

**3. Adım:** Görsel 3.31 ve Görsel 3.32'de görüldüğü gibi, indirilen ilk dosyayı seçerek Inspector>Materials sekmesini tıklayınız. Extract Textures yani doku çıkarma işlemini belirleyeceğiniz ya da yeni oluşturacağınız klasörleri seçerek yapınız. Diğer üç hareket için bu işlemin tekrar edilmesine gerek yoktur.

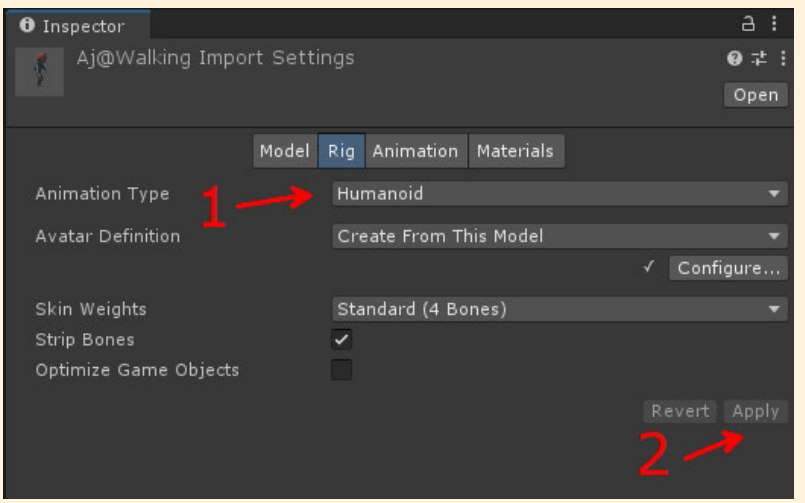



**Görsel 3.31:** Klasör içine doku ve materyallerin çıkarılması



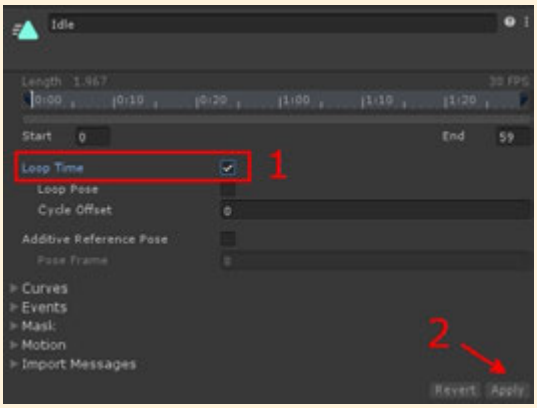

**Görsel 3.32:** Materyal ile ilgili uyarıya Fix now komutu verilir.

**4. Adım:** Görsel 3.33'te görüldüğü gibi dört FBX nesnelerinin Rig kısmından Animation Type olarak Humanoid'i seçiniz.




**Görsel 3.33:** Animasyon tipinin seçilmesi

**5. Adım:** Görsel 3.34'te görüldüğü gibi her birinde Animation sekmesine gelip Loop Time özelliğini seçili hâle getiriniz. Sadece Jumping bir kere zıplanabilir olduğu için döngü hâlinde değildir.




**Görsel 3.34:** Nesnelerin Loop Time seçeneği aktif edilir.

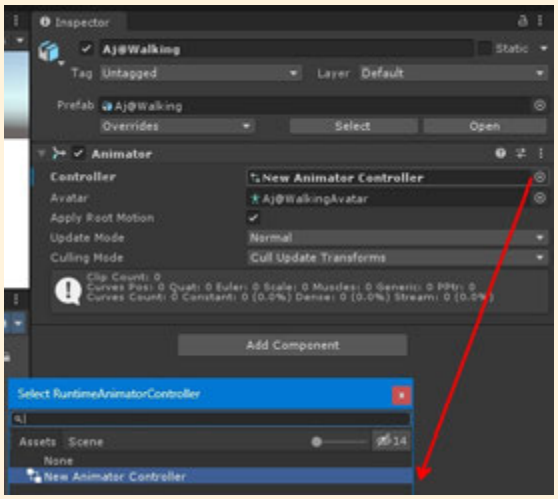
**6. Adım:** Görsel 3.35'te görüldüğü gibi sahneye bir Plane ekleyiniz ve onun boyut bilgilerine 2 değerini giriniz. Sahneye dört karakter eklediğinizde karakterlerin hareket etmediklerini görünüz.



**Görsel 3.35:** Varsayılan hâli ile karakterlerin görünümü

**7. Adım:** Animasyonlar klasörüne Assets>Create>Animator Controller ekleyiniz. Animasyonları kontrol etmek için **Animator Controller** oluşturunuz. Hareketler arasında geçiş organizasyonunu sağlayan **beyin** kısmı bu nesnedir.

**8. Adım:** Görsel 3.36'da görüldüğü gibi yeni Animator Controller bileşenini sahnedeki yürüme animasyonlu karaktere bağlayınız.

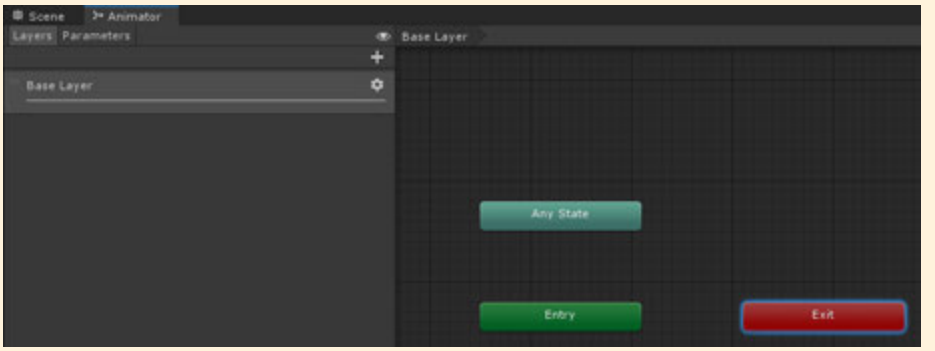



**Görsel 3.36:** Yeni oluşturulan kontrolcü seçilir.

NOT

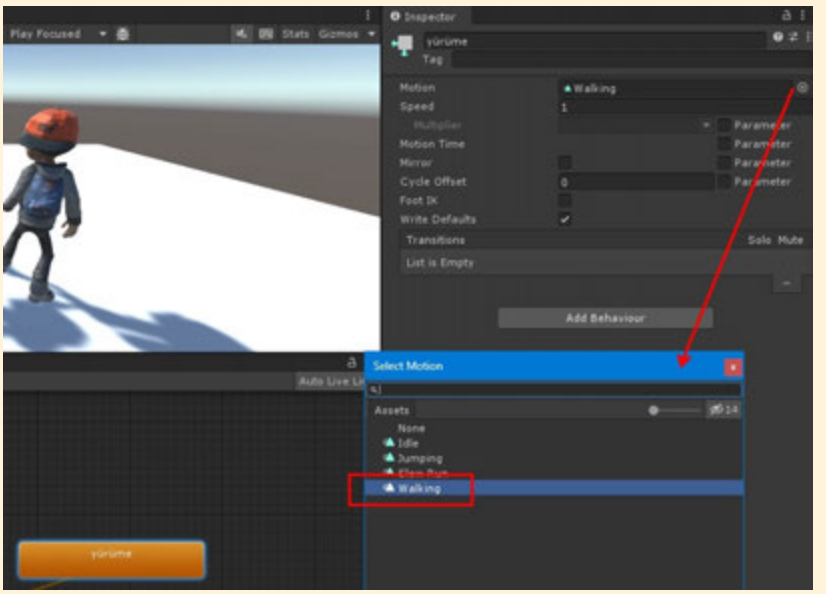
Idle, Jumping ve Slow Run karakterlerine de aynı adımları (7, 8 ve 9. adımlar) uyguladığınızda hareket eder. Yürüyen karakter dışındakiler sadece test amaçlı eklenmiştir.

**9. Adım:** Görsel 3.37'de görüldüğü gibi Assets klasöründe bulunan **New Animator Controller** nesnesini çift tıklayınız. Boş Animator penceresi açılacaktır.




**Görsel 3.37:** Animator penceresi

**10. Adım:** Görsel 3.38'de görüldüğü gibi Base Layer yazan koyu gri alana sağ tıklayıp Create State>Empty'yi seçiniz, oluşan "New State"'in yani yeni durumun Motion özelliği olarak Walking'i seçiniz.




**Görsel 3.38:** Eklenen boş durumun adı yürüme olarak değiştirilir ve hareket (Motion) seçilir.

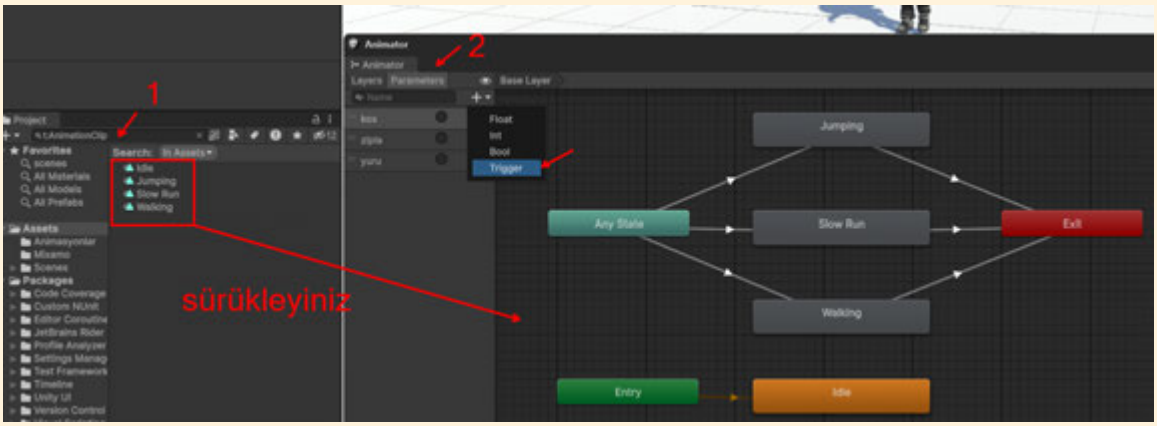
Proje çalıştırıldığında olduğu yerde yürüyen ama fiziksel olarak ilerlemeyen bir karakter vardır. Bundan sonraki hedef bir tuş basılmadığında boşta duran, Space-boşluk tuşu ile zıplayan ve Shift tuşu basılı iken koşan bir karakter tasarlamaktır.

Animator penceresindeki renkli kutuların anlamları şunlardır:

- **Any State:** Animasyon herhangi bir durumdayken çalışır.
- **Entry:** Animasyon bir duruma girdiğinde çalışır.
- **Exit:** Animasyon bir durumdan çıktığında çalışır.

**11. Adım:** Görsel 3.39'da görüldüğü gibi sahnede yürüyen çocuk dışındaki karakterleri siliniz. Yürüyen çocuğun hareketlerini aşağıda listelenen adımlarda görüldüğü gibi düzenleyiniz

- Proje penceresinde **Search by Type** olarak **Animation Clip** türünü seçiniz.
- Bulunan dört hareketi Animator penceresine sürükleyiniz.
- Yeşil renkteki **Entry** kutusunun bağlı olduğu **Idle** animasyonuna sağ tıklayıp Set as Layer Default State komutunu seçiniz.
- **Any State** ve **Exit** arasındaki animasyonları tek tek sağ tıklayıp Make Transition komutu ile bağlantı kurmak için birleştiriniz.

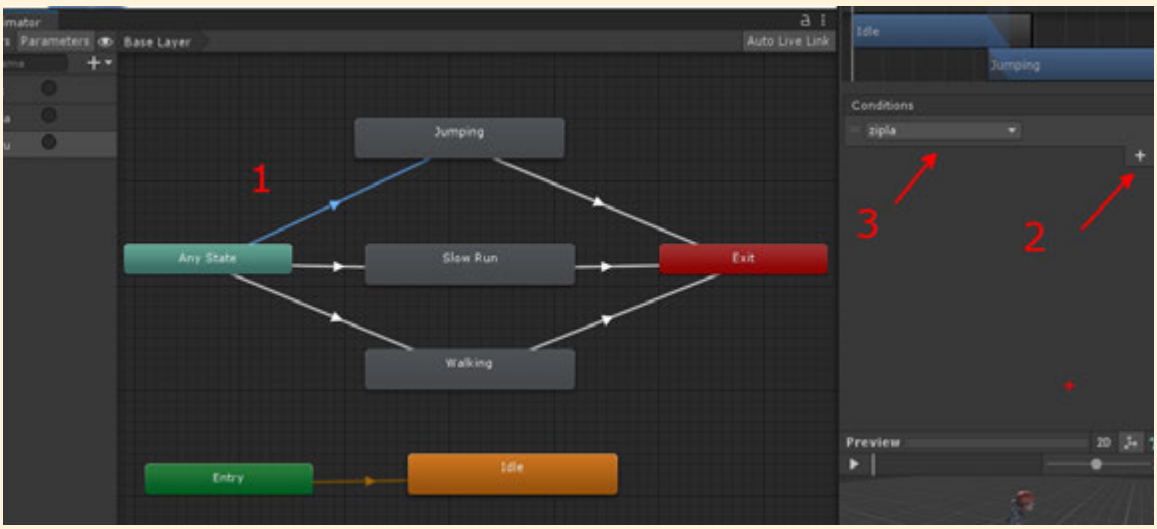



**Görsel 3.39:** Animasyon kliplerini Animator penceresine atıp üç adet Trigger eklenir.

Animator nesnesinde kullanılan parametre türleri şunlardır:

- **Float:** Kayar noktalı bir değere göre çalışan animasyon içindir.
- **Int:** Tam sayı bir değere göre çalışan animasyon içindir.
- **Bool:** Doğru veya yanlış değerine göre çalışan animasyon içindir.
- **Trigger:** Tetiklenerek çalışan animasyon içindir.

**12. Adım:** Görsel 3.40'ta görüldüğü gibi 11. adımdaki tüm bağlantıları yaptıktan sonra Any State ile diğer üç hareket arasındaki geçiş oklarına teker teker tıklayıp kos, zipla ve yuru şartlarını ekleyiniz.




**Görsel 3.40:** Jumping, Slow Run ve Walking animasyonlarının tetiklenme şartları eklenir.

NOT

"kos yerine koş, zipla yerine zıpla, yuru yerine yürü" yazılması sorun çıkarmayabilir. Yine de olabildiğince değişken, fonksiyon, klasör ve dosya adları gibi yerlerde Türkçe karakterler olan "ç, ğ, ı, ö, ş, ü, Ç, Ğ, İ, Ö, Ş, Ü" kullanmayınız. Kullanıcının göreceği metinler ise Türkçe karakterler içerebilir.

**13. Adım:** Projeyi çalıştırıp, fare ile parametre listesindeki kos, zipla ve yuru tetiklemeleri tıklandığında boştaki animasyondan koşmaya, zıplamaya ve yürümeye geçildiğini görünüz.
**14. Adım:** hareketlendir kodunu karakter nesnesine ekleyerek klavye ile etkileşimi sağlayınız.

```
using UnityEngine;
public class hareketlendir : MonoBehaviour
{
    Animator animator;
    void Start()    {
        animator = GetComponent<Animator>();
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.W))
        {//W tuşu ile yürüme tetiklenir.
            animator.SetTrigger("yuru");
        }
        if (Input.GetKeyDown(KeyCode.Space))
        {//Boşluk tuşu ile zıplama tetiklenir.
            animator.SetTrigger("zipla");
        }
        if (Input.GetKeyDown(KeyCode.LeftShift))
        {//Sol shift tuşu ile koşma tetiklenir.
            animator.SetTrigger("kos");
        }
    }
}
```

Karakter, dört tür animasyon arasından amaca göre istenen durum seçilerek oynatılır fakat fiziksel olarak ileri, geri, sağa ve sola dönemez, yer yüzeyinde ilerleyemez.

### 3.1.5. Nesnelerin Hareketlendirilmesi

Oyun motorunda hazır bulunan **Animator Controller** ile herhangi bir nesnenin bir özelliğine hareketlilik katılır. Hazırlanan animasyonlara AnimationClip adı verilir. Kliplerin ne zaman başlatılacağı, ne kadar süre devam edeceği ve diğer özellikleri doğrudan kontrol edilir.

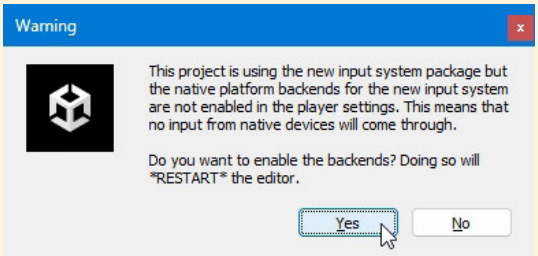
Bir kapı, çekmece veya sandığın açılması için klip hazırlanabilir. Oyuncu yakınına gelindiğinde bir tuşa basılarak açılması ve tekrar kapanması sağlanabilir. Bir lambanın yanıp sönmesi, güneşin doğup batması gibi efektler yapılabilir. Oyun motorunda karmaşık animasyonlar oluşturmak yerine bir tasarım programı kullanılarak gelişmiş animasyonlar yapılabilir. Animasyonlar proje içine *.fbx dosya türünde aktarılabilir. Örnek olarak açık kaynak program olan Blender 3D bilgisayarınızda kurulu ise *.blend dosyaları doğrudan projeye eklenerek kullanılabilir.

## 5. UYGULAMA

Boş bir proje açarak gerekli nesneleri ekleyip kodları uygulayınız.

Sahne içine, istenilen yerlere ödül nesnelerini ekleyiniz. Bunun için kendi etrafında dönen ve hafifçe aşağı yukarı hareket ederek oyuncunun dikkatini çeken bir kutu tasarlayınız. Oyuncu bu kutuya değdiğinde ödül kazanacaktır.

**1. Adım:** Starter Assets-First Person Character Controller paketini kurunuz. Görsel 3.41'de görüldüğü gibi soru sorulursa oyun motorunu yeniden başlatınız.

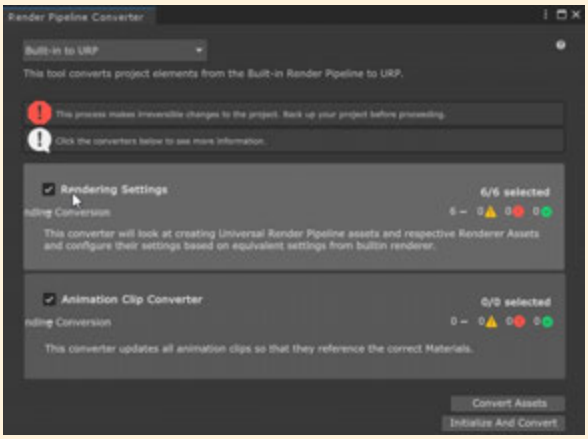



**Görsel 3.41:** Paket kurulumunda çıkan diyalog

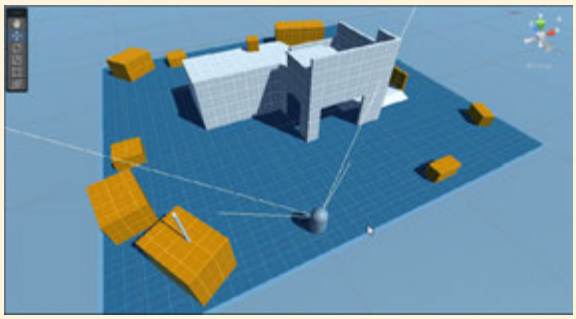
**2. Adım:** Playground ismindeki sahneyi açınız. Dokular pembe renk olarak geldi ise Görsel 3.42'de görüldüğü gibi **Window>Rendering>Render Pipeline Converter** işlemini yapınız. Sonuç, Görsel 3.43'te görüldüğü gibi olmalıdır.

### NOT

Oyunu test ettiğinizde fare ve klavye ile sahne içinde gezinebildiğinizi görebilirsiniz. Oyun çalıştırıldığında fare görünmez olabilir, ESC-Escape tuşu ile fareyi tekrar görünür yapabilirsiniz.



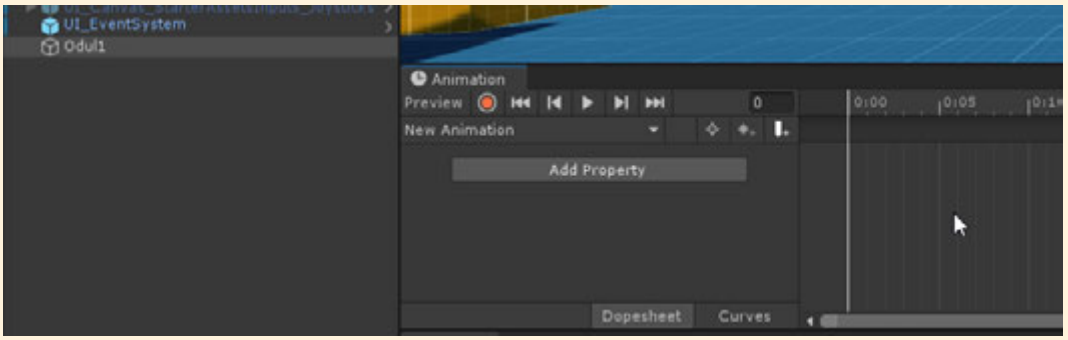

**Görsel 3.42:** Convert Assets ile materyal dönüşümü sağlanır.



**Görsel 3.43:** Örnek sahne kullanıma hazırdır.

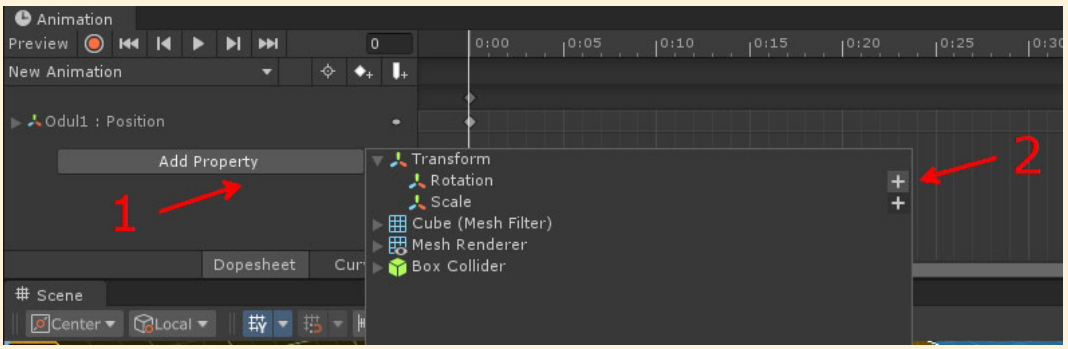
**3. Adım:** Game Object>3D Object>Cube ekleyiniz. İsmini Odul1 olarak değiştiriniz. Yer ve büyüklüğünü ayarlayınız.

**4. Adım:** Görsel 3.44'te görüldüğü gibi ödül nesnesi seçili iken Window>Animation>Animation komutu ile açılan pencerede Create düğmesine basarak *.anim dosya türünde bir animasyon dosyası oluşturunuz.




**Görsel 3.44:** Ödül nesnesine animasyon eklemek

**5. Adım:** Görsel 3.45'te görüldüğü gibi Add Property ile Transform içindeki **Position** ve **Rotation** bilgilerini ekleyiniz.




**Görsel 3.45:** Animasyonu yapılacak özelliklerin eklenmesi

**6. Adım:** Görsel 3.46'da görüldüğü gibi "Position" seçili iken zaman çizgisinde 0:30 numaralı zamana tıklayıp Add keyframe yapınız ve küp nesnesini biraz yukarıya çıkarınız.



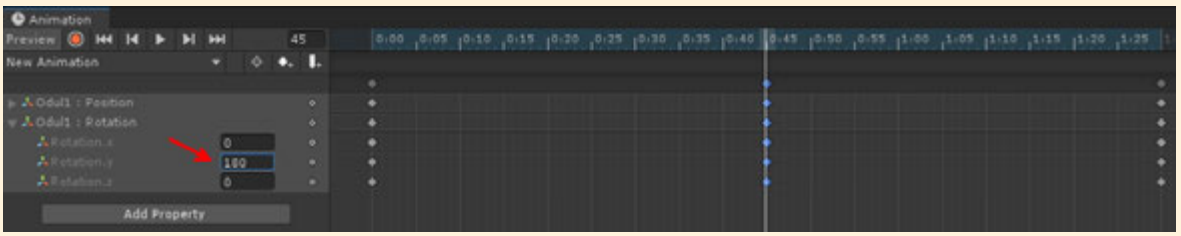

**Görsel 3.46:** 0:30 numaralı zamana anahtar kare eklenmesi

NOT

İki yöntem ile animasyon eklenebilir, kırmızı kaydetme düğmesi ile kaydedilebilir veya elle kare eklenerek animasyon yapılabilir. Anahtar kareler **elmas** şeklindedir ve tıklanarak yer değiştirme, silme gibi işlemler gerçekleştirilebilir.

**7. Adım:** Test etmek için oyunu başlatınız veya tasarımda iken denemek için Play the animation clip düğmesine tıklayınız.

**8. Adım:** Görsel 3.47'de görüldüğü gibi dönme işlemi ekleyiniz, dönme açı değerlerini düzenleyiniz.




**Görsel 3.47:** 0:45 zamanındaki Rotation.y değeri 180 yapılır, ilk ve son karelerde değeri 0'dır.

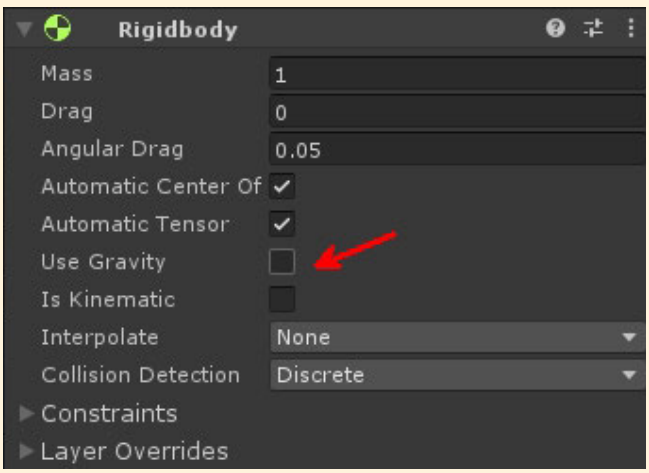
NOT

Hareketin sonlanma süresi ileri alınarak animasyonun yavaşlaması sağlanır. Ana karelerin üstündeki elmas şeklindeki simgeler tutularak ileri geri kaydırılır. Böylece o andaki tüm karelerin yeri değişmiş olur.

**9. Adım:** Oyuncu nesneye değdiğinde ödülün toplanması için gereken odulTopla ismindeki kodu PlayerCapsule adındaki oyuncu nesnesine ekleyiniz.

```
using UnityEngine;
public class odulTopla : MonoBehaviour
{
    void OnCollisionEnter(Collision temas)
    {
        Destroy(temas.transform.gameObject);
    }
}
```

**10. Adım:** Test etmeden önce ödül nesnesine **RigidBody** bileşenini ekleyiniz. Görsel 3.48'de görüldüğü gibi Rigidbody'nin **Use Gravity** özelliğini değiştiriniz.

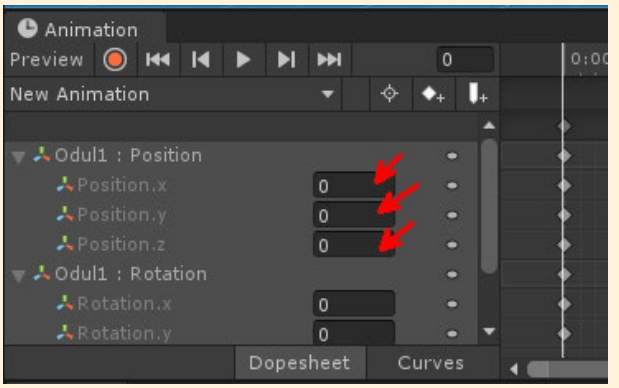



**Görsel 3.48:** Use Gravity seçeneği iptal edilir.

**11. Adım:** Tek ödül nesnesi için animasyon ve değince yok olma özelliği çalışır fakat ödül nesnesi çoğaltılıp sahnede başka yerlere taşındığında, animasyon kaydı sadece ilk yere göre olduğu için hepsi kaydedilen noktada toplanıp hareket etmeye başlar.

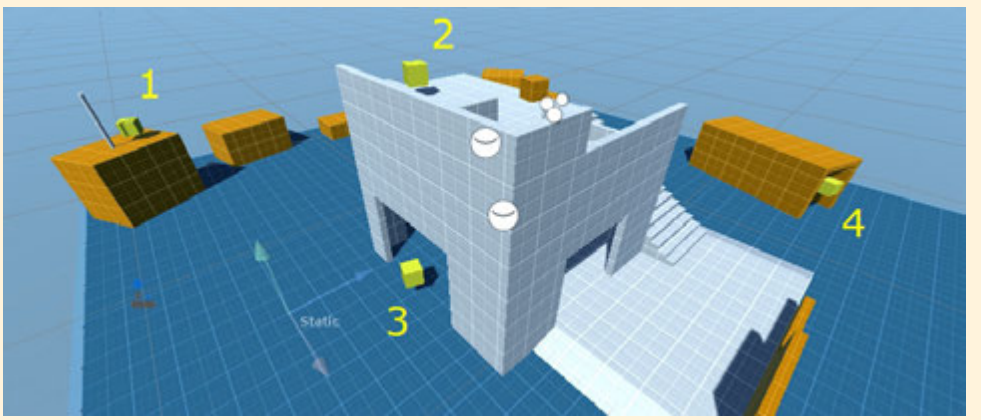
**Position** bilgisi belli bir koordinata göre yapıldığı için bu istenmeyen durum şu adımlar ile iyileştirilir:

- Hierarchy penceresinde ödül nesnesine sağ tıklayıp Create Empty Parent ekleyiniz.
- Eklenen kapsayıcı nesnenin tutup sahnedeki yerini değiştiriniz.
- Artık nesne başka yerde de çalışır fakat yine kapsayıcının içinde biraz kaymış olarak görülür. Bunun nedeni animasyonun 0, 0, 0 konumunda kaydedilmemesidir.
- Görsel 3.49'da görüldüğü gibi tüm ana karelerdeki pozisyon değerlerini düzenleyiniz.




**Görsel 3.49:** Animasyonun ilk, orta ve sondaki değerlerinin düzenlenmesi

**12. Adım:** Görsel 3.50'de görüldüğü gibi ödül nesnesini çoğaltarak sahnede istenen yerlere yerleştiriniz; böylece oyuncu sahnede dolaştıkça, keşfederek ödülleri kazanır.




**Görsel 3.50:** Ödül nesnelerinin yerleştirilmesi

## 3.2. SİMÜLASYON İŞLEMLERİ

**Oyun motoru,** gerçekçi 2D ve 3D oyunlar oluşturmak için kullanılabilen güçlü bir yazılımdır. Geliştiricilerin gerçek zamanlı fizik simülasyonları oluşturmasını sağlar. Fizik motoru; katı cisimleri, sıvıları ve gazları gerçeğe yakın olarak görselleştirir. Simülasyonlar oyunları daha eğlenceli ve sürükleyici hâle getirir.

Örnek simülasyon projeleri şunlardır:

- **Araç Simülatörleri:** Araba, uçak, gemi veya diğer taşıtların gerçeğe yakın sürüş deneyimini yaşatmayı sağlar.
- **Biyolojik Simülasyonlar:** Bir ekosistemdeki farklı organizmaların etkileşimlerinin ve davranışlarının modellendiği, canlı organizmaların davranışlarının taklit edildiği simülasyon yapımını sağlar.

- **Eğitim ve Eğlence Simülasyonları:** Tarih, coğrafya, sanat gibi çeşitli alanlarda eğitim amacıyla interaktif simülasyonlar yapımını sağlar.
- **Fizik Simülasyonları:** Ateş, duman, kumaş ve sıvı simülasyonu gibi gerçekçi fizik simülasyonları oluşturmaya olanak tanır.
- **Robotik Simülasyonlar:** Robotik kollar, insansız hava araçları, otonom araçlar ve diğer robotik cihazların simülasyonları ile robotların davranışlarını öğrenmek için tasarlanır.
- **Sağlık Simülasyonları:** Tıp alanında cerrahi simülasyonlar, anatomi eğitimleri ve tedavi prosedürlerinin uygulamalı eğitimi gibi sağlıkla ilgili çeşitli simülasyonlar oluşturmayı sağlar.
- **Şehir Simülasyonları:** Şehirlerin trafik akışını, nüfus hareketliliğini, planlama ve tasarım kararlarının sonuçlarını incelemek için kullanılır. İnsan ve toplumların davranışlarının taklit edildiği simülasyon türüdür.
- **Uzay Simülasyonları:** Uzayda gezinme, gezegenler arası seyahat ve uzay istasyonu simülasyonları gibi astronomi ve uzay keşif projeleri için simülasyon imkânı sağlar.

### 3.2.1. Çeşitli Etkileri İçeren Uygulamalar

Oyun yapımında çeşitli fizik hesaplamaları ve gerçek zamanlı efektler ile daha doğal ve gerçekçi etkiler sağlanır. Oyun içinde kumaş, nesnelerin kırılması, atmosfer olayları, deniz ve nehirler gibi çok sayıda etkiye yer verilir.

#### Kumaş Etkisi

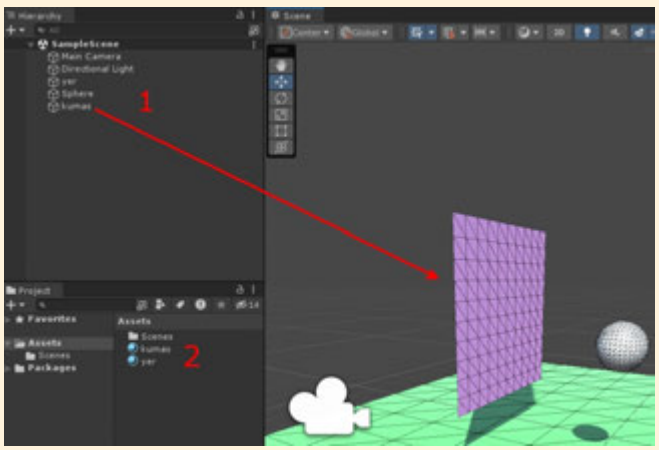
Düzlem şeklindeki bir nesneye Cloth bileşeni eklenerek kumaş etkisi uygulanır. Bu etki; nesnelerin rüzgâr, çekme, itme veya diğer fiziksel etkenler ile nasıl büküldüğünü, esnediğini veya dalgalandığını taklit etmek amacıyla kullanılır. Kumaş etkisi; giysi, bayraklar, perdeler, saçlar ve diğer esnek materyallerin doğru bir şekilde görüntülenmesini sağlar.

6. UYGULAMA

Simülasyon oluşturmak için boş bir proje açarak gerekli nesneleri ekleyip kodları uygulayınız.

**1. Adım:** Görsel 3.51'de görüldüğü gibi kumaş için bir düzlem ve çarpacak nesne için de bir küre ekleyiniz.

- Yer ve kumaşa materyaller ekleyiniz.
- Kumaşı yere dik bir şekilde yerleştiriniz ve döndürünüz.




**Görsel 3.51:** Düzlem ve küre eklenir.

**2. Adım:** Küre nesnesine **Rigidbody** bileşenini ekleyip **Use Gravity** özelliğini iptal ediniz. Küre nesnesini tasarım ekranında kumaşa doğru elle hareket ettiriniz.
**3. Adım:** Görsel 3.52'de görüldüğü gibi kumaş nesnesine Cloth bileşenini ekleyip değişiklikleri uygulayınız.

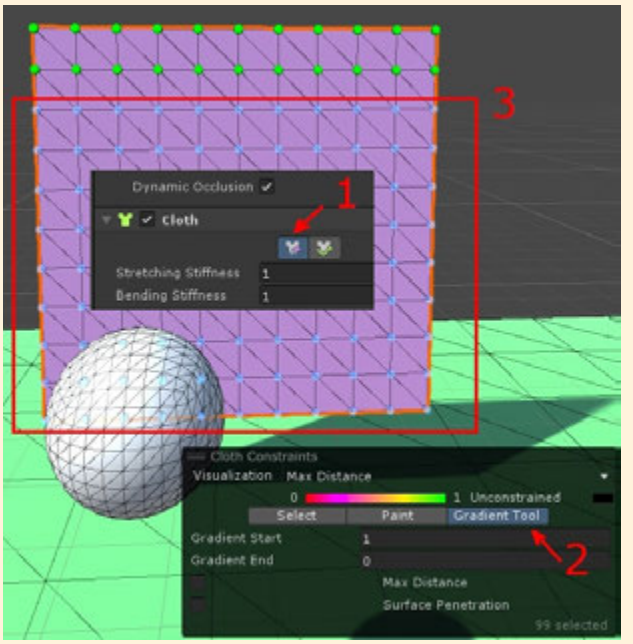



**Görsel 3.52:** Cloth bileşeninin ayarları

Cloth bileşeninin bazı özellikleri şunlardır:

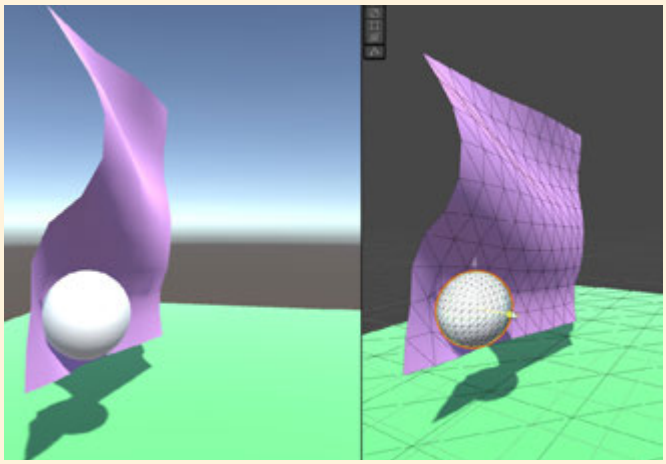
- **Use Gravity:** Yer çekimi desteği sağlar, kumaş havada uçup gitmez.
- **External Acceleration:** İstenen yönde rüzgâr etkisi oluşturur.
- **Sphere Colliders:** Küre ile etkileşim sağlar.

**4. Adım:** Görsel 3.53'te görüldüğü gibi **Edit cloth constraints** düğmesine basarak kumaşın hangi bölgesinde sabit kalınacağını seçiniz. Böylece başka nesne ile etkileşime girdiğinde kumaş yerinden ayrılamaz.




**Görsel 3.53:** Kısıtlama seçeneği ile kumaşın sabit olacağı noktalar seçilir.

**5. Adım:** Görsel 3.54'teki uygulamayı çalıştırarak tasarım ekranı yardımı ile küre nesnesini kumaşa değdiriniz, kumaşın kıvrıldığını görünüz.



**Görsel 3.54:** Kumaşın etkileşimi

NOT

Kumaş içindeki nokta sayısı, kumaşın daha fazla ayrıntıya ve doğruluğa sahip olması demektir. Bu da kumaşın daha bükülebilir ve gerçekçi görünmesini sağlar. Nokta ve yüzeylerin çokluğu oyunun performansını olumsuz etkiler çünkü bilgisayarın daha fazla hesap yapmasını gerektirir.

SIRA SİZDE

Bir oyun nesnesine kumaş etkisini eklemeyi gerçekleştiriniz. Yapım aşamalarındaki hataları not ediniz.

**Değerlendirme**

Çalışmanız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken **Kontrol Listesi**ndeki değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Boş sahne içine kumaş için kullanılacak oyun nesnesi ekledi. | | |
| 2. Kumaş nesnesine Cloth bileşeni ekledi. | | |
| 3. Sahneye Rigidbody bileşeni eklenmiş bir nesne ekledi. | | |
| 4. Projeyi çalıştırarak kumaşın hareket etmesini sağladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## Kırılma Etkisi

Kırılan nesneler, oyun motoru içindeki fizik desteği kullanılarak tasarlanır. Bu sayede oyuncular, nesnelerin gerçekçi bir şekilde kırıldığını deneyimler. Oyun içindeki çarpışma, patlama veya diğer etkileşimler sonucunda nesnelerin parçalanması, oyunun daha heyecan verici ve görsel olarak etkileyici olmasını sağlar.

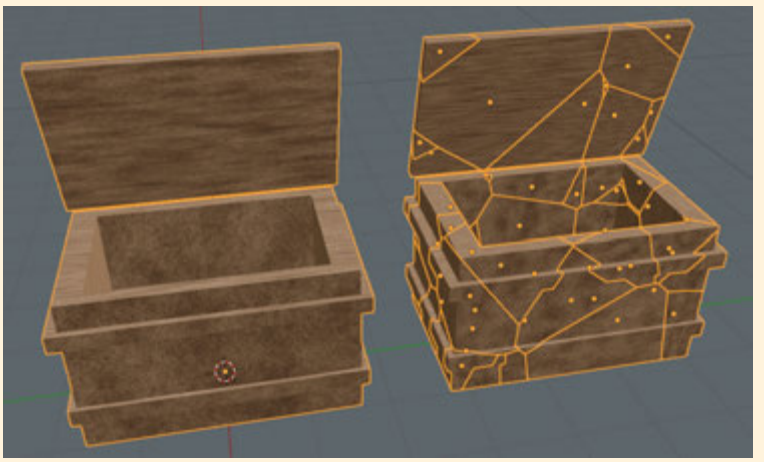
### 7. UYGULAMA

Yeni bir proje açarak hazırlanan nesne ve kodlar ile uygulamayı gerçekleştiriniz.

**1. Adım:** Görsel 3.55'te görüldüğü gibi bir 3D modelleme programı ile bir kutu tasarlayınız. Aynı kutunun parçalanmış hâlini de oluşturunuz.

**NOT**

Açık kaynak program olan Blender 3D içinde **Cell Fracture** eklentisi bulunur. Bu eklenti ile nesne rastgele parçalara ayrılır.



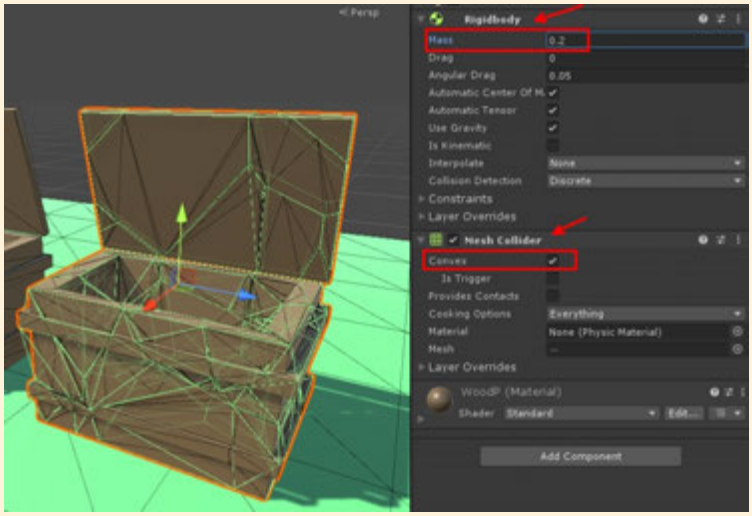
**Görsel 3.55:** Sağlam ve kırık nesne örneği

**2. Adım:** Hasar ve kırılma etkisi için boş bir sahne ekleyiniz.
**3. Adım:** Görsel 3.56'da görüldüğü gibi sağlam ve kırık nesneleri sahneye ekleyiniz.



**Görsel 3.56:** Kutu nesnelerinin sahneye yerleştirilmesi

**4. Adım:** Sağlam nesneye Box Collider ve Rigidbody bileşenlerini ekleyiniz.
**5. Adım:** Görsel 3.57'de görüldüğü gibi kırık olan nesnenin parçalarını seçerek **Rigidbody** ve **Mesh Collider** ekleyiniz. Convex seçeneğini aktif yapınız. Mass ile ağırlıklarını 0.2 yapınız.




**Görsel 3.57:** Kırık parçalara fizik desteği ekleme

**6. Adım:** Kırık olan ikinci nesneyi proje klasörüne sürükleyip Prefab hâline getiriniz ve sahneden siliniz.

NOT

**Prefab,** bir oyun nesnesinin bileşenleri ve ayarlarını içeren yeniden kullanılabilir bir varlıktır. Bir oyun nesnesinin aynı yapılandırmaya sahip birden fazla kopyasını oluşturmak veya orijinalini etkilemeden bir oyun nesnesinin yapılandırmasını değiştirmek için kullanılır.

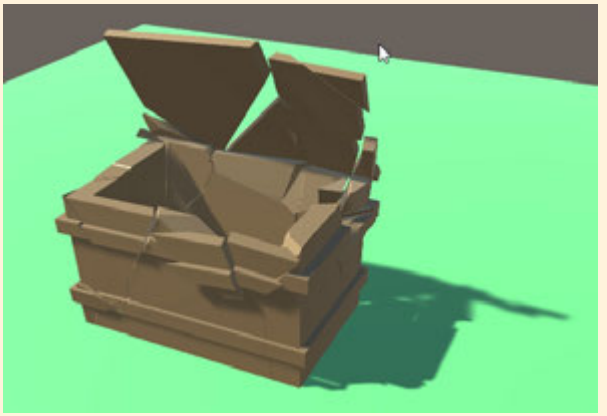
**7. Adım:** Oyunu başlatınız. Sağlam nesneye tıklandığında nesne yok olur ve yerine kırık nesne getirilir. Bu değişim hızlı bir şekilde gerçekleştiğinden oyuncu, nesnenin kırıldığını sanır.

Sağlam nesne üzerine yokEdilebilir adındaki kodu ekleyiniz:

```
using UnityEngine;
public class yokEdilebilir : MonoBehaviour
{
    public GameObject kirikOlanPrefab;
    void OnMouseDown()
    {
        Instantiate(kirikOlanPrefab, transform.position,
transform.rotation);
        Destroy(gameObject);
    }
}
```

**8. Adım:** yokEdilebilir kodu içindeki kirikOlanPrefab özelliğine proje klasöründeki Prefab nesnesini atayınız.

**9. Adım:** Görsel 3.58'de görüldüğü gibi proje çalıştırılınca fare ile sağlam nesne tıklandığında nesnenin parçalandığını görünüz.



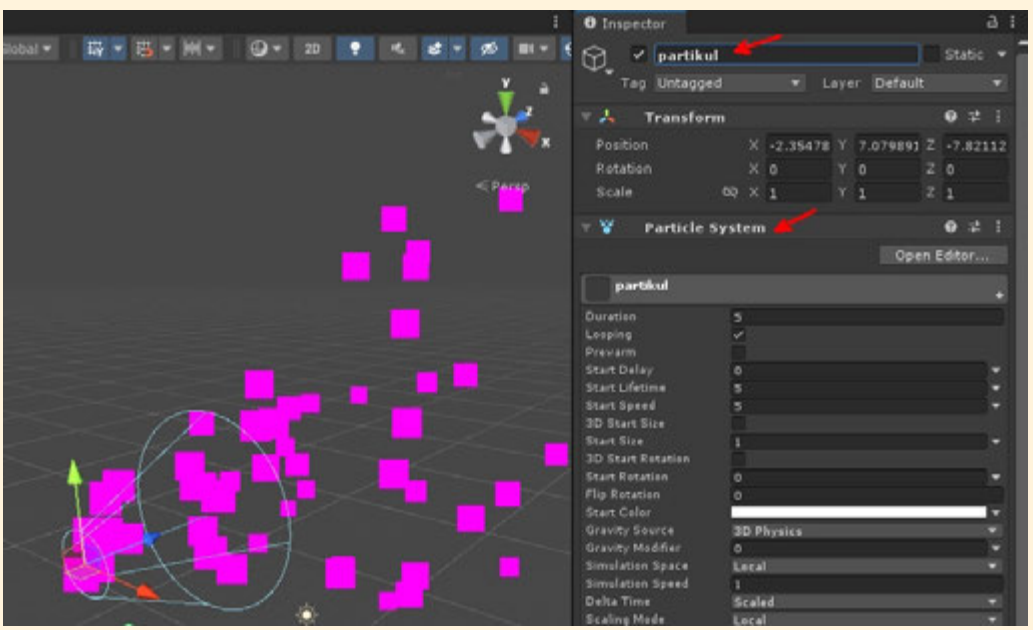
**Görsel 3.58:** Çalışma zamanında kırılan nesne

## Kar Yağma Etkisi

Doğa olaylarının simülasyonu için parçacık sistemleri (**Particle System**) kullanılır. **Particle System** ile yağmur, dolu, kar, havai fişek, şelale, çeşme benzeri örnekler oluşturulur. Bir anlık çalışan patlama sistemi veya belli bir süre kadar çalışan hava olayları sistemi tasarlanır.

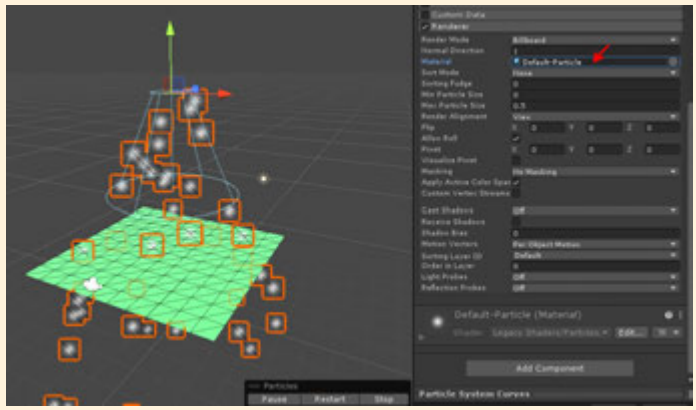
### 8. UYGULAMA

Simülasyon oluşturmak için boş bir proje açarak gerekli nesneleri ekleyiniz.
**1. Adım: Partikül** veya **parçacık** efektleri için boş bir sahne oluşturunuz.
**2. Adım:** Görsel 3.59'da görüldüğü gibi yer nesnesi olarak bir düzlem ve bir boş nesneyi ekleyiniz. Boş nesnenin adını partikul olarak içine Particle System bileşenini ekleyiniz.




**Görsel 3.59:** Boş nesneye Particle System eklenmesi

**3. Adım:** Görsel 3.60'ta görüldüğü gibi parçacıkların yukarıdan aşağı hareket etmesi için partikül nesnesini döndürünüz ve partiküllerin değişik materyallerde olmasını sağlamak için **Renderer>Material** özelliğini değiştiriniz.

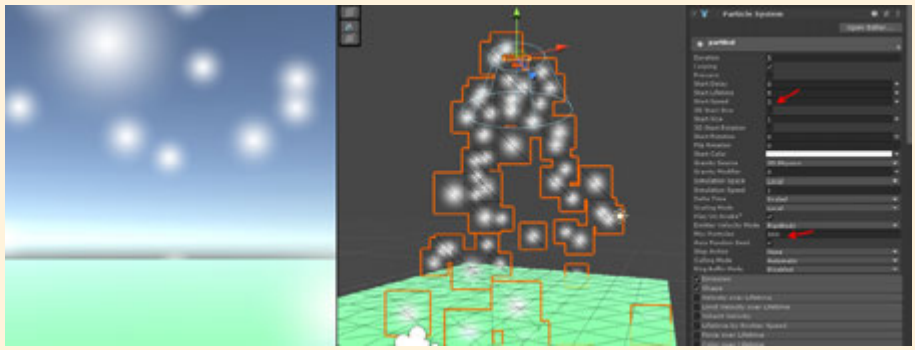


**Görsel 3.60:** Material özelliğini değiştirme

NOT

Play/Pause, Restart ve Stop düğmeleri ile animasyon test edilebilir. Çalıştırmaya gerek kalmadan oyun ekranında sonuç görülür.

**4. Adım:** Görsel 3.61'de görüldüğü gibi Start Speed ve Max Particles değerlerini değiştiriniz.

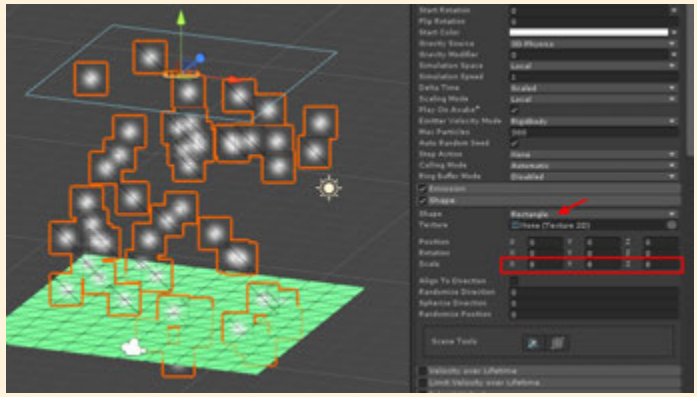


**Görsel 3.61:** Parçacık hızı düşürülür ve sayısı azaltılır.

NOT

Mobil veya WebGL platformları için oyunlar tasarlanıyor ise Particle System ayarları iyileştirilerek oyun performansının artması sağlanabilir.

**5. Adım:** Görsel 3.62'de görüldüğü gibi Shape>Rectangle seçildiğinde parçacıkları bir noktadan konik olarak üretmek yerine geniş bir dikdörtgen yüzeyden çıkarınız.



**Görsel 3.62:** Parçacık üreten şekil Rectangle olarak seçilir.

Parçacık sistemi seçeneklerinden bazıları şunlardır:

- **Size over Lifetime:** Parçacıkların zamanla **boyutları** değişir.
- **Rotation over Lifetime:** Parçacıkların zamanla **açıları** değişir.
- **Color over Lifetime:** Parçacıkların zamanla **renkleri** değişir.
- **Collision:** Parçacıklar belli bir nesneye değince **yön** değişir.
- **Lights:** Bir ışık kaynağı seçilerek parçacıklar onun etrafını aydınlatır.
- **Trails:** Parçacıklar peşlerinden iz bırakarak hareket eder.
- Kod ile parçacık sistemi kontrol edilir.

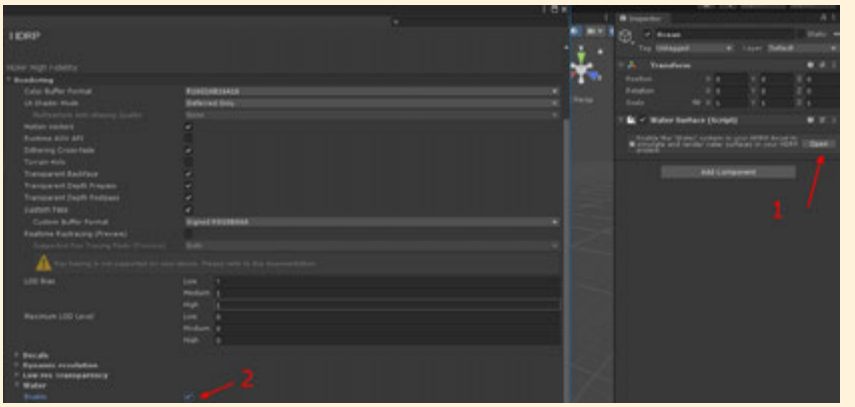
## Okyanus Etkisi

Deniz, göl, havuz gibi su etkileri oyun içine eklenerek oyun dünyası daha canlı ve etkileyici bir hâle getirilir. Bu tür etkiler, su yüzeyinin dalgalanması ve yansıması gibi özellikleriyle oyunculara gerçekçi bir su deneyimi yaşatır.

### 9. UYGULAMA

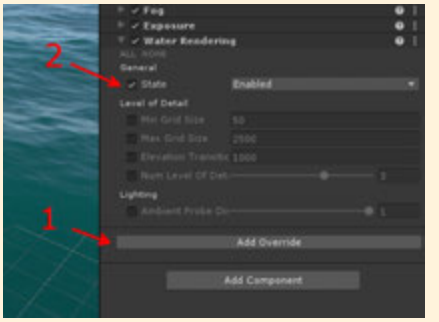
Yeni oyun projesi oluşturulurken 3D Core (**HDRP-High Definition Render Pipeline**) türünde seçildiğinde deniz, havuz, göl ve nehir gibi doğa özellikleri hazır olarak getirilir.

**1. Adım:** Görsel 3.63'te görüldüğü gibi boş bir sahne içine Game Object>Water Surface>Ocean Sea or Lake ekleyiniz. **Inspector** panelinde "Su sistemini aktif ediniz" benzeri bir uyarı görülür.



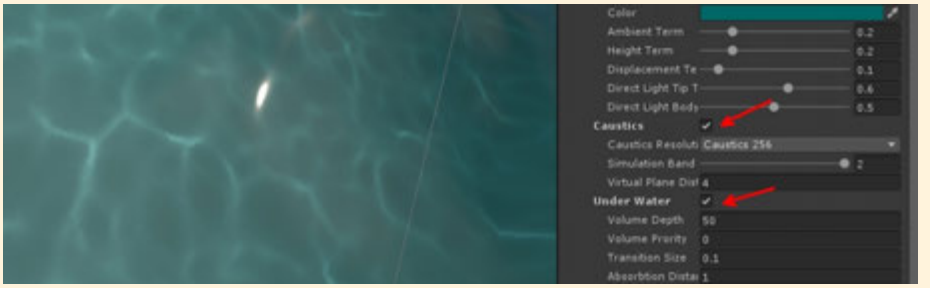

**Görsel 3.63:** Su yüzeyinin görüntülenmesi için gereken ayar yapılır.

**2. Adım:** Görsel 3.64'te görüldüğü gibi **Sky and Fog Volume** nesnesinin üzerine Add Override>Water Rendering ekleyip durumunu aktif (State: Enabled) yapınız.



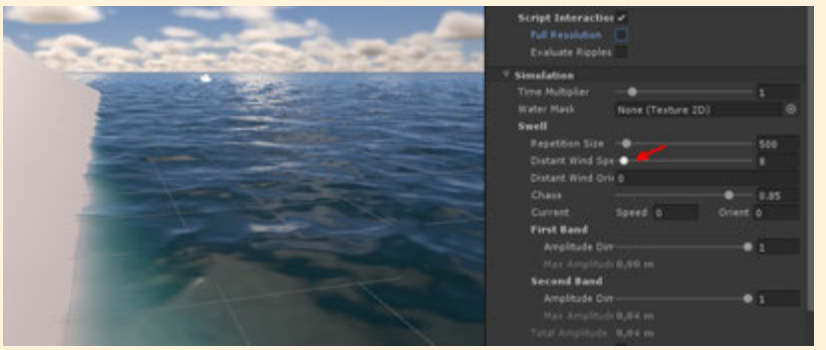

**Görsel 3.64:** Su nesnesinin görüntülenme ayarı

**3. Adım:** Su altında dalgaların görülmesi için bir Plane nesnesi ekleyiniz ve Plane nesnesini eğimli olacak şekilde çeviriniz. Görsel 3.65'te görüldüğü gibi Ocean özelliklerinden **Appearance>Caustics** ve **Under Water** seçeneklerini aktif ediniz.



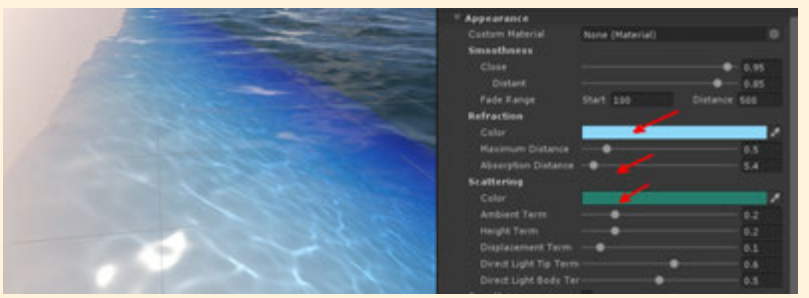

**Görsel 3.65:** Su nesnesinin su altı efektlerinin seçilmesi

**4. Adım:** Rüzgârın hızını değiştirilerek dalga boyutunu değiştiriniz. Görsel 3.66'da görüldüğü gibi Simulation>Distant Wind Speed değerini düşürünüz.




**Görsel 3.66:** Rüzgâr hızının değiştirilmesi

**5. Adım:** Görsel 3.67'de görüldüğü gibi suyun rengini Appearance bölümünde Refraction>Color ve Scattering>Color değerleri ile değiştiriniz. Absorption Distance ile ışık geçirgenliğini örneğin 10 olarak değiştiriniz.




**Görsel 3.67:** Su renginin ayarları

Oyun motorunda **Ocean** nesnesi kullanılarak şu etkiler yapılabilir:

- Gerçekçi dalgalar, akıntılar ve gelgitler oluşturma
- Suyun yüzeyinde ve altında, ışık ve gölge efektleri oluşturma
- Suyun içinde nesneleri yüzdürme
- Su ile etkileşime giren karakterler ekleme
- Su üzerinde gemi ve diğer araçları çalıştırma
- Su altı bitki örtüsü ve dalgakıran gibi görsel efektler oluşturma
- Bir nehir oluşturarak suyun taşlara çarpma etkisini sağlama

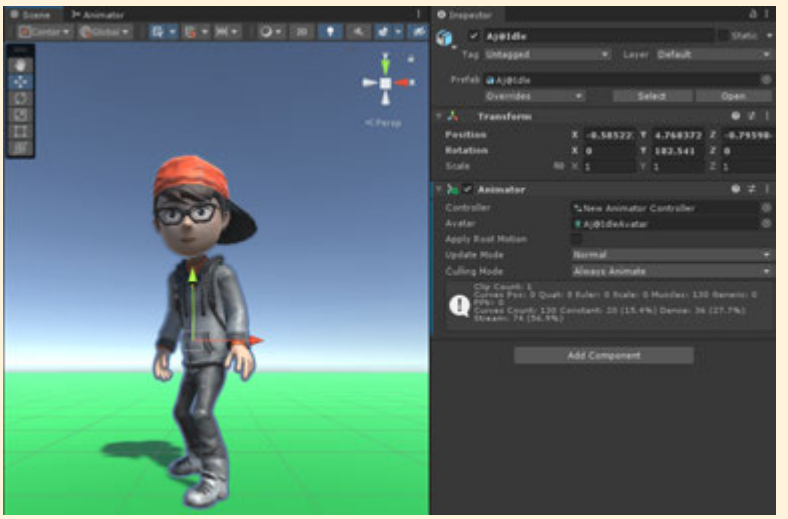
## Ragdoll Etkisi

Oyundaki karakterlere bez bebek (**Ragdoll**) etkisi eklenir. **Ragdoll;** bilinçsiz, kendi hareketlerini kontrol edemeyen bir karakterin fiziksel simülasyonudur. Genellikle oyuncunun hareketinin sonlandırılması anında gerçekleşir.

### 10. UYGULAMA

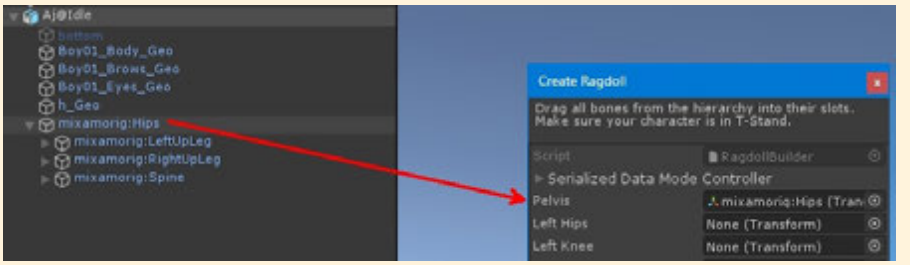
Boş bir 3D Core türünde proje açarak gerekli nesneleri ekleyip kodları uygulayınız.

**1. Adım:** Görsel 3.68'de görüldüğü gibi karakteri boş bir alan üzerine ekleyiniz.



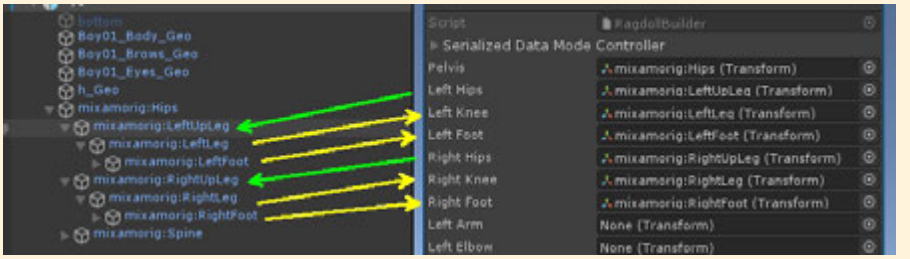
**Görsel 3.68:** Karakter eklenmesi

**2. Adım:** Game Object menüsünden **3D Object>Ragdoll...** komutunu seçiniz. Görsel 3.69'da görüldüğü gibi açılan penceredeki uygun yerlere vücut parçalarını bağlayınız.



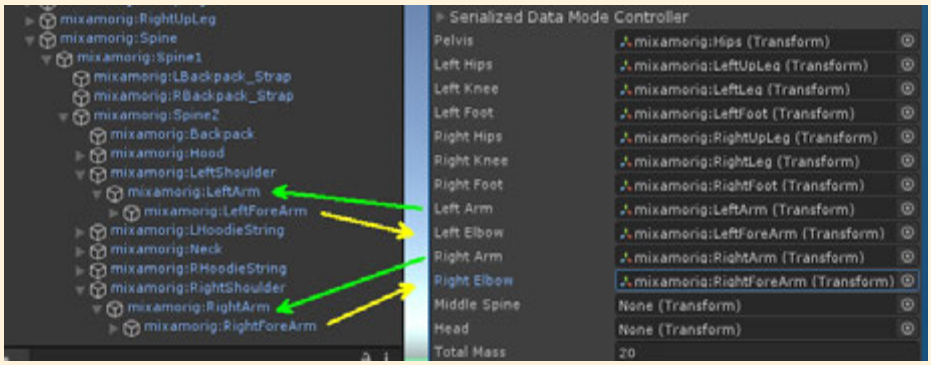

**Görsel 3.69:** Hips parçası Pelvis ile birleştirilir.

**3. Adım:** Görsel 3.70'te görüldüğü gibi bacakları birleştiriniz.



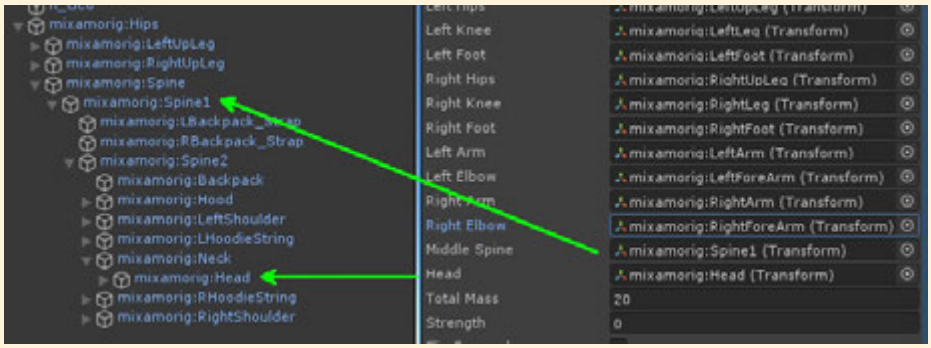

**Görsel 3.70:** Sol ve sağ bacaklar birleştirilir.

**4. Adım:** Görsel 3.71'de görüldüğü gibi kolları birleştiriniz.




**Görsel 3.71:** Omuz ve kollar birleştirilir.

**5. Adım:** Görsel 3.72'de görüldüğü gibi omurga ve başı birleştiriniz.




**Görsel 3.72:** Omurga ve baş birleştirilir.

**6. Adım:** Total Mass ile ağırlık değeri 70 olarak değiştirip Create düğmesine basınız.
**7. Adım:** Görsel 3.73'te görüldüğü gibi baş ve sol kolda düzenleme gerekir.



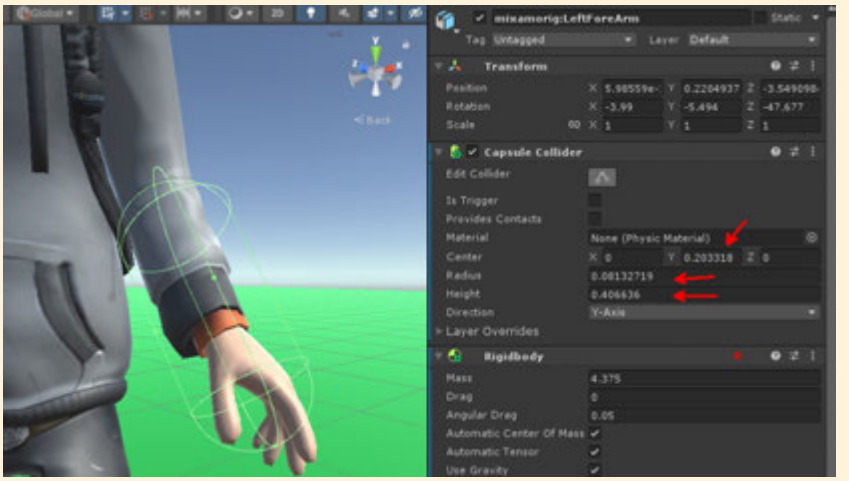
**Görsel 3.73:** Örnekte baş için Sphere Collider küçük kaldı ve sol ön kolda sorun var.

**a)** Görsel 3.74'te görüldüğü gibi mixamorig:Head nesnesini seçerek Sphere Collider özelliklerini düzenleyiniz. Y değeri 0.2 ve Radius değeri 0.25 olarak seçiniz.



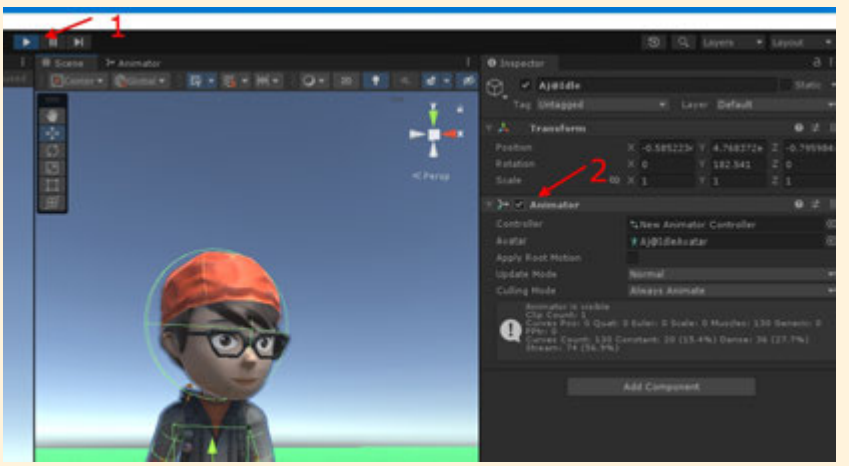

**Görsel 3.74:** Sphere Collider özelliklerini düzenleme

**b)** Görsel 3.75'te görüldüğü gibi mixamorig:LeftForeArm nesnesini seçerek Capsule Collider özelliklerini düzenleyiniz.



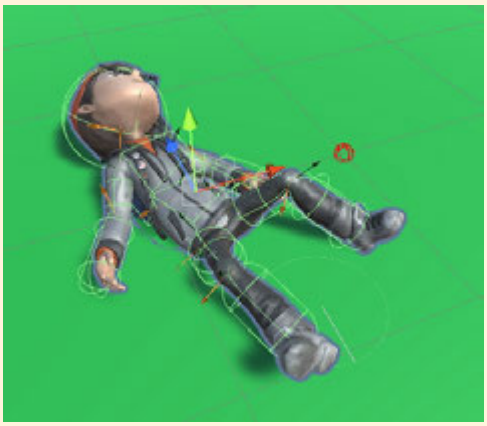

**Görsel 3.75:** Koldaki problemi düzenleme

**8. Adım:** Görsel 3.76'da görüldüğü gibi test etmek için çalışma zamanında karakterin Animator özelliğini kapatınız.




**Görsel 3.76:** Ragdoll animasyonunun test edilmesi

Görsel 3.77'de görüldüğü gibi karakterin olduğu yere düştüğünü görünüz.



**Görsel 3.77:** Karakterin yere düşmüş hâli

**9. Adım:** Ana kameraya fareTikla ve Aj@Idle karakterine durum kodlarını atayınız.

**a)** Fare tıklanması kodunu yazınız.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class fareTikla : MonoBehaviour
{
    [SerializeField]
    private float _maksimumGuc;
    [SerializeField]
    private float _maksimumGucZamani;
    private float _timeMouseButtonDown;
    private Camera _camera;
    void Awake()
    {
        _camera = GetComponent<Camera>();
    }
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            _timeMouseButtonDown = Time.time;
        }
        if (Input.GetMouseButtonUp(0))
        {
            Ray ray = _camera.ScreenPointToRay(Input.
mousePosition);
 if (Physics.Raycast(ray, out RaycastHit hitInfo))
 {
                durum karakter = hitInfo.collider.
GetComponentInParent<durum>();
                if (karakter != null)
                {
float mouseButtonDownDuration = Time.time - _timeMouseButtonDown;
float forcePercentage = mouseButtonDownDuration / _
maksimumGucZamani;
```

```
float forceMagnitude = Mathf.Lerp(1, _maksimumGuc,
forcePercentage);
Vector3 forceDirection = karakter.transform.position - _camera.
transform.position;
                    forceDirection.y = 1;
                    forceDirection.Normalize();
                    Vector3 force = forceMagnitude *
forceDirection;
                    karakter.TriggerRagdoll(force, hitInfo.
point);
                }
            }
        }
    }
}
```

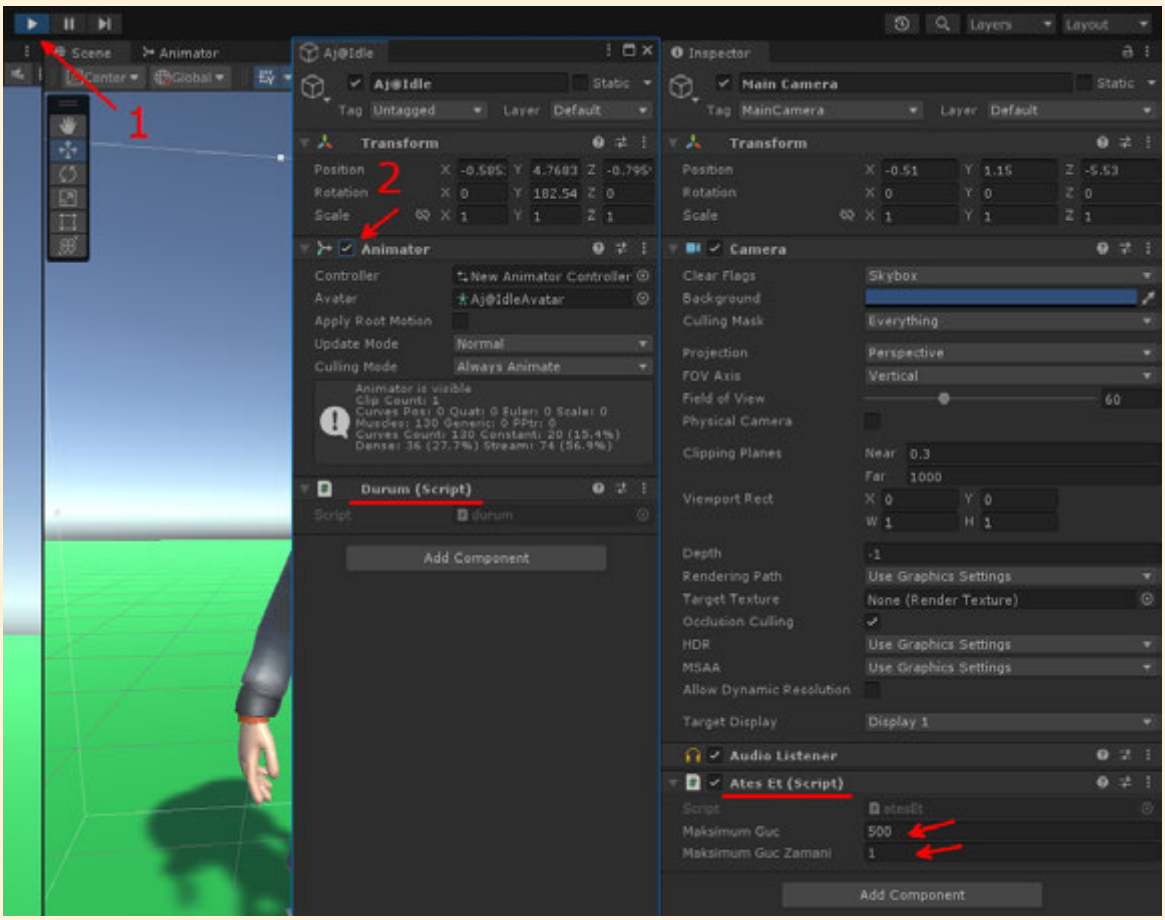
**b)** Durum kodunu yazınız.

```
using System.Linq;
using UnityEngine;
public class durum : MonoBehaviour
{
    private Rigidbody[] _ragdollRigidbodies;
    private Animator _animator;
    private CharacterController _characterController;
    void Awake()
    {
        _ragdollRigidbodies =
GetComponentsInChildren<Rigidbody>();
        _animator = GetComponent<Animator>();
        DisableRagdoll();
    }
    public void TriggerRagdoll(Vector3 force, Vector3 hitPoint)
    {
 EnableRagdoll();
      Rigidbody hitRigidbody = _ragdollRigidbodies.
OrderBy(rigidbody =>
     Vector3.Distance(rigidbody.position, hitPoint)).First();
       hitRigidbody.AddForceAtPosition(force, hitPoint,
ForceMode.Impulse);
    }
   void DisableRagdoll()
    {
        foreach (var rigidbody in _ragdollRigidbodies)
        {
            rigidbody.isKinematic = true;
        }
        _animator.enabled = true;
    }

   void EnableRagdoll()
    {
        foreach (var rigidbody in _ragdollRigidbodies)
        {
            rigidbody.isKinematic = false;
```

```
    }
        _animator.enabled = false;
    }
}
```

**10. Adım:** Görsel 3.78'de görüldüğü gibi test ediniz. Vücudun istenen bölgesine fare ile tıklayıp karakterin devrilmesini sağlayınız. Devrildikten sonra tekrar denemek için özellikler panelinden Animator özelliğini aktif ediniz.




**Görsel 3.78:** Ragdoll etkisi için kodları ekleme

## Eklem Etkisi

Oyun nesnelerini birbirine bağlama ve dönme gibi hareketler için fiziksel kuvvetleri hesaplamak amacıyla eklemler (**Joints**) kullanılır. Örneğin bir kapıya menteşe bağlantısı eklenerek, oyuncu tarafından dokunulduğunda kapının belirli bir açıda açılması sağlanır. Pencere, dolap kapağı veya sallanan bir reklam panosu gibi nesnelerin tasarımında da aynı prensip uygulanır.

Joint adı verilen birleşimlerin üç çeşidi vardır.

- **Sabit Eklem:** Bir nesnenin hareketini başka bir nesneye bağımlı olacak şekilde kısıtlar.
- **Menteşe Eklem:** İki nesneyi bir menteşe etrafında birbirine bağlar.
- **Yay Eklem:** İki nesneyi birbirine bağlar ve belirli bir kuvvetle birbirine çeker.

NOT

Nesne etkileşimleri, hızlı çarpmalar sırasında istenmeyen sonuçlara yol açabilir. Fizik motoru, hızlı gelişen etkileşimleri doğru bir şekilde hesaplayamaz. Bu nedenle, özellikle çoklu oyuncu altyapısına sahip oyunlarda eklemlerin kullanılması tavsiye edilmez.

11. UYGULAMA

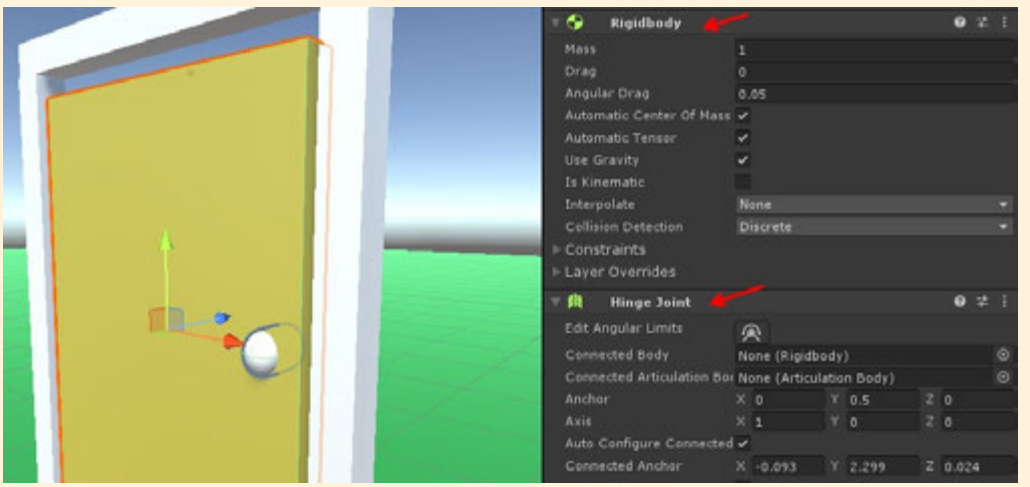
Boş bir proje açarak gerekli nesneleri ekleyip kodları uygulayınız.

**1. Adım:** Görsel 3.79'da görüldüğü gibi küp nesnelerini kullanarak kapı kasası ve kapı gövdesini yapınız.



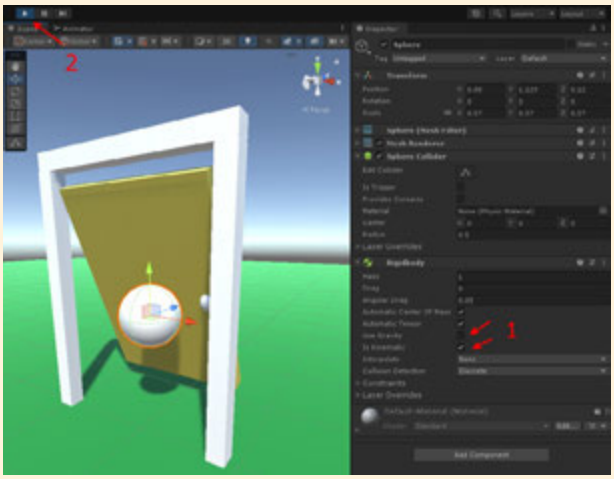
**Görsel 3.79:** Kapı yapımı için küp nesneleri kullanılır.

**2. Adım:** Görsel 3.80'de görüldüğü gibi kapı nesnesine **Hinge Joint** eklentisini ekleyiniz. Aynı zamanda Rigidbody bileşeni kendiliğinden eklenecektir.



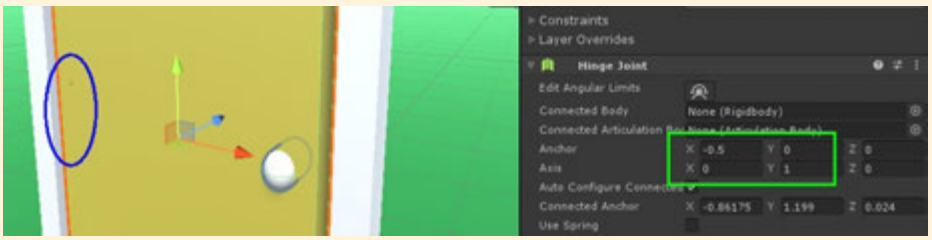

**Görsel 3.80:** Hinge Joint bileşeni eklenir.

**3. Adım:** Görsel 3.81'de görüldüğü gibi kapıyı hareket ettirmek için bir küre ekleyiniz. Seçili küreye Rigidbody bileşenini ilave ediniz.



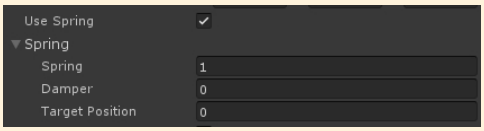

**Görsel 3.81:** Kürenin Rigidbody özellikleri düzenlenir.

**4. Adım:** Görsel 3.82'de görüldüğü gibi kapının hangi eksende döneceği kahverengi bir ok simgesi ile görülür. Dönme noktasının kapının sol ortasına alınması için çapa (**Anchor**) ve eksen (**Axis**) özelliklerini düzenleyiniz.



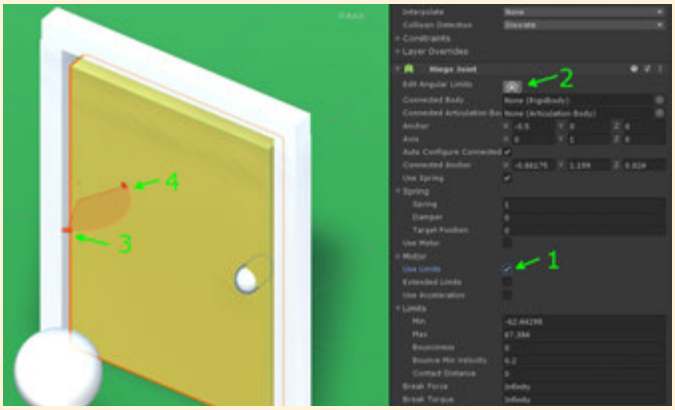

**Görsel 3.82:** Çapa ve eksen özellikleri

**5. Adım:** Görsel 3.83'te görüldüğü gibi kapının kendiliğinden kapanması için yay (**Spring**) özelliğini seçiniz.




**Görsel 3.83:** Yay özelliği

**6. Adım:** Görsel 3.84'te görüldüğü gibi kapının hareketlerini açısal olarak kısıtlayınız.




**Görsel 3.84:** Use Limits ve Edit Angular Limits ile kısıtlama eklenmesi

**7. Adım:** Projeyi çalıştırdıktan sonra Scene panelinde küre ile kapının açılıp kapanmasını test ediniz.

SIRA SİZDE

Kapı yapımı ile ilgili nesneleri ekleyiniz. Yapım aşamalarındaki hataları not ediniz.

**Değerlendirme**

Çalışmanız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken **Kontrol Listesi**ndeki değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Temel nesneler ile kapı oluşturdu. | | |
| 2. Kapı nesnesine Hinge Joint eklentisi ekledi. | | |
| 3. Kapı nesnesini hareket ettirmek için bir küre ekledi. | | |
| 4. Dönme noktasındaki Anchor ve Axis özelliklerini düzenledi. | | |
| 5. Kapının kendiliğinden kapanması için Spring özelliğini düzenledi. | | |
| 6. Kapının hareketleri Edit Angular Limits ile kısıtlandı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 3.3. KARAKTER VE ÇEVRE OLUŞTURMA

Oyun yapımında en temelde, oynayanın kontrol ettiği karakter ve oyun hikâyesinin geçtiği çevre bulunur. **FPS** (First Person Shooter) türünde oyuncunun tüm vücudu görülemezken **TPS** (Third Person Shooter) ve iki boyutlu oyun türlerinde oyuncu ve karakter hareketlerinin tümü görülür.

Yaygın oyun türleri şunlardır:

- **Adventure:** Oyuncular, hikâyeleri takip ederek ilerler. Adventure oyunları, genellikle bulmaca çözme ve problem çözme odaklı oyunlardır.
- **First Person Shooter:** Oyuncu, oyun dünyasını karakterinin gözünden görür ve düşmanları vurarak görevlerini tamamlar.
- **Multiplayer Online Battle Arena (MOBA):** Bu tür oyunlarda iki takım oyuncusu bir harita üzerinde karşı karşıya gelir ve rakip takımın kalesini yok etmeye çalışır. MOBA oyunları hızlı tempolu strateji oyunlarıdır.
- **Platformer:** Oyuncular; platformlar arasında zıplayarak ilerler, ödülleri toplar.
- **Puzzle:** Oyuncular bulmacaları çözerek ilerler. Bunlar genellikle basit kontrollere sahip kısa süreli oynanan oyunlardır.
- **Racing:** Oyuncular arabaları kontrol ederek yarışır.
- **Role Playing Game (RPG):** Oyuncu, bir karakteri kontrol eder ve karakterin hikâyesini takip eder. Oyuncular; karakterlerini geliştirmek için görevler tamamlar, yeni beceriler öğrenir ve ekipmanlar toplar.
- **Real Time Strategy (RTS):** Oyuncu, bir orduyu kontrol eder ve rakip orduyu yok etmeye çalışır.
- **Simulator:** Gerçek hayatta var olan şeylerin simülasyonudur. Simülatör oyunları genellikle gerçekçi ve ayrıntılı oyunlardır.
- **Third Person Shooter:** Oyuncu, oyun dünyasını karakterinin arkasından görür ve düşmanları vurarak görevlerini tamamlar.

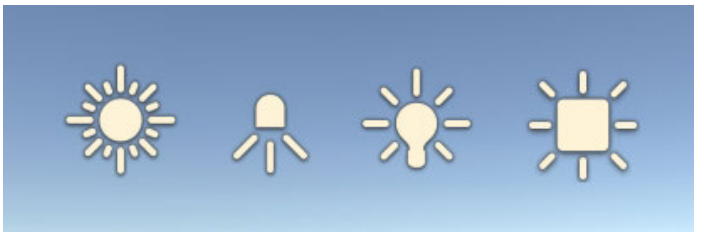
### 3.3.1. Güneş, Işık Türleri ve Etkileri

Yeni bir sahne oluşturulduğunda varsayılan olarak bir güneş ışık kaynağı (Directional Light) ekli gelir.

Işık kaynağının genel özellikleri şunlardır:

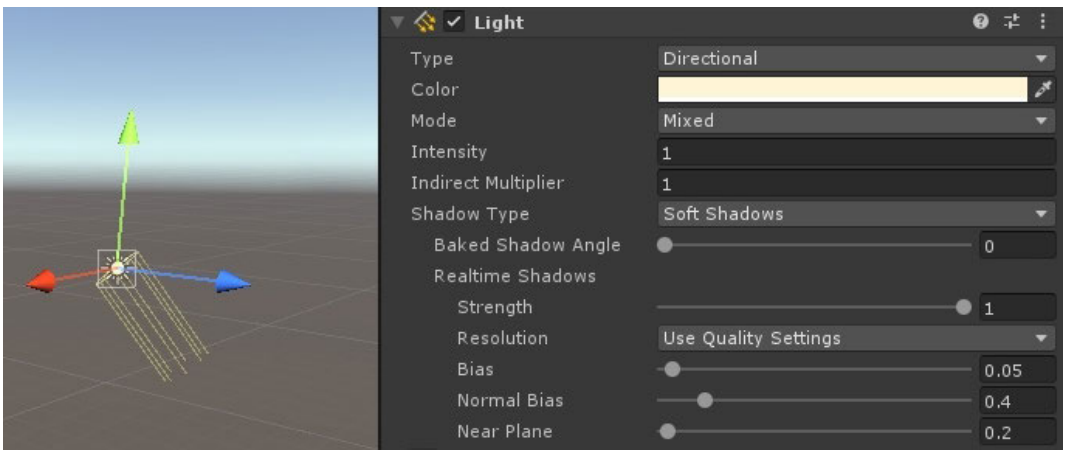
- **Type:** Işık türü değiştirilir. Türe göre bazı özellikler değişir.
- **Color:** Renk değeri değiştirilir.
- **Mode:** Gerçek zamanlı (Realtime), karma (Mixed) ve fırınlanmış (Baked) olarak seçilir.
- **Intensity:** Işık şiddeti değiştirilir.
- **Indirect Multiplier:** Seken ışık miktarı belirlenir.
- **Shadow Type:** Gölge türü ve özellikleri seçilir.
- **Draw Halo:** Işık çevresine toz veya sis benzeri görüntü eklenir.
- **Flare:** Kamera ışığa doğru döndüğünde görünen mercek parlama etkisi eklenir.

Görsel 3.85'te sırasıyla **Directional** (doğrusal), **Spot** (parlak ışık huzmesi), **Point** (nokta) ve **Area** (alan) ışık türleri görülür.



**Görsel 3.85:** Işık türleri

Görsel 3.86'da **Directional** (doğrusal) ışık özellikleri görülür.

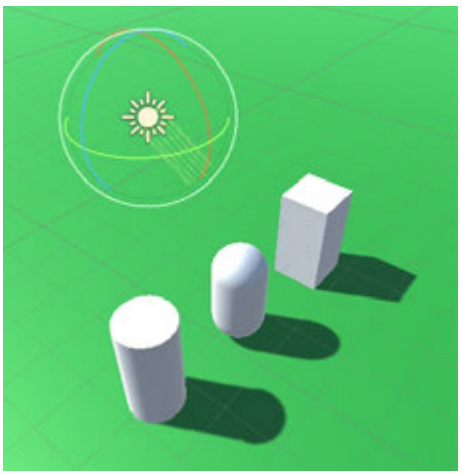



**Görsel 3.86:** Doğrusal ışık kaynağı ve özellikleri

Işık türlerinin kullanım amaçları şunlardır:

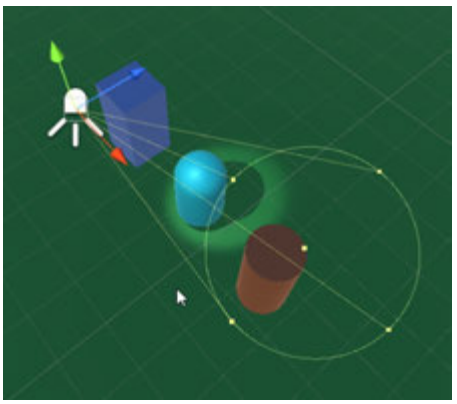
- **Directional:** Güneş olarak kullanılır. Tek yönde sonsuza kadar ışık yayar.
- **Spot:** Koni şeklinde ışık yayar. El veya araç lambası gibi kullanılır.
- **Point:** Belli bir çapta tüm yönlerde ışık yayar. Evde kullanılan ampul aydınlatması gibidir.
- **Area (baked only):** Dikdörtgen veya disk biçimde ışık yayar. Tavandaki floresan lambası gibi kullanılır.

Görsel 3.87'de görüldüğü gibi **güneş** nesnesinin yönü değiştirildikçe gündüz ve gece etkileri oluşturur.



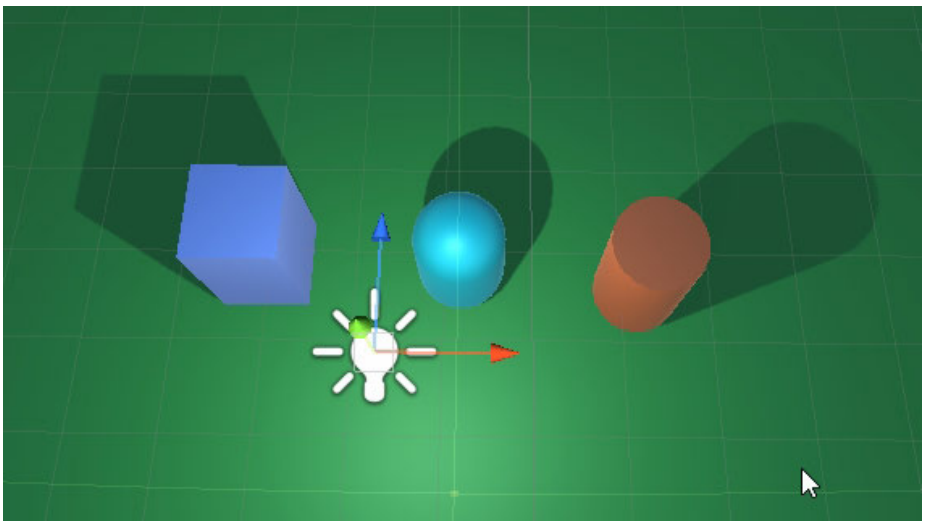
**Görsel 3.87:** Güneş nesnesinin açısını değiştirmek

Görsel 3.88'de görüldüğü gibi **Spot** tipinde bir ışığın yönü ve mesafesi ayarlanıp belli bir yer aydınlatılır.



**Görsel 3.88:** Spot ışığı bir nesneye çevirerek aydınlatmak

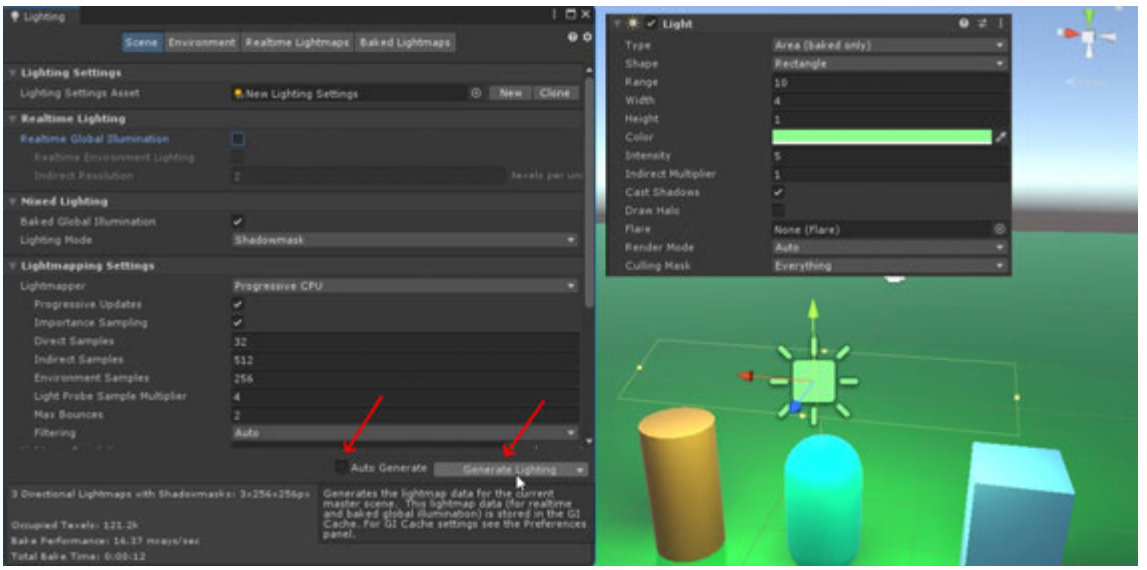
Görsel 3.89'da görüldüğü gibi **nokta ışığı** (Point) ile istenen çapta alan aydınlatılır.



**Görsel 3.89:** Point tipindeki bir ışık ile nesnelerin aydınlatılması

Görsel 3.90'da görüldüğü gibi **Area** ışığının biçimi özelliklerden belirlenir ve sahnenin ışık bilgisi Window>Rendering>Lighting ile açılan pencereden hesaplanır.

Sahnenin karmaşıklığına bağlı olarak **ışık bilgisinin hesaplanması** (Baked Lightmap) uzun sürebilir. Uzun süren hesaplama işlemi, Lighting penceresindeki **Auto Generate** seçeneği iptal edilerek uygun bir zamanda elle yapılabilir.




**Görsel 3.90:** Area ışığının kullanımı

SIRA SİZDE

Sahneye çeşitli ışık kaynakları ekleyiniz. Yapım aşamalarındaki hataları not ediniz.

**Değerlendirme**

Çalışmanız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken **Kontrol Listesi**ndeki değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Boş sahne içine ışık kaynağı için kullanılacak oyun nesnelerini ekledi. | | |
| 2. Nesnelerin Light bileşeni içindeki Type özelliklerini düzenledi. | | |
| 3. Light bileşeninin ayarlarını uygun şekilde düzenledi. | | |
| 4. Projeyi çalıştırarak kumaşın hareket etmesini sağladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

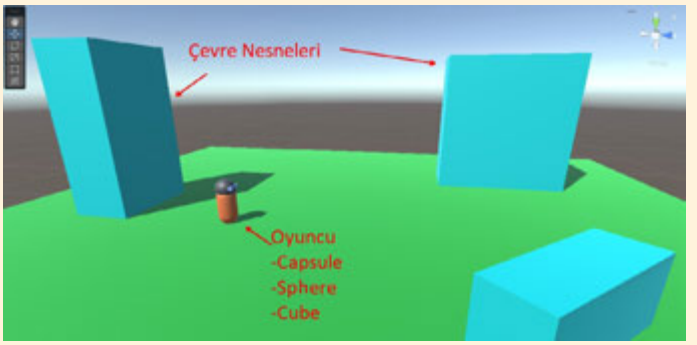
### 3.3.2. Birden Fazla Kamera Kullanma

Birden fazla kamera seçeneği ile oyuncu kamerası değiştirilerek oynanış iyileştirilebilir. Bazı oyuncular **birinci bakış kamerasından** (FPS), bazıları **üçüncü bakış kamerasından** (TPS) oynamak isteyebilir. Belli anlarda kamera değiştirilerek bakış açısına esneklik kazandırılabilir.

12. UYGULAMA

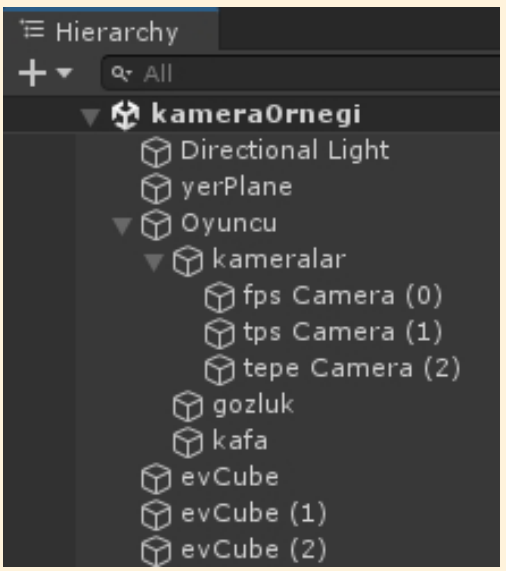
Boş bir proje açarak gerekli nesneleri ekleyip kodları uygulayınız.

**1. Adım:** Görsel 3.91'de görüldüğü gibi sahneye yer, ev ve oyuncu nesnelerini ekleyiniz.

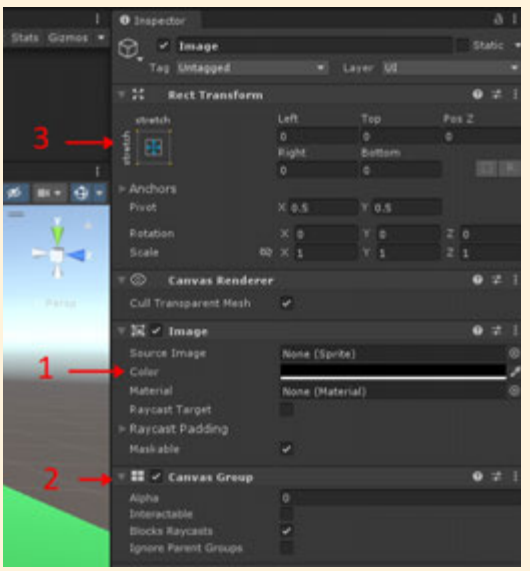



**Görsel 3.91:** Çevre ve oyuncu nesneleri

**2. Adım:** Görsel 3.92'de görüldüğü gibi oyuncu nesnesini düzenlemek için eklenen **Capsule** nesnesine **boş bir nesne** (Empty GameObject) ve onun içine üç adet kamera ekleyiniz. Kameraların ilkini oyuncu kafasının içine, ikincisini biraz arkaya, sonuncusu da daha üstte olacak şekilde konumlandırınız.
**3. Adım:** Görsel 3.93'te görüldüğü gibi sahneye **görsel arayüz** (UI) elemanlarından **Image** nesnesini ekleyiniz. Bu nesne ile kameradan kameraya geçerken göz kırpması benzeri bir etki sağlanır. Image üzerine Add Component>Canvas Group ekleyiniz. Renk ayarını siyah olarak düzenleyiniz. Canvas Group'ta Alpha 0 değeri olarak değiştiriniz. Interactable özelliğini iptal ediniz. Tüm ekranı kaplaması için Strech özelliğini seçiniz. Left, Top, Right ve Bottom değerleri 0 yapınız.




**Görsel 3.92:** Oyuncu içine kameralar ve diğer nesneler eklenir.




**Görsel 3.93:** Image nesnesinin özellikleri

**4. Adım:** Oyuncu PlayerMovement ve kamera değişim CameraTransition kodlarını yazınız.

**a)** Oyuncunun hareket etmesi için gereken kodları yazınız.

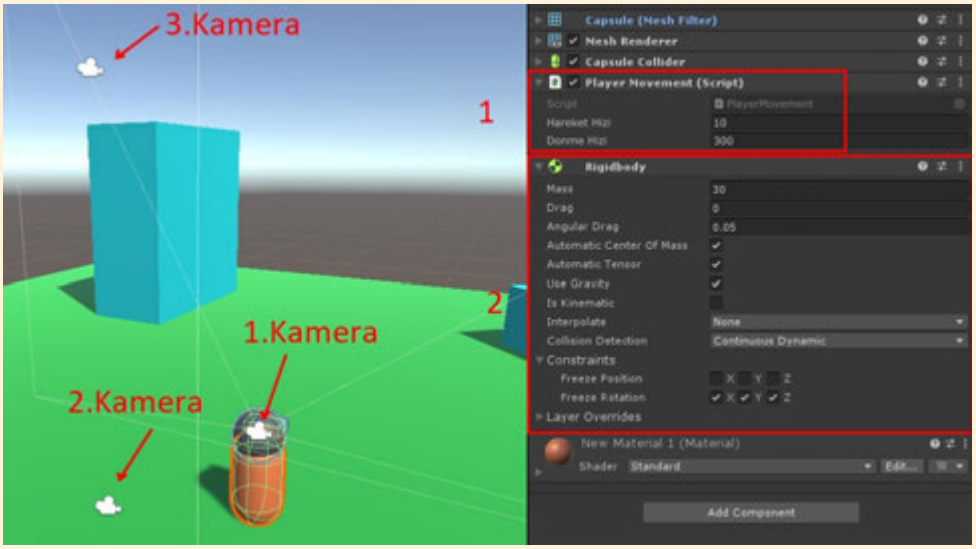
```
using UnityEngine;
public class PlayerMovement : MonoBehaviour
{
    public float hareketHizi = 10.0f;
    public float donmeHizi = 300.0f;
    private Rigidbody rb;
   void Awake()
    {
        rb = GetComponent<Rigidbody>();
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }
   void Update()
    {
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");
        float mouseXInput = Input.GetAxis("Mouse X");
Vector3 rotation = new Vector3(0.0f, mouseXInput, 0.0f) *
donmeHizi * Time.deltaTime;
        rb.rotation *= Quaternion.Euler(rotation);
        Vector3 movement = Quaternion.Euler(0, rb.rotation.
eulerAngles.y, 0) * new Vector3(horizontalInput, 0.0f,
verticalInput);
        if (movement.magnitude > 1.0f) movement.Normalize();
        Vector3 velocity = movement * hareketHizi;
        rb.velocity = new Vector3(velocity.x, rb.velocity.y,
velocity.z);
    }
   void OnDestroy()
    {
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }
}
```

**b)** Oyun oynanışı sırasında kamera değişimi olması için gereken kodu yazınız.

```
using UnityEngine;
using System.Collections;
using UnityEditor.Rendering;
public class CameraTransition : MonoBehaviour
{
    public Camera[] oyuncuKameralari;
    public CanvasGroup fadeCanvasGroup;
 public float fadeSuresi = 1.0f;
```

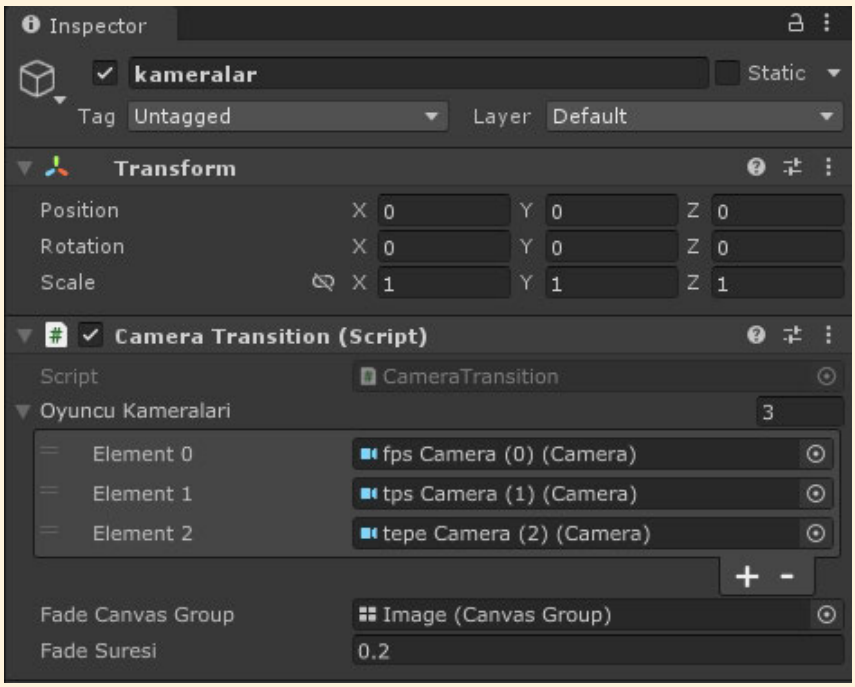
```
    private int aktifKamera = 0;
    void Start()
    {
        for (int i = 1; i < oyuncuKameralari.Length; i++)
        {
            oyuncuKameralari[i].gameObject.SetActive(false);
        }
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.C))
        {
            aktifKamera++;
            if (aktifKamera > 2) aktifKamera = 0;
            StartCoroutine(SwitchCameraWithFade(aktifKamera));
        }
    }
    public void SwitchToCamera(int kameraSirasi)
    {
        StartCoroutine(SwitchCameraWithFade(kameraSirasi));
    }
    private IEnumerator SwitchCameraWithFade(int kameraSirasi)
    {
        fadeCanvasGroup.alpha = 0;
        while (fadeCanvasGroup.alpha < 1)
        {
            fadeCanvasGroup.alpha += Time.deltaTime / fadeSuresi;
            yield return null;
        }
        foreach (Camera cam in oyuncuKameralari)
        {
            cam.gameObject.SetActive(false);
        }
        oyuncuKameralari[kameraSirasi].gameObject.
SetActive(true);
        while (fadeCanvasGroup.alpha > 0)
        {
            fadeCanvasGroup.alpha -= Time.deltaTime / fadeSuresi;
            yield return null;
        }
    }
}
```

**5. Adım:** Görsel 3.94'te görüldüğü gibi oyuncu nesnesinin özelliklerini düzenleyiniz.




**Görsel 3.94:** Kod eklenir ve Rigidbody bileşeni düzenlenir.

**6. Adım:** Görsel 3.95'te görüldüğü gibi oyuncu içinde bulunan boş kameralar nesnesine CameraTransition kodu ekleyiniz ve kod penceresine gereken nesneleri atayınız.




**Görsel 3.95:** Kamera geçişi için gereken nesneler atanır.

**7. Adım:** Oyunu test ediniz. Klavyeden C tuşuna basılarak kameradan diğerine geçilir. Fare ve klavye ile oyuncu hareket ettirilir.

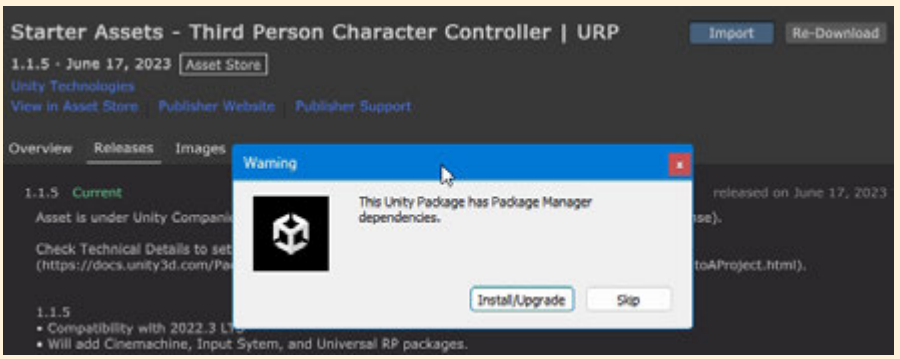
### 3.3.3. Kullanıcı Arayüzüne Kuş Bakışı Harita Ekleme

Oyun ekranının bir kenarına oyuncuyu merkez alarak görüntüleyen küçük bir üstten bakış haritası eklenebilir. Bu haritada önemli nesneler simge olarak gösterilebilir.

13. UYGULAMA

Boş bir proje açarak gerekli nesneleri ekleyip kodları uygulayınız.

**1. Adım:** Starter Assets-Third Person Character Controller paketini Asset Store'dan indirip kurunuz. Görsel 3.96'da görüldüğü gibi soru sorulursa güncelleme için **Install/Upgrade** düğmesine basınız. Yeniden başlatma istenirse yeniden başlatınız.

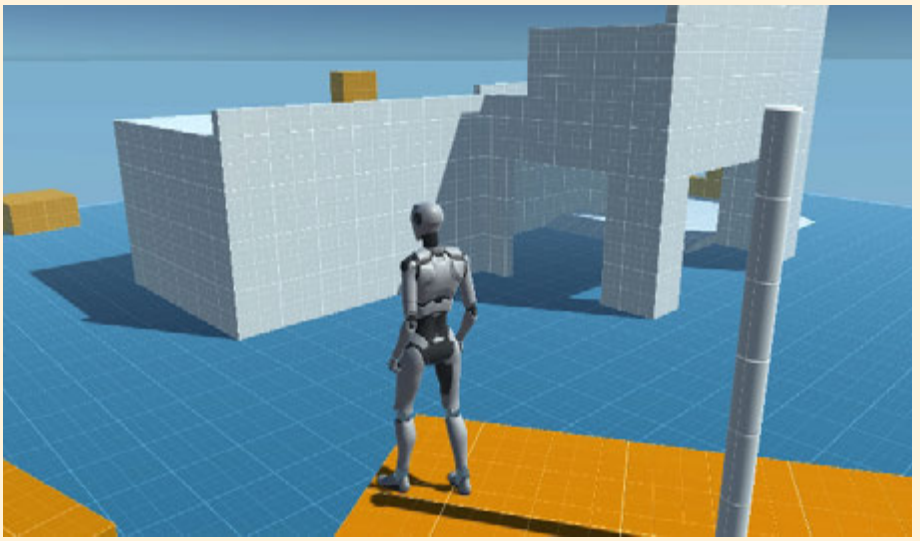



**Görsel 3.96:** Gerekli paketin kurulması

NOT

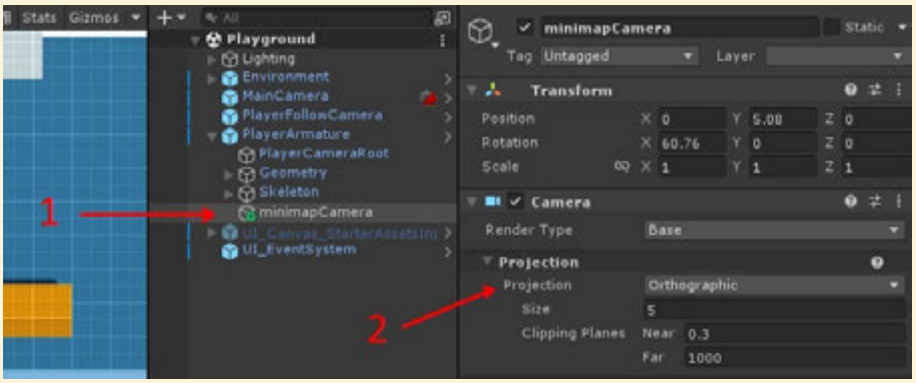
Eğer örnek sahnede nesneler pembe renkli olarak görünüyorsa **Window>Rendering>Render Pipeline Converter** menüsü ile dönüştürme işlemleri yapılır.

**2. Adım:** Görsel 3.97'de görüldüğü gibi Playground ismindeki örnek sahneyi açınız.



**Görsel 3.97:** Örnek sahne

**3. Adım:** Görsel 3.98'de görüldüğü gibi PlayerArmature içine yeni bir kamera ekleyiniz ve **Orthographic** seçeneğini seçiniz. Ortografik görünümde kamera tüm nesneleri eşit boyutta görüntüler. Kameranın Transform değerlerine dikkat ediniz.




**Görsel 3.98:** minimapCamera adında kamera oluşturulması ve Orthographic seçeneğinin seçilmesi

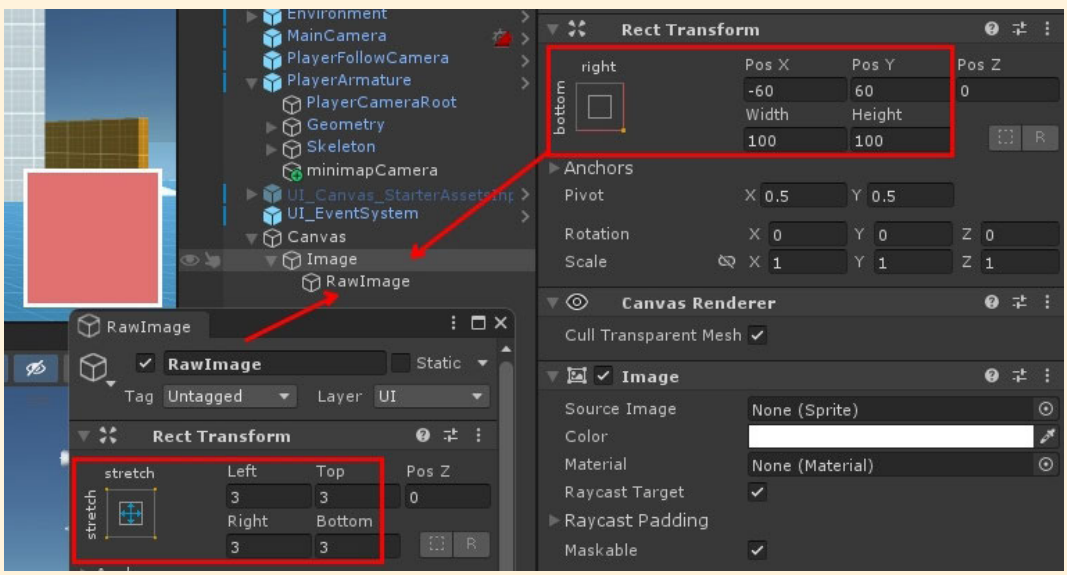
NOT

Konsol penceresinde görülen uyarıyı gidermek için Minimap kamerasındaki **Audio Listener** bileşeni silinir. Bir sahnede bir adet Audio Listener olabilir. Birden fazla kamera olduğunda ana kamerada Audio Listener olması yeterlidir.

**4. Adım:** Oyun başlatıldığında son eklenen kamera ile çevre görülür. Ana kameranın **Rendering>Priority** özelliğini "0" yapınız, yeni kameranın **Rendering>Priority** özelliğini "-1" yapınız. Büyük değerde olan kamera öncelikli hâle gelir.

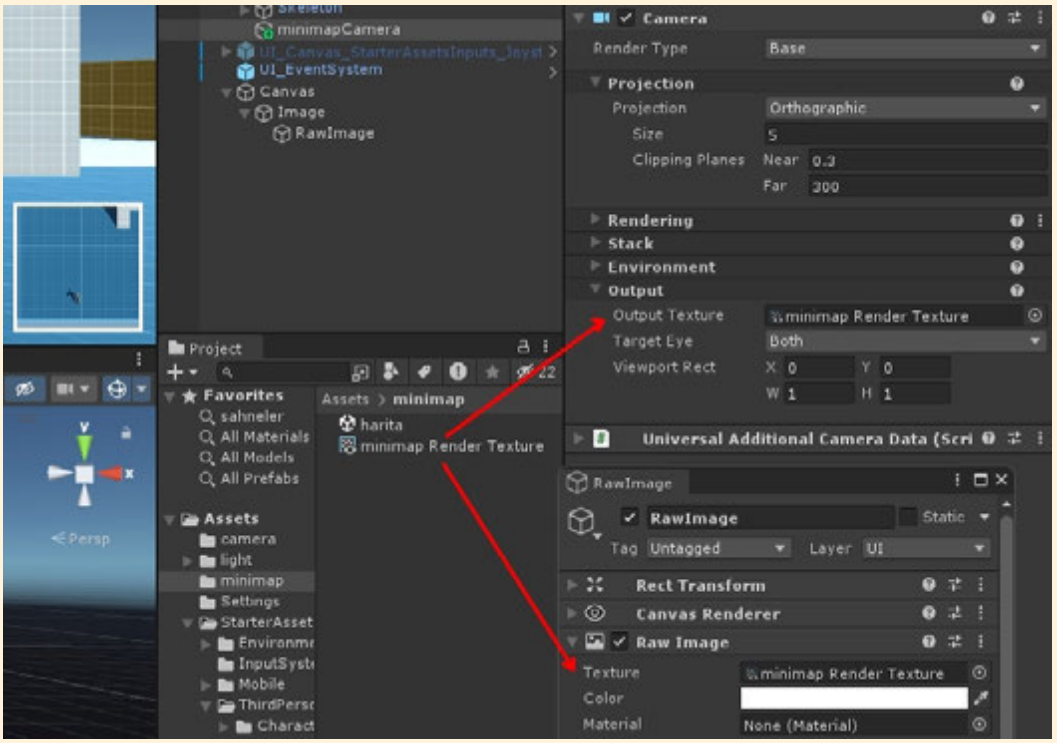
**5. Adım:** Sahneye **GameObject>UI>Image** nesnesini ekleyiniz, eklenen Image nesnesi üzerine sağ tıklayıp **UI>Raw Image** ekleyiniz.

**6. Adım:** Son eklenen RawImage, minimapCamera nesnesinden gelen görüntü ile beslenir. Bu sayede oyuncuyu takip eden diğer kamera nereyi görüyorsa ekranın sol alt köşesinde görülür. Görsel 3.99'da görüldüğü gibi nesnelerin konumlarını ayarlayınız.




**Görsel 3.99:** RawImage ve Image nesnelerinin konum değerleri düzenlenir.

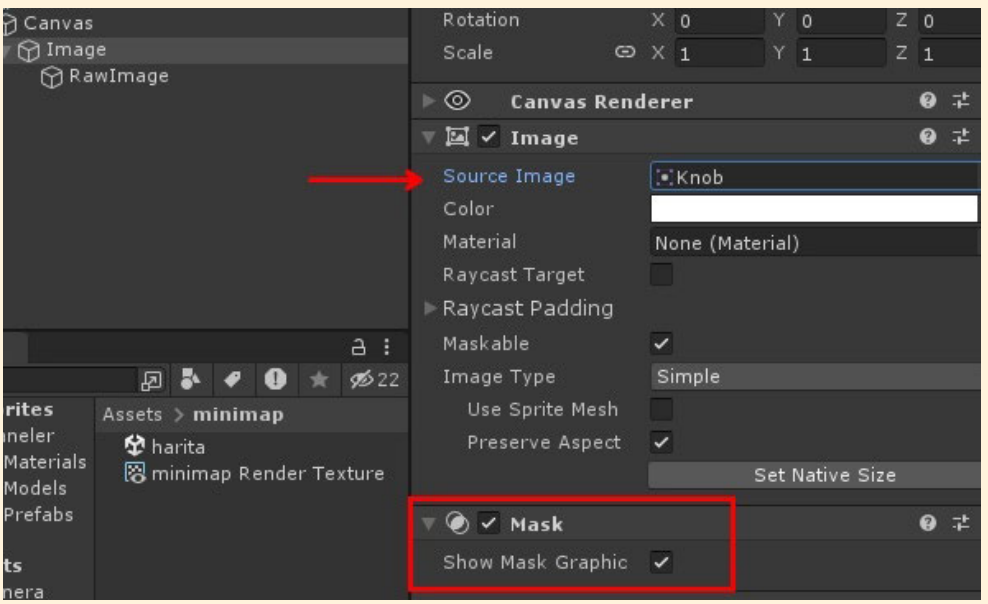
**7. Adım:** Assets klasörüne sağ tıklayıp **Create>Render Texture** ekleyiniz. Görsel 3.100'de görüldüğü gibi minimapCamera'nın Output Texture ve RawImage nesnesinin Texture özelliğini düzenleyiniz.




**Görsel 3.100:** minimapCamera'nın dokusunun düzenlenmesi

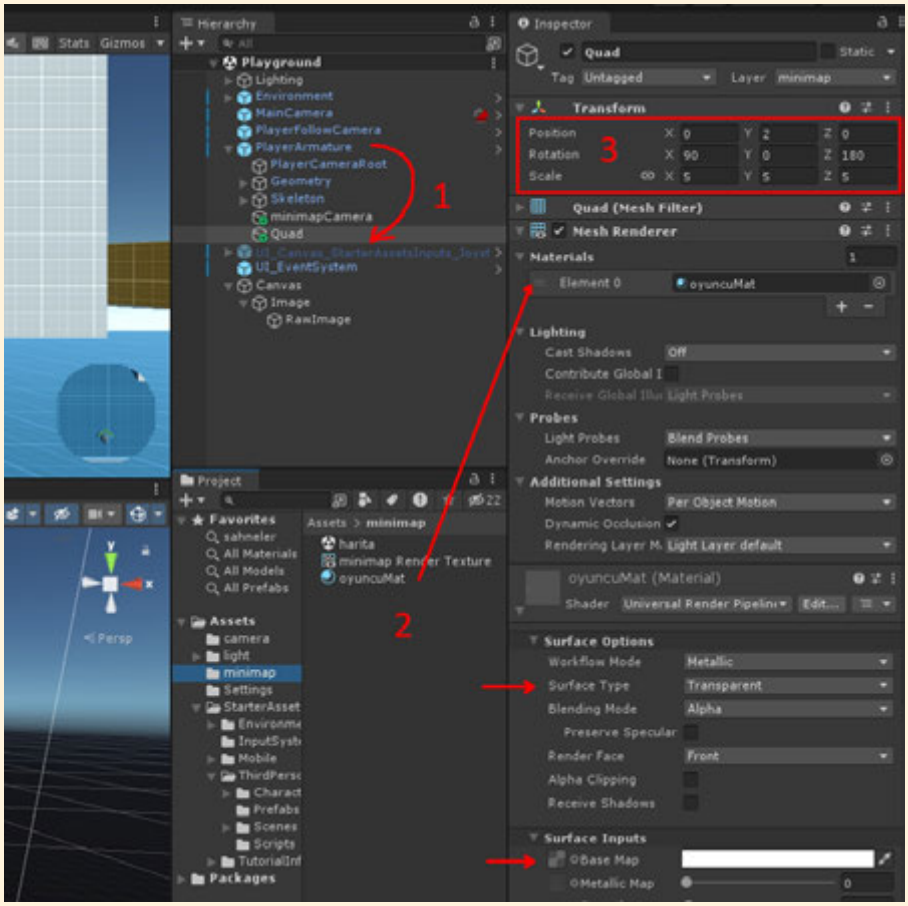
**8. Adım:** Oyun test edildiğinde sağ alt köşede mini harita görülür. minimapCamera ismindeki nesnenin **Size** ve **Clipping Planes** içindeki **Near** ile **Far** özelliklerini önceki görselde görüldüğü gibi düzenleyiniz.

**9. Adım:** Görsel 3.101'de görüldüğü gibi haritayı kare şeklinden yuvarlak hâle getirmek için Canvas içindeki Image nesnesine **Mask** bileşenini ekleyiniz ve **Source Image** özelliğini Knob olarak seçiniz.



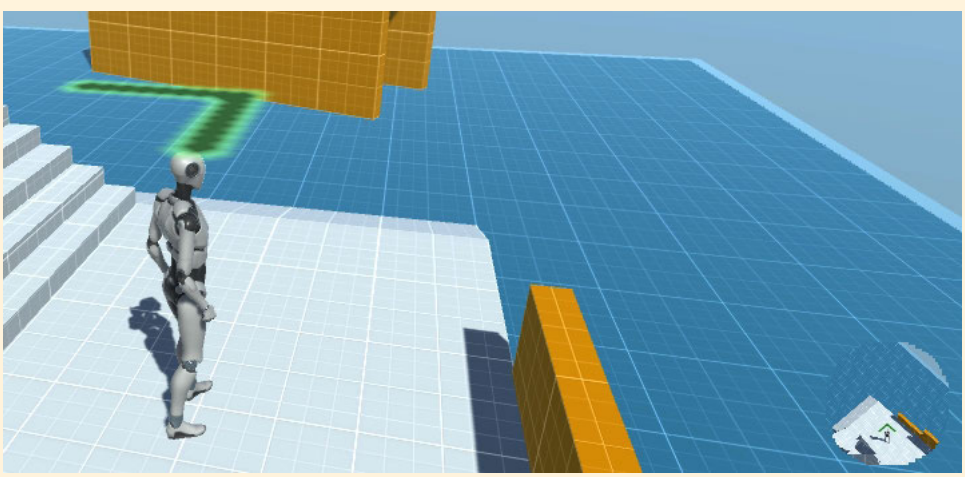

**Görsel 3.101:** Mask bileşeni eklenir ve Source Image seçilir.

**10. Adım:** Görsel 3.102'de görüldüğü gibi yeni bir materyal oluşturunuz. PlayerArmature içine bir **Quad** nesnesi oluşturup yeni materyal ekleyiniz. Base Map kutucuğuna tıklayarak Assets içinde hazır bulunan ok simgesi seçiniz.



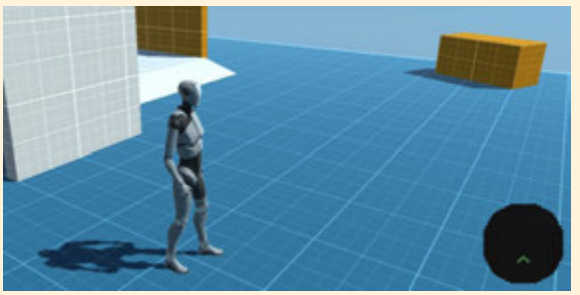

**Görsel 3.102:** Oyuncu içindeki Quad nesnesine yeni bir materyal oluşturma

**11. Adım:** Katman düzenlemesi yapılmadan önceki durum Görsel 3.103'te görüldüğü gibidir.

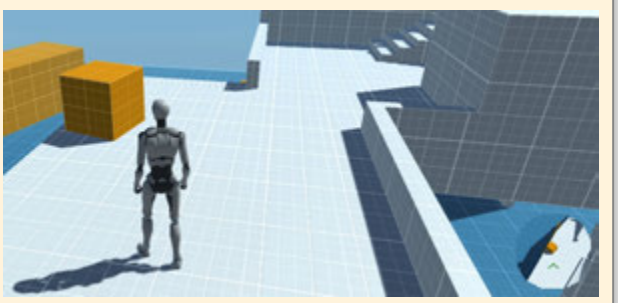


**Görsel 3.103:** Quad nesnesi her iki kamerada görüntülenir.

**12. Adım:** Görsel 3.104'te görüldüğü gibi Quad nesnesinin **Layer** özelliğine **minimap** ekleyiniz ve seçiniz. minimapCamera'nın **Culling Mask** özelliğinde minimap seçiniz. MainCamera'nın **Culling Mask** özelliğindeki minimap seçimini iptal ediniz. Diğer katman seçimleri ise onaylı olmalıdır.
**13. Adım:** Minimap içinde çevresel nesneler görüntülenebilir durumda değildir. Sahnedeki Environment prefab nesnesinin **Layer** özelliğine **cevre** ismindeki katmanı ekleyiniz ve seçiniz. minimapCamera'nın **Culling Mask** özelliğinde minimap ve cevre seçimlerini onaylı hale getiriniz.
**14. Adım:** Oyun karakterinin yeri hissedebilmesi için PlayerArmature içindeki Third Person Controller kodunda Inspector panelinden **Ground Layers** özelliğine cevre katmanını ekleyiniz. Görsel 3.105'te görüldüğü gibi mini haritanın son durumunu inceleyiniz.



**Görsel 3.104:** minimap ismindeki katman ayarı ile görünüm



**Görsel 3.105:** Mini haritanın son durumu

### 3.3.4. Gerçek Zamanlı Yansımalar Ekleme

Gerçek zamanlı yansımalar için **Reflection Probes** nesnesi kullanılır. Ayrıca Global Volume, Light Probe gibi diğer yardımcı nesneler ile sahnedeki ışıklandırma daha gerçekçi hâle getirilir. Bu nesneler ile işlemci ve grafik kartını yoran; ışık, gölge, yansıma gibi işlemler performanslı bir şekilde uygulanır.

- **Reflection Probes (Yansıma Sondaları):** Sahnelerde yansıma bilgisini hesaplamak ve iyileştirmek için kullanılan nesnelerdir. Yansıma sondaları sahnede yansıyan ışığı yakalar, bu bilgiyi malzemelerin ve nesnelerin yüzeylerinde daha gerçekçi yansımalar oluşturmak için kullanır. Özellikle su, cam veya parlak yüzeyler gibi yansıyan yüzeylerin görünümünü iyileştirmek için kullanılır.
- **Global Volume (Küresel Hacim):** Sahnelerde atmosferik etkileri taklit etmek için kullanılan bir tekniktir. Küresel hacim sis, duman veya diğer atmosferik etkilerin görünümünü ayarlamak için kullanılır.
- **Light Probe (Işık Sondası):** Dinamik ışıklandırılan sahnelerde gölgeleri hesaplamak için kullanılır. Işık sondaları sahnedeki ışık kaynaklarının pozisyonunu ve parlaklığını kaydeder, bu bilgiyi nesnelerin üzerine düşen gölgeleri hesaplamak için kullanır.
- **Lightmapping (Işık Haritalaması):** Statik ışıklandırmada ışık kaynakları ve nesnelerin konumları oyunun çalışma zamanında değişmez. Fırınlama (Bake) yöntemi ile ışık ve gölge bilgisi sabit kalır. Sahnede yer alan ışık kaynaklarını ve nesnelerin yüzeylerini inceleyerek sahnenin aydınlatmasını önceden hesaplar. Bu hesaplama işlemi oyun çalıştığında değil, oyun sahnesi düzenlenirken yapılır.
- **Lightmap Textures (Işık Harita Dokuları):** Lightmapping işlemi sonucunda ışıklandırma bilgisi, dokular olarak saklanır. Bu dokular her bir nesne veya yüzeyin nasıl aydınlandığını belirler. Işık haritaları oyun sahnesindeki nesnelerin malzemelerine uygulanır ve sahnedeki nesnelerin görünümünü etkiler.
- **Ambient Occlusion (Ortam Aydınlatma):** Nesnelerin veya yüzeylerin birbirlerine yakın olduğu durumlarda oluşan gölgeleri taklit eder. Işık, nesnelerin yakınlarında olduğunda bu alanlar daha az aydınlanır ve daha fazla gölgelenir. Bu işlem, nesnelerin ve sahnenin daha derin ve hacimli görünmesini sağlar.

- **Volumetric Fog (Hacimsel Sis):** Sahnede bulunan nesnelerin arasından geçen veya havada dağılan ışığın yolunu takip eden bir görsel etki sağlar. Böylece sahneye daha fazla derinlik ve atmosferik bir his kazandırır. Özellikle açık dünya oyunlarında, ormanlarda veya sisli sahnelerde kullanılır.
- **Decals (Çıkartmalar):** Sahnede nesnelerin yüzeylerine eklenen metin, grafikler ya da görsel efektler içeren 2D veya 3D nesnelerdir. Decals, oyun dünyasını daha detaylı ve zengin hâle getirmek, yüzeylere iz, leke, çıkartma veya çamur olarak eklenerek daha fazla gerçeklik katmak için kullanılır.
- **Light Cookies (Işık Kurabiyeleri):** Doku veya sprite (grafik görüntüleri), ışığın yaydığı ışınları geçirecek ve şekillendirecek desenler içerir. Bu yöntem ile ışığın çarptığı bölgelerde desenli gölgeler görülür. Yeryüzüne düşen bulut gölgeleri ve el fenerinin ışığındaki halkalı görünüm Light Cookies ile yapılır.

NOT

HDRP (High Definition Render Pipeline) teknolojisi ile daha gelişmiş ışık ve efektler yapılabilir. Performans ve görsellik açısından çok daha gerçekçi sahneler geliştirilebilir.

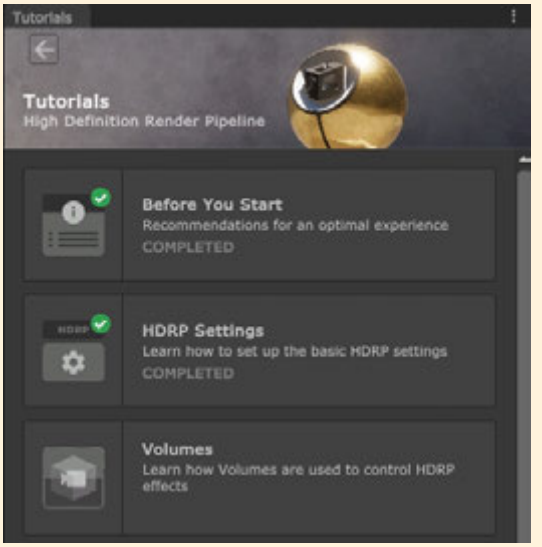
14. UYGULAMA

Uygulama için yeni proje türü olarak **3D Sample Scene (HDRP)** seçiniz.

**1. Adım:** Görsel 3.106'da görüldüğü gibi varsayılan olarak içinde örnek sahnesi olan bir proje açınız.
**2. Adım:** Görsel 3.107'de görüldüğü gibi **Tutorials>Show Tutorials** menüsü ile öğretici sekmesini açınız.



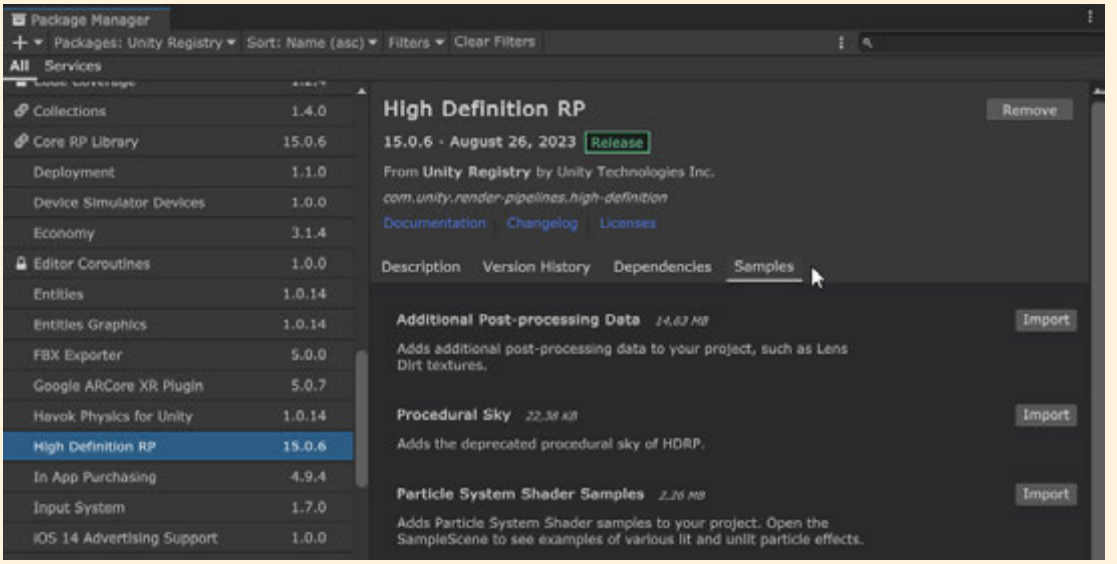
**Görsel 3.106:** Oyuncu nesnesi ve içinde dolaşılabilen sahne




**Görsel 3.107:** HDRP etkilerinin gösterildiği ve anlatıldığı sekme

Örnek sahnede adım adım verilen komutlar ile Volumes, Physically Based Lighting ve Exposure gibi konular işlenir. Görevler tamamlandıkça konu simgesi yeşil olarak işaretlenir.

**3. Adım:** Görsel 3.108'de görüldüğü gibi HDRP ile ilgili daha fazla örnek içeri aktarılır.




**Görsel 3.108:** HDRP ile ilgili örnekler

## 3.3.5. Bitki, Hayvan ve Diğer Doğa Nesnelerini Ekleme

Oyun motorları, **Terrain** adı verilen bir yer ve çevre oluşturma nesnesi içerir. Terrain üzerine ağaç, çimen, taş gibi çeşitli ögeler eklemek için Asset Store'dan indirilebilen hazır paketler veya sizin tasarlayabileceğiniz özel ögeler kullanılabilir.

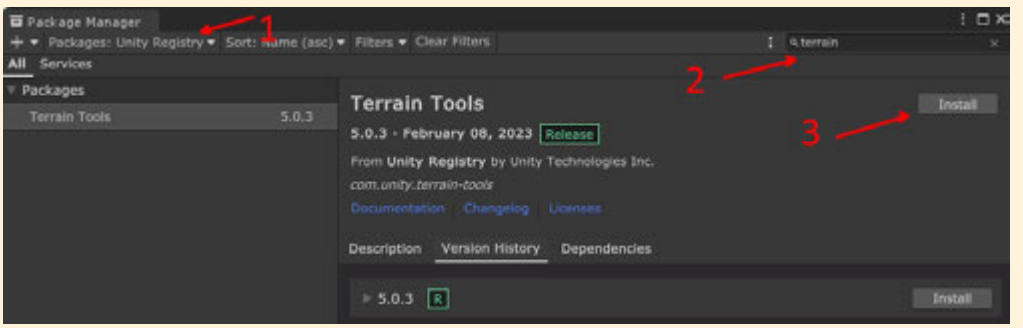
NOT

Yeni bir oyun tasarlanırken oyuna büyük bir sahne ile başlanması tavsiye edilmez. Oyundaki nesne sayısı artacağı için taslak oyunun test edilmesi güçleşir.

15. UYGULAMA

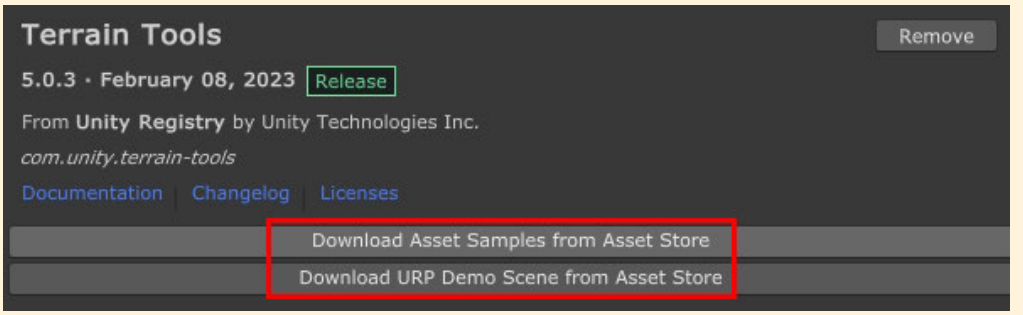
Yeni bir proje açarak projeye çevre ve doğa nesneleri ekleyiniz.

**1. Adım:** Görsel 3.109'da görüldüğü gibi Package Manager penceresi ile **Terrain Tools** kurunuz.



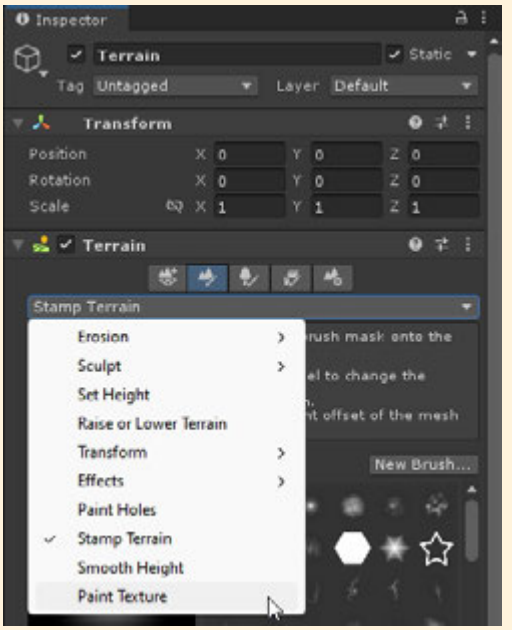

**Görsel 3.109:** Terrain Tools paketinin kurulumu

**2. Adım:** Kurulum sonrasında Görsel 3.110'da görüldüğü gibi örnekleri indiriniz.



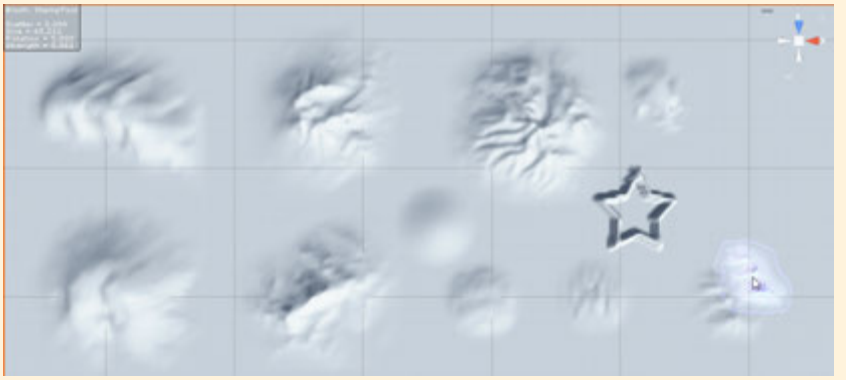

**Görsel 3.110:** Örneklerin indirilmesi ve kurulması

**3. Adım:** Boş sahne içine Terrain nesnesini ekleyiniz. Görsel 3.111'de görüldüğü gibi **Terrain Tools** kurulması ile Terrain nesnesinde Paint Terrain kısmına daha fazla seçenek gelir.




**Görsel 3.111:** Paint Terrain kısmındaki menü

**4. Adım:** Görsel 3.112'de görüldüğü gibi Stamp (damga) fırçası kullanarak desenler oluşturunuz.



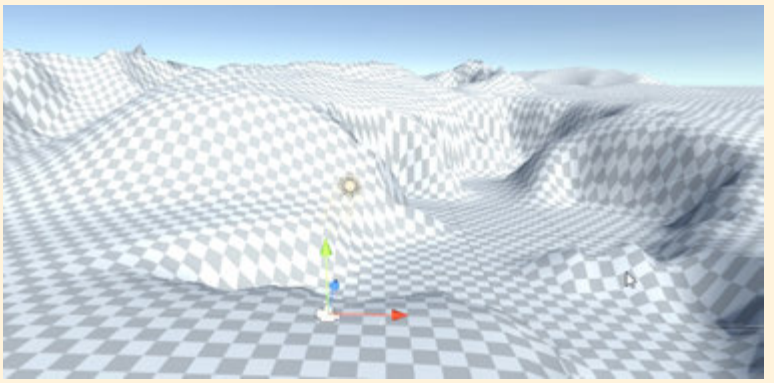
**Görsel 3.112:** Stamp Terrain ile yükseltiler oluşturma

NOT

Brush Strength, Size gibi değerler değiştirilerek Ctrl ve fare tekerleği ile damganın yüksekliği değiştirilebilir. Bazı fırçalarda Shift tuşu basılı iken çizim yapıldığında yeryüzü yumuşatılır. Ctrl basılı iken fırçalandığında tersine işlem yapar.

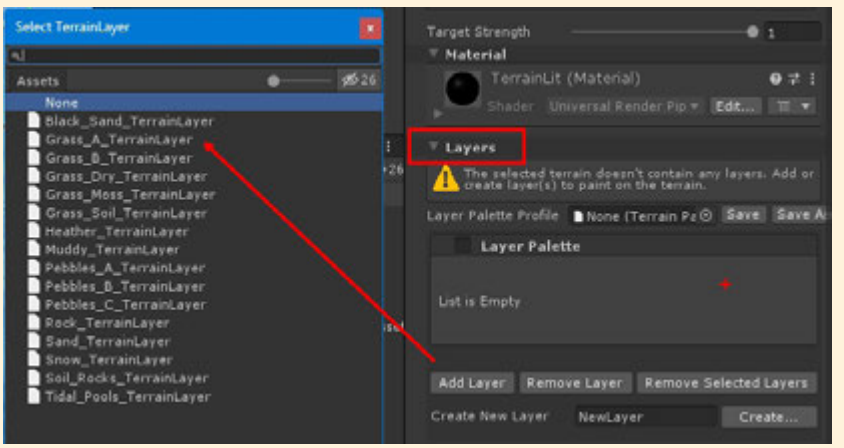
- **Erosion:** Yeryüzü üzerine su, ısı ve rüzgâr aşınması ekler.
- **Sculpt:** Fırça deseni ile yüzey üzerine yükselti veya alçaltı ekler.
- **Set Height:** Set Height Controls bölümünde yeryüzü tamamen yükseltilir veya alçaltılır.
- **Raise or Lower Terrain:** Fırça desenine göre yeryüzüne şekil eklenir veya silinir.
- **Transform:** Twist (döndürme), Pinch (çimdikleme) veya Smudge (bulaştırma) seçildiğinde fare hareketi ile yeryüzüne şekil verilir.
- **Effects:** Slope Flatten (eğim düzleştirme), Sharpen Peaks (tepeleri sivriltme), Contrast (zıtlık) ile düzleştirme ve düzeltmeler yapılır.
- **Paint Holes:** Yeryüzünde delikler açar. Ctrl ile basılarak delikler silinir.
- **Stamp Terrain:** Yeryüzünde fırça şekline göre yüzey çeşitleri elde edilir.
- **Smooth Height:** Keskin yeryüzü şekilleri seçilen fırça çeşitleri ile yumuşatılır.
- **Paint Texture:** Doku boyama ile çimen, kum ve toprak dokuları boyanır.

**5. Adım:** Görsel 3.113'te görüldüğü gibi yeryüzü şekillendirmesini yapınız.



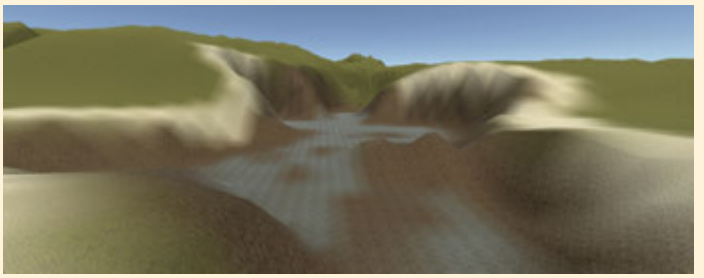
**Görsel 3.113:** Yeryüzü şekillendirmesi

**6. Adım:** Görsel 3.114'te görüldüğü gibi Paint Texture seçeneğinde **Layers** bölümüne gelerek katmanlar ekleyiniz. İlk eklenen doku katmanı varsayılan olarak tüm yeryüzünü kaplar. Terrain Sample Assets kurulu olduğu için doku örneklerini görebilirsiniz.



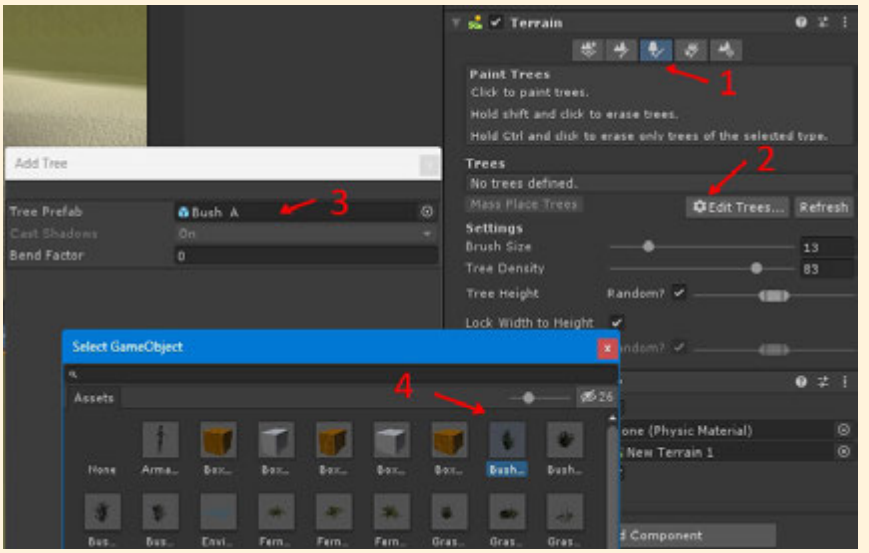

**Görsel 3.114:** Doku katmanlarının eklenmesi

**7. Adım:** Görsel 3.115'te görüldüğü gibi önceki adımda eklenen diğer katman seçili iken istenen yerleri fırça ile kumlu, taşlı ve yosunlu gibi dokularla boyayınız. Örneğin kum katmanı seçili iken Brush Size değeri ve fırça tipi seçilerek kum olan bölgeler boyanarak belirtilebilir.



**Görsel 3.115:** Doku ile boyama

**8. Adım:** Görsel 3.116'da görüldüğü gibi ağaç, çimen ve benzeri nesneleri ekleyiniz.




**Görsel 3.116:** Ağaç ve çimen gibi nesnelerin eklenmesi

Yeryüzüne eklenecekler tek tek konulmak yerine fırça ile boyama yaparak üretilir. Ağaç ve çimenler rüzgârda hareket eder. İstenen bölgedeki nesneler Shift basılı tutularak silinir.

**9. Adım:** Görsel 3.117'de sahnede doğada olabilecek nesneler görülür. **TerrainDemoScene** sahnesini açıp inceleyiniz.



**Görsel 3.117:** Örnek sahnenin görünümü

Ağaçlar, çimenler, otlar ve taşlar doğal bir çevre ortamı oluşturur. Proje türünde **URP** (Universal Render Pipeline) ve **HDRP** (High Definition Render Pipeline) arasından projenizin büyüklüğüne göre seçim yapılır. Her iki yöntemde de görünüm olarak güzel sahneler yapılır. URP yöntemi grafik kartını çok zorlamazken HDRP yöntemi ile maksimum kalite hedeflenir.

### 3.3.6. Diğer Oyuncular İle İletişim ve Diyalog Pencereleri Yapımı

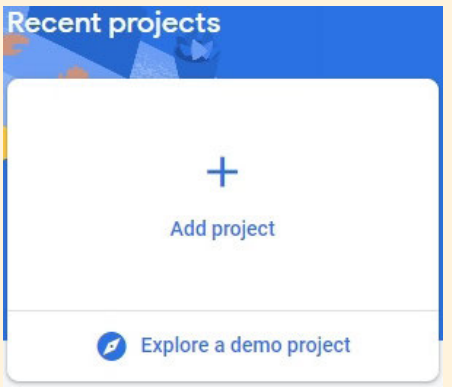
Oyun içinde iletişim **bulut veri tabanı** desteği ile sağlanır. Uygulamadaki proje için veri tabanı yönetim sayfasına giriş yapılarak bir Firestore veri tabanı oluşturulur. Bu veri tabanından gelen bilgiler oyunu açan kişiler tarafından görülür ve aynı veri havuzuna mesaj gönderilir.

Çok oyunculu olarak yapılan projelerde kullanıcı etkileşimi sohbet ekranı ile sağlanır. Mobil, akıllı televizyon ve oyun konsollarında ekran klavyesi desteği olduğu için yazı yazma kolaylığı vardır. Kullanıcı etkileşiminin olduğu yerlerde **güvenlik önlemlerinin** alınması ve kişisel verilerin korunması gereklidir. Sohbette yazılanlar içinde küfür gibi istenmeyen kelimeler var ise sohbetten bu kelimeler temizlenir.

#### 16 .UYGULAMA

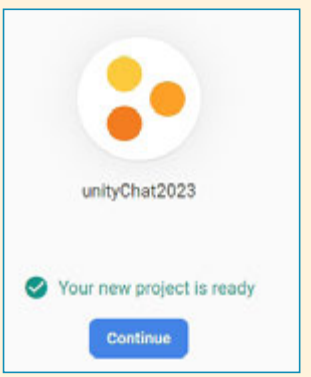
Yeni proje açarak veri tabanı oluşturunuz ve uygulamaya gerekli kullanıcı arayüzü nesnelerini ekleyiniz.

**1. Adım:** Görsel 3.118'de görüldüğü gibi veri tabanı ana sayfasını açarak Firebase Console içinde yeni proje oluşturunuz.



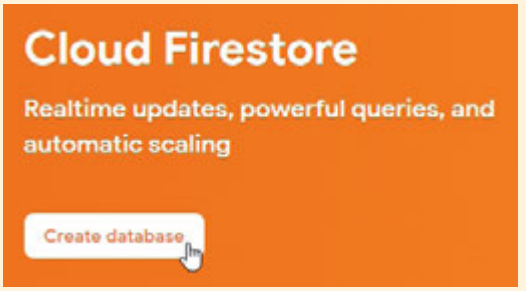

**Görsel 3.118:** Yeni proje oluşturma

**2. Adım:** Projeye isim veriniz. Görsel 3.119'da görüldüğü gibi web uygulamasındaki proje hazırdır.



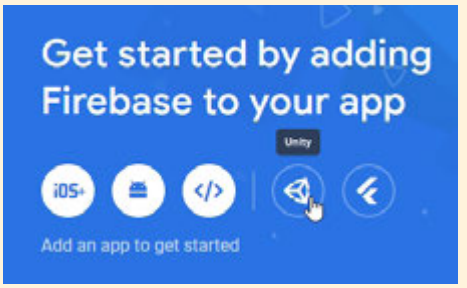

**Görsel 3.119:** Oluşturulan veri tabanı projesi

**3. Adım:** Görsel 3.120'de görüldüğü gibi **Cloud Firestore** veri tabanını oluşturunuz.




**Görsel 3.120:** Veri tabanı oluşturma

**4. Adım:** Görsel 3.121'de görüldüğü gibi veri tabanı projesine oyun motoru desteğini ekleyiniz.



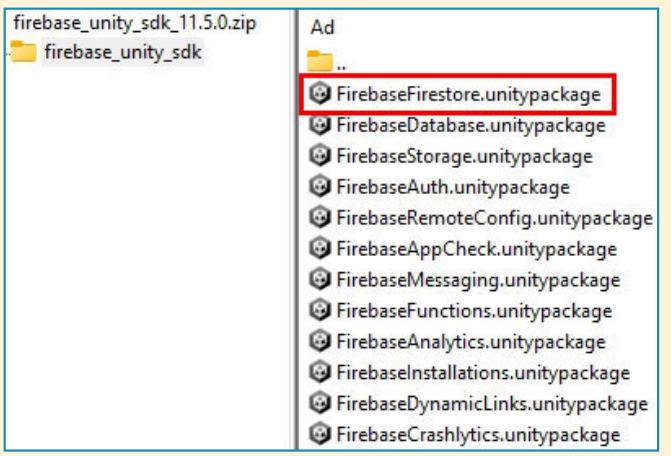

**Görsel 3.121:** Veri tabanı projesine oyun motoru desteğinin eklenmesi

**5. Adım:** Ekrana gelen sayfadaki dört adımı uygulayınız. **google-services.json** ve **SDK** dosyalarını indiriniz.

NOT

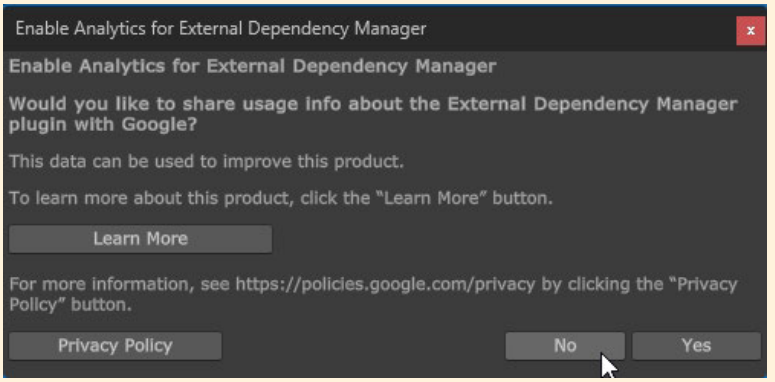
SDK (Software Development Kit) yazılım geliştirme seti anlamına gelir.

**6. Adım:** Görsel 3.122'de görüldüğü gibi indirilen sıkıştırılmış dosyanın içinden **FirebaseFirestore** ile başlayan dosyaya dosyasına çift tıklayıp oyun projesine ekleyiniz.



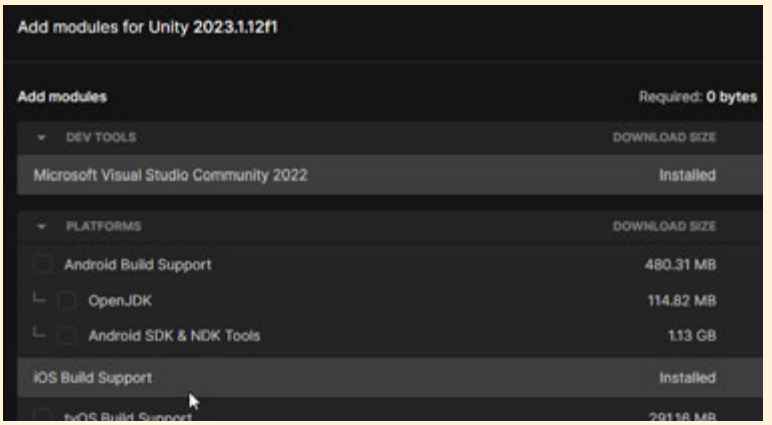

**Görsel 3.122:** SDK dosyası içinden gerekli paketin açılması

**7. Adım:** Görsel 3.123'te görüldüğü gibi oyun projesine paket eklendiğinde bilgi paylaşımı ile ilgili bir uyarı gelir.



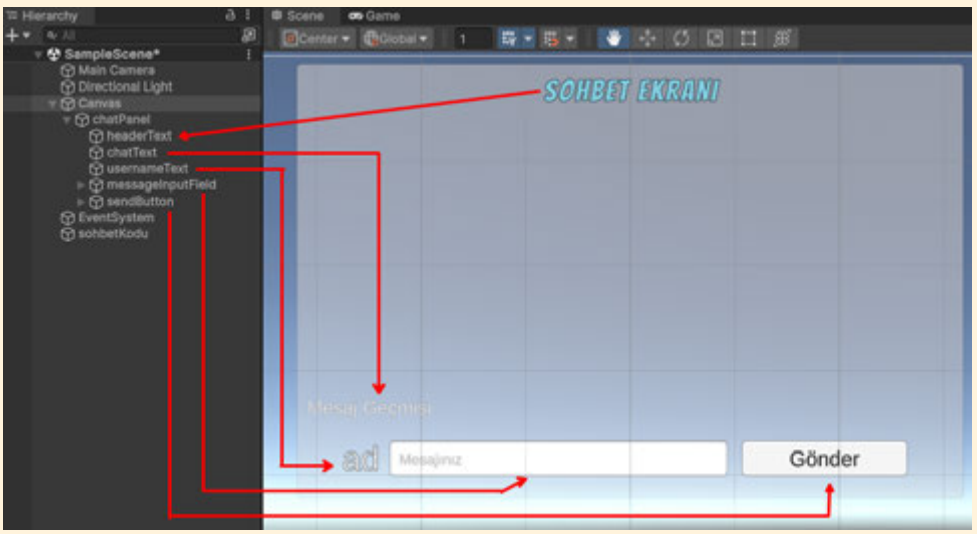

**Görsel 3.123:** Analiz için bilgi paylaşım isteği ile ilgili uyarı

**8. Adım:** Görsel 3.124'te görüldüğü gibi oyun motoruna ilgili derleme desteğinin eklenmesini yapınız.




**Görsel 3.124:** Oyun motoruna derleme desteğinin eklenmesi

**9. Adım:** Görsel 3.125'te görüldüğü gibi oyun sahnesindeki **Canvas** nesnesine gerekenleri ekleyiniz.




**Görsel 3.125:** Oyun sahnesindeki gerekli nesneler

**10. Adım:** Sohbet kodunu düzenleyiniz. Aşağıdaki verilen kodları barındıran bir script dosyası oluşturunuz. İsimlendirmeye dikkat ediniz.

```
using Firebase;
using Firebase.Extensions;
using Firebase.Firestore;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using System;
using UnityEngine.UI;
public class Chat : MonoBehaviour
{
    private FirebaseFirestore db;
    private ListenerRegistration chatListener;
    public TextMeshProUGUI userNameText;
    public TextMeshProUGUI chatText;
    public TMP_InputField messageText;
    public Button submitButton;
    public int messageCount = 10;
    public string ad = "birOyuncu";
    void Start()
    {
     FirebaseApp.CheckAndFixDependenciesAsync().
ContinueWithOnMainThread(task =>
        {
            FirebaseApp app = FirebaseApp.DefaultInstance;
            db = FirebaseFirestore.DefaultInstance;
            StartChatListener();
        });
        userNameText.text = ad;
    }
    void StartChatListener()
    {
        CollectionReference chatCollection =
db.Collection("chatTable");
        chatListener = chatCollection.OrderBy("timeStamp").
Limit(messageCount)
            .Listen((snapshot) =>
            {
                List<ChatMessage> chatMessages = new
List<ChatMessage>();
                foreach (DocumentSnapshot document in snapshot.
Documents)
                {
                    Dictionary<string, object> data = document.
ToDictionary();
if (data["timeStamp"] == null || data["name"] == null ||
data["message"] == null)
```

```
 continue;
                    Timestamp timestamp = (Timestamp)
data["timeStamp"];
                    string name = (string)data["name"];
                    string message = (string)data["message"];
  chatMessages.Add(new ChatMessage(name, message, timestamp.
ToDateTime()));
                }
                UpdateChatUI(chatMessages);
            });
    }

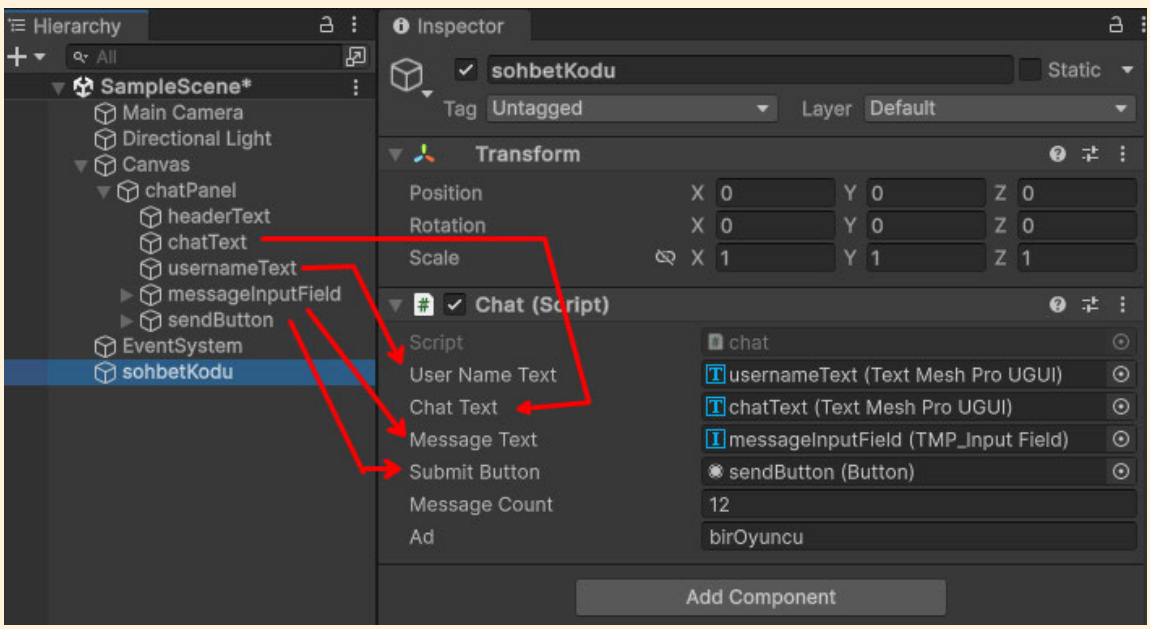
    public void SendChatMessage()
    {
        string userName = userNameText.text;
        string message = messageText.text;
        if (message == "") return;
        Dictionary<string, object> newMessage = new
Dictionary<string, object>
        {
            { "timeStamp", FieldValue.ServerTimestamp },
            { "name", userName },
            { "message", message}
        };

        db.Collection("chatTable").AddAsync(newMessage)
            .ContinueWithOnMainThread(task =>
            {
                if (task.IsCanceled || task.IsFaulted)
                {
                    Debug.LogError("Error sending message: " +
task.Exception);
                }
                else if (task.IsCompleted)
                {
                    messageText.text = "";
                    messageText.Select();
                    messageText.ActivateInputField();
                }
            });
    }

    void UpdateChatUI(List<ChatMessage> messages)
    {
        chatText.text = "";
        foreach (ChatMessage message in messages)
        {
            chatText.text += $"<b>{message.senderName}</b> (" +
                string.Format("{0:00}:{1:00}:{2:00}",
```

```
 message.timestamp.Hour,
                message.timestamp.Minute,
                message.timestamp.Second) +
                $"): {message.messageText}\n";
        }
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Return) ||
            Input.GetKeyDown(KeyCode.KeypadEnter))
            SendChatMessage();
    }
}
public class ChatMessage
{
    public string senderName;
    public string messageText;
    public DateTime timestamp;
    public ChatMessage(string senderName, string messageText,
DateTime timestamp)
    {
        this.senderName = senderName;
        this.messageText = messageText;
        this.timestamp = timestamp;
    }
}
```

**11. Adım:** Görsel 3.126'da görüldüğü gibi **sohbetKodu** adı verilen boş bir oyun nesnesini ekleyip üzerine `Chat` kodunu atayınız. Koddaki nesnelerin içine sürükle bırak ile metin nesnelerini atayınız.




**Görsel 3.126:** Kod içindeki boşluklara nesnelerin atanması

**12. Adım:** Görsel 3.127'de görüldüğü gibi projeyi test ediniz.




**Görsel 3.127:** Projenin test edilmesi

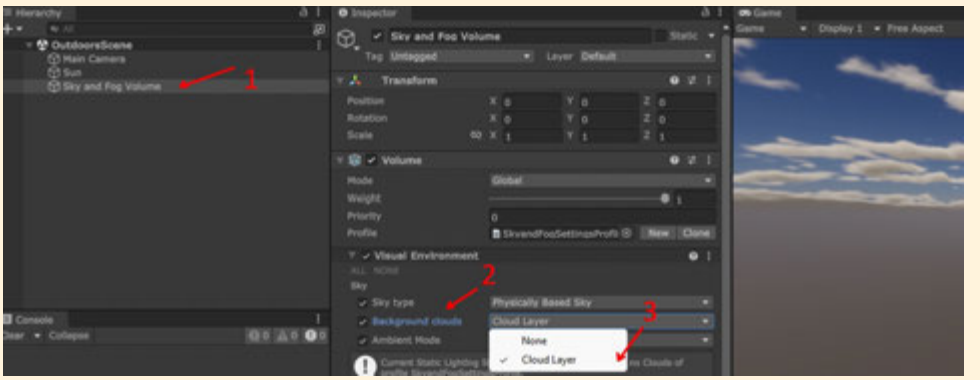
## 3.3.7. Gece ve Gündüz Gibi Zaman İlerlemesi

Oyun çevresini görsel olarak zenginleştirmek için güneş, ay, yıldız ve bulut gibi ögeler kullanılabilir. Güneş ve ayın hareket etmesi oyuncunun zamanın ilerlediğini daha etkili bir şekilde hissetmesine yardımcı olabilir.

### 17. UYGULAMA

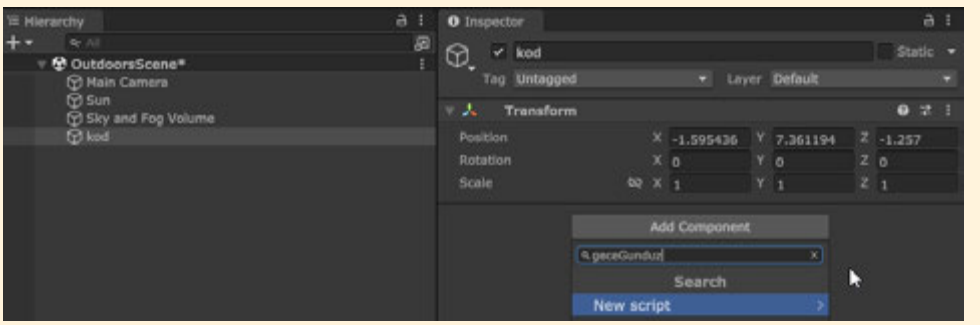
**HDRP** destekli bir proje açarak gerekli nesneleri ekleyiniz.

**1. Adım:** Görsel 3.128'de görüldüğü gibi **Sky and Fog Volume** nesnesine bulut katmanını ekleyiniz.



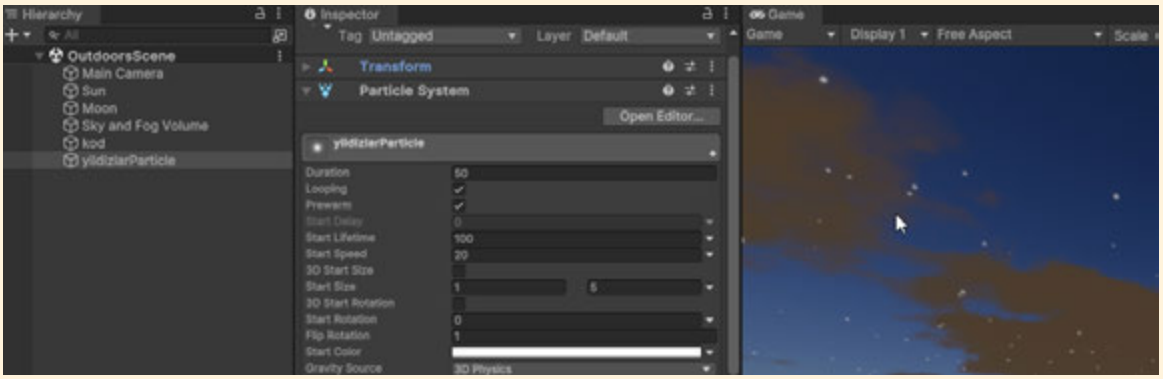

**Görsel 3.128:** Bulut katmanının eklenmesi

**2. Adım:** Görsel 3.129'da görüldüğü gibi boş bir nesne oluşturarak geceGunduz kodunu ekleyiniz.



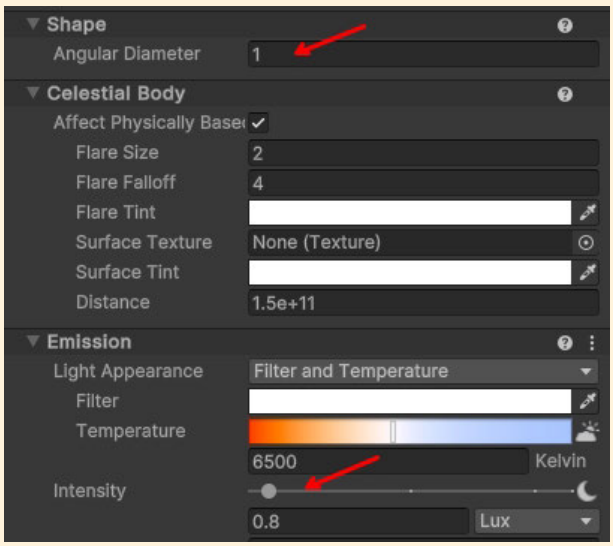

**Görsel 3.129:** Kod nesnesi içine geceGunduz kodunun eklemesi

**3. Adım:** Görsel 3.130'da görüldüğü gibi **yıldızlar** için boş bir nesne oluşturup Particle System ekleyiniz.



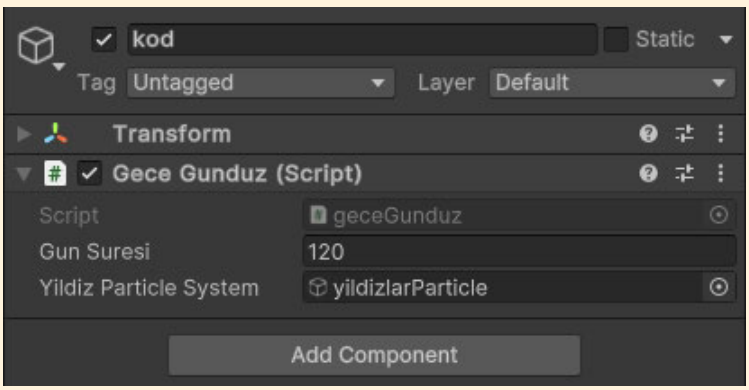

**Görsel 3.130:** Partikül sistemi ile yıldızların eklenmesi

**4. Adım:** Sun (güneş) nesnesini Ctrl+D kısa yolu ile çoğaltıp Moon (ay) nesnesini oluşturunuz. Görsel 3.131'deki gibi özelliklerini düzenleyiniz. Her iki nesnenin adları kodda geçeceği için **Sun** ve **Moon** olarak kalmalıdır.




**Görsel 3.131:** Ay nesnesinin değerlerini değiştirmek

**5. Adım:** Verilen kodu düzenleyiniz ve Görsel 3.132'deki gibi değerlerini ayarlayınız.

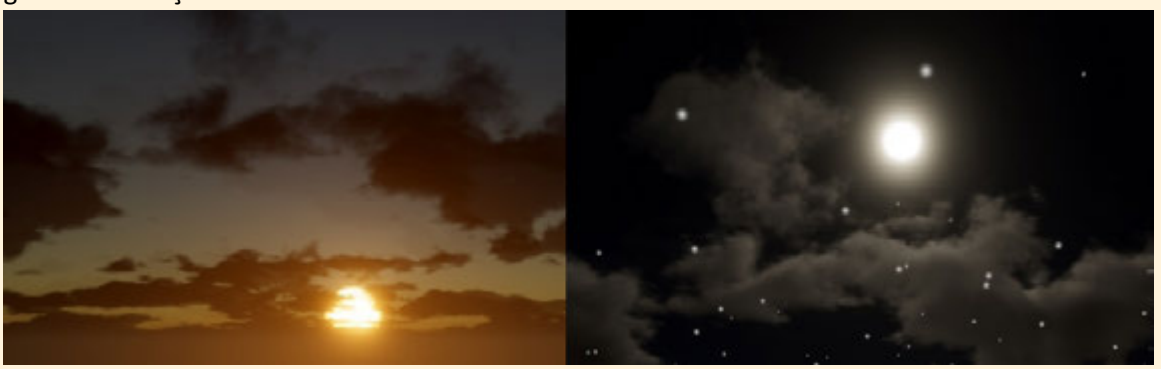



**Görsel 3.132:** Gece gündüz kodundaki özellikler

Gece gündüz değişimi için gereken kod şu şekildedir:

```
using UnityEngine;
public class geceGunduz : MonoBehaviour
{
    public float gunSuresi = 120.0f;
    public GameObject yildizParticleSystem;
    private Transform sunTransform;
    private Transform moonTransform;
    private float gunZamani = 0.0f;
    void Start()
    {
        sunTransform = GameObject.Find("Sun").transform;
        moonTransform = GameObject.Find("Moon").transform;
    }
    void Update()
    {
        gunZamani += Time.deltaTime / gunSuresi;
        gunZamani %= 1.0f;
   sunTransform.rotation = Quaternion.Euler(gunZamani * 360f, 0f,
0f);
   moonTransform.rotation = Quaternion.Euler(gunZamani * 360f +
180f, 0f, 0f);
        yildizParticleSystem.SetActive(gunZamani > 0.55f &&
gunZamani < 0.99f);
    }
}
```

**6. Adım:** Görsel 3.133'te görüldüğü gibi yıldız ve bulutların özellikleri düzenlendiğinde gündüz ve gece etkisi oluşur.



**Görsel 3.133:** Çalışma zamanında gündüz ve gecenin görünümleri

### 3.3.8. Şehir Nesnelerini Ekleme

Oyun sahnesini günlük hayata yakın bir ortam hâline getirmek için binalar, yollar, trafik lambaları gibi nesneler sahneye eklenir. Asset Store içindeki kütüphanelerden hazır nesneler seçilir veya modellenen nesneler kullanılır.

## 18. UYGULAMA

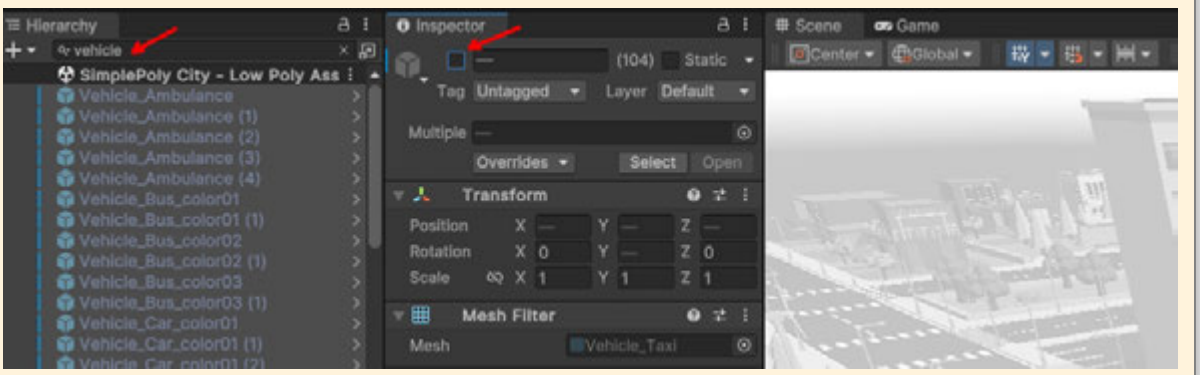
Şehir yapımı için boş bir proje açarak **SimplePoly City - Low Poly Assets** nesnelerini ekleyiniz.

**1. Adım:** Görsel 3.134'te görüldüğü gibi SimplePoly City - Low Poly Assets_Demo Scene'i açınız.



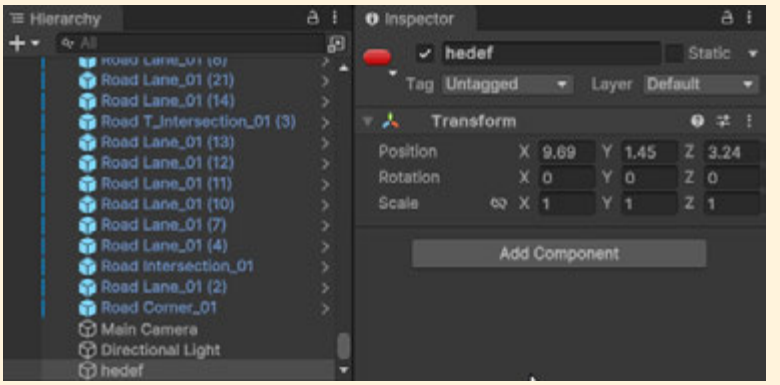

**Görsel 3.134:** SimplePoly City - Low Poly Assets_Demo Scene açılır.

**2. Adım:** Görsel 3.135'te görüldüğü gibi tüm araçları seçerek gizleyiniz.



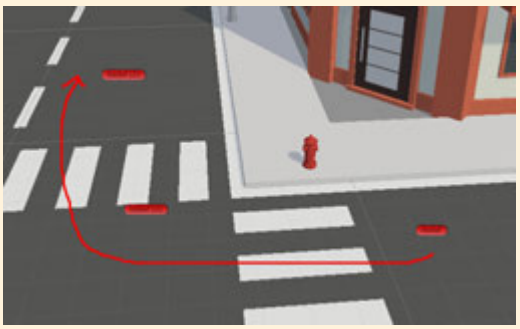

**Görsel 3.135:** Vehicle ismindeki nesnelerin görünürlüğü iptal edilir.

**3. Adım:** Görsel 3.136'da görüldüğü gibi kameraya yakın bir konuma **hedef** adında bir nesne oluşturunuz.




**Görsel 3.136:** Boş bir nesne eklenir ve adı hedef olarak düzenlenir.

**4. Adım:** Görsel 3.137'de görüldüğü gibi hedef nesnelerini çoğaltarak sahnede istenen yerlere ekleyiniz.



**Görsel 3.137:** Daha fazla hedef nesneleri eklenir.

**5. Adım:** Görsel 3.138'de görüldüğü gibi hedef nesnelerine kodu ekleyiniz. Her birinin sonraki hedefine bir sonraki hedef nesnesini seçerek atayınız.



**Görsel 3.138:** Hedef kodlarının eklenmesi

Sonraki hedef için gereken kodlar şunlardır:

```
using UnityEngine;
public class sonrakiHedef : MonoBehaviour
{
    public GameObject sonraki;
    public void OnDrawGizmos()
    {
        Gizmos.color = Color.red;
        if (sonraki != null)
            Gizmos.DrawLine(transform.position, sonraki.
transform.position);
    }
}
```

**6. Adım:** Görsel 3.139'da görüldüğü gibi **Prefab** klasörü içindeki bir aracı seçip sahneye ekleyiniz.



**Görsel 3.139:** Bir araç eklenmesi

**7. Adım:** Görsel 3.140'ta görüldüğü gibi araca Box Collider ve Rigidbody bileşenlerini ekleyiniz.



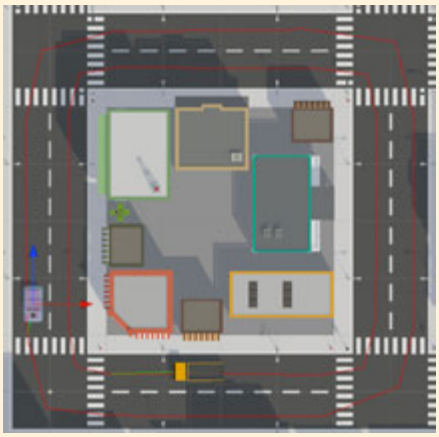

**Görsel 3.140:** Araç için gereken bileşenler

Aracın hareket etmesi için gereken kodlar şunlardır:

```
using UnityEngine;
public class hareketEt : MonoBehaviour
{
    public Transform hedef;
    public float aracHizi = 10f;
    public float donusHizi = 3.5f;
    private Rigidbody rb;
    private void Start()
    {
        rb = GetComponent<Rigidbody>();
    }
    void FixedUpdate()
    {
        Vector3 direction = (hedef.position - transform.
position).normalized;
        rb.velocity = direction * aracHizi;
        Quaternion targetRotation = Quaternion.
LookRotation(direction);
        transform.rotation = Quaternion.Slerp(transform.rotation,
targetRotation, Time.deltaTime * donusHizi);
        if (Vector3.Distance(transform.position, hedef.position)
< 2f)
        {   //Hedefe ulaşıldı ise sonraki hedef seçilir.
            if (hedef.gameObject.GetComponent<sonrakiHedef>().
sonraki)
                hedef = hedef.gameObject.
GetComponent<sonrakiHedef>().sonraki.transform;
            else
                rb.velocity = Vector3.zero;
            //Sonraki hedef yok ise araç durur.
        }
```

```
}
    public void OnDrawGizmos()
    {
        Gizmos.color = Color.green;
        if (hedef != null)
            Gizmos.DrawLine(transform.position, hedef.position);
    }
}
```

**8. Adım:** Görsel 3.141'de görüldüğü gibi başka bir araç ve hedefler ekleyip hareket etmesini sağlayınız.



**Görsel 3.141:** İki araç eklenmesi ve hedeflerin düzenlenmesi

### 3.3.9. Hava, Kara ve Su Araçlarını Kullanma

Oyun sahnesine havada, karada ve suda hareket eden araçlar eklenerek oyun zenginleştirilir. Araç kullanan oyuncu büyük bir sahnede hızlı bir şekilde istediği bir hedefe gider. Her araç türünün kendine has kullanımı olması nedeniyle oynanış çeşitliliği artar.

19. UYGULAMA

Boş bir proje açarak **Oyun Motoru için Standard Varlıklar** nesnelerini ekleyiniz. "Oyun Motoru için Standart Varlıklar" Asset Store içinde bulunmamaktadır.
Bu nesneleri https://cdnmeslek.eba.gov.tr/bilisim/oyun/sp.rar adresinden indiriniz.

**1. Adım:** Görsel 3.142'de görüldüğü gibi indirmiş olduğunuz paketi projenizin içine aktarınız.




**Görsel 3.142:** Gerekli nesnelerin aktarılması

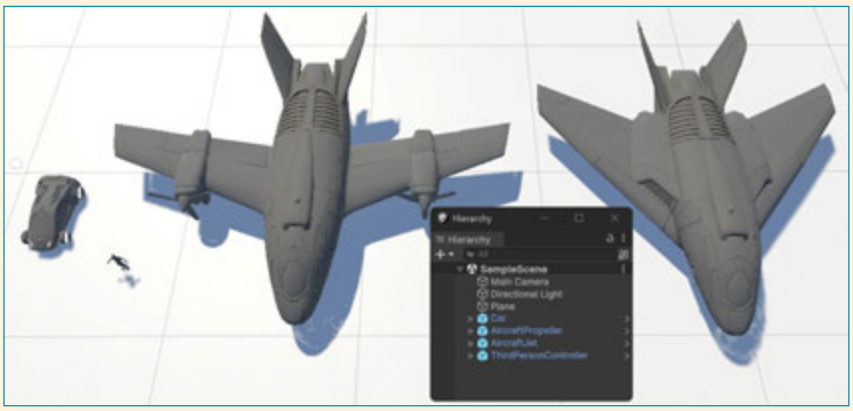
**2. Adım:** Görsel 3.143'te görüldüğü gibi hata mesajı veren SimpleActivatorMenu kodunu düzenleyiniz.

```
using System;
using UnityEngine;
using UnityEngine.UI;          //Gerekli kütüphane

#pragma warning disable 618
namespace UnityStandardAssets.Utility
{
    Unity Script (1 asset reference) | 0 references
    public class SimpleActivatorMenu : MonoBehaviour
    {
        // An incredibly simple menu which, when given references
        // to gameobjects in the scene
        public Text camSwitchButton; //GUIText yerine Text yazılır.
        public GameObject[] objects;
```

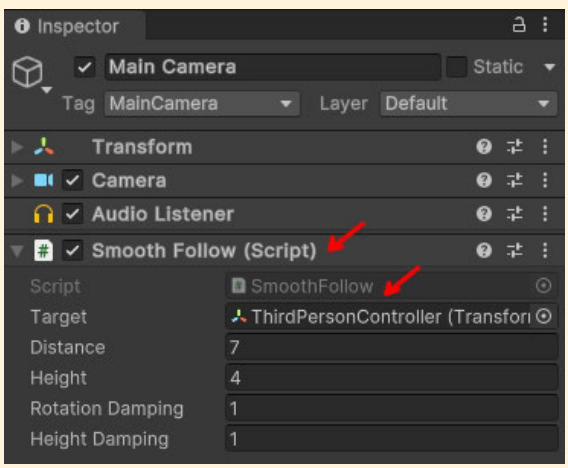
**Görsel 3.143:** SimpleActivatorMenu kodu düzenlenmesi

**3. Adım:** Görsel 3.144'te görüldüğü gibi sahneye Car, AircraftPropeller, AircraftJet ve ThirdPersonController nesnelerini ekleyiniz.



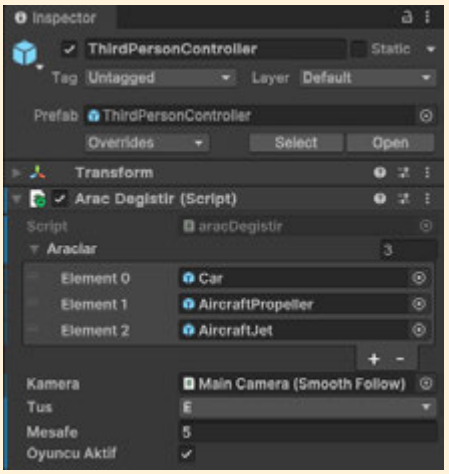
**Görsel 3.144:** Sahneye gerekli nesnelerin eklenmesi

**4. Adım:** Görsel 3.145'te görüldüğü gibi ana kamera nesnesine SmoothFollow kodunu ekleyiniz ve özelliklerini düzenleyiniz.




**Görsel 3.145:** Ana kameraya SmoothFollow kodunun eklenmesi

**5. Adım:** Görsel 3.146'da görüldüğü gibi araçlar arasında geçiş olması için gereken kodu yazınız ve ThirdPersonController nesnesine ekleyiniz.




**Görsel 3.146:** Araç değiştirme kodunun eklenmesi

Araç değiştirme kodunu oluşturunuz:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityStandardAssets.Characters.ThirdPerson;
using UnityStandardAssets.Utility;
using UnityStandardAssets.Vehicles.Aeroplane;
using UnityStandardAssets.Vehicles.Car;
public class aracDegistir : MonoBehaviour
{
    public GameObject[] araclar;    //Oyundaki araçlar
    public SmoothFollow kamera;
    public KeyCode tus = KeyCode.E;
    public float mesafe = 5f;
    //Oyuncunun önündeki nesneye olan mesafe
    public bool oyuncuAktif = true;

    void Update()
    {
        if (!oyuncuAktif && Input.GetKeyDown(tus))
        {   //Araçtan çıkma durumudur.
            oyuncuAktif = true;
            GetComponent<ThirdPersonUserControl>().enabled =
true;
            GetComponent<Animator>().enabled = true;
            araclar[0].GetComponent<CarUserControl>().enabled =
false;
            araclar[1].GetComponent<AeroplaneUserControl4Axis>().
enabled = false;
            araclar[2].GetComponent<AeroplaneUserControl2Axis>().
enabled = false;
            kamera.target = gameObject.transform;
            //Kamera tekrar oyuncuda olur.
        }
```

```
Ray isin = new(new Vector3(transform.position.x,
     transform.position.y + 1, transform.position.z), transform.
forward);
        if (Physics.Raycast(isin, out RaycastHit playerHit,
mesafe) &&
           (playerHit.transform.gameObject.CompareTag("araba") ||
            playerHit.transform.gameObject.CompareTag("ucak1") ||
            playerHit.transform.gameObject.CompareTag("ucak2")))
        {
            if (Input.GetKeyDown(tus))
            {   //Bir araç seçili ve tuş basıldı ise çalışır.
                if (playerHit.transform.gameObject.
CompareTag("araba"))
                {
                    araclar[0].GetComponent<CarUserControl>().
enabled = true;
                    GetComponent<ThirdPersonUserControl>().
enabled = false;
                    GetComponent<Animator>().enabled = false;
                    oyuncuAktif = false;
                    kamera.target = araclar[0].gameObject.
transform;
                }
                if (playerHit.transform.gameObject.
CompareTag("ucak1"))
                {
             araclar[1].
GetComponent<AeroplaneUserControl4Axis>().enabled = true;
                    GetComponent<ThirdPersonUserControl>().
enabled = false;
                    GetComponent<Animator>().enabled = false;
                    oyuncuAktif = false;
                    kamera.target = araclar[1].gameObject.
transform;
                }
                if (playerHit.transform.gameObject.
CompareTag("ucak2"))
                {
            araclar[2].GetComponent<AeroplaneUserControl2Axis>().
enabled = true;
                   GetComponent<ThirdPersonUserControl>().enabled
= false;
                    GetComponent<Animator>().enabled = false;
                    oyuncuAktif = false;
                  kamera.target = araclar[2].gameObject.transform;
                }
            }
        }
    }
}
```

**6. Adım:** Görsel 3.147'de görüldüğü gibi oyunu test ediniz.



**Görsel 3.147:** Oyunun test edilmesi

## 3.4. SES EKLEME İŞLEMLERİ

Oyun içerisinde arka planda müziğin çalması ve çeşitli aksiyonlar sonucunda tepki olarak ses efektlerinin duyulması kullanıcı deneyimini geliştirebilir. Sadece görsel geri bildirimlerin alınması oyunun sıkıcı hâle gelmesine neden olabilir.

NOT

**Creative Commons Zero** (CC0) diğer adı ile **Public Domain** (kamu malı) olarak lisansları belirtilen ses ve müzik dosyaları proje içinde kaynak gösterilmeden kullanabilir.

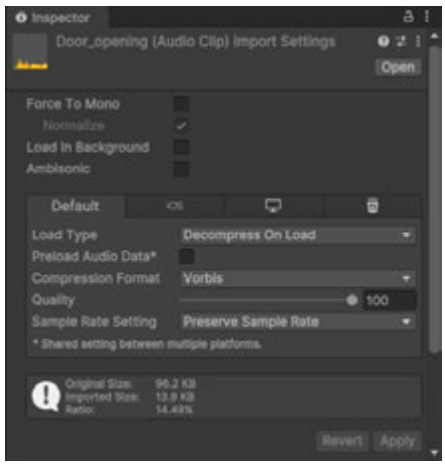
### 3.4.1. Ses Dosyaları Türleri ve Dönüşümleri

Oyun motorunda en fazla kullanılan ses dosya türleri şunlardır:

- **WAV** (Waveform Audio File):
  - Genellikle kayıpsız sıkıştırma kullanılır ve yüksek ses kalitesi sunar.
  - Oyun müzikleri için uygun olabilir.
- **MP3** (MPEG Audio Layer III):
  - Sıkıştırılmış bir ses dosyası biçimidir, bu nedenle ses kalitesini bazen kaybedebilir.
  - Özellikle uzun süreli müzikler için tercih edilebilir.
- **OGG** (Ogg Vorbis):
  - Kayıpsız sıkıştırma kullanabilen bir açık kaynak ses biçimidir.
  - Mobil ve web tabanlı oyunlar için yaygın olarak kullanılır.
- **AIFF** (Audio Interchange File):
  - Yüksek kaliteli kayıpsız ses dosyaları için kullanılır.
  - Özellikle profesyonel ses düzenleme gerektiren oyunlar için uygundur.

Oyun motorunda ses dosyalarının boyutları oyunun boyutu ve performansı üzerinde etkili olabilir, dosyaların sıkıştırma düzeyi ve kalitesini bu nedenle dikkatlice ayarlamak gerekir. Ses dosya türlerini başka bir dosya türüne dönüştürmek için çevrim içi sayfalar veya ücretsiz programlar bulunabilir.

Görsel 3.148'de görüldüğü gibi, seçili ses dosyasının özelliklerinde tek kanala dönüşmesi (Force To Mono), arka planda yüklenmesi (Load in Background), çevresel ses desteği (Ambisonic) ve sıkıştırma biçimi gibi ayarlar bulunur. Kalite ve sıkıştırma biçimi değiştirildiğinde ses ve müzik dosyasının yüklenme boyutu değişir.




**Görsel 3.148:** Ses dosyasının özellikleri

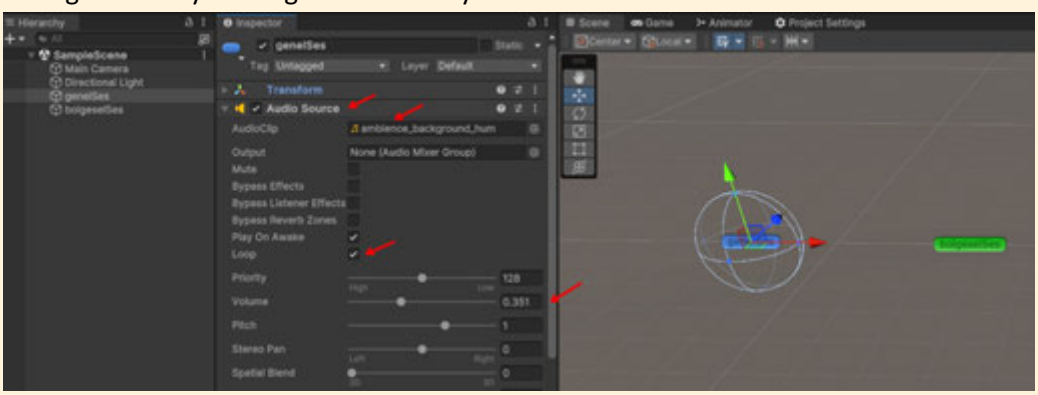
## 3.4.2. Etkileşimde Ses Etkileri

Oyuncunun etkileşimine ve oyundaki diğer nesnelerin durumuna göre ses eklenmesi oyunun daha etkili bir şekilde oynanmasını sağlar. Ses efektleri; oyuncuların oyuna daha fazla bağlanmalarına, oyun dünyasının daha gerçekçi hissedilmesine yardımcı olur ve oyunun atmosferini güçlendirir. Ses, oyun genelinde duyulabilecek şekilde düzenlenebilir veya sadece bir bölgede de duyulabilir.

### 20. UYGULAMA

Boş bir proje açarak gerekli nesneleri projeye ekleyiniz.

**1. Adım:** Görsel 3.149'da görüldüğü gibi iki adet boş oyun nesnesine **Audio Source** bileşeni ekleyiniz. Tüm sahnede duyulan genel ses için uygun bir ses seçiniz. **Loop** özelliği ile döngü hâlinde her zaman çalması sağlanır. Ses yüksekliği **Volume** ile ayarlanır.

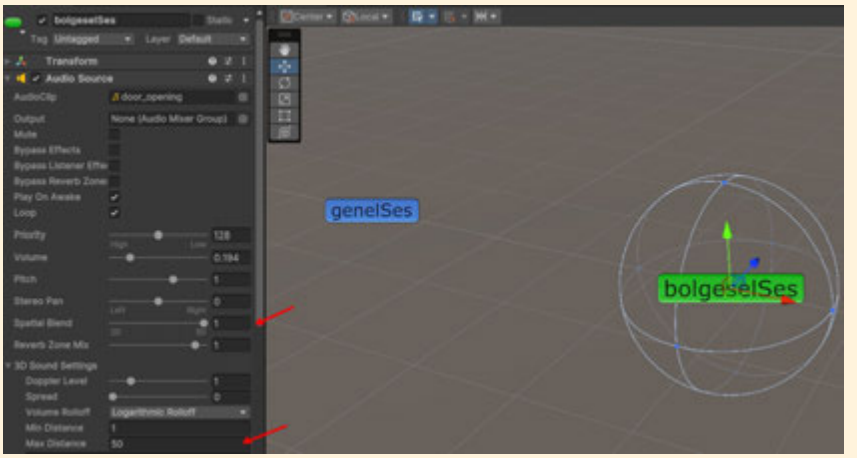



**Görsel 3.149:** Genel ses özelliklerinin düzenlenmesi

**2. Adım:** Görsel 3.150'de görüldüğü gibi bölgesel ses nesnesinin özelliklerini düzenleyiniz. **Spatial Blend** değeri 1 olduğunda ve **Max Distance** (oyuncunun yaklaşma mesafesi) değeri azaltıldığında istenen bölgede çalan ses oluşur. Bu ses Loop hâlinde olmak zorunda değildir.

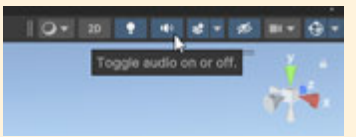
Volume Rolloff seçeneğinde şunlar seçilebilir:

- **Logarithmic Rolloff:** Sesin uzak mesafeden hızla azaldığı durumlar için uygundur ve doğal bir ses azalma efekti oluşturur.
- **Linear Rolloff:** Ses seviyesi doğrusal bir şekilde uzaklığa bağlı olarak azalır.
- **Custom Rolloff:** Ses seviyesi kullanıcı tanımlı olarak değişir.




**Görsel 3.150:** Bölgesel ses özelliklerinin düzenlenmesi

**3. Adım:** Görsel 3.151'de görüldüğü gibi sesler tasarım penceresinde test edilebilir.




**Görsel 3.151:** Tasarım penceresinde ses açma ve kapama

SIRA SİZDE

Sahnedeki oyun nesnelerine ses kaynağı eklemeyi gerçekleştiriniz. Yapım aşamalarındaki hataları not ediniz.

**Değerlendirme**

Çalışmanız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken **Kontrol Listesi**ndeki değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Boş sahne içine ses kaynağı için kullanılacak oyun nesnelerini ekledi. | | |
| 2. Nesneye Audio Source bileşeni ekledi. | | |
| 3. Audio Source bileşeninin ayarlarını uygun şekilde düzenledi. | | |
| 4. Projeyi çalıştırarak sesin duyulmasını test etti. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

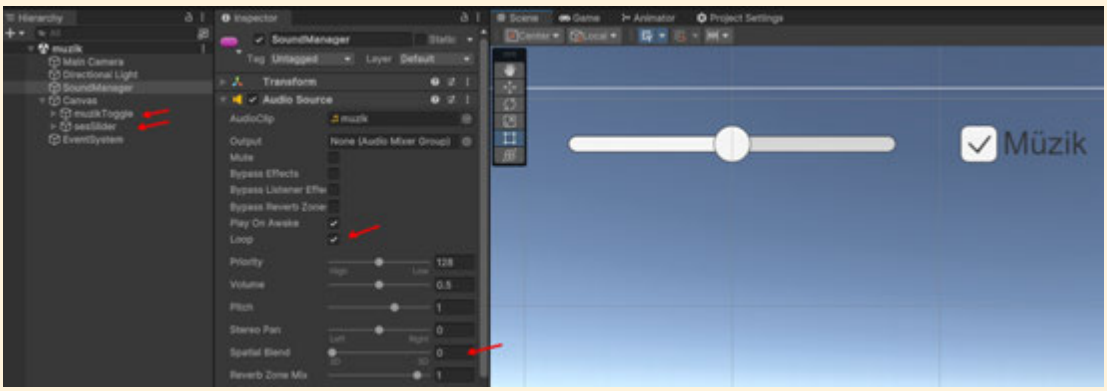
### 3.4.3. Arka Plana Müzik Ekleme

Oyun arka planında müzik olması oyunun daha eğlenceli olmasını sağlar. Oyunun aksiyonuna göre hızlı veya sakin bir müzik oyuna eklenir.

21. UYGULAMA

Boş bir proje açarak gerekli nesneleri projeye ekleyiniz. Oyuncunun istediği anda müziği kapatma ve ses seviyesini düzenleme seçeneklerini sağlayınız.

**1. Adım:** Görsel 3.152'de görüldüğü boş bir nesneye müzik dosyasını ekleyiniz. UI nesneleri olarak **Toggle** ve **Slider** ekleyiniz.

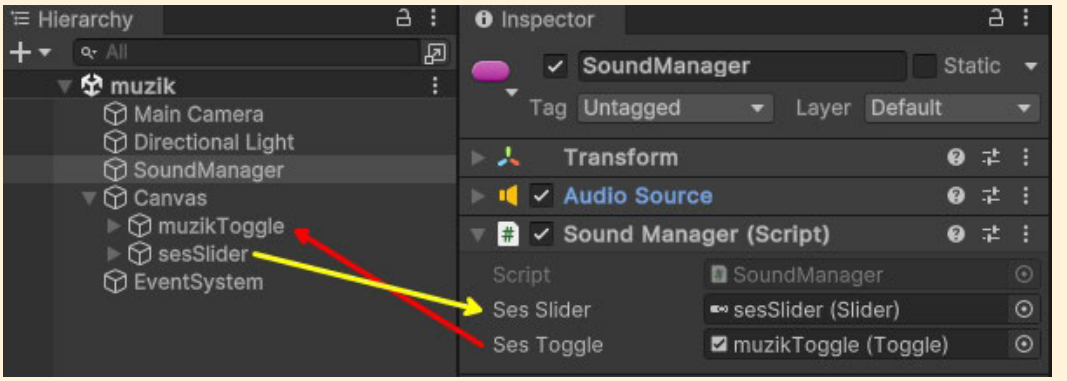



**Görsel 3.152:** Slider ve Toggle nesneleri eklenir.

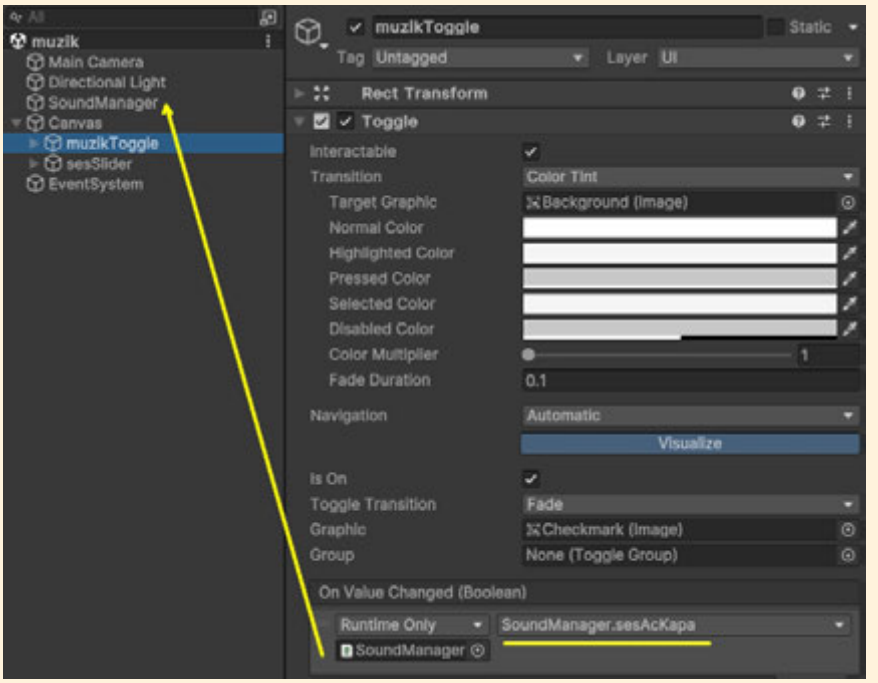
**2. Adım:** Müzik dosyasının bulunduğu nesneye SoundManager kodunu ekleyiniz. Bu kod ile ses durumu bilgileri kullanıcı tercihleri olarak kaydedilir.

```
using UnityEngine;
using UnityEngine.UI;
public class SoundManager : MonoBehaviour
{
    public Slider sesSlider;
    public Toggle sesToggle;
    private void Start()
    {
        sesSlider.value = PlayerPrefs.GetFloat("volume", 0.5f);
        sesToggle.isOn = PlayerPrefs.GetInt("music", 1) == 1 ?
true : false;
    }
    public void sesAcKapa()
    {
        GetComponent<AudioSource>().mute = !sesToggle.isOn;
        PlayerPrefs.SetInt("music", sesToggle.isOn ? 1 : 0);
    }
    public void sesDegeri()
    {
        GetComponent<AudioSource>().volume = sesSlider.value;
        PlayerPrefs.SetFloat("volume", sesSlider.value);
    }
}
```

**3. Adım:** Görsel 3.153 ve Görsel 3.154'te görüldüğü gibi kod içine gerekli nesneleri sürükleyiniz. muzikToggle ve sesSlider nesnelerinin olaylarını SoundManager nesnesine bağlayınız.




**Görsel 3.153:** Gerekli nesnelerin bağlanması




**Görsel 3.154:** Olayların eklenmesi

Ses dosyalarının GetComponent<AudioSource>() kodu ile çalıştırılması ile ilgili örnekler şu şekildedir:

- **Play():** Bu yöntem AudioSource bileşenine bağlı olan sesi oynatır. Özellikle uzun süren sesleri çalmak için kullanılır. Zaten çalan bir ses varsa baştan sesi oynatır.
- **PlayDelayed(float gecikme):** Bu yöntem sesin belirli bir gecikme sonrasında çalınmasını sağlar. Örneğin PlayDelayed(1f) komutu ile ses bir saniye sonra çalacaktır.
- **PlayOneShot(AudioClip sesKlibi):** Bu yöntem bir ses efektini bir kez çalmak için kullanılır. Zaten çalan bir ses varsa o ses kesintiye uğramaz.
- **Stop():** Bu yöntem sesin çalmasını durdurur.

**Örnek:** Çok sayıdaki ses efektlerini yönetebilmek için "ses yöneticisi" kodları örneğini inceleyiniz.

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
public class SesYoneticisi : MonoBehaviour
{
    //Ses ve müziğin kesintisiz olması için Singleton yöntemi kullanılır
    private static SesYoneticisi _instance;
    public static SesYoneticisi Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = FindObjectOfType<SesYoneticisi>();
                if (_instance == null)
                {
                    GameObject go = new
GameObject("SesYoneticisi");
                    _instance = go.AddComponent<SesYoneticisi>();
                }
            }
            return _instance;
        }
    }
    //Ses efektlerini depolamak için bir sözlük kullanılır
    private Dictionary<string, AudioSource> sesEfektleri = new
    Dictionary<string, AudioSource>();
    void Awake()
    {
        if (_instance != null && _instance != this)
        {
            Destroy(this.gameObject);
            return;
        }
        DontDestroyOnLoad(this.gameObject);
    }
}
```

## ÖLÇME VE DEĞERLENDİRME

**Aşağıdaki soruları dikkatlice okuyarak doğru cevabı işaretleyiniz.**

**1. Aşağıdakilerden hangisi animasyon yapımında doğrudan kullanılmaz?**

A) Animator B) Cinemachine C) HDRP

D) Kod yazımı E) Tween

**2. Aşağıdaki hangi kısa yol ile sahnedeki nesneler çoğaltılabilir?**

A) Ctrl+A B) Ctrl+C C) Ctrl+D

D) Ctrl+S E) Ctrl+V

**3. Aşağıda verilen hangi Tween hareketi sona doğru hızlanır?**

A) ease.InElastic
B) ease.Linear
C) ease.OutCubic
D) ease.OutQuad
E) ease.OutSine

**4. Aşağıda verilen hangi komut ile nesne geri doğru 5 birim hareket eder?**

A) transform.position -= Vector3.back * 5.0f;
B) transform.position += Vector3.forward * 5.0f;
C) transform.Translate(Vector3.back + 5f);
D) transform.Translate(Vector3.back * 5f);
E) transform.Translate(Vector3.forward * 5f);

**5. İnsan karakterleri oyun motoruna aktarılırken animasyon türü ne olarak seçilmelidir?**

A) Generic B) Humanoid C) Legacy

D) None E) Standard

**6. Aşağıdakilerden hangisi Animator parametresi olarak kullanılamaz?**

A) Bool B) Int C) Float

D) String E) Trigger

**7. Aşağıda verilen hangi Rigidbody özelliği ile nesne yere doğru düşer?**

A) Constraints B) Drag C) Is Kinematic

D) Mass E) Use Gravity

**8. Aşağıdaki etkilerden hangisi oyun motorunun fizik desteğine ihtiyaç duymaz?**

A) Kırılma B) Kumaş C) Ragdoll

D) Sis E) Yağmur

**9. Aşağıdaki nesnelerden hangisinin yerini değiştirmek gölgelerin yönünü etkilemez?**

A) Area B) Directional C) Light Cookies

D) Point E) Spot

**10. Aşağıdakilerden hangisi ile Terrain nesnesinde delik açılabilir?**

A) Erosion B) Lower C) Paint Holes

D) Raise E) Stamp

# 4. ÖĞRENME BİRİMİ



# Oyun Oluşturma ve Yayınlama

## KONULAR

4.1. YAPAY ZEKÂ VE ÇOKLU OYUNCU DESTEĞİ
4.2. TEST ETME VE İYİLEŞTİRME
4.3. DERLEME VE YAYINLAMA

## NELER ÖĞRENECEKSİNİZ?

- Oyuncu olmayan karakterlerin (NPC) yapay zekâ ile etkileşimlerini gösterme
- Yerel ağda birden fazla oyuncunun oynayabileceği oyun yapma
- İnternet üzerinden oynanabilen oyun yapma
- Alışveriş ve nesne yapma (crafting) sistemlerinin yapımı
- Oyunun oynanışını test etme
- Oyun içi reklam ve analiz desteği sağlama
- Oyun içi ve çevrim içi olarak yardım ekleme
- Oyunun bellek, grafik ve oynanış olarak iyileştirilmesi ile ilgili düzenlemeler yapma
- Derlenebilen platform türlerini (Windows, Linux, WebGL, Android, iOS) gösterme
- Artırılmış, Sanal ve Karma Gerçeklik (AR, VR, XR) türlerini öğrenme
- Uygulama marketlerine uygulama ekleme
- Güncelleme yapma ve yama ekleme

## ANAHTAR KELİMELER

NPC, VR, XR, AR, MR, Android, iOS, Windows, MacOS, Linux, WebGL

**HAZIRLIK ÇALIŞMALARI**

1. Oynadığınız çok oyunculu bir oyun var mı? Varsa bu oyun hakkında bildiklerinizi arkadaşlarınızla paylaşınız.
2. Oyun içine reklam eklemenin olumlu ve olumsuz yönleri sizce nelerdir? Düşüncelerinizi arkadaşlarınızla paylaşınız.

## 4.1. YAPAY ZEKÂ VE ÇOKLU OYUNCU DESTEĞİ

Oyun geliştiricileri; yapay zekâ tekniklerini kullanarak kendi kendine karar verebilen, yolunu bulabilen, duygularını ifade edebilen karakterler oluşturup daha gerçekçi oyunlar yaratabilirler. Günümüzde yapay zekâ destekli karakterlerin bulunduğu tek kişilik oyunlar hâlâ sıklıkla oynansa da çok oyunculu oyunlar tek kişilik oyunlardan daha fazla talep görmektedir çünkü insanlar doğası gereği rekabetçi ortamları sever ve çok oyunculu oyunlar bu ortamı onlara sağlar. Ayrıca yapay zekâ ile yönetilen karakterlerin yapabildikleri, oyun yapımcılarının hayal gücü ile sınırlı kalırken başka insanlara karşı oynamak farklı bir deneyim sağlamaktadır çünkü insanların hayal gücü sınırsızdır ve insanlar yapay zekâdan çok daha farklı davranışlar sergileyerek rekabetçi ortamın oluşmasına katkı sağlamış olurlar.

### 4.1.1. Oyuncu Olmayan Karakterlerin (NPC) Yapay Zekâ ile Etkileşimleri

**Oyuncu olmayan karakterler (NPC)**; herhangi bir oyuncu tarafından oynanmayan, yapay zekâ ile yönetilen karakterlere verilen isimdir. Bu, bir insan modellemesi ya da oyunun türüne göre hayvanlar, bitkiler hatta çeşitli nesneler bile olabilir.

Oyunlarda NPC karakterlere çok fazla ihtiyaç duyulur çünkü bu karakterler bize çeşitli görevler verebilir, oyun dünyasının gezilmesine ve senaryonun ilerlemesine yardımcı olabilir.

Oyuncu olmayan karakterlerin yollarını bulması için NavMesh sistemi kullanılır. Bu sistemle sahnedeki nesnelerin yerleşimine göre karakterlerin ulaşabileceği ve ulaşamayacağı yerler hesaplanır. NavMesh sistemi şu bileşenlerden oluşur:

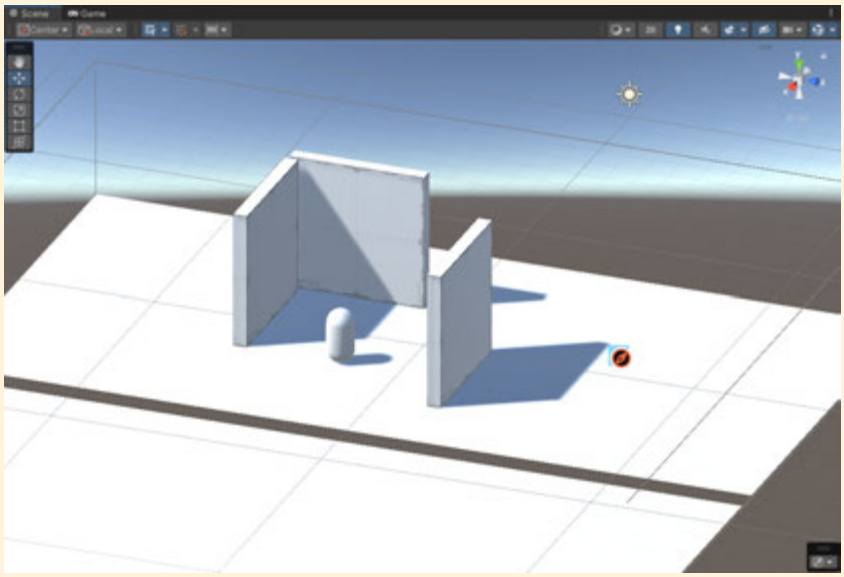
- **NavMesh:** Oyun dünyasındaki karakterlerin nerelere ulaşabileceğini gösteren, mavi ile işaretlenmiş alanlardır.
- **NavMeshAgent:** Bu bileşenin eklendiği karakterler, NavMesh'i kullanarak oyun dünyası hakkında akıl yürütürler ve hareket eden engellerin yanı sıra birbirlerinden de nasıl kaçacaklarını bilirler.
- **NavMeshObstacle:** Karakterlerin engel olarak algıladıkları alanlardır. Duvarlar, çitler vs. bunlara örnek olarak gösterilebilir.
- **NavMeshLink:** Karakterlerin bir çitin üstünden veya bir uçurumdan karşıya atlayabileceği alanlardır.

1. UYGULAMA

İşlem adımlarına göre NPC karakter ekleyip bu karakterin belirtilen koordinata giderken yolunu bulmasını sağlayınız.

**1. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.
**2. Adım: Window>Package Manager** ile gelen pencerede **AI Navigation** seçip **Install** düğmesine basarak projenize ekleyiniz.

**3. Adım: Game Object>3D Object** menüsü altında bulunan **Plane, Cube** ve **Capsule** nesnelerini kullanarak Görsel 4.1'deki sahneyi oluşturunuz.



**Görsel 4.1:** Sahne oluşturma

**4. Adım: Game Object>AI** menüsünden sahneye **NavMesh Surface** nesnesi ekleyiniz.
**5. Adım: Capsule** nesnesini seçip **Inspector** panelinden **Nav Mesh Agent** bileşenini ekleyiniz.
**6. Adım: Hierarchy** panelinden **NavMesh Surface** bileşenini seçip **Inspector** panelindeki **Bake** tuşuna basınız.
**7. Adım: Capsule** nesnesini seçip **Inspector** panelindeki **Add Component** butonuna basarak "hareket" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

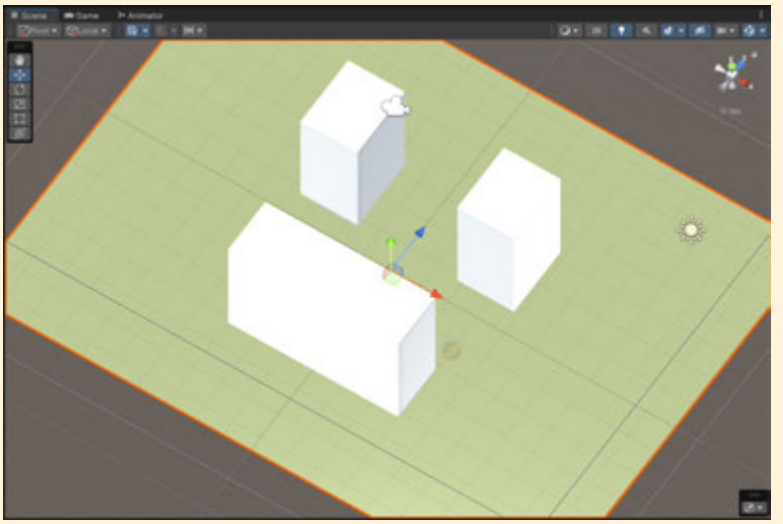
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class hareket : MonoBehaviour
{
    public Transform hedef;
    UnityEngine.AI.NavMeshAgent agent;
    void Start()
    {
        agent = GetComponent<UnityEngine.AI.NavMeshAgent>();
    }
    void Update()
    {
        agent.destination = hedef.position; //hedefe git
    }
}
```

**8. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **Create Empty** seçeneğini seçerek boş bir nesne ekleyiniz. Nesneyi gezilebilen alanın üzerinde istediğiniz bir noktaya taşıyınız.
**9. Adım: Capsule** nesnesini seçerek hareket scriptinin altında bulunan hedef değişkenine 8. adımda oluşturduğunuz boş nesneyi ekleyiniz.
**10. Adım:** Uygulamayı çalıştırdığınız zaman capsule nesnesi, belirtilen koordinatlara hareket edecektir.

## 2. UYGULAMA

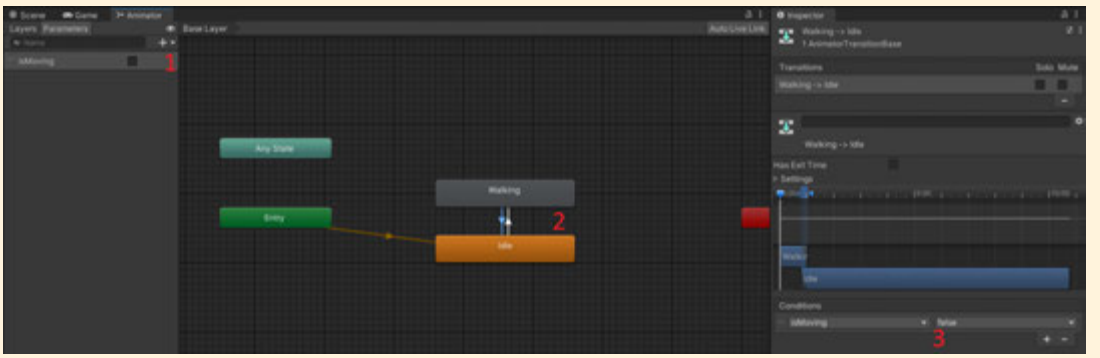
İşlem adımlarına göre oyuncu olmayan karakter (NPC) ekleyip, bu karakterin belirtilen koordinatlara giderek burada devriye atmasını sağlayınız.

**1. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.
**2. Adım: Window>Package Manager** ile gelen pencerede **AI Navigation** seçip, **Install** düğmesine basarak projenize ekleyiniz.
**3. Adım: Game Object>3D Object** menüsü altında bulunan **Plane** ve **Cube** nesnelerini kullanarak Görsel 4.2'deki sahneyi oluşturunuz.



**Görsel 4.2:** Sahne oluşturma

**4. Adım:** Oyun karakteri oluşturma sitelerinin birinden istediğiniz bir karakteri ve ona uygun Idle ve Walking animasyonunu indirerek projenize dâhil ediniz. Karakteri sahnede istediğiniz bir yere yerleştirip karakterin adını **Karakter** olarak değiştiriniz. **Inspector** panelinde bulunan **Add Component** butonuna basarak **Nav Mesh Agent** ve **Animator** bileşenlerini ekleyiniz.
**5. Adım: Assets** panelinde boş bir yere sağ tıklayarak **Create>Animator Controller** seçeneğini seçip adını **Animasyon** olarak ayarlayınız.
**6. Adım:** Animasyon kontrolcüsünü açarak indirdiğiniz Idle ve Walking animasyonlarını içine atınız. Parameters kısmından isMoving isimli Bool değişken ekleyiniz. Idle ve Walking arasında **Transition** oluşturarak Idle durumundan Walk durumuna geçerken isMoving değişkenini ''true'', Walk durumundan Idle durumuna geçerken de isMoving değişkenini ''false'' olacak şekilde ayarlayınız (Görsel 4.3).



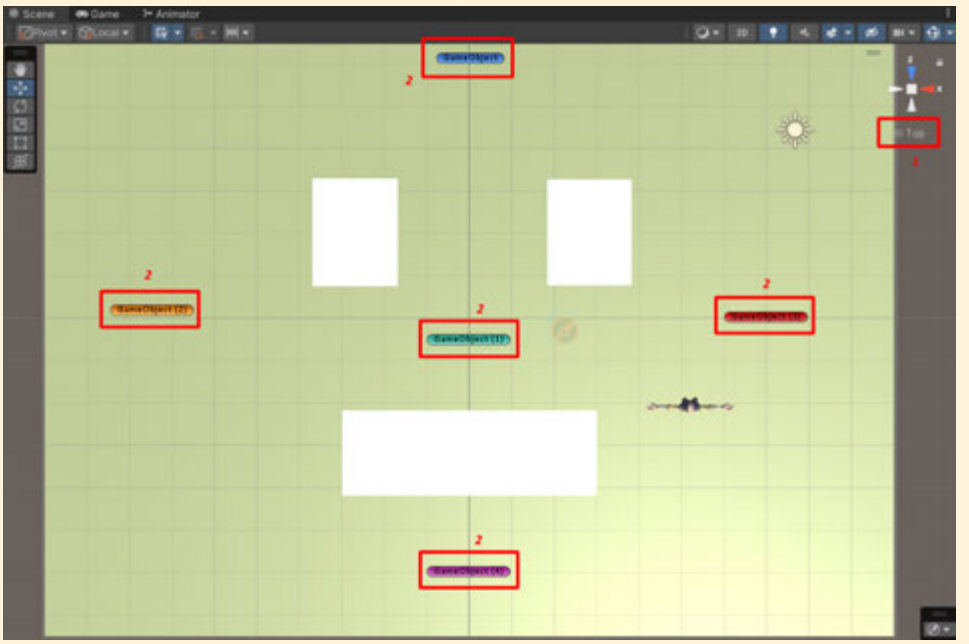
**Görsel 4.3:** Animator Controller nesnesini ayarlama

**7. Adım: Hierarchy** panelinden **Karakter** nesnesini seçip **Inspector** panelindeki **Animator** bileşeninin **Controller** özelliğine 5. adımda oluşturulan animasyon kontrolcüsünü ekleyiniz.
**8. Adım: Hierarchy** panelinden **Karakter** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak "devriye" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

```
using UnityEngine;
using UnityEngine.AI;
using System.Collections;
public class devriye : MonoBehaviour
{
    public Transform[] hedefler;
    private int hedefNokta = 0;
    private NavMeshAgent agent;
    private Animator anim;
    void Start()
    {
        agent = GetComponent<NavMeshAgent>();
        anim = GetComponent<Animator>();
        GotoNextPoint();
    }
    void GotoNextPoint()
    {
        // Seçili olan hedefe git
        agent.destination = hedefler[hedefNokta].position;
        // Dizideki sonraki noktayı hedef olarak seç,
        // gerekirse ilk elemana dön
        hedefNokta++;
        if(hedefNokta+1>hedefler.Length)
        {
            hedefNokta = 0;
        }
    }
    void Update()
    {
        if (agent.isStopped)
            anim.SetBool("isMoving", false);
        else
            anim.SetBool("isMoving", true);
        // Hedefe yaklaşınca sonraki noktaya git
        if (!agent.pathPending && agent.remainingDistance < 1f)
            GotoNextPoint();
    }
}
```

**9. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp, **Create Empty** seçeneğini seçerek 5 tane boş nesne ekleyiniz.

**10. Adım:** Kamerayı üstten görünecek şekilde ayarlayıp (top,ortographic), nesneleri gezilebilen alanın üzerinde gösterilen noktalara taşıyınız (Görsel 4.4).



**Görsel 4.4:** Devriye noktaları

**11. Adım: Hierarchy** panelinden **Karakter** nesnesini seçerek devriye scriptinin altında bulunan hedefler değişkeninin değerini 5 yapıp bir önceki adımda oluşturulan boş nesneleri sırayla buraya ekleyiniz.
**12. Adım: Game Object>AI** menüsünden sahneye **NavMesh Surface** nesnesi ekleyiniz ve **Inspector** panelindeki **Bake** tuşuna basınız.
**13. Adım:** Uygulamayı çalıştırdığınız zaman **Karakter** nesnesi belirtilen koordinatlara sırayla hareket edecektir.

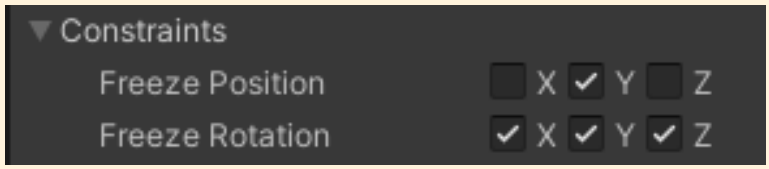
## 3. UYGULAMA

İşlem adımlarına göre NPC ekleyip bunun karakterinizi takip etmesini sağlayınız.

**1. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.
**2. Adım: Window>Package Manager** ile gelen pencerede **AI Navigation** seçip, **Install** düğmesine basarak projenize ekleyiniz.
**3. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak açılan **3D Object** menüsündeki **Plane** ve **Cube** nesneleri ile istediğiniz gibi bir sahne oluşturunuz.
**4. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Object** menüsünden 2 Adet **Capsule** nesnesi ekleyiniz. **Capsule** nesnelerinden birini seçerek **Inspector** panelinden NPC adını veriniz ve **Add Component** butonuna basarak **NavMesh Agent** bileşenini ekleyiniz. **NavMesh Agent** bileşeninin **Stopping Distance** özelliğini 5 yapınız. Bu özellik sayesinde NPC sizi belirli bir mesafeden takip edecektir.
**5. Adım:** NPC nesnesini seçip Inspector panelindeki **Add Component** butonuna basarak "takip" isimli bir script ekleyip aşağıdaki kodları yazınız.

```
using UnityEngine;
using UnityEngine.AI;
using System.Collections;
public class takip : MonoBehaviour
{
    public Transform hedef;
    private NavMeshAgent agent;
    void Start()
    {
        agent = GetComponent<NavMeshAgent>();
    }
    void Update()
    {
        agent.destination = hedef.position;
    }
}
```

**6. Adım: Hierarchy** panelinden diğer **Capsule** nesnesini seçip **Inspector** panelinden **Oyuncu** adını veriniz ve **Add Component** butonuna basarak **Rigidbody** bileşenini ekleyiniz. **Rigidbody** bileşeninin **Constraints** özelliğinin ayarlarını Görsel 4.5'teki gibi yapınız.



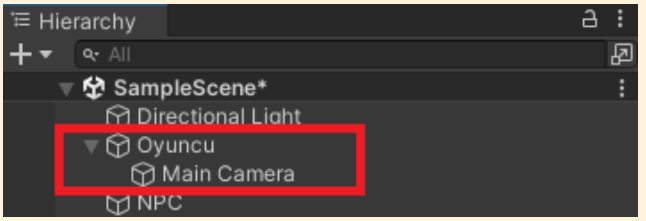

**Görsel 4.5:** Constraints özelliğini ayarlama

**7. Adım: Hierarchy** panelinden **Oyuncu** nesnesini seçip **Inspector** panelindeki **Add Component** butonuna basarak "hareket" isimli bir script ekleyip aşağıdaki kodları yazınız.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class hareket : MonoBehaviour
{
    float hiz = 10;
    float v,h;
    private Transform t;
    private void Start()
    {
        t = GetComponent<Transform>();
    }
    void Update()
    {
        v = Input.GetAxis("Vertical") * hiz*Time.deltaTime;
        h = Input.GetAxis("Horizontal") * hiz*Time.deltaTime;
        t.position += new Vector3(v, 0, h);
    }
}
```

**8. Adım: Game Object>AI** menüsünden sahneye **NavMesh Surface** nesnesi ekleyiniz ve **Inspector** panelindeki **Bake** tuşuna basınız.

**9. Adım: Main Camera** nesnesini **Oyuncu** karakterinin alt ögesi yapınız (Görsel 4.6).




**Görsel 4.6:** Main Camera nesnesini ayarlama

**10. Adım:** NPC nesnesinin "takip" scriptinde bulunan **Hedef** değişkenine **Hierarchy** panelinde bulunan **Oyuncu** nesnesini sürükleyip atayınız.
**11. Adım:** Uygulamayı çalıştırıp sahnede gezmeye başlayınca NPC sizi takip edecektir.

SIRA SİZDE

3. Uygulamaya birden fazla NPC ekleyip hepsinin karakterinizi takip etmesini sağlayınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Sahneye nesneler ekledi. | | |
| 2. Sahneye NPC nesneleri ekledi. | | |
| 3. Sahneye NavMesh Surface bileşenini ekledi. | | |
| 4. NavMesh Agent bileşenini NPC nesnelerine ekledi. | | |
| 5. Bake işlemini uyguladı. | | |
| 6. Gerekli kodları yazarak NPC nesnelerinin oyuncuyu takip etmesini sağladı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 4.1.2. Yerel Ağda Birlikte Oynanan Oyun Yapımı

**Yerel ağda birlikte oynanan oyunlar**, 2 veya daha fazla oyuncunun bir cihaza bağlanarak birlikte oyun oynaması şeklinde tanımlanabilir. Bunun için internet bağlantısı gerekmez. Cihazların aynı ağda olması yeterlidir. Oyunu başlatmak için cihazlardan biri oyunu kurmalı (host), diğer cihazlar da ona bağlanmalıdır (client). Host cihazlar hem sunucu hem istemci olarak çalışır.

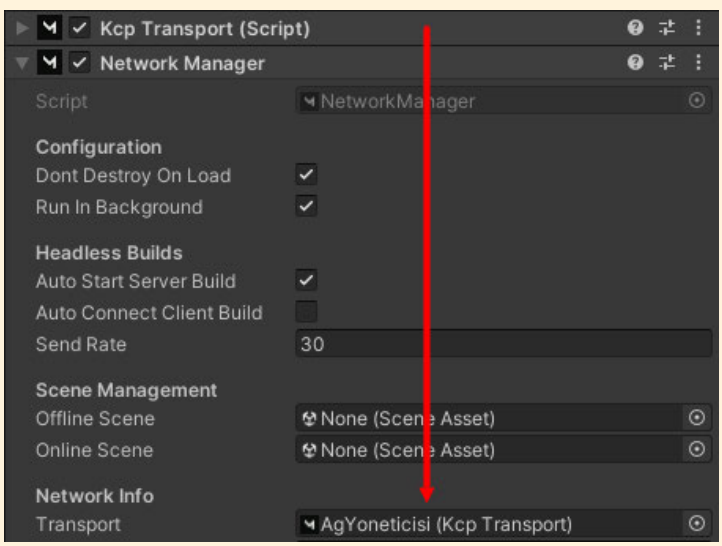
Sunucu; karakterlerin pozisyonlarını hesaplama, oyuncuların bağlantı durumları ile ilgilenme, matematiksel hesaplamalar yapma gibi pek çok işlemi gerçekleştirir ve sonuçları, kendine bağlı olan cihazlara gönderir. İstemci cihazlar ise sadece oyuncuların etkileşimlerini alıp bunları sunucuya göndermek ve oradan gelen sonuçlara göre yeni durumları ekranda göstermekten sorumludur. Bu yüzden çok oyunculu oyunların programlama mantığı tek oyunculu oyunlardan farklıdır.

Çok oyunculu oyunlar genelde iki kısımdan oluşur. Bunlar **lobi (lobby)** ve oyuncuların birbirleriyle etkileşime girdikleri **oyun** kısmıdır. **Lobi;** oyuncuların oyunla ilgili ayarları yaptığı, karakter seçebildiği, başka oyuncularla iletişim kurabildiği ve kurulu olan oyunları görüp bunlara katılma isteği gönderdikleri kısımdır. Oyuncular arasında eşleşme sağlanana kadar oyuncular lobide beklerler. Eşleşme sağlandıktan sonra da oyun başlar.

## 4. UYGULAMA

Uygulama adımlarını takip ederek yerel ağda 2 oyuncunun birbirine bağlanmasını sağlayınız.

**1. Adım:** Oyun motorunda **Lobi** adında yeni bir 3D Core uygulaması oluşturunuz.
**2. Adım:** Asset Store kullanarak, Mirror eklentisini indirip projeye ekleyiniz.
**3. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp, **Create Empty** seçeneğini seçerek sahneye boş bir nesne ekleyiniz ve **Inspector** panelinden adını **AgYoneticisi** olarak değiştiriniz.
**4. Adım:** AgYoneticisi nesnesini seçip, **Inspector** panelinden **Add Component** butonuna basarak **Network Manager** bileşenini ekleyiniz.
**5. Adım:** AgYoneticisi nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Manager HUD** bileşenini ekleyiniz.
**6. Adım:** AgYoneticisi nesnesini seçip **Inspector** panelindeki **Add Component** butonuna basarak **Kcp Transport** bileşenini ekleyiniz.
**7. Adım:** AgYoneticisi nesnesini seçip, **Inspector** panelinde bulunan **Kcp Transport** bileşenini **Network Manager** bileşeninin **Transport** özelliğine tutup sürükleyiniz (Görsel 4.7).




**Görsel 4.7:** Transport özelliğini ayarlama

**8. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **3D Object>Capsule** nesnesini seçiniz ve **Inspector** panelinden adını **Oyuncu** olarak değiştiriniz.
**9. Adım:** Oyuncu nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Identity** bileşenini ekleyiniz. Bu sayede oyuncumuz sunucu ile iletişim kurmak için bir ağ kimliği kazanmış olacaktır.
**10. Adım:** Oyuncu nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Transform (Unreliable)** bileşenini ekleyiniz ve bu bileşende bulunan **Client Authority** seçeneğini aktif ediniz. Bu sayede oyuncumuzun konumu önce sunucuya, sunucudan da diğer oyunculara iletilecektir.

**11. Adım:** Oyuncu nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak "hareket" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız. **MonoBehaviour** kısmını **NetworkBehaviour** olarak değiştirmeyi unutmayınız.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mirror;
public class hareket : NetworkBehaviour
{
    float hiz = 10;
    void Update()
    {
        if (isLocalPlayer)
        {
            if (Input.GetKey(KeyCode.LeftArrow))
            {
                transform.position += new Vector3(-hiz, 0, 0) *
Time.deltaTime;
            }
            else if (Input.GetKey(KeyCode.RightArrow))
            {
                transform.position += new Vector3(hiz, 0, 0) *
Time.deltaTime;
            }
        }
    }
}
```

NOT

isLocalPlayer komutu, hareket komutlarının sadece kendi cihazınızda çalışmasını sağlar. Bu satır yazılmazsa oyunu kuran oyuncu diğer oyuncuların hareketlerini kontrol edebilir ve bu istenmeyen bir durumdur.

**12. Adım:** Oyuncu nesnesini **Asset** penceresine tutup sürükleyerek Prefab oluşturunuz.
**13. Adım:** AgYoneticisi nesnesini seçip **Inspector** panelindeki **Network Manager** bileşeninin **Player Prefab** özelliğine, **Project** panelinde bulunan **Assets** klasöründen Oyuncu prefabını sürükleyerek atayınız.
**14. Adım: File>Save** seçeneğini seçip sahneyi Lobi ismiyle kaydediniz.
**15. Adım: File>Build Settings** menüsünü açınız. Açılan pencerede **Add Open Scenes** düğmesine basıp Lobi isimli sahneyi derlemeye ekleyiniz. Ardından **Build And Run** düğmesine basıp uygulamayı kaydetme konumunu seçiniz. Açılan uygulamanın lobi ekranında Host seçeneğini seçiniz.
**16. Adım:** Oyun motorunda uygulamayı çalıştırıp lobby penceresindeki Client düğmesine tıklayınız. Oluşturduğunuz Capsule prefabları sahneye gelecektir. Hierarchy panelinde de oyuncuların farklı ID numaraları ile oluştuğu görülecektir.
**17. Adım:** Oyuncuları sağ-sol yön tuşları ile hareket ettiriniz. Hareket ettirdiğiniz oyuncu, diğer oyuncunun ekranında da yer değiştirmiş olacaktır.

Windows Görev Yöneticisi ile uygulamanızı durdurabilirsiniz.

## 5. UYGULAMA

Uygulama adımlarını takip ederek yerel ağda 2 kişiyle oynanabilecek bir oyun tasarlayınız.

**1. Adım:** Oyun motorunda yeni boş bir 2D Core uygulaması oluşturunuz.
**2. Adım:** Asset Store kullanarak, Mirror eklentisini indirip projeye ekleyiniz.
**3. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp, **Create Empty** seçeneğini seçerek sahneye boş bir nesne ekleyiniz ve **Inspector** panelinden adını AgYoneticisi olarak değiştiriniz.
**4. Adım:** AgYoneticisi nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak AgYoneticisi isimli bir script ekleyip aşağıdaki kodları yazınız.

```
using UnityEngine;
using Mirror;
public class agyoneticisi : NetworkManager
{
    public Transform solRaketOlusmaNoktasi;
    public Transform sagRaketOlusmaNoktasi;
    GameObject top;

    public override void
OnServerAddPlayer(NetworkConnectionToClient baglanti)
    {
        //Raket oluşma noktası seçiliyor
        Transform baslangic = numPlayers == 0 ?
solRaketOlusmaNoktasi : sagRaketOlusmaNoktasi;
        //Seçilen noktada raket oluşturuluyor
        GameObject oyuncu = Instantiate(playerPrefab, baslangic.
position, baslangic.rotation);
        //Oyuncunun bağlantısı sağlanıyor
        NetworkServer.AddPlayerForConnection(baglanti, oyuncu);
        // iki kişi olduysa topu oluştur
        if (numPlayers == 2)
        {
            top = Instantiate(spawnPrefabs.Find(prefab => prefab.
name == "Top"));
            NetworkServer.Spawn(top);
        }
    }
    public override void
OnServerDisconnect(NetworkConnectionToClient baglanti)
    {
        // bağlantı koparsa topu yok et
        if (top != null)
            NetworkServer.Destroy(top);

        base.OnServerDisconnect(baglanti);
    }
}
```

**5. Adım:** AgYoneticisi nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Manager HUD** bileşenini ekleyiniz.

**6. Adım:** AgYoneticisi nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Kcp Transport** bileşenini ekleyiniz. Bu bileşeni tutup sürükleyerek AgYoneticisi scriptinin **Transport** özelliği olarak ayarlayınız.
**7. Adım: Hierarchy** panelinde bulunan **Main Camera** nesnesini seçip **Inspector** panelinde bulunan **Projection** özelliğini Orthographic, **Size** özelliğini 40, **Background** özelliğini siyah olarak ayarlayınız.
**8. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **Create Empty** seçeneği ile sahneye boş bir nesne ekleyiniz ve adını Çerçeve olarak değiştiriniz.
**9. Adım: Hierarchy** panelinde bulunan **Çerçeve** nesnesine sağ tıklayıp **2D Object>Sprites** menüsü altından sahneye **Square** nesnesi ekleyiniz ve adını **Üst Duvar** olarak değiştiriniz. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini -3, 15, 0; **Scale** değerlerini 50, 1, 1 olarak ayarlayınız. Yine bu nesnenin **Inspector** panelinde bulunan **Add Component** butonuna basarak **Box Collider 2D** bileşenini ekleyiniz.
**10. Adım: Hierarchy** panelinde bulunan **Üst Duvar** nesnesine sağ tıklayarak **Duplicate** seçeneğini seçiniz ve oluşan kopyanın adını **Alt Duvar** olarak değiştiriniz. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini -3, -15, 0; **Scale** değerlerini 50, 1, 1 olarak ayarlayınız.
**11. Adım: Hierarchy** panelinde bulunan **Üst Duvar** nesnesine sağ tıklayarak **Duplicate** seçeneğini seçiniz ve oluşan kopyanın adını **Sağ Duvar** olarak değiştiriniz. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini 22, 0, 0; **Scale** değerlerini 1, 30, 1 olarak ayarlayınız.
**12. Adım: Hierarchy** panelinde bulunan **Sağ Duvar** nesnesine sağ tıklayarak **Duplicate** seçeneğini seçiniz ve oluşan kopyanın adını **Sol Duvar** olarak değiştiriniz. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini -28, 0, 0; **Scale** değerlerini 1, 30, 1 olarak ayarlayınız.
**13. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **Create Empty** seçeneği ile sahneye boş bir nesne ekleyiniz ve adını **SolRaketOluşturmaNoktası** olarak değiştiriniz. Bu nesnenin **Position** değerlerini -22, 0, 0 olarak ayarlayınız.
**14. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **Create Empty** seçeneği ile sahneye boş bir nesne ekleyiniz ve adını **SağRaketOluşturmaNoktası** olarak değiştiriniz. Bu nesnenin **Position** değerlerini 16, 0, 0 olarak ayarlayınız.
**15. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **2D Object>Sprites** menüsü altından sahneye **Square** nesnesi ekleyiniz ve adını **Raket** olarak değiştiriniz. Bu nesnenin **Scale** değerlerini 1, 5, 1 olarak ayarlayınız.
**16. Adım: Hierarchy** panelinden **Raket** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Box Collider 2D** bileşenini ekleyiniz.
**17. Adım: Hierarchy** panelinden **Raket** nesnesini seçip **Inspector** panelindeki **Add Component** butonuna basarak **RigidBody 2D** bileşenini ekleyiniz. **RigidBody 2D** bileşeninin **Gravity** özelliğini 0, **Interpolate** özelliğini **Interpolate** olarak ayarlayınız. Ardından **Constrains Freeze Position X** ve **Constrains Freeze Rotation Z** seçeneklerini işaretleyiniz.
**18. Adım: Hierarchy** panelinde bulunan **Raket** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Identity** bileşenini ekleyiniz. Bu sayede oyuncumuz sunucu ile iletişim kurmak için bir ağ kimliği kazanmış olacaktır.
**19. Adım: Hierarchy** panelinde bulunan **Raket** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Transform (Unreliable)** bileşenini ekleyiniz ve bu bileşende bulunan **Client Authority** seçeneğini aktif ediniz. Bu sayede oyuncumuzun konumu önce sunucuya, sunucudan da diğer oyunculara iletilecektir.
**20. Adım: Hierarchy** panelinde bulunan **Raket** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak "oyuncu" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

```
using UnityEngine;
using Mirror;
public class oyuncu : NetworkBehaviour
{
    float hiz = 1000;
    float dikeyHareket;
    Rigidbody2D rigidbody2d;
    private void Start()
    {
        rigidbody2d=GetComponent<Rigidbody2D>();
    }
    void FixedUpdate()
    {
        dikeyHareket = Input.GetAxisRaw("Vertical");
        if (isLocalPlayer)
            rigidbody2d.velocity = new Vector2(0, dikeyHareket *
hiz * Time.fixedDeltaTime);
    }
}
```

**21. Adım: Raket** nesnesini **Asset** penceresine tutup sürükleyerek **Prefab** oluşturunuz. Ardından **Hierarchy** panelinden **Raket** nesnesini siliniz.

**22. Adım: Assets** panelinde boş bir yere sağ tıklayarak **Create>2D** menüsünden **Physic Material 2D** seçeneğini seçiniz ve **Topmalzemesi** adını veriniz. Bu nesnenin **Friction** değerini 0, **Bounciness** değerini 1 olarak ayarlayınız.

**23. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **2D Object>Sprites** menüsü altından sahneye **Circle** nesnesi ekleyiniz ve adını **Top** olarak değiştiriniz.

**24. Adım: Hierarchy** panelinden **Top** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Circle Collider 2D** bileşenini ekleyiniz. 22. adımda oluşturulan **Topmalzemesi** nesnesini **Circle Collider 2D** bileşeninin **Material** özelliğine sürükleyip bırakınız.

**25. Adım: Hierarchy** panelinden **Top** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **RigidBody 2D** bileşenini ekleyiniz. Bu bileşenin **Gravity** özelliğini **0**, **Mass** özelliğini **0.0001, Collision Detection** özelliğini **Continuous** olarak ayarlayınız. 22. adımda oluşturulan **Topmalzemesi** nesnesini bu bileşenin **Material** özelliğine sürükleyip bırakınız.

**26. Adım: Hierarchy** panelinde bulunan **Top** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Identity** bileşenini ekleyiniz. Bu sayede topumuz sunucu ile iletişim kurmak için bir ağ kimliği kazanmış olacaktır.

**27. Adım: Hierarchy** panelinde bulunan **Top** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak **Network Transform (Unreliable)** bileşenini ekleyiniz ve bu bileşende bulunan **Sync Direction** seçeneğini **Server to Client olarak** ayarlayınız. Bu sayede topun konumu sunucuda hesaplanıp diğer oyunculara iletilecektir.

**28. Adım: Hierarchy** panelinde bulunan **Top** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak "top" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

```
using UnityEngine;
using Mirror;
public class top : NetworkBehaviour
{
    float hiz = 30;
    public Rigidbody2D rigidbody2d;
    public override void OnStartServer()
```

```
    {
        base.OnStartServer();
        rigidbody2d.simulated = true;
        rigidbody2d.velocity = Vector2.right * hiz;
    }
    float VurusYonuHesapla(Vector2 topPozisyonu, Vector2
raketPozisyonu, float raketYuksekligi)
    {
        return (topPozisyonu.y - raketPozisyonu.y) /
raketYuksekligi;
    }

    // sadece serverda çalışacak kodlar
    [ServerCallback]
    void OnCollisionEnter2D(Collision2D carpisma)
    {
        // top rakete çarptıysa
        if (carpisma.transform.GetComponent<oyuncu>())
        {
            // rakete çarptıktan sonra topun y yönünü hesaplama
            float y = VurusYonuHesapla(transform.position,
carpisma.transform.position, carpisma.collider.bounds.size.y);
            // rakete çarptıktan sonra topun x yönünü hesaplama
            float x = carpisma.relativeVelocity.x > 0 ? 1 : -1;
            // hesaplanan x ve y değerlerine göre topu //////
            //yönlendirme
            Vector2 yon = new Vector2(x, y).normalized;
            rigidbody2d.velocity = yon * hiz;
        }
    }
}
```

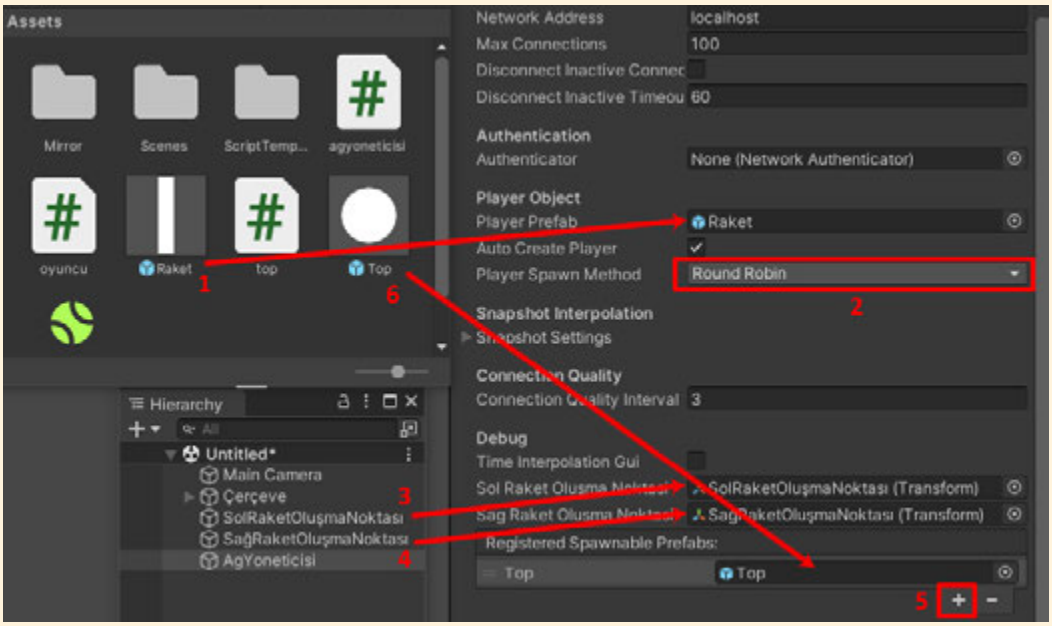
**29. Adım: Hierarchy** panelinde bulunan **Top** nesnesinin **Inspector** panelinde bulunan **Top** scriptindeki **Rigidbody2d** değişkenini Görsel 4.8'deki gibi ayarlayınız.




**Görsel 4.8:** Rigidbody 2D değişkenini ayarlama

**30. Adım: Hierarchy** panelinde bulunan **Top** nesnesini **Asset** penceresine tutup sürükleyerek **Prefab** oluşturunuz. Ardından **Hierarchy** panelindeki **Top** nesnesini siliniz.

**31. Adım:** AgYoneticisi nesnesini seçip **Inspector** panelinden **AgYoneticisi (Script)** bileşeninin altında bulunan değişkenleri Görsel 4.9'daki gibi ayarlayınız.




**Görsel 4.9:** Değişkenleri ayarlama

**32. Adım: File>Save** seçeneğini seçip sahneyi **Pong** ismiyle kaydediniz.
**33. Adım: File>Build Settings** menüsünü açınız. Açılan pencerede **Add Open Scenes** düğmesine basıp **Pong** isimli sahneyi derlemeye ekleyiniz. Ardından **Build And Run** düğmesine basıp uygulamayı kaydetme konumunu seçiniz. Açılan uygulamanın lobi ekranında **Host** seçeneğini seçiniz.
**34. Adım:** Oyun motorunda uygulamayı çalıştırınca **lobby** penceresindeki **Client** düğmesine tıklayınız. Oluşturulan raket ve top nesneleri sahneye gelecektir. Yukarı-aşağı yön tuşları ile raketleri hareket ettirebilirsiniz.

### 4.1.3. İnternet Üzerinden Oynanan (Online) Oyun Yapımı

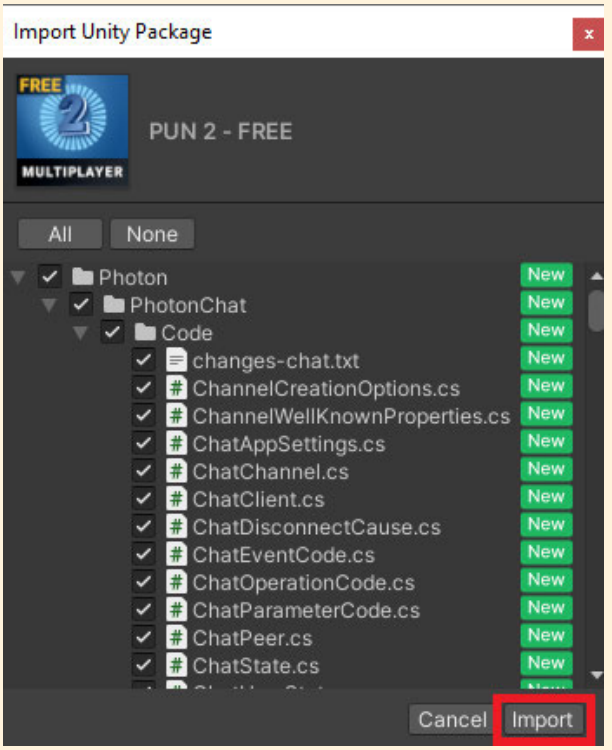
**İnternet üzerinden oynanan oyunlar,** dünyanın farklı bölgelerinden yüzlerce hatta binlerce oyuncunun bir sunucuya (server) bağlanarak oynadığı oyunlardır. Tüm cihazlar (client) oyunun başında bu sunucuya bağlanır ve oyun süresince de sunucuya bağlı kalmak zorundadır. Bu yüzden bu tarz oyunlardaki sunucuların çok güçlü cihazlar olması gerekir. Sunucu işlemleri haricinde bu oyunların yapım aşamaları yerel ağda birlikte oynanan oyunlardan çok da farklı değildir.

Oyun motorunda online oyun yapmak için çeşitli eklentiler bulunmaktadır. Bu eklentileri hazırlayan firmaların, oyun geliştirme aşamasında deneme yapmak için hazır sunucu hizmetleri de bulunmaktadır. Oyun, etkileşim aldıkça ücretli olan daha üst paketlere geçmek gerekir. Bunun yanında oyun motorunda kendi sunucunuzu oluşturmak için araçlar da bulunmaktadır.

## 6. UYGULAMA

Uygulama adımlarını takip ederek bir arenada savaşan robotlarla ilgili internet üzerinden oynanabilecek bir oyun tasarlayınız.

**1. Adım:** Oyun motorunda yeni boş bir 3D uygulaması oluşturunuz.
**2. Adım: Asset Store** sayfasından **Photon Pun 2 Free** paketini indirip projenize ekleyiniz (Görsel 4.10).




**Görsel 4.10:** Eklentinin projeye eklenmesi

**3. Adım:** Eklentinin sitesine gidip üye olunuz. Ardından **Dashboard** sayfasına giriniz ve **Create A New App** butonuna basınız.
**4. Adım: Select Photon SDK** kısmını **Realtime** olarak seçip, **Application Name** kısmına da **Online Oyun** yazarak **Create** butonuna basınız. Oluşturulan **App ID** değerini kopyalayınız.
**5. Adım:** Oyun motorunda **Window>Photon Networking** menüsünden **PUN Wizard** seçeneğini seçiniz. Açılan pencerede **Setup Project** seçeneğini seçerek, bir önceki adımda kopyalanan App ID değerini yapıştırıp **Setup Project** butonuna tıklayınız.
**6. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp, **Create Empty** seçeneğini seçerek yeni boş bir nesne oluşturunuz ve adını **Lobi** olarak değiştiriniz. Ardından **Inspector** panelinde bulunan **Add Component** butonuna basarak "Lobi" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

```
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;
public class Lobi : MonoBehaviourPunCallbacks
{
    private byte maksOyuncuSayisi = 4;
    string gameVersion = "1";
    public GameObject girisPaneli;
```

```
    public GameObject yukleniyorMetni;
    bool isConnecting;
    void Awake()
    {
        PhotonNetwork.AutomaticallySyncScene = true;
    }
    void Start()
    {
        yukleniyorMetni.SetActive(false);
        girisPaneli.SetActive(true);
    }
    public void Baglan()
    {
        yukleniyorMetni.SetActive(true);
        girisPaneli.SetActive(false);
        if (PhotonNetwork.IsConnected)
        {
            PhotonNetwork.JoinRandomRoom();
        }
        else

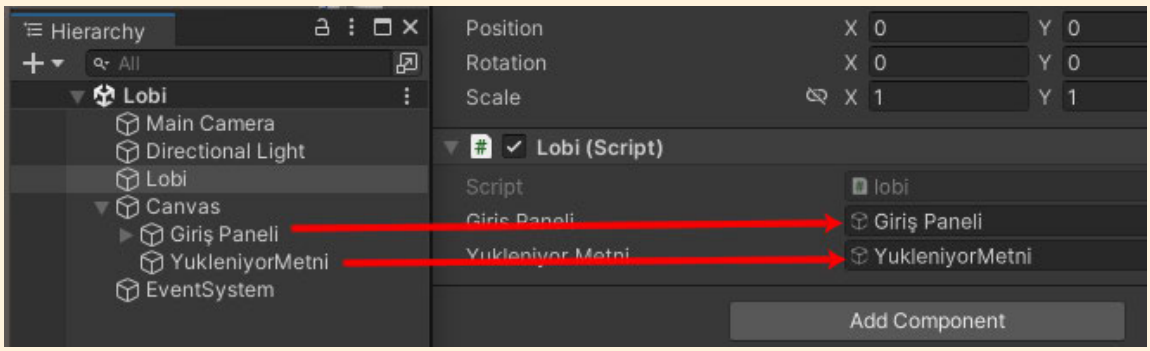

        {
            isConnecting = PhotonNetwork.
ConnectUsingSettings();
            PhotonNetwork.GameVersion = gameVersion;
        }
    }
    public override void OnConnectedToMaster()
    {
        if (isConnecting)
        {
            PhotonNetwork.JoinRandomRoom();
            isConnecting = false;
        }
    }
    public override void OnDisconnected(DisconnectCause
cause)
    {
        isConnecting = false;
        yukleniyorMetni.SetActive(false);
        girisPaneli.SetActive(true);
    }
    public override void OnJoinRandomFailed(short returnCode,
string message)
    {
        //girilebilecek oyun yoksa yeni oyun oluştur
        PhotonNetwork.CreateRoom(null, new RoomOptions {
MaxPlayers = maksOyuncuSayisi });
    }
    public override void OnJoinedRoom()
    {
        if (PhotonNetwork.CurrentRoom.PlayerCount == 1)
            PhotonNetwork.LoadLevel("Oda 1");
    }
}
```

**7. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **UI>Panel** nesnesini seçiniz ve adını **Giriş Paneli** olarak değiştiriniz.
**8. Adım: Hierarchy** panelinde bulunan **Giriş Paneli** nesnesine sağ tıklayarak **UI>Button** nesnesini seçiniz ve adını **Oyuna Katıl** olarak değiştiriniz. Bu nesnenin **Inscpector** panelinde bulunan **PosY** değerini **-35** olarak ayarlayınız. **Oyuna Katıl** butonunun altında bulunan **Text** nesnesini seçip, **Inspector** panelinde bulunan **Text** özelliğini **Oyuna Katıl** olarak değiştiriniz.
**9. Adım: Hierarchy** panelinden **Oyuna Katıl** nesnesini seçip **Inspector** panelinde bulunan **On Click** özelliğinin altındaki "+" simgesine basınız. Nesne olarak **Lobi** nesnesini, fonksiyon olarak da **Lobi scriptinde** bulunan **Baglan** fonksiyonunu seçiniz.
**10. Adım: Hierarchy** panelinde bulunan **Giriş Paneli** nesnesine sağ tıklayıp **UI>Input Field** nesnesini seçiniz ve adını **İsim Alanı** olarak değiştiriniz. **Inspector** panelinde bulunan **Add Component** butonuna basarak "isimkaydetme" adlı bir script ekleyiniz ve aşağıdaki kodları yazınız.

```
using UnityEngine;
using UnityEngine.UI;
using Photon.Pun;
public class isimkaydetme : MonoBehaviour
{
    string oyuncuismi = "oyuncuismi";
    void Start()
    {
        string varsayilanisim = "";
        InputField isimalani = GetComponent<InputField>();
        if (isimalani != null)
        {
            if (PlayerPrefs.HasKey(oyuncuismi))
            {
                varsayilanisim = PlayerPrefs.
GetString(oyuncuismi);
                isimalani.text = varsayilanisim;
            }
        }
        PhotonNetwork.NickName = varsayilanisim;
    }
    public void Oyuncuisminiayarla(string yazilan)
    {
        if (string.IsNullOrEmpty(yazilan))
        {
            Debug.LogError("Oyuncu ismi boş olamaz");
            return;
        }
        PhotonNetwork.NickName = yazilan;
        PlayerPrefs.SetString(oyuncuismi, yazilan);
    }
}
```

**11. Adım: Hierarchy** panelinden **İsim Alanı** nesnesini seçip **Inspector** panelinde bulunan **On Value Changed** özelliğinin altındaki "+" simgesine basınız. Nesne olarak **İsim Alanı** nesnesini, fonksiyon olarak da isimkaydetme scriptinin altındaki **Dynamic String** alanından **Oyuncuisminiayarla** fonksiyonunu seçiniz.
**12. Adım: Hierarchy** panelinde bulunan **Giriş Paneli** nesnesine sağ tıklayarak **UI>Text** nesnesini seçiniz ve adını **Yükleniyor Metni** olarak değiştiriniz. Bu nesnenin **Inspector** panelinde bulunan **PosY** değerini 35, **Text** özelliğini ise "Yükleniyor..." olarak ayarlayınız.

**13. Adım: Hierarchy** panelinden Lobi nesnesini seçip **Inspector** panelinde bulunan **Lobi** scriptindeki değişkenleri Görsel 4.11'deki gibi ayarlayınız.



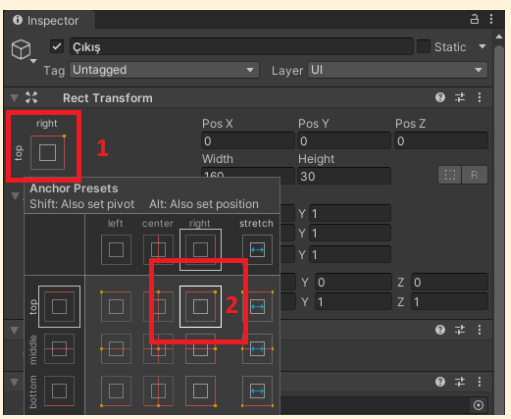

**Görsel 4.11:** Lobi scriptindeki değişkenleri ayarlama

**14. Adım: File>Save As** seçeneğini seçerek sahneyi **Lobi** ismiyle kaydediniz.
**15. Adım: File>New Scene** seçeneğini seçerek yeni boş bir sahne oluşturunuz.
**16. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Object>Cube** nesnesini seçiniz ve adını **Zemin** olarak ayarlayınız. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini **0, 0, 0; Scale** değerlerini ise **20, 1, 20** olarak ayarlayınız.
**17. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Object>Cube** nesnesini seçiniz ve adını **Duvar 1** olarak ayarlayınız. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini **0, 2, -10; Scale** değerlerini ise **20, 3, 1** olarak ayarlayınız.
**18. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Object>Cube** nesnesini seçiniz ve adını **Duvar 2** olarak ayarlayınız. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini **0, 2,10; Scale** değerlerini ise **20, 3, 1** olarak ayarlayınız.
**19. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Object>Cube** nesnesini seçiniz ve adını **Duvar 3** olarak ayarlayınız. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini **-10, 2, 0; Scale** değerlerini **20, 3, 1; Rotation** değerlerini **0, 90, 0** olarak ayarlayınız.
**20. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Object>Cube** nesnesini seçiniz ve adını **Duvar 4** olarak ayarlayınız. Bu nesnenin **Inspector** panelinde bulunan **Position** değerlerini **10, 2, 0; Scale** değerlerini **20, 3, 1; Rotation** değerlerini ise **0, -90, 0** olarak ayarlayınız.
**21. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak, **Create Empty** seçeneğini seçip boş bir nesne oluşturunuz ve adını **GameManager** olarak ayarlayınız. Bu nesnenin **Inspector** panelinde bulunan **Add Component** butonuna basarak "GameManager" isimli bir script ekleyip aşağıdaki kodları yazınız.

```
using UnityEngine;
using UnityEngine.SceneManagement;
using Photon.Pun;
using Photon.Realtime;
public class GameManager : MonoBehaviourPunCallbacks
{
    public static GameManager Instance;
    public GameObject oyuncuPrefab;
    void Start()
    {
        Instance = this;
    }
    public override void OnLeftRoom()
    {
```

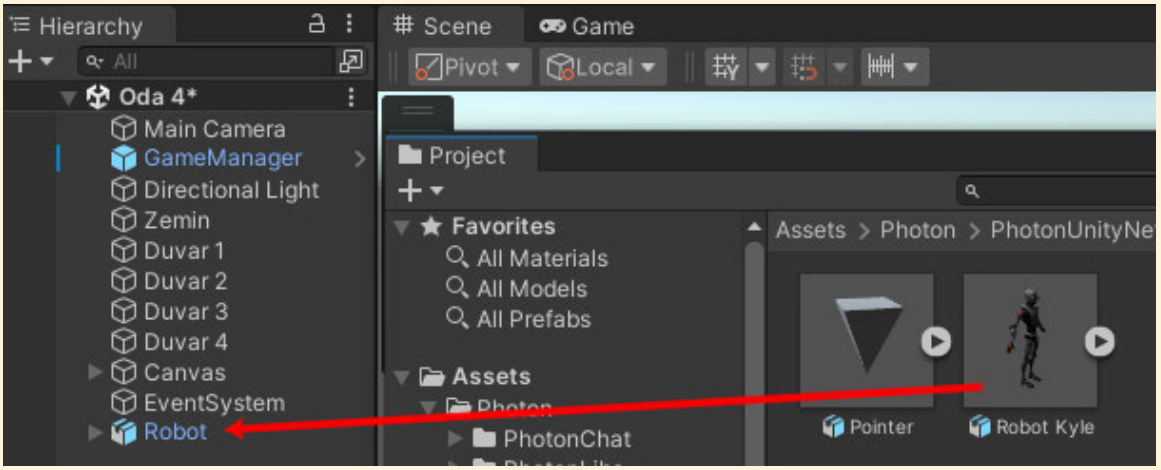
```
    SceneManager.LoadScene(0); // Oyuncu ayrıldığında yükleme
                               //ekranına dön
     }
     public void LeaveRoom()
     {
         PhotonNetwork.LeaveRoom();
     }
     void LoadArena()
     {
         if (!PhotonNetwork.IsMasterClient)
             return;
         PhotonNetwork.LoadLevel("Oda " + PhotonNetwork.
CurrentRoom.PlayerCount);
     }
     public override void OnPlayerEnteredRoom(Player other)
     {
         if (PhotonNetwork.IsMasterClient)
             LoadArena();
     }
     public override void OnPlayerLeftRoom(Player other)
     {
         if (PhotonNetwork.IsMasterClient)
             LoadArena();
     }
}
```

**22. Adım: Project** panelinde bulunan **Assets** klasörüne sağ tıklayıp **Create>Folder** seçeneğini seçiniz ve yeni boş bir klasör oluşturunuz. Bu klasörün adını **Prefab** olarak ayarlayınız. **Hierarchy** panelinde bulunan **GameManager** nesnesini tutup **Assets>Prefab** klasörüne sürükleyerek **Prefab** oluşturunuz.
**23. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **UI>Panel** nesnesini seçiniz. Bu nesnenin **Inspector** panelinde bulunan **Canvas Renderer** ve **Image** bileşenlerini siliniz.
**24. Adım: Hierarchy** panelinde bulunan **Panel** nesnesine sağ tıklayarak **UI>Button** nesnesini seçiniz ve adını **Çıkış** olarak ayarlayınız. Bu butonun altında bulunan **Text** nesnesini seçip, **Inspector** panelinde bulunan **Text** özelliğini **Çıkış** olarak değiştiriniz.
**25. Adım: Hierarchy** panelinde bulunan **Çıkış** butonunu seçip **Inspector** panelinde bulunan **Rect Transform** bileşeninin hizalama kısmını açınız ve Shift+Alt tuşlarına basılı tutarken **Top-Right** seçeneğini seçiniz. Bu sayede çıkış butonu sağ üst köşede görünecektir (Görsel 4.12).




**Görsel 4.12:** Çıkış butonunun Rect Transform özelliğini ayarlama

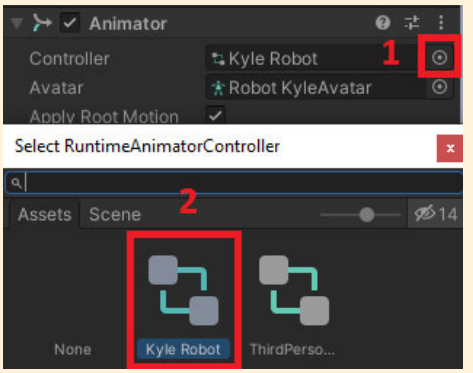
**26. Adım: Hierarchy** panelinde bulunan **Çıkış** butonunu seçip **Inspector** panelinde bulunan **On Click** özelliğinin altındaki "+" simgesine basınız. Nesne olarak **GameManager** nesnesini, fonksiyon olarak da **GameManager** scriptinde bulunan **LeaveRoom** fonksiyonunu seçiniz.
**27. Adım: File>Save** seçeneğini seçerek sahneyi **Oda 1** ismiyle kaydediniz.
**28. Adım: Assets** panelinde bulunan **Oda 1** sahnesini seçip Ctrl+D tuşları ile 3 sefer kopyasını oluşturunuz. Yeni oluşan sahnelerin isimlerini **Oda 2, Oda 3** ve **Oda 4** olarak ayarlayınız.
**29. Adım: Oda 2** sahnesini açıp **Hierarchy** panelinde bulunan **Zemin** nesnesinin **Scale** değerlerini **36, 1, 36; Duvar 1** nesnesinin **Position** değerlerini **0, 2, -18; Scale** değerlerini **36, 3, 1**; Duvar 2 nesnesinin **Position** değerlerini **0, 2, 18**; **Scale** değerlerini **36, 3, 1;** Duvar 3 nesnesinin **Position** değerlerini **-18, 2, 0; Scale** değerlerini **36, 3, 1; Duvar 4** nesnesinin **Position** değerlerini **18, 2, 0; Scale** değerlerini **36, 3, 1** olarak ayarlayınız. Sahneyi Ctrl+S tuşları ile kaydediniz.
**30. Adım: Oda 3** sahnesini açıp **Hierarchy** panelinde bulunan **Zemin** nesnesinin **Scale** değerlerini **50, 1, 50; Duvar 1** nesnesinin **Position** değerlerini **0, 2, -25; Scale** değerlerini **50, 3, 1; Duvar 2** nesnesinin **Position** değerlerini **0, 2, 25; Scale** değerlerini **50, 3, 1; Duvar 3** nesnesinin **Position** değerlerini **-25, 2, 0; Scale** değerlerini **50, 3, 1; Duvar 4** nesnesinin **Position** değerlerini **25, 2, 0; Scale** değerlerini **50, 3, 1** olarak ayarlayınız. Sahneyi Ctrl+S tuşları ile kaydediniz.
**31. Adım: Oda 4** sahnesini açıp **Hierarchy** panelinde bulunan **Zemin** nesnesinin **Scale** değerlerini **60, 1, 60; Duvar 1** nesnesinin **Position** değerlerini **0, 2, -30; Scale değerlerini 60, 3, 1; Duvar 2** nesnesinin **Position** değerlerini **0, 2, 30; Scale** değerlerini **60, 3, 1; Duvar 3** nesnesinin **Position** değerlerini **-30, 2, 0; Scale** değerlerini **60, 3, 1; Duvar 4** nesnesinin **Position** değerlerini **30, 2, 0; Scale** değerlerini **60, 3, 1** olarak ayarlayınız. Sahneyi Ctrl+S tuşları ile kaydediniz. Bu ayarlar sayesinde oyuncu sayısı arttıkça oyuncular daha büyük bir alanda mücadele edebilecekler.
**32. Adım: Project** panelinden "Assets>Photon>Photon Networking>Demos>Shared Assets>Models" klasörüne girip **Robot Kyle** isimli modeli sahneye ekleyiniz ve adını **Robot** olarak değiştiriniz (Görsel 4.13).




**Görsel 4.13:** Robot Kyle modelini sahneye ekleme

**33. Adım: Hierarchy** panelinden **Robot** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak **Character Controller** bileşenini ekleyiniz. Bu bileşenin **Center** değerlerini **0, 0.9, 0; Radius** değerini **0.3, Height** değerini de **1.8** olarak ayarlayınız.

**34. Adım: Hierarchy** panelinden **Robot** nesnesini seçip **Inspector** panelinde bulunan **Animator** bileşeninin **Controller** özelliğini **Kyle Robot** olarak ayarlayınız (Görsel 4.14).




**Görsel 4.14:** Animator Controller özelliğinin ayarlanması

**35. Adım: Hierarchy** panelinden **Robot** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak "oyuncuanimasyonyoneticisi" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

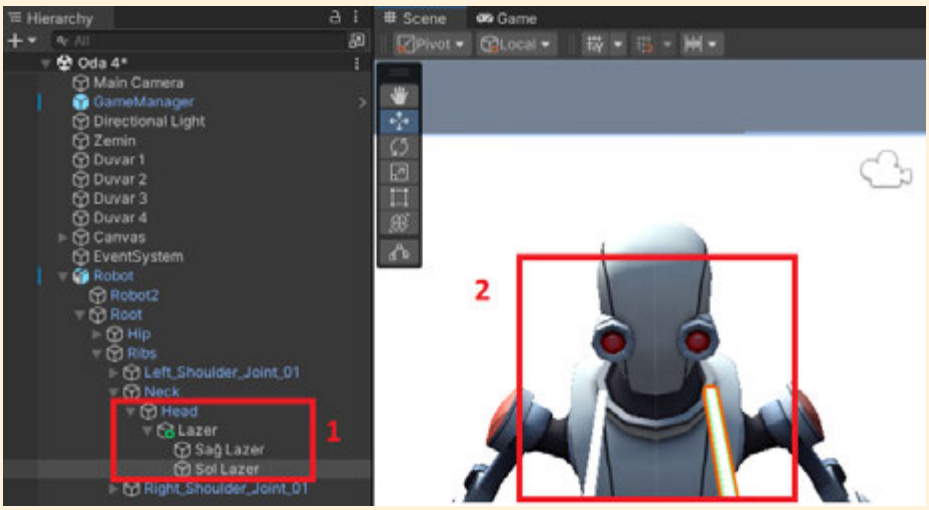
```
using UnityEngine;
using Photon.Pun;
public class oyuncuanimasyonyoneticisi : MonoBehaviourPun
{
    private float directionDampTime = 2f;
    private Animator animator;
    void Start()
    {
        animator = GetComponent<Animator>();
        if (!animator)
            Debug.LogError("Bulunamadı", this);
    }
    void Update()
    {
        if (photonView.IsMine == false && PhotonNetwork.
IsConnected == true)
            return;
        if (!animator)
            return;
        AnimatorStateInfo stateInfo = animator.
GetCurrentAnimatorStateInfo(0);
        if (stateInfo.IsName("Base Layer.Run"))
        {
            if (Input.GetButtonDown("Fire2"))
                animator.SetTrigger("Jump");
        }
        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");
        if (v < 0)
            v = 0;
        animator.SetFloat("Speed", h * h + v * v);
        animator.SetFloat("Direction", h, directionDampTime,
Time.deltaTime);
    }
}
```

**36. Adım: Hierarchy** panelinden Robot nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak **Camera Work** bileşenini ekleyiniz. Bu bileşenin **Center Offset** değerlerini **0, 4, 0** olarak ayarlayınız. Yine bu bileşenin **Follow on Start** özelliğindeki işareti kaldırınız.
**37. Adım: Hierarchy** panelinden **Robot** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak **Photon View** bileşenini ekleyiniz. Bu bileşenin **Synchronization** değerini **Unreliable on Change**, **Observable Search** değerini **Auto Find All** olarak ayarlayınız. Bu bileşen sayesinde oyuncumuz ağ kimliği kazanacaktır.
**38. Adım: Hierarchy** panelinden **Robot** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak **Photon Transform View** bileşenini ekleyiniz. Bu bileşen sayesinde oyuncunun hareketleri diğer oyunculara iletilecektir.
**39. Adım: Hierarchy** panelinden **Robot** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak **Photon Animator View** bileşenini ekleyiniz. Bu bileşen sayesinde oyuncunun animasyonları diğer oyunculara da iletilecektir. Bu bileşenin **Speed, Direction** ve **Jump** değerlerini **Discrete** olarak ayarlayınız. Discrete özelliği animasyonu saniyede 10 kez diğer oyunculara iletir.
**40. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Object>Cube** nesnesini seçiniz ve adını **Sağ Lazer** olarak değiştiriniz. Bu nesnenin **Inspector** panelinde bulunan **Scale** değerlerini **0.01, 0.01, 10; Transform** değerlerini de lazer robotun sağ gözünden çıkıyormuş gibi ayarlayınız. Lazerin konumunu ayarlarken robotun **Collider** bileşeni alanının dışında kalmasına dikkat ediniz. Yoksa lazer kullanıldığı zaman robot kendine zarar verir (Görsel 4.15).



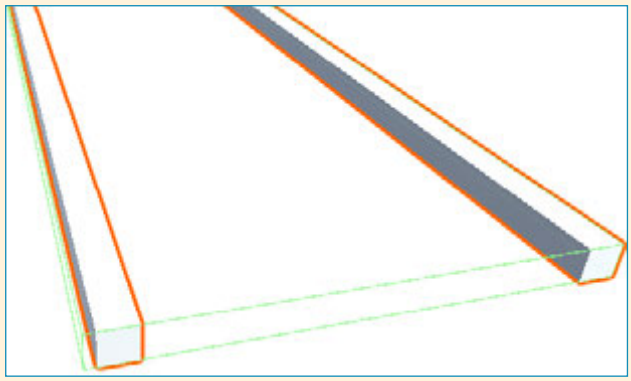
**Görsel 4.15:** Lazerin konumunun ayarlanması

**41. Adım: Hierarchy** panelinde bulunan **Robot** nesnesinin altındaki **Head** nesnesine sağ tıklayıp **Create Empty** seçeneği ile yeni boş bir nesne oluşturunuz ve adını **Lazer** olarak değiştiriniz. **Sağ Lazer** nesnesini **Lazer** nesnesinin alt elemanı olarak ayarlayınız. Ardından **Sağ Lazer** nesnesine sağ tıklayıp, **Duplicate** seçeneği ile çoğaltıp **Sol Lazer** adını veriniz. **Sol Lazer** nesnesinin **Transform** değerlerini lazer robotun sol gözünden çıkıyormuş gibi ayarlayınız (Görsel 4.16).




**Görsel 4.16:** Lazer nesnesini hiyerarşik olarak ayarlama

**42. Adım: Sol Lazer** nesnesinin **Inspector** panelinde bulunan **Box Collider** bileşenini siliniz.
**43. Adım: Sağ Lazer** nesnesinin **Box Collider** bileşeninin alanını **Center** ve **Size** değerleri ile oynayarak iki lazeri de kapsayacak şekilde genişletiniz (Görsel 4.17).



**Görsel 4.17:** Box Collider bileşenini ayarlama

**44. Adım: Sağ Lazer** nesnesinin **Box Collider** bileşeninin **IsTrigger** seçeneğini işaretleyiniz.
**45. Adım: Project** panelinde bulunan **Assets** klasörüne sağ tıklayıp, **Create>Material** seçeneğini seçip **Kırmızı Lazer** adını veriniz ve rengini kırmızı olarak ayarlayınız. Daha sonra bu materyali **Sağ Lazer** ve **Sol Lazer** nesnelerine materyal olarak atayınız.
**46. Adım: Hierarchy** panelinden **Robot** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak "oyuncuyoneticisi" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

```
using UnityEngine;
using Photon.Pun;
using Photon.Pun.Demo.PunBasics;
public class oyuncuyoneticisi : MonoBehaviourPunCallbacks,
IPunObservable
{
    public GameObject oyuncuarayuzuprefab;
    public static GameObject LocalPlayerInstance;
    public float can = 1f;
    public GameObject lazer;
    bool atesediyormu;
    public void OnPhotonSerializeView(PhotonStream stream,
PhotonMessageInfo info)
    {
        if (stream.IsWriting)
        {
            // diğer oyunculara bilgi gönderiliyor
            stream.SendNext(atesediyormu);
            stream.SendNext(can);
        }
        else
        {
            // diğer oyunculardan bilgi alınıyor
            this.atesediyormu = (bool)stream.ReceiveNext();
            this.can = (float)stream.ReceiveNext();
        }
    }
```

```
    void Awake()
    {
        if (photonView.IsMine)
            PlayerManager.LocalPlayerInstance = this.gameObject;
            lazer.SetActive(false);
            //DontDestroyOnLoad(this.gameObject);
    }
    void Start()
    {
        GameObject _uiGo = Instantiate(oyuncuarayuzuprefab);
        _uiGo.SendMessage("SetTarget", this, SendMessageOptions.
RequireReceiver);
        CameraWork _cameraWork = this.gameObject.
GetComponent<CameraWork>();
        if (photonView.IsMine)
            _cameraWork.OnStartFollowing();
        UnityEngine.SceneManagement.SceneManager.sceneLoaded +=
OnSceneLoaded;
    }
    void Update()
    {
        if (photonView.IsMine)
        {
            ProcessInputs();
            if (can <= 0f)
                GameManager.Instance.LeaveRoom();
        }
        if (lazer != null && atesediyormu != lazer.
activeInHierarchy)
            lazer.SetActive(atesediyormu);
    }
    void OnTriggerEnter(Collider other)
    {
        if (!photonView.IsMine)
            return;
        if (!other.name.Contains("Lazer"))
            return;
        can -= 0.1f;
    }
    void OnTriggerStay(Collider other)
    {
        if (!photonView.IsMine)
            return;
        if (!other.name.Contains("Lazer"))
            return;
        can -= 0.1f * Time.deltaTime;
    }
    void ProcessInputs()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            if (!atesediyormu)
                    atesediyormu = true;
        }
        if (Input.GetButtonUp("Fire1"))
```

```
        {
            if (atesediyormu)
                atesediyormu = false;
        }
    }
    void OnSceneLoaded(UnityEngine.SceneManagement.Scene scene,
UnityEngine.SceneManagement.LoadSceneMode loadingMode)
    {
        this.CalledOnLevelWasLoaded(scene.buildIndex);
    }
    void CalledOnLevelWasLoaded(int level)
    {
        if (!Physics.Raycast(transform.position, -Vector3.up, 5f))
        {
            transform.position = new Vector3(0f, 5f, 0f);
        }
        GameObject _uiGo = Instantiate(this.oyuncuarayuzuprefab);
        _uiGo.SendMessage("SetTarget", this, SendMessageOptions.
RequireReceiver);
    }
    public override void OnDisable()
    {
        base.OnDisable();
        UnityEngine.SceneManagement.SceneManager.sceneLoaded -=
OnSceneLoaded;
    }
}
```

**47. Adım: GameManager** scriptini açıp **Start ()** metodunu aşağıdaki şekilde güncelleyiniz.

```
void Start()
{

    Instance = this;
    if (oyuncuyoneticisi.LocalPlayerInstance == null)
      PhotonNetwork.Instantiate(this.oyuncuPrefab.name,new Vector3(0f,
5f,0f),Quaternion.identity,0);
}
```

**48. Adım: Hierarchy** panelinde bulunan **Canvas** nesnesine sağ tıklayarak **UI>Slider** nesnesini seçiniz ve adını **Oyuncu Arayüzü** olarak değiştiriniz. Bu nesnenin **Inspector** panelinde bulunan **Width** değerini **80, Height** değerini **15** olarak ayarlayınız.

**49. Adım: Oyuncu Arayüzü** nesnesinin altındaki **Background** nesnesinin **Inspector** panelinde bulunan **Color** özelliğini kırmızı olarak ayarlayınız.

**50. Adım: Oyuncu Arayüzü** nesnesinin altındaki **Fill Area>Fill** nesnesinin **Inspector** panelinde bulunan **Color** özelliğini yeşil olarak ayarlayınız.

**51. Adım: Oyuncu Arayüzü** nesnesine sağ tıklayıp **UI>Text** nesnesini seçiniz ve adını **Oyuncu İsmi** olarak değiştiriniz.

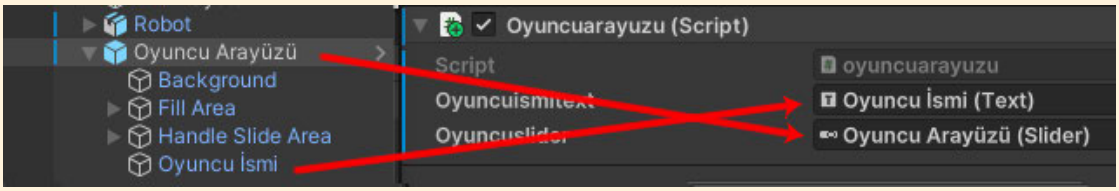
**52. Adım: Oyuncu Arayüzü** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak **Canvas Group** bileşeni ekleyiniz. Bu bileşenin **Interactable** ve **Blocks Raycasts** özelliklerindeki seçimleri kaldırınız.

**53. Adım: Oyuncu Arayüzü** nesnesinin **Inspector** panelinde bulunan **Add Component** butonuna basarak "oyuncuarayuzu" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

```
using UnityEngine;
using UnityEngine.UI;
public class oyuncuarayuzu : MonoBehaviour
```

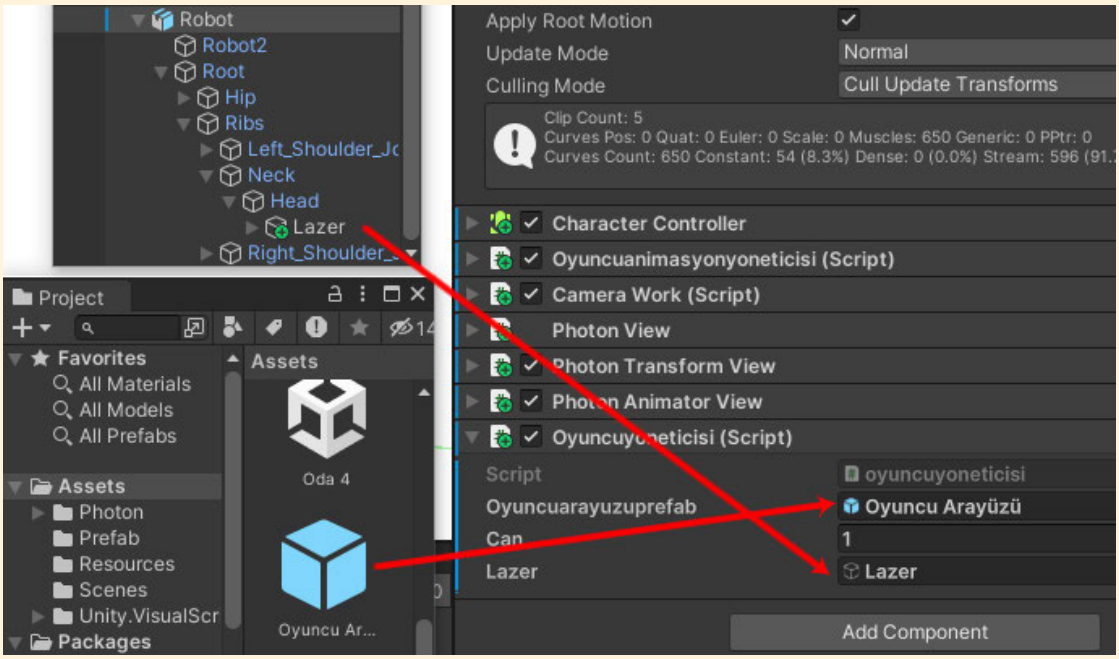
```
{
    private Vector3 offset = new Vector3(0f, 30f, 0f);
    float karakterkontrolcusununyukseligi = 0f;
    Transform hedefTransform;
    Renderer hedefRenderer;
    CanvasGroup canvasgroup;
    Vector3 pozisyon;
    private oyuncuyoneticisi hedef;
    public Text oyuncuismitext;
    public Slider oyuncuslider;
    void Awake()
    {
        this.transform.SetParent(GameObject.Find("Canvas").
GetComponent<Transform>(), false);
        canvasgroup = this.GetComponent<CanvasGroup>();
    }
    void LateUpdate()
    {
        if (hedefRenderer != null)
            this.canvasgroup.alpha = hedefRenderer.isVisible ? 1f
: 0f;
        if (hedefTransform != null)
        {
            pozisyon = hedefTransform.position;
            pozisyon.y += karakterkontrolcusununyukseligi;
            this.transform.position = Camera.main.
WorldToScreenPoint(pozisyon) + offset;
        }
    }
    void Update()
    {
        if (hedef == null)
        {
            Destroy(this.gameObject);
            return;
        }
        if (oyuncuslider != null)
            oyuncuslider.value = hedef.can;
    }
    public void SetTarget(oyuncuyoneticisi _target)
    {
        if (_target == null)
            return;
        hedef = _target;
        hedefTransform = this.hedef.GetComponent<Transform>();
        hedefRenderer = this.hedef.GetComponent<Renderer>();
        CharacterController characterController = _target.
GetComponent<CharacterController>();
        if (characterController != null)
            karakterkontrolcusununyukseligi =
characterController.height;
        if (oyuncuismitext != null)
            oyuncuismitext.text = hedef.photonView.Owner.
NickName;
    }
}
```

**54. Adım: Oyuncu Arayüzü** nesnesini seçip **Inspector** panelinde bulunan oyuncuarayuzu scriptindeki değişkenleri Görsel 4.18'deki gibi ayarlayınız.




**Görsel 4.18:** Oyuncuarayuzu scriptindeki değişkenleri ayarlama

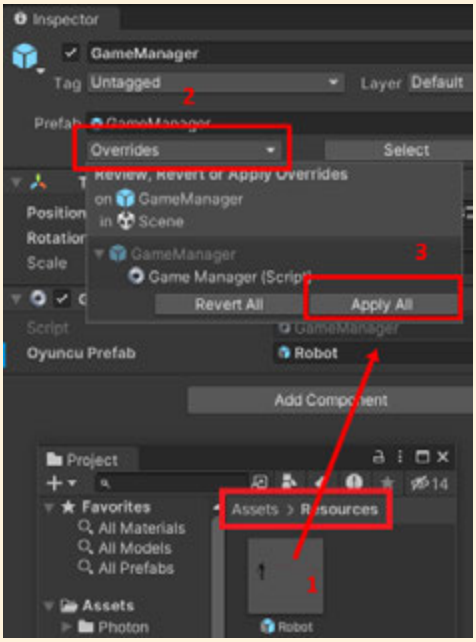
**55. Adım: Oyuncu Arayüzü** nesnesini **Project** panelinde bulunan "Assets>Prefabs" klasörüne tutup sürükleyerek **Prefab** oluşturunuz. Daha sonra **Oyuncu Arayüzü** nesnesini sahneden siliniz.
**56. Adım: Hierarchy** panelinde bulunan **Robot** nesnesini seçip **Inspector** panelinde bulunan oyuncuyoneticisi scriptindeki değişkenleri Görsel 4.19'daki gibi ayarlayınız.




**Görsel 4.19:** Oyuncuyoneticisi scriptindeki değişkenlerin ayarlanması

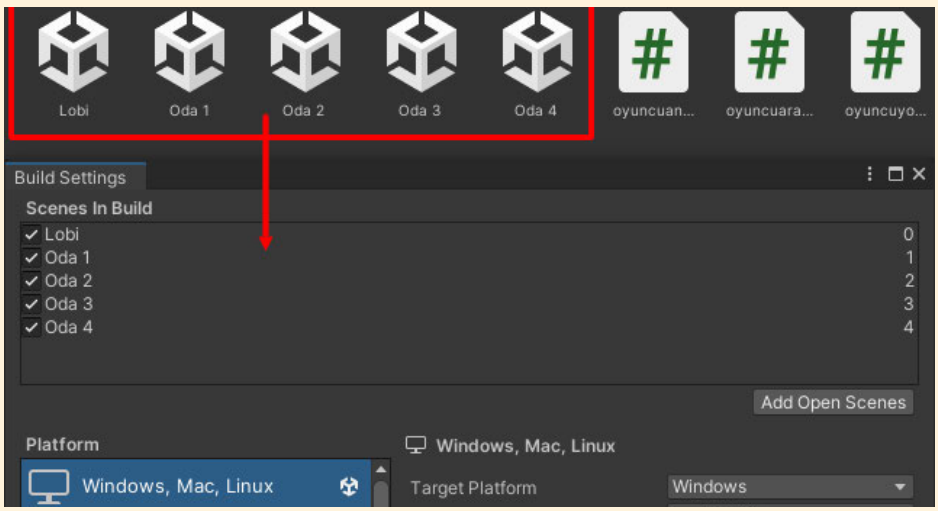
**57. Adım: Project** panelinde bulunan **Assets** klasörüne sağ tıklayıp **Create>Folder** seçeneğini seçiniz ve yeni boş bir klasör oluşturunuz. Bu klasörün adını **Resources** olarak değiştiriniz. Daha sonra **Hierarchy** panelinde bulunan **Robot** nesnesini tutup, bu klasöre sürükleyerek **Prefab** oluşturunuz. Ardından **Robot** nesnesini sahneden siliniz.
**58. Adım: Hierarchy** panelinden **GameManager** nesnesini seçip **Inspector** panelinde bulunan **GameManager** scriptindeki **Oyuncu Prefab** değişkenine bir önceki adımda oluşturulan **Robot Prefabını** atayınız.

Ardından **Overrides** bölümüne girerek **Apply All** butonuna basınız (Görsel 4.20).




**Görsel 4.20:** GameManager bileşenini ekleme

**59. Adım: Oda 1, Oda 2** ve **Oda 3** sahnelerindeki **GameManager** nesnelerinin **Oyuncu Prefab** değişkenlerine de aynı atamaları yapıp bu sahneleri **File>Save** seçeneği ile kaydediniz.
**60. Adım: File>Build Settings** menüsüne girerek sahnelerin hepsini derlemeye dâhil ediniz. Lobi sahnesinin "0" numaralı sahne olmasına dikkat ediniz (Görsel 4.21).




**Görsel 4.21:** Sahnelerin derlemeye dâhil edilmesi

**61. Adım: Build Settings** penceresinde bulunan **Build** butonuna basarak, kaydedilecek klasörü seçip oyunu istediğiniz yere kaydediniz.
**62. Adım:** Farklı bilgisayarlardan oyunu çalıştırıp deneyiniz. Oyuncu sayısı arttıkça oyun alanı genişleyecektir. Bir oyunda en fazla 4 oyuncu bulunabilir. 5. oyuncu oyuna bağlandığı zaman yeni oyun oluşturulacaktır.

SIRA SİZDE

6. Uygulamadaki oyunu 5 kişi oynayacak şekilde yeniden düzenleyiniz. Bunun için Oda5 isminde, 5 kişinin etkileşime girebileceği genişlikte bir sahne daha oluşturup kodlarda gerekli düzenlemeleri yapınız.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken bu ölçütleri dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Oda5 isimli sahneyi oluşturdu. | | |
| 2. Sahnede bulunan duvar ve zemin nesnelerini gerekli genişlikte ayarladı. | | |
| 3. Kodlarda gerekli düzenlemeleri yaptı. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 4.1.4. Alışveriş ve Nesne Yapma Sistemleri

Oyun içinde sanal veya gerçek para ile satın alma işlemine **Alışveriş**, farklı nesneleri birleştirerek yeni nesne oluşturma işlemine ise **Nesne Yapma (Crafting)** denir.

Oyunların ilk zamanlarında alışveriş ve nesne yapma sistemleri son derece basitti. Oyun içinde geçilen bir bölüm veya tamamlanan bir görevden sonra oyun içi para kazanılır ve bu para ile alışveriş yapılırdı. Nesne yapma sistemi ise bu dönemde oldukça kısıtlıydı.

Teknoloji geliştikçe oyun yapımcıları tarafından yeni mekanikler de denenmeye başlandı. **Rol Yapma Oyunlarında (RPG-Role Playing Game)** oyuncular, malzeme toplayıp bunlarla yeni eşyalar üretebilir ve satabilir hâle geldi. Bu sistem beğenilince başka oyun türlerinde de kullanılmaya başlandı.

İnternet üzerinden oynanan oyunlar popülerleştikçe oyunlarda mikro ödeme sisteminin kullanımı da yaygınlaştı. Mikro ödeme sistemi ile gerçek para, oyun içi para ile beraber kullanıldı hatta onun yerine geçti. Böylece oyuncular; karakterlerinin görünümünü değiştirecek kozmetik ürünler, kendilerine avantaj sağlayacak paketler ve güçlendirmeler almaya başladı.

7. UYGULAMA

Uygulama adımlarını takip ederek alışveriş yapılabilecek bir arayüz tasarlayınız.

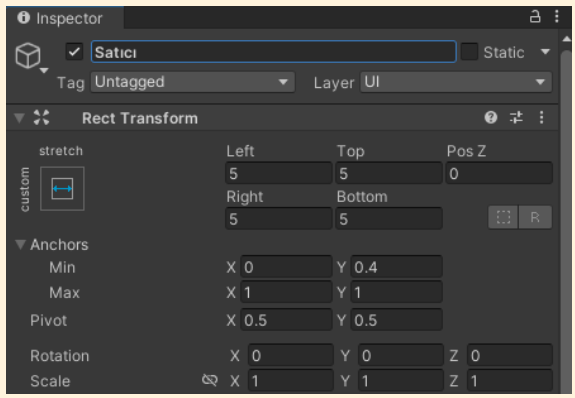
**1. Adım:** Oyun motorunda yeni boş bir 2D Core uygulaması oluşturunuz.

**2. Adım: Asset Store** kullanarak, **Fantasy Inventory Icons [Free]** paketini indirip projeye ekleyiniz.

**3. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **UI>Canvas** seçeneğini seçiniz.

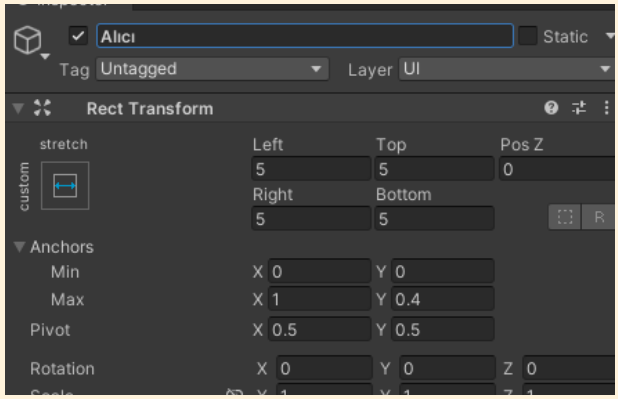
**4. Adım: Canvas** nesnesini seçip, sağ tıklayarak **UI** menüsünden 2 adet **Image** nesnesi ekleyiniz.

**5. Adım: 1. Image** nesnesinin adını **Satıcı** olarak değiştirip, **Inspector** panelinden **Source Image** özelliğini **Grey** olarak ayarlayınız ve **Rect Tranform** ayarlarını Görsel 4.22'deki gibi yapınız.




**Görsel 4.22:** Satıcının Rect Transform Ayarı

**6. Adım: 2. Image** nesnesinin adını **Alıcı** olarak değiştirip, **Inspector** panelinden **Source Image** özelliğini **Grey** olarak ayarlayınız ve **Rect Tranform** ayarlarını Görsel 4.23'teki gibi yapınız.

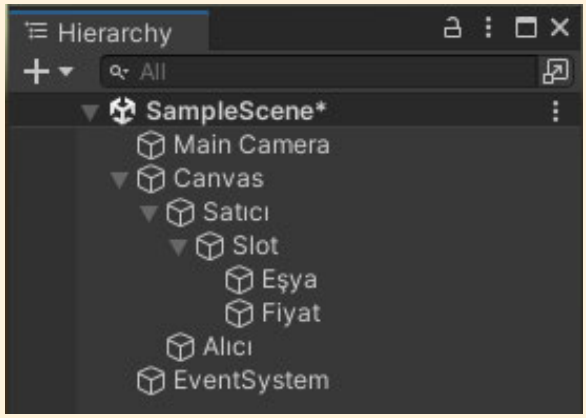



**Görsel 4.23:** Alıcının Rect Transform ayarı

**7. Adım: Hierarchy** panelinden **Satıcı** nesnesini seçip, sağ tıklayarak **UI>Image** nesnesini seçiniz ve adını **Slot** olarak değiştiriniz. **Inspector** panelinden **Slot** nesnesinin **Source Image** özelliğini **Brown** olarak ayarlayınız. **Add Component** kısmından **Button** bileşenini ekleyiniz.

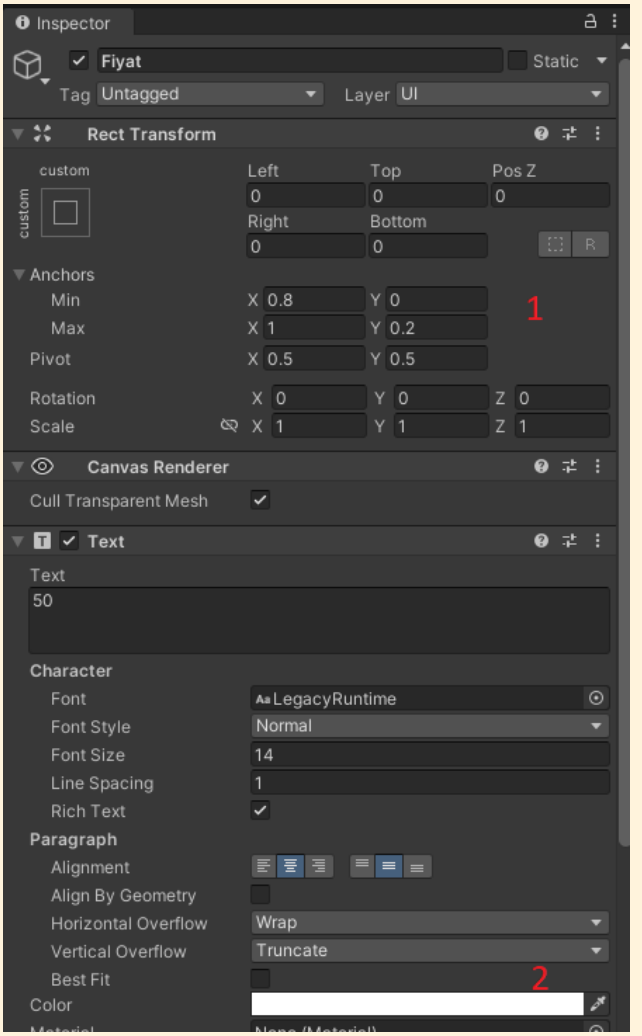
**8. Adım: Hierarchy** panelinden **Slot** nesnesini seçip, sağ tıklayarak **UI>Image** nesnesini seçiniz ve adını **Eşya** olarak değiştiriniz.

**9. Adım: Hierarchy** panelinden **Slot** nesnesini seçip, sağ tıklayarak **UI>Text** nesnesini seçiniz ve adını **Fiyat** olarak değiştiriniz. Sonuç olarak Görsel 4.24'teki gibi hiyerarşik bir yapı elde etmelisiniz.



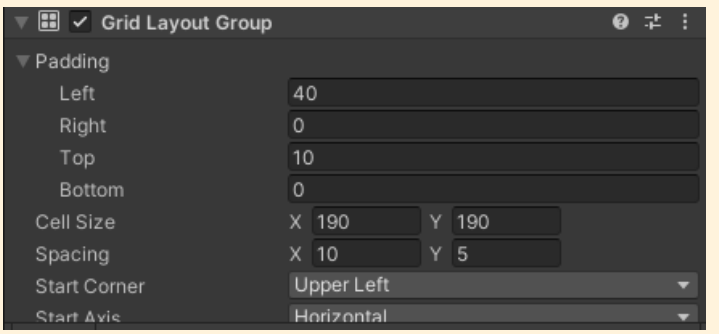

**Görsel 4.24:** Arayüz tasarımı

**10. Adım: Hierarchy** panelinden **Fiyat** nesnesini seçip **Inspector** panelinde bulunan **Rect Transform** kısmındaki değerleri görseldeki gibi ayarlayınız. Yazı rengini de beyaz olarak değiştiriniz (Görsel 4.25).




**Görsel 4.25:** Fiyat nesnesinin Rect Transform ve Color özelliklerini ayarlama

**11. Adım:** Slot nesnesini seçip, Ctrl+D tuş kombinasyonunu kullanarak 9 adet kopyasını oluşturunuz.
**12. Adım: Satıcı** nesnesini seçip **Inspector** panelinde bulunan **Add Component** kısmından **Grid Layout Group** ekleyiniz ve ayarlarını Görsel 4.26'daki gibi yapınız.




**Görsel 4.26:** Grid Layout Group ayarı

**13.Adım:** 11. adımda çoğalttığınız Slot nesnelerinin altında bulunan Eşya nesnelerine 2. adımda indirdiğiniz paketten istediğiniz ikonları ekleyiniz. Yine Slot nesnelerinin altında bulunan Fiyat nesnelerine de istediğiniz fiyatları yazarak Görsel 4.27'dekine benzer bir görünüm elde etmeye çalışınız.

**14. Adım: Alıcı** nesnesine de **Satıcı** nesnesine uygulanan ayarları kullanarak Görsel 4.28'deki görünümü elde etmeye çalışınız.



**Görsel 4.27:** Satıcı arayüzü



**Görsel 4.28:** Alışveriş arayüzü

**15. Adım: Hierarchy** panelinden **Canvas** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak "alisveris" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

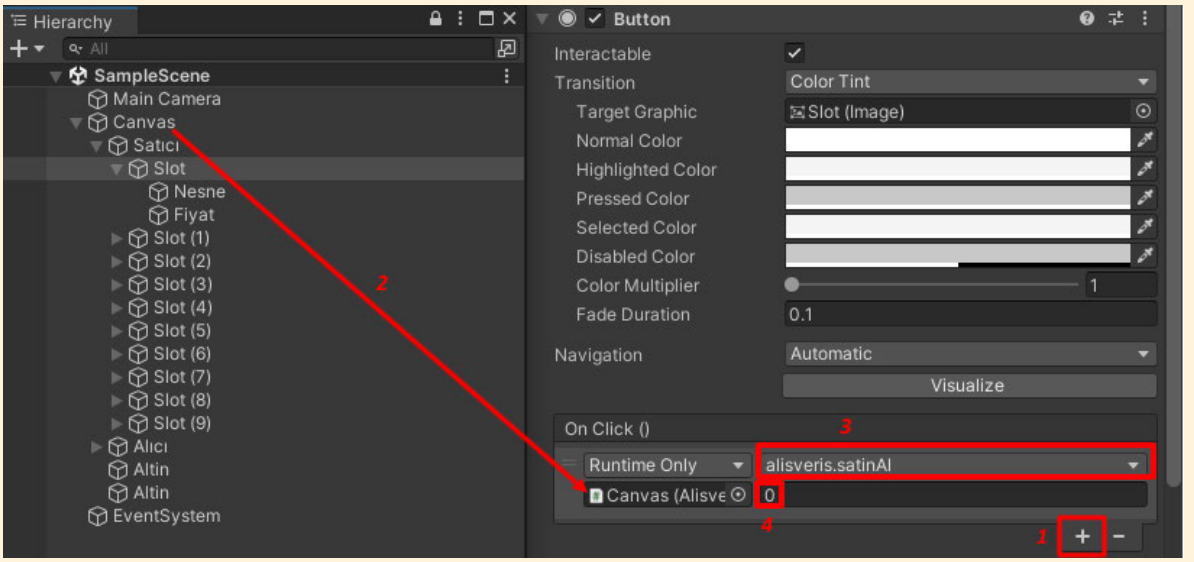
```
using UnityEngine;
using UnityEngine.UI;
public class alisveris : MonoBehaviour
{
    public GameObject[] satilanlar;//satıcı slotları
    public GameObject[] alinanlar;//alıcı slotları
    public Text altin;//ne kadar altınımız kaldığını gösteren
Text
    [HideInInspector]public int bos=0;//alıcıdaki boş slot
    public int oyuncuAltin=500;
    public void satinAl(int c)
    {
        //Tıklanan nesnenin fiyatı alınıyor
        int esyaDegeri = int.Parse(satilanlar[c].
GetComponentInChildren<Text>().text);
        if (esyaDegeri<=oyuncuAltin)//Altınımız yeterli mi
        {
            if (bos<=4)//alıcı da boş slot var mı
            {
                oyuncuAltin -= esyaDegeri;
                altin.text = oyuncuAltin.ToString();
                alinanlar[bos].GetComponentsInChildren<Image>()
[1].sprite = satilanlar[c].GetComponentsInChildren<Image>()[1].
sprite;
                bos++;
            }
        }
    }
}
```

**16. Adım: Canvas** nesnesindeki alisveris scriptine görseldeki eklemeleri yapınız (Görsel 4.29).




**Görsel 4.29:** Alisveris scriptinin ayarlanması

**17. Adım: Satıcı** kısmındaki birinci **Slot** nesnesini seçerek **Button** bileşeninin en altındaki "+" düğmesine basınız. **On Click()** kısmına **Canvas** nesnesini tutup sürükleyiniz. **No Function** yazan kısma **satinAl (int)** fonksiyonunu ayarlayıp değer olarak "0" giriniz. Diğer Slot nesnelerine de aynı işlemleri uygulayarak "0" değeri yerine "1" den "9" a kadar değerleri sırayla giriniz (Görsel 4.30).




**Görsel 4.30:** Button ayarlarının yapılması

**18. Adım:** Uygulamayı çalıştırıp satıcıdaki herhangi bir nesneye tıkladığınız zaman, tıklanan nesne alıcı kısmına eklenecek ve altınımızdan düşecektir. Envanterimiz dolduğunda veya altınımız yeterli olmadığında satın alma işlemi gerçekleşmeyecektir.

## 4.2. TEST ETME VE İYİLEŞTİRME İŞLEMLERİ

Oyun geliştirmeye başlamadan önce karakterlerin, senaryonun, oyun mekaniklerinin nasıl olacağıyla ilgili taslak çalışmalar yapılır ve oyunlar buna göre geliştirilir ama bu çalışmalar oyuna aktarılırken öngörülemeyen pek çok sorunla karşılaşılabilir. Bu yüzden oyun geliştirme sürecinde test etme ve iyileştirme işlemleri sürekli yapılmalıdır. Oyun piyasaya çıktıktan sonra da kullanıcı geri bildirimlerine göre oyunda iyileştirmeler yapılmaya devam edilir.

### 4.2.1. Oyunun Oynanışını Test Etme

**Test işlemi,** geliştirme sürecinin en önemli aşamalarından birisidir. **Test süreci;** oyun geliştirme aşamasında karşılaşılan hataları gidermeyi, zorluk dengesini ve oyun mekaniklerini ayarlamayı, oyunun farklı platformlarda nasıl çalıştığını görmeyi, oyunun kalitesini ve oyuncu deneyimini artırmayı sağlayan bir süreçtir.

Test sürecinin kısımları şunlardır:

- **Dâhili Testler:** Geliştirme ekibi, oyunu sık sık test etmeli ve oyunda gördükleri hataları gidermelidir.
- **Alfa Testi:** Oyunun alfa sürümü oyunun ilk versiyonudur. Bu sürümde oyun mekanikleri denenir. Çevre ve karakter etkileşimleri ise çok azdır. Bu sürümde geliştiriciler oyundaki hataları mümkün olduğunca gidermeye çalışırlar.
- **Kapalı ve Açık Beta Testi:** Beta sürümü oyunun son hâline en yakın sürümdür. Beta testinin amacı oyunun performansını ve kullanıcı deneyimini test etmektir. Alfa testinde giderilemeyen hatalar varsa bu hatalar da giderilir. Kapalı beta testi yine kısıtlı sayıda oyuncu ile yapılır. Açık beta testine ise daha çok kullanıcı katılır ve farklı cihazlardaki kullanım senaryolarını görmek için çok kullanışlıdır.
- **Platform Testleri:** Oyun birkaç farklı platforma geliştiriliyor ise platformların hepsinde oyunu ayrı ayrı denemek gerekir çünkü farklı platformlarda donanımlar, ekran çözünürlükleri, işletim sistemleri değişkenlik gösterir. Arayüz, birinde iyi görünürken diğerinde nesneler birbirine girebilir; animasyonlar birinde iyi çalışırken diğerinde kötü görünebilir.

#### 8. UYGULAMA

Uygulama adımlarını takip ederek, daha önce yapılan uygulamalardan birini Windows uygulaması hâline getirip test ediniz.

**1. Adım:** Oyun motorunda test etmek istediğiniz bir uygulamayı açınız.

**2. Adım: File>Build Settings** menüsüne giriniz.

**3. Adım:** Oyun tek bir sahneden oluşuyorsa **Add Open Scenes** butonuna basarak derleme işlemine dâhil ediniz. Birden fazla sahneniz varsa sahneleri **Project** panelinde bulunan **Assets** klasöründen seçip sürükle bırak ile derleme işlemine dâhil ediniz.

**4. Adım:** Yanında ''0'' yazan sahne, oyunun açılış sahnesidir. Açılmasını ilk istediğiniz sahneyi ''Fare'' ile tutup sürükleyerek en üste alınız (Projede olmasını istemediğiniz sahnelerin başındaki tik işaretini kaldırabilirsiniz).

**5. Adım: Build And Run** butonuna basıp oyunu nereye kaydedeceğinizi seçiniz.

**6. Adım:** Oyunu farklı bilgisayarlarda deneyerek, oyunda ne gibi sorunların olduğunu not alıp oyundaki bu hataları gideriniz.

### 4.2.2. Oyun İçi Reklam ve Analiz Desteği

**Oyun İçi Reklam Ekleme**

Günümüzde mobil oyunların çoğu ücretsizdir ve yapımcılar gelirini oyun içi reklamlar ile sağlarlar. Bu reklamlar, oyun yapımcıları tarafından stratejik bir şekilde yerleştirilmelidir. Çok fazla reklam kullanılırsa oyuncuların oyun deneyimi kesintiye uğrayabilir ve oyuncular oyunu terk edebilirler. Az reklam kullanılırsa da yapımcıyı tatmin edecek bir gelir elde edilmeyebilir. Bu yüzden dengeyi iyi kurmak gerekir.

Oyunlara eklenebilecek reklam türleri şunlardır:

- **Banner (Afiş):** Ekranın bir köşesinde sürekli duran, pozisyonu ayarlanabilir, küçük boyutlu reklamlardır. Az yer kapladığı için kullanıcıyı fazla rahatsız etmez ama geliri düşüktür.
- **Interstitial (Geçiş):** Ekranı tamamen kaplayan, fotoğraf veya video reklamlarıdır. Bölüm geçişlerinde daha çok kullanılır. Geliri yüksektir ama çok sık kullanılırsa kapatmak için belirli süre gerektiğinden (genellikle 5 sn.) oyuncular sıkılıp oyunu oynamayı bırakabilirler.
- **Rewarded (Ödüllü):** Ekranı tamamen kaplayan, sonuna kadar izlenince (genellikle 30 sn.) oyuncuya ödül kazandıran reklamlardır. Oyuncu, süresi dolmadan reklamı kapatırsa ödül alamaz. Ayrıca oyuncu, reklamı izledikten sonra vadedilen ödül oyuncuya verilmelidir. Oyuncu, ödülünü alamazsa oyunu terk edebilir. Bu reklam türü, geliri en yüksek reklam türüdür.
- **Native (Doğal):** Ekranın tamamını kaplayan, oyunun bir parçasıymış gibi görünen hatta bazılarında kullanıcıların etkileşime girebildiği reklamlardır.

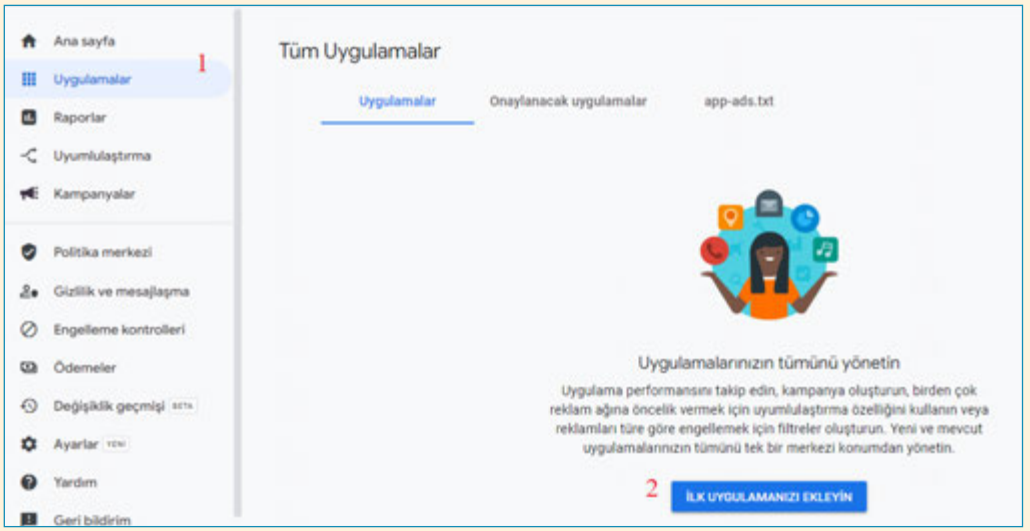
Reklam gelirlerini belirleyen faktörler şunlardır:

- **Platform:** Farklı platformlarda yayımlanan reklamların gelirleri farklıdır.
- **Bölge:** Reklam gelirleri reklam veren firmaların bölgelerine göre büyük değişkenlik göstermektedir çünkü bu reklamların parasını reklam veren firmalar ödemektedir ve yabancı firmalar reklam işine daha fazla bütçe ayırmaktadır. Bu da genellikle İngilizce içerikli bir oyunun Türkçe içerikli bir oyundan daha fazla reklam geliri elde edeceği anlamına gelir.
- **Dönem:** Reklam gelirleri belirli dönemlere göre de farklılık göstermektedir. Bazı reklam türleri, bazı dönemlerde daha fazla gelir getirmektedir. Reklam sistemlerinin reklam analizi araçları ile bunlar takip edilebilir.
- **Geri Bildirim:** Gösterilen reklamlarda geri bildirim alanları bulunmaktadır. Kullanıcı, gösterilen reklamın uygulama ile alakasız olduğu geri bildiriminde bulunursa o reklamdan gelir elde edilemez.
- **Bot Kullanımı:** Reklam sistemlerinin, yapımcıların kendi reklamını tıklaması veya bot (sizin yerinize reklamlara tıklayan sistemler) satın alarak haksız kazanç elde etmesiyle ilgili çok katı kuralları ve bunları algılayabilen gelişmiş algoritmaları vardır. Bu yüzden reklamların oyunda nasıl göründüğü denenirken test reklamları kullanılmalıdır. Aksi hâlde bu sistemler reklam akışını durdurabilir, ücret ödemesi yapmayabilir hatta hesabınızı kalıcı olarak kapatabilir.

## 9. UYGULAMA

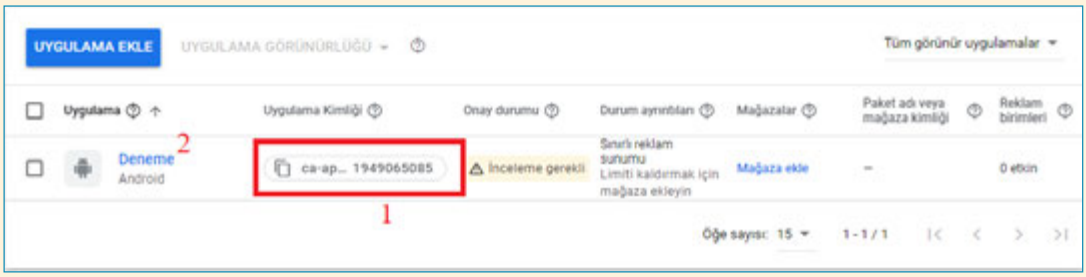
Uygulama adımlarını takip ederek oyununuza **Google AdMob** sistemiyle, **Afiş (Banner)** reklam ekleyiniz.

**1. Adım: Google AdMob** hizmetine üye olunuz.
**2. Adım: Google AdMob** hizmetine giriş yaptıktan sonra sol taraftaki menüden **Uygulamalar** seçeneğini seçip açılan ekranda **İlk Uygulamanızı Ekleyin** butonuna basınız (Görsel 4.31).




**Görsel 4.31:** Uygulama ekleme

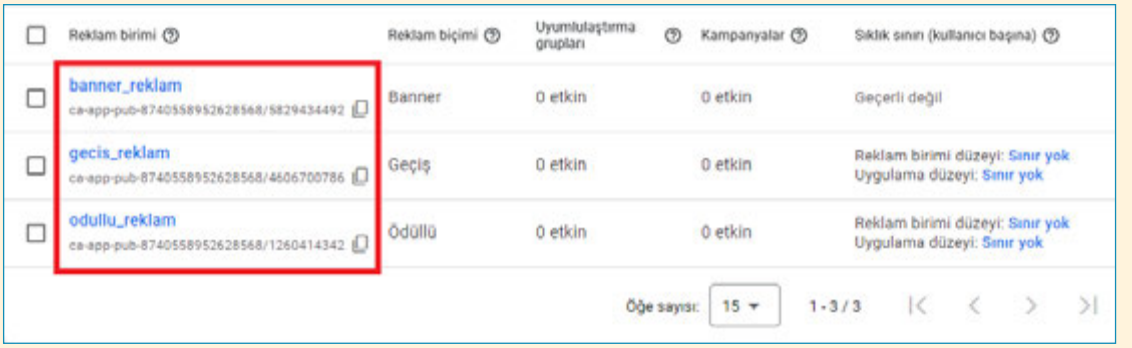
**3. Adım:** Yeni açılan pencerede **Platform** seçeneğini **Android**, **Uygulama desteklenen bir uygulama mağazasında listeleniyor mu?** seçeneğini **Hayır** olarak seçip devam ediniz (Uygulamanızı bir uygulama marketine yükledikten sonra bu seçeneği değiştirebilirsiniz.).
**4. Adım:** Sonraki pencerede oyununuza bir isim giriniz. **Kullanıcı metrikleri** seçeneği de seçili olmalıdır. Bu sayede reklamların kullanıcı analizlerini de görebilirsiniz.
**5. Adım:** Uygulamanızın reklam kimliğini bir yere not ediniz ve kimse ile paylaşmayınız. Oyun ismine tıklayıp sonraki adıma geçiniz (Görsel 4.32).




**Görsel 4.32:** Reklam kimliği

**6. Adım: Reklam Birimi Ekleyin** butonuna basıp **Banner** reklam ekleyiniz ve adını banner_reklam yaparak **Reklam birimi oluşturun** butonuna basınız.
**7. Adım:** 6. adımı tekrarlayarak bu sefer **Geçiş** reklamı ekleyip adını **gecis_reklam** yapınız.
**8. Adım:** 6. adımı tekrarlayarak **Ödüllü** reklam ekleyip adını **odullu_reklam** olarak ayarlayınız. Ödüllü reklamlarda diğer reklamlardan farklı olarak **Ödül Ayarları** kısmı da bulunmaktadır. **Ödül Miktarı** değerini **100**, **Ödül Ögesi** değerini **Para** olarak ayarlayınız.

**9. Adım:** Eklenen reklam birimlerinin reklam kimliklerini bir yere not ediniz (Görsel 4.33).



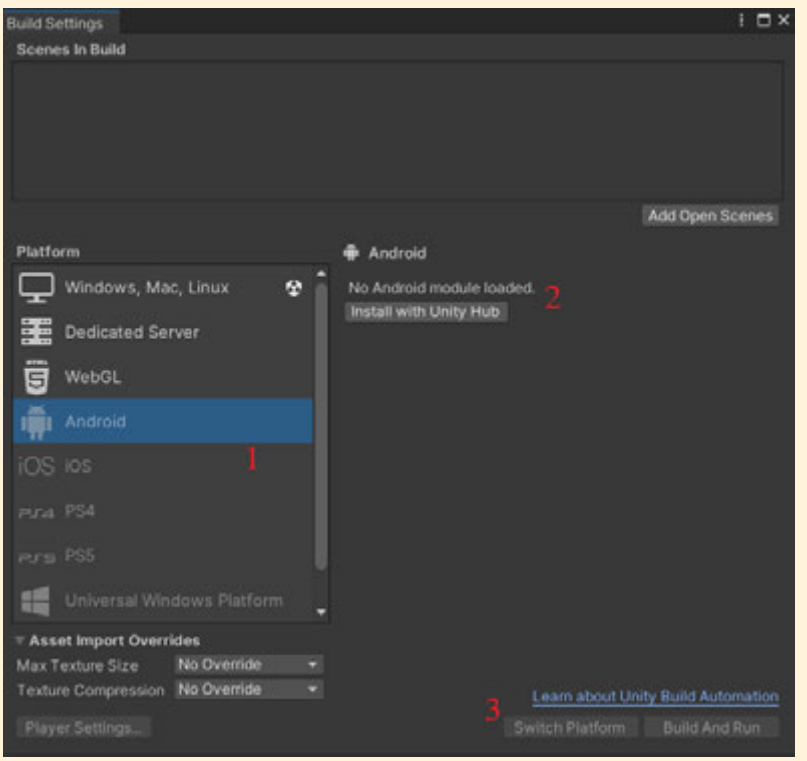

**Görsel 4.33:** Reklam kimlikleri

**10. Adım:** https://cdnmeslek.eba.gov.tr/bilisim/oyun/mp.rar adresinden oyun motoruna reklam eklemek için gerekli olan eklentiyi indiriniz.

**11. Adım:** Android işletim sistemine oyun geliştirebilmek için bilgisayarınızda **Android Studio** kurulu olmalıdır. Android Studio'da More Actions>SDK Manager menüsündeki **SDK Tools** kısmından **Google Play Services** bileşenini kurunuz. Kurulu ise güncel olduğundan emin olunuz.

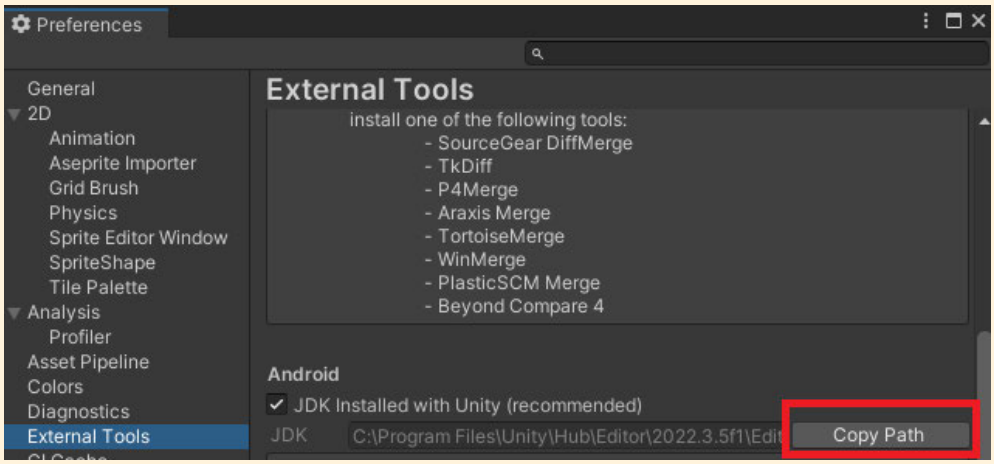
**12. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.

**13. Adım: File>Build Settings** menüsünden **Android** platformunu seçip **Switch Platform** butonuna basınız. Bu buton aktif değilse **Install With Hub** butonuna basarak **Android** platformunu kullanmak için gerekli dosyaların kurulumunu yapınız. Kurulum bittikten sonra **Switch Platform** butonuna basınız. Oyun motoru, oyun dosyalarını **Android** platformuna göre ayarlayacaktır (Görsel 4.34).



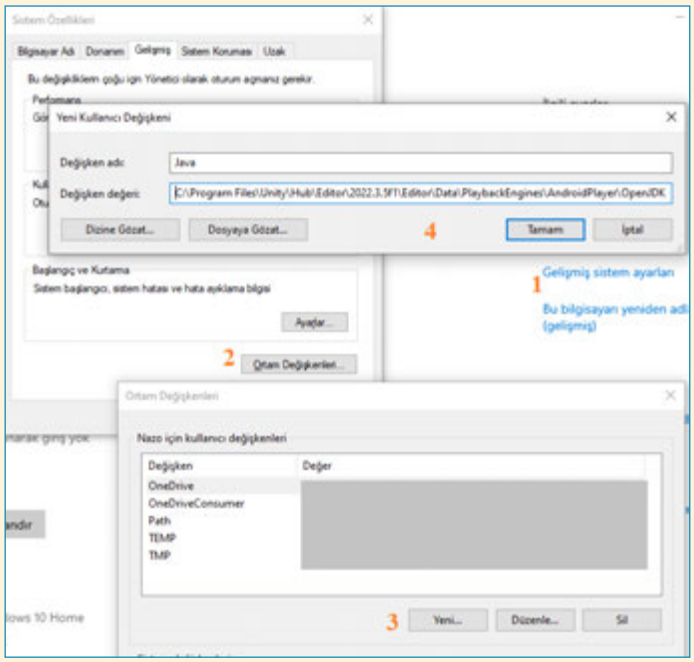

**Görsel 4.34:** Android platformu seçimi

**14. Adım: Edit/Preferences** menüsü ile açılan pencerede **External Tools** menüsüne geçip **JDK** seçeneğinin yanındaki **Copy Path** butonuna basınız (Görsel 4.35).




**Görsel 4.35:** JDK yolunu kopyalama

**15. Adım:** Masaüstünde **Bu Bilgisayar** simgesine sağ tıklayarak **Özellikler** seçeneğini seçiniz. Açılan pencerede **Gelişmiş Sistem Ayarları** menüsüne giriniz. Yeni açılan pencerede **Ortam Değişkenleri** butonuna basınız. **Kullanıcı değişkenleri** veya **Sistem değişkenleri** bölümünde Java ile ilgili bir girdi yoksa **Yeni** butonuna basarak, açılan penceredeki **Değişken Adı** kısmına **Java** yazıp **Değişken Değeri** kısmına da bir önceki adımda kopyalanan **JDK** yolunu yapıştırınız (Görsel 4.36).




**Görsel 4.36:** Java ayarları

NOT

Bu adım sayesinde Android derlemesi için gerekli olan Java dosyalarının yolu Windows'a tanıtılmıştır.

**16. Adım: Assets>Import Package>Custom Package** menüsünden 10. adımda indirdiğiniz **GoogleMobileAds** eklentisini seçip projenize ekleyiniz.
**17. Adım: Assets>External Dependency Manager>Android Resolver** menüsünden **Resolve** seçeneğini seçerek işlemin bitmesini bekleyiniz.
**18. Adım: Assets>Google Mobile Ads** menüsünden **Settings** seçeneğini seçip **Inspector** panelinde Android kısmına 5.adımda oluşturulan reklam kimliğinizi yapıştırınız.
**19. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **Create Empty** seçeneği ile boş bir nesne oluşturunuz ve adını reklam olarak değiştiriniz.
**20. Adım:** Reklam nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak "reklambanner" isimli bir script ekleyerek aşağıdaki kodları yazınız.

```
using UnityEngine;
using GoogleMobileAds.Api;
public class reklambanner : MonoBehaviour
{
    private BannerView reklamNesnesi;
    private string bannerKimlik="ca-app-
pub-3940256099942544/6300978111";
    void Start()
    {
        // Initialize metodu reklam gösterilmeden önce bir sefer
        //çağrılmalıdır.
        MobileAds.Initialize( reklamDurumu => {});
        // BannerView nesnesi ile reklam boyutu ve pozisyonu //
        //ayarlanır.
        reklamNesnesi = new BannerView(bannerKimlik, AdSize.
SmartBanner, AdPosition.Top);
        //AdRequest nesnesi ile reklam havuzundan reklam //
        //çağrılır.
        AdRequest reklamAl = new AdRequest();
        //LoadAd nesnesi ile reklam yüklenir.
        reklamNesnesi.LoadAd(reklamAl);
    }
    void OnDestroy()
    {
        //Reklamlar kullanıldıktan sonra silinmelidir. Yoksa
        //bellek sorunlarına yol açar.
        if(reklamNesnesi !=null)
        reklamNesnesi.Destroy();
    }
}
```

**21. Adım:** Uygulamayı çalıştırınca üst kısımda bir şerit belirecektir. Oyunu derleyip, mobil cihazda deneyince **bu şerit yerine AdMob sisteminin test reklamı** görünecektir. "ca-app-pub-3940256099942544/6300978111" AdMob'un banner reklam almaya yarayan test kimliğidir. Oyununuzu markette yayımladığınız zaman bu reklam kimliğini 6. adımda oluşturduğunuz banner reklam kimliğiniz ile değiştirmelisiniz.

## 10. UYGULAMA

Uygulama adımlarını takip ederek oyununuza, **Google AdMob** sistemiyle, **Geçiş (Interstitial)** türünde reklam ekleyiniz.

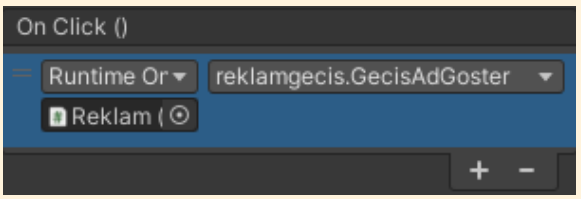
**1. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.
**2. Adım: File>Build Settings** menüsünden **Android** platformunu seçip **Switch Platform** butonuna basınız.
**3. Adım:** https://cdnmeslek.eba.gov.tr/bilisim/oyun/mp.rar adresinden oyun motoruna reklam eklemek için gerekli olan eklentiyi indiriniz. Daha sonra **Assets>Import Package>Custom Package** menüsünden eklentiyi seçip projenize ekleyiniz.
**4. Adım: Assets>External Dependency Manager>Android Resolver** menüsünden **Resolve** seçeneğini seçerek işlemin bitmesini bekleyiniz.
**5. Adım: Assets>Google Mobile Ads** menüsünden **Settings** seçeneğini seçip **Inspector** panelinde **Android ve IOS kısımlarına** 9. Uygulamada oluşturulan reklam kimliğinizi yapıştırınız.
**6. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **Create Empty** seçeneği ile boş bir nesne oluşturunuz ve adını **Reklam** olarak değiştiriniz.

**7. Adım: Reklam** nesnesini seçip, **Inspector** panelindeki **Add Component** butonuna basarak "reklamgecis" isimli bir script ekleyerek aşağıdaki kodları yazınız.

```
using UnityEngine;
using System.Collections;
using GoogleMobileAds.Api;
public class reklamgecis : MonoBehaviour
{
    private InterstitialAd reklamNesnesi;
    private string gecisKimlik="ca-app-
pub-3940256099942544/1033173712";
    void Start()
    {
        MobileAds.Initialize( reklamDurumu => {});
        GecisAdYukle();
    }
    public void GecisAdYukle()
    {
        if (reklamNesnesi != null)
        {
            reklamNesnesi.Destroy();
            reklamNesnesi = null;
        }
        var adRequest = new AdRequest();
        InterstitialAd.Load(gecisKimlik, adRequest,
        (InterstitialAd reklam, LoadAdError hata) =>
        {
            if (hata != null || reklam == null)
            {
                Debug.LogError("Hata: " + hata);
                return;
            }
            Debug.Log("Reklam Yüklendi : "+ reklam.
GetResponseInfo());
            reklamNesnesi = reklam;
        });
```

```
    }
    public void GecisAdGoster()  // Show
    {
        if (reklamNesnesi != null && reklamNesnesi.CanShowAd())
        {
            reklamNesnesi.Show();
        }
    }
}
```

**8. Adım: Hierarchy** paneline sağ tıklayarak **UI>Button** nesnesini seçiniz ve en alttaki "+" butonuna basınız. Reklam nesnesini buraya sürükleyip bırakarak **reklamgecis scriptindeki GecisAdGoster ()** metodunu seçiniz (Görsel 4.37).




**Görsel 4.37:** Butona reklam gösterme olayını atama

**9. Adım:** Uygulamayı çalıştırınca **Console** panelinde reklamın yüklendiğini gösteren bir mesaj belirecek ama butona basınca reklam görüntülenmeyecektir. Reklamın görünmesi için oyunu derleyip mobil cihazda deneyiniz. "ca-app-pub-3940256099942544/1033173712" AdMob'un geçiş reklamı almaya yarayan test kimliğidir. Oyununuzu markette yayımladığınız zaman bu reklam kimliğini 9. Uygulamada oluşturduğunuz **geçiş reklamı kimliğiniz ile** değiştirmelisiniz.

SIRA SİZDE

Herhangi bir projenize Ödüllü (Rewarded) türünde reklam ekleyiniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

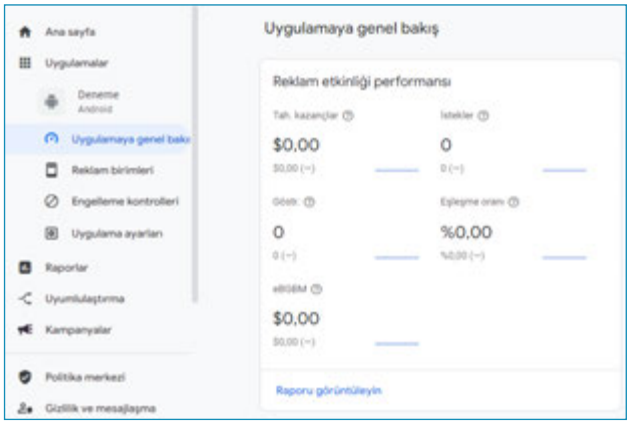
**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Yeni proje oluşturup GoogleMobileAds paketini projeye ekledi. | | |
| 2. Google reklam kimliğini projeye ekledi. | | |
| 3. Ödüllü (Rewarded) reklam için gerekli scripti yazdı. | | |
| 4. Scripti bir butona ekledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

**Reklam Analiz Desteği**

Reklam sistemlerinin reklam havuzları bulunmakta ve daha çok indirilen ve oynanan oyunlara daha fazla reklam sağlanmaktadır. Oyun, uygulama marketlerinden birinde yayımlandıktan sonra reklamların gözükmesi bu yüzden biraz zaman alabilir hatta ilk günlerde hiç reklam görüntülenmeyebilir. Oyun, indirildikçe ve etkileşim aldıkça reklam havuzundan reklam alınmaya başlanacaktır.

**Google AdMob** sitesinde **Uygulamaya Genel Bakış** gösterge tablosunda oyunda gösterilen reklamlarla ilgili analizler yer alır. Analizler sayesinde hesapla ilgili değişimler takip edilebilir (Görsel 4.38).




**Görsel 4.38:** Reklam analizi

Bu ekrandaki bazı önemli göstergeler şunlardır:

- **Tahmini Kazançlar:** Gösterilen reklamların tahmini olarak ne kadar gelir getirdiğini gösteren değerdir. Ay sonunda gerçek değer hesaplandığında tahmini kazançla farklılık gösterebilir.
- **İstekler:** Oyundan gelen reklam isteği sayısıdır.
- **Gösterim:** Kullanıcılara gösterilen reklam sayısıdır.
- **Eşleşme Oranı:** Uygulamadan gelen istek sayısı ile eşleşen istek sayısının oranıdır. Bu oran ne kadar yüksekse oyun, reklam havuzundan o kadar çok reklam alıyordur.
- **eBGBM:** Bin reklam başına tahmini gelir değeridir. Bu değer, oyuna göre değişkenlik gösterir çünkü oyun ne kadar çok etkileşim alıyorsa reklam havuzundan o kadar yüksek ücretli reklamlar ile eşleşir.

Bunun yanında reklam kimliği oluştururken **Kullanıcı Metrikleri** seçeneği seçildiyse 48 saati bulan bir incelemenin ardından **Kullanıcı Metrikleri** bölümü de aktif hâle gelecektir. Bu kısımda da günlük kullanıcı sayısı, oyunu yükleyip oynamaya devam eden kullanıcı sayısı, kullanıcıların oyunda harcadığı ortalama süre, reklamlara tıklama oranı gibi önemli göstergeler bulunur (Görsel 4.39).




**Görsel 4.39:** Kullanıcı metrikleri

Sonuç olarak reklam analizleri ve kullanıcı metrikleri sayesinde, hesaptaki ani veya uzun süreli değişimler takip edilebilir, hangi değişiklikler yapıldıktan sonra düşüş veya yükseliş olduğu tespit edilebilir, gerekli müdahaleler yapılabilir.

### 4.2.3. Oyun İçi ve Çevrim İçi Yardım Ekleme

Oyunlarda yardım sistemi, oyuncuya destek sağlamak ve oyun deneyimini geliştirmek için kullanılır. Bu sistem; oyuncuları oyun mekaniği, arayüzü, karakterler hakkında bilgilendirmek, oyunla ilgili ipuçlarını onlara göstermek ve oyuncuların sorunlarını çözmelerine yardımcı olmak için kullanılır.

**Oyun İçi Yardım Ekleme**

Geliştirilen oyun türüne göre oyun içine pek çok yardım sistemi eklenebilir. Güncel oyunlarda kullanılan bazı oyun içi yardım sistemleri şunlardır:

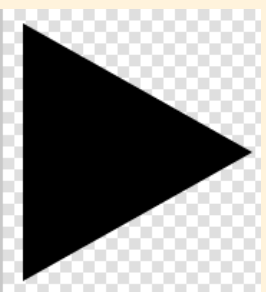
- **Öğretici (Tutorial):** Oyuna yeni başlayan oyunculara oyunun arayüzünü, temel mekaniklerini öğretmek için hazırlanan öğretici bölümdür.
- **İpucu (Hint):** Oyuncuların oyun içinde karşılaştıkları bulmacalar, nesneler veya karakterlerle ilgili bilgilendirme ve yardım sistemidir. **Bulmaca ve Point and Click** tarzı macera oyunları bu sistem üzerine kurulmuştur. Oyuncu, bu tarz oyunlarda nesnelere veya karakterlere tıklayarak çeşitli bilgiler alır ve bu bilgiler ile bulmacaları çözer. Bazı oyunlarda belirli bölümlerde, belirli bir süre sonra otomatik olarak ipuçları gösterilebilir.
- **Yardım Menüleri:** Oyuncuların oyun mekanikleri, kontroller, görevler veya karakterle ilgili bilgi edinebilecekleri menülerdir. Strateji tarzı oyunlarda bu bölüm, ansiklopedik bilgiler içerebilecek şekilde kapsamlı olabilir.
- **Yol Gösterme:** Oyunculara belirli görevlerde, hedeflerine nasıl ulaşabilecekleri ile ilgili işaretler veya yollar gösteren yardım sistemidir. Bu yardım sistemi genelde **MMORPG tarzı** büyük çaplı haritaları olan oyunlarda kullanılır. Oyuncu, işaretleri takip ederek hedefine ulaşır.
- **Kullanıcılar Arası Etkileşim:** Oyun içinde, oyuncular arasında sohbet bölümü bulunabilir. Bu sayede oyuncular takıldıkları yerleri diğer oyunculara sorabilir veya başka oyunculardan bu yerlerle ilgili yardım talep edebilir.
- **Oyun İçi Rehber:** Oyuncuların karmaşık mekanikleri, tuş kombinasyonlarını veya zor bölümleri anlamaları için hazırlanan görsel veya video içerikli yardım sistemidir.

Bu yardım sistemlerinin dışında oyunun türüne ve hedef kitlesine göre daha farklı yardım sistemleri de eklenebilir.

#### 11. UYGULAMA

Uygulama adımlarını takip ederek oyundaki bir görevi takip eden yol gösterici ekleyiniz.

**1. Adım:** Oyun motorunda yeni boş bir 2D Core uygulaması oluşturunuz.
**2. Adım:** Herhangi bir resim hazırlama programı ile arka planı transparan olan bir ok görseli hazırlayıp projenize ekleyiniz (Görsel 4.40). İsterseniz İnternetten arka planı transparan olacak şekilde hazır ok görseli de bulabilirsiniz.



**Görsel 4.40:** Ok görseli hazırlama

**3. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **2D Objects>Sprites>Square** nesnesini seçiniz ve adını **Oyuncu** olarak değiştiriniz.
**4. Adım: Oyuncu** nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna tıklayarak "hareket" isimli bir script ekleyiniz ve aşağıdaki kodları yazınız.

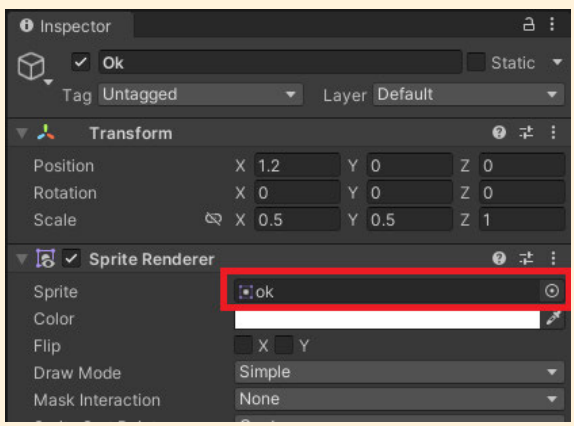
```
using UnityEngine;
public class hareket : MonoBehaviour
{
    void Update()
    {
        float x = Input.GetAxis("Horizontal");
        float y = Input.GetAxis("Vertical");
        Vector3 hareket = new Vector3(x, y);
        transform.position += hareket * 5 * Time.deltaTime;
    }
}
```

**5. Adım: Hierarchy** panelinde bulunan **Oyuncu** nesnesini seçip, sağ tıklayarak **Create Empty** seçeneği ile boş bir nesne oluşturunuz ve adını **Gosterge** olarak değiştiriniz.
**6. Adım: Gosterge** nesnesini seçiniz ve **Inspector** panelindeki **Add Component** butonuna tıklayarak "hedefgoster" isimli bir script ekleyip aşağıdaki kodları yazınız.

```
using UnityEngine;
public class hedefgoster : MonoBehaviour
{
    public Transform hedefNokta;
    public Vector3 yon;
    public float aci;
    void Update()
    {
        yon = hedefNokta.position - transform.position;
        float aci = Mathf.Atan2(yon.y, yon.x) * Mathf.Rad2Deg;
        transform.rotation = Quaternion.AngleAxis(aci, Vector3.
forward);
    }
}
```

**7. Adım: Hierarchy** panelindeki **Gosterge** nesnesini seçip, sağ tıklayarak **Create Empty** seçeneği ile boş bir nesne ekleyiniz ve adını **Ok** olarak değiştiriniz.

**8. Adım:** Ok nesnesini seçip, **Inspector** panelinde bulunan **Add Component** butonuna basarak **Sprite Renderer** bileşenini ekleyiniz. Ardından **Sprite Renderer** bileşeninin **Sprite** özelliğini 2. adımda oluşturduğunuz **ok nesnesi** olarak ayarlayınız (Görsel 4.41).



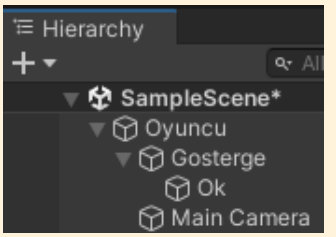

**Görsel 4.41:** Sprite özelliğini ayarlama

**9. Adım:** Ok nesnesinin boyutu ve pozisyonu ile oynayarak Görsel 4.42'deki gibi bir görüntü elde ediniz.

**10. Adım: Hierarchy** panelinde bulunan **Main Camera** nesnesini **Oyuncu** nesnesinin alt elemanı yapınız. Sonuçta Görsel 4.43'teki gibi hiyerarşik bir yapı elde etmelisiniz.



**Görsel 4.42:** Ok nesnesinin boyutunu ve pozisyonunu ayarlama



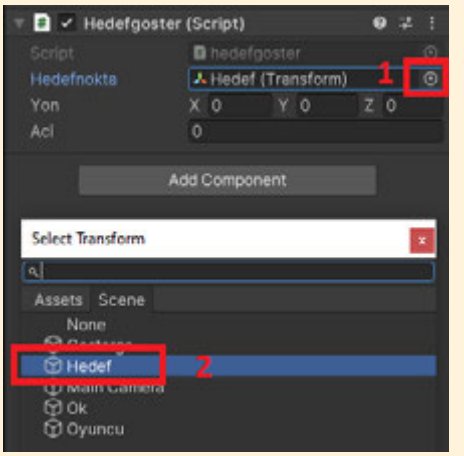

**Görsel 4.43:** Main Camera nesnesini ayarlama

**11. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **2D Object>Sprites>Circle** nesnesini seçiniz ve adını **Hedef** olarak değiştiriniz. Bu nesneyi sahnede istediğiniz bir yere taşıyınız (Görsel 4.44).



**Görsel 4.44:** Hedef nesnesinin konumunu ayarlama

**12. Adım: Hierarchy** panelinde bulunan **Gosterge** nesnesini seçip, **Inspector** panelindeki **Hedefgoster scriptinin Hedefnokta değişkenini Hedef nesnesi olarak** ayarlayınız (Görsel 4.45).




**Görsel 4.45:** Hedef nokta değişkenini ayarlama

**13. Adım:** Uygulamayı çalıştırıp oyuncu nesnesini yön tuşlarıyla hareket ettirdiğiniz zaman ok, hedefe doğru dönecektir.

### Çevrim İçi Yardım Ekleme

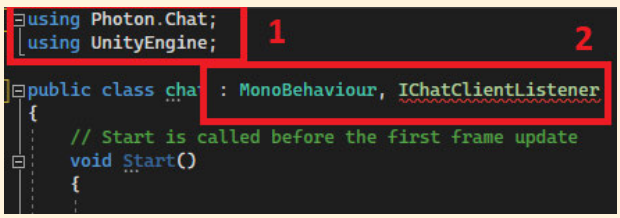
**Çevrim içi yardım;** genellikle online oyunlarda kullanılan, oyuncuların oyunun teknik ekibinden oyunla, hesap ayarlarıyla veya oyun için satın almalarla ilgili yaşadığı sıkıntılarla ilgili yardım talep ettikleri sistemdir.

Teknik destek kısmında otomatik cevaplar için bot, daha karmaşık sorunlarda ise bu iş için görevlendirilen insanlar kullanılabilir. Kullanıcılar teknik ekiple canlı görüşüp anında cevap alabilecekleri gibi bilet (ticket) sistemi ile soru-cevap şeklinde de teknik ekipten yardım alabilirler. Daha önce sorulan soruların ve sorulara alınan yanıtların görüldüğü bir arayüz olması da kullanım açısından yararlı olur. Ayrıca kullanıcıların kendi sordukları sorulara hızlı bir şekilde yanıt alması da oyun deneyimine katkı sağlar.

## 12. UYGULAMA

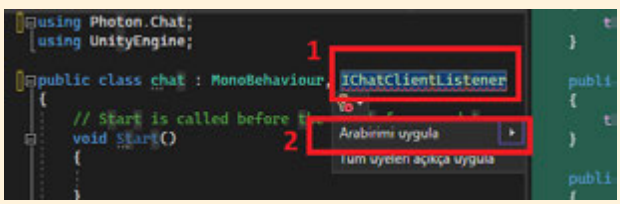
Uygulama adımlarını takip ederek oyun içinde sohbet ekranı oluşturunuz.

**1. Adım:** Oyun motorunda yeni boş bir 2D Core uygulaması oluşturunuz.
**2. Adım: Asset Store** sayfasından **Photon Pun 2 Free** eklentisini indirip eklentiyi projenize ekleyiniz.
**3. Adım:** Eklentinin web sitesine gidip üyeliğinizle siteye giriniz. Ardından **Dashboard** sayfasına giriniz ve **Create A New App** butonuna basınız.
**4. Adım: Select Photon SDK** kısmını **Chat** olarak seçip, **Application Name** kısmına da **Destek** yazarak **Create** butonuna basınız. Oluşturulan **App ID** değerini kopyalayınız.
**5. Adım:** Oyun motorunda **Window>Photon Networking** menüsünden **PUN Wizard** seçeneğini seçiniz. Açılan pencerede **Setup Project** seçeneğini seçip, bir önceki adımda kopyalanan App ID değerini yapıştırıp **Setup Project** butonuna tıklayınız.
**6. Adım: Assets** panelinden "Photon>Photon Unity Networking>Resources" klasörüne girerek **PhotonServiceSettings.assets** dosyasına tıklayınız. **Inspector** panelinde bulunan **App ID Pun** özelliğindeki değeri kopyalayıp **App ID Chat** seçeneğine yapıştırınız.
**7. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp, **Create Empty** seçeneğini seçerek yeni boş bir nesne oluşturunuz ve adını ChatYoneticisi olarak değiştiriniz. Ardından **Inspector** panelinde bulunan **Add Component** butonuna basarak "chat" isimli bir script ekleyip Görsel 4.46'daki kodları yazınız.




**Görsel 4.46:** Chat scriptine kod ekleme

**8. Adım:** Bu aşamada **IChatClientListener** hata verecek. Üzerine tıklayarak Alt+Enter tuşlarına basınız ve **Arabirimi uygula** seçeneğini seçiniz. Bu sayede **Photon Chat** için gereken bağlantı ve mesajlaşma fonksiyonları otomatik olarak oluşacaktır (Görsel 4.47).




**Görsel 4.47:** IChatClientListener arayüzünü ayarlama

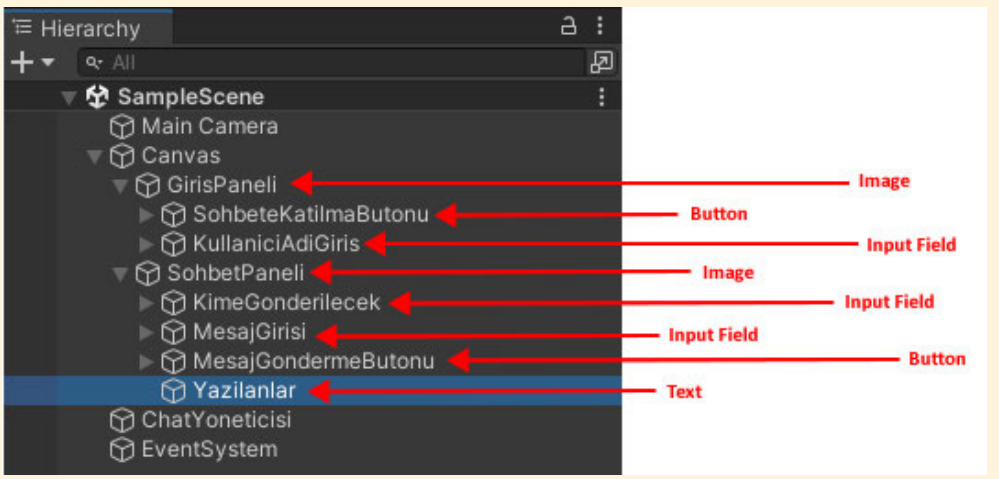
**9. Adım:** chat scriptine aşağıdaki kodları yazınız.

```
using ExitGames.Client.Photon;
using Photon.Chat;
using Photon.Pun;
using UnityEngine;
using UnityEngine.UI;
public class chat : MonoBehaviour, IChatClientListener
{
    ChatClient chatClient;
    string alici = "";
    string yazilanmesaj;
    public GameObject sohbetpaneli;
    public GameObject sohbetekatilmabutonu;
    public string kullaniciadi;
    public InputField mesajgirisi;
    public Text sohbetgoruntuleme;
    bool baglimi;
    public void OnChatStateChange(ChatState durum) //Bağlantı
    //durumu değiştiğinde
    {
        if (durum == ChatState.Uninitialized)
        {
            baglimi= false;
            sohbetekatilmabutonu.SetActive(true);
            sohbetpaneli.SetActive(false);
        }
    }
    public void OnConnected() //Sunucuya bağlandığında
    {
        sohbetekatilmabutonu.SetActive(false);
        chatClient.Subscribe(new string[] { "HerkeseAcikKanal"
}); //Sohbet kanalına katılma isteği
    }
    public void OnDisconnected() //Sunucu ile bağlantı
    //kesildiğinde
    {
        baglimi = false;
        sohbetekatilmabutonu.SetActive(true);
        sohbetpaneli.SetActive(false);
    }
    //Genel mesaj geldiğinde sohbette görüntüle
    public void OnGetMessages(string kanalismi, string[]
gonderilecekler, object[] mesajlar)
    {
        string mesaj = "";
        for (int i = 0; i <gonderilecekler.Length; i++)
        {
            mesaj = string.Format("{0}: {1}", gonderilecekler[i],
mesajlar[i]);
            sohbetgoruntuleme.text += "\n" + mesaj;
        }
    }
    //Özel mesaj geldiğinde sohbette görüntüle
    public void OnPrivateMessage(string gonderen, object
```

```
mesajlar, string kanalismi)
    {
        string mesaj = "";
        mesaj = string.Format("(Private) {0}: {1}", gonderen,
mesajlar);
        sohbetgoruntuleme.text += "\n " + mesaj;
    }
    //Bir kanala katıldığında sohbet panelini aktif et
    public void OnSubscribed(string[] channels, bool[] results)
    {   sohbetpaneli.SetActive(true);    }
    public void KullaniciAdiDegistiginde(string isim)
    {   kullaniciadi = isim;    }
    public void SohbeteBaglan()
    {
        baglimi = true;
        chatClient = new ChatClient(this);
        chatClient.Connect(PhotonNetwork.PhotonServerSettings.
AppSettings.AppIdChat, PhotonNetwork.AppVersion, new Photon.Chat.
AuthenticationValues(kullaniciadi));
    }
    void Update()
    {
        if (baglimi)
        {
            chatClient.Service();     //Sunucuya bağlı olduğu
            //sürece bu fonksiyon sürekli çalışmalıdır.
        }
        if (mesajgirisi.text != "" && Input.GetKey(KeyCode.
Return))
        {
            Sohbet();
        }
    }
    public void YazilanMesajTextDegistiginde(string yazilan)
    {     yazilanmesaj = yazilan;    }
    public void AliciTextiDegistiginde(string yazilan)
    {     alici = yazilan;      }
    public void Sohbet()
    {
        if (alici == "")
        {
            chatClient.PublishMessage("HerkeseAcikKanal",
yazilanmesaj);     //Genel mesaj gönder
        }
        else
        {
            chatClient.SendPrivateMessage(alici, yazilanmesaj);
//Özel mesaj gönder
        }
        mesajgirisi.text = "";
        yazilanmesaj = "";
    }
    public void DebugReturn(DebugLevel level, string message)
    {
        //throw new System.NotImplementedException();
```

```
    }
    public void OnStatusUpdate(string user, int status, bool
gotMessage, object message)
    {
        throw new System.NotImplementedException();
    }
    public void OnUnsubscribed(string[] channels)
    {
        throw new System.NotImplementedException();
    }
    public void OnUserSubscribed(string channel, string user)
    {
        throw new System.NotImplementedException();
    }
    public void OnUserUnsubscribed(string channel, string user)
    {
        throw new System.NotImplementedException();
    }
}
```

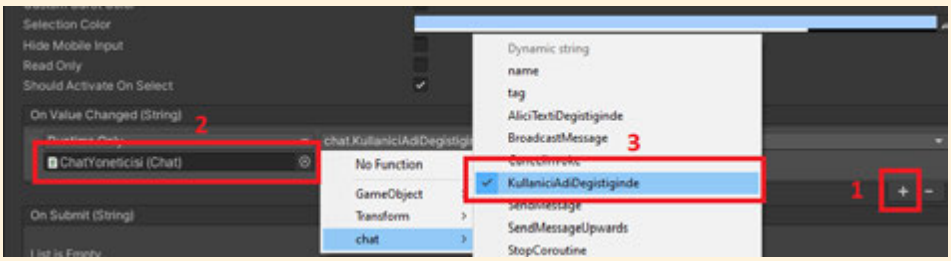
**10. Adım: Hierarchy** panelinde boş bir yere sağ tıklayıp **Canvas** nesnesi ekleyiniz.
**11. Adım: Canvas nesnesinin altına** Görsel 4.48'deki hiyerarşik yapıyı oluşturunuz.




**Görsel 4.48:** Arayüz hazırlama

**12. Adım: Hierarchy** panelinde bulunan **GirisPaneli ve SohbetPaneli** nesnelerini seçip **Inspector** panelindeki **Anchors Min X** ve **Anchors Min Y** değerlerini **0**, **Anchors Max X** ve **Anchors Max Y** değerlerini **1** yapınız. **Left, Top, Right, Bottom** değerlerini ise **30** yapınız.
**13. Adım: Hierarchy** panelinde bulunan SohbeteKatilmaButonu nesnesini seçip **Inspector** panelindeki **Anchors Min X** değerini **0.4** ve **Anchors Min Y** değerini **0.3**, **Anchors Max X** değerini **0.6** ve **Anchors Max Y** değerini **0.6** yapınız. **Left, Top, Right, Bottom** değerlerini ise **0** yapınız. **OnClick** özelliğinin altında bulunan "+" simgesine basarak fonksiyon ekleyiniz. Nesne olarak **ChatYoneticis**i nesnesini, fonksiyon olarak da **SohbeteBaglan ()** seçiniz.
**14. Adım: Hierarchy** panelinde bulunan **KullaniciAdiGiris** nesnesini seçip **Inspector** panelindeki **Anchors Min X** değerini **0.35 ve Anchors Min Y** değerini **0.5**, **Anchors Max X** değerini **0.65** ve **Anchors Max Y** değerini **0.6** yapınız. **Left, Top, Right, Bottom** değerlerini ise **0** yapınız.
**15. Adım: Hierarchy** panelinde bulunan **KullaniciAdiGiris** nesnesini seçip, **Inspector** panelinde bulunan **On Value Changed** özelliğinin altındaki "+" simgesine basarak fonksiyon ekleyiniz.

Nesne olarak **ChatYoneticisi** nesnesini, fonksiyon olarak da chat scriptinde bulunan **Dynamic String** alanındaki **KullaniciAdiDegistiginde fonksiyonunu** seçiniz (Görsel 4.49).




**Görsel 4.49:** KullaniciAdiDegistinde fonksiyonunu ekleme

**16. Adım: Hierarchy** panelinde bulunan **KimeGonderilecek** nesnesini seçip **Inspector** panelindeki **Anchors Min X** değerini **0** ve **Anchors Min Y** değerini **0.1**, **Anchors Max X** değerini **1** ve **Anchors Max Y** değerini **0.2** yapınız. **Left, Top, Right, Bottom** değerlerini ise **0** yapınız. **On Value Changed** özelliğinin altındaki "+" simgesine basarak fonksiyon ekleyiniz. Nesne olarak **ChatYoneticisi** nesnesini, fonksiyon olarak da chat scriptinde bulunan **Dynamic String** alanındaki **AliciTextiDegistiginde** fonksiyonunu ekleyiniz.

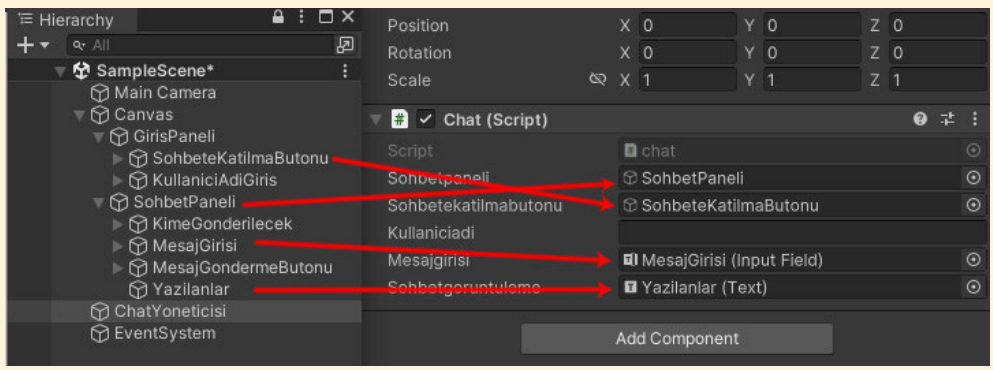
**17. Adım: Hierarchy** panelinde bulunan **MesajGirisi** nesnesini seçip **Inspector** panelindeki **Anchors Min X** değerini **0** ve **Anchors Min Y** değerini **0**, **Anchors Max X** değerini **0.8** ve **Anchors Max Y** değerini **0.1** yapınız. **Left, Top, Right, Bottom** değerlerini ise **0** yapınız. **On Value Changed** özelliğinin altındaki "+" simgesine basarak fonksiyon ekleyiniz. Nesne olarak **ChatYoneticisi** nesnesini, fonksiyon olarak da chat scriptinde bulunan **Dynamic String** alanındaki **YazilanMesajTextDegistiginde** fonksiyonunu ekleyiniz.

**18. Adım: Hierarchy** panelinde bulunan **MesajGondermeButonu** nesnesini seçip **Inspector** panelindeki **Anchors Min X** değerini **0.8** ve **Anchors Min Y** değerini **0**, **Anchors Max X** değerini **1** ve **Anchors Max Y** değerini **0.1** yapınız. **Left, Top, Right, Bottom** değerlerini ise **0** yapınız. **On Click** özelliğinin altındaki "+" simgesine basarak fonksiyon ekleyiniz. Nesne olarak **ChatYoneticisi** nesnesini, fonksiyon olarak da **Sohbet** fonksiyonunu ekleyiniz.

**19. Adım: Hierarchy** panelinde bulunan **Yazilanlar** nesnesini seçip **Inspector** panelindeki **Anchors Min X** değerini **0** ve **Anchors Min Y** değerini **0.2**, **Anchors Max X** değerini **1** ve **Anchors Max Y** değerini **1** yapınız. **Left, Top, Right, Bottom** değerlerini ise **0** yapınız. **Text** özelliğinde bulunan yazıları siliniz.

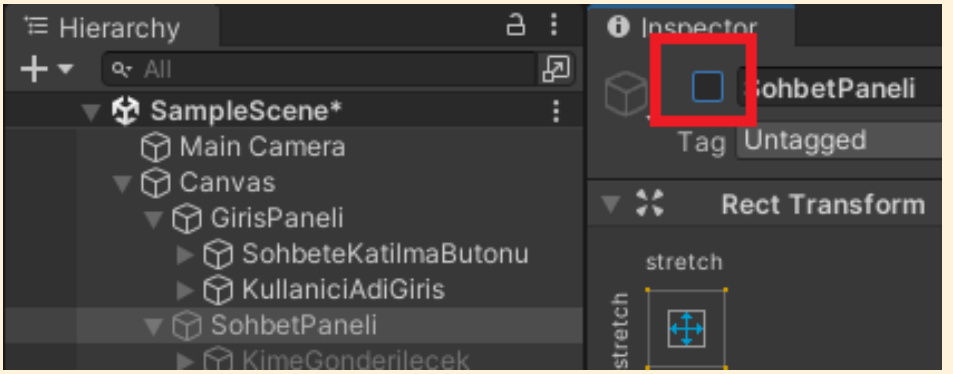
**20. Adım: Hierarchy** panelinde bulunan **MesajGirisi** nesnesini seçip **Inspector** panelindeki **Anchors Min X** değerini **0** ve **Anchors Min Y** değerini **0**, **Anchors Max X** değerini **0.8** ve **Anchors Max Y** değerini **0.1** yapınız. **Left, Top, Right, Bottom** değerlerini ise **0** yapınız.

**21. Adım: Hierarchy** panelinde bulunan **ChatYoneticisi** nesnesini seçip değişkenleri Görsel 4.50'deki gibi ayarlayınız.




**Görsel 4.50:** ChatYoneticisi nesnesinin değişkenlerini ayarlama

**22. Adım: Hierarchy** panelinde bulunan **SohbetPaneli** nesnesini pasif hâle getiriniz (Görsel 4.51).

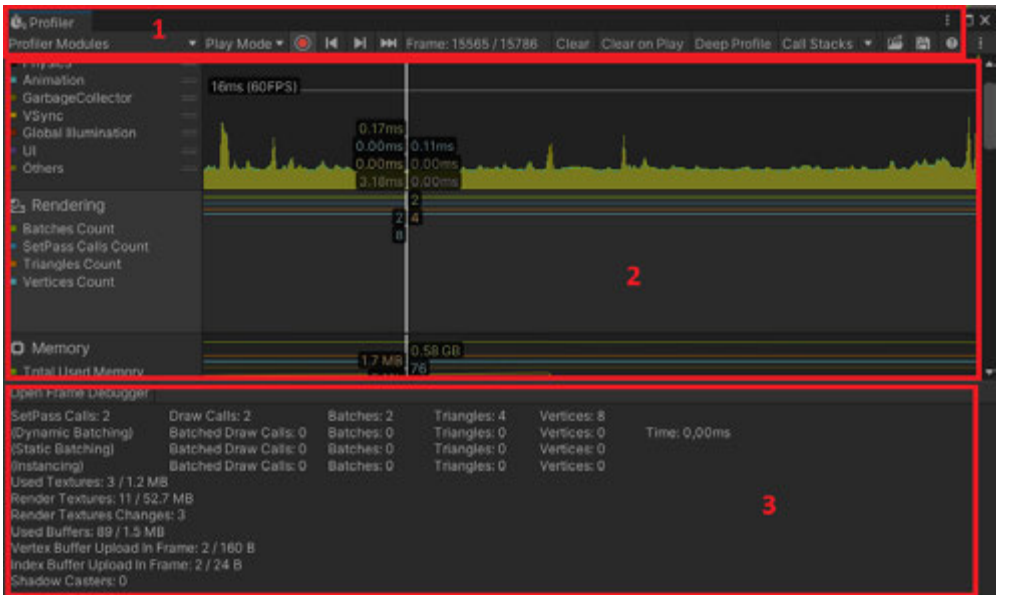



**Görsel 4.51:** SohbetPaneli nesnesini pasif hâle getirme

**23. Adım: File>Build Settings** penceresinden **Build And Run** seçeneğini seçerek oyunu çalıştırınız. Oyun motorunda da oyunu çalıştırıp ikisi arasında sohbet edebilirsiniz. Uygulamayı birden fazla bilgisayarda çalıştırıp farklı isimlerle deneyebilirsiniz. **Kime kısmı** boş bırakılırsa yazılanlar herkese, **isim yazılırsa** sadece o kullanıcıya gönderilecektir.

### 4.2.4. Oyunun Bellek, Grafik ve Oynanış Olarak İyileştirilmesi

Oyun geliştirme sürecinde bellek, grafik ve oynanışın iyileştirilmesi, oyun kalitesini artırmak için önemli adımlardandır. Bu işlemlere **optimizasyon** adı verilir. Optimizasyon işleminin genelgeçer bir kuralı yoktur. Amaç; hangi bileşenlerin sistemi daha çok yorduğunu tespit edip, bunları mümkün olduğunca düzeltmektir. Oyun motorunda bu iş için **Profiler** aracından yararlanılır (Görsel 4.52).




**Görsel 4.52:** Profiler aracı

Bu araca **Window>Analysis** menüsü ile ulaşılabilir. Oyuna yeni özellikler eklendikçe bu araç çalıştırılır ve sistem kullanımında ne gibi değişiklikler olduğu takip edilir.

Profiler aracı 3 kısımdan oluşmaktadır.

- **Menü Barı:** Profiler aracını kontrol etmeye yarayan butonları barındırır. Oyunu test ederken veri akışı yoksa bu kısımda bulunan kayıt butonu aktif edilmelidir.
- **Modüller:** Toplanan veriler işlemci, bellek, grafik, ses, ağ gibi farklı kategoriler altında grafik olarak gösterilir.
- **Modül Detayları:** Seçili frame hakkında bilgi gösterilen kısımdır. Ani değişim görülen frameler seçilerek sıkıntının nerede olduğu buradan görülebilir.

Profiler aracı, hem oyun editörde denenirken hem de derleme alınıp cihazda test edilirken kullanılabilir. Geliştirme yapılan cihazla, test yapılan cihazın donanımları farklı olacağı için oyunu editörde denerken farklı, cihazda test ederken farklı veriler gelebilir.

**Bellek İyileştirmeleri**

Belleğin düzgün kullanılması oyunun performanslı çalışması için önemlidir. Gereksiz bellek kullanımını bu yüzden en aza indirgemek gerekir.

Bellek optimizasyonu ile ilgili en önemli kavram "Garbage Collector" terimidir. **Garbage Collector** hafıza yönetiminden sorumlu sistem bileşenidir. Sahnede yeni bir nesne oluşturulacağı zaman Garbage Collector çalışıp kullanılan bellek alanını tarar ve gereksiz olan kısımları bellekten boşaltır. Bu alan, oluşturulacak nesne için yeterliyse nesne oluşturulur. Alan yeterli gelmezse Garbage Collector sistemden yeni bellek alanı talep eder. Bu işlem biraz zaman alır ve o esnada oyun donar. Ram az kullanıldığı zaman donma süresi çok kısadır ve kullanıcı bu donmayı fark etmez ama ram çok sık kullanılmaya başlarsa oyunda takılmalar başlar. Kullanılan bellek miktarı arttıkça Garbage Collector bileşeni daha fazla çalışmaya başlar. Oyun, sistem belleğini tamamen kullanıyorsa Garbage Collector daha fazla alan isteyemez ve oyun çöker. Bu da istenmeyen durumların başında gelmektedir. Bu yüzden bellek optimizasyonu oldukça önemlidir.

Bellek kullanımını azaltmak için şunlar önerilir:

- **Assets:** Yüksek boyutlu bileşenler gereksiz bellek harcar. Kullanılan ses dosyalarının, karakter dokularının, kaplamaların boyutu düşürülüp, oyun test edildiğinde kalite olarak çok fark etmiyorsa o şekilde kullanmak bellek kullanımı açısından çok büyük fark yaratır.
- **Uzaktaki Nesneler:** Uzakta bulunan ve oyuncunun hiçbir zaman yanına gitmeyeceği nesneler daha düşük poligonlu modeller ve kaplamalar ile değiştirilebilir.
- **Büyük Haritalar:** Büyük haritaları tek seferde yüklemek yerine parçalara bölüp, sadece görünen kısmı yüklemek ve oyuncu başka bir bölgeye yaklaştığında o bölgeyi bellekte hazırda tutmak bellek yönetimi için doğru bir adımdır.
- **Ses Dosyaları:** Bir ses dosyasının sesi tamamen kısılsa bile AudioSource bileşeni müziği sessiz bir şekilde çalmaya devam eder. Bu yüzden kullanılmayan sesler **Stop fonksiyonu** ile durdurulmalıdır.
- **Arayüz:** Tüm arayüz objeleri tek bir Canvas bileşeninin altında durmamalıdır çünkü Canvas bileşeninin altında bulunan tek bir UI elemanının herhangi bir özelliği değişse bile diğer tüm elemanlar yenilenmek zorunda kalır.
- **Coroutine:** Bir Coroutine başlatıldığında normal program akışının dışına çıkıldığı için fazladan bellek kullanılır. Bu yüzden gereksiz Coroutine kullanımından kaçınmak gerekir.
- **Döngüler:** Foreach döngüsü Garbage Collector bileşenine yük bindirir. Bu yüzden çok gerekmiyorsa foreach yerine for döngüsü kullanmak, bellek kullanımı açısından daha iyidir.
- **Değişken Tanımlama:** Tanımlama işlemlerini **Update veya FixedUpdate fonksiyonlarının içinde değil de** programın başında yapmak Garbage Collector için iyidir.

**Optimizasyondan önce:**

```
public class deneme : MonoBehaviour
{
   public Transform hedef;
   void Update()
   {
      GetComponent<UnityEngine.AI.NavMeshAgent>().destination = hedef.position;
   }
}
```

**Optimizasyondan sonra:**

```
public class deneme : MonoBehaviour
{
   public Transform hedef;
   private NavMeshAgent agent;
   void Update()
   {
      agent.destination = hedef.position;
   }
}
```

- **Gereksiz Yere Çağrılan Fonksiyonlar:** Update fonksiyonu içinde her Frame'de çağrılan bir fonksiyonun 3-4 Frame'de bir çalışması yetiyorsa buna uygun kodlama yapılabilir.

**Optimizasyondan Önce:**

```
void Update()
{
   Fonksiyon();
}
```

**Optimizasyondan Sonra:**

```
void Update()
{
   if (Time.frameCount % 4 == 0)
   {
      Fonksiyon();
   }
}
```

- **Debug.Log Komutu:** Oyun geliştirilirken bilgi amaçlı kullanılan "Debug.Log" komutları, oyun derlendiğinde de çalışmaya devam eder. Bu komutlar derleme almadan önce silinmeli ya da sadece editörde görünecek şekilde değiştirilmelidir.

```
void Update()
{
   #if UNITY_EDITOR
      Debug.Log("Sadece editörde gözükecek");
   #endif
}
```

- **Nesne Havuzu (Object Pooling):** Bir nesnenin işi bittiğinde o nesneyi yok etmek yerine bir havuza atıp, ihtiyaç duyulduğunda yeniden oluşturmak yerine yine bu havuzdan alma yöntemine **Object Pooling** denir. Bu yöntem, sonsuz koşu tarzı oyunlarda veya karakterin sürekli ateş ettiği oyunlarda kullanılabilir ve bellek optimizasyonu açısından son derece etkilidir.

**Grafik İyileştirmeleri**

Oyunlarda **grafik optimizasyonu**, oyunun daha düşük donanım gereksinimleri olan sistemlerde bile akıcı bir şekilde çalışmasını sağlamak için grafiksel detayları ve performansı dengelemeyi amaçlayan bir süreçtir. Bu işlem için de genel bir kural yoktur. Burada amaç, ekranda mümkün olduğunca az ve kaliteli bileşen göstermektir.

Bazı grafik optimizasyonu yöntemleri şunlardır:

- **Doku ve Kaplamalar:** Kullanılan dokular ve kaplamalar, kalite olarak fark etmeyecek en düşük seviyeye ayarlanmalıdır. Böylece yüksek performanslı sistemlerde bile oyunun daha iyi görünmesi sağlanırken düşük donanım seviyelerine sahip oyuncuların da oyunu keyifle oynaması mümkün olur.
- **Grafik Ayarları:** Oyun içinde grafik bileşenleri ile ilgili ayarlar sunularak oyunculara performans dengesini tutturmak için fırsat verilebilir.
- **LOD (Level of Detail):** Uzaktaki nesneleri daha basit, yakındaki nesneleri daha detaylı olarak çizen sisteme **LOD** denir. Bu ayar kullanılarak grafik performansı artırılabilir.
- **Culling Teknikleri:** Kamerada görünmeyen nesnelerin çizilmesini engelleyen **Frustum Culling** ile başka bir objenin arkasında kaldığı için çizilmesine gerek olmayan nesneleri engelleyen **Occlusion Culling** tekniklerinin düzgün kullanımı grafik performansını en çok etkileyen yöntemlerden biridir.
- **Çözünürlük Ayarı:** Düşük ekran çözünürlüğü ekranda daha az nesne çizilmesi anlamına gelir. Bu da performansı artıran bir durumdur. Bu yüzden oyuna çözünürlük ayarı koymak iyi bir fikirdir.
- **Gölge Optimizasyonları:** Gölge hesaplamaları cihazı en çok yoran ayarlardan biridir. Çok küçük boyutlu nesnelerde gölgelendirmeyi kapatmak, oyun boyunca yeri değişmeyecek nesnelerde statik gölgelendirme kullanmak, dinamik gölgelendirme kullanılacaksa çok fark olmadığı sürece **Soft Shadows** yerine **Hard Shadows** kullanmak performansı oldukça artıracaktır.
- **Partikül Efekti Optimizasyonu:** Partikül efekti düzgün kullanılmadığında sistemi en çok yoran bileşenlerdendir. Patlamalar, sis efekti ve diğer görsel efektler oyunun etkileyici görünmesini sağlasa da performans üzerinde çok büyük etkisi vardır. Bu yüzden partikül efekti dikkatli kullanılmalıdır.
- **Animasyonlar:** Basit animasyonlar için **Animator** bileşeni yerine **DOTween** kullanmak performansı artırabilir.
- **Kamera Ayarı:** Kamera nesnesinde bulunan Far Clipping Plane değeri ekrana çizilen nesnenin kameradan ne kadar uzakta olacağını belirler. Bu değerden daha ileride olan nesneler ekrana çizilmez. Bu değeri olabildiğince azaltmak performans açısından önemlidir.
- **3D Modeller:** Oyunda kullanılan 3D modellerde, oyuncunun hiç göremeyeceği yüzeyler varsa o kısımları silmek performansı artırabilir.

**Oynanış İyileştirmeleri**

Oynanış iyileştirmeleri daha iyi bir oyun deneyimi için önemlidir. Oynanış iyileştirmeleri alfa-beta aşamalarında yapıldığı gibi oyun piyasaya çıktıktan sonra kullanıcı geri bildirimleri ile de yapılabilir.

Oynanış iyileştirmeleri ile ilgili bazı genel öneriler şunlardır:

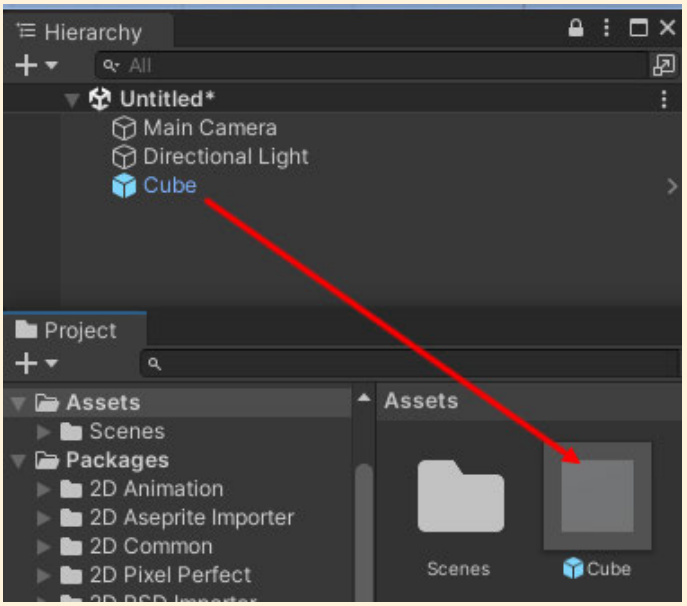
- **Zorluk Seviyeleri:** Oyunlara zorluk ayarı koyarak farklı beceri seviyesine sahip insanların farklı deneyimler yaşaması sağlanabilir.
- **Eğitim Bölümleri:** Oyuncuların oyunun temel mekaniklerini öğrenebilecekleri eğitici bölümler tasarlanabilir.

- **Düşük Gecikme Süreleri:** Online oyunlarda düşük gecikme süreleri önemlidir. Ağ optimizasyonu ve daha kaliteli sunucular ile gecikme süreleri düşürülebilir.
- **Farklı Oyun Mekanikleri:** Oyunun monotonlaşmasını önlemek için mini oyunlar, farklı bulmacalar, gizli bölümler veya farklı oyun modları oyuna eklenebilir.
- **Ödüllendirme:** Tamamlanan görevlerden ve başarımlardan sonra oyuncuları ödüllendirmek oyun deneyimi açısından önemlidir. Oyunculara ödül olarak farklı ses efektleri, animasyonlar ve görünüm çerçeveleri verilebilir.
- **Güncellemeler:** Kullanıcı geri bildirimleri ile oyunu güncel tutmak son derece önemlidir. Oyunu hatalardan arındırmak, yeni seviyeler, karakterler, eşyalar ve farklı etkinlikler eklemek oyuncuların oyuna ilgisini canlı tutabilir.

## 13. UYGULAMA

Uygulama adımlarına göre **Profiler** aracını kullanarak **Object Pooling** yöntemi kullanılan bir uygulamayı bellek kullanımı yönünden analiz ediniz.

**1. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.
**2. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **3D Objects>Cube** nesnesi ekleyiniz. Bu nesneyi **Assets** paneline sürükleyerek Prefab olarak ayarlayınız (Görsel 4.53).



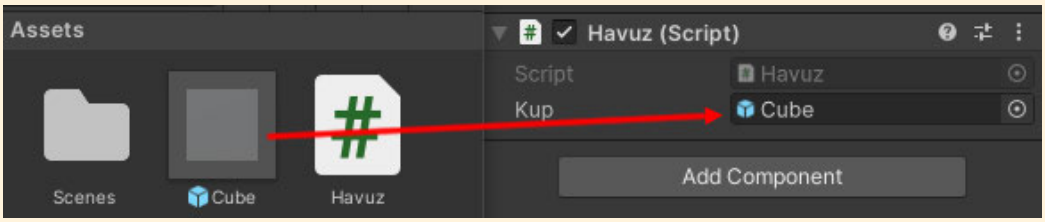

**Görsel 4.53:** Prefab oluşturma

**3. Adım: Hierarchy** panelinde boş bir yere sağ tıklayarak **Create Empty** seçeneği ile boş bir nesne oluşturunuz. Daha sonra **Inspector** panelindeki **Add Component** butonuna basarak "Havuz" adında bir script oluşturup yandaki kodları yazınız.

```
using System.Collections;
using UnityEngine;
public class Havuz : MonoBehaviour
{
    public GameObject kup;
    void Start()
    {

        StartCoroutine(SonsuzObje());
    }
    IEnumerator SonsuzObje()
    {
        while (true)
        {
            // x, y ve z değerleri -4 ile 4 arasında olan
            //rastgele bir Vector3 döndürür
            Vector3 konum = Random.onUnitSphere * 4f;
            GameObject nesne = Instantiate(kup, konum,
Quaternion.identity);
            yield return new WaitForSeconds(1f);
            Destroy(nesne);
        }
    }
}
```

**4. Adım:** Cube isimli prefab nesnesini **Havuz scriptine** ekleyiniz (Görsel 4.54).

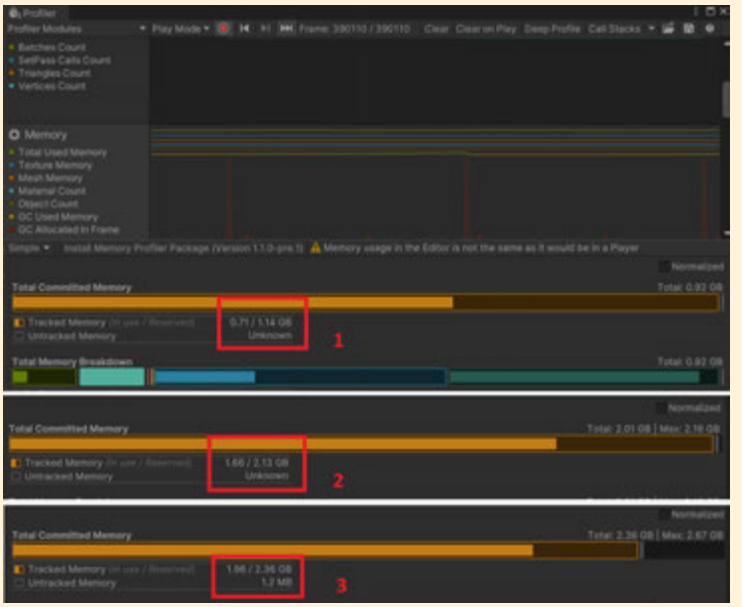



**Görsel 4.54:** Cube prefabını scripte ekleme

**5. Adım: Window>Analysis** menüsünden **Profiler** aracını açınız.
**6. Adım:** Uygulamayı çalıştırıp, **Profiler** aracının **Memory** sekmesine geçerek takip ediniz. Şu aşamada bellek kullanımı ile ilgili bir sıkıntı görmeyeceksiniz.
**7. Adım:** Havuz scriptini açarak **Start** fonksiyonuna aşağıdaki for döngüsünü ekleyiniz. Böylece aynı anda 5000 küp nesnesi oluşturulup 1 sn. sonra yok edilecektir.

```
void Start()
 {
         for (int i = 0;i<5000;i++)
         StartCoroutine(SonsuzObje());
 }
```

**8. Adım:** Uygulamayı çalıştırıp **Profiler** aracında bellek kullanımını takip ediniz. Uygulamayı 3-4 dk. boyunca çalıştırınca kullanılan alanın gitgide arttığını, alan doldukça da oyunun sistemden daha fazla alan talep ettiğini göreceksiniz (Görsel 4.55).



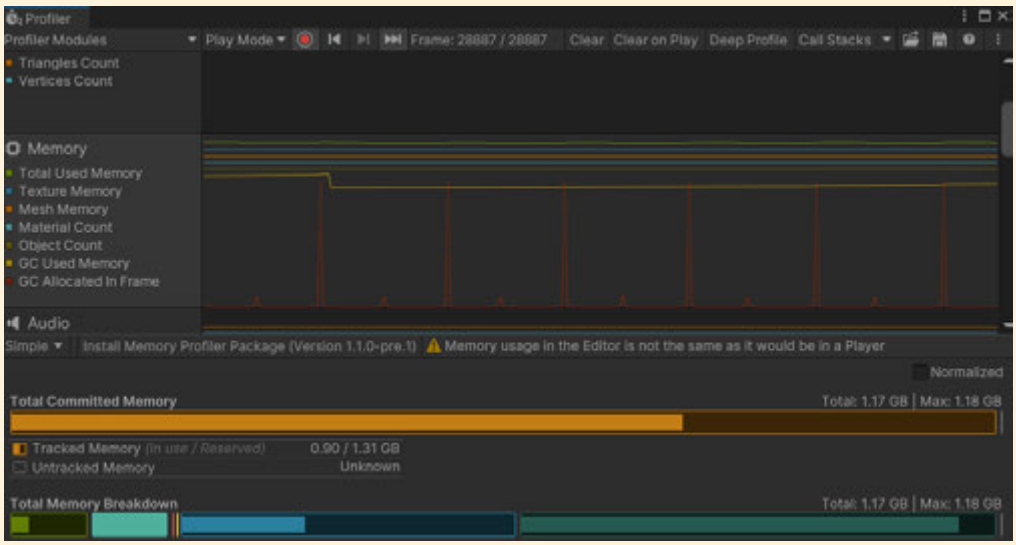

**Görsel 4.55:** Optimizasyondan önce bellek kullanımı

**9. Adım:** Havuz scriptini aşağıdaki gibi değiştiriniz. Bu script sayesinde pasif hâle gelen nesneler bir listede tutulup, gerektiğinde yeniden kullanılabilir.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Havuz : MonoBehaviour
{
    public GameObject kup;
    private List<GameObject> nesneHavuzu = new
List<GameObject>();
    void Start()
    {
        for (int i = 0; i < 5000; i++)
        StartCoroutine(SonsuzObje());
    }
    IEnumerator SonsuzObje()
    {
        while (true)
        {
            Vector3 konum = Random.onUnitSphere * 4f;
            GameObject nesne = HavuzdanAl();
            nesne.transform.position = konum;
            yield return new WaitForSeconds(1f);
            HavuzaGonder(nesne);
        }
    }
    GameObject HavuzdanAl()
    {
        if (nesneHavuzu.Count > 0)
        {
```

```
            int sonIndex = nesneHavuzu.Count - 1;
            GameObject nesne = nesneHavuzu[sonIndex];
            nesne.gameObject.SetActive(true);
            nesneHavuzu.RemoveAt(sonIndex);
            return nesne;
        }
        // Havuz boşsa mecbur yeni bir kup oluştur ve gönder
        return Instantiate(kup);
    }
    void HavuzaGonder(GameObject nesne)
    {
        nesne.gameObject.SetActive(false);
        nesneHavuzu.Add(nesne);
    }
}
```

**10. Adım:** Uygulamayı çalıştırıp **Profiler** aracında bellek kullanımını takip ediniz. Uygulama ne kadar çalışırsa çalışsın bellek kullanımı belirli bir seviyede kalacaktır (Görsel 4.56).




**Görsel 4.56:** Optimizasyon sonrası bellek kullanımı

SIRA SİZDE

Daha önce yaptığınız oyunlardan birisini Profiler aracını kullanarak bellek ve grafik yönünden optimize ediniz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Profiler aracını kullanarak oyunu bellek yönünden optimize etti. | | |
| 2. Profiler aracını kullanarak oyunu grafik yönünden optimize etti. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 4.3. DERLEME VE YAYINLAMA İŞLEMLERİ

Oyun hangi platform için yayımlanacaksa o platformda çalıştırılabilecek hâle getirilme işlemine **derleme**, oyunun kullanıcıların beğenisine sunulma işlemine de **yayınlama** denir. Oyun motoru ile pek çok platforma oyun geliştirip yayımlanabilir.

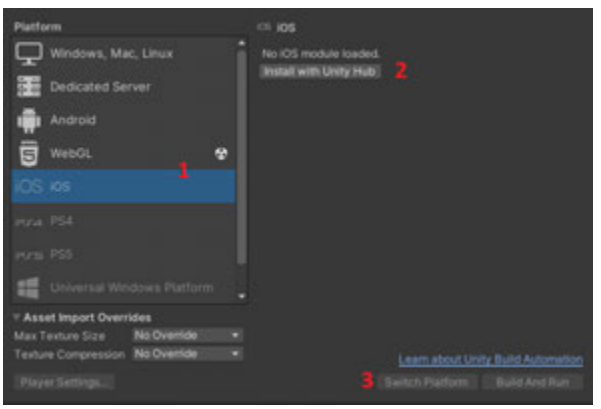
### 4.3.1. Derlenebilen Platform Türleri

Oyun motorunda basit birkaç adımla farklı platformlar için derleme yapılabilir. Desteklenen bazı platformlar şunlardır:

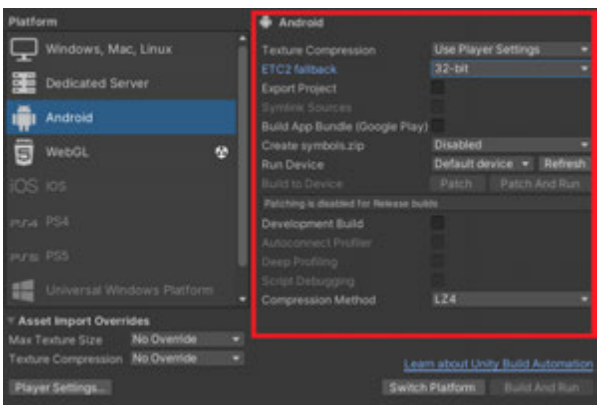
- **PC:** Oyun motoru ile Windows masaüstü bilgisayarları, tabletleri ve macOS platformu için tam destek sunulur ama Linux dağıtımlarının hepsi desteklenmez.
- **Mobil Cihazlar:** Oyun motorunda iOS ve Android telefonlar ve tabletler için oyun geliştirmeyi ve derlemeyi destekler.
- **Konsollar:** Oyun motoru ile Playstation, Xbox, Nintendo Switch, Google Stadia gibi oyun konsolları için de oyun geliştirilebilir. Bu konsollara oyun geliştirmek için oyun motorunun ücretli sürümü gereklidir.
- **Web Tarayıcıları:** Oyun motoru, WebGL teknolojisi ile tarayıcı tabanlı oyunlar oluşturmanıza ve derlemenize de izin verir. WebGL teknolojisi web tarayıcılarında grafik çizmeyi sağlayan bir web standardıdır. Ancak tüm tarayıcıların WebGL desteği yoktur.
- **Sanal Gerçeklik (VR), Artırılmış Gerçeklik (AR) ve Karma Gerçeklik (MR) Cihazları:** Oyun motoru; HTC Vive, Oculus Rift, ARKit, Playstation VR, Microsoft HoloLens, Meta Quest3, Oculus Quest 2, Apple Vision Pro gibi cihazlar için oyun ve uygulama geliştirmeyi destekler.
- **Smart TV'ler:** Oyun motoru, bazı Smart TV'ler için de oyun ve uygulama geliştirmeyi destekler.

Oyun motorunun platform desteği modüler yapıdadır. Bu sebeple, farklı platformlara oyun geliştirilecekse **Hub programı kullanılarak** gereken dosyaların indirilmesi gerekir. **File>Build Settings** menüsü ile açılan pencerede platform seçimi ve her platform için farklı derleme seçenekleri görüntülenir. Derleme işlemi yapılmak istenilen platform seçilip, **Install with Hub** butonuna basılarak gereken dosyalar indirilebilir. **Switch Platform** butonu ile de istenilen platform seçilir (Görsel 4.57).

Kullanılmak istenen platform daha önceden kurulmuş ise **Install with Hub** butonu yerine seçilen platformun derleme ayarları görüntülenecektir. Bu ayarlar her platform için farklıdır (Görsel 4.58).

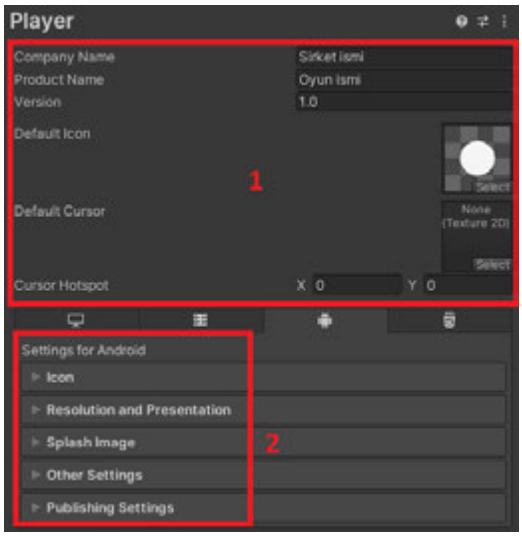



**Görsel 4.57:** Platform seçimi




**Görsel 4.58:** Derleme ayarları

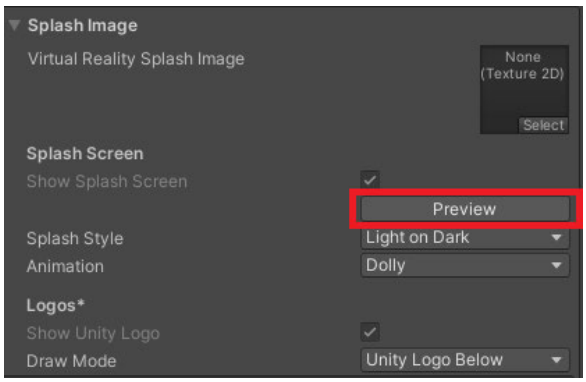
**Build Settings** penceresinde bulunan **Player Settings** butonuna basılarak daha ayrıntılı ayarların olduğu ekrana ulaşılabilir. Bu ekranda tüm platformlar için ortak ayarlar bulunduğu gibi seçilen platforma göre farklı ayarlar da bulunmaktadır (Görsel 4.59).




**Görsel 4.59:** Uygulama ayarları

İlk kısım tüm platformlar için aynıdır. Bu kısımda **şirket ismi, oyun ismi, sürüm numarası ve oyunun ikon dosyası ile ilgili ayarlar** yapılır. İkinci kısımdaki ayarlar ise şöyledir:

- **Icon:** Seçilen ikonun farklı çözünürlüklerde nasıl gösterileceği ile ilgili ayarlar buradan yapılır.
- **Resolution and Presentation:** Oyunun ekran çözünürlüğü, ekran oranı gibi ayarlar bu kısımdan yapılır.
- **Splash Image:** Oyun açılmadan önce firma logosunun gösterilmesi ile ilgili ayarların yapıldığı bölümdür. Logonun ekranda kaç saniye kalacağı, hangi animasyonla çalışacağı, arka plan renginin ne olacağı gibi ayarlardır. Oyun motorunun ücretsiz versiyonunda ilk önce oyun motorunun logosu gösterilir. Daha sonra buraya eklenen logolar gösterilir. Yapılan değişiklikler **Preview butonuna basılarak** görülebilir (Görsel 4.60).




**Görsel 4.60:** Firma logosu gösterimi ile ilgili ayarlar

- **Other Settings:** Grafik, derleme, oyunun nereye kurulacağı gibi geriye kalan ayarlar bu kısımda bulunur. Bu kısım platforma göre değişkenlik gösterir.
- **Publishing Settings:** Yayınlama ile ilgili ayarların bulunduğu kısımdır.

## 4.3.2. Artırılmış (AR), Sanal (VR) ve Karma (MR) Gerçeklik Türleri

Günümüzde AR, VR, MR teknolojileri güncel şekilde kullanılmakta ve sağlık, eğitim, savunma sanayisi, eğlence, mimari gibi pek çok alanda bunların kullanımları hızla artmaktadır. Bu teknolojiler her ne kadar sıklıkla kullanılsa da birbirinden farklı kavramlardır. Oyun motorunda bu teknolojilerin hepsi desteklenmektedir.

**Artırılmış Gerçeklik (AR-Augmented Reality)**

Gerçek dünyanın bilgisayar tarafından üretilen görüntü, ses gibi verilerle zenginleştirilmesine **Artırılmış Gerçeklik (AR)** denir (Görsel 4.61). AR sayesinde, gerçekte var olan nesnelerin üzerine sanal nesneler bindirilerek gerçeklik artırılmış olur. Bu işlem için artırılmış gerçeklik gözlüğü, akıllı telefon ya da tablet gibi cihazlar kullanılır. Ayrıca artırılmış gerçeklik uygulamalarından birinin de cihaza yüklenmiş olması gerekir. Bu programlar gerçek nesneyi algılayıp onun üzerine sanal nesneyi ekler.



**Görsel 4.61:** Artırılmış gerçeklik

AR teknolojisinin güncel olarak kullanıldığı alanlardan bazıları şunlardır:

- **Fotoğrafçılık:** Sıklıkla kullanılan fotoğraf filtreleri AR teknolojisi için en iyi örneklerdendir.
- **Alışveriş:** Giyim mağazalarının kıyafet deneme sistemleri, mobilya mağazalarından alınan mobilyaların evde nasıl duracağı ile ilgili uygulamalar AR teknolojisi için en kullanışlı örneklerindendir.
- **Reklam:** Bu sektörde AR teknolojisi ile tasarlanan afişler ve billboardlar kullanılmaktadır.
- **Eğitim:** AR teknolojisi ile sanal nesneler sınıf ortamına taşınabilir. Öğrenciler; gökyüzünü, denizi, gezegenleri kitaplardan, videolardan değil de zengin 3D animasyonlarla artırılmış gerçeklik teknolojisi ile görebilirler. Cihaz kamerasına gösterilen eğitsel kartların üzerinde artırılmış gerçeklik görüntüleri gösteren uygulamalar da AR teknolojisinin eğitim alanında kullanılmasına örnektir.
- **Bilgisayar Oyunları:** AR teknolojisinin en iyi kullanıldığı alanlardan biri de oyunlardır.

**Sanal Gerçeklik (VR-Virtual Reality)**

Kullanıcıların VR başlıklar ve farklı aparatlar kullanarak tamamen dijital bir dünyaya adım attıkları teknolojiye **Sanal Gerçeklik (VR)** denir (Görsel 4.62). Bu dünyalar gerçek dünyanın yansıması olabileceği gibi hayalî dünyalar da olabilir. Günümüzde geliştiriciler, yapay zekâ desteği ile şaşırtıcı derecede gerçekçi dünyalar oluşturabilmektedir. Bu sayede VR teknolojisini kullanan kullanıcılar, kendilerini sanal dünyada yaşıyormuş gibi hissederler.



**Görsel 4.62:** Sanal gerçeklik

Sanal gerçeklik teknolojisi, kullanılan donanım özelliklerine göre farklılık göstermektedir. Bunlar şu şekildedir:

- **Bilgisayar Tabanlı:** Bu tarz sanal gerçeklik başlıkları bir kablo ile bilgisayara bağlanarak kullanılır. İşlemler için bilgisayar gücü kullanıldığından çok gerçekçi dünyalar oluşturulabilir ancak kullanılan kablolar kullanıcıların hareketlerini kısıtlamaktadır.
- **Mobil Tabanlı:** Bir akıllı telefonun sanal gerçeklik başlığının içine yerleştirilmesi ile görüntü ve veri işlemleri kablosuz olarak sağlanır. Kablosuz olduğu için kullanıcılar daha rahat hareket edebilir.
- **Bağımsız:** Bu başlıklar kendi donanımını barındırdığı için başka bir cihaza ihtiyaç duyulmaz. Kablosuz olduğu için de kullanıcılar çok rahat hareket edebilirler.

VR teknolojisi pek çok alanda kullanılmaktadır. Bunlardan bazıları şunlardır:

- **Sağlık:** Doktor adayları, yaratılan sanal gerçeklik dünyasında cerrahi konularda pratik yapıp deneyim kazanabilirler.
- **Eğitim:** Pandemi sürecinde adı çok fazla konuşulan uzaktan eğitim, VR teknolojisi ile bambaşka bir boyuta geçebilir. Öğrenciler evlerinden hiç çıkmadan dünyanın herhangi bir yerindeki okulda istedikleri eğitimi alabilir, öğrenmesi tehlikeli konuları sanal dünyada güvenli bir şekilde deneyebilirler.
- **İnşaat:** VR teknolojisi sayesinde insanlar daha yapılmadan evi gezip, satın alıp almayacağına karar verebilirler.
- **Savunma Sanayi:** Askerler, sanal gerçeklik simülasyonları sayesinde farklı senaryolarda pratik yaparak taktik becerilerini geliştirebilirler.
- **Bilgisayar Oyunları:** VR teknolojisi, bilgisayar oyunları için adeta bir devrim niteliğindedir. Oyuncular tüm duyularıyla oyunun içindeymiş gibi hissederek değişik bir deneyim yaşarlar.

**Karma Gerçeklik (MR-Mixed Reality)**

Sanal gerçeklik ile artırılmış gerçekliğin özelliklerini bir araya getiren teknolojiye **Karma Gerçeklik (MR)** denir (Görsel 4.63). Bu teknoloji sayesinde kullanıcılar hem gerçek dünyayı görebilir hem de gerçek dünyaya entegre edilmiş nesnelerle etkileşime girebilirler. Karma gerçeklik gözlüklerinde gerçek dünyayı görmek için saydam bir lens veya ortamı algılayan bir kamera olmalıdır.

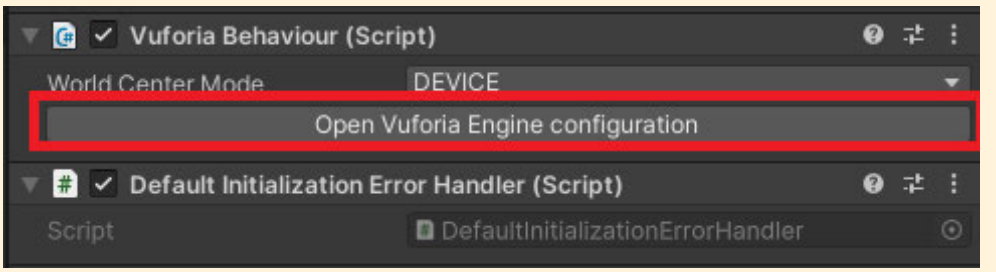


**Görsel 4.63:** Karma gerçeklik

## 14. UYGULAMA

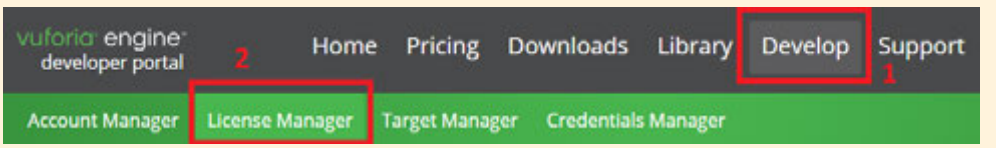
Uygulama adımlarını takip ederek telefon kamerasının bir görseli gördüğü zaman üzerinde 3 boyutlu nesne gösteren bir AR uygulamasını geliştiriniz.

**1. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.
**2. Adım: Asset Store** sitesinden **Vuforia Engine** paketini oyun motoruna ekleyiniz. **Vuforia Engine** için güncelleme uyarısı gelirse **Update seçeneği ile** uygulamayı güncelleyiniz.
**3. Adım: Project** panelinde bulunan "Assets>Sample Resources>Scenes" klasöründen **Main** isimli örnek sahneyi açınız.
**4. Adım: Hierarchy** panelinde bulunan **AR Camera** nesnesini seçip **Inspector** panelindeki **Vuforia Behaviour** scriptinin altında bulunan **Open Vuforia Engine configuration** butonuna basınız (Görsel 4.64).



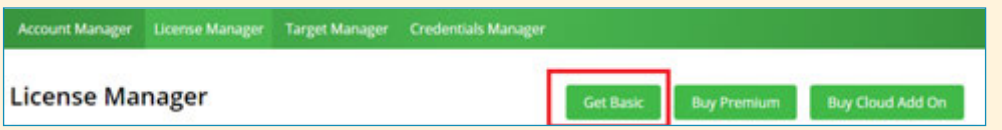

**Görsel 4.64:** Ayarları açma

**5. Adım:** Eklentinin sitesine giderek siteye üyelik oluşturunuz.
**6. Adım: Develop** sayfasından **License Manager** butonuna basınız (Görsel 4.65).



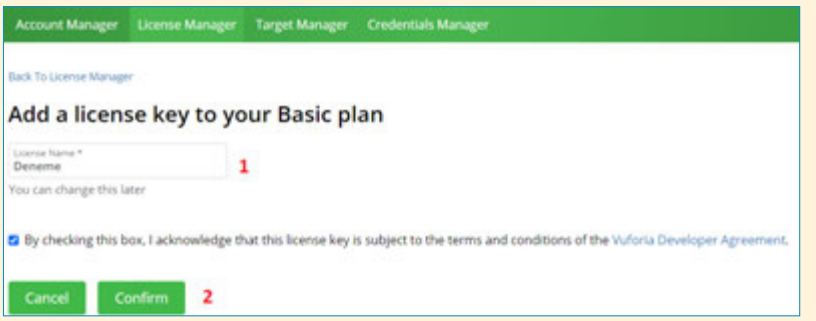

**Görsel 4.65:** License Manager menüsüne girme

**7. Adım: License Manager** sayfasında **Get Basic** butonuna basarak lisans anahtarı oluşturma sayfasına gidiniz (Görsel 4.66).




**Görsel 4.66:** Lisans anahtarı oluşturma sayfasına gitme

**8. Adım:** Lisans anahtarına istediğiniz ismi verip **Confirm** butonuna basınız (Görsel 4.67).



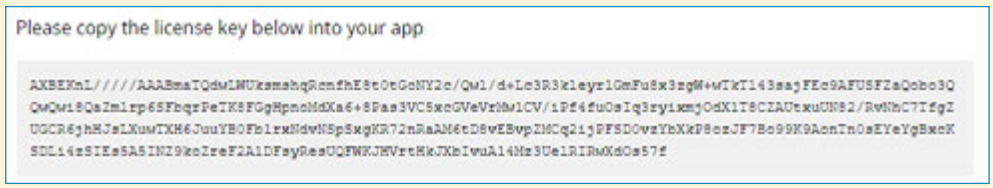

**Görsel 4.67:** Lisans anahtarı oluşturma

**9. Adım:** Oluşturulan lisans ismine tıklayınız (Görsel 4.68).



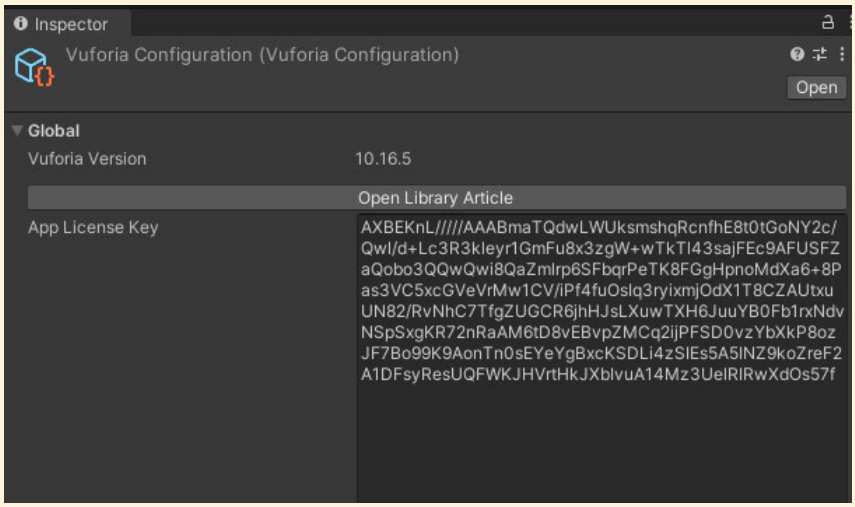

**Görsel 4.68:** Lisans ismine tıklama

**10. Adım:** Oluşan lisans anahtarının üzerine tıklayarak anahtarı kopyalayınız (Görsel 4.69).



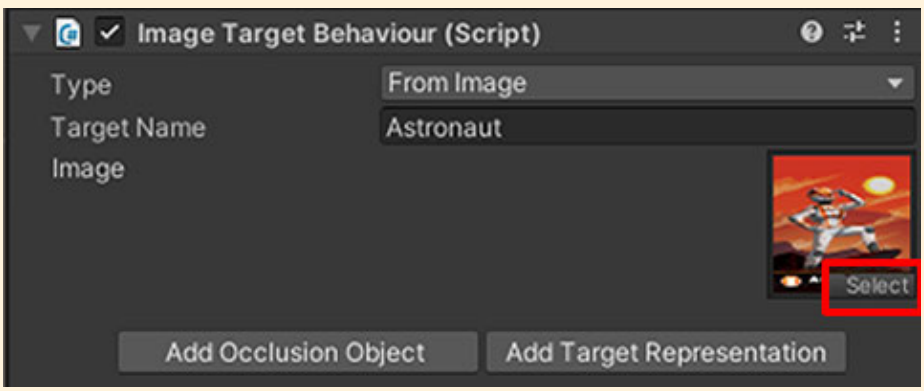

**Görsel 4.69:** Lisans anahtarını kopyalama

**11. Adım:** 9. adımda kopyalamış olduğunuz lisans anahtarını oyun motorunda **App License Key kısmına** ekleyiniz (Görsel 4.70).




**Görsel 4.70:** Lisans anahtarını ekleme

**12. Adım: Hierarchy** panelinde bulunan **Image Target** nesnesini seçip, **Select** butonuna basarak istediğiniz bir görseli seçiniz. İsterseniz astronot görseli de durabilir (Görsel 4.71).

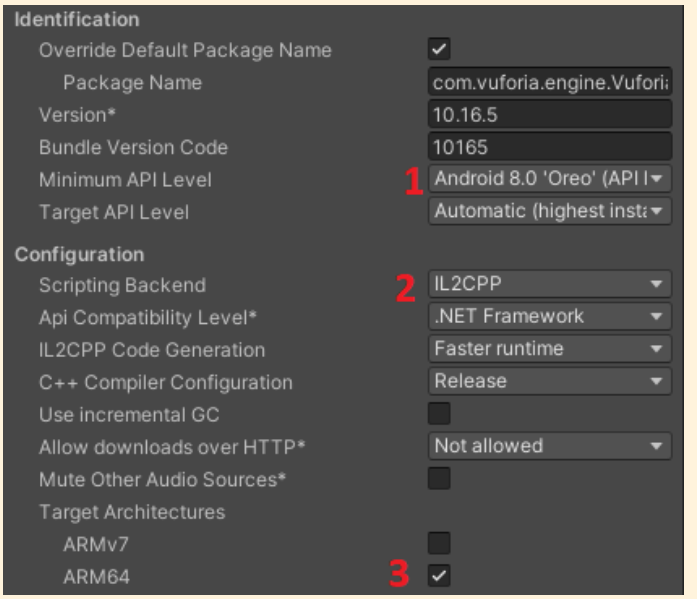



**Görsel 4.71:** ImageTarget nesnesi ile yüzey ekleme

NOT

Telefon kamerası bu adımda eklenilen görseli gördüğü zaman üzerinde Image Target nesnesinin çocuk nesnesi olan Astronaut nesnesi belirecektir. İsterseniz Astronaut nesnesi yerine başka bir 3 boyutlu model ekleyebilirsiniz.

**13. Adım: File>Save** komutu vererek sahneyi kaydediniz.
**14. Adım: File>Build Settings** menüsünden **Android platformunu seçip Switch Platform** butonuna basınız.
**15. Adım: Player Settings** butonuna basarak ayarları açınız.
**16. Adım: Other Settings** kısmından **Minimum API Level** seçeneğini **Android 8.0 (API level 26)** olarak ayarlayınız. **Scripting Backend** seçeneğini **IL2CPP** olarak değiştiriniz. **Target Architectures** seçeneğini de **ARM64** seçiniz (Görsel 4.72).




**Görsel 4.72:** Other Settings ayarları

NOT

Vuforia Engine yalnızca Android 8.0 üzeri versiyonlarda ve 64 bit platformda çalışmaktadır.

**17. Adım: Build Settings** penceresinden **Add Open Scenes** butonuna basarak sahne projeye ekli değilse sahneyi ekleyiniz. Ardından **Build** butonuna basarak, istediğiniz bir ismi verip APK dosyasını oluşturunuz.

NOT

APK dosyaları projenizin klasörü dışında bir klasörde yer almalıdır.

**18. Adım:** Android cihazınızı bilgisayarınıza data kablosu ile bağlayarak cihaz içinde belirlediğiniz bir klasöre bir önceki adımda oluşturduğunuz APK dosyasını yapıştırınız. Daha sonra APK dosyasını çalıştırıp cihazınıza kurunuz. Cihaz kamerası 12. adımda eklediğiniz görseli gördüğü zaman ekranda 3 boyutlu astronot modeli belirecektir.

NOT

APK dosyalarının cihazınıza aktarılması ve Kurulması için cihazınız geliştirici modunda olmalıdır. Cihazınızı geliştirici moduna almak için gereken ayarları İnternette bulabilirsiniz.

SIRA SİZDE

14. Uygulamadaki örneği farklı yüzey nesnesi ve 3 boyutlu model kullanarak yeniden oluşturunuz.

**Değerlendirme**

Çalışmalarınız aşağıda yer alan **Kontrol Listesi** kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

**Kontrol Listesi**

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| 1. Artırılmış Gerçeklik eklentisini projeye ekledi. | | |
| 2. Eklenti ayarlarını yaptı. | | |
| 3. Yüzey nesnesi ekledi. | | |
| 4. 3 boyutlu model ekledi. | | |
| "Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz. | | |

## 4.3.3. Uygulama Marketlerine Uygulama Ekleme

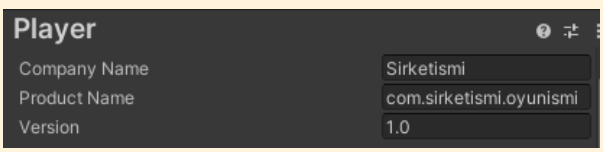
Büyük emekler verilerek yapılan oyunlarla ilgili son aşama, bunları başka insanların beğenisine sunmaktır. Eskiden bu iş için yayıncı bir firma ile anlaşılır; bu firma oyunun reklam, yayın, dağıtım ve satış konuları ile ilgilenirdi. Günümüzde büyük firmalar hâlâ yayıncılarla anlaşıp oyunlarını fiziksel olarak yayımlamaya devam etmekte ya da kendi dijital satış sistemlerini kullanmaktadır. Bunun yanında pek çok firma dijital satış için uygulama marketlerini kullanmaktadır.

Bu uygulama marketlerinin hepsinin ücretlendirmesi farklıdır. Kimisi oyunu yayımlamak için ücreti en baştan alır, kimisi oyun sattıkça oyundan belirli oranda komisyon alır.

15. UYGULAMA

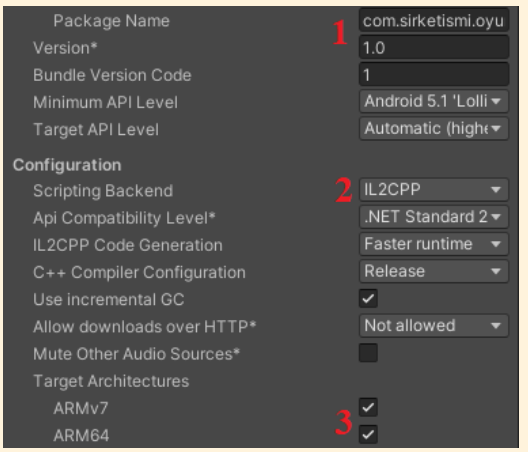
Uygulama adımlarına göre markette yayımlamak üzere bir oyunun ayarlarını yapınız.

**1. Adım:** Oyun motorunda yeni boş bir 3D Core uygulaması oluşturunuz.
**2. Adım: File>Build Settings** menüsünü açınız. Açılan pencerede Android platformunu seçip **Switch Platform** butonuna basınız. **Build App Bundle** seçeneğini seçiniz.
**3. Adım: Build Settings** penceresinde **Player Settings** butonuna basınız.
**4. Adım: Player Settings** penceresinde **Company Name** kısmına şirket ismini, **Product Name** kısmına oyun adını, **Version** kısmına da sürüm numarasını yazınız (Görsel 4.73).



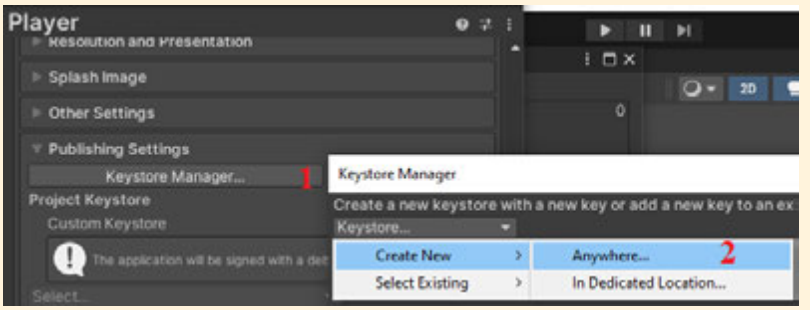

**Görsel 4.73:** Oyun ayarları

**5. Adım: Player Settings** penceresinde **Other Settings** kısmında bulunan **Package Name** seçeneğini **com.sirketismi.oyunismi** şeklinde ayarlayınız. **Scripting Backend** seçeneğini **I2LCPP** olarak ayarlayıp, **ARM64** seçeneğini seçiniz. Böylece oyununuz 64 Bit olarak derlenecektir (Görsel 4.74).




**Görsel 4.74:** Projeyi 64 bit olarak ayarlama

**6. Adım:** Oyununuzu markete yüklemeden önce dijital olarak imzalanması gerekmektedir. **Player Settings** penceresinin en altında bulunan **Publishing Settings** kısmında bulunan **Keystore Manager** butonuna basınız. Yeni açılan pencerede **Keystore>Create New>Anywhere** seçeneğini seçiniz. Dosyaya bir isim vererek bu dosyayı kaydediniz (Görsel 4.75).

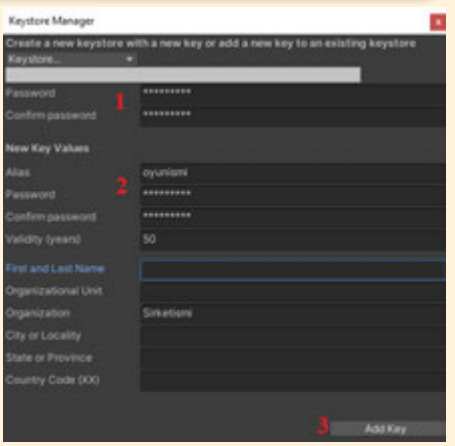



**Görsel 4.75:** Keystore Manager

NOT

İmzalanan bu dosyayı ileride oyunu güncellemek için kullanabilirsiniz. Ayrıca geliştirici hesabınız çalınsa bile kötü niyetli kişi, imzalı dosyanız olmadan oyunla ilgili bir değişiklik yapamaz. Bu yüzden bu dosya çok iyi saklanmalıdır.

**7. Adım: Password** ve **Confirm Password** kısımlarına istediğiniz bir şifreyi giriniz. **Alias** kısmına oyun adını yazıp buradaki **Password** ve **Confirm Password** seçeneklerine de istediğiniz şifreyi giriniz ve en alttaki **Add Key** butonuna basınız. Böylece oyununuz dijital olarak imzalanmış olacaktır. Bu adımda, yazdığınız şifreleri de unutmamalı ve çok iyi saklamalısınız. Yoksa oyunu güncellemeyle ilgili bir işlem yapamazsınız (Görsel 4.76).




**Görsel 4.76:** Keystore Manager ayarları

**8. Adım: Build Settings** penceresinden **Build App Bundle** seçeneğini seçiniz. **Build** butonuna basıp, dosyanıza isim vererek istediğiniz yere kaydediniz. Böylece AAB uzantılı oyununuz markete yüklenmeye hazırdır.

## 16. UYGULAMA

Uygulama adımlarına göre markette bir oyun yayımlayınız.

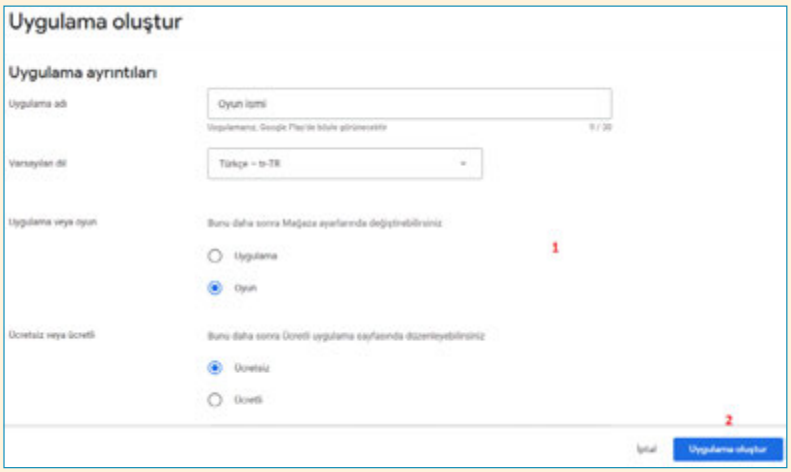
**1. Adım:** Geliştirici hesabınıza giriniz.

**2. Adım: Uygulama Oluştur** butonuna basınız (Görsel 4.77).



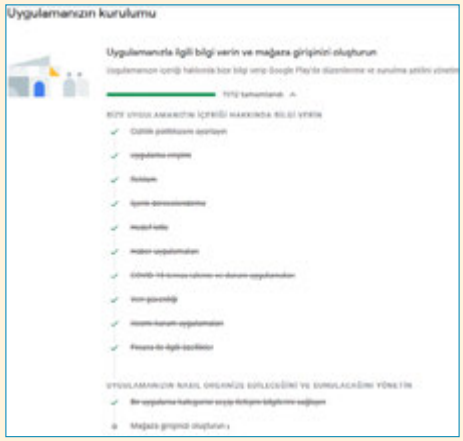

**Görsel 4.77:** Yeni uygulama oluşturma

**3. Adım:** Uygulama ayrıntılarını girerek **Uygulama Oluştur** butonuna basınız (Görsel 4.78).



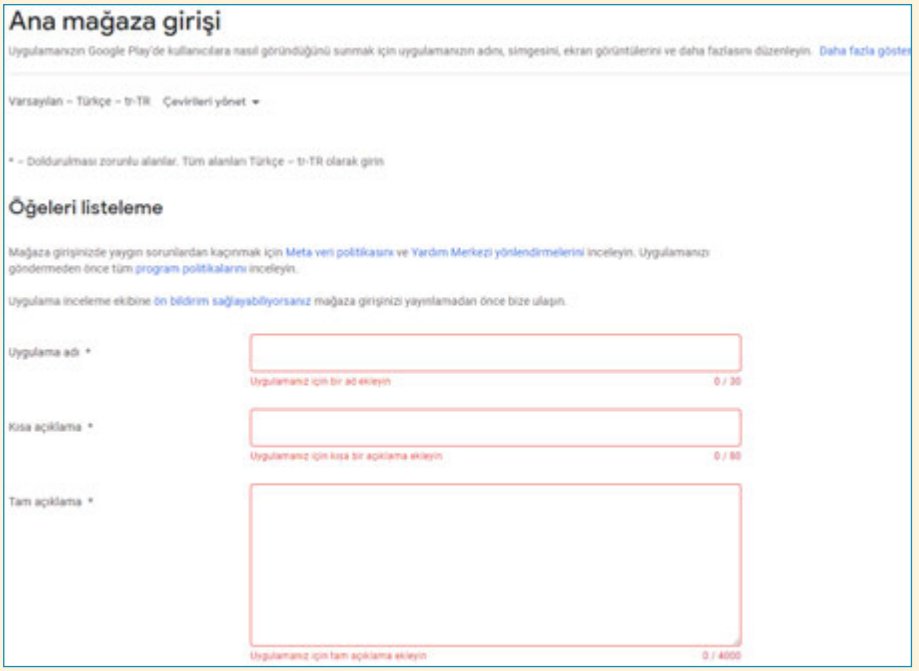

**Görsel 4.78:** Uygulama ayrıntılarını girme

**4. Adım:** Sonraki sayfada **Uygulama Kurulumu** ile ilgili adımlar bulunmaktadır. Google, verdiğiniz bilgilere göre oyununuzu inceleyip markette yayımlanması için oyunun uygun olup olmadığını denetleyecektir. Bu adımları tek tek tamamlamadan oyun yayımlama işlemi yapılamaz (Görsel 4.79).



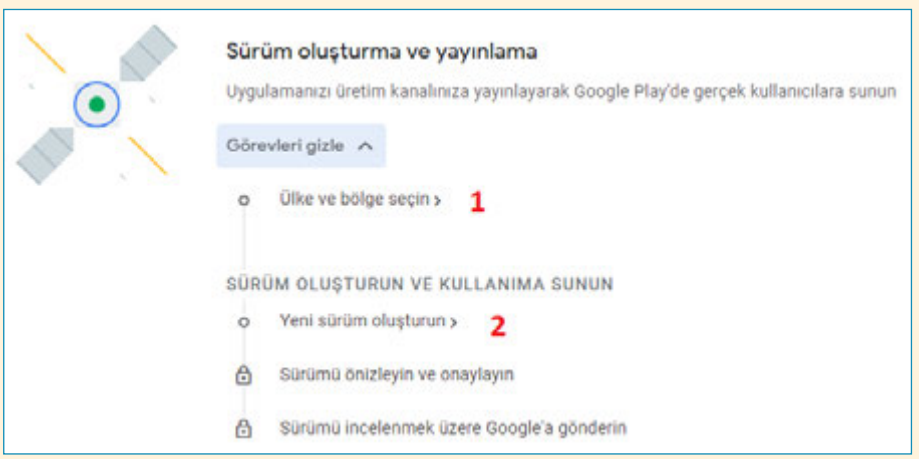

**Görsel 4.79:** Uygulama kurulumu

**5. Adım:** Uygulama kurulum adımlarının son aşaması mağaza girişinin oluşturulmasıdır. Kullanıcılar markette oyununuzun sayfasına girdikleri zaman burada verdiğiniz bilgilere göre oluşturulacak bir ekranla karşılaşacakları için bu adım çok önemlidir. Oyunun neyi amaçladığı, nasıl oynandığı ile ilgili tanıtım yazısı ve görsellerle etkileyici bir mağaza sayfası hazırlamalısınız (Görsel 4.80).




**Görsel 4.80:** Mağaza giriş sayfasını ayarlama

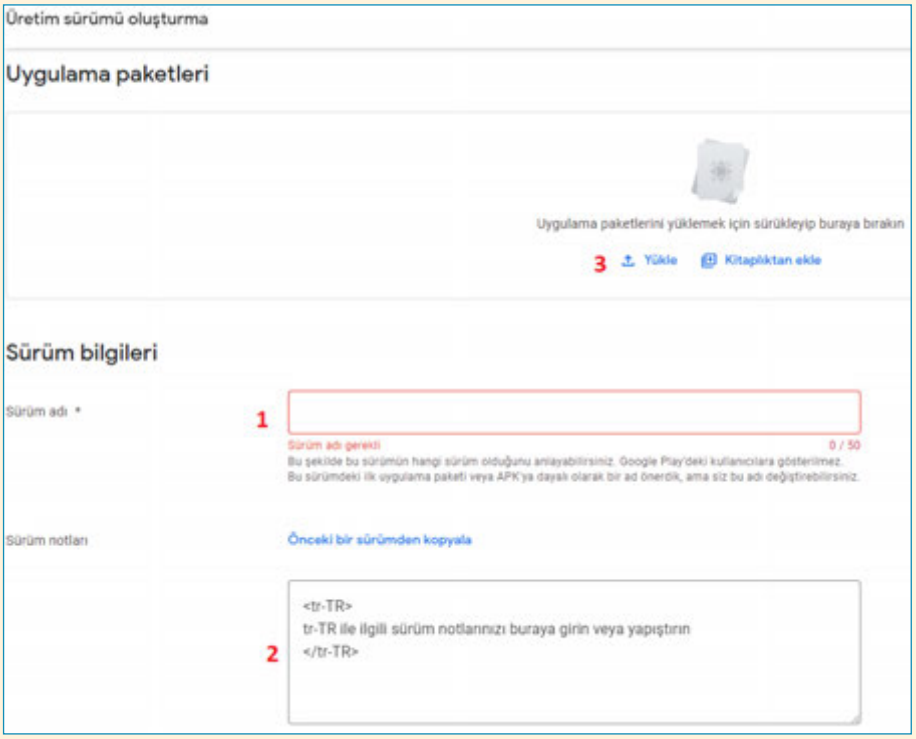
**6.Adım**: **Sürüm oluşturma ve yayımlama** kısmından oyunun hangi ülkelerde yayımlanacağını seçiniz. Bu işlemden sonra artık **Yeni sürüm oluşturun** butonuna basarak uygulamayı markete yükleme adımına geçebilirsiniz (Görsel 4.81).




**Görsel 4.81:** Sürüm oluşturma ve yayınlama

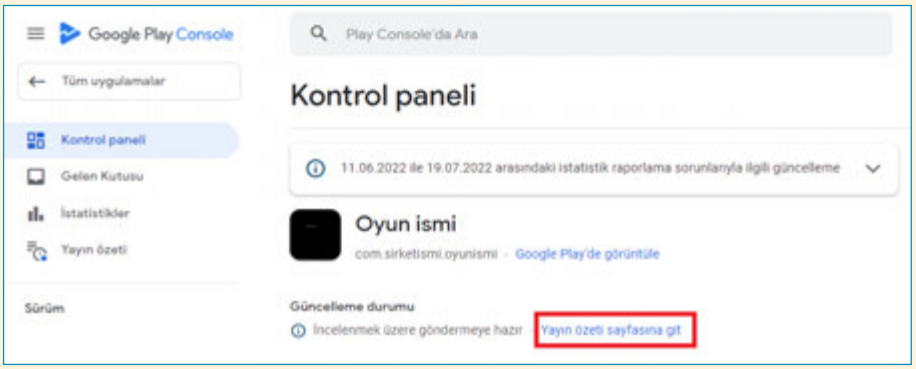
**7. Adım: Sürüm adı** kısmına istediğiniz bir ismi veriniz. Bu kısım, kullanıcılara görüntülenmeyecek sadece hangi versiyonu yüklediğinizle ilgili size bilgi verecektir. **Sürüm notları** kısmını da bu sürümün neler içerdiği ile ilgili kullanıcılara bilgi vermek için kullanabilirsiniz.

En son işlem olarak **Yükle** butonuna basarak oyunun APK veya AAB dosyasını yükleyiniz (Görsel 4.82).



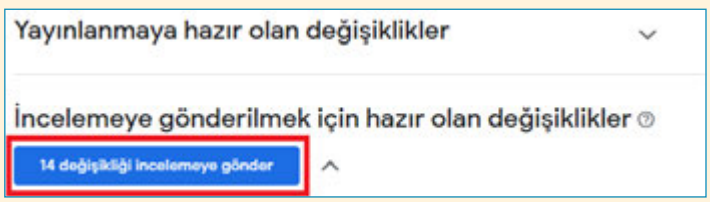

**Görsel 4.82:** Yeni sürüm oluşturma

**8. Adım: Kontrol paneli** sayfasında **Yayın özeti sayfasına git** butonuna basınız (Görsel 4.83).




**Görsel 4.83:** Yayın özeti sayfasına gitme

**9. Adım: Yayın özeti** sayfasında **Değişiklikleri incelemeye gönder** butonuna bastığınız zaman oyununuz incelemeye gönderilecek ve uygunsa birkaç gün içinde yayınlanacaktır (Görsel 4.84).




**Görsel 4.84:** Oyunu incelemeye gönderme

### 4.3.4. Güncelleme (Update) ve Yama (Patch) Ekleme

Oyunlar yayımlanmadan önce her ne kadar test edilse ve oyunların hataları giderilse de yayımlandıktan sonra da oyunlarda bazı hatalarla karşılaşılabilir çünkü oyun, test aşamasında kısıtlı sayıda kullanıcı ve cihazda denenirken yayımlandıktan sonra binlerce hatta milyonlarca kullanıcının kullanımına sunulur. Oyunu deneyen kullanıcılar, uygulama marketlerinin yorum kısımlarına oyunla ilgili geri bildirim yaparlar ve yapımcılar da buna göre oyuna müdahale ederler. Bu sayede oyun gittikçe kusursuz hâle gelir.

Ayrıca oyun yapımcılarının oyunun üretim aşamasında yetiştiremedikleri veya kullanıcının geri bildirimlerine göre sonradan eklemek istedikleri özellikler de bulunabilir.

Oyun yayımlandıktan sonra ortaya çıkan küçük hataları gidermek için yayımlanan küçük paketlere **yama (patch),** büyük çaplı hataları gidermek ve yeni özellikler eklemek için yayımlanan paketlere de **güncelleme (update)** denir.

17. UYGULAMA

Uygulama adımlarına göre bir oyununuzu güncelleyiniz.

**1. Adım:** Oyun motorunda güncellemek istediğiniz oyunu açınız.
**2. Adım: File>Build Settings** menüsünü açıp **Player Settings** butonuna tıklayınız.
**3. Adım: Other Settings** kısmında **Version** değerini 1.1, **Bundle Version Code** değerini 2 yapınız (Görsel 4.85).

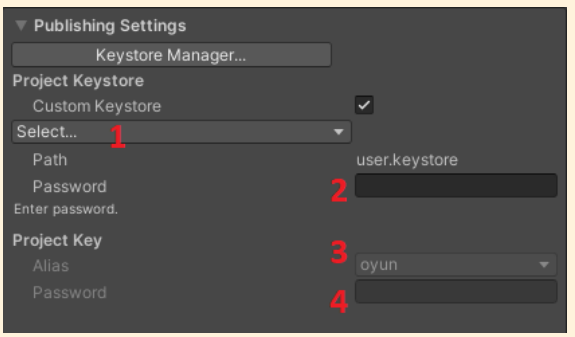



**Görsel 4.85:** Versiyonu ayarlama

NOT

Version kısmındaki değerler **soldan sağa doğru** değerlendirilir. Büyük çaplı bir güncelleme yapıyorsanız soldaki değeri, küçük bir yama yapıyorsanız sağdaki değeri artırmalısınız. **Bundle Version Code değerini** de her güncellemede bir önceki değerden yüksek bir değer girmelisiniz.

**4. Adım: Publishing Settings** kısmında oyunun ilk versiyonunu yüklerken oluşturulan ".keystore" uzantılı imza dosyasını seçiniz. **Password** kısmına imza dosyasını oluştururken girdiğiniz parolayı giriniz. **Alias** değerini seçiniz ve şifresini giriniz (Görsel 4.86).

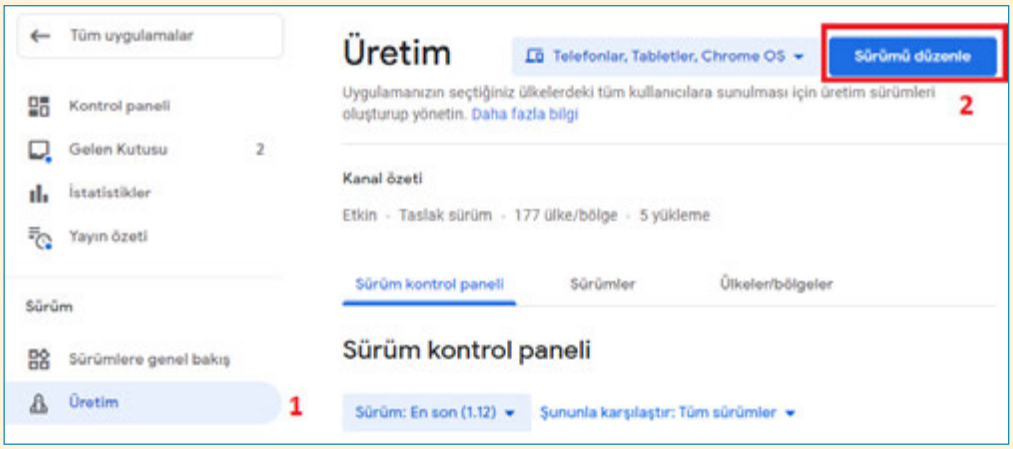



**Görsel 4.86:** İmza dosyasını seçme

NOT

İmza dosyasını kaybettiyseniz veya şifreleri hatırlamıyorsanız oyunu güncelleyemezsiniz.

**5. Adım: Build Settings** penceresinde **Build** düğmesine basarak oyunu derleyiniz. Oyununuz markette güncellenmeye hazırdır.
**6. Adım:** Geliştirici hesabınıza giriş yapınız.
**7. Adım: Tüm uygulamalar** sayfasında uygulamanızı seçiniz.
**8. Adım: Üretim** sayfasında **Sürümü düzenle** butonuna basınız (Görsel 4.87).




**Görsel 4.87:** Sürümü düzenle

**9. Adım: Sürüm adı** kısmına istediğiniz bir ismi, **sürüm notları** kısmına da bu sürümde hangi değişiklikleri yaptığınızı yazınız. 5. Adımda oluşturulan dosyayı yükleyerek incelemeye gönderiniz. Yapılan incelemenin ardından oyunun yeni versiyonu yayımlanacaktır.

## ÖLÇME VE DEĞERLENDİRME

**Aşağıdaki soruları dikkatlice okuyarak doğru cevabı işaretleyiniz.**

**1. Aşağıdakilerden hangisi oyuncu olmayan karakterlerin (NPC) özelliklerinden olamaz?**

A) Kendi kendine karar verme

B) Yol bulma

C) Oyuncu diyaloglarına yanıt verme

D) Oyunculara görev verme

E) Oyundan çıkma

**2. Aşağıdakilerden hangisini oyuncular lobi ekranında hangisini yapamaz?**

A) Oyunla ilgili ayarlar yapma

B) Başka oyuncularla iletişim kurma

C) Oluşturulmuş olan oyunları görme

D) Oyunu oynama

E) Başka oyuncuların profillerini görme

**3. Eklendiği nesneye ağ kimliği kazandırmaya yarayan Photon bileşeni aşağıdakilerden hangisidir?**

A) Transform

B) Animator View

C) View

D) Teams Manager

E) Rigidbody View

**4. Aşağıdakilerden hangisi oyunu test etme aşamalarından birisi değildir?**

A) Kapalı Beta B) Gama C) Açık Beta

D) Dâhili E) Alfa

**5. Aşağıdakilerden hangisi reklam türlerinden birisi değildir?**

A) Banner B) Interstitial C) Native

D) Platform E) Rewarded

**6. Aşağıdakilerden hangisi oyun içi yardım türlerinden değildir?**

A) Yol gösterici B) İpucu C) Çevrim içi

D) Öğretici E) Yardım menüsü

**7. Aşağıdakilerden hangisi oyun motorunda bulunan optimizasyon aracıdır?**

A) Helper B) Optimizer C) Profiler

D) Updater E) VR

**8. Aşağıdakilerden hangisi oyun motorunda derlenebilen platformlardan değildir?**

A) Web tarayıcı B) Mobil cihaz C) Konsol

D) PC E) 3D yazıcılar

**9. Oyuncuların kendini tamamen oyunun içinde hissettikleri gerçeklik türü aşağıdakilerden hangisidir?**

A) AR B) BR C) CR

D) MR E) VR

**10. Nesneleri birleştirerek yeni nesne oluşturma işlemine ne ad verilir?**

A) Crafting B) Satın alma C) Test

D) Patch E) Update

## CEVAP ANAHTARLARI

| 1. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1) B | 2) D | 3) A | 4) A | 5) B | 6) D | 7) A | 8) D | 9) C | 10) E |
| 2. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI | | | | | | | | | |
| 1) E | 2) D | 3) A | 4) C | 5) E | 6) D | 7) C | 8) B | 9) A | 10) B |
| 3. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI | | | | | | | | | |
| 1) C | 2) C | 3) A | 4) D | 5) B | 6) D | 7) E | 8) D | 9) B | 10) C |
| 4. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI | | | | | | | | | |
| 1) E | 2) D | 3) C | 4) B | 5) D | 6) C | 7) C | 8) E | 9) E | 10) A |

## KAYNAKÇA


- Türk Dil Kurumu Güncel Türkçe Sözlük
- Türk Dil Kurumu Güncel Yazım Kılavuzu
- Bilişim Teknolojileri Alanı Çerçeve Öğretim Programı, Ankara 2023


## KAYNAKÇAYA AİT KAREKOD



Kitaba ait genel ağ kaynakçasına ve görsel kaynakçasına ulaşmak için yandaki karekodu tarayınız.

Karekodu tarayacak cihazınız yoksa aşağıdaki linkten genel ağ kaynakçasına ve görsel kaynakçaya ulaşabilirsiniz.
» http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=3551

DERS NOTLARIM