# Artırılmış Gerçeklik ile Güneş Sistemi Simülasyonu

# Solar System Simulation With Augmented Reality

*Kaan Toraman, Eda Obuz, Hacer Sueda Efe*

Bilişim Sistemleri Mühendisliği - Teknoloji Fakültesi

Kocaeli Üniversitesi

kaan41tor@gmail.com , edaobuz4@gmail.com , suedaefe@gmail.com

## Özet

Bu rapor Unity'de geliştirilen Artırılmış gerçeklik tabanlı güneş sistemi simülasyonu mobil uygulaması projesine ait problem tanımı, yapılan araştırmalar, akış şeması, kullanıcı arayüzü, projenin backendi ve projenin genel yapısıyla ilgili bilgiler içermektedir. Bu uygulamada ARCore ve C# kullanılmıştır.

## Abstract

This report contains information about the problem definition, research, flow chart, user interface, background of the project and general structure of the Augmented reality-based solar system simulation mobile application project developed in Unity. ARCore and C# are used in this application.

## 1. Problem Tanımı

Bu uygulamayı geliştirmedeki amacımız ilkokul çağındaki çocuklar için eğitici ve öğretici bir içerik sunmaktı. Bazı şeyleri öğrenmek bu yaş grubundaki çoğu çocuk için zorlayıcı olabilir, sıkılıp öğrenmek istemeyebilirler bu yüzden geliştirdiğimiz uygulamada artırılmış gerçeklik kullanarak çocukların ilgisi çekmeyi, bu görsel araçlarla bilgi öğrenmelerini daha eğlenceli bir hale getirmeyi planladık. Öğrendikleri bilgiler taze kalsın, tekrar edilsin diye de interaktif bir quiz ortamı oluşturduk. Bu quiz ortamında yanlış verilen cevaplarda açıklama pencereleri kullanarak öğrenme sürecini pekiştirmeyi hedefledik.

## 2. Yapılan Araştırmalar

Proje gereklilikleri doğrultusunda, Arkit veya ARCore kullanılması gerektiği için, kullanılabilecek motorlar araştırıldı. Araştırma sürecinde, birbirinden farklı motorlarla karşılaşıldı ancak dikkatimizi çeken iki seçenek oldu: Android Studio ve Unity Engine. Bu motorlar arasından yapılan araştırmalar ve ekip oylamaları sonucunda, Unity Engine kullanma kararı alındı.

Unity Engine, ARKit ve ARCore'u AR Foundation ismi altında başarıyla destekliyor ve bu nedenle projemizle uyumlu bir çözüm sunuyor. Unity içerisinde, AR Foundation ile entegre bir şekilde çalışabilen demo sahneleri ve kapsamlı dökümanlar bulunması da bu kararımızı destekledi.

Projemiz, bir uzay simülasyonu olduğu için gezegen modellerine ihtiyaç duyduk. Üç boyutlu bir deneyim sunulacağı için 3D modeller gerekliydi. Neyse ki, oyunlar için ücretsiz modeller ve kaynakları sağlayan çeşitli siteler sayesinde bu ihtiyacımız kolayca karşılanabilindi.

Uygulamada gezegenlerle ilgili detaylı bilgiler sunulması gerektiği için, bu konuda derinlemesine araştırmalar yapıldı. Ayrıca, uygulama içinde kullanıcıları eğlendirmek ve bilgilendirmek amacıyla quiz soruları hazırlamak için bu araştırmalar temel alındı. Yanlış cevapların ardındaki mantığı anlatan açıklamalar da eklenerek, kullanıcıların öğrenme deneyimini artırmayı hedefledik.

Projenin merkezine Güneş Sistemi'ni yerleştirirken, bu büyük ölçekli uzay simülasyonu için detaylı araştırmalara odaklandık. Güneş Sistemi, güneş, gezegenler, uydular, asteroid kuşakları ve diğer gök cisimlerinden oluşan karmaşık bir ekosistemdir. Her bir gezegenin yörüngesi, atmosferi ve benzersiz özellikleri üzerinde derinlemesine bilgi sahibi olabilmek için detaylı kaynaklar üzerinde yoğunlaşıldı.

Güneş'in merkezi rolü ve diğer gök cisimleriyle olan etkileşimleri üzerinde durularak, enerji üretimi ve bileşenleri hakkında ayrıntılı bir analiz yapıldı. Her bir gezegenin kendi içindeki dinamikleri ve özellikleri, Güneş Sistemi'nin genel yapısını anlamak adına özel bir vurgu aldı.

Kullanıcıları eğlendirmek ve bilgilendirmek amacıyla hazırladığımız quiz soruları içinde, Güneş Sistemi'nin derinliklerindeki ilginç gerçeklere ve özel özelliklere odaklanılarak, bu bilgileri etkili bir şekilde iletmeyi hedefledik.

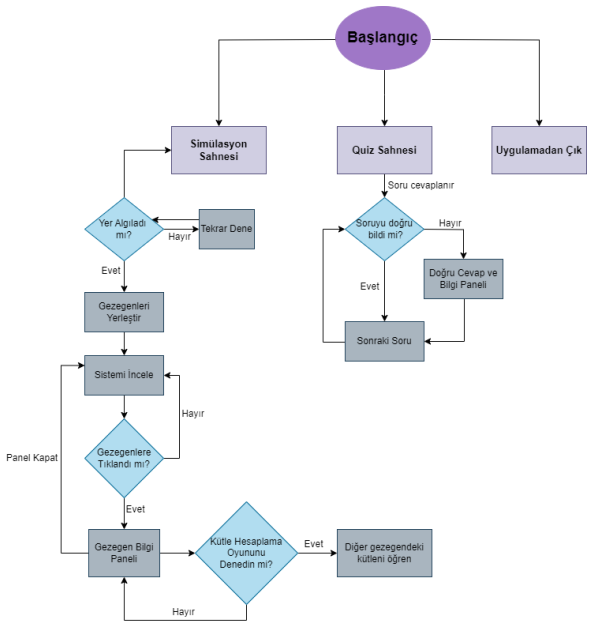
Simülasyon deneyimini zenginleştirmek adına, kullanıcıların Güneş Sistemi ile etkileşimde bulunmasını önemli bir hedef olarak belirlendi. Ancak, bu projeyi gerçekleştirmek için yeterli sayıda kaynağa sahip değildik. Bu zorluğun üstesinden gelmek adına, AR teknolojisi dışında Unity projelerinde yaygın olarak kullanılan 2D ve 3D kaynakları inceleme ve kullanma fırsatı bulduk. Unity Asset Store, itch.io ve opengamearts.org gibi platformlar üzerinde kapsamlı araştırmalar gerçekleştirdik. Bu platformlardan temin edilebilecek çeşitli 2D ve 3D kaynaklarını keşfetmek, simülasyonumuzun görsel ve etkileşim açısından daha zengin bir deneyim sunmasına olanak tanıdı.

## 3. Akış şeması

Örnek akış şeması Şekil-1'de verilmiştir. Uygulama açıldıktan sonra karşımıza üç buton çıkmaktadır. Bu butonlar sırasıyla ana simülasyonu açan buton, quizi açan buton ve uygulamadan çıkmaya yarayan butondur.

Ana simülasyona gidildiği zaman öncelikli olarak yer algılama çalışır. Eğer başarısız olursa başarılı olana kadar yer algılamaya devam eder. Başarılı olduğu zaman gezegenleri yerleştirebilir. Gezegenler yerleştirildiklerin de çeşitli etkileşimlere girilebilir.

Quiz sahnesine gidildiği zaman kullanıcılara gezegenler ve güneş sistemi hakkında sorular soruyor ve kullanıcıların bilgi edinmesini sağlıyor.




*Şekil 1: Akış Şeması*

## 4.Genel Yapı

Proje ilk açıldığında karşımıza ilk olarak ana sayfa çıkmaktadır. Ana sayfadan iki farklı sahneye ulaşılabilmektedir. İlk sahnede AR simülasyonu bulunmaktadır. Diğer sahnede simülasyonda verilen bilgilere yönelik oluşturulmuş küçük bir quiz bulunmaktadır. Ek olarak konulmuş buton ile de uygulamadan direkt çıkış yapılabilinmektedir.

## 5. Tasarımlar

### 5.1 3D Modelleme

Proje içinde kullanılacak modellere karar verildikten sonra modeller projede kullanılmak üzere indirilmiştir. Web sitesi üzerinde modeller texture ile beraber bulunmalarına rağmen bilgisayar üzerinde tasarımlar gözükmemektedir. Tasarımları modellere giydirmek için Blender uygulamasından yararlanılmıştır. FBX formatında bulunan modeller Blender içerisine aktarılmıştır. İndirilen dosya içinde bulunan texture tasarımları uygulama içinden modeller üzerine giydirilmiştir.

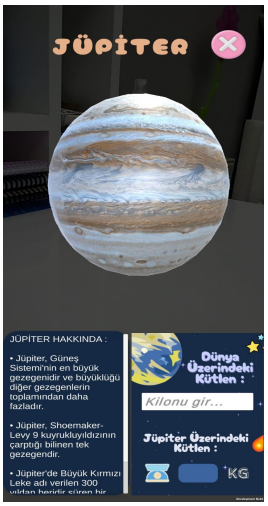
### 5.2 Kullanıcı Arayüzü (UI)

Uygulamanın menü UI'ı için önce menü sahnesindeki canvas içerisine bir arkaplan resmi eklendi, bu resmin tasarımı canva kullanılarak yapıldı. Resmi unity de kullanabilmek için resmi unity'ye ekledikten sonra texture type'ı sprite olarak değiştirmek gerekti. Bu işlemi yaptıktan sonra canvas'ın içerisine eklenen image elementeninin source image kısmına bu resim sürüklenip bırakıldı. Arka planın ardından butonlar eklendi, bu butonların image kısmına internetten bulunan bir asset eklenerek butonlara şekil verildi. Butona gölgeli bir görüntü vermek için add component kısmından shadow seçilerek gölge efekti x=15 y = -15 olarak ayarlandı.


*Şekil 2 : Menü*

Simülasyona girdikten sonra gezegen bilgisi görüntülemek istediğimizde çıkan ekran için yine canvas içerisinde işlem yapıldı. Üst kısımda text ile gezegen adı yazıldı, çıkış işlemi için buton eklendi. Aşağı kısma bilgi için UI->scrollview kısmından scrollview eklendi, sadece

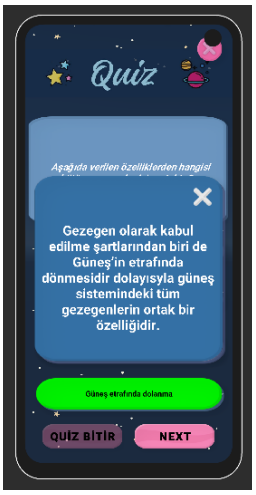
yukarıdan aşağı hareket istendiği içi scrollview'in horizontal ögeleri kapatıldı ( scrollrect horizontal ve scroll bar horizontal kısımları). Daha sonra content kısmına textmesh pro ve content size fitter eklendi content size fitter'da vertical fit = preferred size olarak ayarlandı. Ardından scrollview'in arka planına düz renk bir image konarak düzgün bir görüntü oluşturdu. Bilgi kısmının yanına ise başka gezegenler üzerinde kütle hesabı için bir alan oluşturuldu, bu alanın arka planı yine canvada tasarlandı, input ve output için de alan belirlendi arka planın üstüne eklendi.




*Şekil 3 : Gezegen bilgi paneli*

Quiz sayfasında canvas içerisine bir panel eklendi bu panelin rengi temaya uygun ayarlandı, ekranın üst kısmına başlık eklendi, süslendi. Sorunun gösterileceği alan için bir panel daha eklendi bu panelin üstüne text eklendi. Soru panelinin altına ise 4 adet buton yerleştirildi. Butonlara gölge vermek için add component kısmından shadow eklendi x=10 y=-10 olacak şekilde. Şıkların altına ise quiz bitti ve sıradaki soru şeklinde 2 buton daha eklendi. Bu butonların şekli için hazır asset kullanıldı.Yanlış bilinen soruda gösterilmek üzere ek bir panel daha yapıldı bu panelin şekli için de asset kullanıldı. Hoş bir görüntü vermek için de kullanılan asset klasöründen yıldız efekti veren bir asset kullanıldı.


*Şekil 4 : Quiz*




*Şekil 5 : Quiz açıklama*

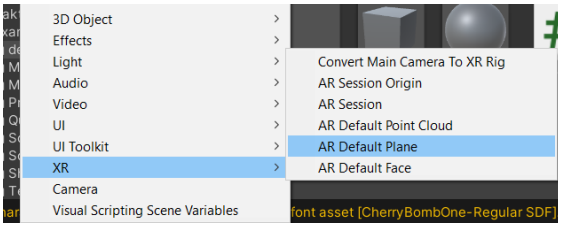
## 6. Kodlama

Proje Unity oyun motoruyla geliştirildi. Bu geliştirme süresi boyunca oluşturulan kodlar Unity kütüphane desteğiyle birlikte C# dili kullanılarak yazılmıştır.

### 6.1 Yer algılama

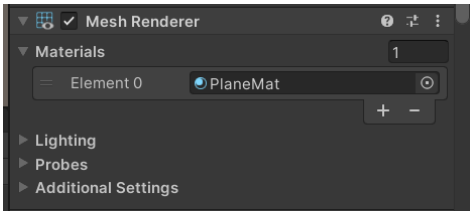
Yer algılama için sahnenin AR Session Origin bölümünde add component seçeneğine tıklayarak AR Plane Manager, AR Point Cloud, AR Raycast Manager ve AR Anchor Manager ögeleri eklenir.

Projenin sol kısmındaki menüde sağ tık yaparak XR -> AR Default Plane ve AR Default Point Cloud eklenir. Bu eklenenler assetts kısmında bir dosya oluşturularak oraya atılıp ilk eklenen yerden silinir.




*Şekil 6: AR Default Plane ekleme*

Eklenen plane'in materyali Mesh Renderer kısmından değiştirilebilir. Bu projede PlaneMat kullanıldı.




*Şekil 7: Plane materyali değiştirme*

Ayrıca AR Session Origin'e Plane Manager isminde de bir script eklenir. Bu scriptte PlanesFound adında bir method oluşturuldu. Bu method yer algılandığında eğer 3D model henüz yerleştirilmemişse algılanan bir düzleme dokunma ile 3D modeli yerleştirmemizi sağlar.

```
private void PlanesFound(ARPlanesChangedEventArgs planeData)
{
    if (planeData.added != null && planeData.added.Count > 0)
    {

        planes.AddRange(planeData.added);
    }
    foreach(var plane in planes)
    {
        if (model3DPlaced == null )
        {
            if (TryGetTouchPosition(out Vector2 touchPosition))
            {
                List<ARRaycastHit> hits = new List<ARRaycastHit>();
                if (m_RaycastManager.Raycast(touchPosition, hits, TrackableType.PlaneWithinPolygon))
                {
                    ARRaycastHit hit = hits[0];
                    Pose hitPose = hit.pose;
                    Vector3 newPosition = hitPose.position + new Vector3(10f, 1f, 10f);
                    model3DPlaced = Instantiate(model3DPrefab, newPosition, hitPose.rotation);

                    float scaleFactor = 1.5f;
                    model3DPlaced.transform.localScale = new Vector3(scaleFactor, scaleFactor, scaleFactor);

                    fixedPosition = newPosition;
                    StopPlaneDetection();
                }
            }
        }
        else
        {
            model3DPlaced.transform.position = fixedPosition;
        }
```

*Şekil 8: Cismi algılanan konuma yerleştirme*

StopPlaneDetection adındaki bir method ile de 3D model zemine yerleştirildikten sonra yer algılama kapatılır böylece model yerleştirildikten sonra yer algılamaya devam edilmez. TryGetTouchPosition methoduyla da dokunulan yerin konumu alınır.

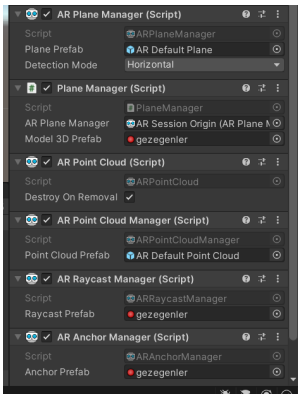
```
1 reference
public void StopPlaneDetection()
{
    aRPlaneManager.requestedDetectionMode = PlaneDetectionMode.None;
    foreach (var plane in planes)
    {
        plane.gameObject.SetActive(false);
    }
}

1 reference
bool TryGetTouchPosition(out Vector2 touchPosition)
{
    if (Input.touchCount > 0)
    {
        touchPosition = Input.GetTouch(0).position;
        return true;
    }

    touchPosition = default;
    return false;
}
```

*Şekil 9: Yer algılamayı durdurma*

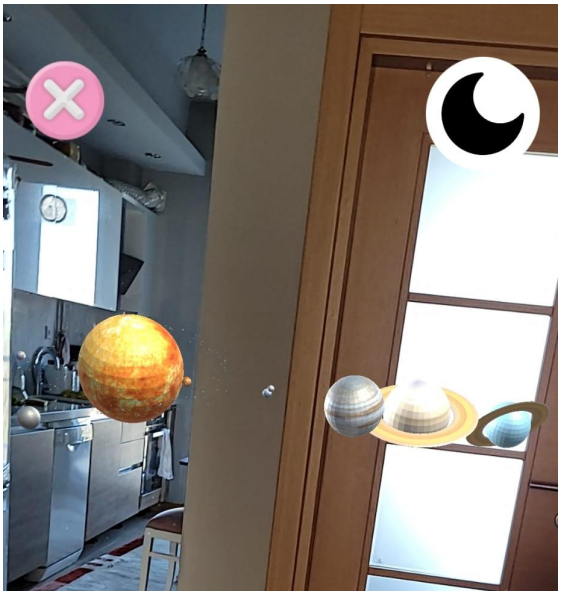
AR Plane Manager'ın prefab kısmına önceden eklediğimiz AR Default Plane yerleştirilir. AR Point Cloud Manager'ın prefabına da AR Default Point Cloud eklenir. AR Raycast Manager ve AR Anchor Manager prefab bölümüne 3D gezegen modeli yerleştirilir. Scriptin plane manager bölümüne AR Session Origin, model3D Prefab kısmına ise gezegen modeli yerleştirildikten sonra yer algılama ayarlaması tamamlanmış olur.




*Şekil 10: Yer algılama için AR Session Origin*

## 6.2 Gezegenler

Gezegenler sahnesi, kullanıcıların bir simulasyon deneyimi yaşayacağı ana sahnedir. Bu sahnede, güneş sistemimizin üç boyutlu bir modeli kullanılmaktadır. Bu model aracılığıyla kullanıcılar, gezegenlerin gerçek zamanlı hareketlerini gözlemleyebilirler. Aynı zamanda, gezegenlere tıkladıklarında detaylı bilgilere erişme imkanına sahiptirler. Güneş sistemi modeli, kullanıcılara yakınlaştırma, döndürme ve büyütme gibi interaktif eylemlerle güneş sistemini daha yakından inceleme olanağı sunmaktadır. Bu sayede, kullanıcılar gezegenler arasındaki ilişkileri ve her bir gezegenin özelliklerini keşfederken etkileşimli bir öğrenme deneyimi yaşayabilirler.



*Şekil 11: Güneş sisteminin simülasyondan görünüşü*

### 6.2.A Gezegen Hareketleri

Sahnede bulunan gezegenlerin iki çeşit hareketi var. Bunlardan ilki, gezegenin kendi etrafında dönme hareketidir. Bu dönme hareketini kontrol etmek için 'PlanetRotation' adında bir kod dosyası oluşturuldu. Bu kod dosyasında, gezegenin dönme açılarını saklamak için 'Vector3' sınıfından bir nesne, dönme hızını belirtmek için kayan noktalı bir sayı ve gezegenin saat yönünde mi yoksa saat yönünün tersine mi döneceğini belirleyen bir bool değeri vardır. Bu değerler 'public' olarak tanımlıdır, bu da demektir ki Unity üzerinde bu kod dosyasına sahip öğeler bu bilgilere kolayca erişilebilir ve düzenleyebilir. Bu sayede, her gezegen için ayrı bir kod dosyası oluşturmak zorunda kalmadan, gezegenlerin davranışlarını Unity arayüzü üzerinden ayarlamak mümkündür.

Kodun devamında, Unity kütüphanesinde bulunan 'Update' metodu var. 'Update' metodu, her framede otomatik olarak Unity tarafından çağrılan bir metottur. Bu metodun içinde, gezegenlerin dönmesini sağlamak için gerekli olan kodlar bulunuyor. Bu kod satırları sayesinde gezegenlerimiz belirtilen hız, açı ve yönde dönmektedir.

```
ic class PlanetRotation : MonoBehaviour

ublic Vector3 rotationAxis = Vector3.up;
ublic float rotationSpeed = 10f;
ublic bool rotateClockwise = true;

Unity İletisi | 0 başvuru
oid Update()
{
    float currentRotationSpeed = rotateClockwise ? rotationSpeed : -rotationSpeed;
    transform.Rotate(rotationAxis, currentRotationSpeed * Time.deltaTime);
}
```

*Şekil: 12 - Planet Rotation kod dosyası*

İkinci hareket, gezegenlerin bir yörüngede dönmesidir. Bu hareketin uygulanması için "Orbit" adında bir kod dosyası oluşturuldu. Kod dosyası, dünyanın etrafında dönen ay ve güneşin etrafında dönen gezegenler gibi sistem içinde dönen objeleri kontrol etmek amacıyla kullanılır. "Orbit" dosyası, yörüngesinde dönen objenin konumunu belirlemek için "Transform" sınıfından bir nesne, yörünge mesafesini belirleyen bir kayan noktalı sayı değeri, yörüngede bir saniyede kat edilen derece miktarını belirten bir kayan noktalı sayı değeri ve y ekseni etrafında dönme hızını belirleyen bir "Vector3" nesnesi içerir.

'Orbit' kod dosyamızda bulunan Update metodunda öncelikli olarak Transform sınıfından üretilen nesnenin boş olma durumu kontrol edilir. Eğer boş değilse OrbitAround ve SelfRotate adında iki metot çalışır. OrbitAround metodu gezegenlerin belirtilen Transform, saniyelik dönme açısı ve mesafeye göre yörüngesel bir dönme hareketi yapılmasını sağlar. SelfRotate metodu ise gezegenin döneceği yörüngede açısal bir dönme eylemi gerçekleştirmesini sağlar.

```
public class Orbit : MonoBehaviour
{
    public Transform target;
    public float orbitDistance = 5.0f;
    public float orbitDegreesPerSec = 90.0f;
    public Vector3 selfRotateSpeed = new Vector3(0f, 30f, 0f);

    Unity İletisi | 0 başvuru
    void Update()
    {
        if (target != null)
        {
            OrbitAround();
            SelfRotate();
        }
    }

    1 başvuru
    void OrbitAround()
    {
        transform.RotateAround(target.position, Vector3.up, orbitDegreesPerSec * Time.deltaTime);
        Vector3 desiredPosition = (transform.position - target.position).normalized * orbitDistance + target.position;
        transform.position = desiredPosition;
    }
    1 başvuru
    void SelfRotate()
    {
        transform.Rotate(selfRotateSpeed * Time.deltaTime);
    }
```

*Şekil: 13- Orbit kod dosyası*

### 6.2.B Etkileşimler

Gezegenler ile etkileşime girmek, üstlerine tıklamak, yaklaştırmak ve uzaklaştırmak, hareket ettirmek gibi unsurlar kullanıcıların deneyimini zenginleştiren eylemlerdir. Etkileşimler sayesinde öğrenme faktörü artmış ve daha eğlenceli bir deneyimde sunulmuş olur. Bunun için ilk olarak gezegenlerin üzerlerine tıklandığı zaman bir bilgi paneli açılmasını sağlandı (*Şekil : Gezegen bilgi paneli)*. Bu panellerin açılması ve dokunma eyleminin gerçekleşmesi için "ClickObject" adında bir kod dosyası oluşturuldu. Bu kod dosyası içerisinde GetClickedObject, isPointerOverUIObject, OpenInfo ve CloseInfo metotları bulunuyor.

ClickObject kod dosyasında, uygulama üzerindeki etkileşimi yönetmek adına temel bir kontrol Update metodu üzerinden gerçekleştirilir. Kullanıcı ekrana dokunduğunda, bu durum kontrol edilir ve dokunmanın gerçekleşip gerçekleşmediği belirlenir.Dokunma tespit edildiğinde, dokunulan noktaya bir raycast gönderilir. Raycast, bu işlem sırasında belirlenen nesneye ulaştığında, yani dokunulan nokta üzerinde belirlediğimiz bir obje varsa, bir kontrol başlatılır. Öncelikle, bilgi panelinin açık olup olmadığı kontrol

edilir. Eğer bilgi paneli açık değilse ve raycast ile tespit edilen obje belirlediğimiz nesneyle aynıysa, OpenInfo metodunu kullanarak bilgi panelini açarız. Bu sayede, kullanıcılar dokundukları nesnenin detaylı bilgilerini içeren bir bilgi paneli ile etkileşimde bulunabilirler.

```
if (Input.touchCount > 0)
{
    Touch touch = Input.GetTouch(0);

    if (touch.phase == TouchPhase.Began)
    {
        Ray ray = Camera.main.ScreenPointToRay(touch.position);
        RaycastHit hehe;

        if (Physics.Raycast(ray, out hehe, 100, targetLayer))
        {
            cube = hehe.transform.gameObject;
            Debug.Log(hehe.transform.gameObject.name);
        }

        if (cube == GetClickedObject(out RaycastHit hit))
        {
            if (!isInfoOpen)
            {
                OpenInfo();
                isInfoOpen = true;
            }
            else
            {
                isInfoOpen = false;
            }

        }
    }
    else if (touch.phase == TouchPhase.Ended)
    {
        if(!isInfoOpen)
        CloseInfo();
    }
}
```

*Şekil: 14 - Click Object kodunun bir kısmı*

Açılan bilgi panelinde, kullanıcılar için küçük bir uygulama bulunmaktadır. Bu uygulama, incelediğimiz gezegende dünya üzerindeki kütlenin karşılığını öğrenmemizi sağlar. Bu amaçla 'Kilo' adında bir kod dosyası oluşturuldu. Kod dosyası içerisinde bir bilgi giriş kutusu, bir yazı objesi ve kayan noktalı sayı türünde bir katsayı değeri tanımlandı.

Kilo kod dosyası, her bir gezegen paneline entegre edilerek kullanıcı arayüzüne bir bileşen olarak dahil edildi. Bu dosya, gezegenlerin panellerinden seçilen özel değişkenlerle ilişkilendirilmiş katsayı değerlerini içerir.

Kullanıcı, herhangi bir gezegen panelinde yer alan giriş kutusuna bir kütle değeri girdiğinde, Kilo kod dosyası bu değeri alır ve belirtilen katsayılarla çarpıp sonucu hesaplar. Sonuç olarak, kullanıcı girdiği kütle bilgisine karşılık gelen değeri anında öğrenir. Bu uygulama, gezegenler arası kütlenin karşılaştırılmasını kolaylaştırarak kullanıcılara interaktif bir öğrenme deneyimi sunar.

Kilo kod dosyası bileşen olarak her bir gezegenlerin panellerine verildi. Tanımlanan değişkenler gezegenlerin panellerinden seçildi ve katsayı değerleri girildi. Sonuç olarak kullanıcı bir gezegenin paneli içinde kütle bilgisini girdiği zaman karşılığını öğrenir.

```
public class Kilo : MonoBehaviour
{
    public TMP_InputField kiloInput;
    public TextMeshProUGUI kiloOutput;
    public float katsayı;

    0 Unity İletisi | 0 başvuru
    private void Update()
    {
        if(!string.IsNullOrEmpty(kiloInput.text))
        {
            float x= ((Convert.ToInt32(kiloInput.text) / 9.81f) * katsayı);
            x = Mathf.Round(x);
            kiloOutput.text = x.ToString();
        }
    }
}
```

*Şekil: 15 -Kilo kod dosyası*

Gezegenlerin boyutlarını kontrol etmek ve onları daha iyi gözlemlemek amacıyla sistem hareket ettirme özelliği eklenmiştir. Bu işlevleri gerçekleştirmek için Unity Asset Store'da bulunan "Lean Touch" eklentisi kullanılmıştır.

"Lean Touch", Unity oyun motoru için geliştirilmiş bir dokunmatik ve hareket kontrol eklentisidir. Bu eklenti, kullanıcıların dokunmatik ekranlar veya fare kullanarak oyun veya uygulamalarda dokunmatik hareketler, kaydırma, yakınlaştırma gibi işlevleri kolayca eklemelerine veya entegre etmelerine olanak tanır.

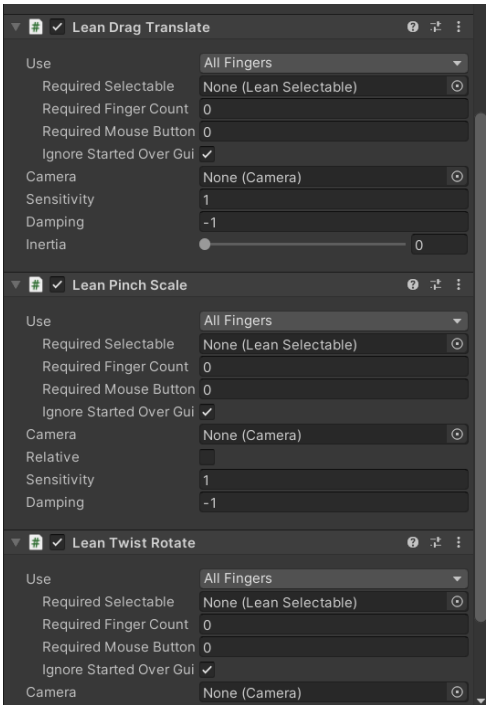
Temel olarak, kullanıcı arayüzü deneyimini geliştirmek, nesneleri yakınlaştırmak, uzaklaştırmak, döndürmek, sürüklemek gibi dokunmatik ekran hareketleriyle etkileşimi sağlamak için tasarlanmıştır. Oyun veya uygulama geliştiricileri, Lean Touch kullanarak dokunmatik ekranlarda kullanıcı etkileşimlerini daha rahat yönetebilir ve geliştirebilir.

Genellikle oyun veya uygulama geliştirme sürecinde, kullanıcı arayüzü (UI) öğelerinin dokunmatik etkileşimlerini optimize etmek veya oyuncuların oyun dünyasıyla etkileşimini artırmak için Lean Touch eklentisi kullanılır.

Proje içerisinde Lean Touch'u kullanmak için ilk adım olarak Lean Touch objesi oluşturuldu. Bu obje, dokunmatik ekran etkileşimlerini yönetmek ve kontrol etmek için temel bir yapı sağlar. Daha sonra, bu Lean Touch objesi içerisinde TouchSimulator adı verilen bir özellik aktive edildi.

Gezegenlerin hareketlerini sağlamak için ise LeanDragTranslate, LeanPinchScale ve LeanTwistRotate gibi bileşenler eklendi. LeanDragTranslate, gezegenlerin sürüklenerek hareket ettirilmesini sağlarken, LeanPinchScale kullanıcıların yakınlaştırma ve uzaklaştırma hareketlerini gerçekleştirmesine olanak tanır. LeanTwistRotate bileşeni ise gezegenlerin döndürülmesini mümkün kılar.

Bu bileşenler sayesinde, kullanıcılar güneş sistemi modeli üzerinde dokunmatik ekran etkileşimleriyle etkileşime girebilirler. Örneğin, parmaklarını birleştirerek gezegenleri yakınlaştırabilir, gezegenleri sürükleyebilir veya döndürebilirler. Bu şekilde, kullanıcılar güneş sistemi modelini keşfederken doğal ve sezgisel bir etkileşim deneyimi yaşayabilirler




*Şekil: 16 - Gezegenlere verilen LeanTouch bileşenleri*

Simülasyon içerisine, uzayda bulunuyormuş hissi vermek amacıyla güneş sisteminin görünümünü daha karanlık göstermek için bir gece filtresi eklendi. Bu filtre, bir butona bağlandı ve kullanıcılar butona bastıklarında gece filtresini açıp kapatabiliyorlar.

Gece filtresini kontrol etmek için "NightFilter" adında bir kod dosyası oluşturuldu. Bu kod dosyası, butona basıldığında filtreyi etkinleştirir ve bir kez daha basıldığında devre dışı bırakır. Bu sayede, kullanıcıların gözlerinin ışıktan dolayı yorulmasının önüne geçilirken aynı zamanda daha gerçekçi bir deneyim elde edilmesi hedefleniyor.

```
Unity Betiği (2 varlık başvurusu) | – başvurular
public class NightFilter : MonoBehaviour
{
    public GameObject[] filterButtons;
    public GameObject filter;
    public Camera arCam;

    Unity İletisi | – başvurular
    private void Start()
    {
        filterButtons[1].SetActive(false);
        filter.SetActive(false);
    }

    – başvurular
    public void FilterOn()
    {

        filterButtons[0].SetActive(false);
        filterButtons[1].SetActive(true);
        filter.SetActive(true);
    }

    – başvurular
    public void FilterOff()
    {

        filterButtons[0].SetActive(true);
        filterButtons[1].SetActive(false);
        filter.SetActive(false);
    }
}
```

*Şekil: 17 - NightFilter kod dosyası*

## 6.3 Quiz

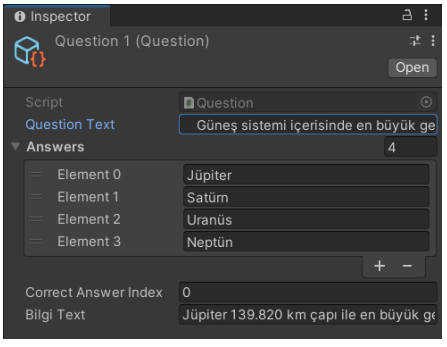
30 soru ve her sorunun 4 şıkka sahip olduğu simülasyonun öğreticiliğini test eden küçük bir quiz sahnesi bulunmaktadır. Bu sahnedeki verileri tutabilmek için Scriptable Object yapısından yararlanılmıştır. Scriptable object, her türlü veriyi dosya olarak tutarak üzerinde işlem yapılmasına yarayan bir yapıdır. Unity içinden ilk olarak bir kod dosyası oluşturuldu. Bu dosya, kod ile scriptable object yapısına dönüştürülmüştür. Şekil- da verilen ekran görüntüsünde görüldüğü üzere sınıf ismi üstüne yazılan kod, dosya adını ve obje oluşturulduğunda ekranda nasıl görüneceğini göstermektedir ve karşısına yazılan kod ile yapı Scriptable Object sınıfına bağlanmıştır. Bu kod içerisine sorunun, şıkların, sorunun yanlış cevaplanmasında ekranda belirecek panelde bulunan sorunun doğru cevabı hakkında bilgi veren string yapılar ve hangi şıkkın doğru olduğunu tutan int yapısı tanımlanmıştır. Bu kod tamamlandıktan sonra editör içinden bu kod dosyasını kullanarak Şekil- 'de görüldüğü üzere Question yapısında scriptable objectler oluşturuldu. Şekil-19 de görüldüğü gibi soru kayıtları istenilen şekilde bir dosya içinde tutulabilmektedir.

```
[CreateAssetMenu(fileName ="New Question",menuName ="Question")]
public class Question : ScriptableObject
{
    public string questionText;
    public string[] answers;
    public int correctAnswerIndex;
    public string bilgiText;

}
```

*Şekil : 18 - Question kod dosyası*


*Şekil: 19 - Soru Scriptable objectleri*




*Şekil: 20 - Soru Örneği*

Yukarıda yazılan adımlar ile sorular hazırlanmıştır. Bu soruları sahne içinde kullanabilmek için ayrı bir kod dosyası oluşturulmuştur. Bu dosyada bir dizi oluşturulmuş ve diziye, hazırlanan soruları tutan scriptable objectler atanmıştır. Hangi soruda olunduğu PlayerPrefs yapısı ile tutulmaktadır. Bu yapıya

kaydedilen bilgiler her zaman hafızada kalmaktadır. Editör veya telefondaki uygulama kapatılsa dahi veriler kaybolmamaktadır. Editör içinden soru ve butonların text yapılarını tutacak tasarımlar hazırlanmış ve bunlar kod dosyasına atanmıştır. Soru yanlış bilindiğinde sorunun doğru cevabı hakkında bilgi paneli aktif olmaktadır. Bir sonraki soruya geçebilmek için bulunulan soru cevaplanmalı ve next butonuna tıklanmalıdır. Cevaba bağlı olarak cevaplanan şık doğru ise buton rengi yeşil, yanlış ise kırmızı renge dönmektedir. Quiz içerisindeki 30 soru cevaplandıktan sonra quiz bittiğini gösteren panel aktif olmakta ve ana menüye gönderen buton bulunmaktadır. Her quiz tamamlandıktan sonra tekrar oynanmak istenirse sorular en baştan sorulmaktadır. Şekil- ‘de soru sisteminin çalışma mantığını gösteren kod bulunmaktadır.

```
void SetQuestion()
{
    questionText.text = questions[currentLevel].questionText;
    aciklamaText.text = questions[currentLevel].bilgiText;
    for (int i = 0; i < buttonTexts.Length; i++)
    {
        buttonTexts[i].text = questions[currentLevel].answers[i];
    }
    correctanswerIndex = questions[currentLevel].correctAnswerIndex;
    currentLevel++;
    nextButton.interactable = false;
}
```

*Şekil: 21 - Soru oluşturma metodu*

## 7. Test Süreci

Projeyi test etmek için AR teknolojisi destekleyen bir cihaz kullanılması gerekliydi. Test etmek için, ya build alıp apk’yı telefonda çalıştırmak ya da telefonu geliştirici moduna alıp usb ile bilgisayara bağlayarak uygulamayı çalıştırmak gerekti. Projeyi geliştirme süresince pek çok defa test etmek gerekti, her seferinde apk almak çok zaman aldığından dolayı usb kullanmak daha verimli oldu.

Test süresince pek çok hatayla karşılaşıldı. Bunlardan ilki yer algılama sürecinde oldu. Yer algılama için pek çok farklı kod denendi, ayrıca cismi yerleştirdikten sonra yer algılamanın kalkması istendiği için o kısımda da başka kodlar deneyerek sorun çözüme kavuşturuldu, bu kodlara raporun yer algılama bölümünde yer verildi.

Yer algılama için ilk başta 3D cisim olarak basit bir küple çalışıldı, yer algılamanın sorunsuz çalıştığından emin olunduktan sonra küp yerine gezegenler kondu. İlk test aşamasında gezegenler görünürken güneş görünmüyordu. Kamera açısıyla ilgili bir problem olduğu düşünüldü, kamera açısı değiştirildikten sonra sorun çözüldü.

Gezegenlerin üstüne tıklayınca gezegenin daha yakından bir görüntüsüne erişmek ve görüntüyle birlikte gezegenle ilgili bilgiler vermek amaçlanmaştı bunun için yapılan ilk denemede gezegen hareket halinde ve arka plan dönüyor bir haldeydi bu iyi bir görüntü oluşturmadığı için her gezegene ayrı bir AR kamerası atandı, daha sonra bu kameranın görüntüsü ekrana ikinci bir kamera gibi eklendi, bir ekranda iki kamera üst üste iyi bir görüntü oluşturmadığı için ayrıca bir panel oluşturuldu görüntü bu panele atandı, panelde de alt kısımda gerekli bilgiler verildi. Gezegene tıklayınca bu panel açılmakta. Panelin arka planı başlangıçta siyah olarak da test edildi ama proje bir AR projesi olduğu için kamera görüntüsünün daha iyi olacağı düşünüldü. Ayrıca bazı gezegenler kamera ayarlaması yapılırken tam yapılamadığı için yarım görünüyordu, bu ayarlamalar tekrar yapılınca sorun düzeldi.

## 8- Sonuç

Belirlenen problem ve amaç doğrultusunda Unity üzerinde Artırılmış Gerçeklik tabanlı bir uygulama geliştirilmiştir. Bu uygulama ilkokul çağındaki çocuklara gezegenleri eğlenceli bir şekilde öğretmekte, quiz sayesinde de öğrenilen bilgileri pekiştirmede yardımcı olmaktadır.

Uygulama çocuklara yönelik kolay kullanılabilir kullanıcı dostu bir arayüz ile tasarlanmıştır.

Uygulamada simülasyon sahnesine giriş yapıldığında aygılanan yerlerden istenilene tıklandığı zaman güneş sistemi 3 boyutlu bir şekilde bu alana konmakta ve AR kamerada görünür hale gelmektedir. Bu 3 boyutlu sistemde istenen gezegene tıklandığında ise gezegenin görüntüsü ve gezegenle ilgili bilgilerin bulunduğu bir panel kullanıcıya sunulmaktadır. Bu sunulan panelde ayrıca dünyadaki kilo girilerek ilgili gezegende kaç kilo olunduğu da öğrenilebilmektedir. Ekranın sağ üst köşesinde bulunan butonlar kullanılarak gece ve gündüz moduna geçiş sağlanabilmektedir.

Quiz sahnesine girildiğinde ise 30 adet quiz sorusu bulunmaktadır. Sorular doğru cevaplandığı takdirde sıradaki soruya geçilmekte, yanlış cevap verildiğine ise doğru cevabın şıkkı yeşil yanarak kullanıcıya bildirilmekte, ekrana gelen küçük popup içerisinde de sorunun doğru cevabının açıklaması verilmektedir. Quiz yarım bırakılıp çıkılırsa tekrar girildiğinde kalınan sorudan devam edilebilmekte olup quiz bittiğinde quize tekrar başlanabilmektedir.

## Referanslar


- https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.1/manual/index.html [04.11.2023]
- KIRIKKAYA, Esma BULUŞ, and Melek ŞENTÜRK. "Güneş sistemi ve ötesi ünitesinde artırılmış gerçeklik teknolojisi kullanılmasının öğrenci akademik başarısına etkisi." *Kastamonu Eğitim Dergisi* 26.1 (2018): 181-189.
- Özdemir, Muzaffer. "Artırılmış gerçeklik teknolojisi ile öğrenmeye yönelik deneysel çalışmalar: sistematik bir inceleme." *Mersin Üniversitesi Eğitim Fakültesi Dergisi* 13.2 (2017): 609-632.

- https://science.nasa.gov/solar-system/ [01.11.2023]
- https://carloswilkes.com/Documentation/LeanTouch [05.11.2023]
- https://developers.google.com/ar/develop/unity-arf/getting-started-ar-foundation?hl=tr [04.11.2023]