AR Oyun Uygulaması

AR Game Application

1. Betül Erdik   
*Bilişim Sistemleri Mühendisliği*  
*Kocaeli Üniversitesi*Kocaeli, Türkiye  
211307033@uzem.educatin

2. Buse Çakal   
*Bilişim Sistemleri Mühendisliği*  
*Kocaeli Üniversitesi*Kocaeli, Türkiye  
221307084@uzem.education

3. Narti Gurmani   
*Bilişim Sistemleri Mühendisliği*  
*Kocaeli Üniversitesi*Kocaeli, Türkiye  
191307085@uzem.education

*Özetçe*—Bu belge, TUAC bildirisi hazırlanması adına bir taslak içermektedir. Bu sebeple lütfen taslaktaki başlık, özet ve diğer format stillerini kullanınız. *\***DİKKAT: Bildiri Başlığı’nda ve Özetçe’de Sembol, Özel ve Matematiksel Karakterler kullanmayınız*. (*\*Abstract*)

Anahtar Kelimeler — ARCore, Unity, Oyun Uygulaması, 3D Uygulama

# Giriş

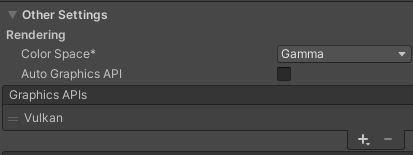
Dünyada kullanımı gittikçe yayılan yeni bir teknoloji olan AR(Augmented Reality) yani Türkçesi arttırılmış gerçeklik olan uygulamalar aslında günümüz teknolojisiyle yapılması oldukça kolaylaştırılmış uygulamalardır. ARCore ortam tanıma özelliği sayesinde gerçek nesnenin boyutunu tanır ve uygulamanın bu boyutlar doğrultusunda daha doğru ve sorunsuz bir şekilde çalışmasını sağlar. Ortam tanıma, hareket izleme ve ışık tahmini olmak üzere üç temel nokta üzerine kurulmuştur. Günümüzde birçok farklı yöntemle uygulamalar geliştirilmektedir. Unity de bunlardan birisidir.

# Unity kurulumu ve proje dosyası oluşturma

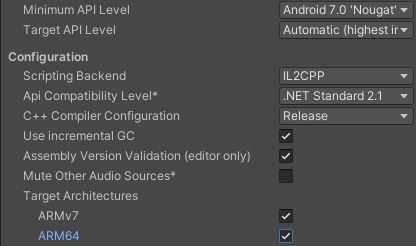
Projeyi geliştirmek için Unity platformunu kullanıyoruz. Bu projeyi Unity’nin 2021.3.6f1 sürümünü kullanarak yaptık. Unity’nin 2021.3.6f1 sürümünü indirdikten sonra modül ekleme bölümünden Unity’nin ihtiyaç duyduğu Android SDK, NDK, JDK ve uygun gördüğü Visual Studio programını indiriyoruz. İndirmeleri tamamladıktan sonra yeni proje başlatıyoruz ve Unity 2021.3.6f1 sürümünü seçtiğimizden emin oluyoruz. Ardından 3D özelliğini seçip projeyi isimlendiriyoruz. Ve projeyi oluşturuyoruz.

# proje paket yüklemeleri ve ayarlarının yapılması

İlk olarak proje için gerekli olan paketleri yüklüyoruz. Paket yöneticisinden AR Foundation ve ARCore XR Plugin paketlerini yüklüyoruz. Ardından uygulamayı başlatmak için kullanacağımız bir resim, 3D karakter ve oyun kolu belirliyoruz. Kullandığımız kaynaklara kaynaklar bölümünden ulaşabilirsiniz. İndirdiğimiz 3D karakteri ve oyun kolunu Unity’de Package Manager’dan import ediyoruz. Sonrasında indirdiğimiz resmi de assets dosyası altına atıyoruz. Eklentilerimizi tamamlayıp proje dosyası oluşturmaya geçiyoruz. Build setting üzerinden Android’i seçip Switch Platform diyerek Android Platform’a geçiyoruz. İşlem bittiğinde oyuncu > diğer ayarlardan Auto Graphics API özelliğini seçmeyip vulkan öğesini kaldırıyoruz. Böylece açılış API’siyle devam ediyoruz. (Resim1)Multithreaded rendering seçeneğini kaldırıyoruz ve istediğimiz minimum API Leveli seçiyoruz. Scripting backend seçeneğinde mono’yu değiştirip IL2CPP seçeneğini seçiyoruz. (Resim2) ARM64 seçeneğini de seçtikten sonra XR Pug-in management üzerinden Android bölümüne gelip ARCore seçeneğini işaretliyoruz. Böylece proje için gerekli ayarlamaları bitirmiş oluyoruz.



Resim1



Resim2

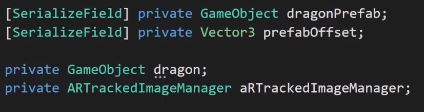
# başlangıç resmi ekleme

İlk olarak main camera yerine AR Camera kullanmamız gerekiyor. Bunun için de main camera’yı kaldırıp yerine sağ tuş XR > AR session origin diyerek AR oturum kaynağını ekliyoruz. AR camera da bu eklentiyle beraber geliyor. Aynı şekilde yine sağ tuş ile AR session’ı ekliyoruz. Daha sonra seçtiğimiz resmi AR Session kaynağı olarak belirtmemiz gerekiyor. Bunun için AR session’da add component > AR Tracked Image Manager seçiyoruz. Amacımız buraya bir görüntü vermek ve görüntü algılandığında Prefab oluşturacak ayrı bir komut dosyası yazmak. Bu yüzden ilk olarak assets altında sağ tuş Create > XR > Referance Image Library seçeneğini seçiyoruz. Oluşturulan dosyada add image diyerek assets altındaki başlangıç resmini buraya ekliyoruz. Ardından AR Session Origin altındaki Tracked Image Manager’da Seriallized Library kısmına referans verdiğimiz resim kütüphanesini sürüklüyoruz. Add component ile yeni bir script dosyasını isimlendirme yaparak oluşturuyoruz.

# objeler için script kullanılması

## Nesne Tanımlamaları

Açılan Visual Studio uygulamasında UnityEngine.XR.ARFoundation kütüphanesini ekliyoruz. Sınıfın içerisine serileşmiş özel alandan dragonPrefab adında bir oyun nesnesi oluşturuyoruz. Sonrasında prefabOffSet adında 3 boyutlu eksenleri ifade eden bir vector3 tanımlıyoruz. Ardından somutlaştırılmış dragonPrefab’ ı depolayacak bir dragon nesnesi oluşturuyoruz. Projede kullandığımız görüntü yöneticisini de kodda tanımlıyoruz. (Resim3)



Resim3

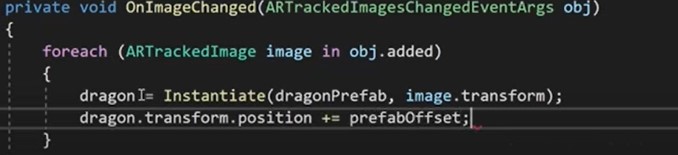
## Fonksiyon Tanımlamaları

Nesneyi etkinleştirebilmek için onEnable adında bir fonksiyon tanımlıyoruz. Uçuş görüntüsünü almak için ARTractedImageManager’ ı gameObject e eşliyoruz. Değişen görüntü için ARTractedImageManager’ dan ulaştığımız tractedImagesChanged’i ekrandaki görüntüye ekliyoruz. (Resim4). OnImageChanged üzerinde sağ tık yaparak buradan hazır fonksiyon oluşturuyoruz. Bu fonksiyondan obje elde edeceğiz.



Resim4

Bu fonksiyonda ejderha örneğini oluşturmak ve sonra onu ejderha değişkeni içinde saklamak istiyoruz. Ejderhayı somutlaştırmak için Instantiate kullanılır. Konumunu ayarlamak için image.transform’ u kullanıyoruz. Görüntüyü dengelemek için ejderha nokta konumunu sahip olduğumuz offset değerine eşitliyoruz (prefabOffset). (Resim5)

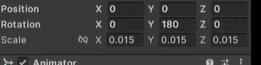


Resim5

## Unity Görüntü Ayarlamaları

Bu işlemleri kaydedip Unity’ ye geri döndük. İnspector içindeki Prefab Creator da bir ejderha örneği saklamamız gerekli. Burada sahip olduğumuz herhangi bir ejderha örneği kullanılabilir. Burada yüklediğimiz ejderha dosyasının üstüne tıklayarak Insprector penceresinde boyut ayarlamalarını yaparız. Bu ejderha aynı zamanda bir animasyona sahip. Bu animatörü açtık. Bu animatörde ihtiyacımız olan havalanmak, ileri uçmak bu yüzden diğer animasyonları siliyoruz. Bu animasyonlar arasında bir geçiş oluşturuyoruz. Ekrana geri dönüyoruz ve oynata basıldığında ejderha havalanır ve aynı konumda uçar. Örneğimiz artık hazır.

Sonrasında ejderha örneğine geliyoruz ve Overrides’ tan Apply All’ a tıklıyoruz. Ayarları tümüne uygulamış oluyoruz. Rotation dan X’ i 0, Y’ i de 180 olarak ayarlıyoruz.



Resim6

Daha sonra Hierarchy’ deki ejderha örneğini siliyoruz. AR sistemini seçiyoruz. Oluşturduğumuz örnek ejderha dosyasını İnspector içindeki Prefab Creator içine sürükleyip bırakıyoruz. Dengeleme yapmak için Prefab Offset‘ deki konumlandırmaları kullanıyoruz.

Hareket etsin istiyoruz. Bu yüzden ejderhayı seçip bir bileşen daha ekleyeceğiz. Yeni bir komut dosyası yaratacağız. Add Component’ e tıklayıp Drogon Controller oluşturuyoruz.

Visual Studio’ ya geçiyoruz. Burada ilk olarak hız değişkenini tanımlıyoruz.



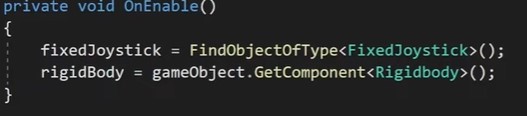
Resim7

İki değişken daha tanımlayacağız ve bunların biri sabit joystick bileşenini saklayacak, ikincisi sabit gövde için olacak. fixedJoystick diye sabit joystick tanımlıyoruz. Rigidbody adında gövde bileşeni tanımlıyoruz.



Resim8

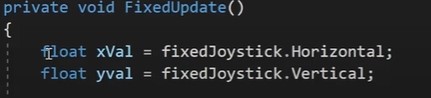
OnEnable adında bir fonksiyon tanımlaması yapıyoruz. Sabit joystick tanımlamasını yapıyoruz ve onu sabit olmasını istediğimiz türden nesneye eşitliyoruz. Dijital gövde bileşenini oyun nesnesine eşitliyoruz.



Resim9

FixedUpdate fonksiyon tanımlamasını yapıyoruz. Ejderhayı hareket ettirmek için joycticklerin X ve Y değerlerini almayız. Bu x ve y değerlerini float olarak tanımlıyoruz.

X değişkeni sabit joystickin nokta yatay eksenine eşit olacak. Y değişkeni de sabit joystick nokta dikey eksenine eşitleriz. Böylece hangi yöne hareket etmesini istediğimizi tanımlayabiliriz.



Resim10

Sonrasında Vector3 anını yeni Vector3’e eşitleriz. Ejderhanın yukarı ve aşağıya hareket etmesini istemediğimiz için

(a)Vector3(xVal, 0, Yval), olarak tanımlama yaparız.

Böylece yukarı ve aşağıya hareket olmaz ama x ve z düzleminde hareket sağlanır. Yani artık konum bulunduğunda onu belli bir yönde hareket ettirebiliriz. Bunun için rigitBody nokta hızın z düzlemine eşitleriz.



Resim11

Şimdi ejderha hareket eder ama ejderha hareket yönüne bakmaz. Bunu çözmek için x’in sıfıra eşit olmadığı ve y’nin de sıfıra eşit olmadığı durumda;



Resim12

Ejderhanın y ekseni etrafında dönmesini istiyoruz. Bu da x ve z ekseni etrafında dönüşün sabit kaldığı anlamına gelir. Bu yüzden x ve z bu şekilde kullanılır. Şimdi bu komut dosyasıyla oynatıcıyı hareket ettirebilmeliyiz. Kaydedip Unity’ ye dönüyoruz. O halde joystick’i eklememiz gerekli bunu için sağ tık yaptıktan sonra UI içinde Canvas’a tıklıyoruz. Canvas’ta Cancvas Scaler da Scale With Screen Size olarak değiştirelim. Ölçeği ekran boyutuna göre ölçeklendireceğiz. Bunun için Reference Resolution’ ı kullanacağız.



Resim13

Boyutunu Resim13 te verdim. Bu ayarlamayı yaptıktan sonra joystick prefabriklerine gideceğiz ve buradan sabit joystickimizi Hiyerarchy içine sürükleyip Canvas üzerine bırakıyoruz. Şimdi joystick ekranda görülür. Üzerine çift tıklayıp kolay kullanım için ölçeğini 2 ye çıkaracağız.

Son olarak ejderha prefabriğimizde bazı ayarlamalar yapmamız gerekiyor. Inspector içinde bir Rigibody bileşeni ekliyoruz. Burada Use Gravity sekmesindeki işareti kaldırıyoruz. Sonra Dragon Controller bileşenini de ejderha prefabriğine ekliyoruz. Burada hızı 0,5 e ayarlıyoruz. Sonrasında Overrides’a tıklayıp Apply All yapıyoruz. Sonra ejderha prefabriğini kaldırıyoruz. Sonrasında sahneyi kaydediyoruz. Cihazda oluşturacağız.

## Cihazda Çalıştırma

File, Build Settings, yaptıktan sonra cihazımızı bağlamamız gerekli. USB hata ayıklama etkinleştirilir, sonrasında Run Device a cihazınızı işaretleyin. Build And Run a tıklayıp açılan sekmede sağ tık yapılır ve yeni dosya oluşturulur(Builds). Oluşturulan dosya açılır ve test yazılıp çalıştırılır. Sonuç olarak telefon kamerasıyla çalıştırılır.

##### Kaynaklar

1. <https://media.glamour.com/photos/5cd306ed0d36451ee721de38/master/pass/8c25cdbef5f71ceebb6242d7b654ac6c0ed94cce2ad088337a686dbca480913257b862fe1511330dd9e212e9be8c3cb3.jpg>
2. <https://assetstore.unity.com/packages/3d/characters/creatures/dragon-for-boss-monster-hp-79398>
3. <https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631>
4. <https://github.com/betlet1/AR_App_2023>