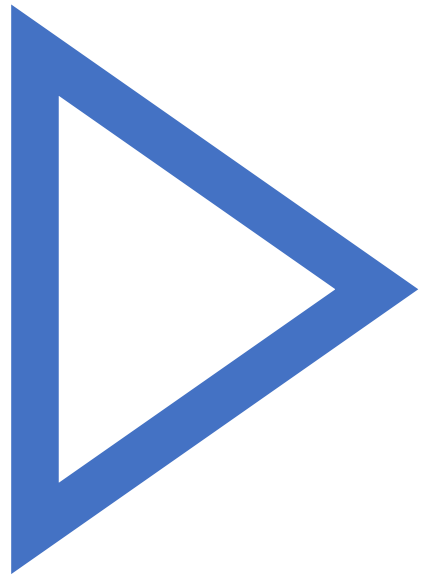
Unity

Öğr. Gör. Gözde Mihran ALTINSOY

Dinamik Obje Oluşturmak (Spawn)

- Oyun objelerinin dinamik olarak belirli bir duruma göre sahnede var olması işlemine oyun dünyasında spawn denir.
- Objelerini Spawn edebilmemiz için objeyi öncelikle Prefab haline getirmemiz gerekmektedir.



Objeyi Public Tanımlamak

- Objemize editörden erişebilmenin bir yolu objemizi Public olarak tanımlamaktır.
- Ama oyun objemizi Public olarak tanımladığımızda her yerden objemize erişilebilir. Yani bütün Script'ler bu objemize erişebilir.
- Bu durum her zaman istenen bir durum değildir.

```
public class Spawner : MonoBehaviour
{


    //public herkes tarafından
erişilebilir
    public GameObject astreoidPrefab;
}
```

Script Serialization (Script Serileştirmesi)

- Serileştirme, veri yapılarını veya nesne durumlarını Unity'nin daha sonra depolayabileceği ve yeniden yapılandırabileceği bir biçime dönüştüren otomatik işlemdir.
- Unity'nin yerleşik özelliklerinden bazıları serileştirme kullanır.
- Bunlar;
  - Kaydetme ve yükleme,
  - Inspector penceresi,
  - Instantiation (Örnekleme) ve Prefab gibi özellikler.
- Daha ayrıntılı bilgi için:
  - https://docs.unity3d.com/ScriptReference/Serializable.html
  - https://docs.unity3d.com/Manual/script-Serialization.html

SerializeField

Unity komut dosyalarınızı serileştirdiğinde, yalnızca ortak alanları serileştirir. Unity'nin özel alanlarınızı serileştirmesini istiyorsanız, bu alanlara SerializeField özniteliğini ekleyebilirsiniz.

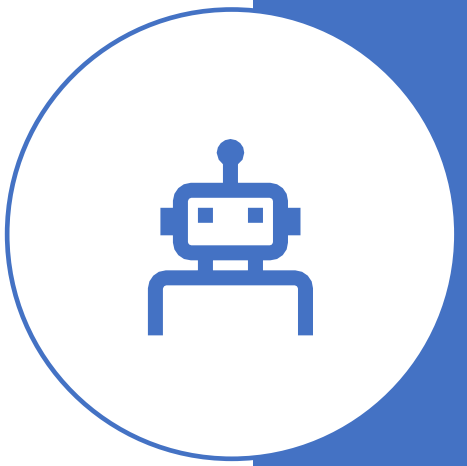
Unity tüm komut dosyası bileşenlerinizi serileştirir, yeni derlemeleri yeniden yükler ve komut dosyası bileşenlerinizi serileştirilmiş sürümlerden yeniden oluşturur. Bu serileştirme işlemi .NET'in serileştirme işlevinin yerine dahili Unity serileştirme sistemi ile yapılır.

Serileştirme Sistemi (The serialization system)

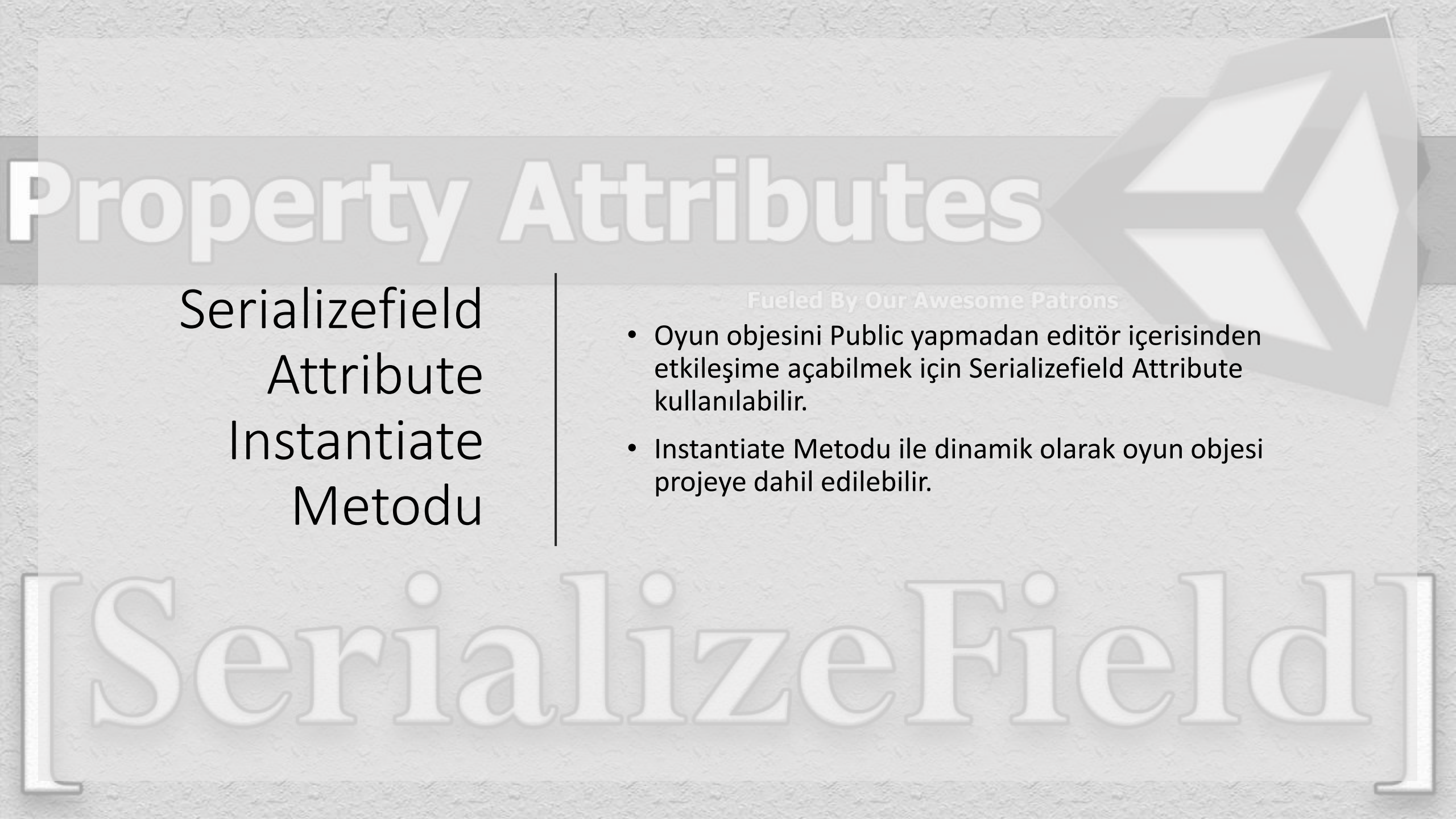
- Serileştirme sistemi aşağıdakileri yapabilir:
  - Public, statik (static) olmayan alanları serileştirebilir (serileştirilebilir tiplerde)
  - SerializeField özniteliğiyle işaretlenmiş, public olmayan, statik (static) olmayan alanları serileştirebilir.
  - Statik alanlar serileştirilemiyor.
  - Özellikler serileştirilemiyor.

Serileştirilebilir tipler (Serializable types)

- Unity, aşağıdaki türlerdeki alanları serileştirebilir:
  - UnityEngine.Object öğesinden devralınan tüm sınıflar, örneğin GameObject, Component, MonoBehaviour, Texture2D, AnimationClip.
  - int, string, float, bool gibi tüm temel veri türleri.
  - Vector2, Vector3, Vector4, Quaternion, Matrix4x4, Color, Rect, LayerMask gibi bazı yerleşik türler.
  - Serileştirilebilir tipte diziler (arrays)
  - Serileştirilebilir tipte listeler (lists)
  - Enums (Numarandırmalar)
  - Structs (Yapılar)



Property Attributes



Fueled By Our Awesome Patrons

Serializefield Attribute

Instantiate Metodu

- Oyun objesini Public yapmadan editör içerisinden etkileşime açabilmek için Serializefield Attribute kullanılabilir.
- Instantiate Metodu ile dinamik olarak oyun objesi projeye dahil edilebilir.

[SerializeField]

Attribute Tanımlama

[AttributeTürü]

ObjeTuru objeAdı;

- Ör:

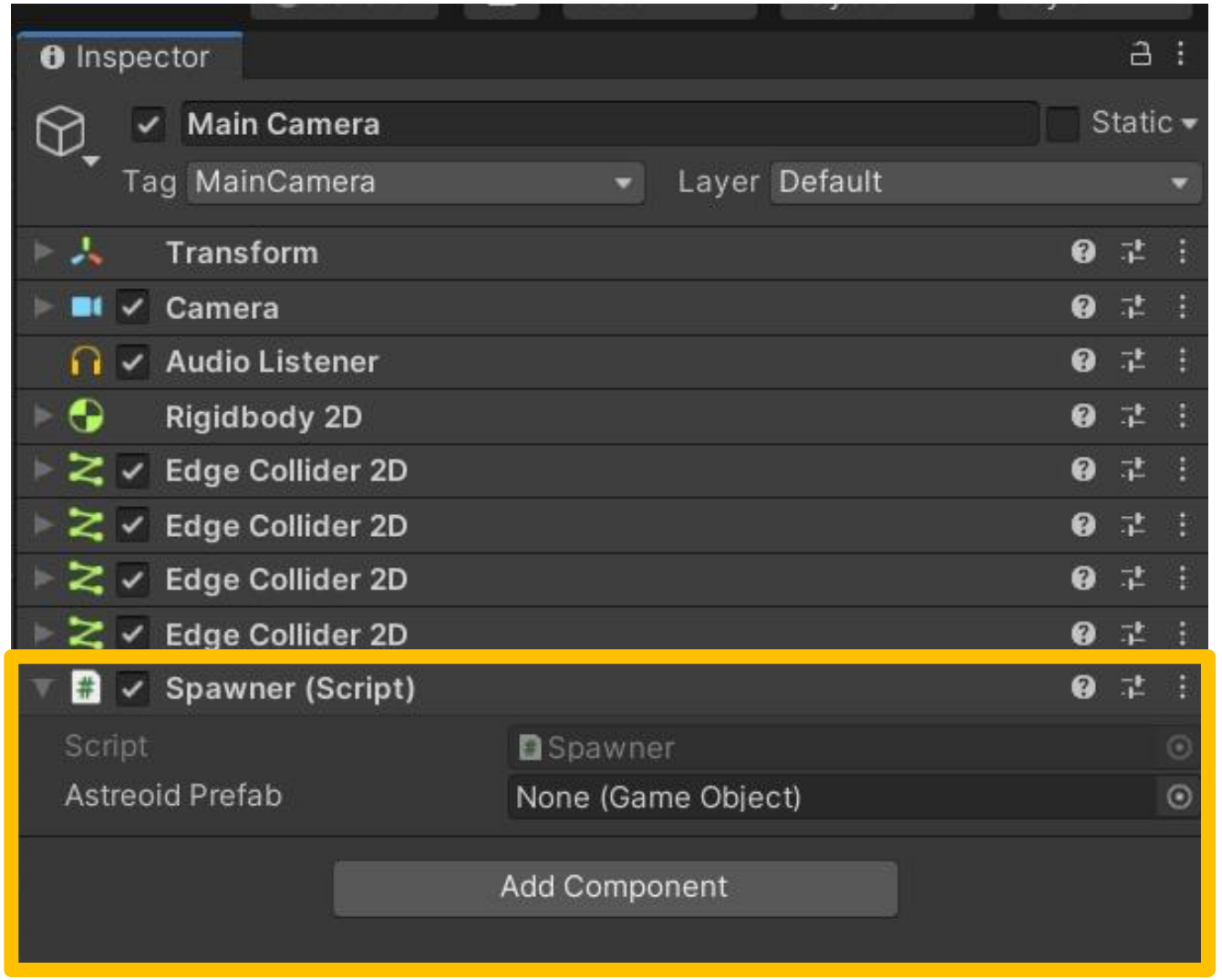
```
[SerializeField]
GameObject astreoidPrefab;
```

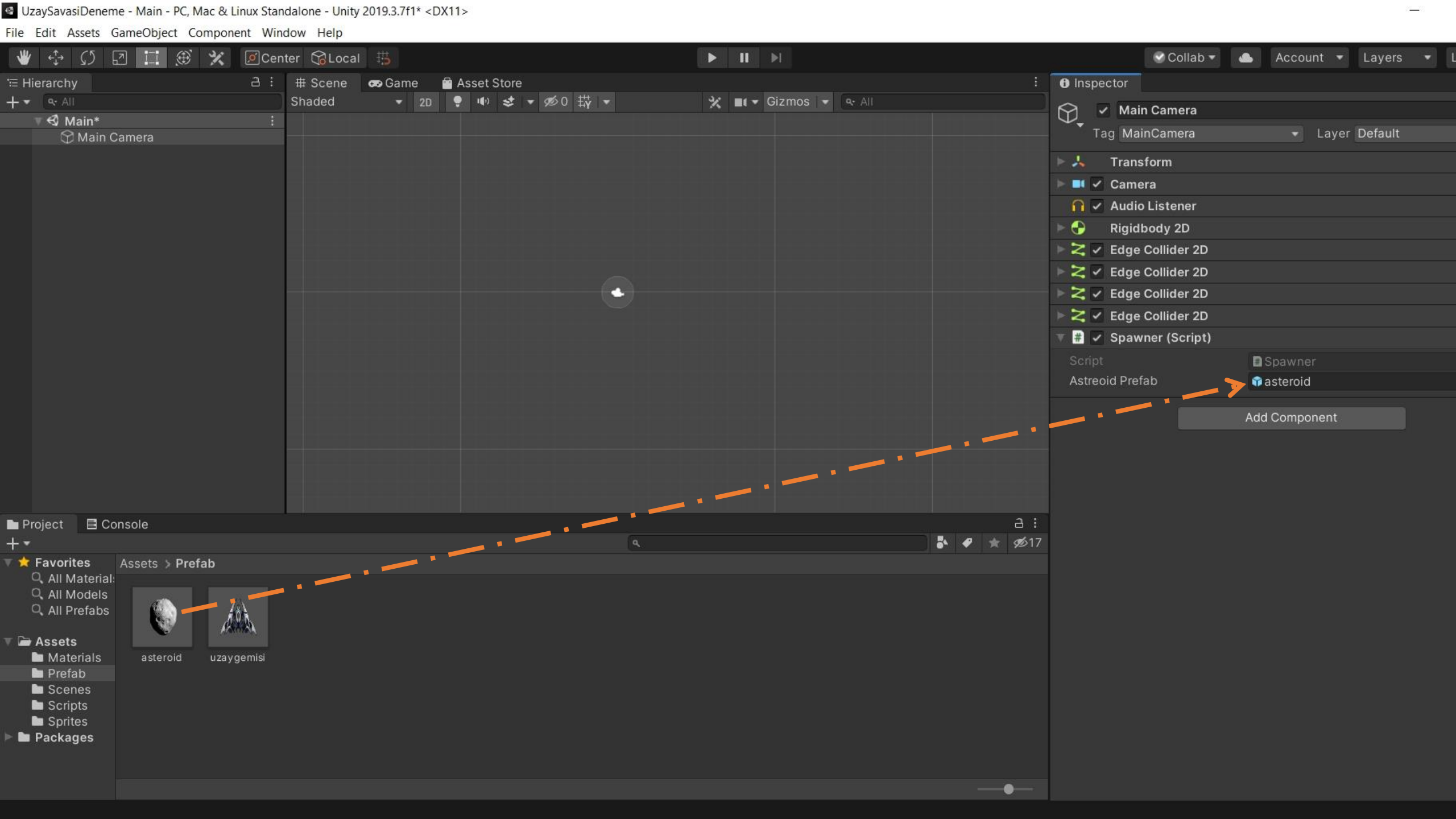
Spawner.cs

```
public class Spawner : MonoBehaviour
{
    //Serializefield editor
içerisinden müdahale edebiliriz
    [SerializeField]
    GameObject astreoidPrefab;
}
```

SerializeField ile Objeye Editör İçerisinden Müdahale Etmek Mümkün

- Oyun objemiz Public olmamasına rağmen, SerializeField ile Editör içerisindeki Inspector penceresinden objemize müdahale edebiliyoruz.




UzaySavasiDeneme - Main - PC, Mac & Linux Standalone - Unity 2019.3.7f1* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers
Hierarchy
Main*
Main Camera
Scene Game Asset Store
Shaded 2D Gizmos
Inspector
Main Camera
Tag MainCamera Layer Default
Transform
Camera
Audio Listener
Rigidbody 2D
Edge Collider 2D
Edge Collider 2D
Edge Collider 2D
Edge Collider 2D
Spawner (Script)
Script Spawner
Astreoid Prefab asteroid
Add Component
Project Console
Favorites
All Materials
All Models
All Prefabs
Assets
Materials
Prefab
Scenes
Scripts
Sprites
Packages
Assets > Prefab
asteroid
uzaygemisi

Geri Sayım Sayacını Projeye Dahil Etmek

```
GeriSayimSayaci geriSayimSayaci;
    void Start()
    {
        geriSayimSayaci = gameObject.AddComponent<GeriSayimSayaci>();
        geriSayimSayaci.ToplamSure = 3;
        geriSayimSayaci.Calistir();
    }
    void Update()
    {
        if(geriSayimSayaci.Bitti)
        {
            geriSayimSayaci.Calistir();
        }
    }
```

Instantiate (Örneklendirmek)

- Instantiate: Örnek çıkarmak, örnek oluşturmak

```
void SpawnAsteroid()
{
        //Oyun objesini ekrana basan fonksiyon
        Instantiate(astreoidPrefab);
}
```

SpawnAsteroid() Fonksiyonu Update Fonksiyonu İçerisinde Çağrılması

- Objemizin belirli bir süre geçtikten sonra ekrana basılmasını istiyorsak aşağıdaki işlemler uygulanabilir.
  - Update fonksiyonunun içerisinde geriSayimSayaci örneğindeki (nesnesindeki) Bitti boolean değerini kontrol eden if koşulu yazılır.
  - Bu koşul içerisinde Instantiate ile örnek oluşturan SpawnAsteroid fonksiyonu çağrılır.

```
void Update()
    {
        if(geriSayimSayaci.Bitti)
        {
            //Belirlediğimiz süre geçtikten sonra Bitti
bool değeri True değer döndürür ve bir oyun objesi
ekrana basılır
            SpawnAsteroid();
            //Spawn Game Object (Oyun objesini var etme
işlemi uyguladık)
            geriSayimSayaci.Calistir();
        }
    }
```

UzaySavasiDeneme - Main - PC, Mac & Linux Standalone - Unity 2019.3.7f1 <DX11>

File Edit Assets GameObject Component Window Help

Center Local

Collab Account Layers

Hierarchy

All

Main

Main Camera

asteroid(Clone)

asteroid(Clone)

Scene Game Asset Store

Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos

Inspector

Main Camera

Tag MainCamera Layer Default

Transform

Camera

Audio Listener

Rigidbody 2D

Edge Collider 2D

Edge Collider 2D

Edge Collider 2D

Edge Collider 2D

Spawner (Script)

Script Spawner

Astreoid Prefab asteroid

Geri Sayim Sayaci (Script)

Script GeriSayimSayaci

Add Component

Project Console

Clear Collapse Clear on Play Clear on Build Error Pause Editor

6 0 0

[13:13:59] Çarpışma oldu
UnityEngine.Debug:Log(Object) 9

[13:13:59] Main Camera ile çarpıştı
UnityEngine.Debug:Log(Object) 9

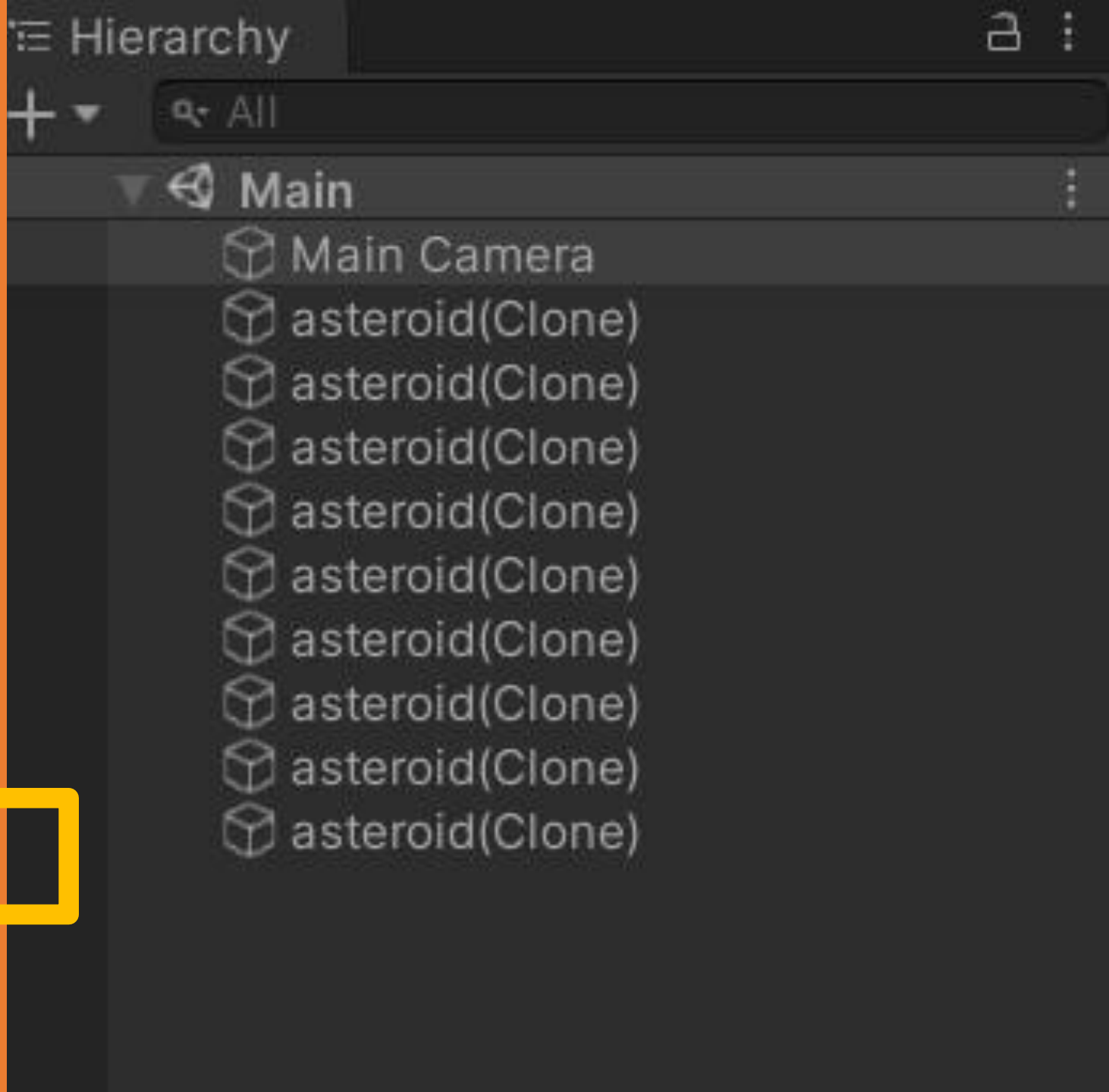
[13:13:59] Main Camera ile çarpıştı
UnityEngine.Debug:Log(Object) 9

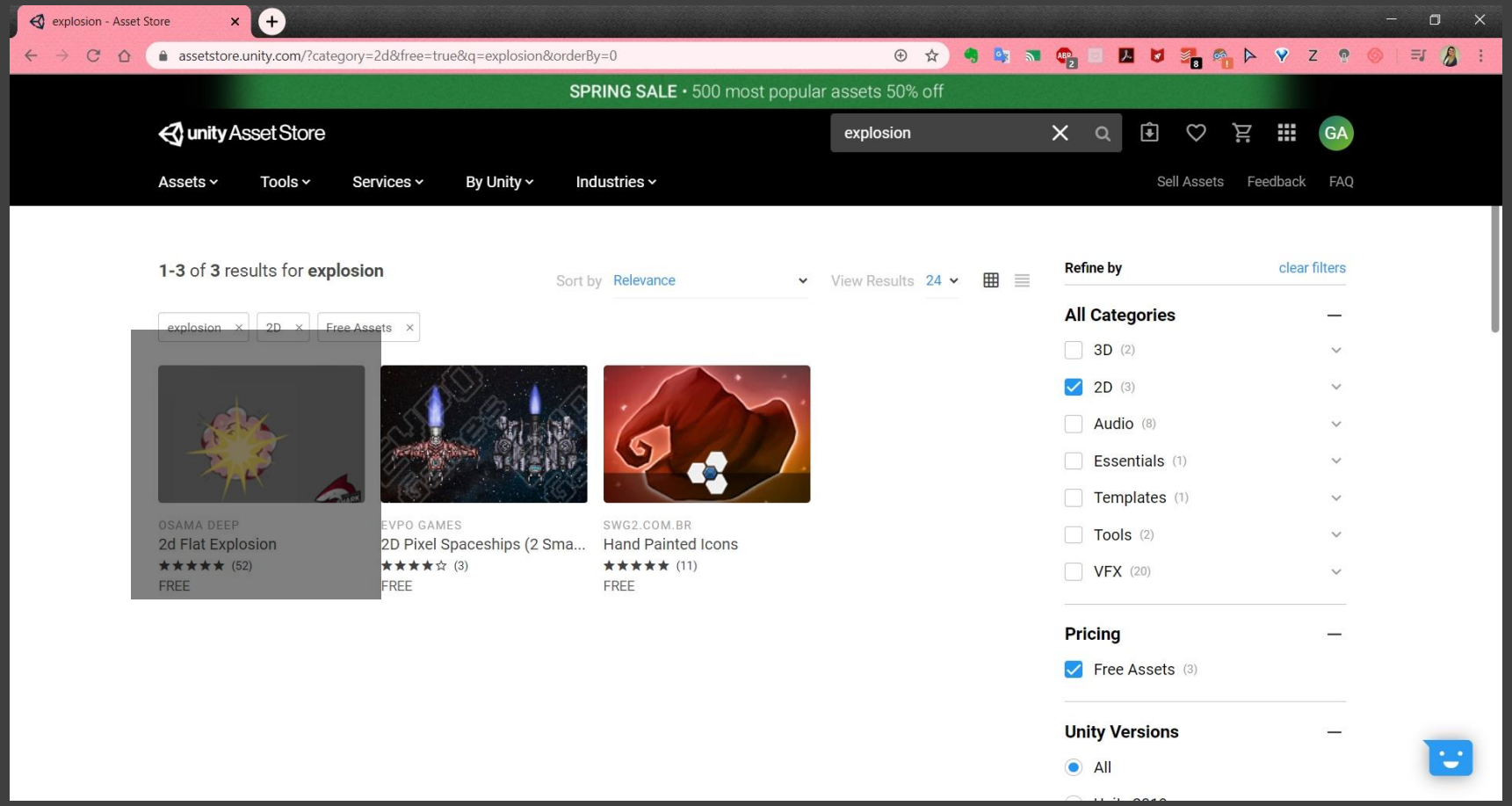
[13:13:59] 10
UnityEngine.MonoBehaviour:print(Object) 9

[13:13:59] Main Camera tag'li nesne ile çarpışmayı bıraktı

Main Camera tag'li nesne ile çarpışmayı bıraktı

astreoid

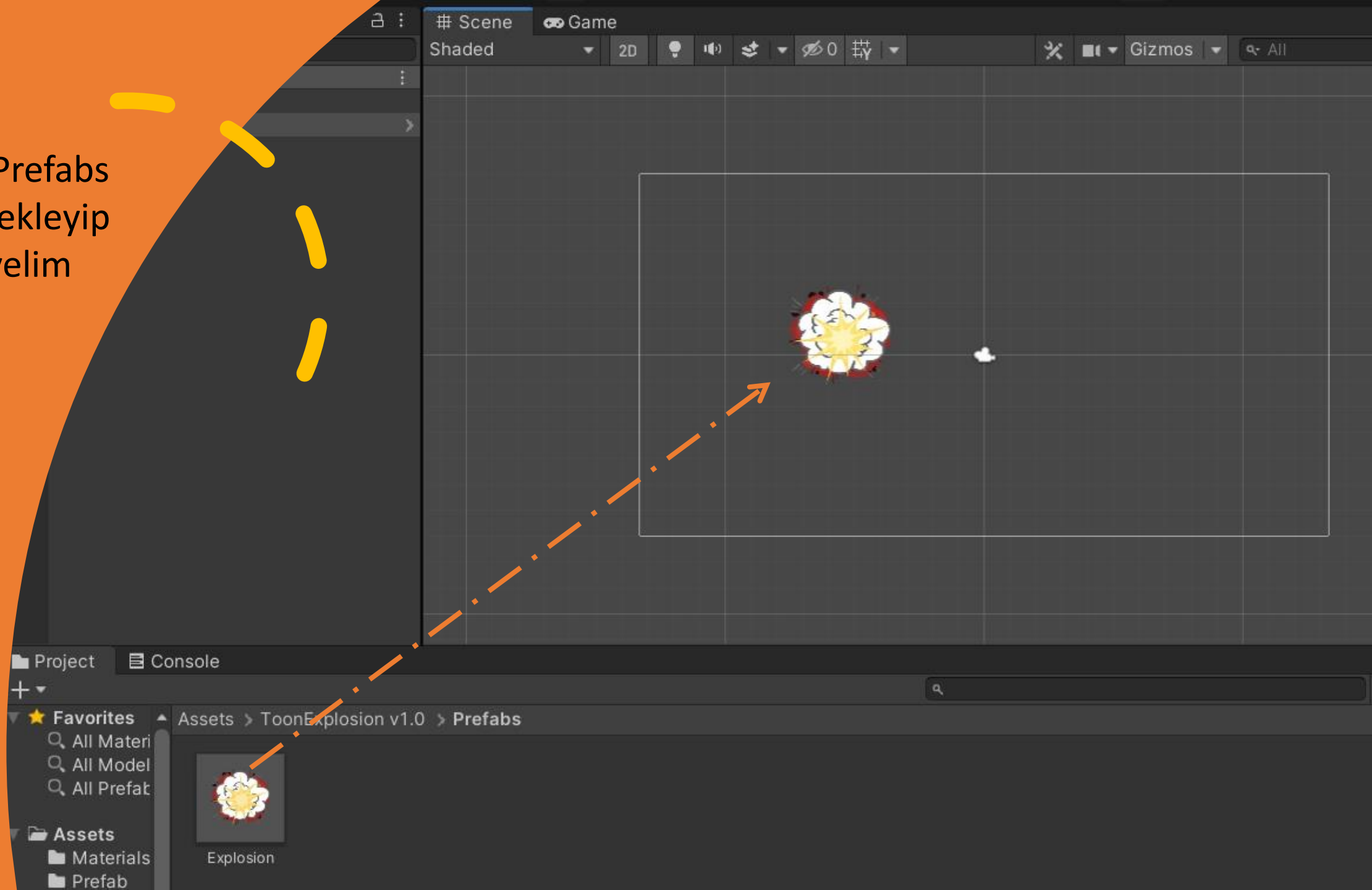

Hierarchy
All
Main
Main Camera
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)



Asset Store üzerinden indirdiğimiz Explosion (Patlama) efektini projemize dahil edelim

- Bunun için Asset Store açalım
- Explosion kelimesini aratalım
- Arama seçeneklerini Free ve 2D olarak filtreleyelim
- Çıkan Asset'lerden ilk asset'i indirip, projemize dahil edelim.
- https://assetstore.unity.com/packages/2d/textures-materials/2d-flat-explosion-66932 Erişim Tarihi: 19.04.2020

Explosion Prefabs projemize ekleyip görüntüleyelim



Dinamik Obje Yok Etmek

GERİSAYİMSAYACİ SCRİPT'İNİ ÇALIŞTIRALIM

PATLAMA EFEKTİ VERELİM

OBJEMİZİ YOK EDELİM

YokEdici.cs

```
public class YokEdici : MonoBehaviour
{
    [SerializeField]
    GameObject patlamaPrefab;
    //GameObject Class'ın adını ifade eder.

    GeriSayimSayaci yokEdiciGeriSayim;

}
```

```
void Start()
    {
        yokEdiciGeriSayim = 
gameObject.AddComponent<GeriSayimSayaci>();
        //gameObject bu script'i eklediğimiz objeyi 
ifade eder.
        yokEdiciGeriSayim.ToplamSure = 3;
        yokEdiciGeriSayim.Calistir();
    }
```

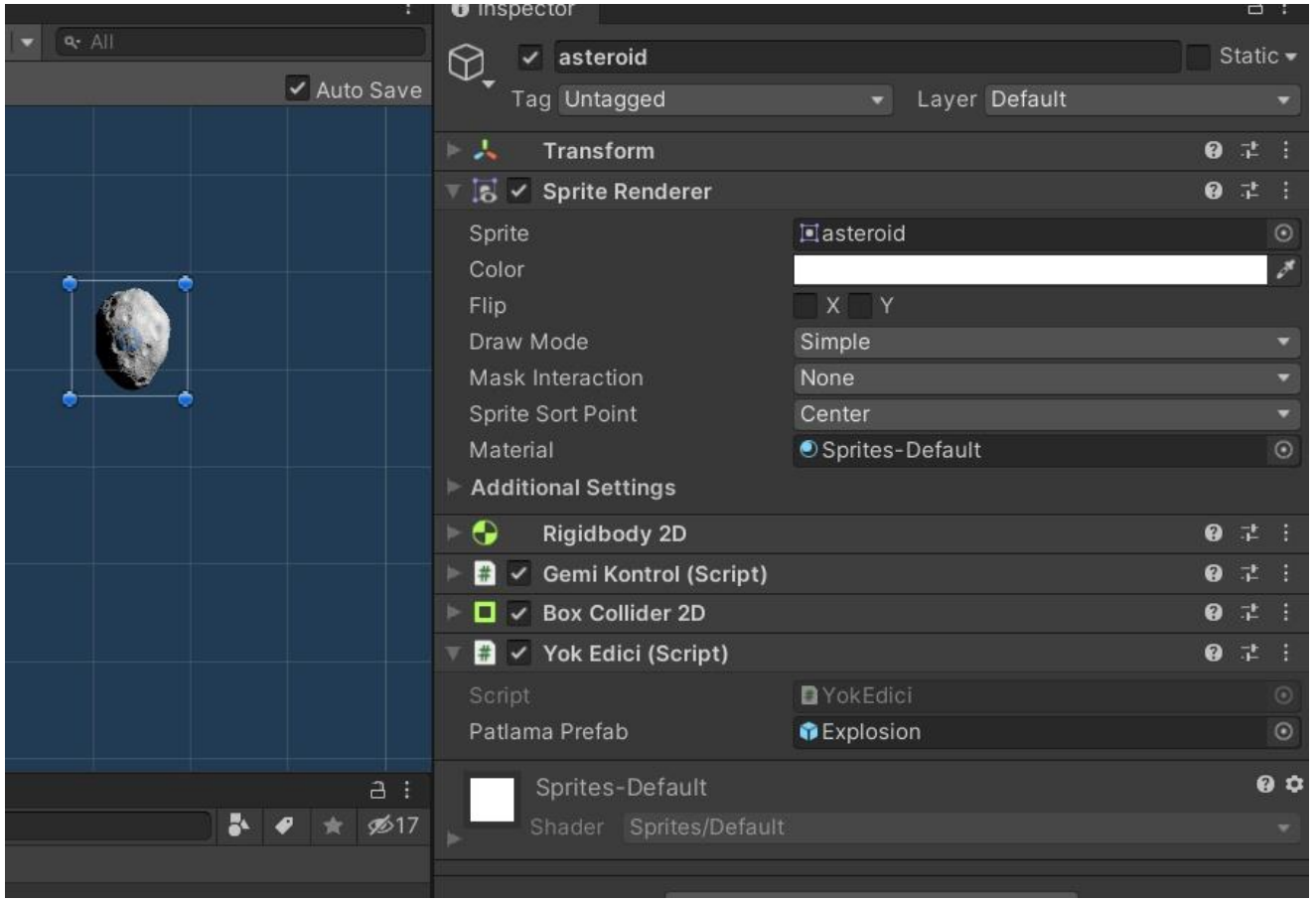
YokEdici.cs

```
void Update()
    {
        if (yokEdiciGeriSayim.Bitti)
        {
            Instantiate(patlamaPrefab);
            Destroy(gameObject);
        }
    }
```

YokEdici.cs

YokEdici Component

- astreoid Prefab’ına YokEdici Component’ini ekledik
- Patlama Prefab’ı için Explosion Prefab objemizi ayarladık.



Instantiate

```
Instantiate(patlamaPrefab,
gameObject.transform.position,
Quaternion.identity);

//gameObject.transform.position :
patlamaPrefab position değerini
objemizin position değeri olarak
belirledik.

//Quaternion.identity : Patlamanın
notasyonunda herhangi bir değişiklik
olmamasını sağladık.
```

YokEdici.cs

Patlamayı Yok Etmek

Bu Script'i çalıştırdığımızda patlama efektini gözlemleyemeyiz. Bunun nedeni patlamanın yok olmasının anlık olarak gerçekleşmesidir. Bunu bir süre boyunca yani sayaç ile yapmamız gerekmektedir.
Bu nedenle patlama objesini yok etme işlemini YokEdici.cs Script'inde Update() metodunun içinde yapmamalıyız.

```
void Update()
    {
        if (yokEdiciGeriSayim.Bitti)
        {
            GameObject patlama = Instantiate(patlamaPrefab,
gameObject.transform.position, Quaternion.identity);
            //gameObject.transform.position : patlamaPrefab position
değerini objemizin position değeri olarak belirledik.
            //Quaternion.identity : Patlamanın notasyonunda herhangi
bir değişiklik olmamasını sağladık.
            Destroy(gameObject);
            patlamaGeriSayim =
gameObject.AddComponent<GeriSayimSayaci>();
            //gameObject bu script'i eklediğimiz objeyi ifade eder.
            Destroy(patlama);
        }
    }
```

YokEdici.cs

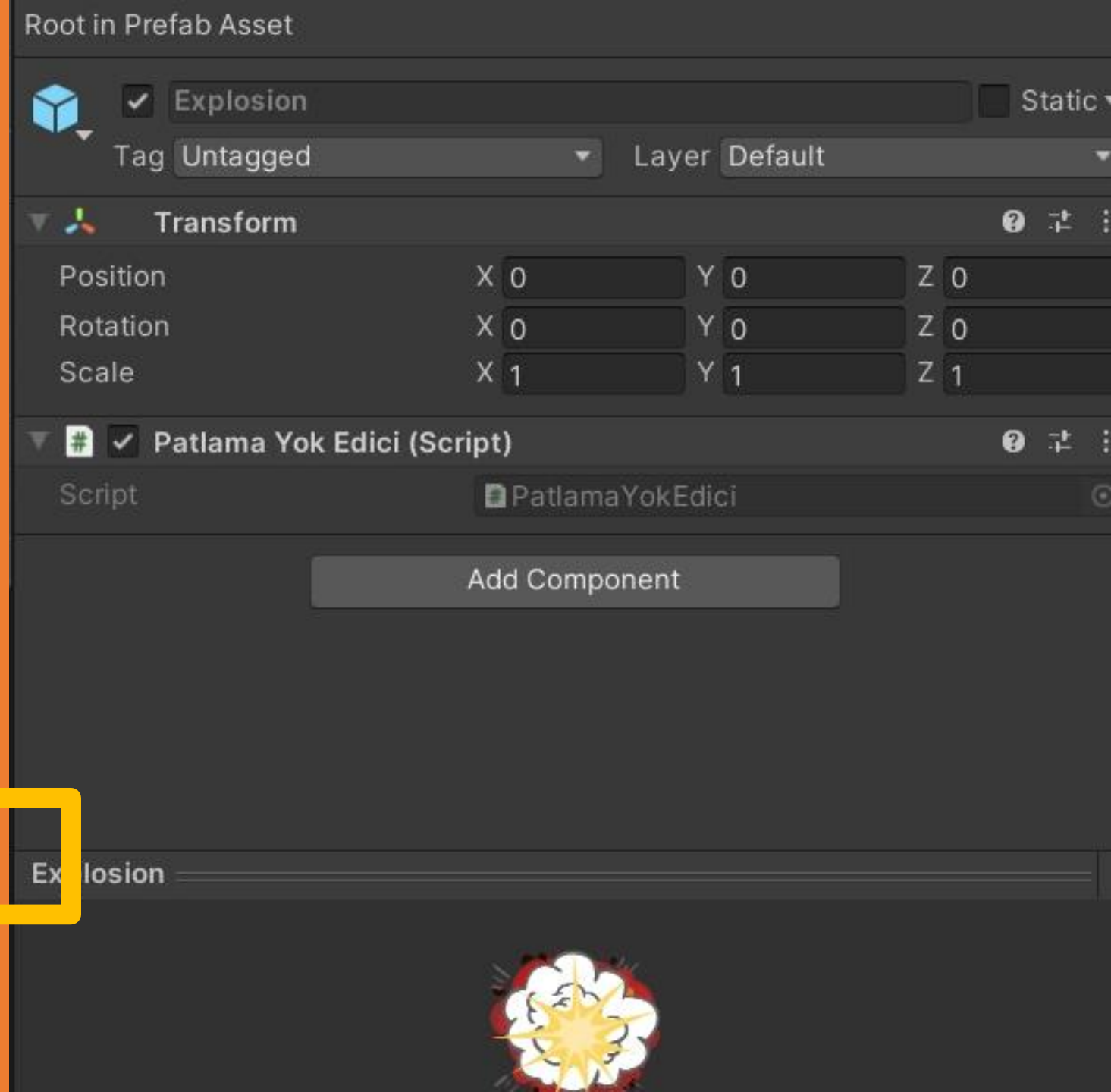
Bu Script içerisinde sadece bu Script'in Component olarak eklenmiş olduğu oyun objesini yok ediyoruz.

```
void Update()
    {
        if (yokEdiciGeriSayim.Bitti)
        {
            Instantiate(patlamaPrefab, gameObject.transform.position, Quaternion.identity);
            //gameObject.transform.position : patlamaPrefab position değerini objemizin position değeri olarak belirledik.
            //Quaternion.identity : Patlamanın notasyonunda herhangi bir değişiklik olmamasını sağladık.
            Destroy(gameObject);
        }
    }
```

Patlama Yok Etmeyi Ayrı Bir Script'te Planlayalım. PatlamaYokEdici.cs

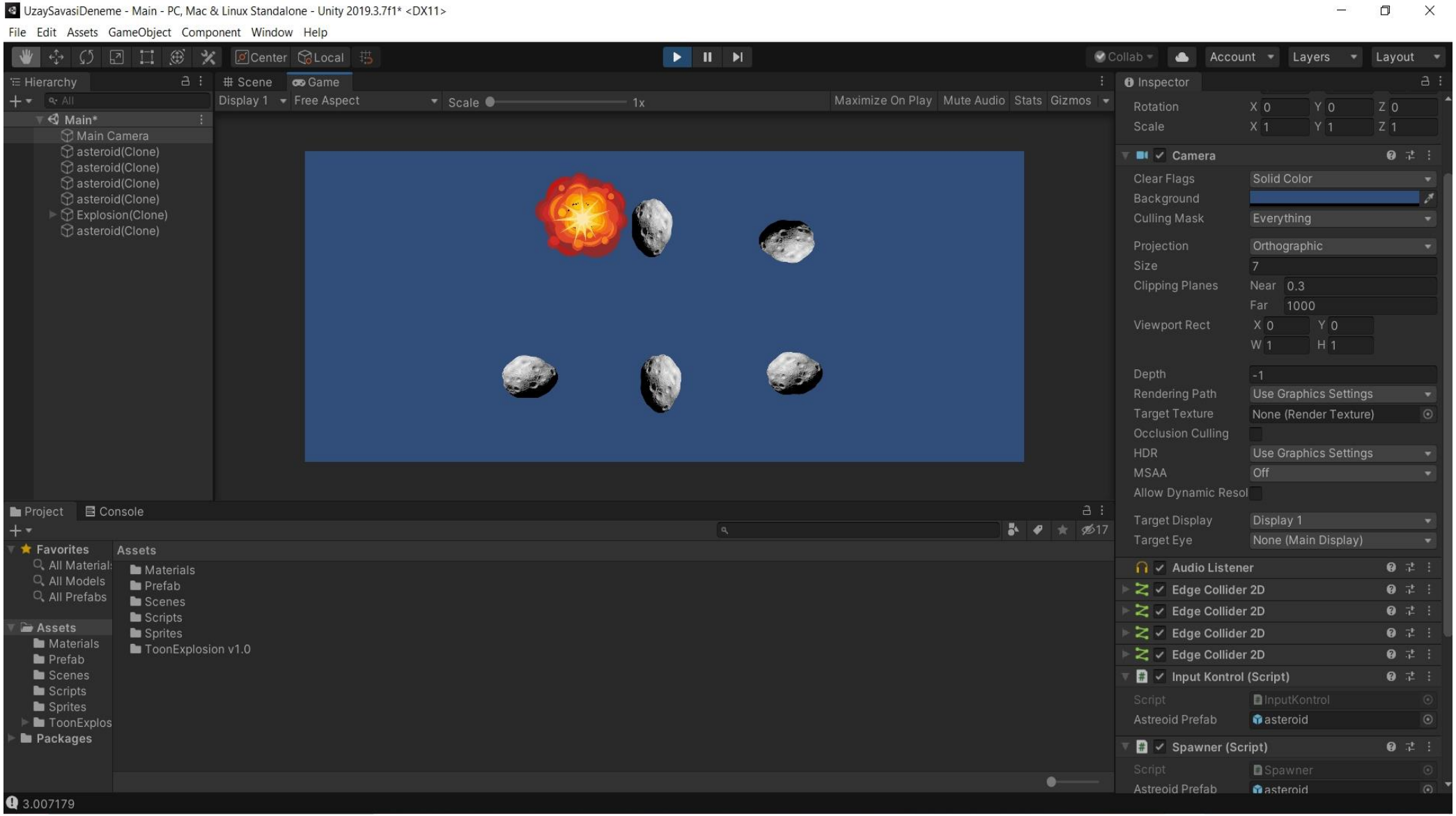
```
//Ekli olduğu objeyi 1 sn'de yok edecek. (PatlamaPrefab)
public class PatlamaYokEdici : MonoBehaviour
{
    GeriSayimSayaci gerisayimsayaci;
    // Start is called before the first frame update
    void Start()
    {
        gerisayimsayaci = gameObject.AddComponent<GeriSayimSayaci>();
        gerisayimsayaci.ToplamSure = 1;
        gerisayimsayaci.Calistir();
    }

    // Update is called once per frame
    void Update()
    {
       if (gerisayimsayaci.Bitti)
        {
            Destroy(gameObject);
        }
    }
}
```

Patlama Yok Edici Component’i Explosion Prefab’e ekleyelim

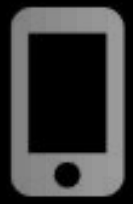


YokEdici.cs
Süreyi 1 ile 20 arasında rastgele bir değer yapalım.

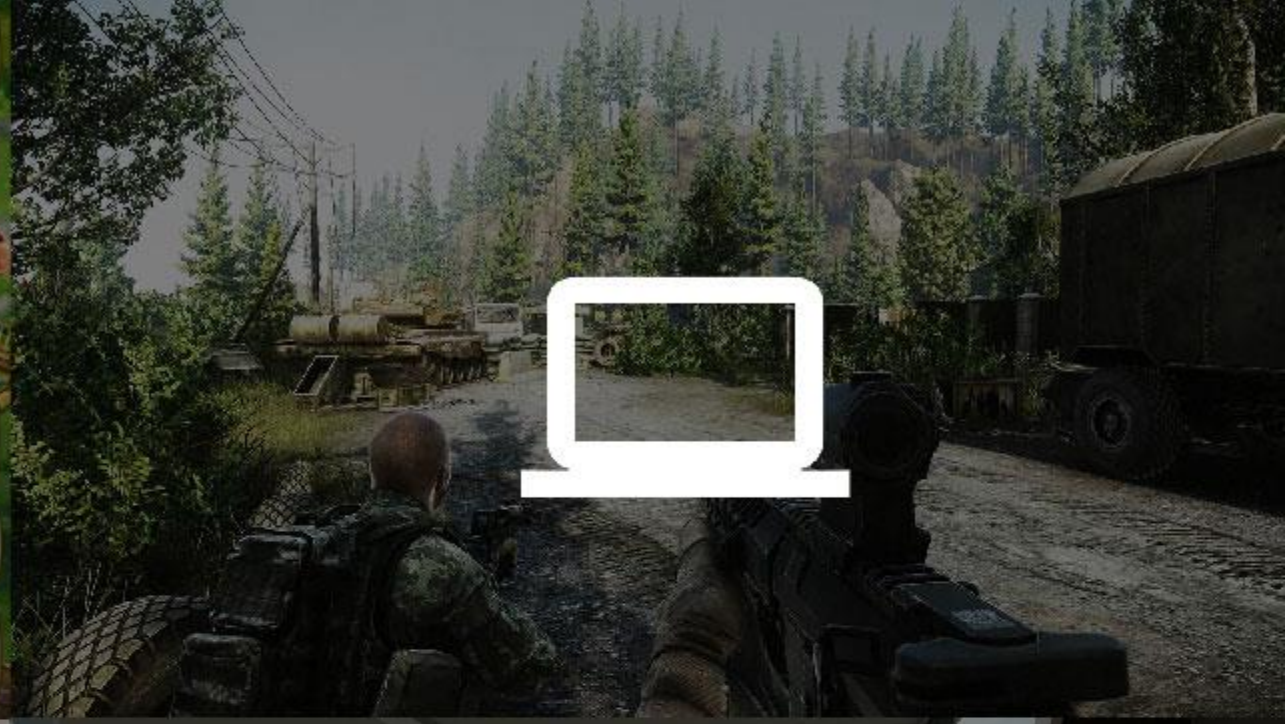
```
void Start()
    {
        yokEdiciGeriSayim = gameObject.AddComponent<GeriSayimSayaci>();
        //gameObject bu script'i eklediğimiz objeyi ifade eder.

yokEdiciGeriSayim.ToplamSure = Random.Range(1,20);

yokEdiciGeriSayim.Calistir();
    }
```

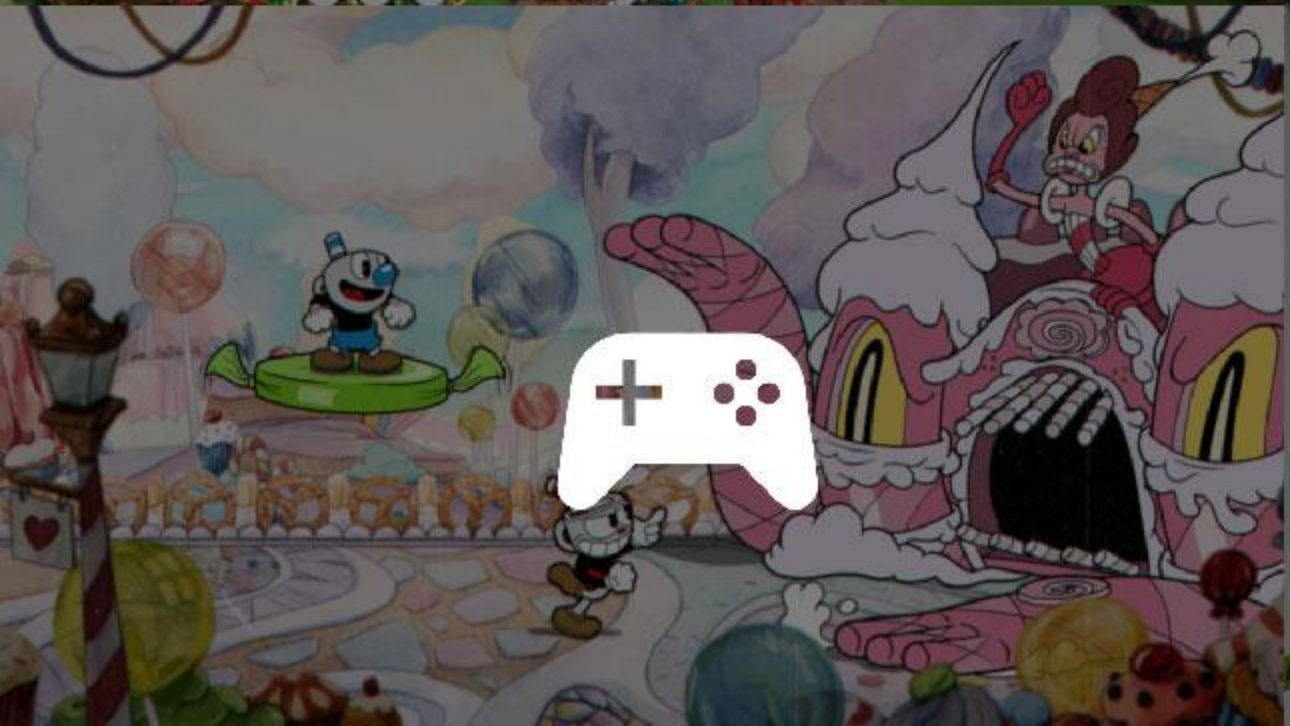

UzaySavasiDeneme - Main - PC, Mac & Linux Standalone - Unity 2019.3.7f1* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Main*
Main Camera
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
Explosion(Clone)
asteroid(Clone)
Scene Game
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Camera
Clear Flags Solid Color
Background
Culling Mask Everything
Projection Orthographic
Size 7
Clipping Planes Near 0.3
Far 1000
Viewport Rect X 0 Y 0
W 1 H 1
Depth -1
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling
HDR Use Graphics Settings
MSAA Off
Allow Dynamic Resol
Target Display Display 1
Target Eye None (Main Display)
Audio Listener
Edge Collider 2D
Edge Collider 2D
Edge Collider 2D
Edge Collider 2D
Input Kontrol (Script)
Script InputKontrol
Astreoid Prefab asteroid
Spawner (Script)
Script Spawner
Astreoid Prefab asteroid
Project Console
Favorites
All Materials
All Models
All Prefabs
Assets
Materials
Prefab
Scenes
Scripts
Sprites
ToonExplos
Packages
Assets
Materials
Prefab
Scenes
Scripts
Sprites
ToonExplosion v1.0
3.007179

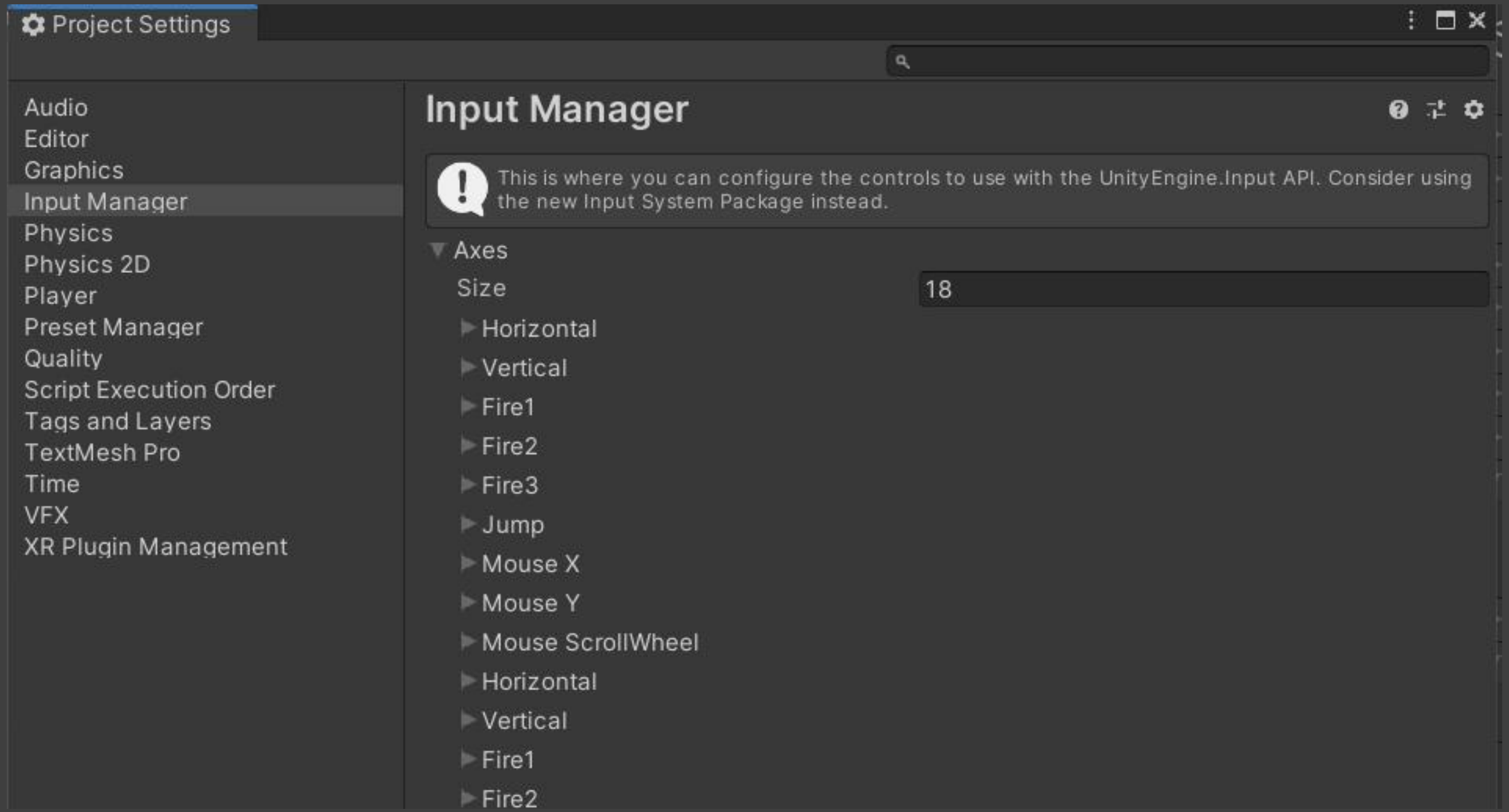
Oyuncu Etkileşimi


81,335
3
0
28
9905/12100
98
3d 11h 55m

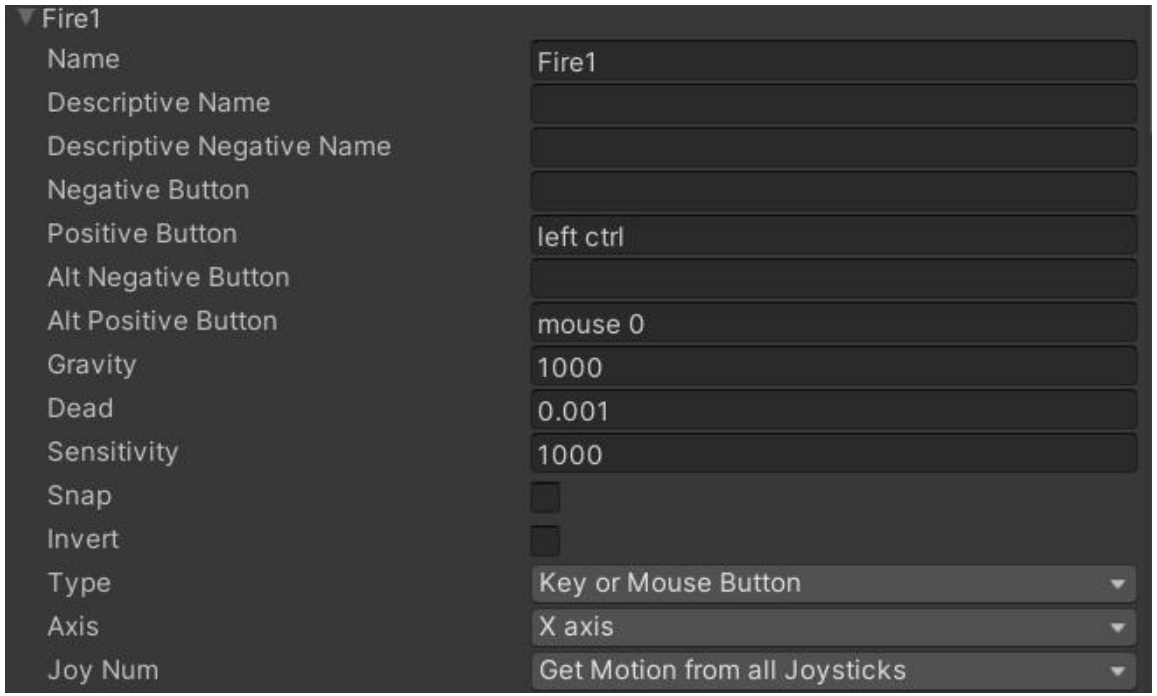


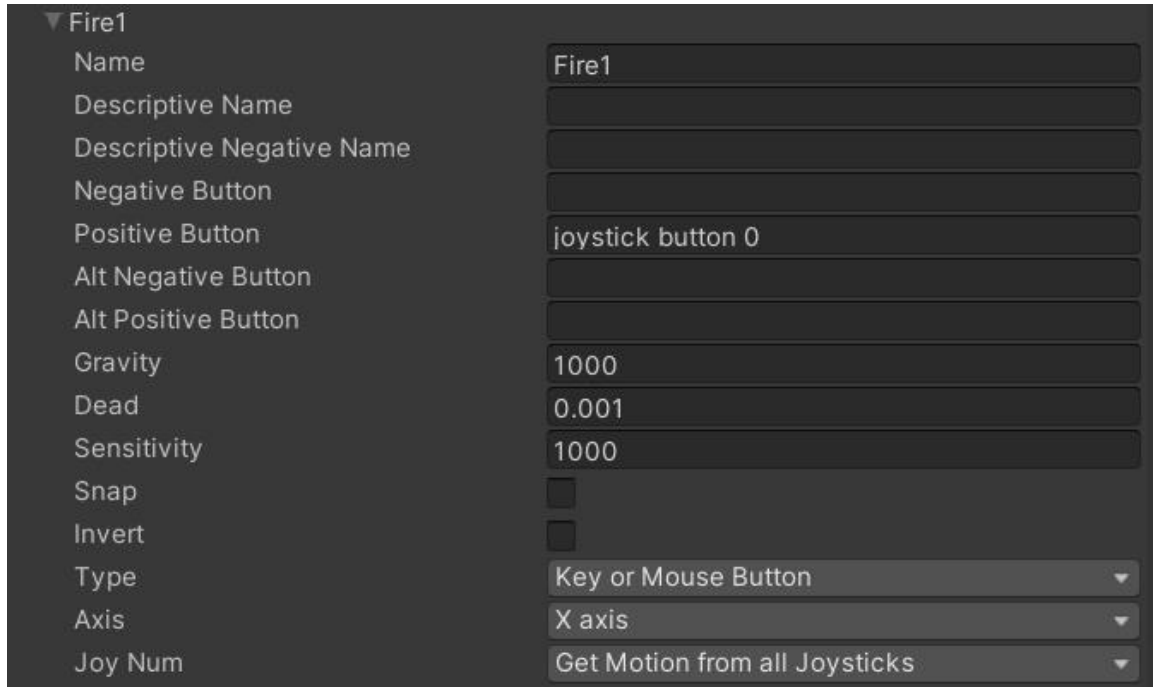



Project Settings
Audio
Editor
Graphics
Input Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Script Execution Order
Tags and Layers
TextMesh Pro
Time
VFX
XR Plugin Management
Input Manager
This is where you can configure the controls to use with the UnityEngine.Input API. Consider using the new Input System Package instead.
Axes
Size
18
Horizontal
Vertical
Fire1
Fire2
Fire3
Jump
Mouse X
Mouse Y
Mouse ScrollWheel
Horizontal
Vertical
Fire1
Fire2

Fire1

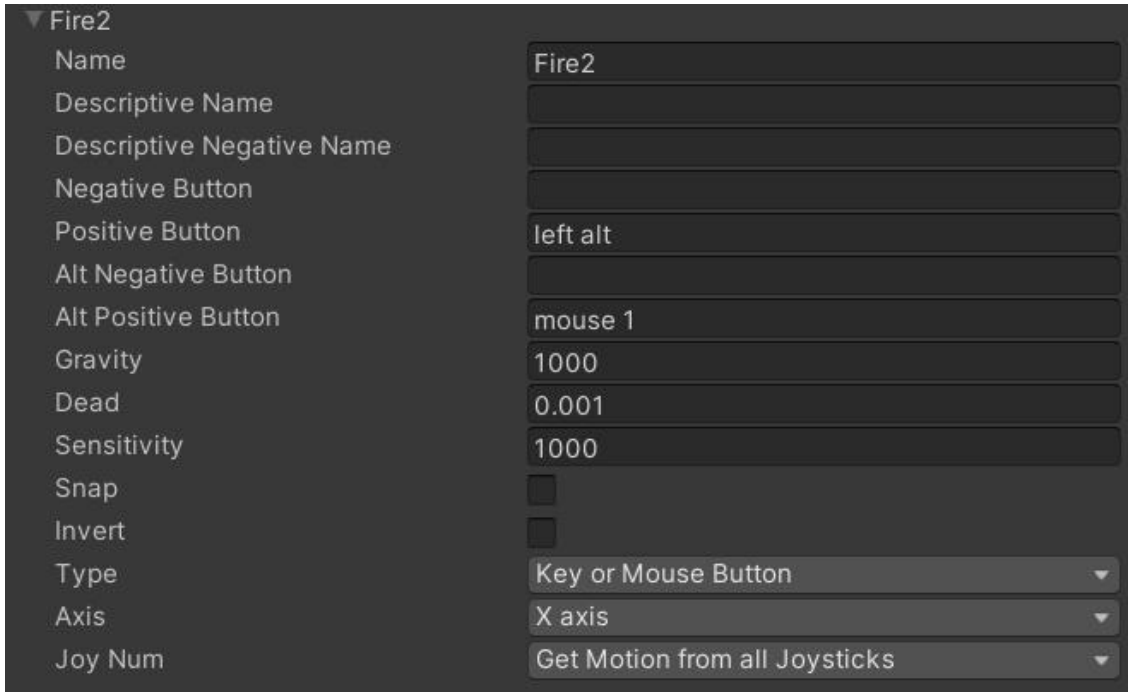
Mouse sol tuşu veya Joystick 0. butonu Fire 1 girdisini ile kullanabiliriz.

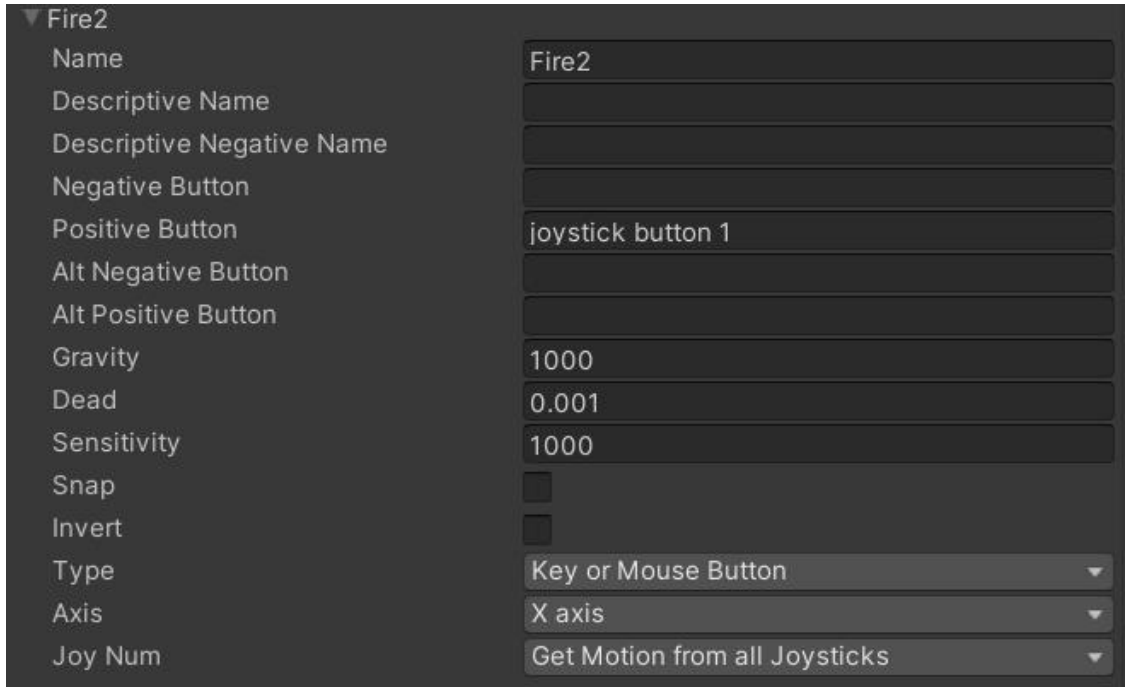


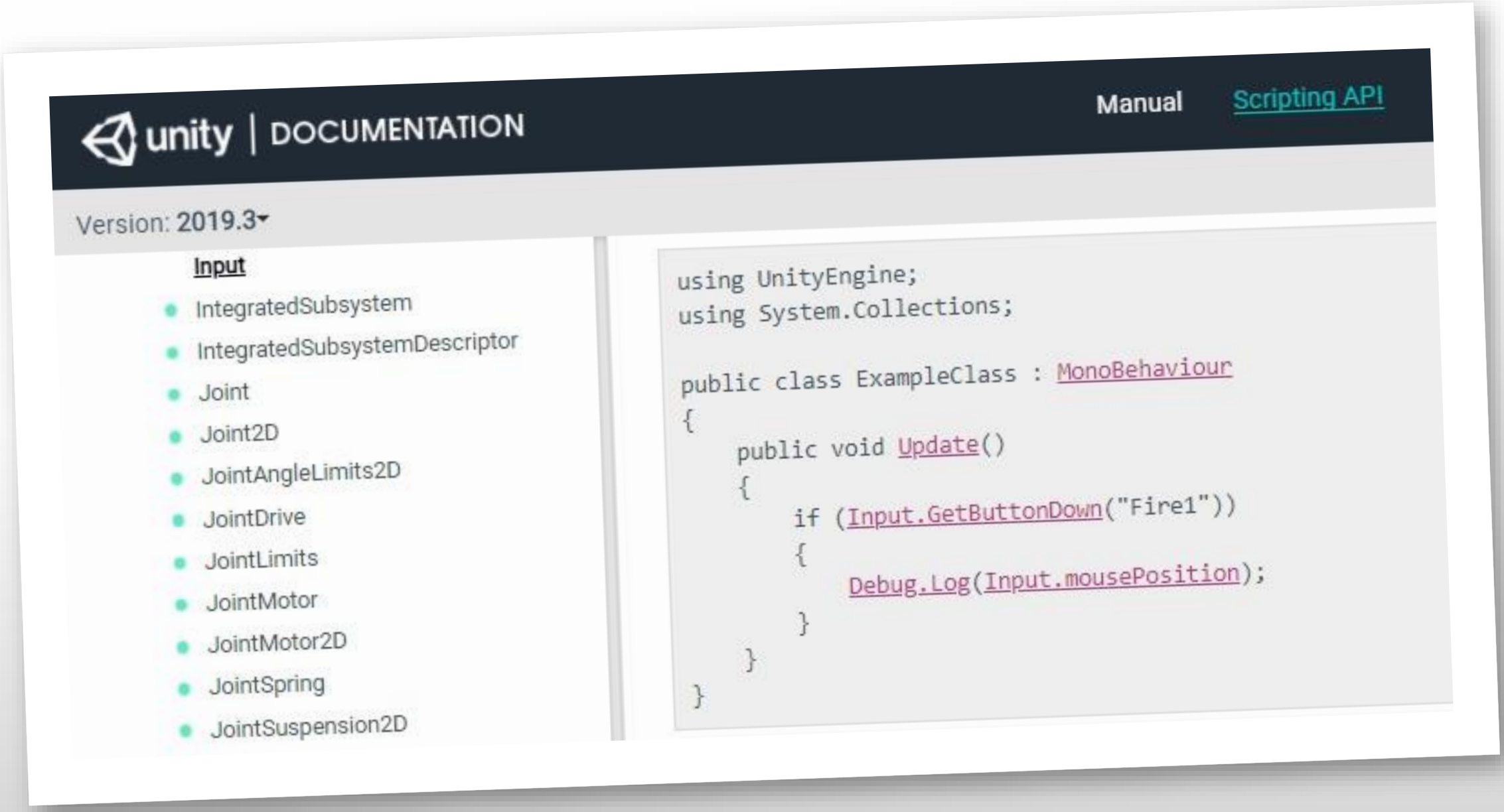


Fire2

Mouse sağ tuşu veya Joystick 1. butonu Fire2 girdisini ile kullanabiliriz.



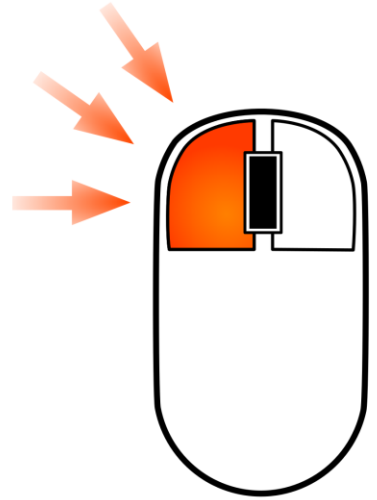




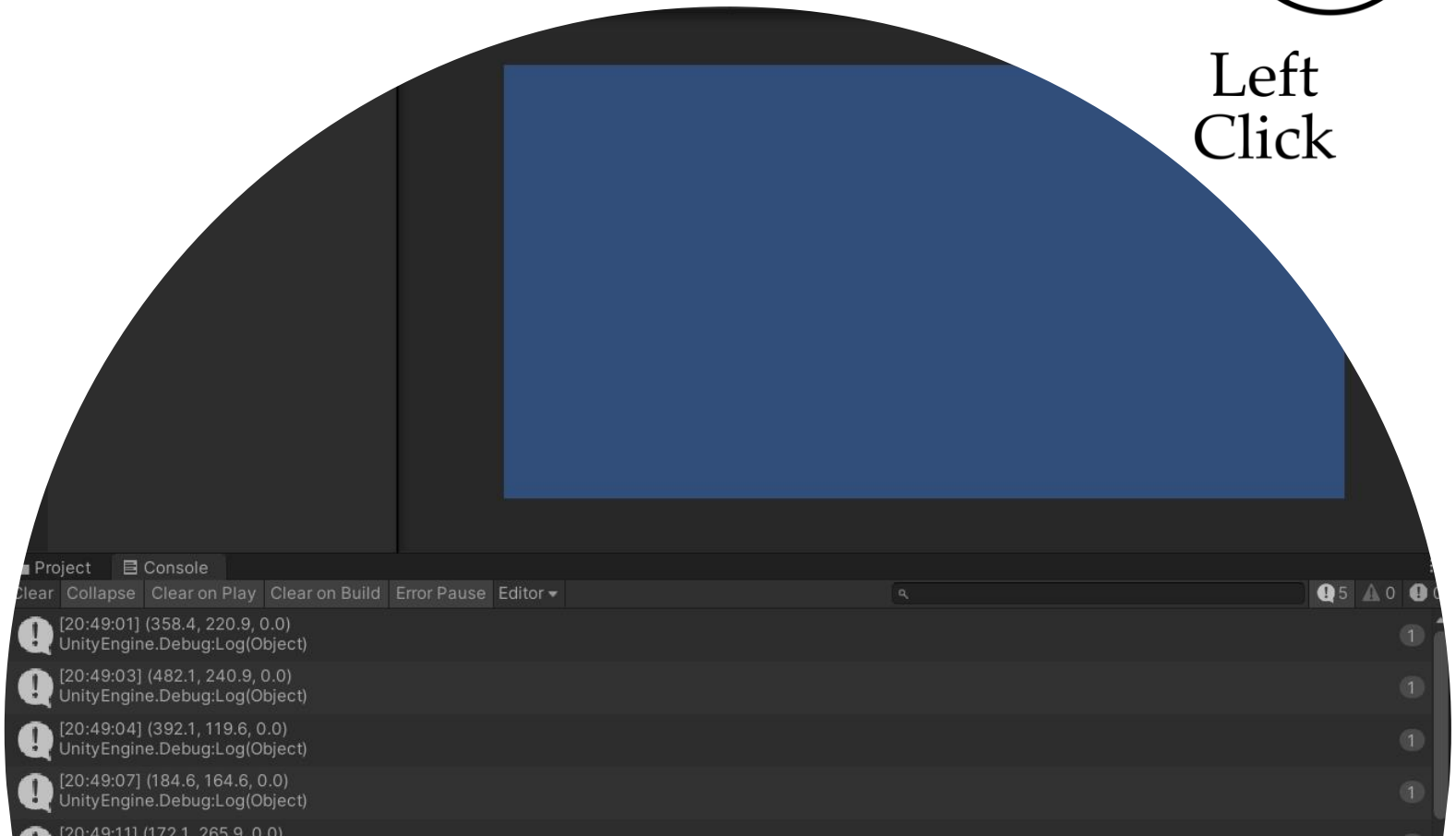
https://docs.unity3d.com/ScriptReference/Input.html

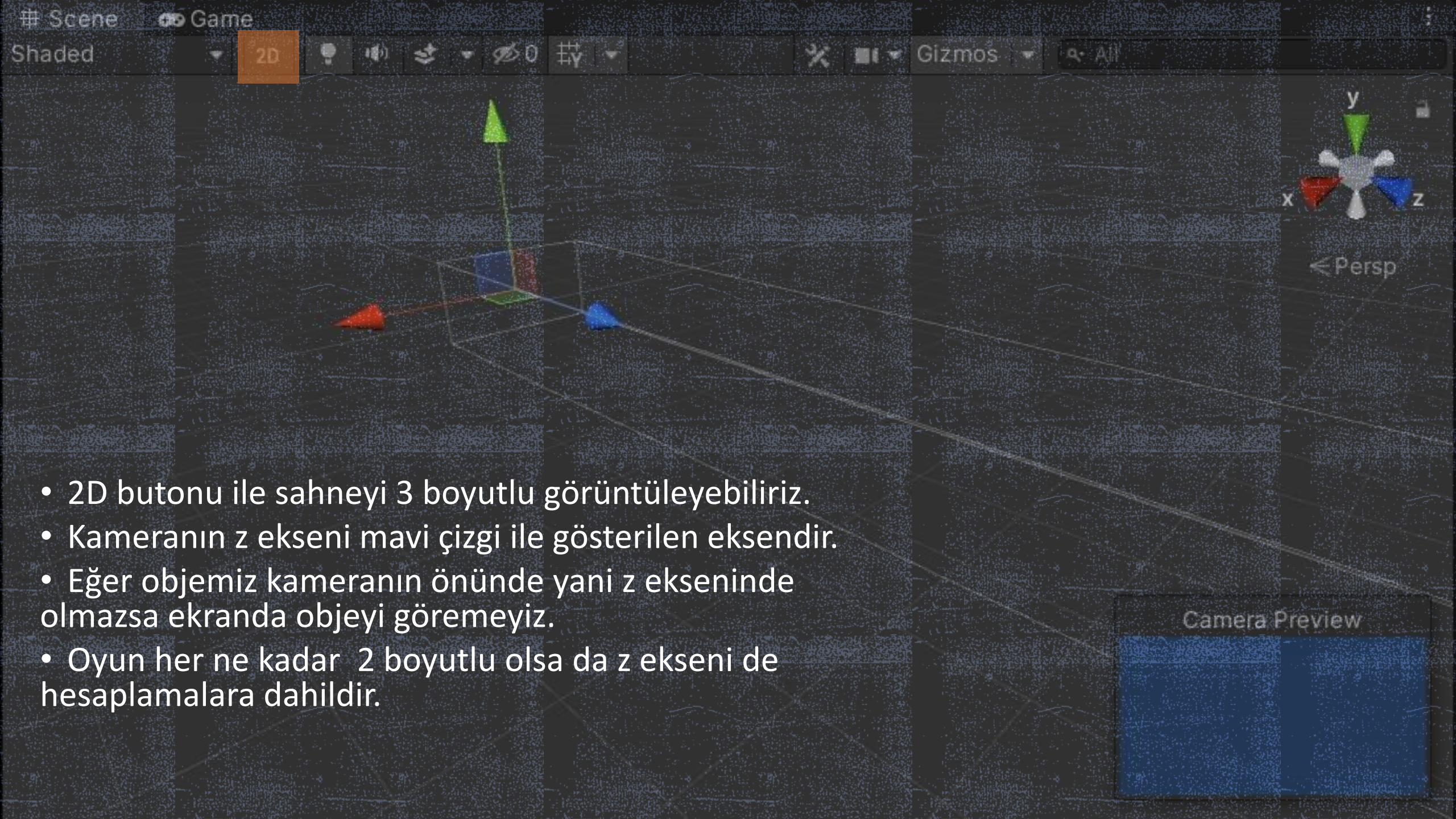
InputKontrol.cs

```
public class InputKontrol : MonoBehaviour
{
    // Update is called once per frame
    void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            Debug.Log(Input.mousePosition);
        }
    }
}
```



Left
Click

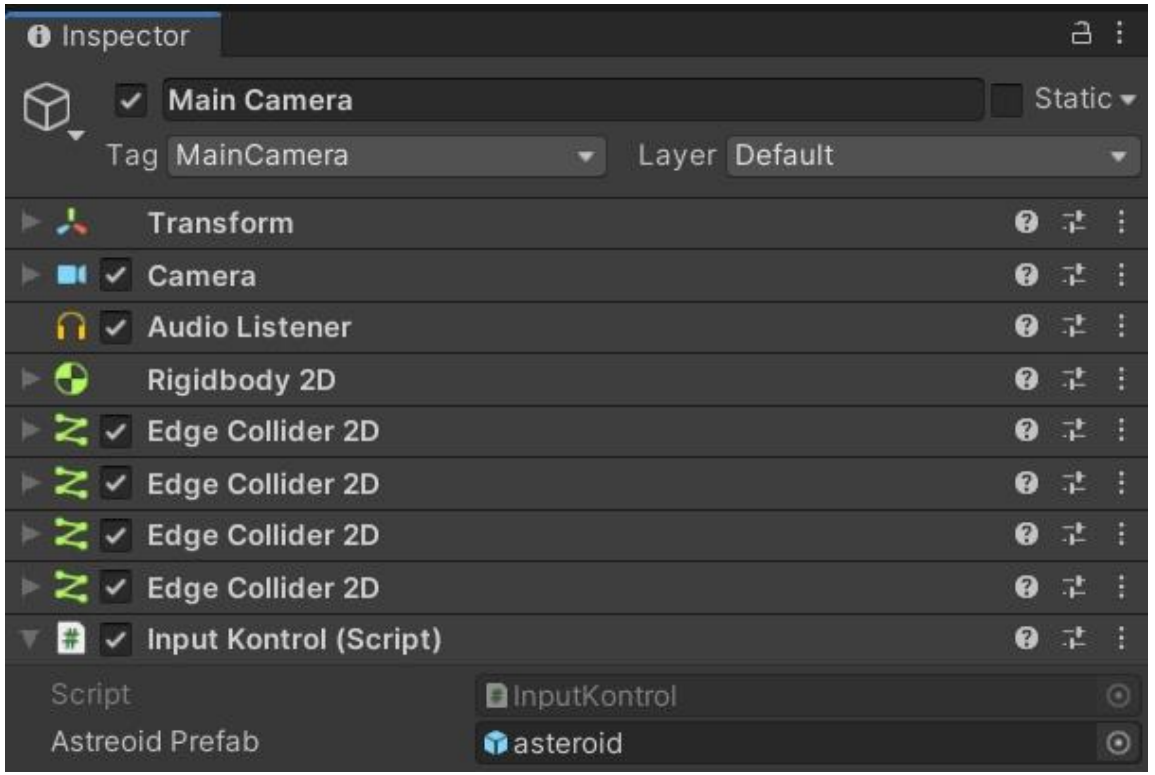

Project
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
5
0
[20:49:01] (358.4, 220.9, 0.0)
UnityEngine.Debug:Log(Object)
[20:49:03] (482.1, 240.9, 0.0)
UnityEngine.Debug:Log(Object)
[20:49:04] (392.1, 119.6, 0.0)
UnityEngine.Debug:Log(Object)
[20:49:07] (184.6, 164.6, 0.0)
UnityEngine.Debug:Log(Object)



- 2D butonu ile sahneyi 3 boyutlu görüntüleyebiliriz.
- Kameranın z ekseni mavi çizgi ile gösterilen eksendir.
- Eğer objemiz kameranın önünde yani z ekseninde olmazsa ekranda objeyi göremeyiz.
- Oyun her ne kadar 2 boyutlu olsa da z ekseni de hesaplamalara dahildir.

InputKontrol.cs Ekranda Tıklanan Yere Obje Eklemek

```
public class InputKontrol :
MonoBehaviour
{
    [SerializeField]
    GameObject astreoidPrefab;
}
```



InputKontrol.cs

```
void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            Debug.Log(Input.mousePosition);
            Instantiate(astreoidPrefab,
Input.mousePosition, Quaternion.identity);
        }
    }
```

- Ekrandaki piksel değerlerinin gösterdiği lokasyon ile oyun dünyamız aynı değildir.
- Bu yüzden mousePosition ile oluşturmuş olduğumuz oyun objelerimiz sahnenin dışında oluşur.
- Position z değerini kameramızın z değerine göre ayarlamamız gerekmektedir.

InputKontrol.cs

```
Vector3 position = Input.mousePosition;
```

- Mouse ile tıklanan noktanın piksellere göre pozisyonunu Vector3 (3 boyutlu) position değişkeninde tutuyoruz.

```
position.z = -Camera.main.transform.position.z;
```

- Bu position'ın z değerini kamerayı referans alarak negatife çeviriyoruz.
- Burada position değerlerinin kendisi ekranın piksellerine göre ayarlandı.
- Onu oyun dünyasındaki değerlere çevirmemiz gerekiyor.

InputKontrol.cs

```
position = Camera.main.ScreenToWorldPoint(position);
```

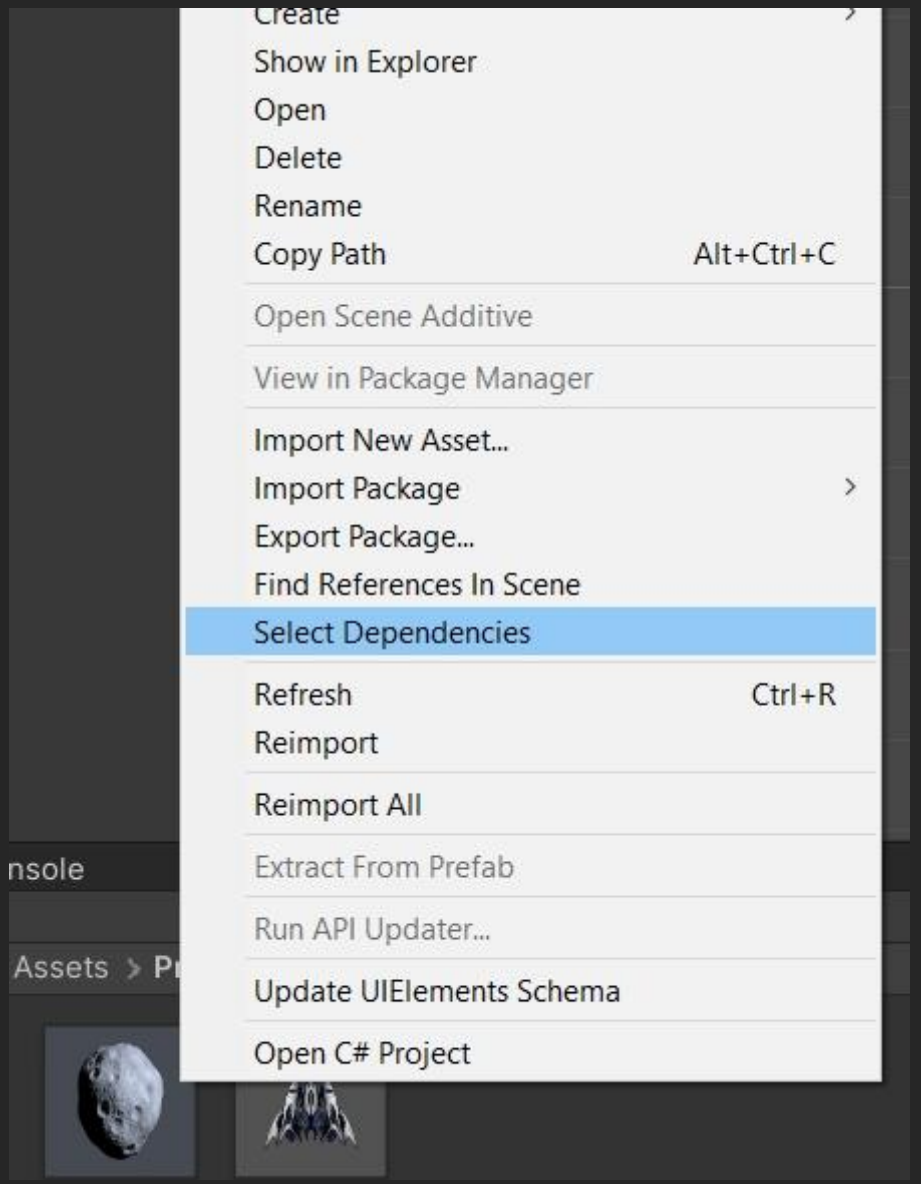
- ScreenToWorldPoint -> piksel değerlerimizi oyun dünyasındaki değerlere dönüştürür.
- Mouse position değeri olarak Unity'nin ekran piksellerine göre verdiği değeri oyun dünyasının kendi koordinatlarına çevirdik.
- Kamerayı referans alıyoruz. Çünkü kameranın yerini değiştirsek bile bu durum objemizin yerini etkilemeyecektir.

```
Instantiate(astreoidPrefab, position, Quaternion.identity);
```

- Bu koordinatlara göre objemizi ekrana ekledik.

InputKontrol.cs

```
[SerializeField]
GameObject astreoidPrefab;
void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            Debug.Log(Input.mousePosition);
            Vector3 position = Input.mousePosition;
            position.z = -
Camera.main.transform.position.z;
            position =
Camera.main.ScreenToWorldPoint(position);
            Instantiate(astreoidPrefab, position,
Quaternion.identity);


        }
    }
```



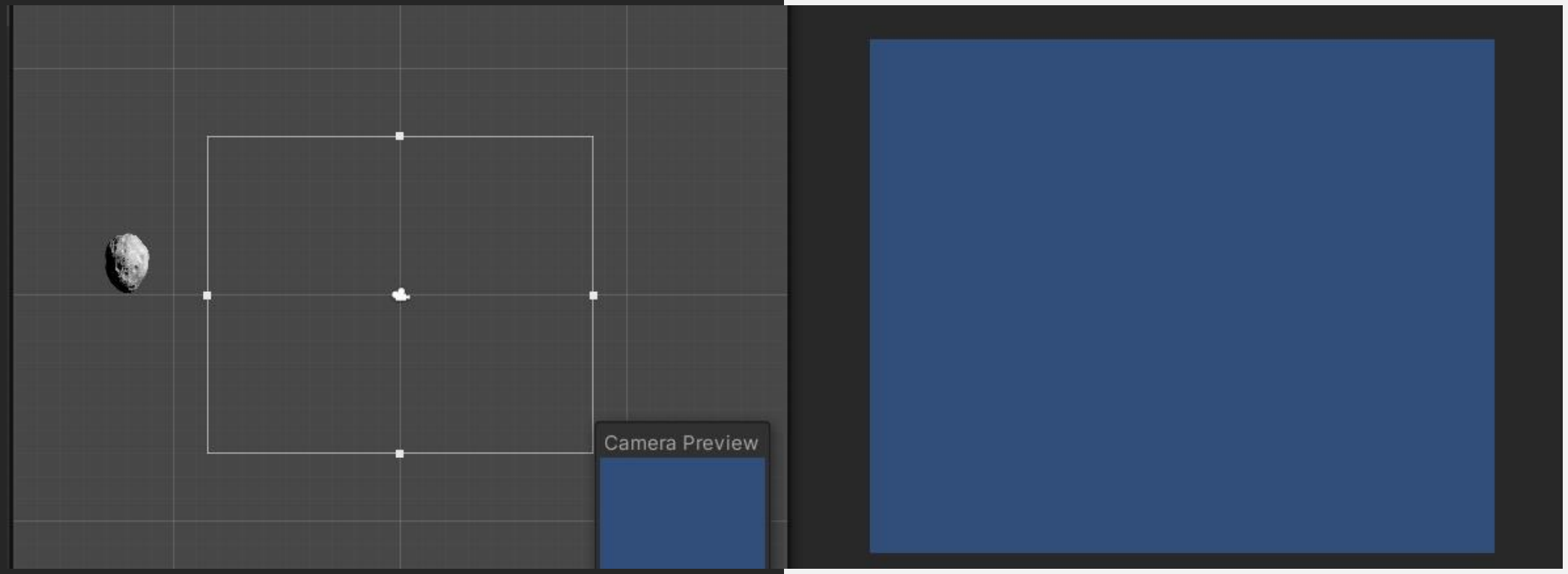
Select Dependencies

Bu Prefab ile ilişkisi olan tüm materyaller listelenir.

Oyun Objesinin Mouse İmlecini Takip Etmesi

Oyun Objesinin Mouse İmlecini Takip Etmesi

```
public class HareketKontrol : MonoBehaviour
{
 void Update()
    {
        //Astreoid mouse imlecini takip edecek
        Vector3 position = Input.mousePosition;
        position.z = -Camera.main.transform.position.z;
        position = Camera.main.ScreenToWorldPoint(position);
        transform.position = position;
    }

}
```



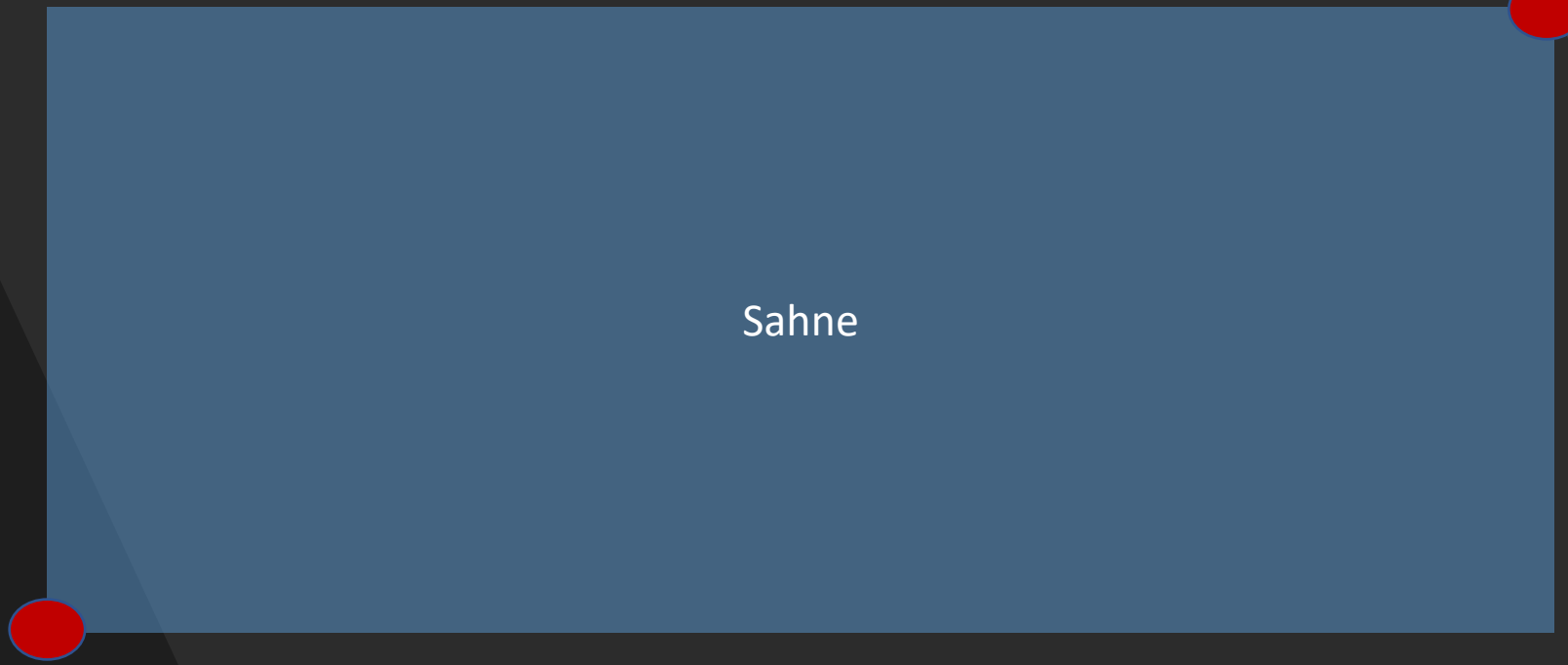
Kontrolcünün Sahnenin Dışına Çıkmasını Engellemek

- Mouse ile objeyi oyun ekranının dışına hareket ettirdiğimizde oyun objemiz sahnenin dışındaki alana geçer.
- Oyun objesinin kameranın gördüğü alanların dışına çıkmaması için belirli bir alan içerisinde hareket etmesine izin verilmesi gerekir.

Vector3(x,y,z)

sagUstKose

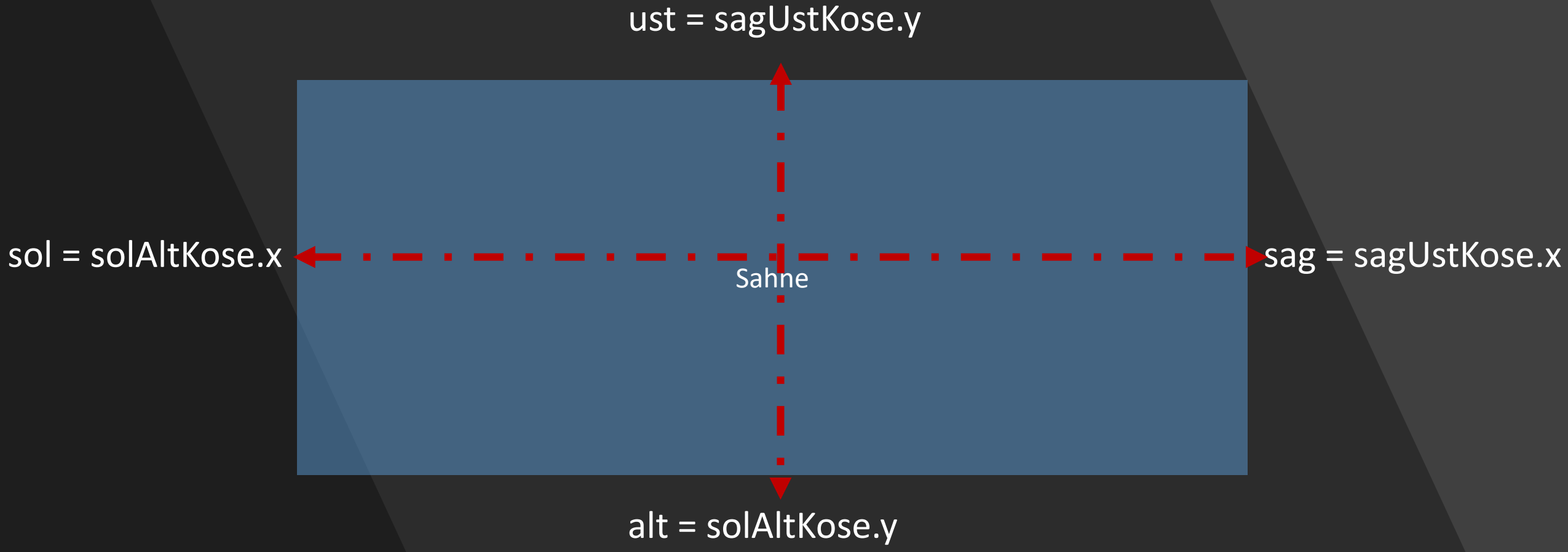
Vector3(Screen.width, Screen.height, zEkseni)



Vector3(0, 0, zEkseni)

solAltKose

Vector3(x,y,z)


ust = sagUstKose.y
sol = solAltKose.x
sag = sagUstKose.x
Sahne
alt = solAltKose.y

```
public static class EkranHesaplayici
```

```
    static float sol, sag, ust, alt;
```

Sol Properties

```
    /// <summary>
    /// Ekranın sol koordinatlarını
verir
    /// </summary>
    public static float Sol
    {
        get
        {
            return sol;
        }
    }
```

Sag Properties

```
    /// <summary>
    /// Ekranın sağ koordinatlarını
verir
    /// </summary>
    public static float Sag
    {
        get
        {
            return sag;
        }
    }
```

Ust Properties

```
    /// <summary>
    /// Ekranın üst koordinatlarını
verir
    /// </summary>
    public static float Ust
    {
        get
        {
            return ust;
        }
    }
```

Alt Properties

```
/// <summary>
    /// Ekranın alt koordinatlarını
verir
    /// </summary>
    public static float Alt
    {
        get
        {
            return alt;
        }
    }
```

Init() Fonksiyonu

```
public static void Init()
    {
        float zEkseni = -Camera.main.transform.position.z;
        Vector3 solAltKose = new Vector3(0, 0, zEkseni);
        Vector3 sagUstKose = new Vector3(Screen.width,
Screen.height, zEkseni);


        Vector3 solAltKoseOyunDunyasi =
Camera.main.ScreenToWorldPoint(solAltKose);
        Vector3 sagUstKoseOyunDunyasi =
Camera.main.ScreenToWorldPoint(sagUstKose);


        sol = solAltKoseOyunDunyasi.x;
        sag = sagUstKoseOyunDunyasi.x;
        ust = sagUstKoseOyunDunyasi.y;
        alt = solAltKoseOyunDunyasi.y;
    }
```

Tetikleyici.cs

```
public class Tetikleyici : MonoBehaviour
{
    // Start is called before the first
frame update
    void Awake()
    {
        //Awake Start işleminden önce
çalışan fonksiyondur.
        EkranHesaplayici.Init();
         Debug.Log(EkranHesaplayici.Sol + "
" + EkranHesaplayici.Sag + " " +
EkranHesaplayici.Ust + " " +
EkranHesaplayici.Alt);
    }
}
```

Tetikleyici.cs

Tetikleyici component’ini Main Camera objemize ekleyelim.

Oyun Başladığında EdgeCollider 2D Component'inin Main Camera'ya Eklenmesini Sağlamak

EkranHesaplayici Componenti ile elde ettiğimiz sınırları EdgeCollider2D Componentine uygulayacağız.

EdgeColliderEkle.cs

- Oyun çalıştırıldığında kameranın üst, alt, sağ ve sol olmak üzere 4 kenarına EdgeCollider2D Component'inin eklenmesi için AddComponent fonksiyonunu yazarız.
- Bu fonksiyon start fonksiyonunda olabileceği gibi proje ilk çalıştığında tetiklenen Awake() vb. fonksiyonlar da tercih edilebilir.

```
void Start()
    {
        EdgeCollider2D edgeUst = gameObject.AddComponent<EdgeCollider2D>();
        EdgeCollider2D edgeAlt = gameObject.AddComponent<EdgeCollider2D>();
        EdgeCollider2D edgeSol = gameObject.AddComponent<EdgeCollider2D>();
        EdgeCollider2D edgeSag = gameObject.AddComponent<EdgeCollider2D>();
    }
```

EdgeCollider2D Componentini tanımlamış olduğumuz fonksiyona tüm kenarları için point noktalarını belirleyecek olduğumuz kodları yazalım.

Üst Kenarı İçin Point Tanımlama

```
Vector2[] colliderpointsUst;
colliderpointsUst = edgeUst.points;
colliderpointsUst[0] = new Vector2(EkranHesapla.Sol, EkranHesapla.Ust);
colliderpointsUst[1] = new Vector2(EkranHesapla.Sag, EkranHesapla.Ust);
edgeUst.points = colliderpointsUst;
```

Alt Kenarı İçin Point Tanımlama

```
Vector2[] colliderpointsAlt;
colliderpointsAlt = edgeAlt.points;
colliderpointsAlt[0] = new Vector2(EkranHesapla.Sol, EkranHesapla.Alt);
colliderpointsAlt[1] = new Vector2(EkranHesapla.Sag, EkranHesapla.Alt);
edgeAlt.points = colliderpointsAlt;
```

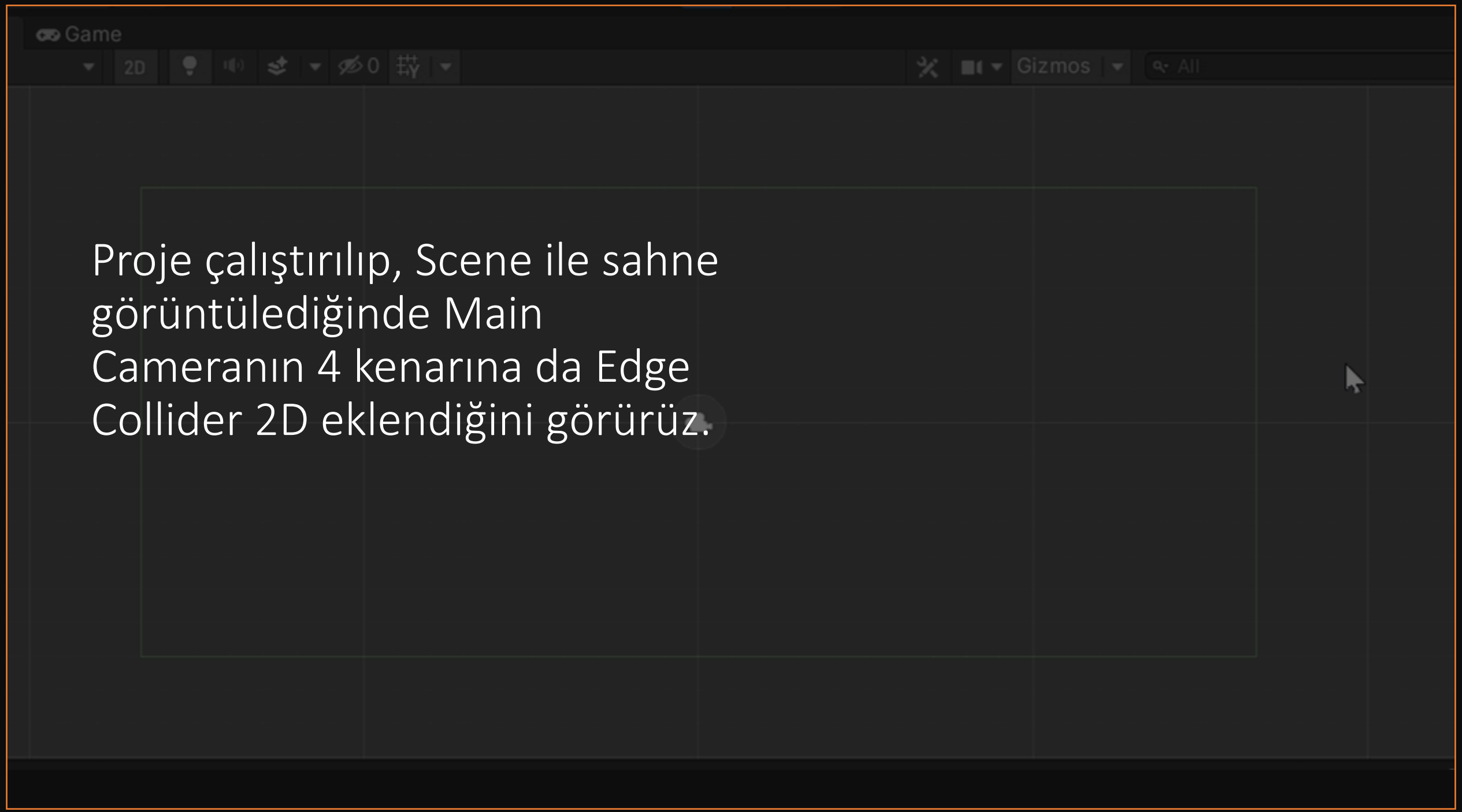
Sol Kenarı İçin Point Tanımlama

```
Vector2[] colliderpointsSol;
colliderpointsSol = edgeAlt.points;
colliderpointsSol[0] = new Vector2(EkranHesapla.Sol, EkranHesapla.Ust);
colliderpointsSol[1] = new Vector2(EkranHesapla.Sol, EkranHesapla.Alt);
edgeSol.points = colliderpointsSol;
```

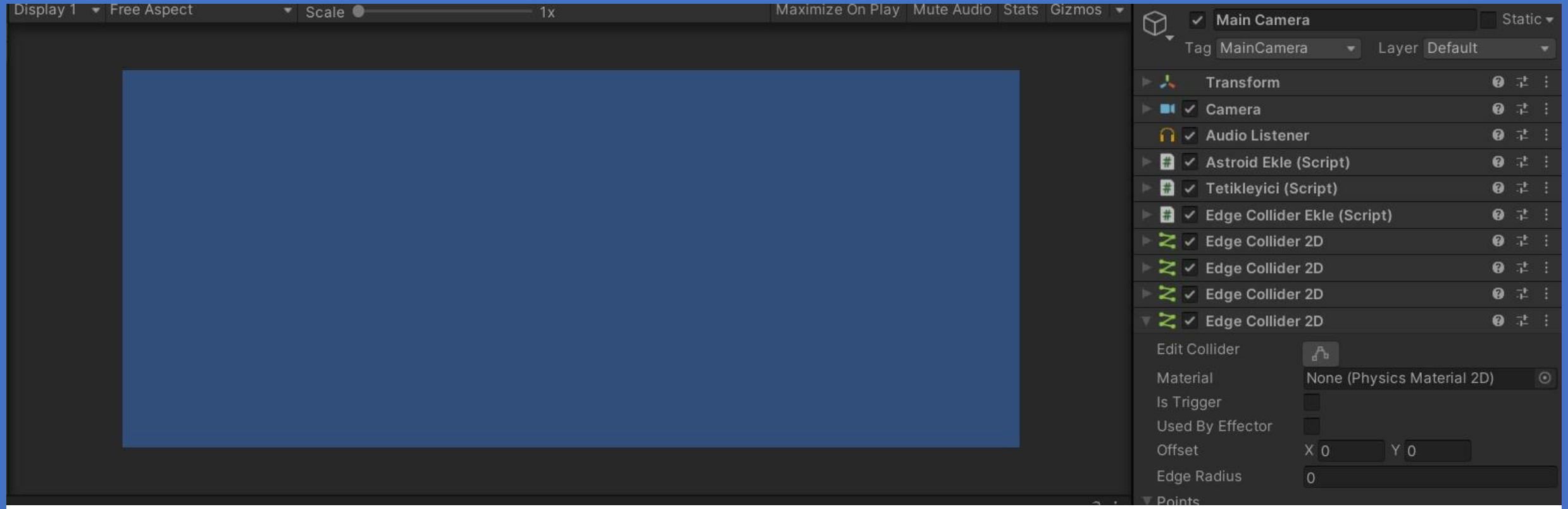
Sağ Kenarı İçin Point Tanımlama

```
Vector2[] colliderpointsSag;
colliderpointsSag = edgeAlt.points;
colliderpointsSag[0] = new Vector2(EkranHesapla.Sag, EkranHesapla.Ust);
colliderpointsSag[1] = new Vector2(EkranHesapla.Sag, EkranHesapla.Alt);
edgeSag.points = colliderpointsSag;
```

Main Camera objesine Edge Collider Ekle Componenti eklenir.



Proje çalıştırılıp, Scene ile sahne görüntülediğinde Main Cameranın 4 kenarına da Edge Collider 2D eklendiğini görürüz.



Oyun çalışırken Inspector penceresinde Edge Collider 2D Componentleri ile ilgili özellikleri görüntüleyebiliriz.

Friction (Sürtünme) ve Bounciness (Sıçrama) Değerlerini Değiştirmek

```
PhysicsMaterial2D materyal = new PhysicsMaterial2D();
materyal.friction = 0;
materyal.bounciness = 1;
edgeUst.sharedMaterial = materyal;
```