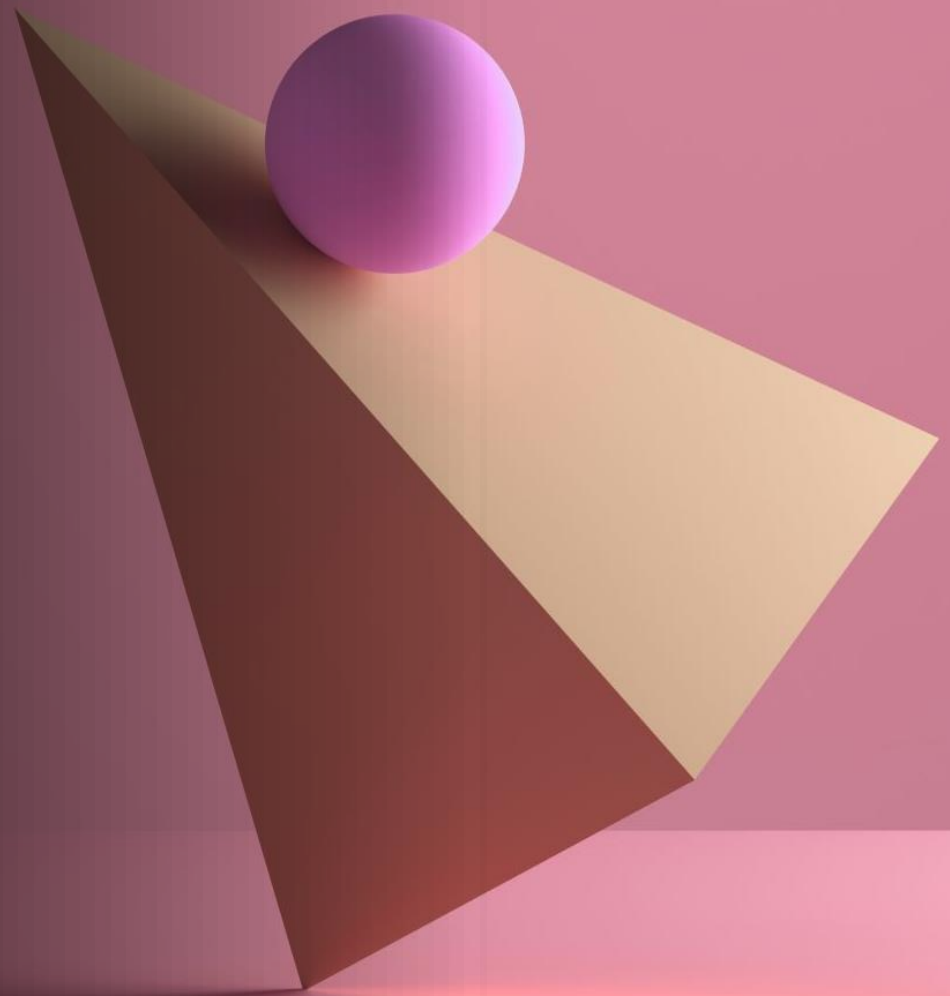
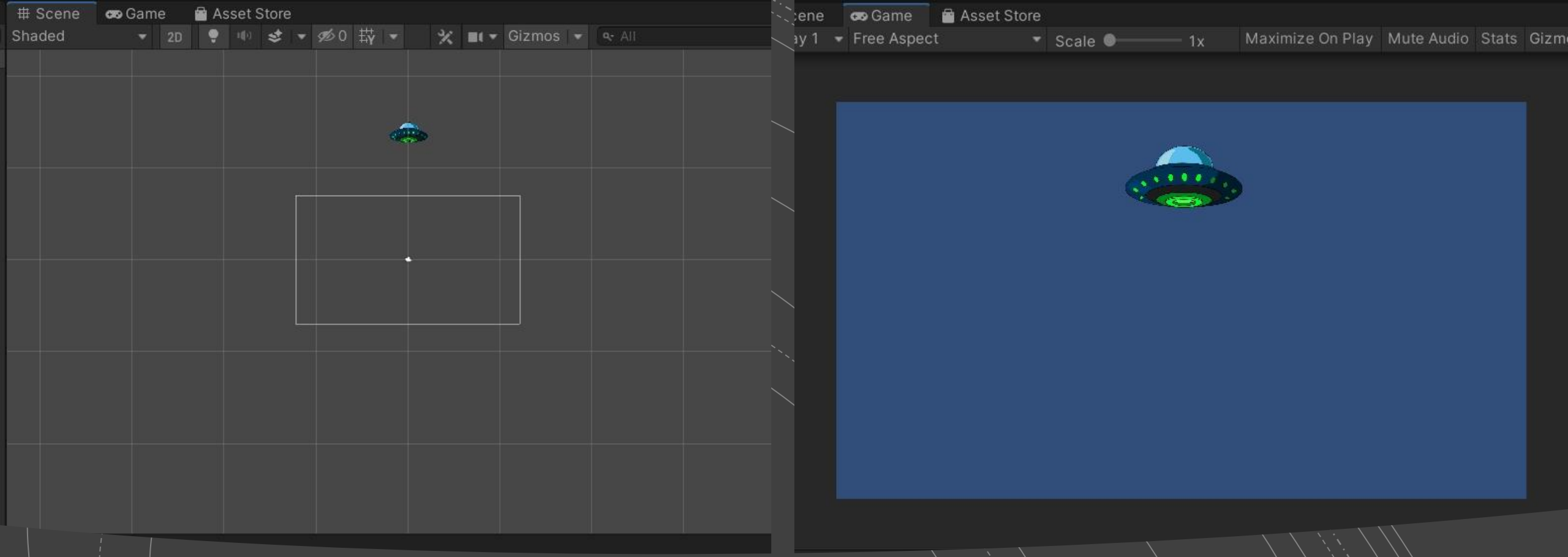
Unity

Öğr. Gör. Gözde Mihran Altınsoy



Oyun Objelerinin Vektörel Hareketi

- Rigidbody Component'ını kullanabilmek için öncelikle onun referansını almalıyız.
- Kendi kodumuz içerisinde kullanabileceğimiz bir şekilde tanımlamalıyız.
- Önce Component adını yazarız, sonra referans olmasını istediğimiz nesne adını yazarız. Component'ımız getirmek için GetComponent ile <> sembolleri içerisinde Component'ımızın adını yazıp, Method olduğu için (Generic Method) en sonuna parantez açıp kapatmamız gerekiyor.
- Bu referans ile RigidBody2D'nin Public methodlarına erişebiliriz.
- Rigidbody2D myRigidBody2d = GetComponent<Rigidbody2D>();
- Bu tarz bir nesneyi vektörel hareket ettirmek için Impulse modunu kullanırız. Impulse itme kuvvetidir.s
- myRigidBody2d.AddForce(new Vector2(0,5),ForceMode2D.Impulse);



Scene (Sahne) Görünümü

- Play ile oyunu oynattığımızda Scene (sahne) penceresine geçtiğimizde oyun objemizin harekete devam ettiğini görüyoruz.

Çarpışma Algılama (Collision Detection)

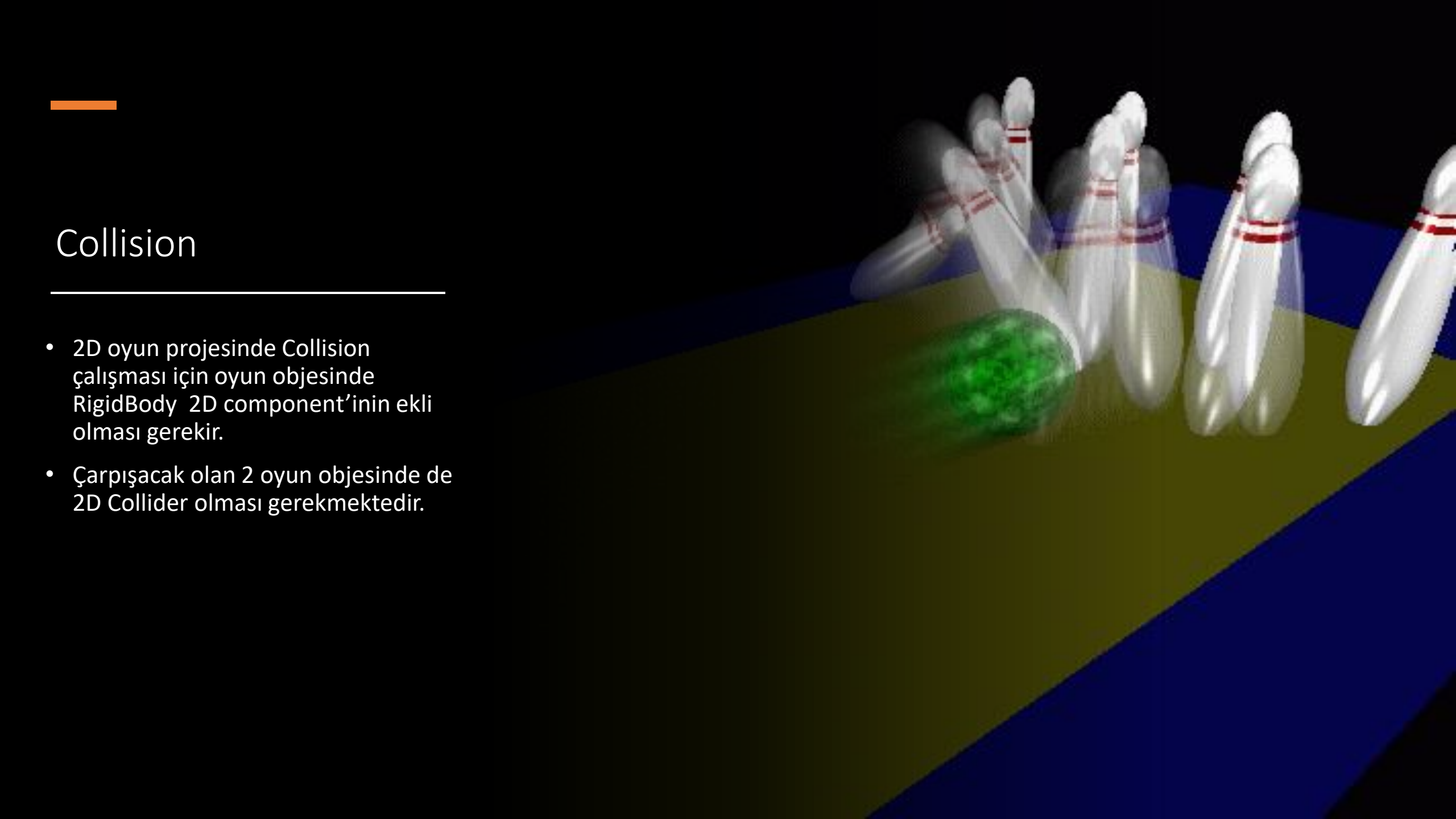
Collection Detection (Çarpışma Algılama) işleminin en temeldeki amacı 2 oyun objesinin arasındaki etkileşime bakmaktır.

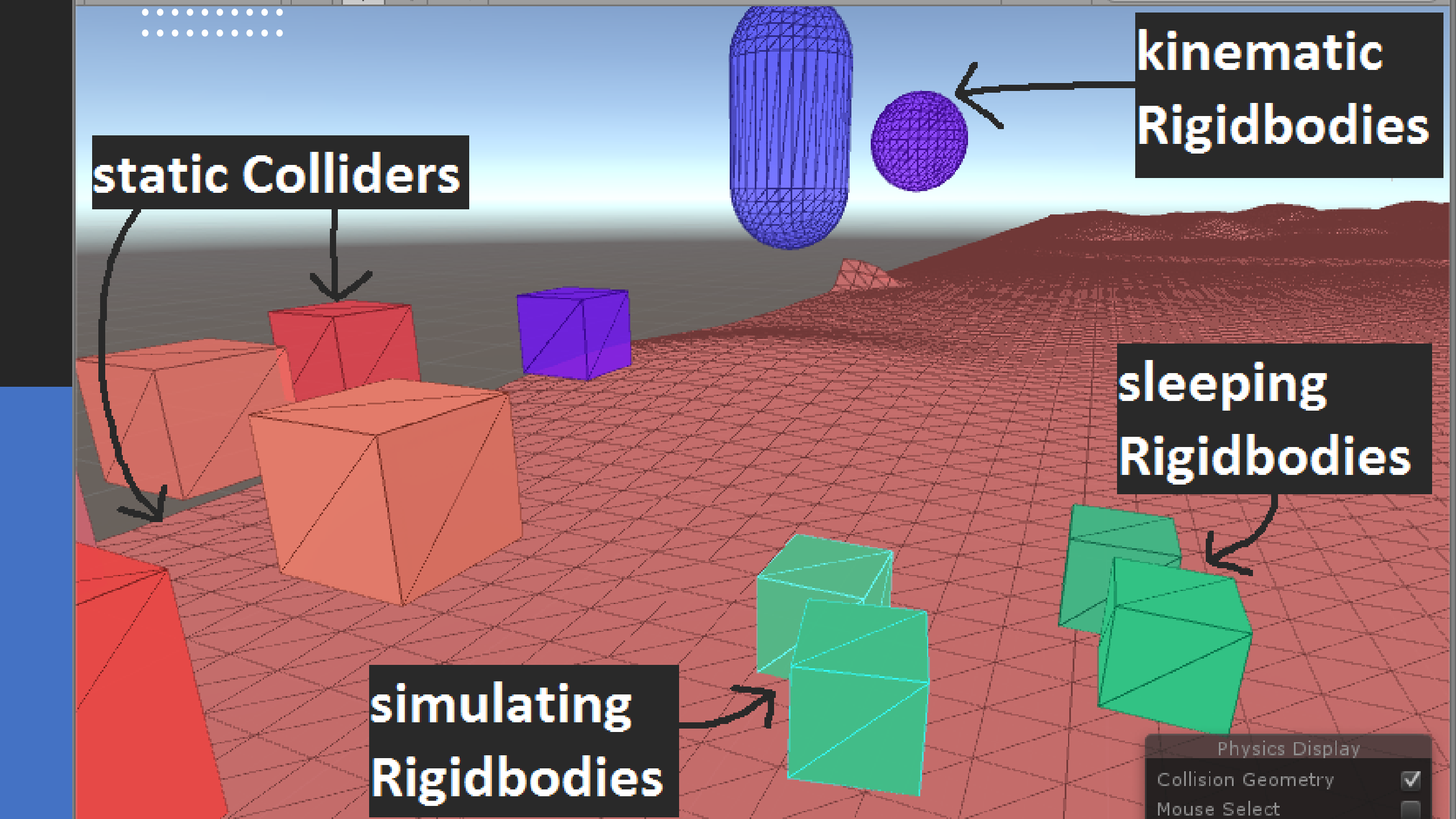
Unity de bir objenin diğer objeye çarpmasını anlamak için Collision veya Trigger kullanılır.

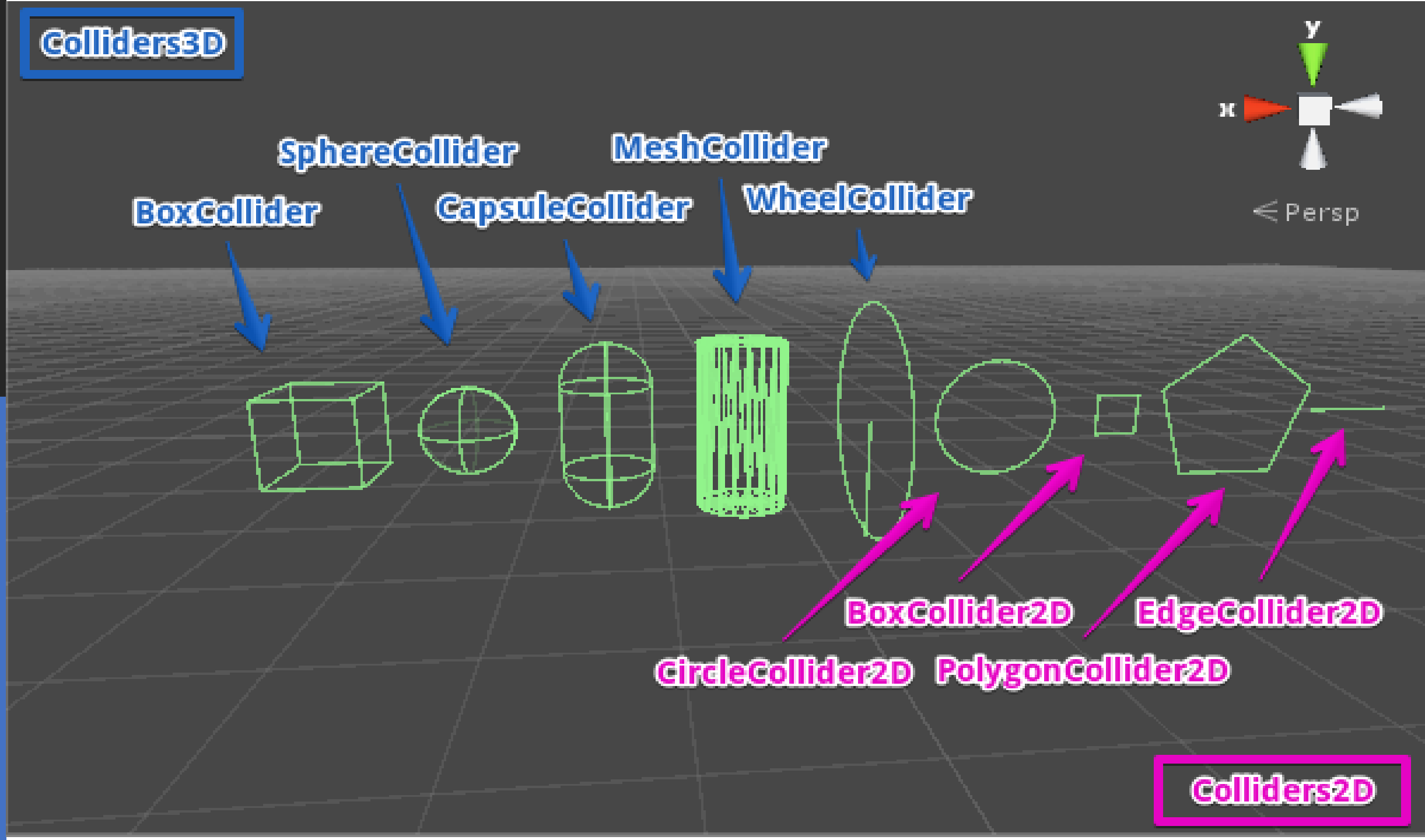
Bu ikisinin temel farkı Trigger’de objemizin objelerin içinden geçebilmesidir.

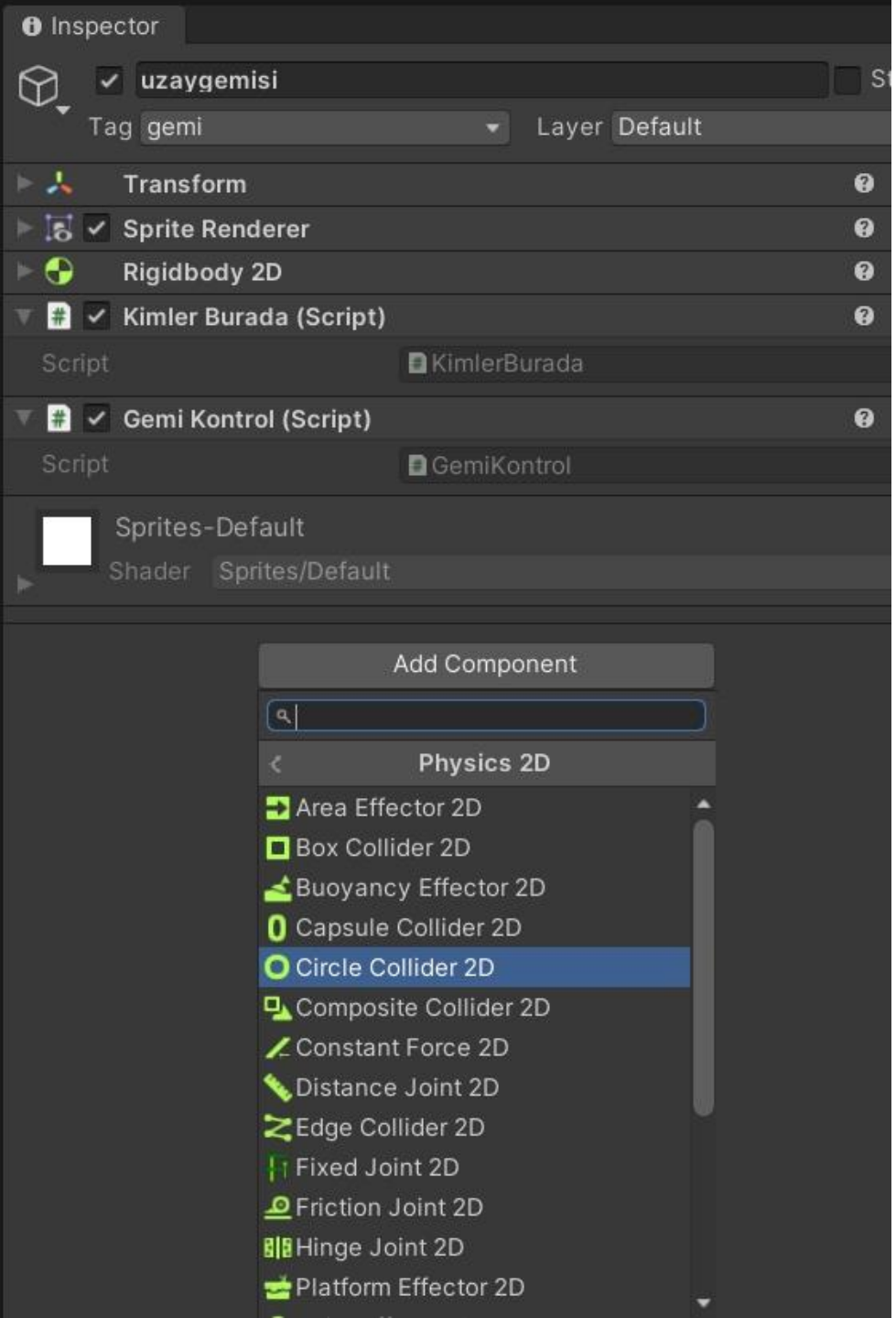
Collision

- 2D oyun projesinde Collision çalışması için oyun objesinde RigidBody 2D component’inin ekli olması gerekir.
- Çarpışacak olan 2 oyun objesinde de 2D Collider olması gerekmektedir.




kinematic
Rigidbodies
static Colliders
sleeping
Rigidbodies
simulating
Rigidbodies
Physics Display
Collision Geometry
Mouse Select

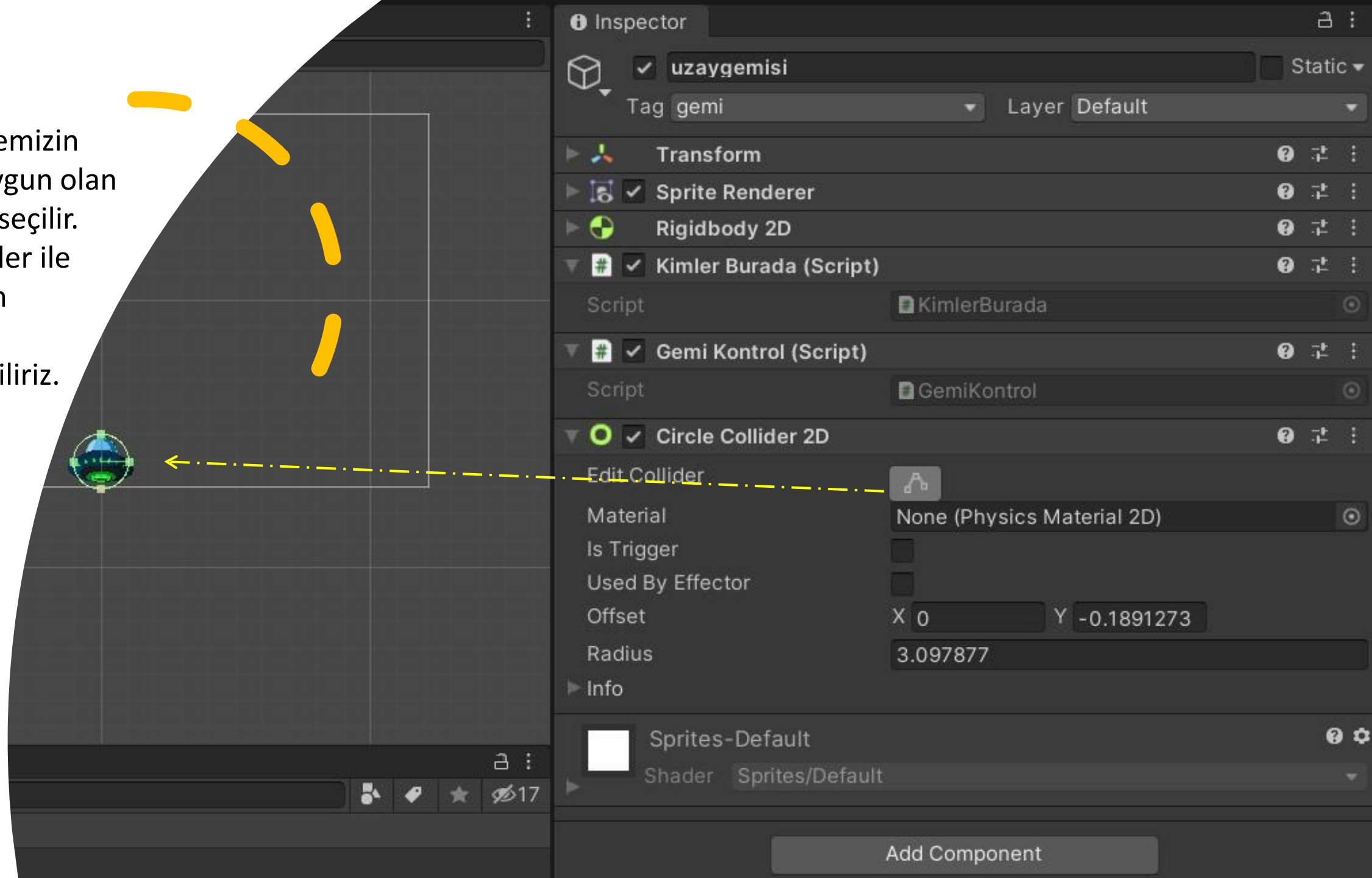

Colliders3D
SphereCollider
MeshCollider
BoxCollider
CapsuleCollider
WheelCollider
y
x
Persp
BoxCollider2D
EdgeCollider2D
CircleCollider2D
PolygonCollider2D
Colliders2D

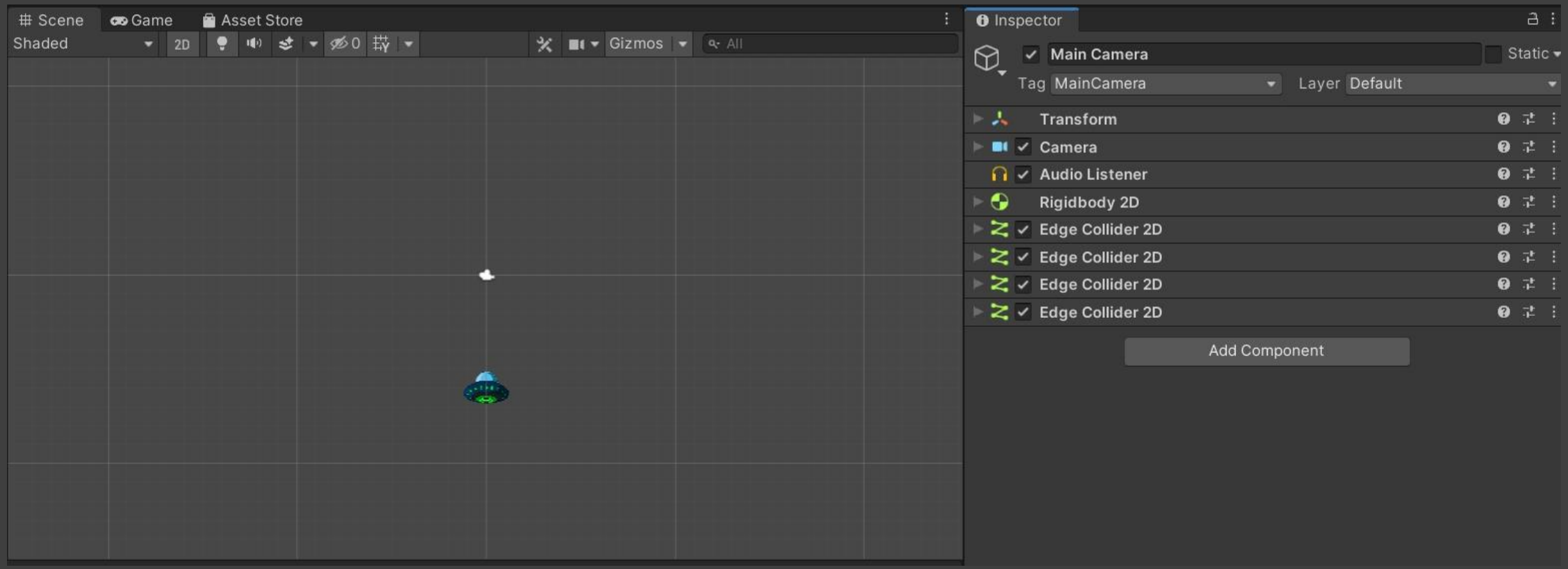


GameObject'e Collider Eklemek

- Inspector penceresinden Add Component ile Circle Corrider 2D component'ini seçerek ekleyebiliriz.

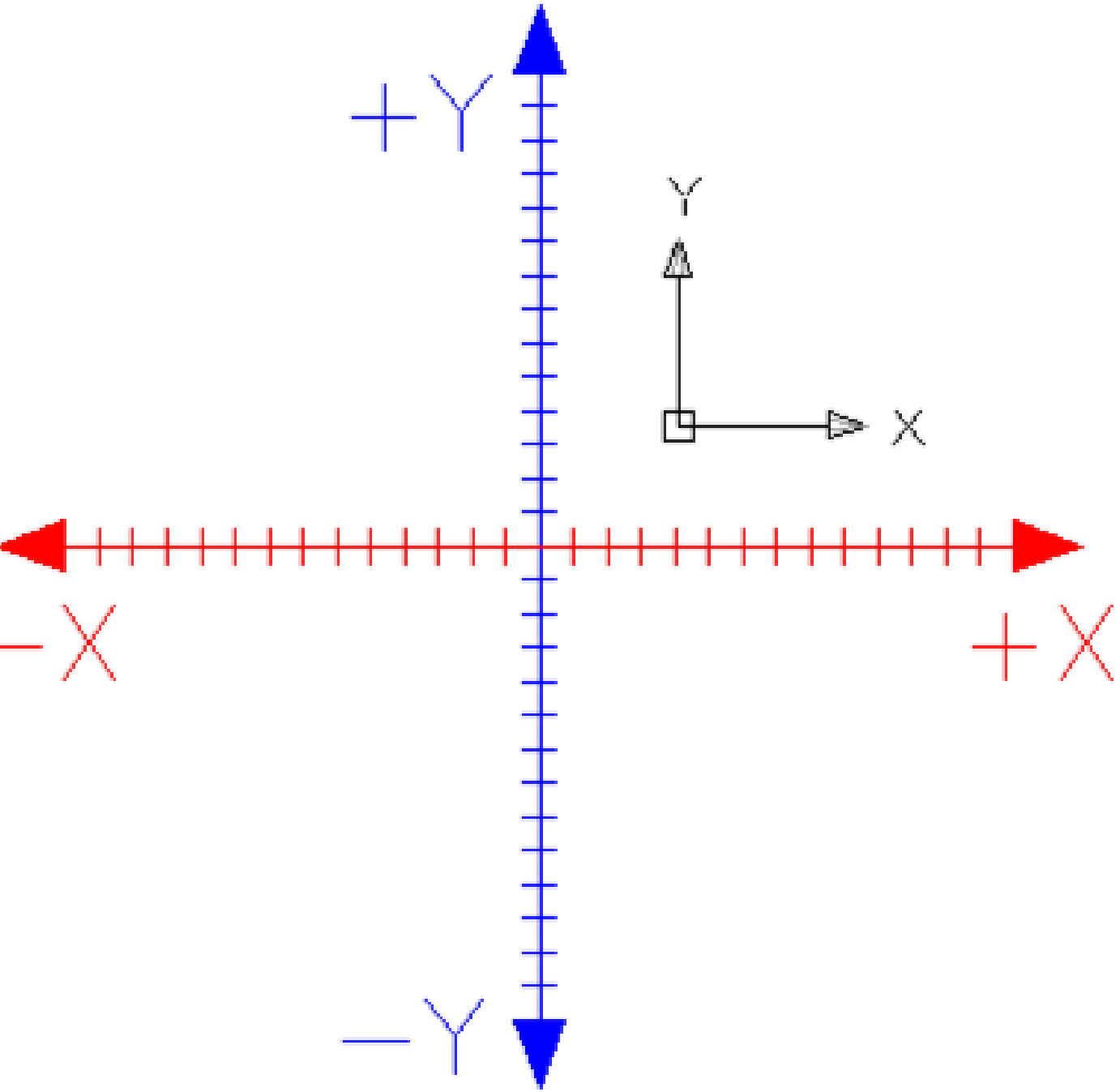
- Oyun objemizin şekline uygun olan Collider'ı seçilir.
- Edit Collider ile Collider'ın sınırlarını ayarlayabiliriz.



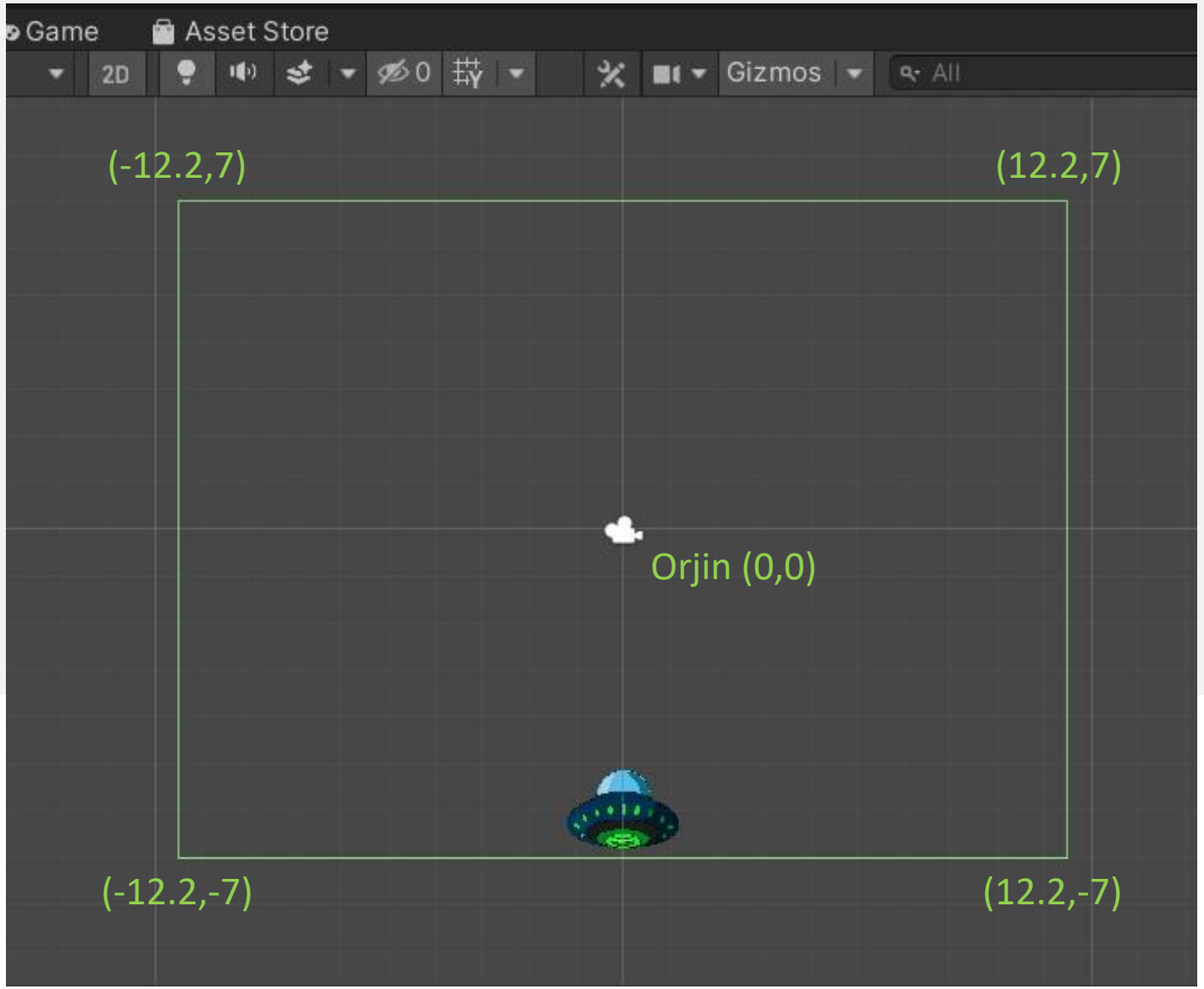


MainCamera'ya Collider Eklemek

- Tüm sınırlarını çizebilmek adına 4 tane Egde Collider 2D ekleyebiliriz.



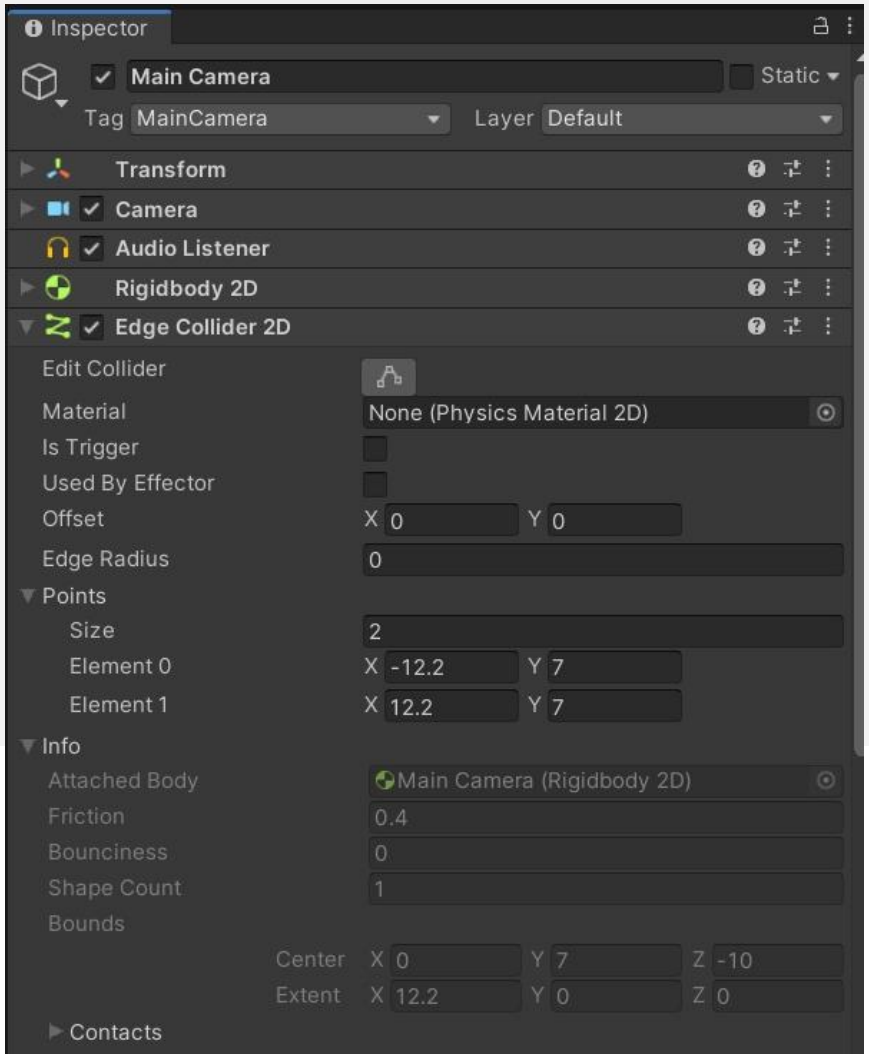
Koordinat Düzlemi



Edge Collider Points (x,y)

Koordinatlar
Main Camera **size=7** durumuna göre oluşturulmuştur.

Edge Collider - Üst Sınır



Edge Collider - Alt Sınır

Üst ve Alt Sınırlara Edge Collider Oluşturulması

Edge Collider - Sol Sınır

Edge Collider - Sağ Sınır

Sol ve Sağ Sınırlara Edge Collider Oluşturulması

Scene Game Asset Store

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

GemiKontrol Script-1

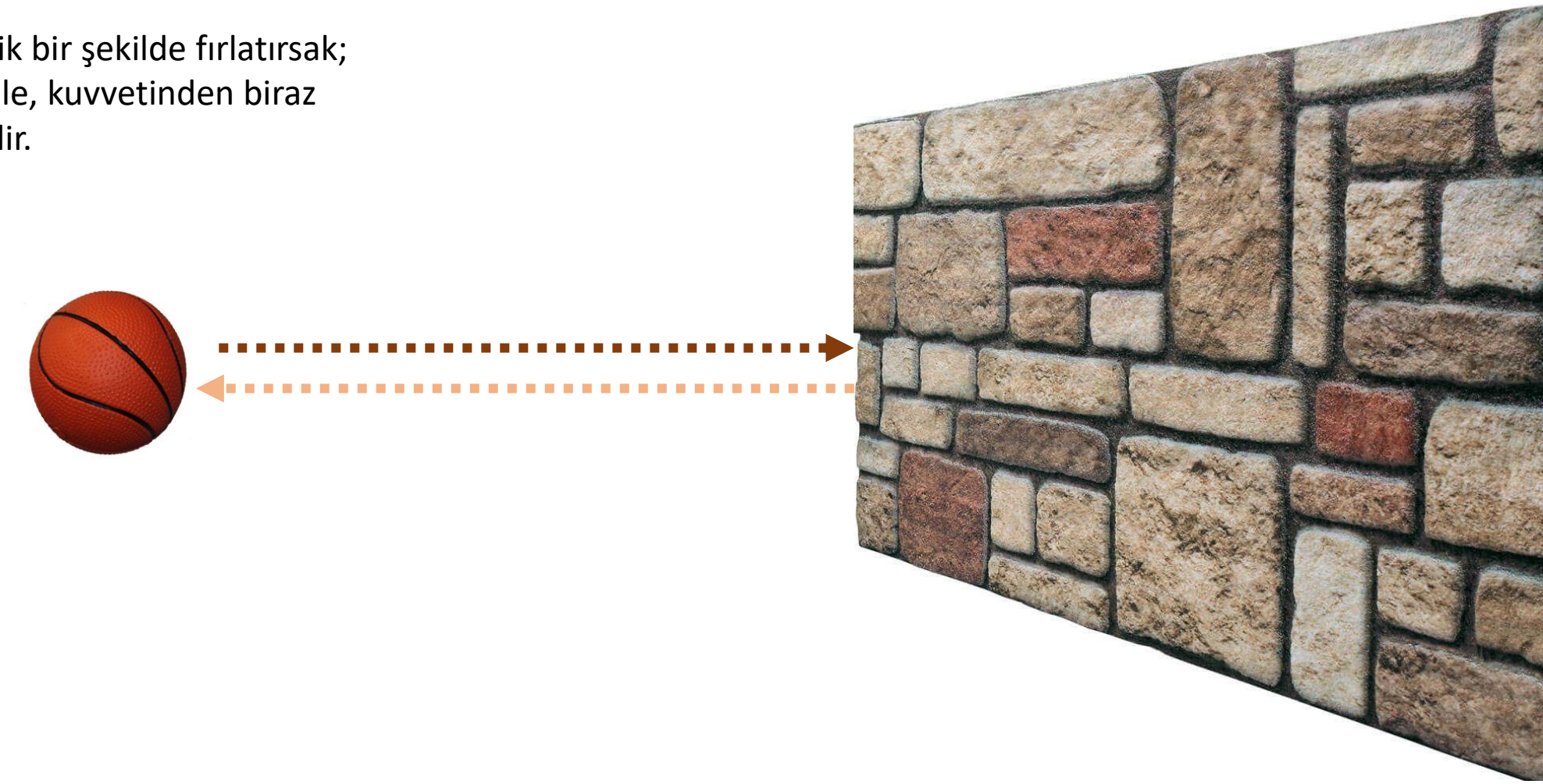
```
void Start()
    {
        //Uzay gemisini hareket ettir

        Rigidbody2D myRigidbody2d = GetComponent<Rigidbody2D>();
        myRigidbody2d.AddForce(new Vector2(0,5),ForceMode2D.Impulse);

    }
```

GemiKontrol Script-2

```
void Start()
    {
        //Uzay gemisini hareket ettir

        Rigidbody2D myRigidbody2d = GetComponent<Rigidbody2D>();
        myRigidbody2d.AddForce(new Vector2(-5,5),ForceMode2D.Impulse);

    }
```

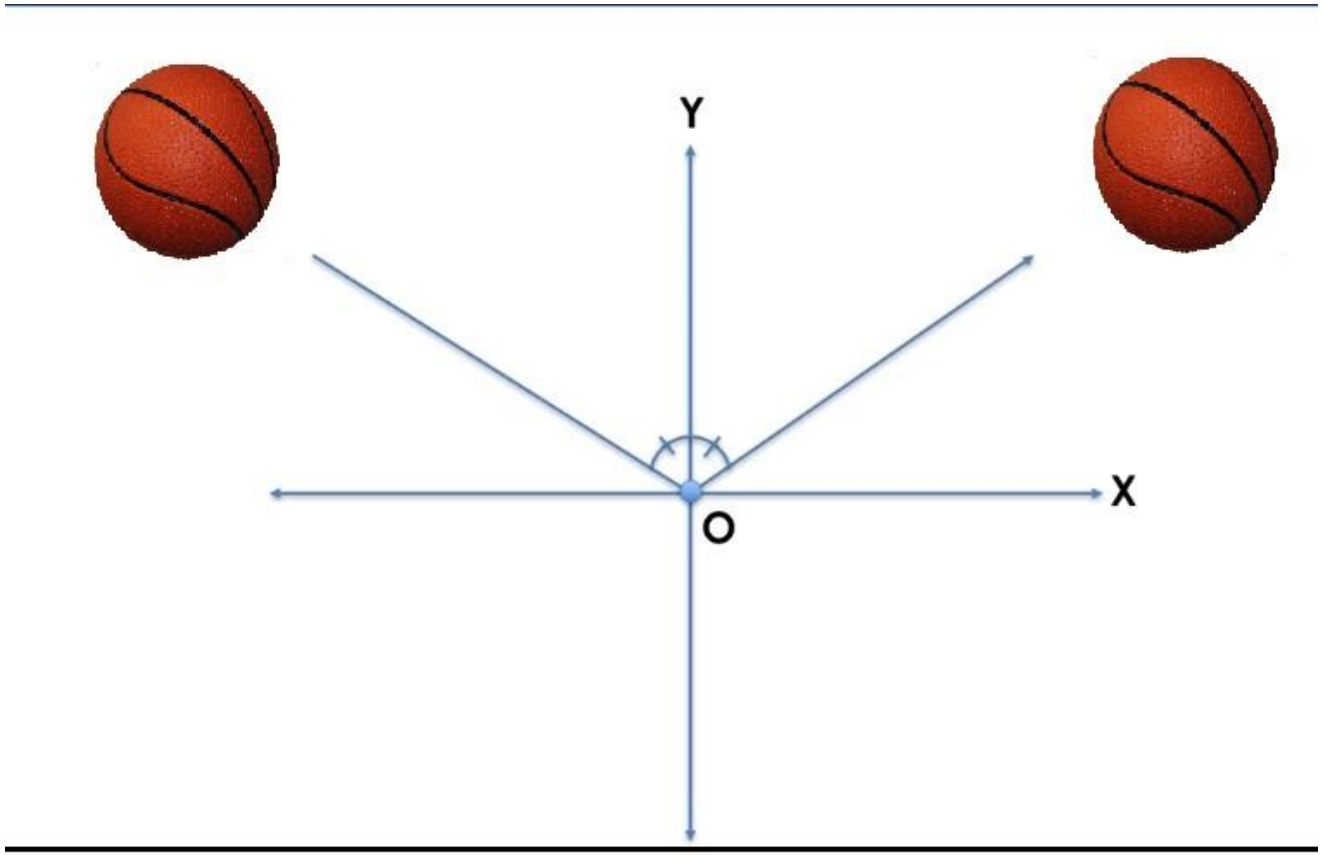
Fizik Materyalleri

Eğer topu duvara tam dik bir şekilde fırlatırsak; hemen hemen aynı açı ile, kuvvetinden biraz kaybederek bize geri gelir.



Fizik Materyalleri

- Topu açısal bir şekilde duvara fırlatırsak; çarpma noktasından ayna görüntüsü olacak şekilde hareket eder.



Fizik Materyalleri

?

- Duvara farklı nesneler attığımızda bize nasıl tepki verir?
  - Örneğin; peluş bir oyuncak attığımızda bize basket topuna verdiği tepkiden daha farklı bir tepki verecektir.
- Bu farklılığı sağlayan nedir?
- Oyunumuzun içerisinde farklı olan oyun objelerinin bu şekilde birbirinden farklı karakteristikler göstermesini nasıl sağlarız?
  - Burada fizik materyalleri devreye giriyor.

Fizik Materyalleri

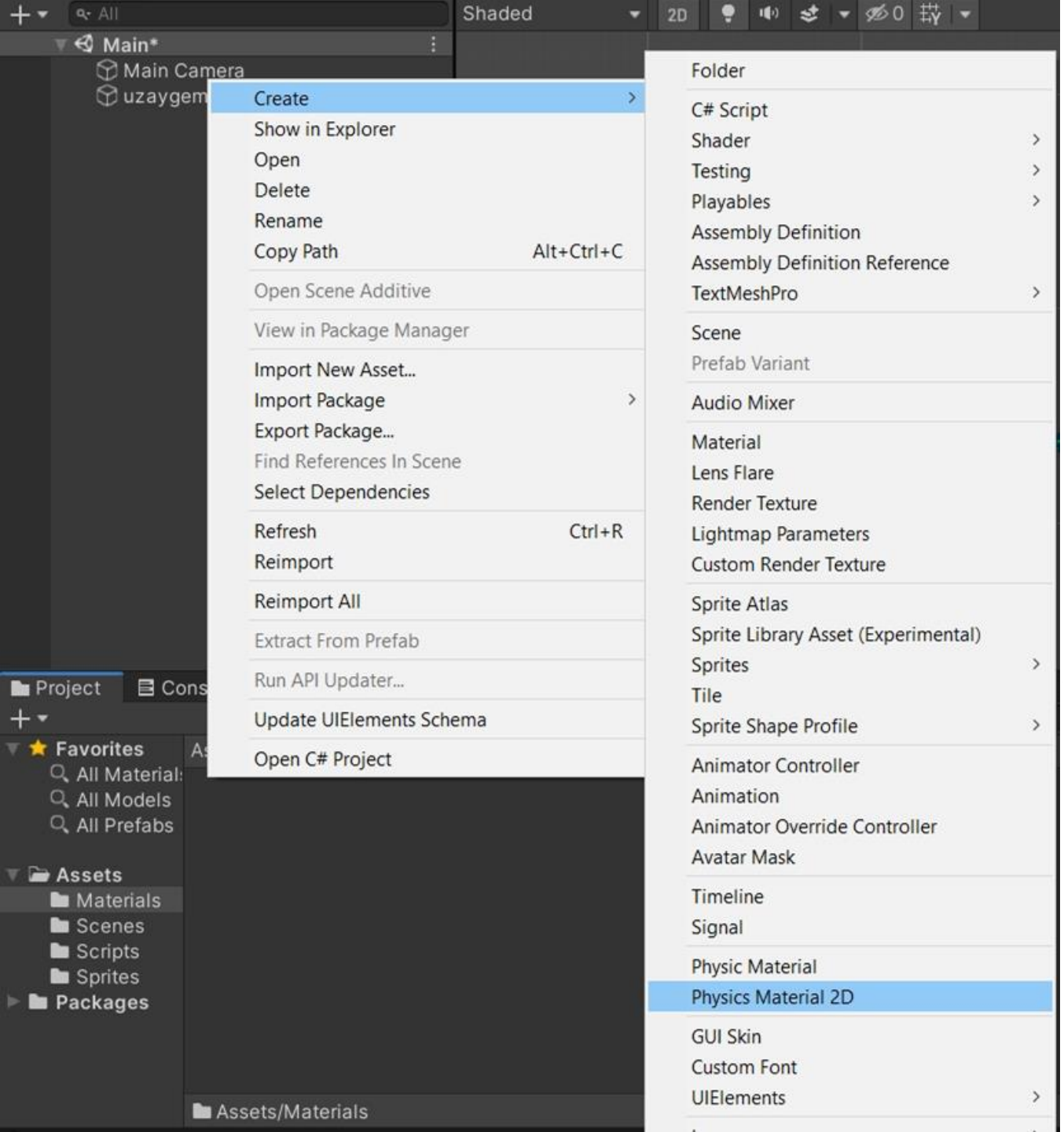
- Fizik materyalleri oyun objemizin **sürtünme** ve **sıçrama** gibi fiziksel özelliklerini ayarlamamızı sağlarlar.
  - Friction (Sürtünme)
  - Bounciness (Sıçrama)

Fizik Materyalleri

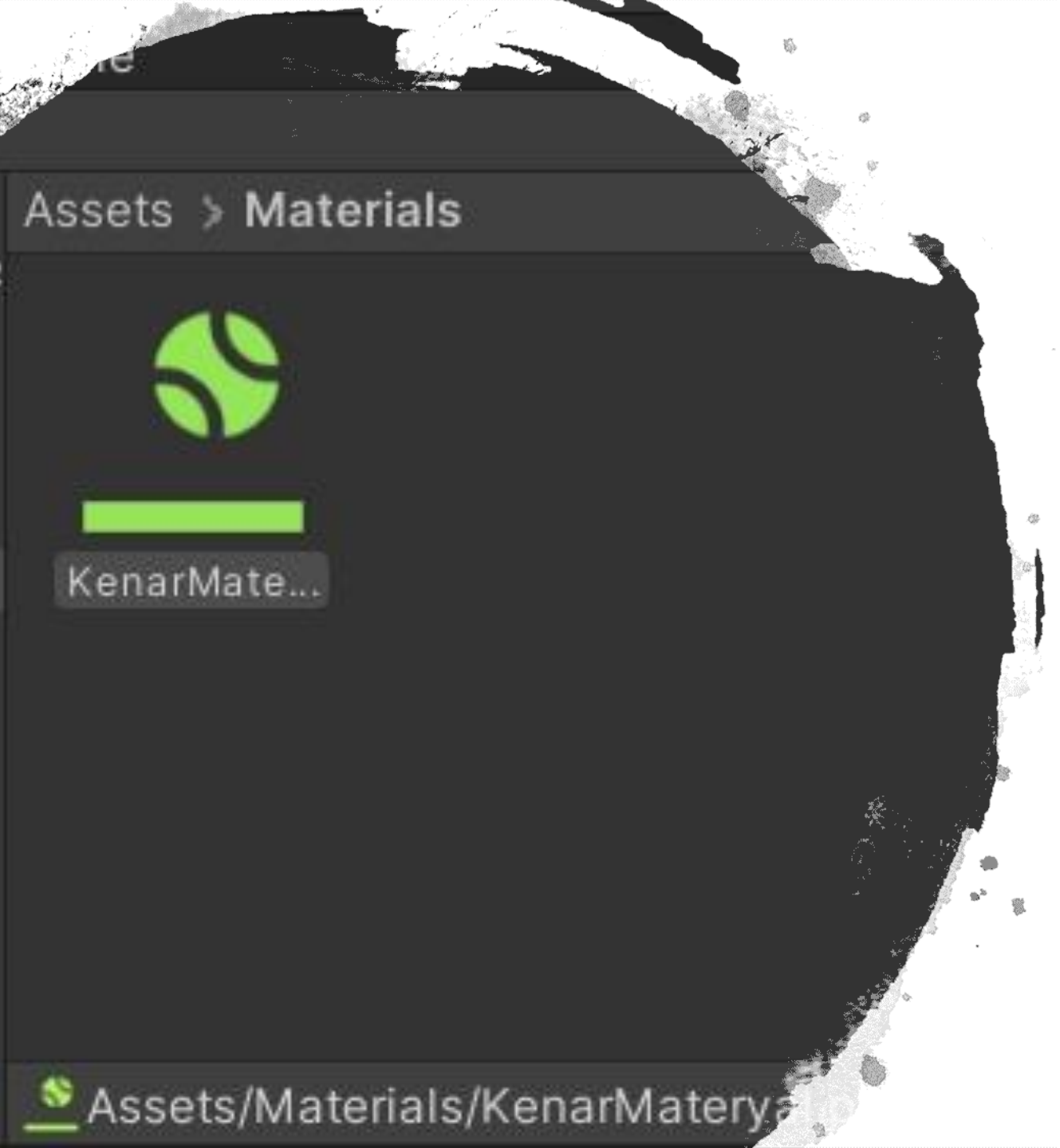
- Materyaller (Materials) Collider'lar ile beraber kullanılırlar.
- Unity'nin içerisinde bulunan fizik motoru fizik materyalleri uygulanmış Collider'ları algıladıktan sonra eğer o objelere uygulanan bir kuvvet varsa materyallerin karakteristiğine göre o kuvvetin değerini ya da yönünü değiştirerek cismin hareketine devam etmesini sağlar.
- Çarpışma algılama işleminden sonra oyunda bir takım değişiklikler yapabiliriz.
  - Örneğin uzay gemisinin sahneden yok olması ya da kenarlara her çarptığında hasar alması gibi.

Materials klasörünü oluşturalım.

Materials

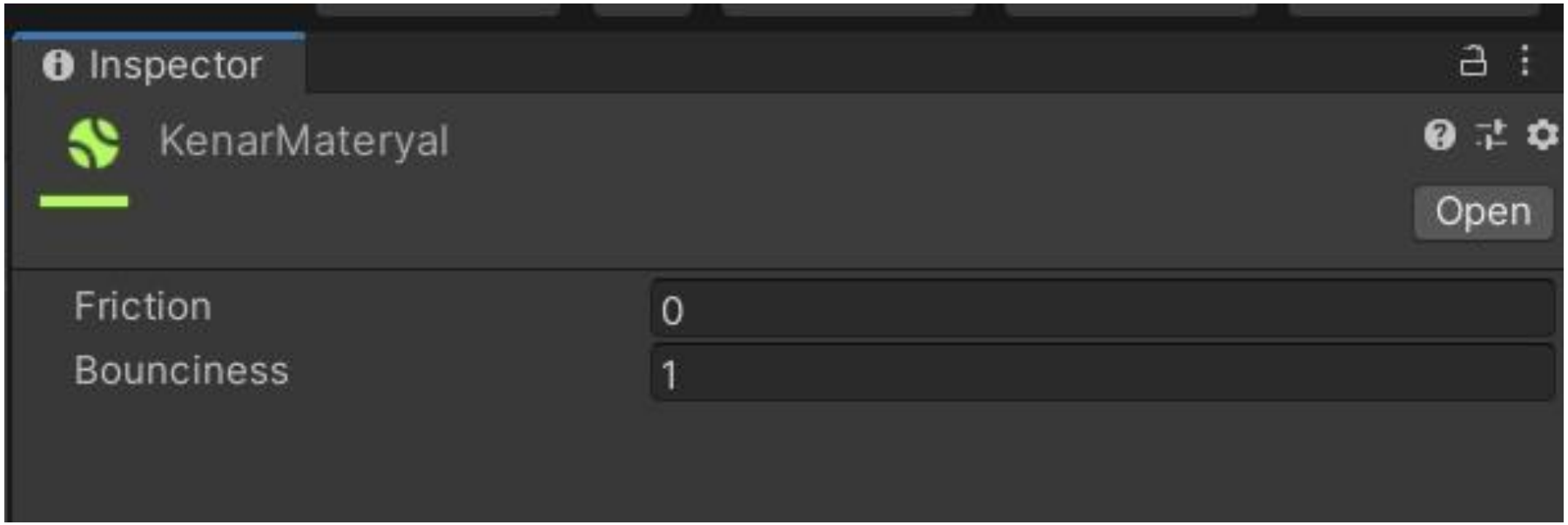


Materials klasörüne Create seçenekleri ile **Physics Materials 2D** ekliyoruz.



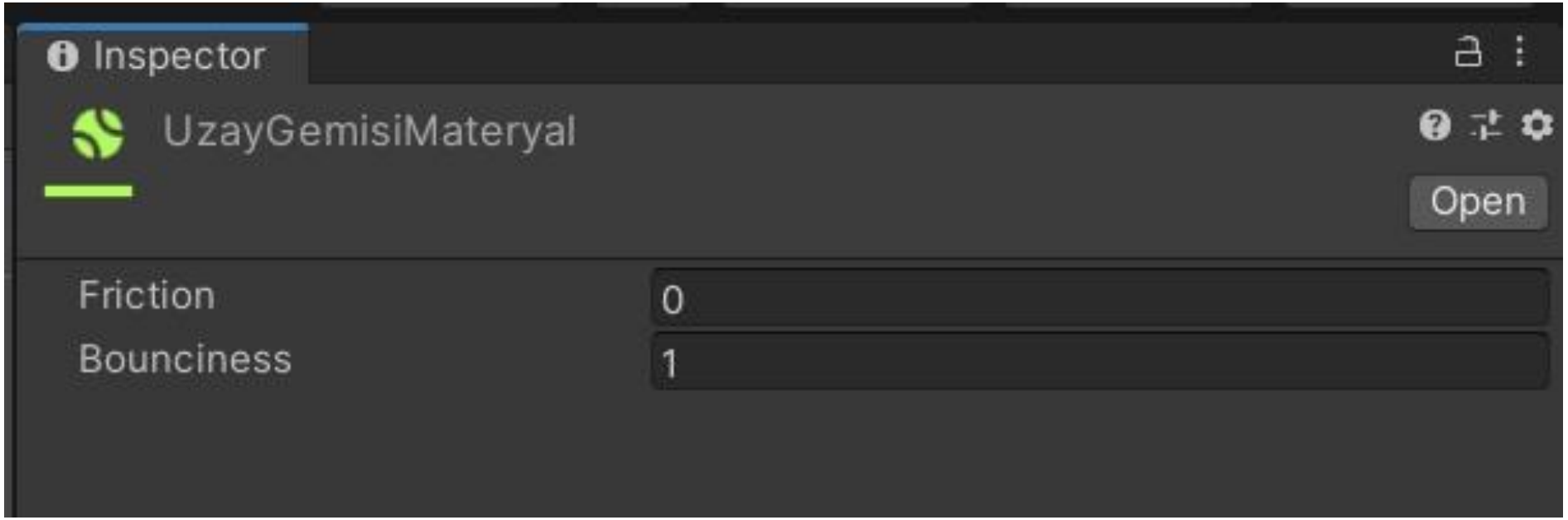
- Önce kenarlarda kullanacağımız materyali oluşturalım.
- Onun için adına **KenarMateryal** diyebiliriz.

Kenar Materyal



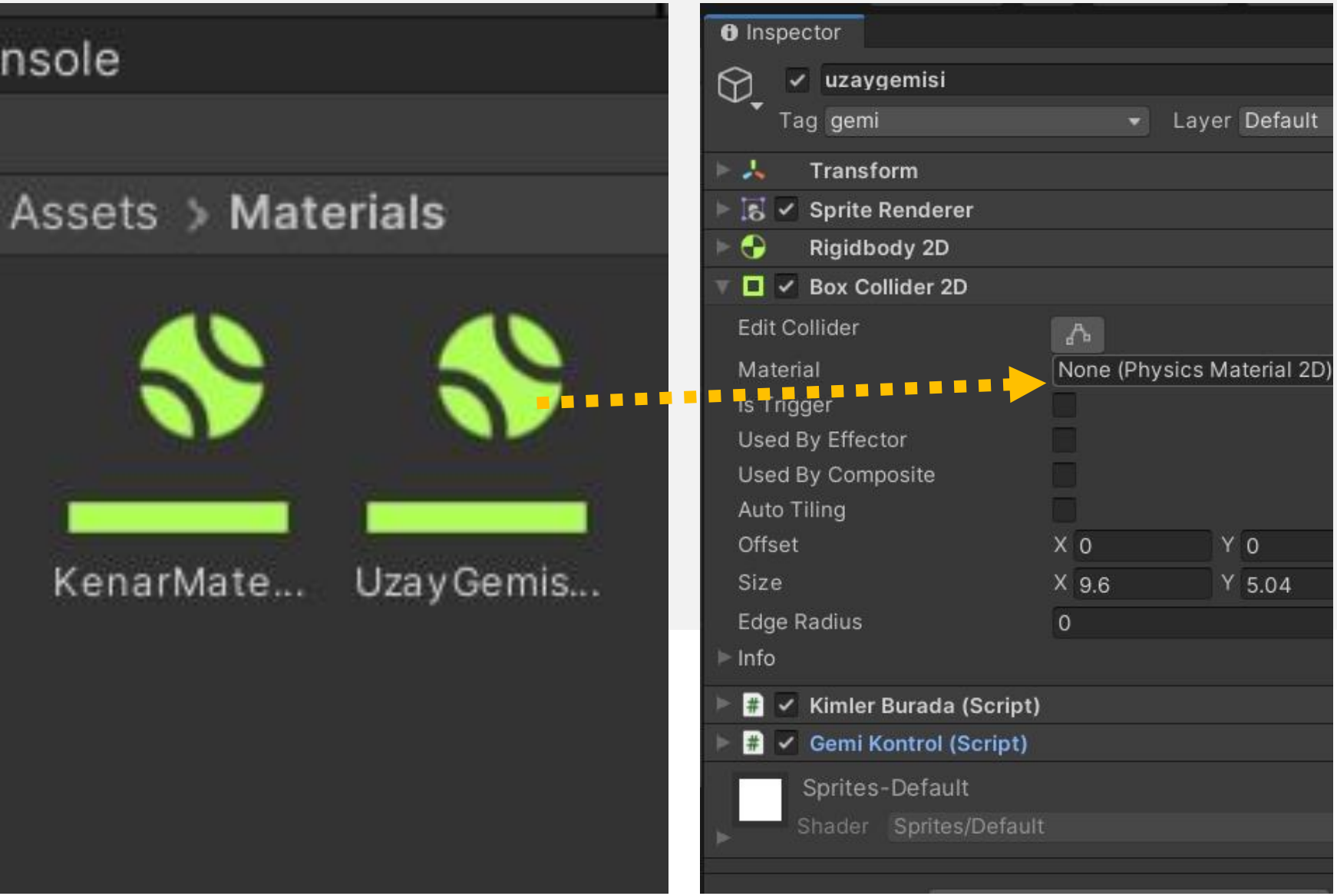
Kenar Materyal Inspector

- Oyunda sürtünmeyi dikkate almayacağımız için KenarMateryal seçiliyken Friction (Sürtünme) değerini 0 yapıyoruz.
- Bounciness (Sıçrama) değerini 1 yapacağız.



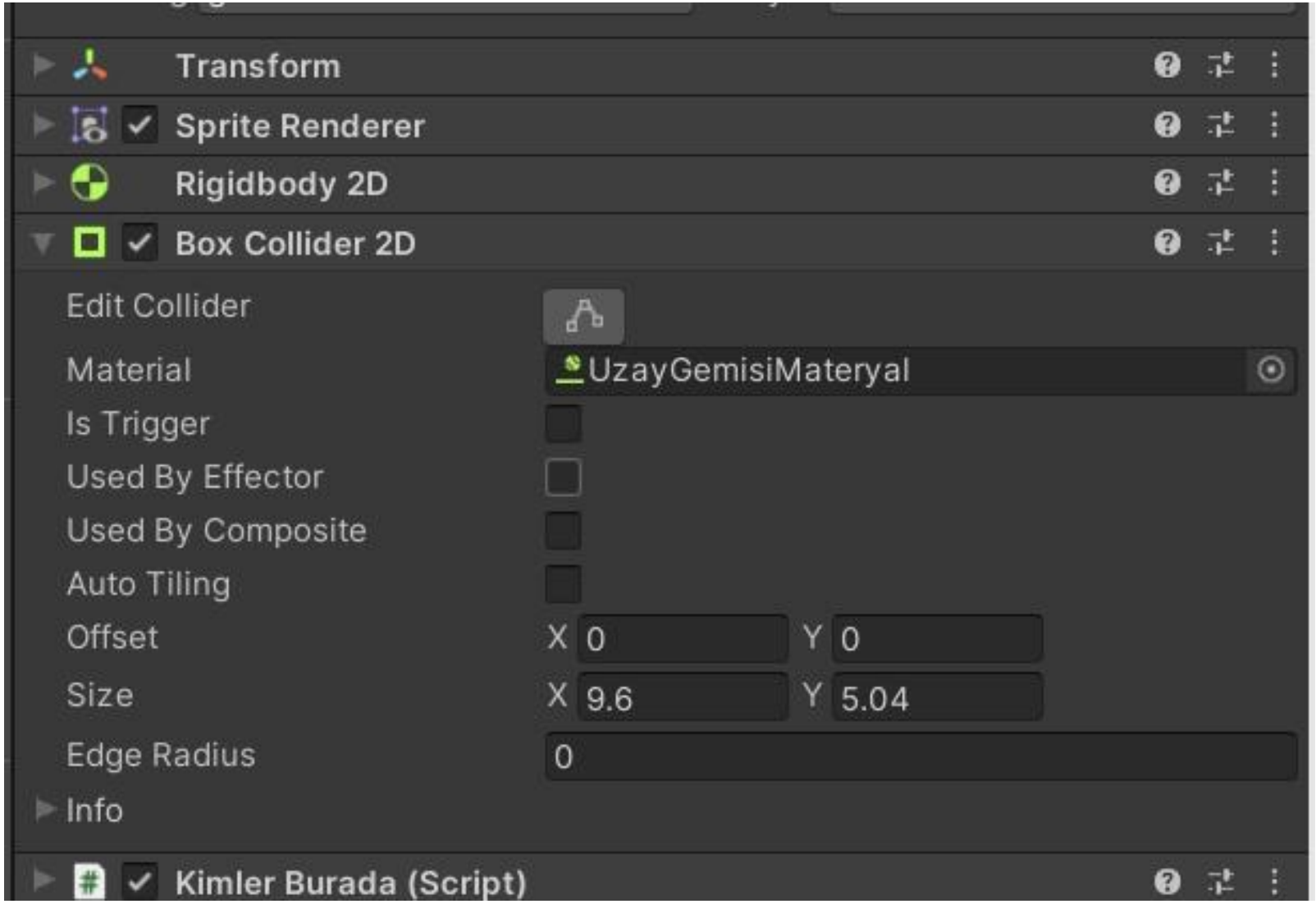
Uzay Gemisi Materyal Inspector

- UzayGemisiMateryal değerleri için de aynı ayarları yapıyoruz.
  - Fiction → 0
  - Bounciness →1

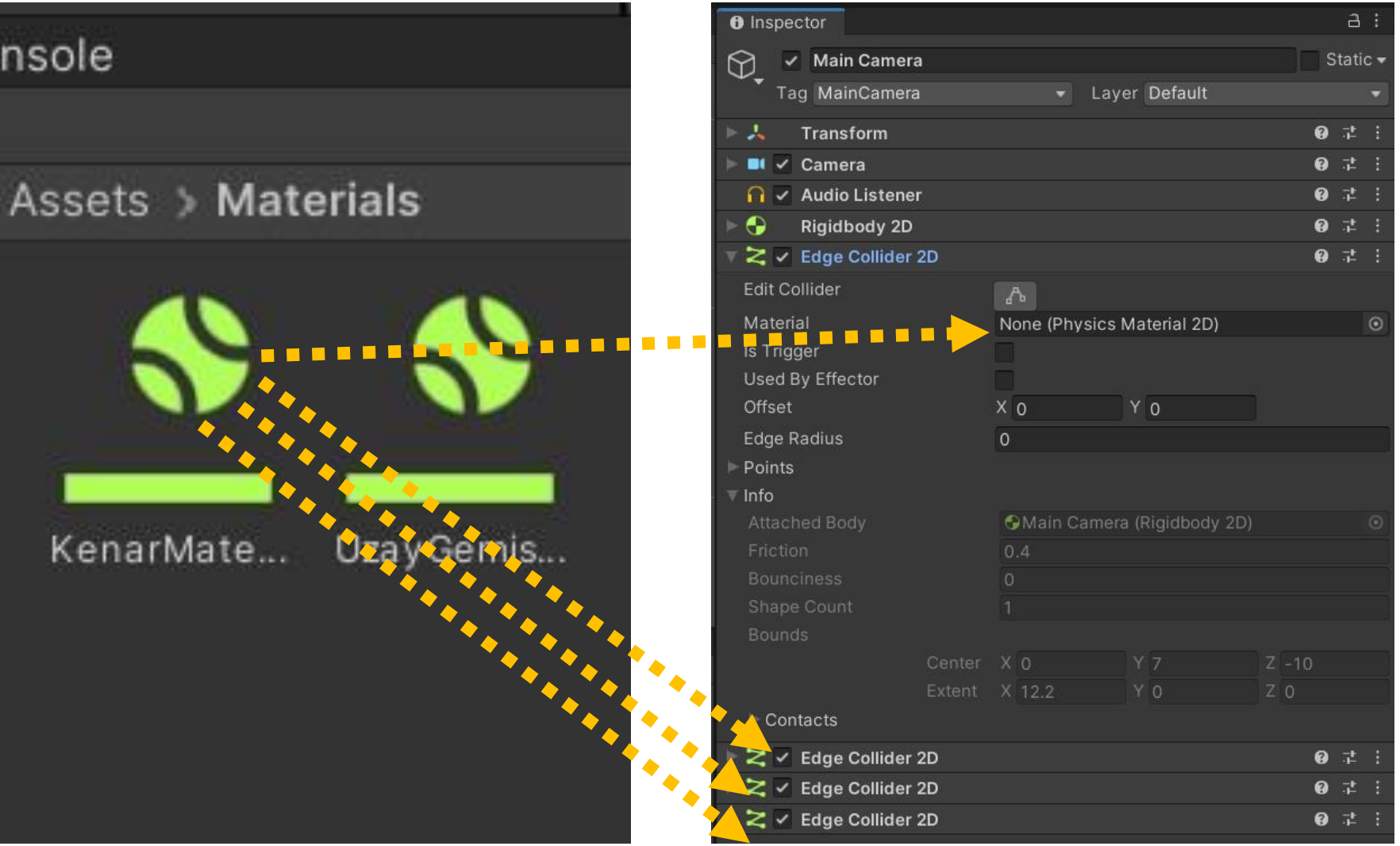


Uzay Gemisi Material

Materyalimizi sürükleyerek uzaygemisi objemizin Material kısmına bırakırız.



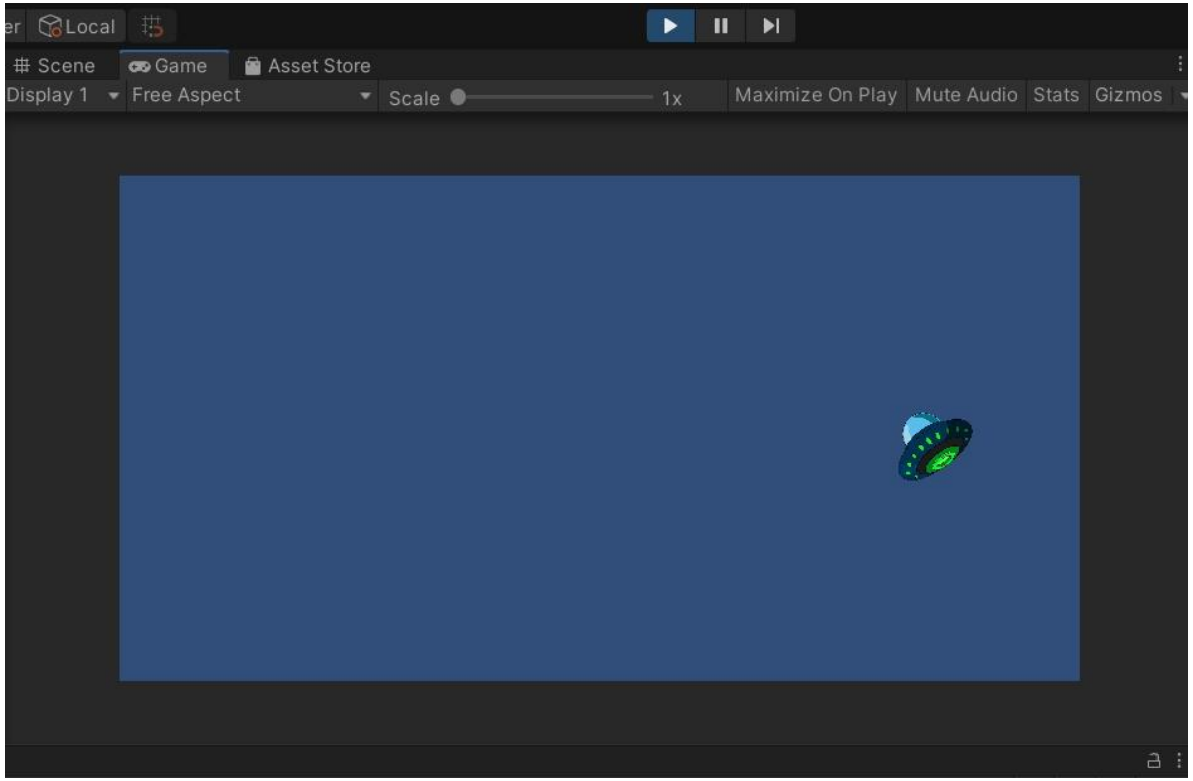
Materyali ekledik.



Main Camera Material

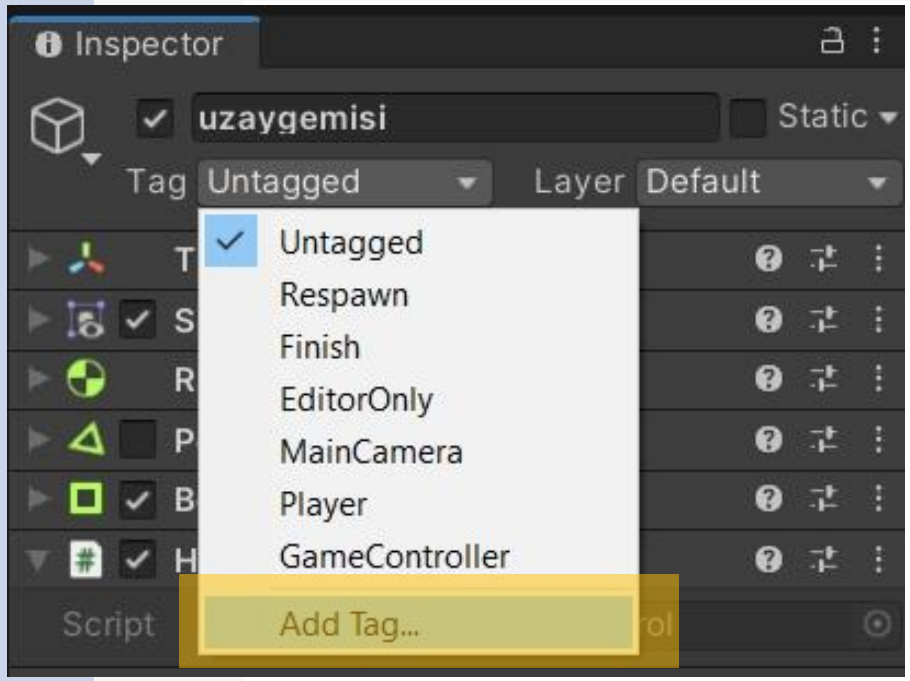
Materyalimizi sürükleyerek MainCamera objemizin Material kısmına bırakırız.
Bu işlemi tüm Edge Collider'lar için yapmamız gerekmektedir.

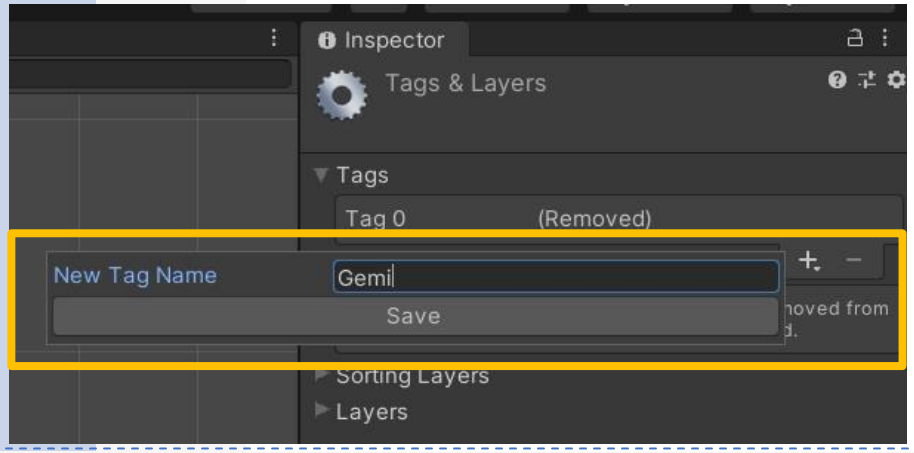
Materyali ekledik.



Game Play

- Play tuşuna basıp oyunu çalıştırdığımızda uzay gemimiz çarptığı açıya göre kuvvetin yönü değişerek hareketine devam eder.
- Ortamda bir sürtünme olmadığı için bu hareket biz oyunu durduruna kadar devam eder.



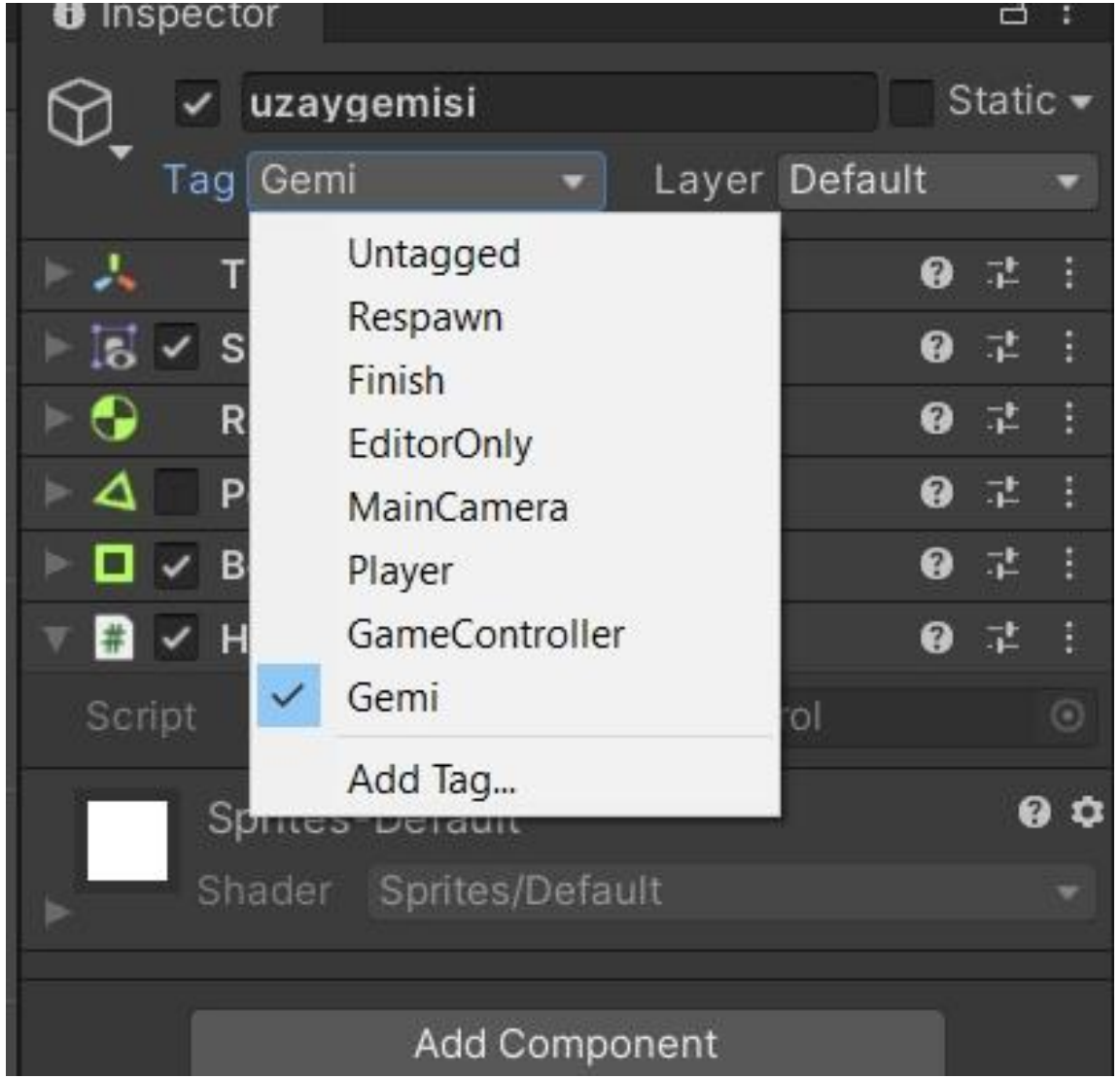


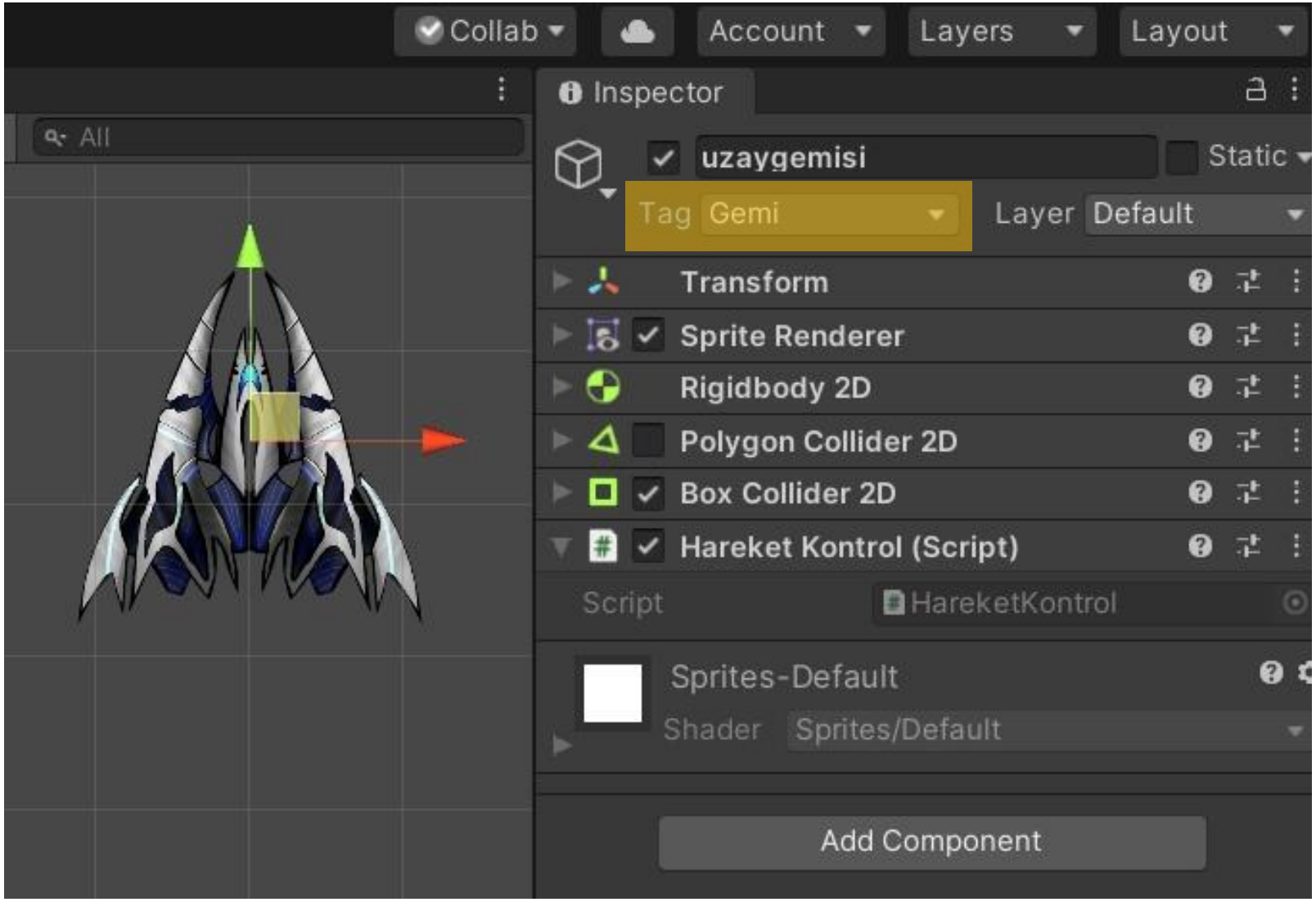
Tag Değeri Eklemek

- Mevcut Tag değerleri arasında objeye atamak istediğimiz Tag değeri mevcut değil ise aşağıdaki adımlar izlenmelidir.
  - Add Tag… seçeneğine tıklanır.
  - Karşımıza gelen penceredeki + düğmesine tıklanır.
  - Ekrana gelen pencereden Tag ismi yazılarak Save düğmesine tıklanır.
  - Eklemiş olduğumuz bu Tag değeri listeden seçilerek objemize aktarılması sağlanır.

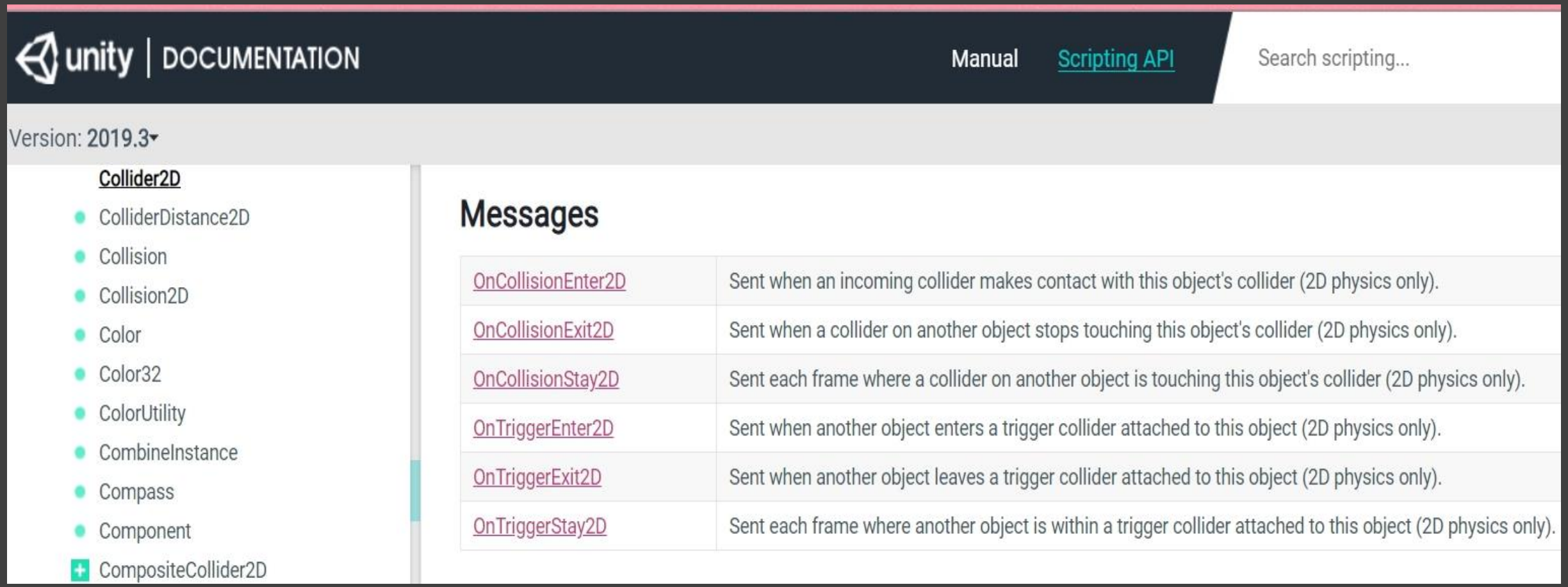
Oyun Objesine Tag Değeri Atamak

- Inspector penceresinde yer alan Tag seçenekleri arasından objeye atamak istediğimiz Tag değeri seçilir.





«uzaygemisi» isimli objemiz bu seçimden sonra «Gemi» tag değerine sahip olur.



Unity Documentation Scripting API

- Collider'ların herhangi bir çarpışma anında haber vermesi Collider2D Messages kısmında yer alıyor.

Collider2D.OnCollisionEnter2D(Collision2D)

Parameters

other	The Collision2D data associated with this collision.

Description

Sent when an incoming collider makes contact with this object's collider (2D physics only).

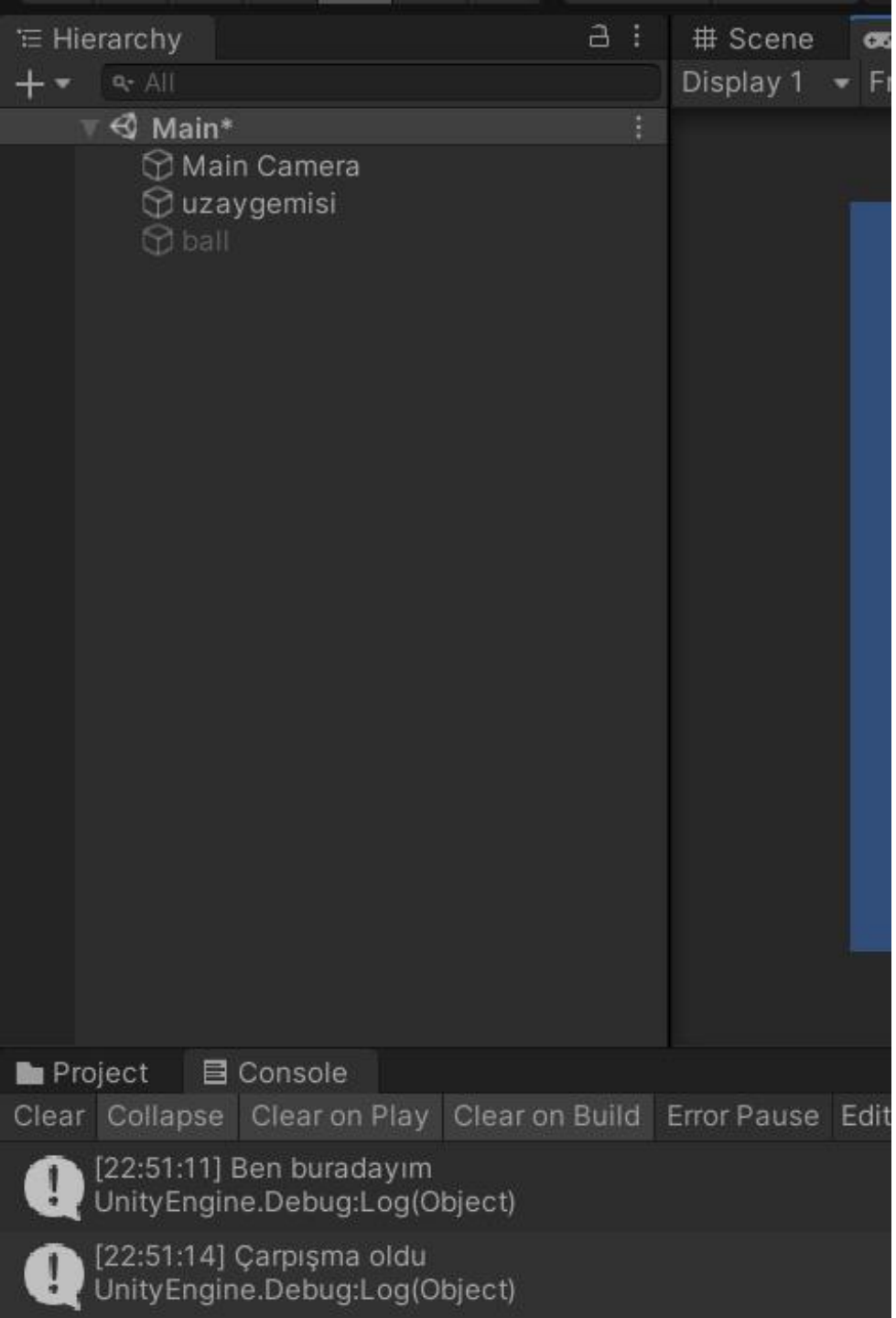
Further information about the collision is reported in the Collision 2D parameter passed during the call.

Notes: Collision events will be sent to disabled MonoBehaviours, to allow enabling Behaviours in response to collisions.

See Also: Collision2D class, OnCollisionExit2D, OnCollisionStay2D.

```
using UnityEngine;

public class Example : MonoBehaviour
{
    void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "Enemy")
        {
            collision.gameObject.SendMessage("ApplyDamage", 10);
        }
    }
}
```

onCollisionEnter2D
mesaj türünde
2 boyutlu oyunlarda
herhangi bir çarpışma
olduğu zaman mesaj
gönderileceğini
belirtiyor.



«Çarpışma oldu» Log Mesajı Script

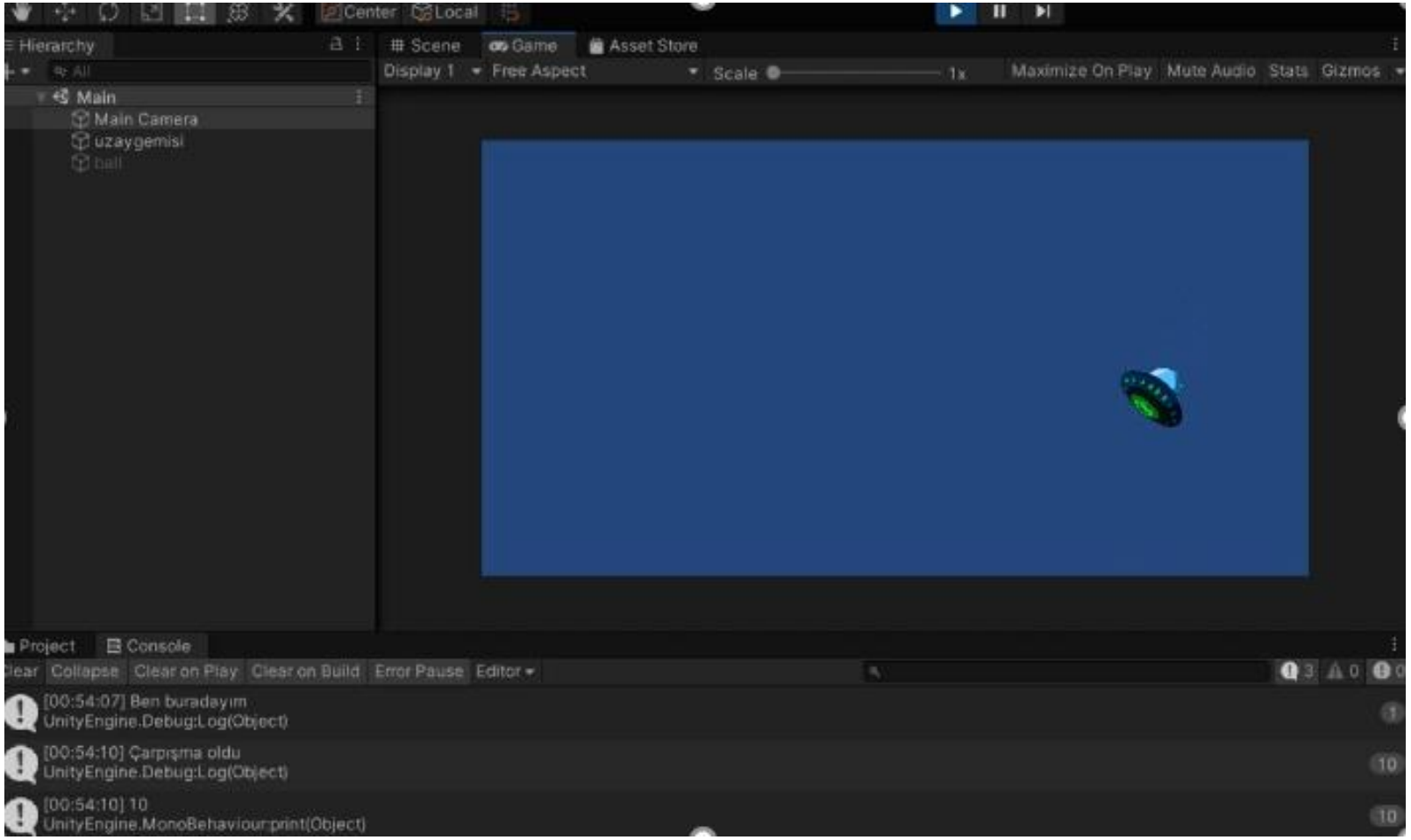
- Olası bir çarpışmadan geminin haberinin olması için geminin Script'ine onCollisionEnter2D metodunu yazmamız gerekiyor.

```
//Çarpışma olduğunda ekrana "Çarpışma oldu" Log'unu yazdıralım.
    private void OnCollisionEnter2D(Collision2D collision)
    {
        Debug.Log("Çarpışma oldu");
    }
```

Çarpışma Hasar Uygula Script

- Gemi her kenara çarptığında bir zarar vermesini sağlayalım.

```
private void OnCollisionEnter2D(Collision2D collision)
    {
        //Çarpışma olduğunda ekrana "Çarpışma oldu"
Log'unu yazdıralım.
        Debug.Log("Çarpışma oldu");
        //Eğer bu objenin tag’ı MainCamera ise
HasarUygula fonksiyonu tetiklensin.
        if (collision.gameObject.tag == "MainCamera")
        {
             gameObject.SendMessage("HasarUygula", 10);
        }
      }
    public void HasarUygula(int hasar)
    {
        print(hasar);
    }
```

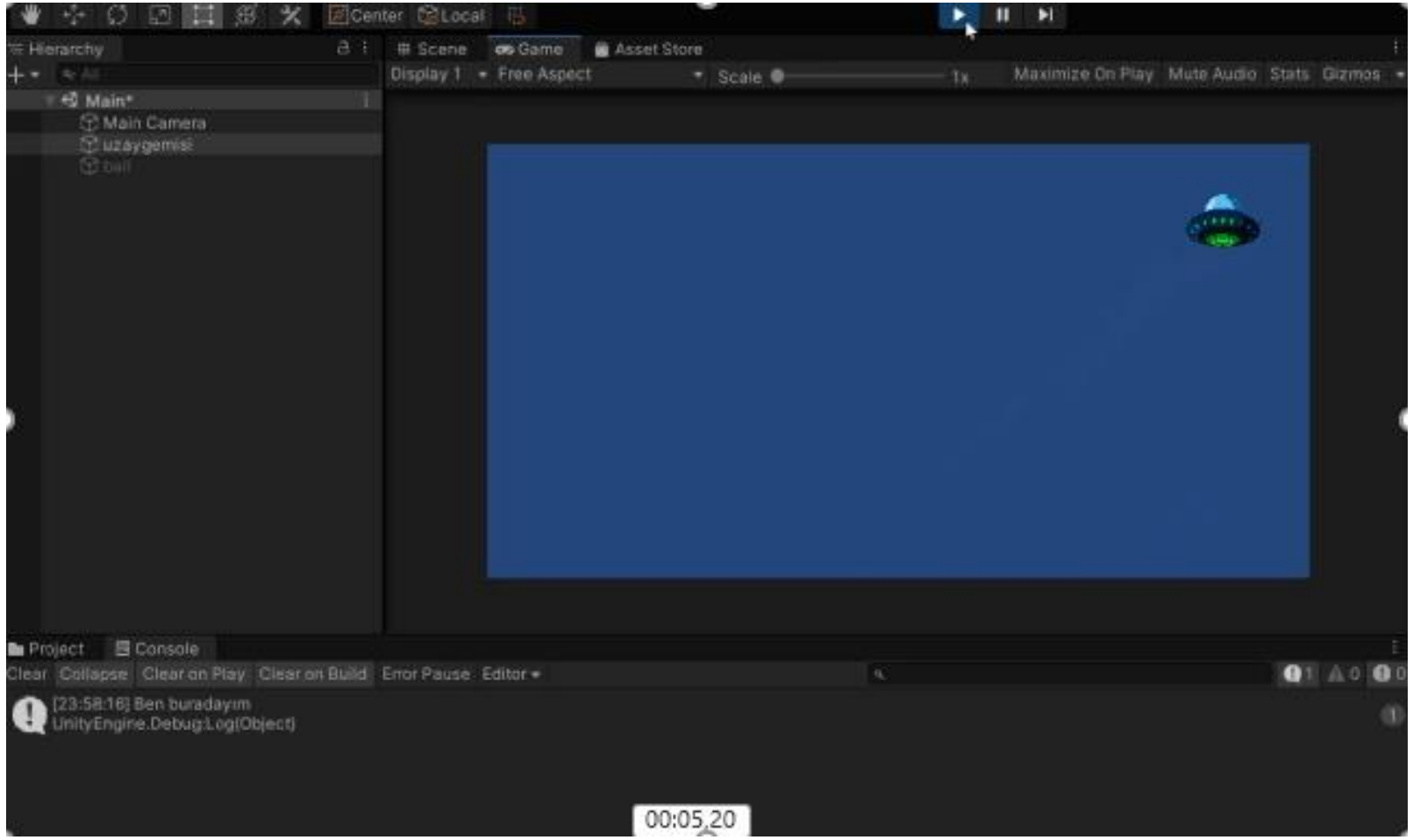


Çarpışma Hasar Uygula Script Play

Çarpışma Collider Component Pasif Yapma Script

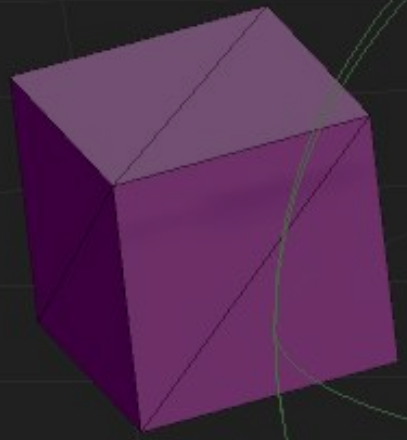
- Çarpışma olduğunda Main Camera collider componenti aktif ise pasif yapalım.

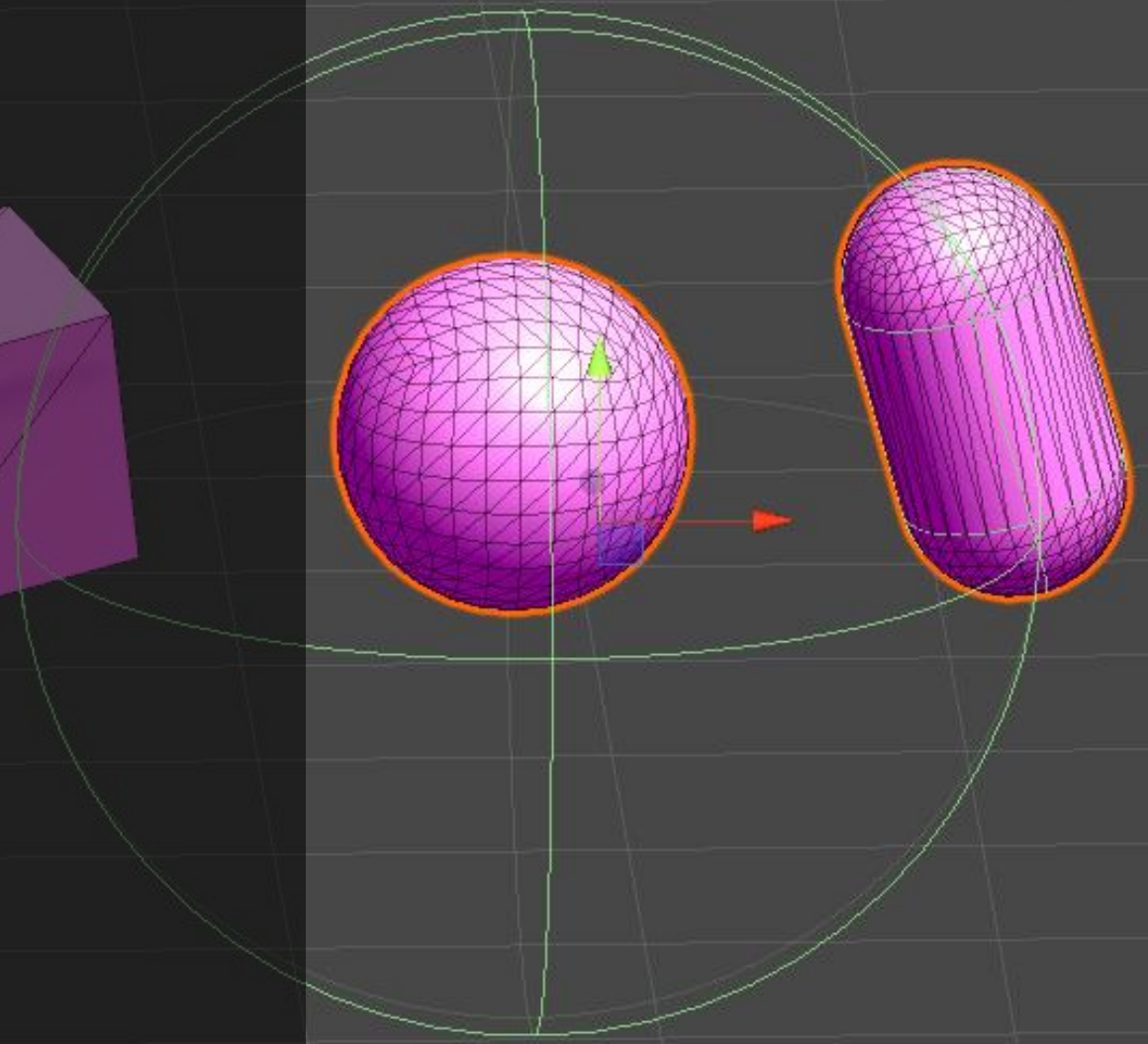
```
private void OnCollisionEnter2D(Collision2D collision)
    {
if (collision.collider == true)
        {
            // Çarpışma olduğunda Main Camera collider componenti aktif ise pasif yapılır.
            collision.collider.enabled = false;
        }
    }
```



Çarpışma Collider Component Pasif Yapma Script Play

Collision





- 3 tür fonksiyonu vardır.
  - onCollisionEnter
  - onCollisionStay
  - onCollisionExit

OnCollisionEnter2D

OnCollisionEnter bir objenin çarpma durumunda 1 kere çalışır.

```
private void OnCollisionEnter2D (Collision2D collision)
    {
        //Herhangi bir objeye çarpışma ilk olduğu anda çalışır
        if (collision.gameObject.tag == "MainCamera")
        {
            //Main Camera tag'lı objeye çarptığında çalışır
            Debug.Log("Main Camera ile çarpıştı");
        }
        if (collision.gameObject.name == "MainCamera")
        {
            //MainCamera isimli bir objeye çarptığında çalışır
            Debug.Log("Main Camera ile çarpıştı");
        }
    }
```

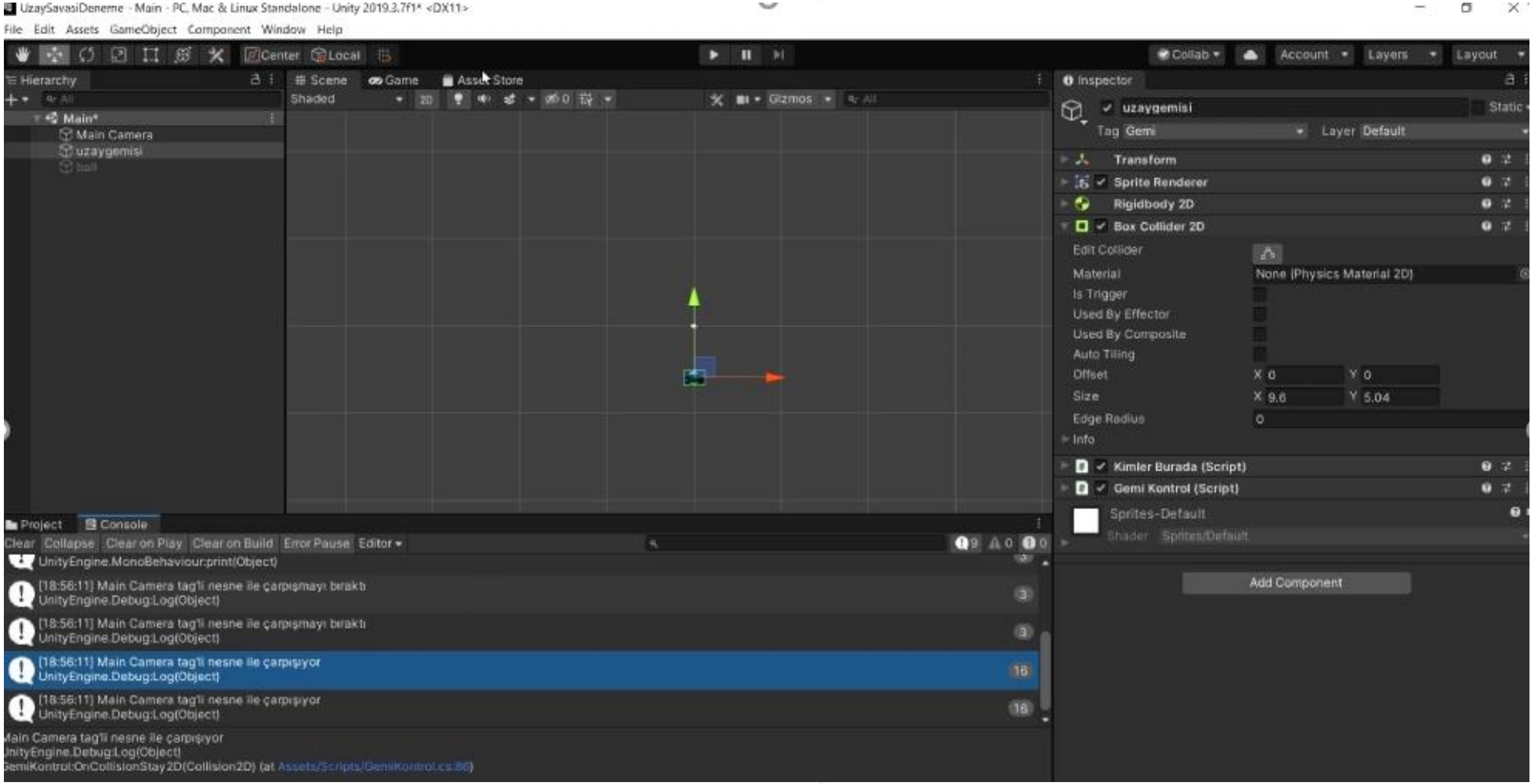
OnCollisionStay2D

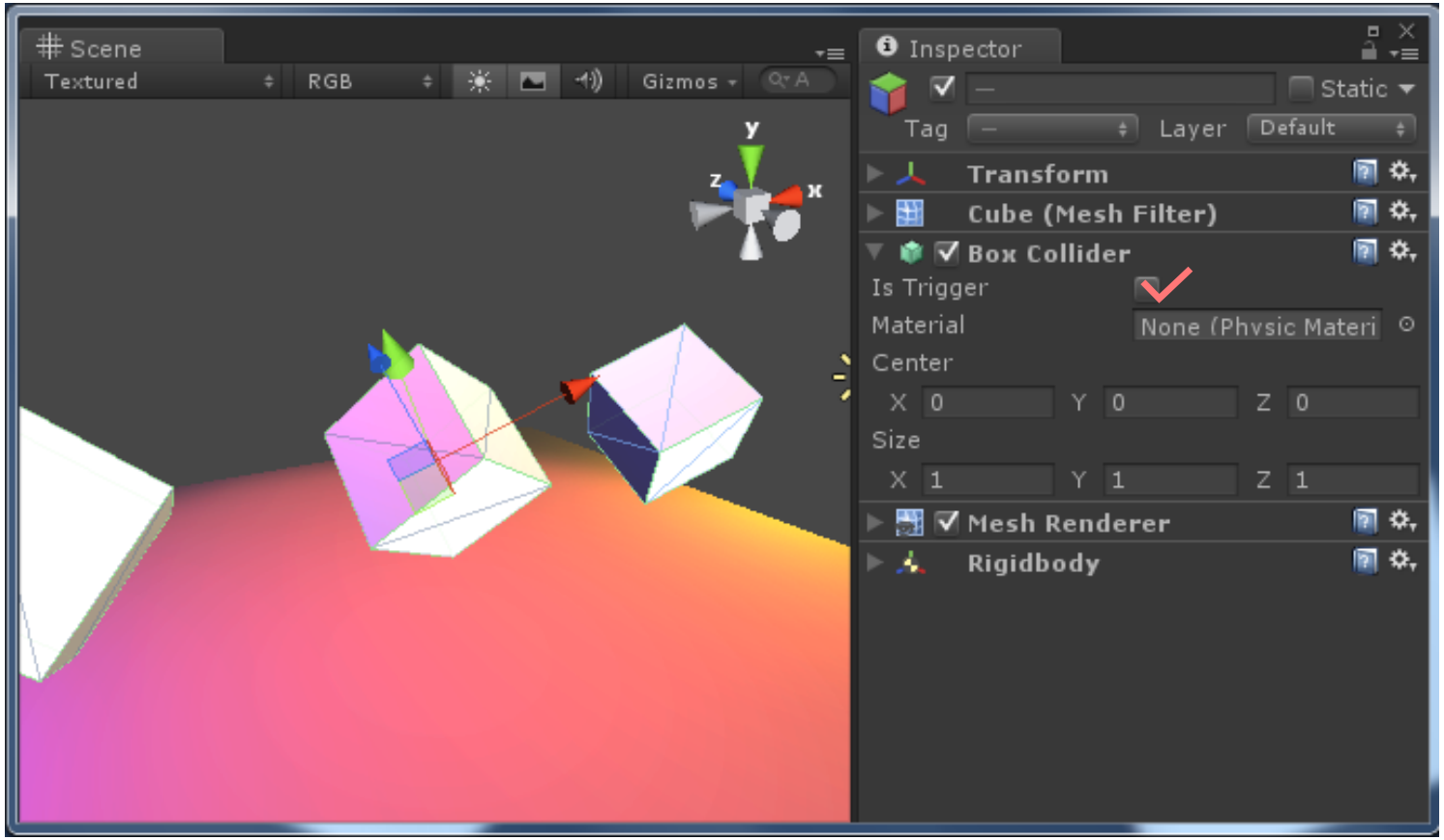
```
//Bir çarpıştırıcının objemizin çarpıştırıcısına temas ettiği her kareyi
gönderir.

private void OnCollisionStay2D(Collision2D collision)

    {

        //Bir çarpıştırıcının objemizin çarpıştırısına temas ettiği her kareyi
gönderir.

        if (collision.gameObject.tag == "MainCamera")

        {

            //MainCamera tag'li objeye çarptığı sürece çalışır

            Debug.Log("Main Camera tag'li nesne ile çarpışıyor");


        }

        if (collision.gameObject.name == "Main Camera")

        {

            //MainCamera isim'li objeye çarptığı sürece çalışır

            Debug.Log("Main Camera tag'li nesne ile çarpışıyor");


        }

    }
```

OnCollisionExit 2D

- OnCollisionStay bir objenin çarpma durumu bırakıldığı anda çalışır.

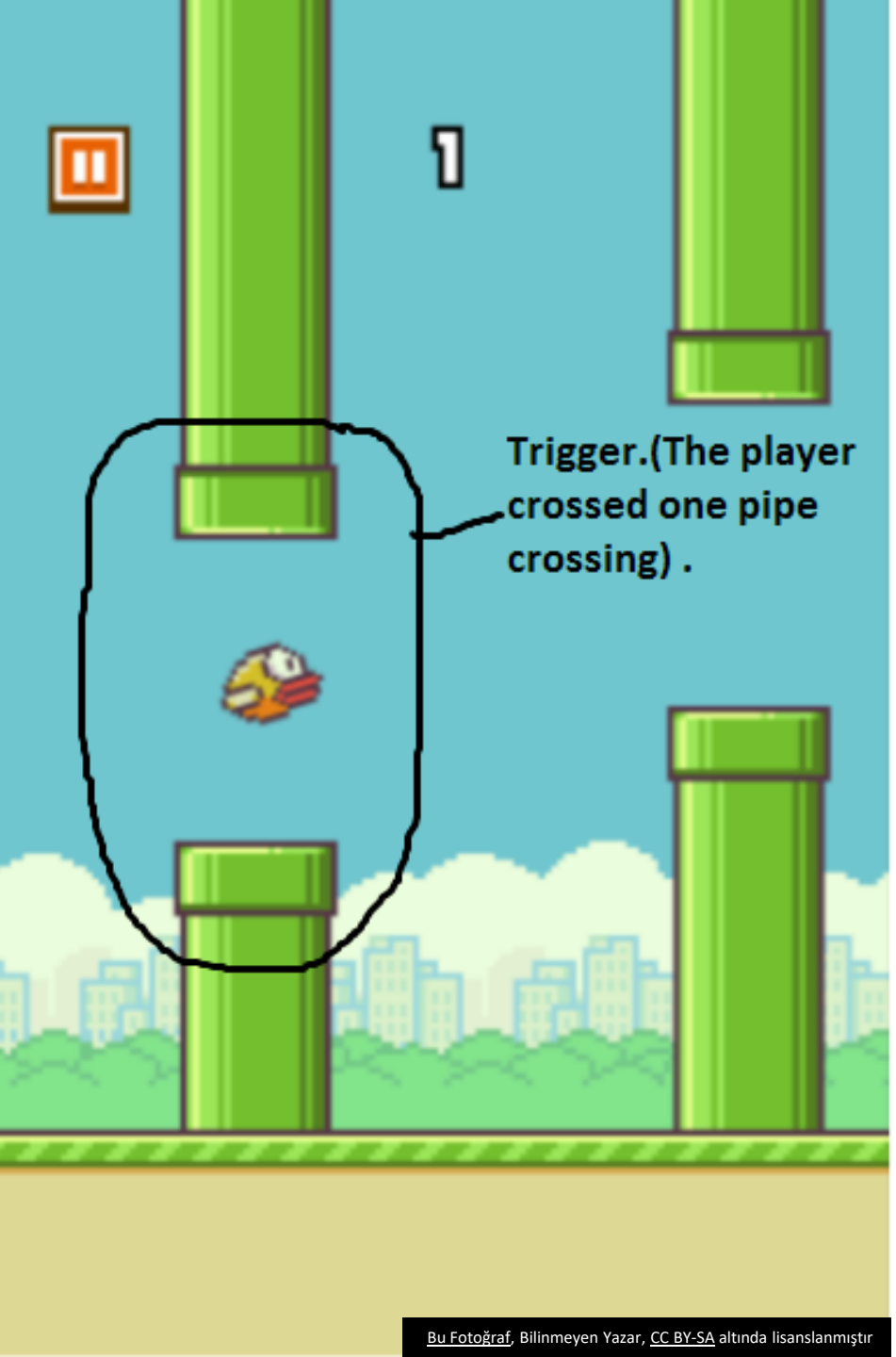
```
private void OnCollisionExit2D(Collision2D collision)
    {
        //Herhangi bir objeye çarpmayı bıraktığı an çalışır
        if (collision.gameObject.tag == "MainCamera")
        {
            //MainCamera tag'li objeye çarpmayı bıraktığında
çalışır
            Debug.Log("Main Camera tag’li nesne ile
çarpışmayı bıraktı");
        }
        if (collision.gameObject.name == "Main Camera")
        {
            //MainCamera isim'li objeye çarpmayı bıraktığında
çalışır
            Debug.Log(" Main Camera isimli obje ile
çarpışmayı bıraktı ");
        }
    }
```


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
uzaygemisi
Scene Game Asset Store
Shaded 2D Gizmos
Inspector
uzaygemisi
Static
Tag Gemi
Layer Default
Transform
Sprite Renderer
Rigidbody 2D
Box Collider 2D
Edit Collider
Material None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset X 0 Y 0
Size X 9.6 Y 5.04
Edge Radius 0
Info
Kimler Burada (Script)
Gemi Kontrol (Script)
Sprites-Default
Shader Sprites/Default
Add Component
Project Console
Clear Collapse Clear on Play Clear on Build Error Pause Editor
UnityEngine.MonoBehaviour:print(Object)
[18:56:11] Main Camera tagli nesne ile çarpışmayı bıraktı
UnityEngine.Debug:Log(Object)
[18:56:11] Main Camera tagli nesne ile çarpışmayı bıraktı
UnityEngine.Debug:Log(Object)
[18:56:11] Main Camera tagli nesne ile çarpışıyor
UnityEngine.Debug:Log(Object)
[18:56:11] Main Camera tagli nesne ile çarpışıyor
UnityEngine.Debug:Log(Object)
Main Camera tagli nesne ile çarpışıyor
UnityEngine.Debug:Log(Object)
GemiKontrol:OnCollisionStay2D(Collision2D) (at Assets/Scripts/GemiKontrol.cs:86)



Trigger

- Trigger çalışması için Colider componentinin ve rigidbody ekli olması gerekir.
- Colider Componentinin içindeki **Is trigger**'in işaretli olması gerekir.



Trigger

- 3 çeşit fonksiyonu vardır.
- OnCollisionEnter
- OnCollisionStay
- OnCollisionExit

OnTrigerEnter

- OnTriggerEnter bir objenin çarpma durumunda 1 kere çalışır.

```
private void OnTriggerEnter2D(Collider2D
collision)
    {
        Debug.Log("Trigger çarpma
gerçekleşti");
    }
```

OnTrigerStay

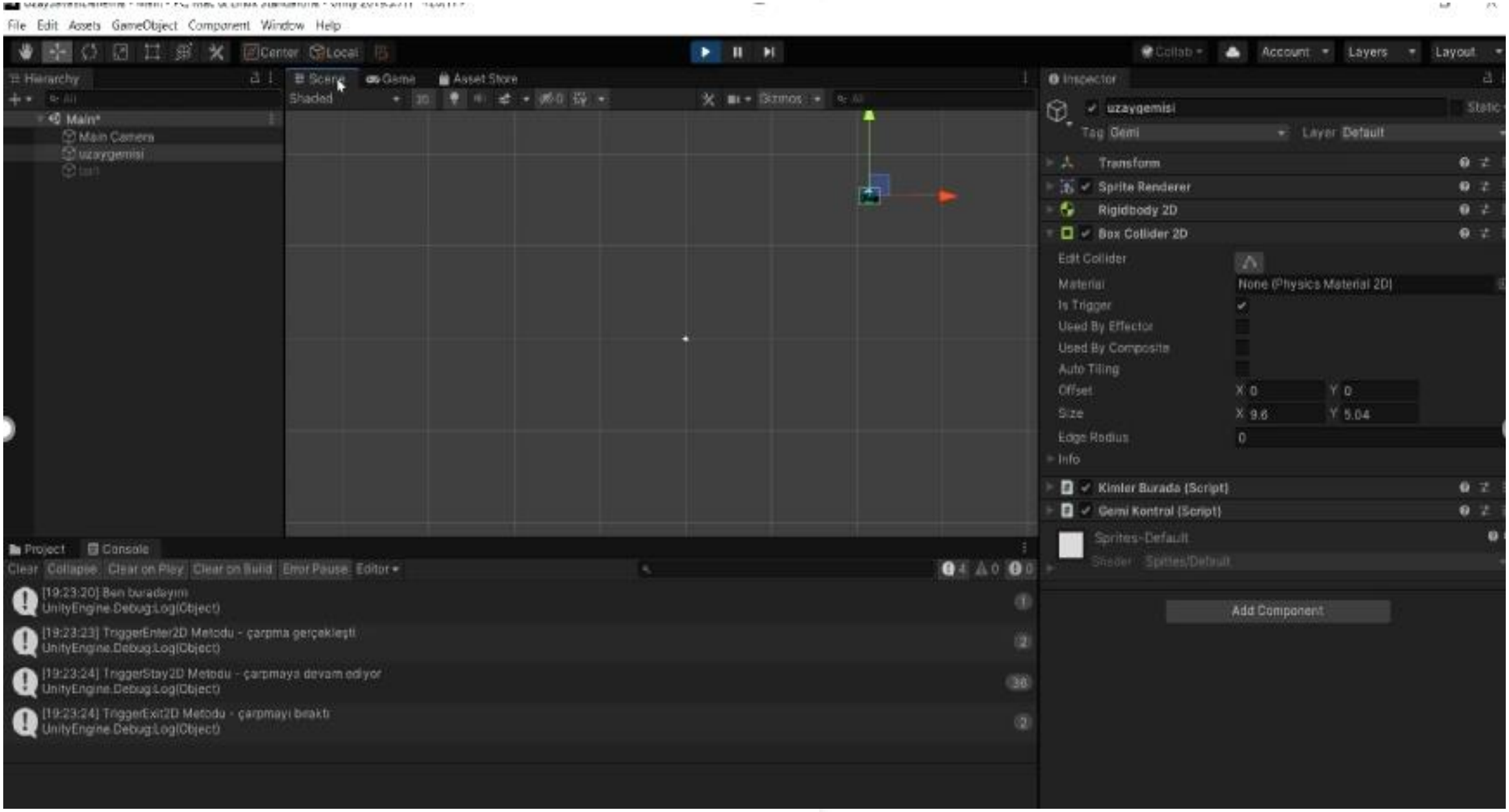
- OnTriggerStay bir objenin çarpma durumu sürdüğü sürece çalışır.

```
private void OnTriggerStay2D(Collider2D
collision)
    {
        Debug.Log("TriggerStay2D Metodu -
çarpmaya devam ediyor");
    }
```

OnTrigerExit

- OnTriggerExit bir objenin çarpma durumu bittiğinde 1 kere çalışır.

```
private void OnTriggerExit2D(Collider2D
collision)
    {
        Debug.Log("TriggerExit2D Metodu -
çarpmayı bıraktı");
    }
```

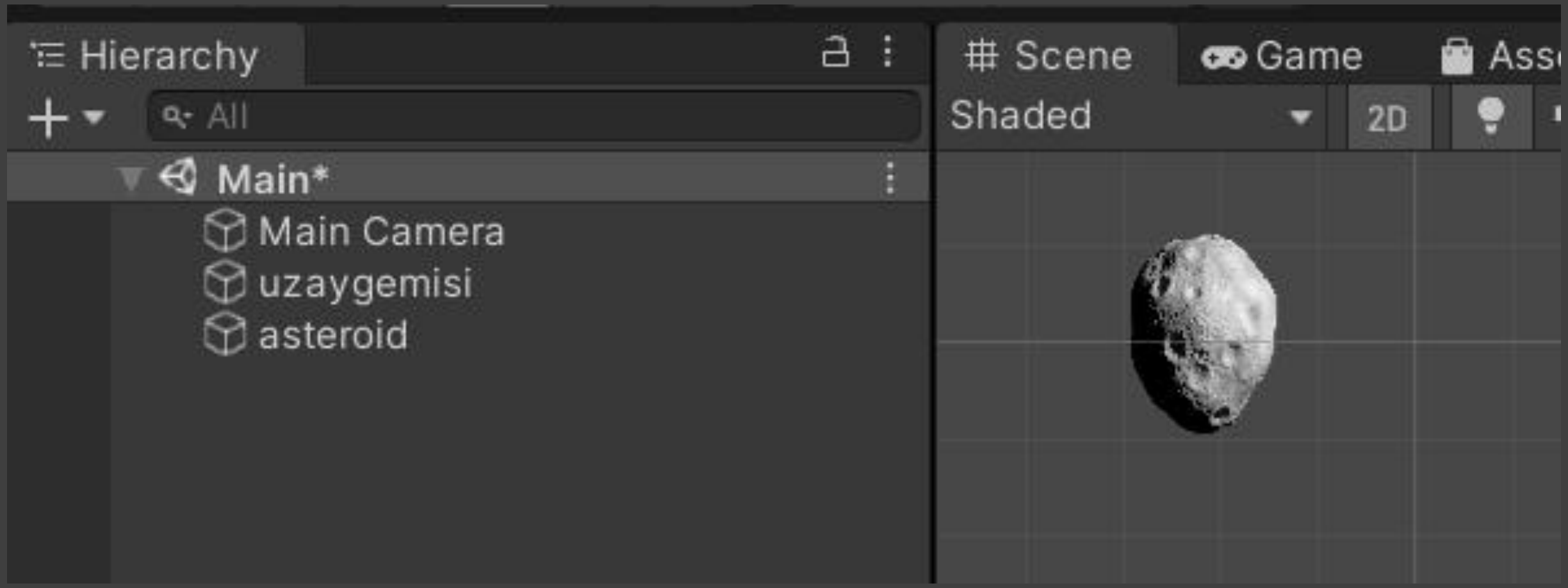

File Edit Assets GameObject Component Window Help
Hierarchy
Main*
Main Camera
uzaygemisi
Scene
Game
Asset Store
Shaded
Inspector
uzaygemisi
Tag Gemi
Layer Default
Transform
Sprite Renderer
Rigidbody 2D
Box Collider 2D
Edit Collider
Material None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset X 0 Y 0
Size X 9.6 Y 5.04
Edge Radius 0
Info
Kimler Burada (Script)
Gemi Kontrol (Script)
Sprites-Default
Shader Sprites/Default
Add Component
Project
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
[19:23:20] Ben buradayım
UnityEngine.Debug:Log(Object)
[19:23:23] TriggerEnter2D Metodu - çarpma gerçekleşti
UnityEngine.Debug:Log(Object)
[19:23:24] TriggerStay2D Metodu - çarpmaya devam ediyor
UnityEngine.Debug:Log(Object)
[19:23:24] TriggerExit2D Metodu - çarpmayı bıraktı
UnityEngine.Debug:Log(Object)

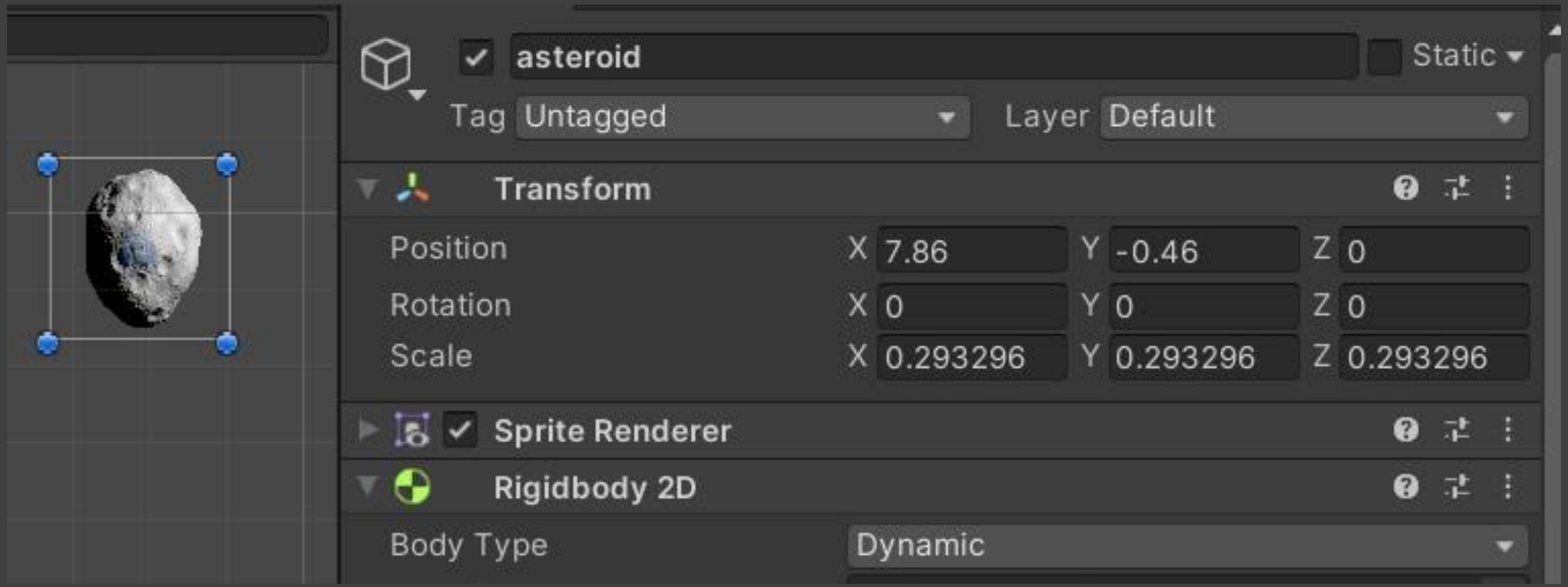
Çoklu Oyun Objeleri

Prefab

Oyun Objesi

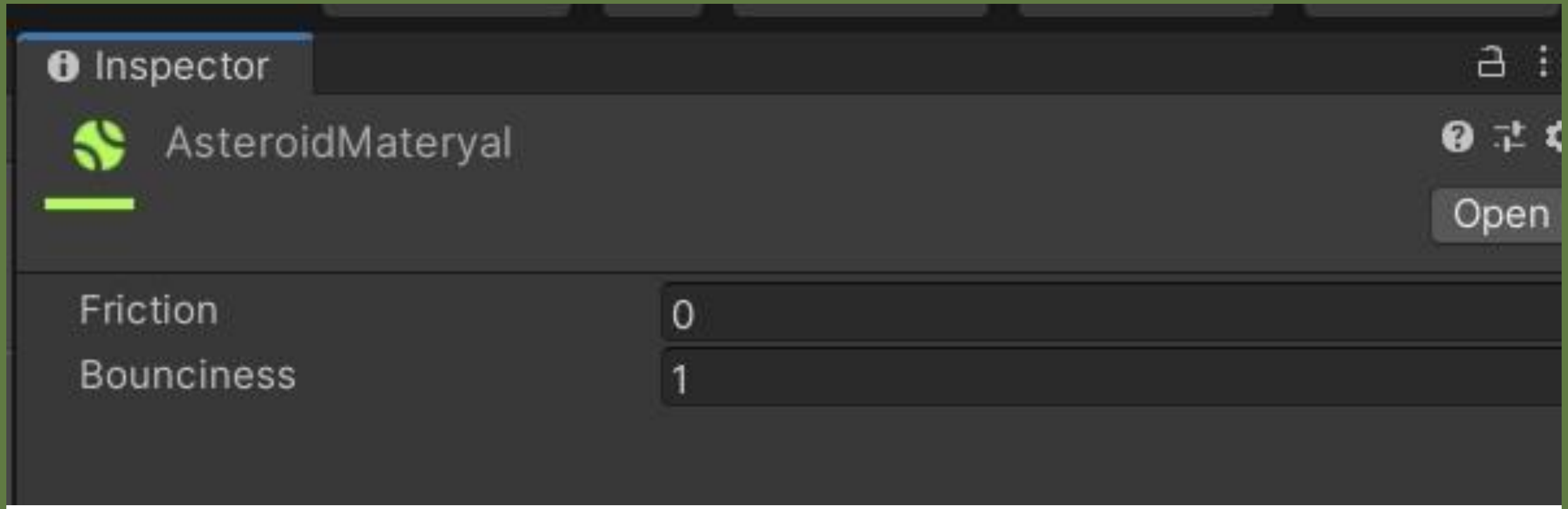
- Projeye bir Sprite oyun objesi ekleriz.



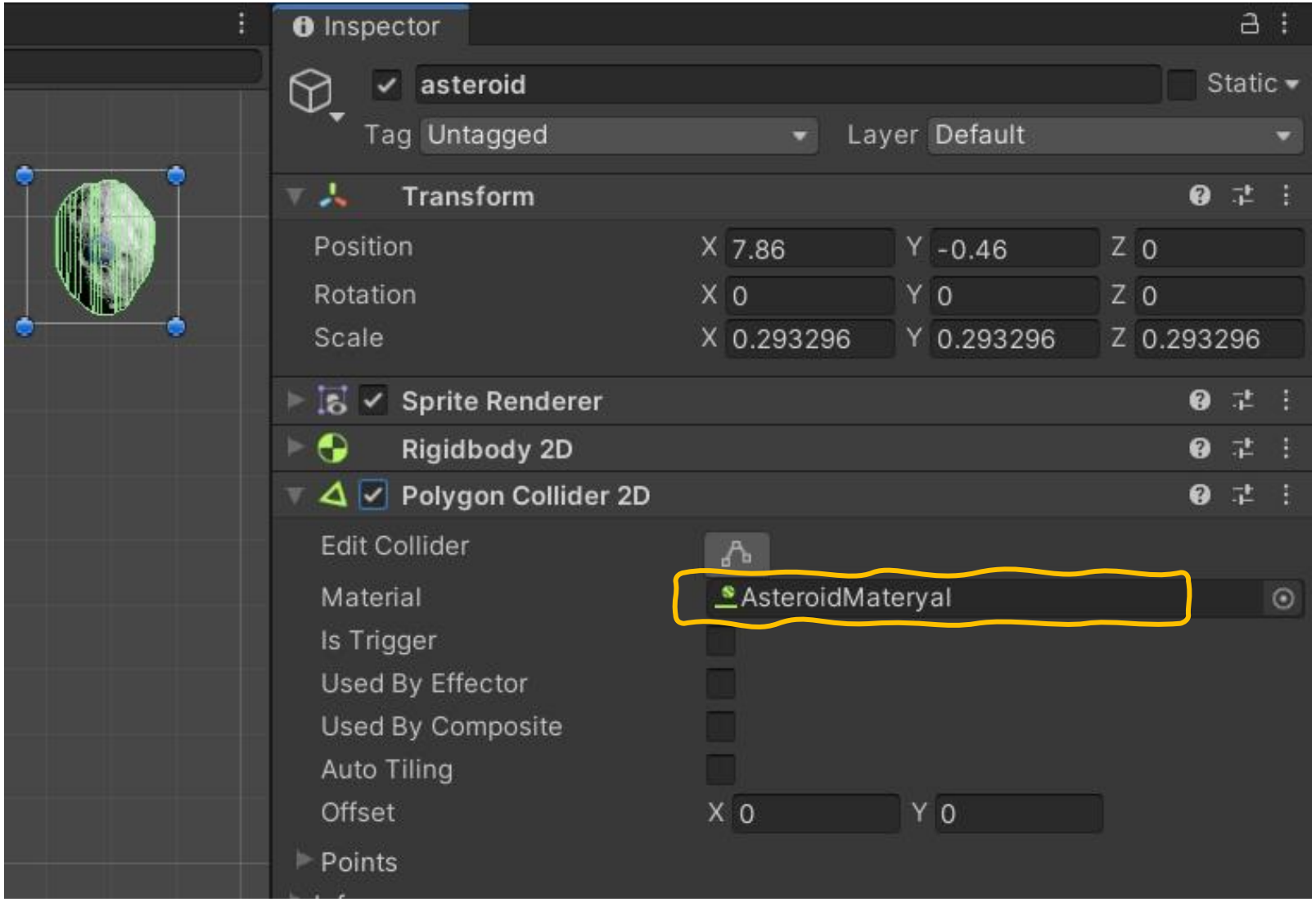


Rigidbody 2D Component

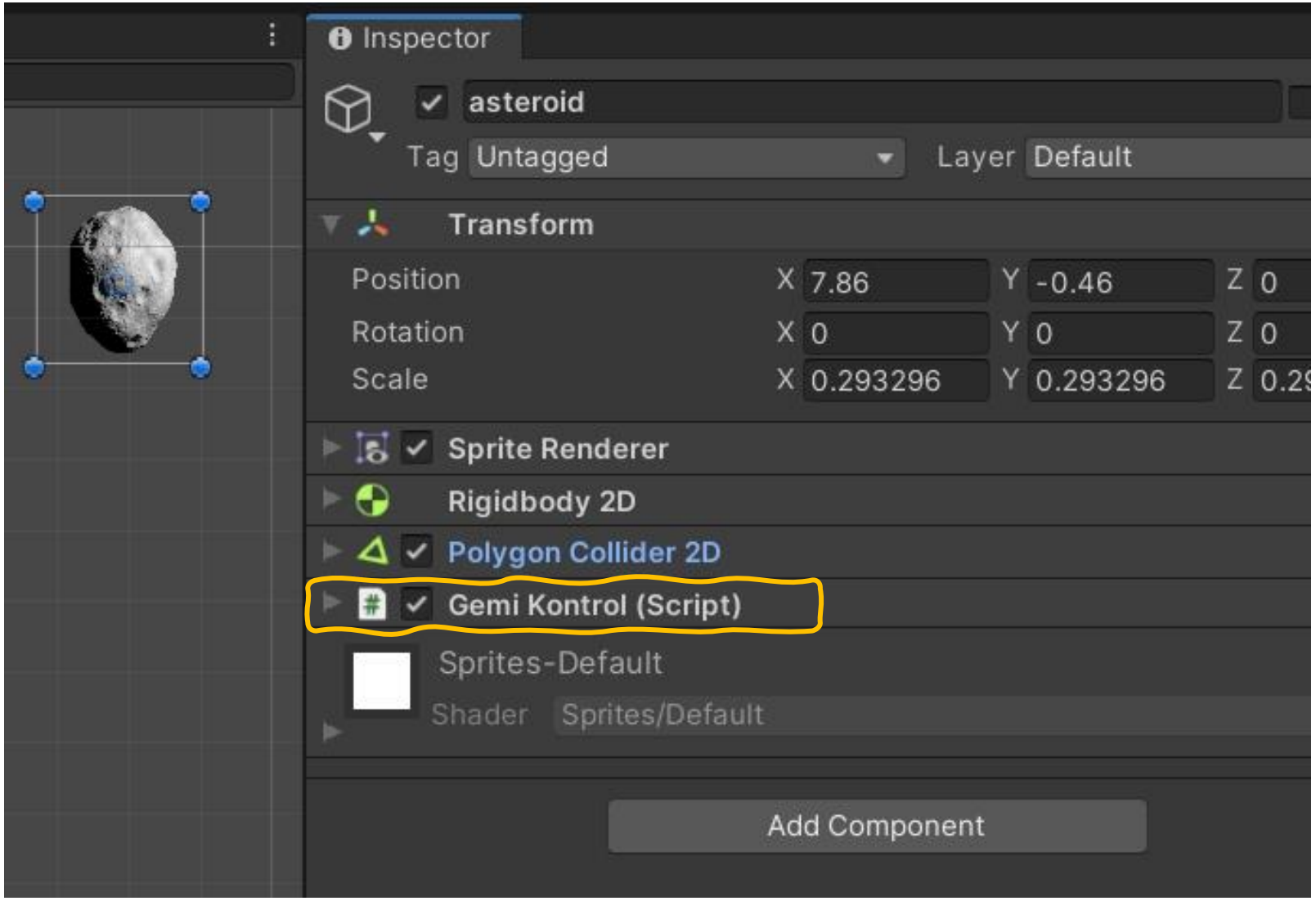
Oyun objemize Rigidbody 2D Component’ini ekledik.



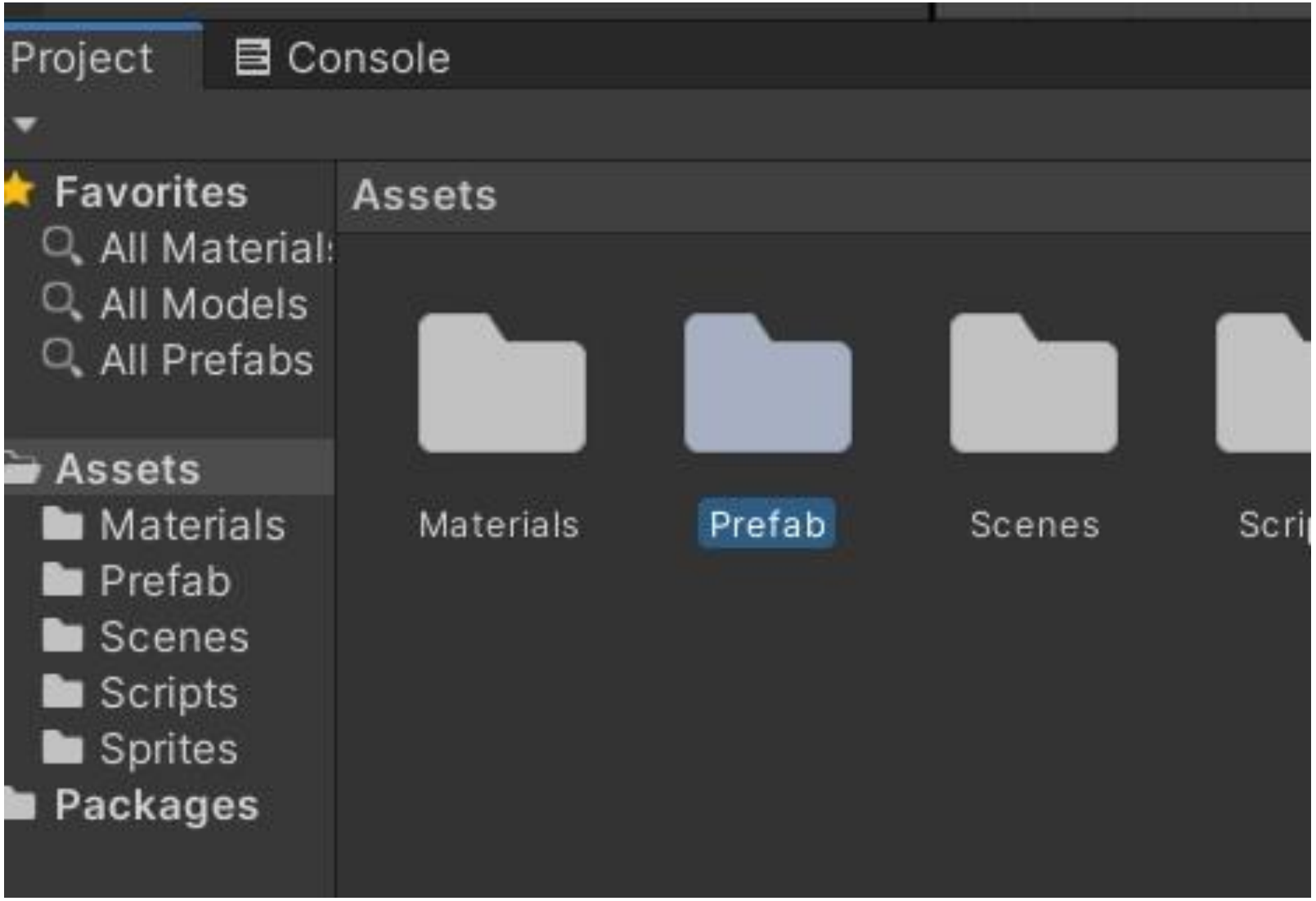
AsteroidMateryal ekledik



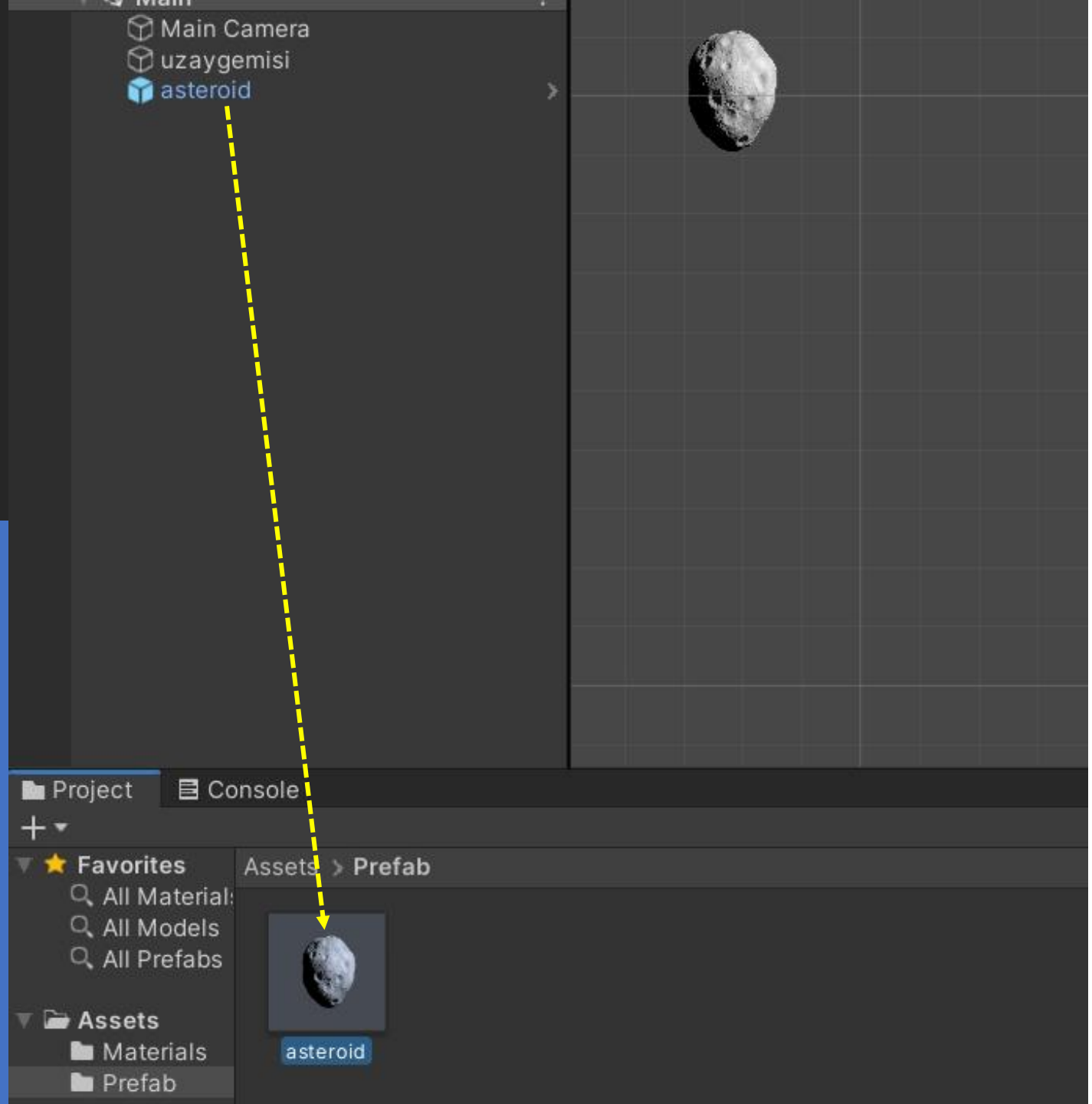
Objemize uygun olan Collider'ioyun objemize ekleyerek, Material kısmından bu obje için oluşturmuş olduğumuz materyali seçeriz.



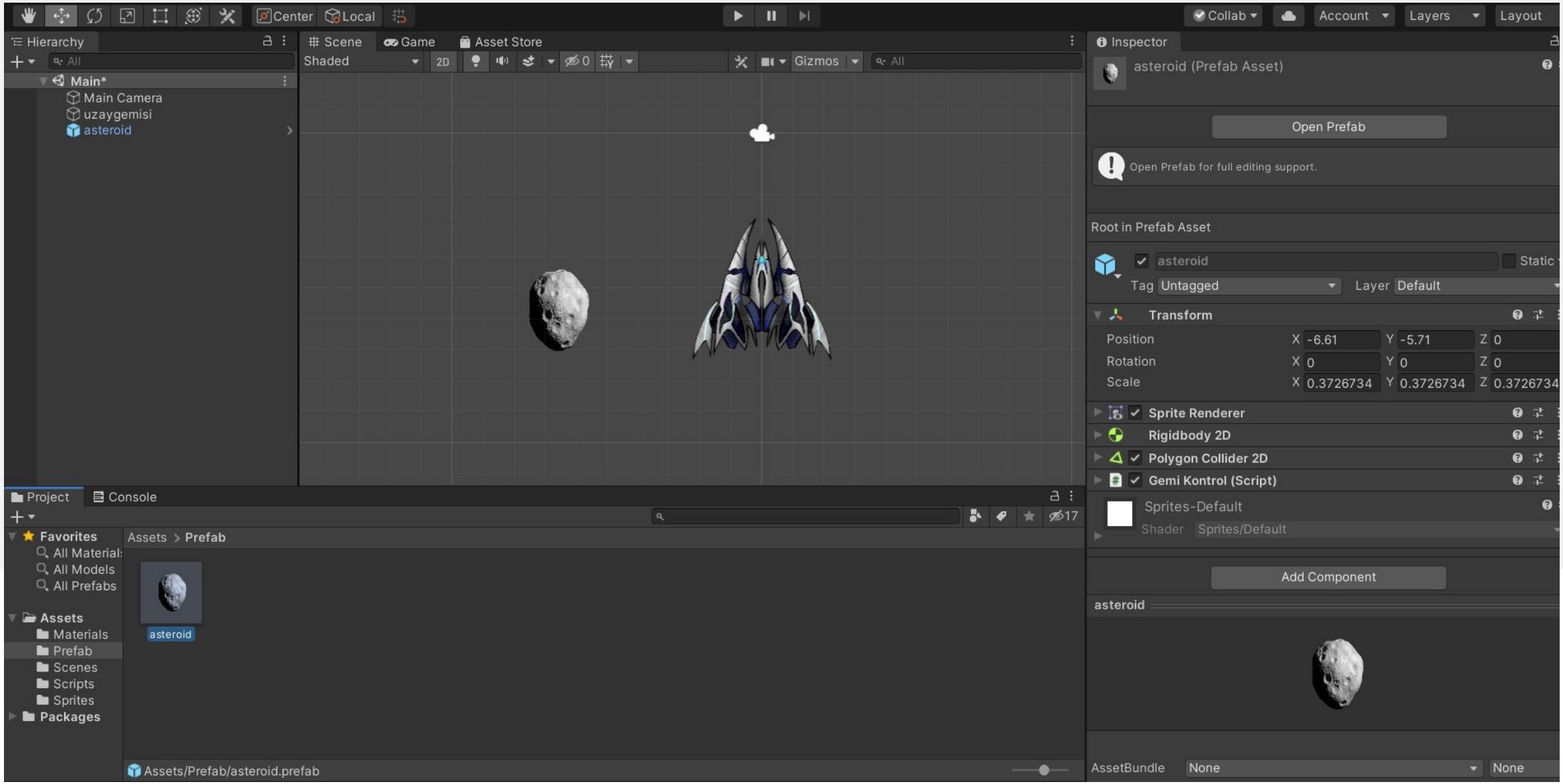
Oyun Objesi Hareket Script

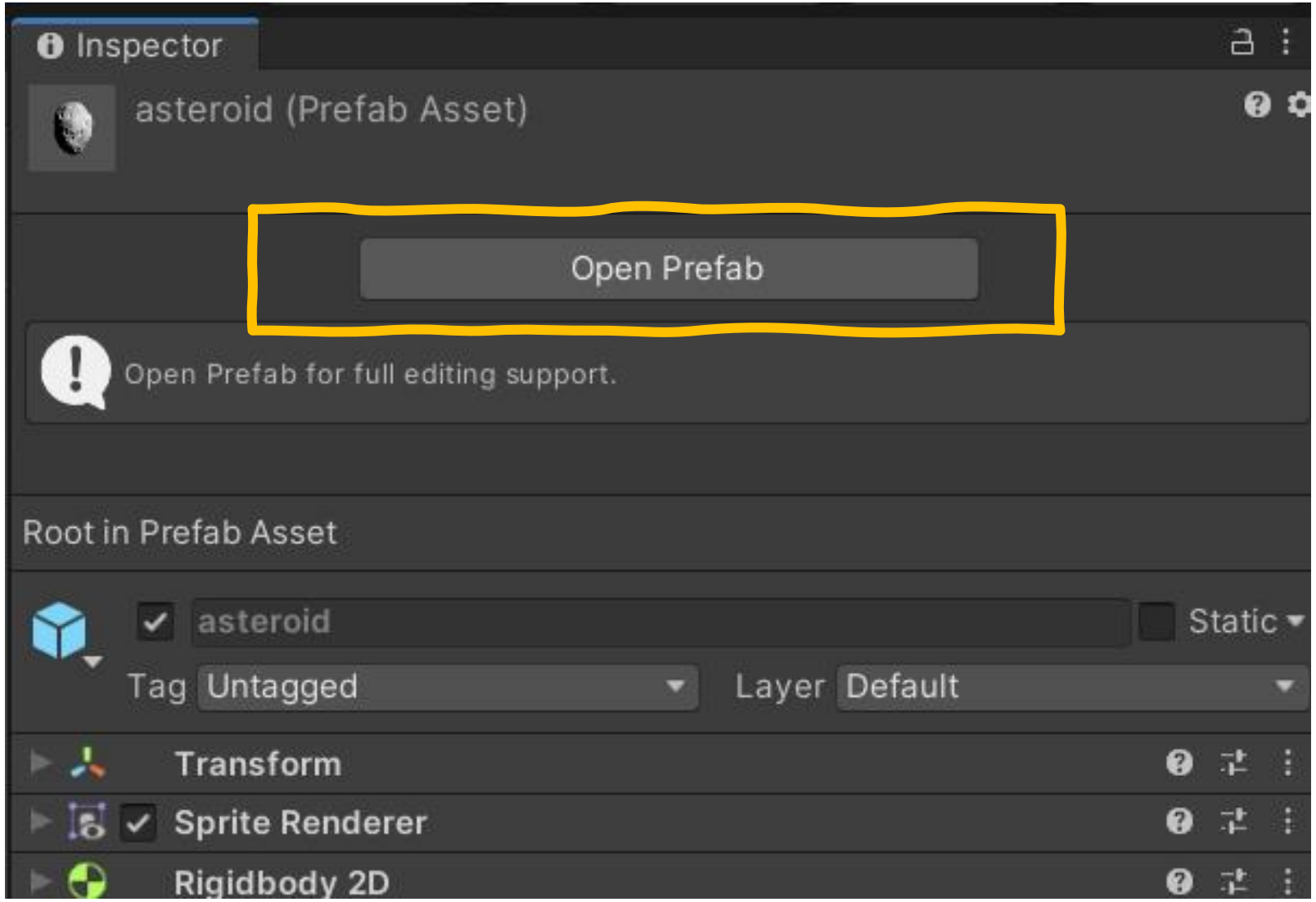


Prefab klasörünü oluştururuz

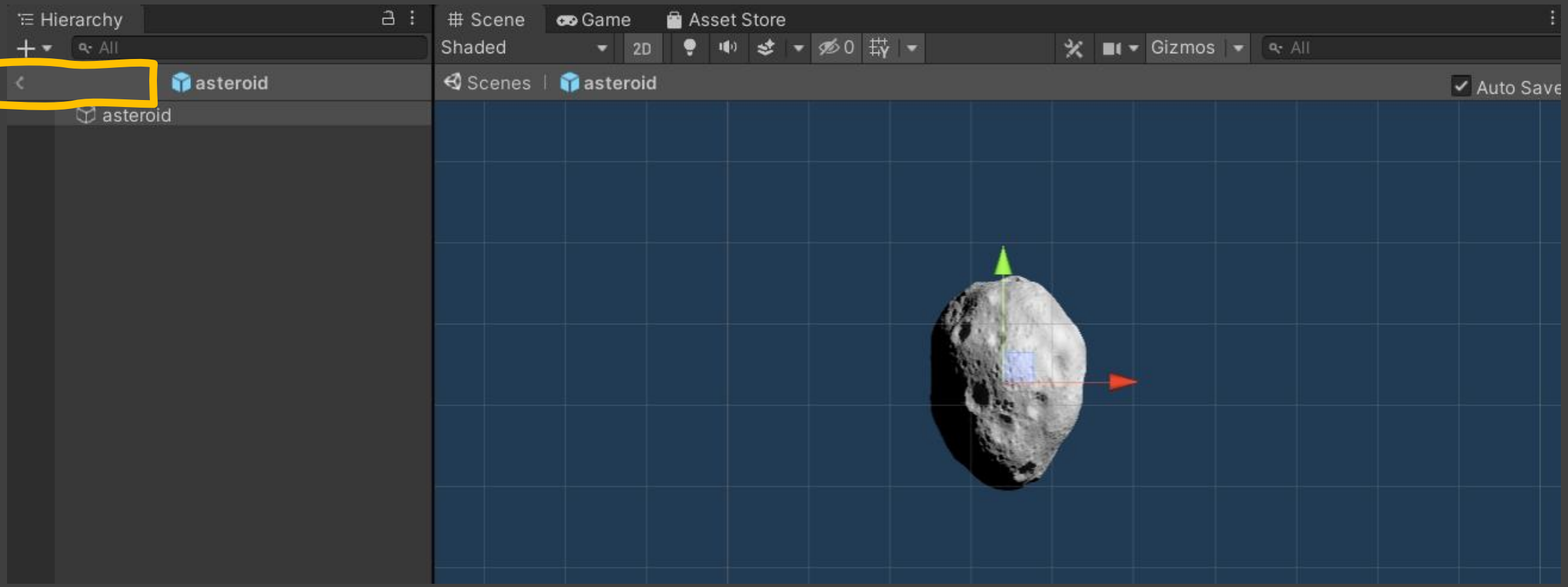


Oyun objemizi Prefab klasörüne sürükleyerek bırakırız


Center
Local
Collab
Account
Layers
Layout
Hierarchy
Main*
Main Camera
uzaygemisi
asteroid
Scene
Game
Asset Store
Shaded
2D
Gizmos
Inspector
asteroid (Prefab Asset)
Open Prefab
Open Prefab for full editing support.
Root in Prefab Asset
asteroid
Static
Tag Untagged
Layer Default
Transform
Position X -6.61 Y -5.71 Z 0
Rotation X 0 Y 0 Z 0
Scale X 0.3726734 Y 0.3726734 Z 0.3726734
Sprite Renderer
Rigidbody 2D
Polygon Collider 2D
Gemi Kontrol (Script)
Sprites-Default
Shader Sprites/Default
Add Component
asteroid
AssetBundle None None
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Materials
Prefab
Scenes
Scripts
Sprites
Packages
Assets > Prefab
asteroid
Assets/Prefab/asteroid.prefab

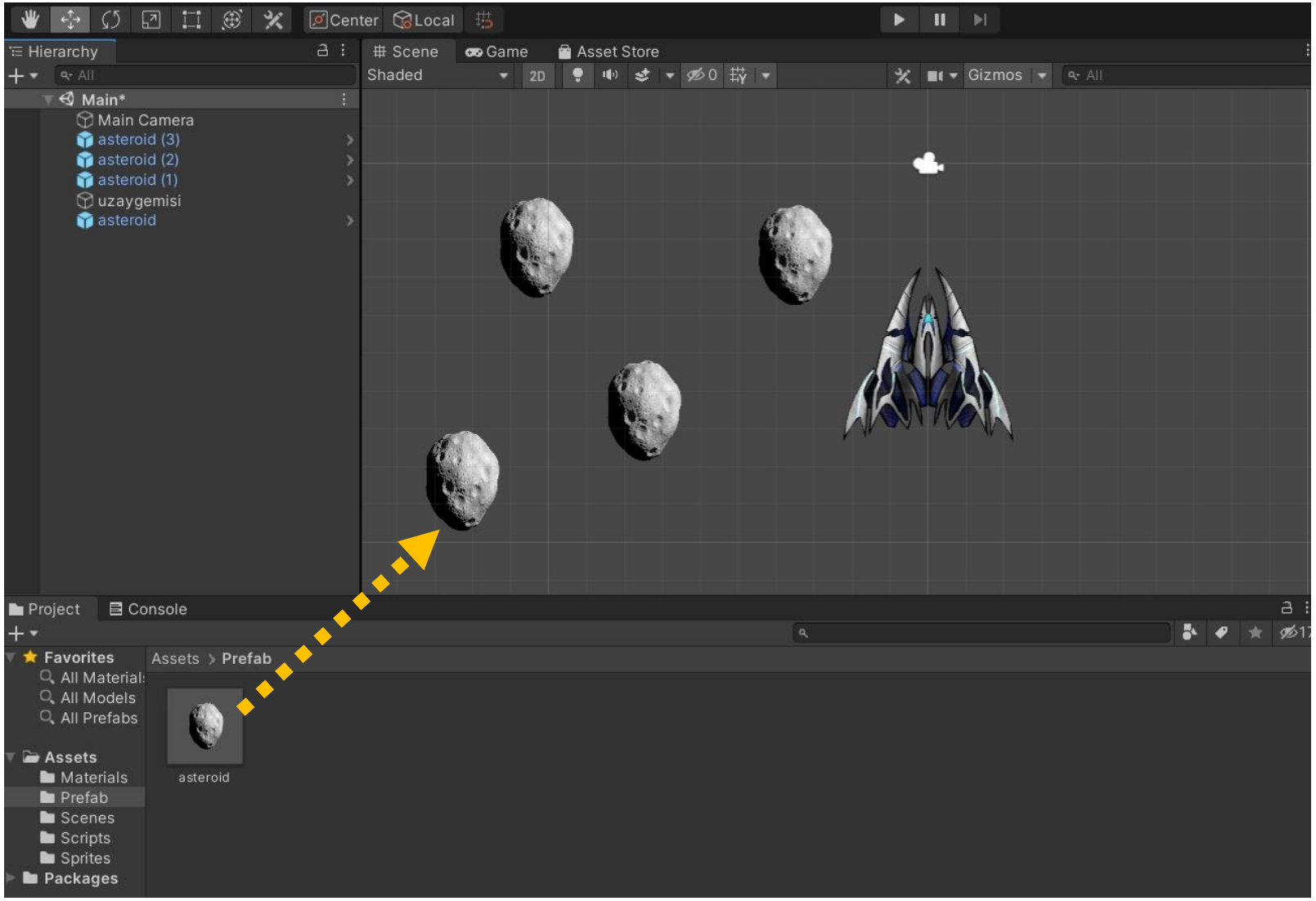


Prefab üzerinde bir değişiklik yapmak için Open Prefab düğmesine tıklarız

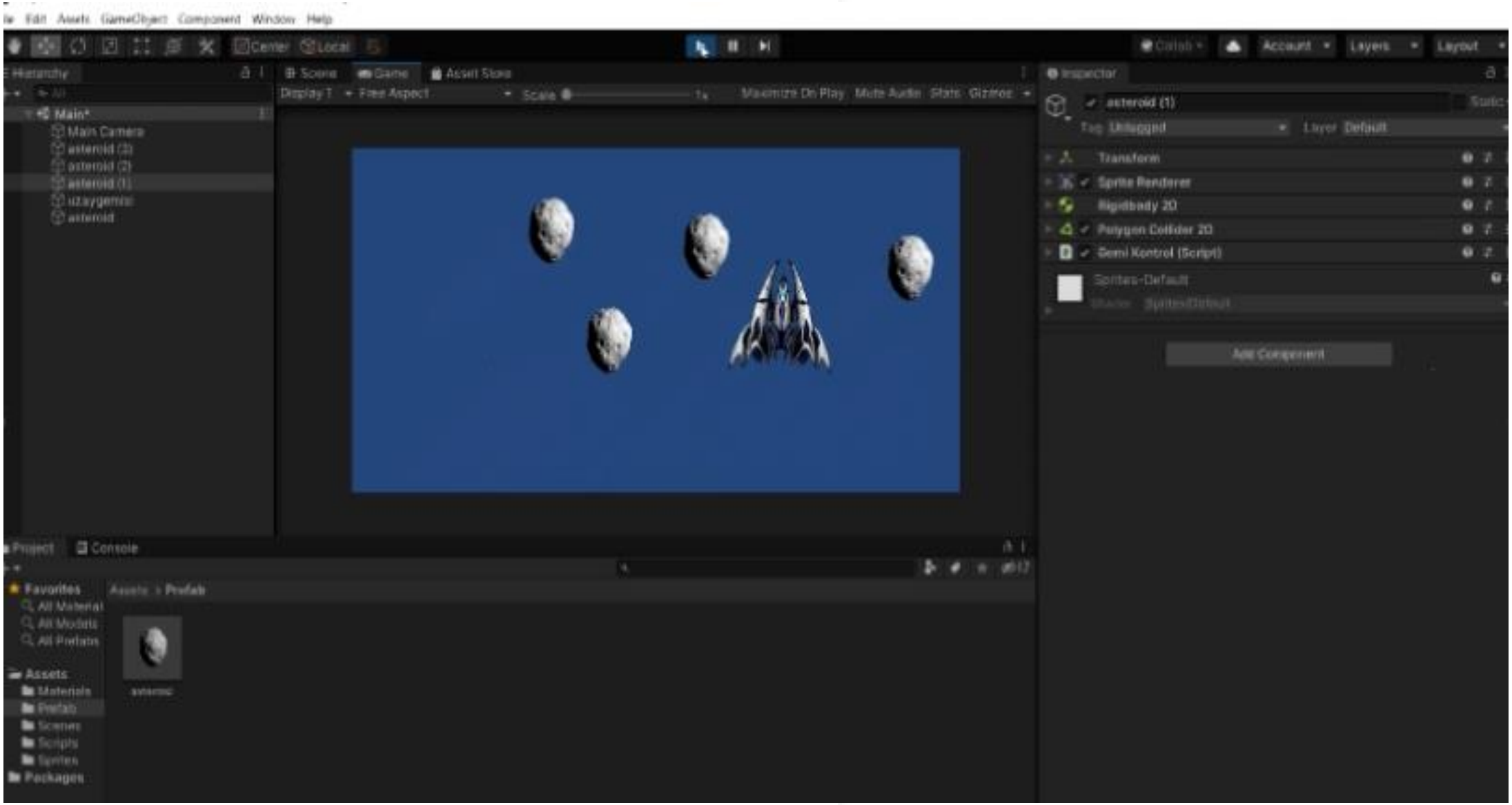


Prefab Modundan Çıkış

Prefab değişiklik modundan çıkmak için Hierarchy kısmındaki ok işaretine tıklarız



Prefab objemizi projemize dahil etmek için nesnemizi sürükleyerek sahneye bırakırız


Edit Assets GameObject Component Window Help
Center Local
Account Layers Layout
Hierarchy
Main*
Main Camera
asteroid (3)
asteroid (2)
asteroid (1)
uzaygemisi
asteroid
Scene Game Asset Store
Display 1 Free Aspect Scale Maximize On Play Mute Audio Stats Gizmos
Inspector
asteroid (1)
Tag Untagged Layer Default
Transform
Sprite Renderer
Rigidbody 2D
Polygon Collider 2D
Gemi Kontrol (Script)
Sprites-Default
Add Component
Project Console
Favorites
All Materials
All Models
All Prefabs
Assets
Materials
Prefab
Scenes
Scripts
Sprites
Packages
Assets > Prefab
asteroid

Mantıksal Operatörler

Operatör	Açıklama
&&	VE (AND) Tüm koşulların doğru olması sonucu true değer üretilir.
||	VEYA (OR) Bir koşulun doğru olması sonucu true değer üretilir.
!	DEĞİL (NOT) Koşul sonucu true ise false, false ise true yapar.

Geri Sayım Sayacı

Level bitmesi

Süreli yok etme

Bölüm sonu canavarlarının çıkması (Boss)

Süreli zorluk

Geri Sayım Sayacı

Think twice code once (İki defa düşün bir defa kodla)

Reusable code (Yeniden kullanılabilir kod)

Fields:

- toplamSure
- gecenSure
- calismaDurumu → true, false
- baslamaDurum → true, false

Methods:

- Calistir()

GeriSayimSayaci.cs

```
float toplamSure = 0, gecenSure = 0;
bool calismaDurumu = false, baslamaDurumu = false;


/// <summary>
/// Geri sayım sayacının toplam süresini ayarlar
/// </summary>
public float ToplamSure
{
    set
    {
        if (!calismaDurumu) //sayaç çalışmıyorsa
            toplamSure = value;
    }
}
```

GeriSayimSayaci.cs

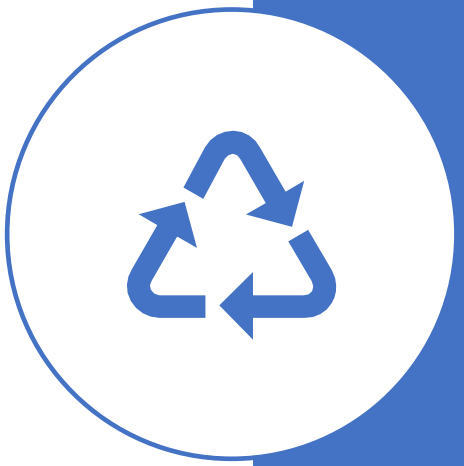
```
/// <summary>
/// Sayacı çalıştırır
/// </summary>
public void Calistir()
{
    if (toplamSure>0)
    {
        calismaDurumu = true;
        baslamaDurumu = true;
        gecenSure = 0;
    }
}
```

GeriSayimSayaci.cs

```
/// <summary>
/// Geri sayımın bitip bitmediğini söyler
/// </summary>
public bool Bitti
{
    get
    {
        return baslamaDurumu && !calismaDurumu;
    }
}
```

GeriSayimSayaci.cs

```
void Update()
    {
        if (calismaDurumu)
        {
            gecenSure += Time.deltaTime;
            //Çalışma zamanında geçen süreyi deltaTime properties ile alırız.
            //Bu değer fps değeridir. Yani çalıştığı cihaza göre değişiklik gösterir.
            //Update her frame'de çağrırır. Örneğin sistem 30 fbs ise burası saniyede 30 kere çalışır.
            if (gecenSure>=toplamSure)
            {
                calismaDurumu = false;
            }
        }
    }
```



GeriSayimTest.cs

```
GeriSayimSayaci geriSayimSayaci;
//Bir örnek (nesne) oluşturduk

float baslangicZamani;
```

GeriSayimTest.cs

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

        baslangicZamani = Time.time;
    }
```

GeriSayimTest.cs

```
void Update()
  {
      //Her frame'de sayacımıza bitti mi diye soracağız
      if (geriSayimSayaci.Bitti)
      {
          float gecensure = Time.time - baslangicZamani;
          Debug.Log(gecensure);

          baslangicZamani = Time.time;
          geriSayimSayaci.Calistir();
      }
  }
```

Spawn

- Oyun objelerinin dinamik olarak belirli bir duruma göre sahnede var olması işlemine oyun dünyasında spawn denir.

Spawner.cs

Gelecek haftaya devam edeceğiz…