Kocaeli Üniversitesi

Teknoloji Fakültesi

Bilişim Sistemleri Mühendisliği

Yazılım Geliştirme Laboratuvarı

Muhammet İkbal Çakır
Bilişim Sistemleri Mühendisliği
Teknoloji Fakültesi
Kocaeli, Türkiye
mamicakir30@gmail.com

Ayşenur Elibüyük
Bilişim Sistemleri Mühendisliği
Teknoloji Fakültesi
Kocaeli, Türkiye
ayselibuyuk171@gmail.com

Mustafa Toprak
Bilişim Sistemleri Mühendisliği
Teknoloji Fakültesi
Kocaeli, Türkiye
musttoprakk@gmail.com

***Özet*— Bizim projemiz, Unity ve artırılmış gerçeklik (AR) kullanılarak geliştirilen bir uygulamadan oluşmaktadır. Renkli kutular ve bu kutulara sürüklenerek bırakılan nesnelerin eşleştirilmesini temel alan bir uygulamadır. Projenin temel amacı, kullanıcıların AR ortamında etkileşimli bir şekilde renkli nesnelerin doğru kutulara yerleştirilmesini sağlamaktır.**

Anahtar Kelimeler—Unity, artırılmış gerçeklik (AR), eğitim teknolojisi, renkler

***Amaç ve Temel Hedef*— Hedef kitlemiz olan çocuklara renkleri eğlenceli bir şekilde öğretebilmek temel hedefimiz. Renkleri öğrenmek, çocukların görsel hafızalarının güçlenmesine ve yaratıcılıkların gelişmesine katkı sağlayacağı için hedef kitlemizi çocuklar olarak seçtik.**

I. GIRIŞ

Çağımızda teknolojinin hızla ilerlemesi, eğitim alanında da önemli değişikliklere sebep olmuştur. Bu bağlamda, çocuklara renkleri öğretme sürecini daha etkili ve eğlenceli hale getirme amacıyla geliştirdiğimiz proje, artırılmış gerçeklik (AR) ve interaktif öğrenme yöntemlerini birleştirmektedir. Çocukların, renkleri sadece kelimelerle değil, aynı zamanda görsel deneyimlerle öğrenmeleri kalıcı bilgi edinmelerine olanak sağlayacaktır. Projemizde kullandığımız AR, gerçek dünyayı dijital içerikle birleştirerek oldukça güzel ve öğretici bir ortam sunar.

II. UNITY ORTAMININ KURULMASI

Proje geliştirme sürecine başlamadan önce, Unity geliştirme ortamının kurulumu önemli bir adımdır. Aşağıda, projemizin geliştirilmesi için Unity'nin nasıl kurulacağından ve projemize artırılmış gerçeklik özelliklerini eklemek için gerekli olan AR session tools'larının eklenmesinden bahsedelim;

A. Unity Hub İnderme

Unity'nin resmi web sitesinden bir hesap oluşturduk. Bu sayede Unity Asset Store'a erişim sağlayabildik ve projemize ek öğeler ekleyebildik. Aynı şekilde Unity resmi web sitesinden Unity Hub'I indirip kurduk.

B. Yeni Proje Oluşturma

Unity Hub'ı açtıktan sonra, kullanmak istediğimiz Unity sürümünü seçtik. Bu adım projemizin isterlerini karşılayabilmek için önemlidir. Daha sonra Unity Hub üzerinden "New Project" (Yeni Proje) seçeneğini kullanarak yeni bir proje oluşturduk.

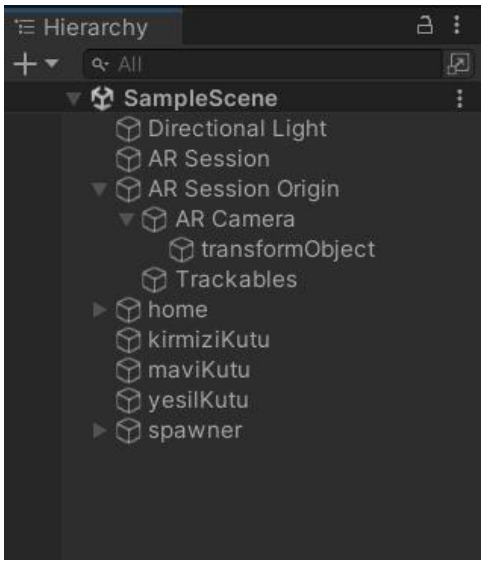
C. AR Paketini Ekleme

Unity Asset Store'dan AR Foundation paketini bulup ve projemize ekledik. Bu, artırılmış gerçeklik özelliklerini destekleyen temel bir pakettir.

D. AR Değerlerine Göre Şekillendirme

Hierarchy penceresinde, AR özelliklerini kullanmak istediğimiz sahne üzerinde boş bir alan seçip "GameObject" menüsünden "XR" altında "AR Session" ve "AR Session Origin" ekledik. Bu nesneler, AR session yönetimini ve sahne içinde AR içeriğini yerleştirmeyi sağlar. "GameObject"

menüsünden “XR” altında “AR Camera” ekleyerek kameramızı AR ortamında kullanmaya hazır hale getirdik.

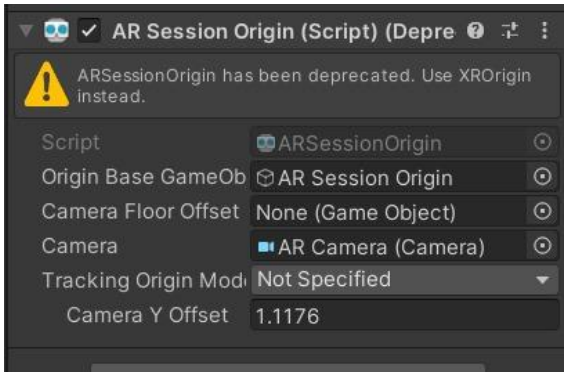


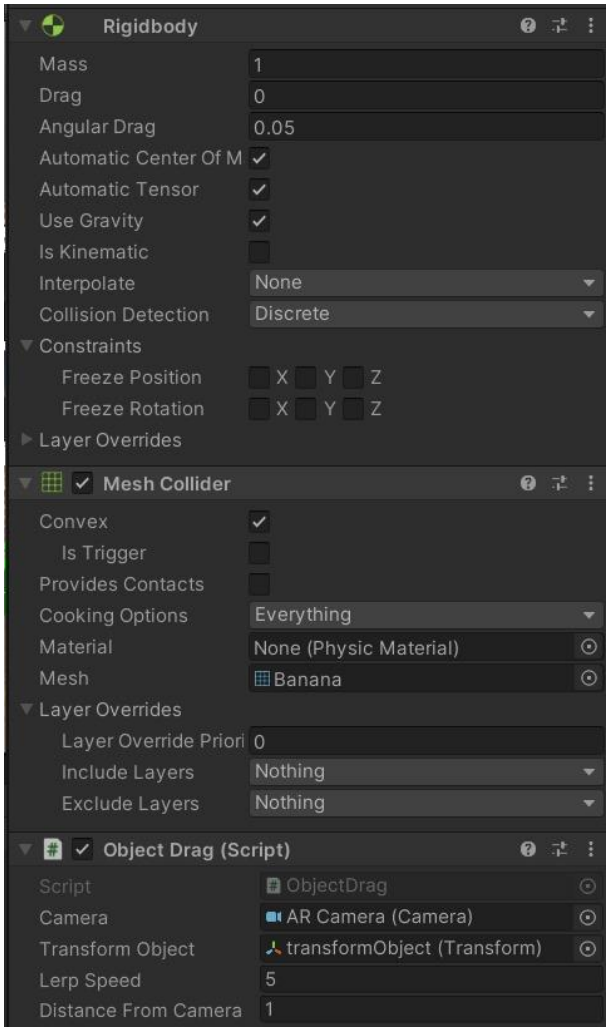
Bu adımlar dikkate alınarak Unity ortamının doğru bir şekilde kurulması, projemizin sağlıklı çalışmasını sağlar.

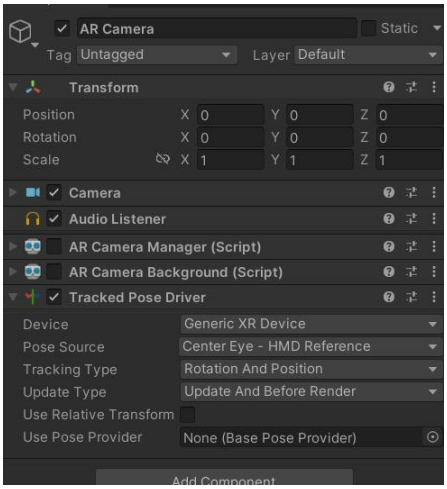
AR Session ve AR Session Origin, Unity içinde artırılmış gerçeklik (AR) projelerini geliştirmek için önemli bileşenlerdir. Detaylı anlatacak olursak;

AR Session: Artırılmış gerçeklik teknolojilerini yöneten temel bir bileşendir. Bu bileşen, ARCore veya ARKit gibi altta yatan AR platformlarına bağlanır ve cihazın konumunu, hareketini ve çevresini takip eder. AR Session, AR uygulamamızın başlatılmasını, durdurulmasını ve ölçeklendirilmesini yönetir. AR uygulamamız başladığında AR Session, AR özelliklerini başlatır ve cihazın çevresini kamera ile takip etmeye başlar. Uygulama durduğunda veya kullanıcı çıkış yaptığında ise AR Session, AR özelliklerini durdurur. Kısaca AR Session sanal içeriği gerçek dünyaya yerleştirmek için temel bilgileri sağlar.

AR Session Origin: Sahnede artırılmış gerçeklik içeriğini yerleştirmek ve cihazın konumunu düzenlemek için kullanılan bir bileşendir. Genellikle kameralardan elde edilen bilgileri temel alarak, sanal objeleri dünyanın gerçek boyutlarına ve konumlarına yerleştirir. Sahnede sanal objeleri düzenlemek ve AR içeriğini yerleştirmek için kullanılır. Sanal objeler, AR Session Origin'in cihazın konumunu ve yönelimini temel alarak doğru bir şekilde yerleştirilir.

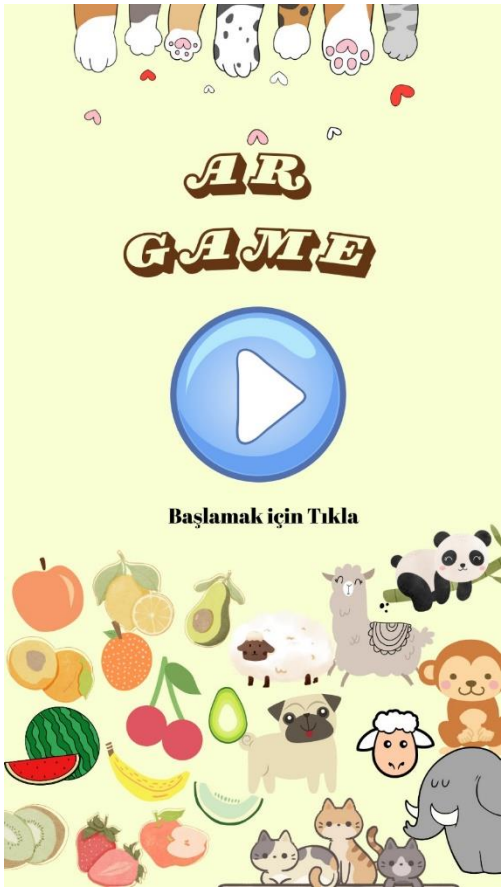






III. Oyunun Başlangıç Ekranı

Oyunun başlangıcında kullanıcıyı karşılayan ana menü tasarımımız mevcuttur. Ana menüde odak noktamız, kullanıcının oyunu başlatmasına olanak tanıyan bir "Oyna" tuşudur. Bu tuş, oyunun başlatılmasını tetikleyen ve kullanıcıyı ana oyun sahnesine yönlendiren buton işlevini görür. Ayrıca ana menümüzde bu eğitici oyunun kullanıcının dikkatini çekmesi için renkli ve şekilli bir tasarım mevcut.



IV. KOD AÇIKLAMALARI

Projenin her bir kod bloğu, projenin belirli bir yönünü ele alır. Bu kod bloklarından bahsedecek olursak;

A. *Nesne Sürükleme İşlemi*

Bu kod bir nesneyi fareyle sürüklemek için bir yöntemdir.

a) 'OnMouseUp()' Fonksiyonu: Fare tuşu bırakıldığında tetiklenir. `isDragging` değeri false olduğundan nesnenin sürüklendiğini durdurur. Nesnenin Rigidbody'sini tekrar etkinleştirir ve yerçekimini aktifleştirir.

b) 'OnMouseDrag()' Fonksiyonu: Fare sürüklendiği sürece sürekli olarak tetiklenir. Fare pozisyonunu dünya koordinatlarına dönüştürerek nesnenin konumunu belirler. Bu konumu, nesnenin mevcut konumu ile bir Lerp (lineer interpolasyon) kullanarak yumuşak bir şekilde günceller.

c) 'OnMouseDown()' Fonksiyonu: Fare tuşu tıklandığında tetiklenir. `isDragging` değeri true olduğundan nesnenin sürüklendiğini belirtir. Nesnenin Rigidbody'sini etkisizleştirir, bu sayede fiziksel etkileşimlerin sürükleme sırasında devre dışı kalmasını sağlar.

```
public class DragAndDrop : MonoBehaviour
{
    private bool isDragging = false;
    public float timeLerp = 4f;
    0 Unity iletisi | 0 başvuru
    private void OnMouseUp()
    {
        // Sürükleme bitti
        isDragging = false;
        this.GetComponent<Rigidbody>().isKinematic = false;
        transform.SetParent(GameObject.Find("spawner").transform);
    }

    0 Unity iletisi | 0 başvuru
    private void OnMouseDown()
    {
        // Dokunulan objeyi al
        isDragging = true;
        this.GetComponent<Rigidbody>().isKinematic = true;

    }

    0 Unity iletisi | 0 başvuru
    private void OnMouseDrag()
    {
        // Objeyi sürükle
        Vector3 touch = Input.mousePosition;
        if (isDragging)
        {
            transform.SetParent(GameObject.Find("transformObject").transform);
            Vector3 newPosition =
                Camera.main.ScreenToWorldPoint(new Vector3(touch.x, touch.y, 1f)) ;
            transform.position = Vector3.Lerp(transform.position, newPosition, timeLerp * Time.deltaTime);
        }
    }
}
```

B. *Random Obje Üreten Kod*

`RandomObjectSpawner` sınıfındaki fonksiyon belirtilen sayıda rastgele nesne oluşturur ve bunları belirtilen aralıklarda rastgele konumlarda sahnede yerleştirir.

a) `Start()` Fonksiyonu: Oyun nesnelerini rastgele bir şekilde sahnede yerleştirmek için kullanılır. `myObjects` dizisinde bulunan nesnelerinden birini rastgele seçer. Rastgele bir konum ve rastgele bir dönme açısıyla yeni bir spawn konumu oluşturur. Oluşturulan nesneyi `RandomObjectSpawner` nesnesine bağlı olarak sahne içindeki bir nesnenin alt nesnesi yapar.

```
public class RandomObjectSpawner : MonoBehaviour
{
    public GameObject[] myObjects;
    public Material[] colors;
    0 Unity iletisi | 0 başvuru
    private void Start()
    {
        for (int i = 0; i < 20; i++)
        {
            int randomObjectIndex = Random.Range(0, myObjects.Length);
            int randomColorIndex = Random.Range(0, colors.Length);
            Vector3 randomSpawnPosition = new Vector3(Random.RandomRange(-2, 4), -1, Random.RandomRange(-3, 6));
            Quaternion quaternion = new Quaternion();
            quaternion.x = Random.Range(1, 360);
            quaternion.y = Random.Range(1, 360);
            quaternion.z = Random.Range(1, 360);

            var gameObject =  Instantiate(myObjects[randomObjectIndex], randomSpawnPosition, quaternion);
            gameObject.transform.SetParent(this.transform);
            gameObject.GetComponent<MeshRenderer>().material = colors[randomColorIndex];
        }

    }
}
```

C. *Obje Kutuya Konulduğunda Olacak İşlemler*

Unity oyun motoru içinde kullanılmak üzere bir MonoBehaviour olan "PutABox" sınıfını tanımlar. Bu sınıf, bir oyun yöneticisi (GameManager) ile etkileşime geçer ve bir renk değişkeni içerir.

a) `Start` Fonksiyonu: Nesnenin başlangıcında çağrılır ve nesnenin renk bilgisini alır, ardından bu bilgiyi ekrana yazdırır.

b) `OnTriggerEnter` Fonksiyonu: Bu nesnenin bir başka nesneyle etkileşime girdiğinde tetiklenir. Eğer etkileşime giren nesne "sekil" etiketliyse, renk kontrolü yapar. Eğer renkler eşleşiyorsa, GameManager'deki `trueObject` metodunu başlatır; aksi takdirde `falseObject` metodunu başlatır.

c) `OnTriggerExit` Fonksiyonu: Bu nesnenin bir başka nesneyle etkileşimini sonlandırdığında tetiklenir. Eğer etkileşim sonlanan nesne "sekil" etiketliyse ve renkler eşleşiyorsa, GameManager'deki `puanCikar` metodunu çağırır.

```
public class PutABox : MonoBehaviour
{
    public GameManager gameManager;
    private string color;

    Unity İletisi | 0 başvuru
    private void Start()
    {
        color = gameObject.GetComponent<MeshRenderer>().materials[0].name.ToString();
        Debug.Log(color);
    }
    Unity İletisi | 0 başvuru
    private void OnTriggerEnter(Collider other)
    {

        if (other.CompareTag("sekil")) // Eğer istediğimiz obje bu tag'a sahipse
        {
            if (other.GetComponent<MeshRenderer>().materials[0].name.ToString() == color)
            {
                StartCoroutine(gameManager.trueObject());
            }
            else {
                StartCoroutine(gameManager.falseObject());
            }
        }
    }
    Unity İletisi | 0 başvuru
    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("sekil")) // Eğer istediğimiz obje bu tag'a sahipse
        {
            if (other.GetComponent<MeshRenderer>().materials[0].name.ToString() == color)
            {
                gameManager.puanCikar();
            }
        }
    }
}
```

D. Game Manager

Bir nesnenin ölçek ve görünürlük özelliklerini değiştirerek animasyonlu bir geri bildirim sağlar ve belirli bir sıra ile işlemler gerçekleştirir.

```
public IEnumerator trueObject()
{
    resultSprite.GetComponent<CanvasGroup>().alpha = 1;
    resultSprite.GetComponent<Image>().sprite = sprites[0];

    // Başlangıç ölçeği
    Vector3 startScale = resultSprite.transform.localScale;
    // Hedef ölçek
    Vector3 targetScale = spriteScale;

    float timer = 0.0f;
    float duration = 0.5f; // Değişim süresi

    while (timer < duration)
    {
        resultSprite.transform.localScale = Vector3.Lerp(startScale, targetScale, timer / duration);
        timer += Time.deltaTime;
        yield return null;
    }

    resultSprite.transform.localScale = targetScale;

    yield return new WaitForSeconds(1);

    // Tekrar animasyon: Küçültme
    startScale = targetScale;
    targetScale = Vector3.zero;
    timer = 0.0f;

    while (timer < duration)
    {
        resultSprite.transform.localScale = Vector3.Lerp(startScale, targetScale, timer / duration);
        timer += Time.deltaTime;
        yield return null;
    }

    resultSprite.transform.localScale = targetScale;

    // Diğer işlemler
    resultSprite.GetComponent<CanvasGroup>().alpha = 0;
    Debug.Log("değişti");
    puanEkle();
}
```