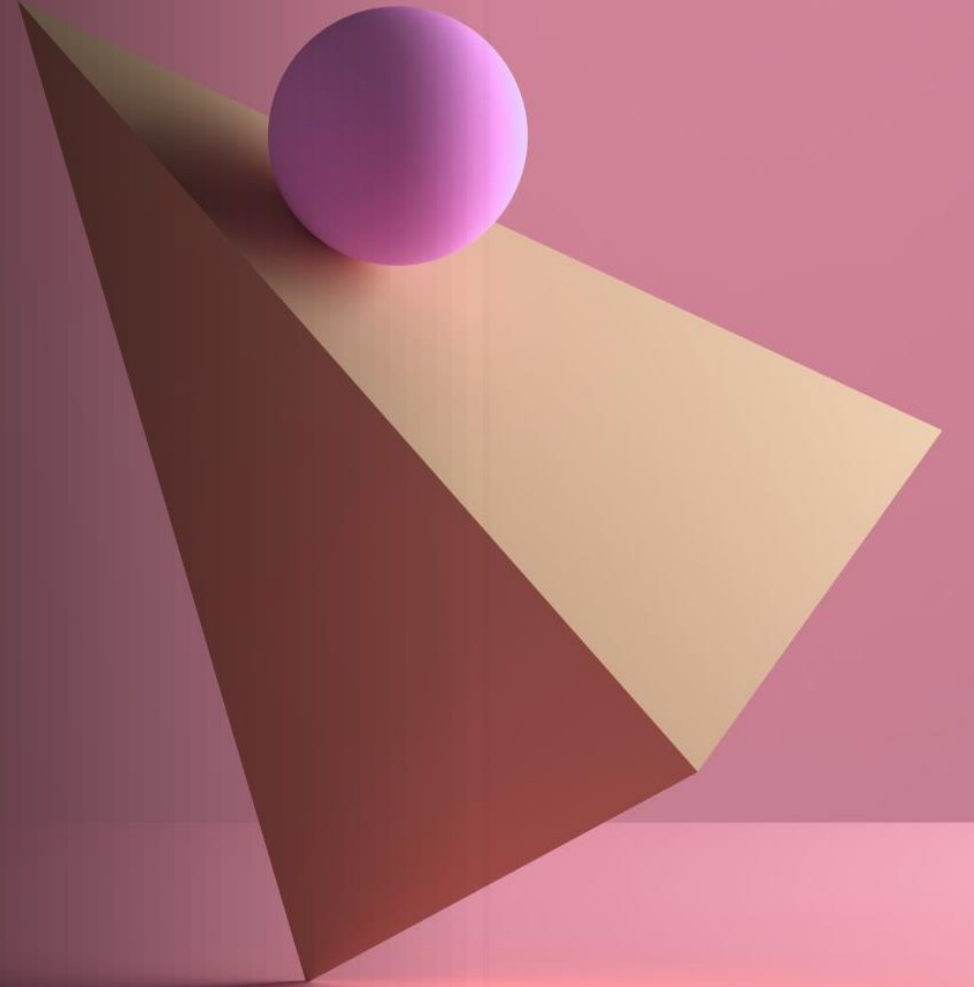




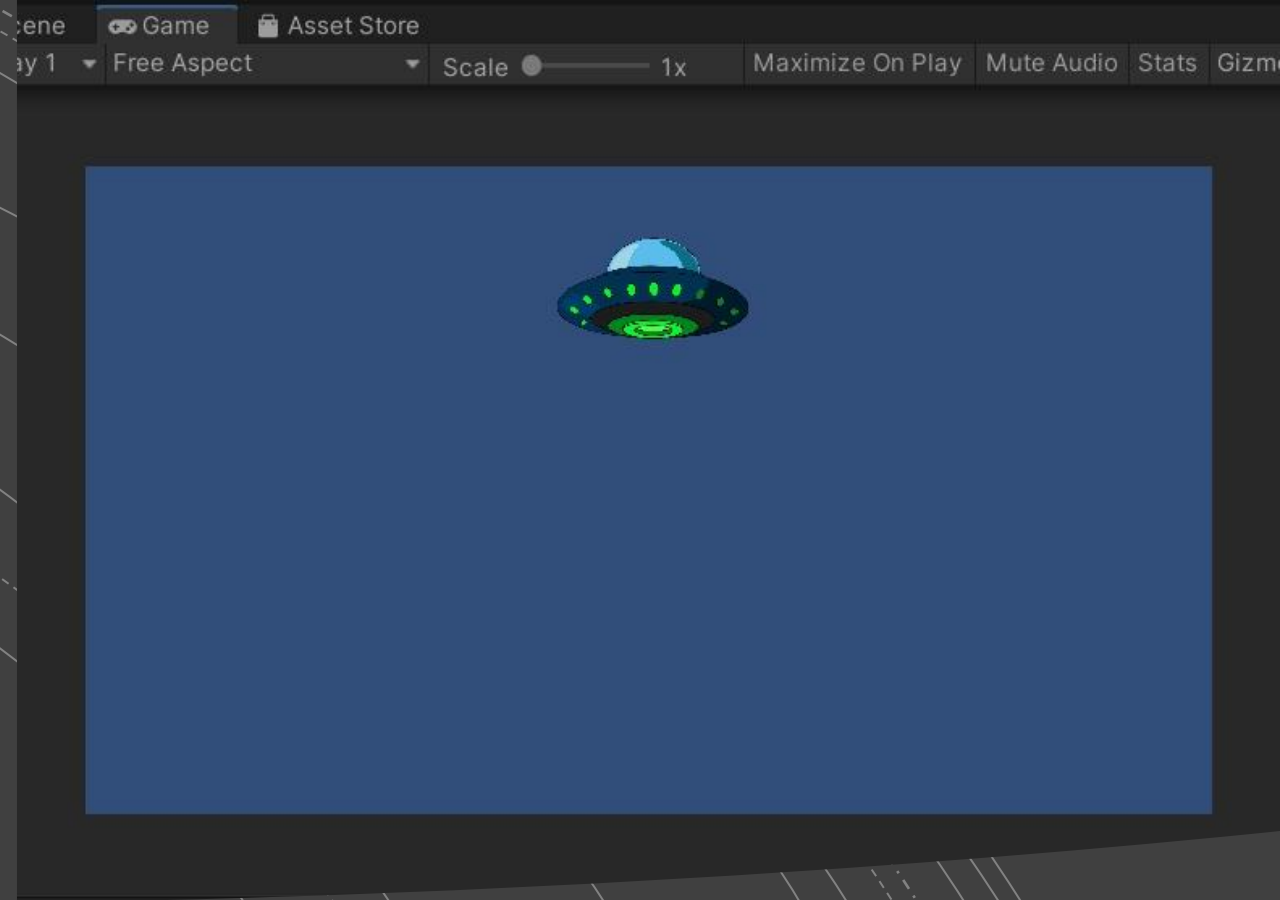
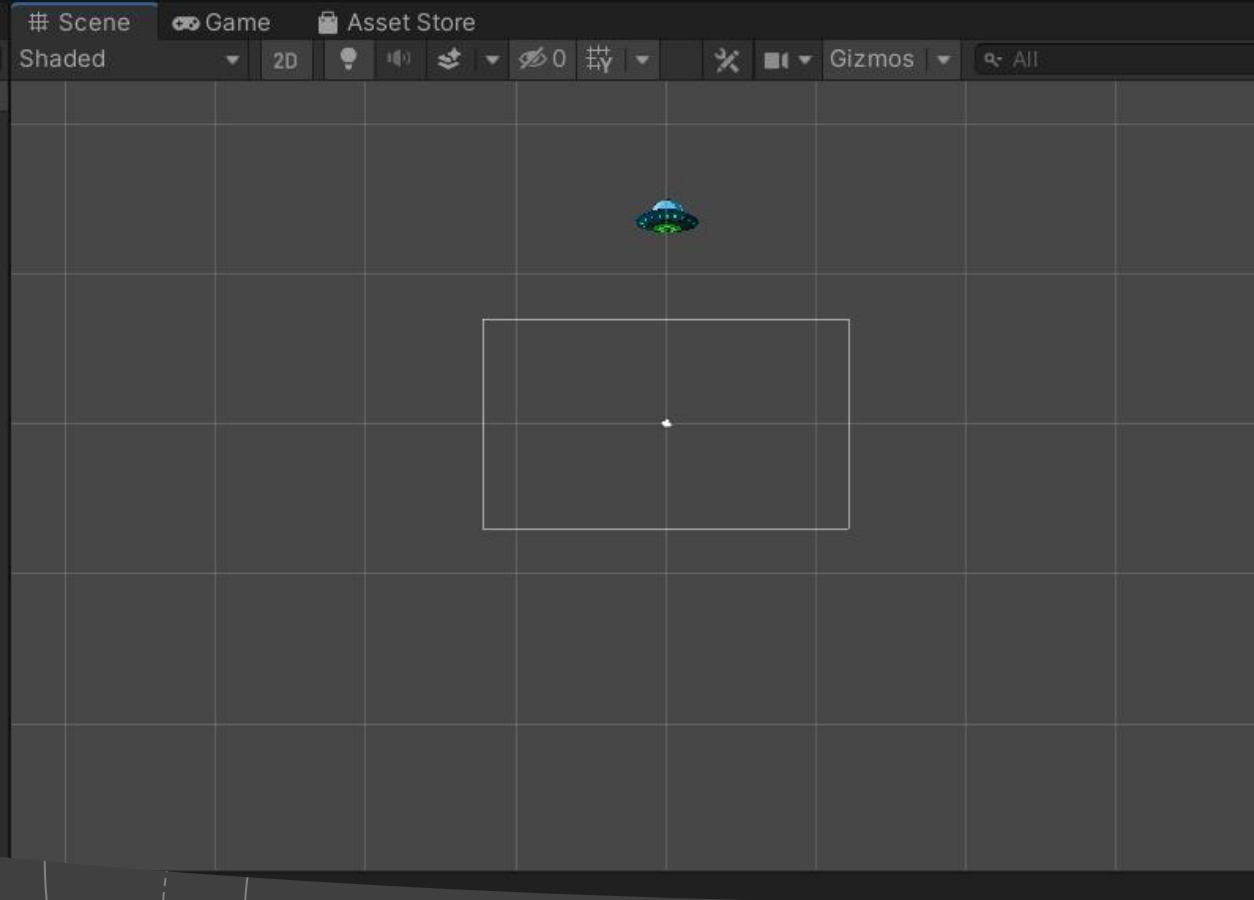
Unity

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.



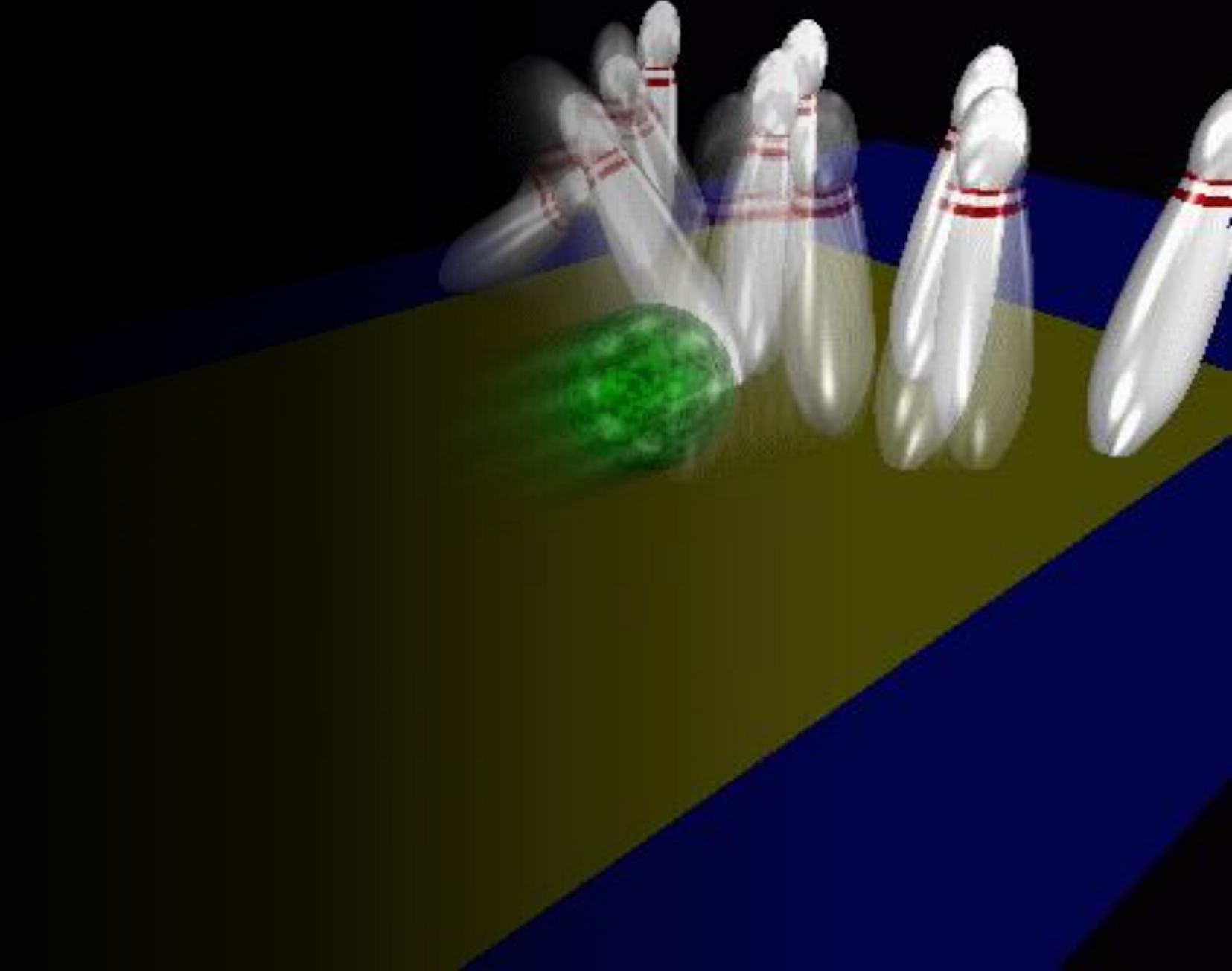
Unity de bir objenin diğer objeye çarpmasını anlamak için Collision veya Trigger kullanılır.



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.

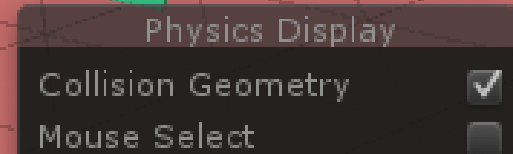


static Colliders

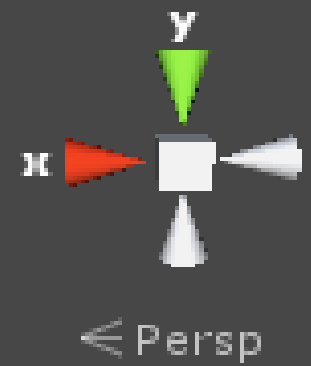
kinematic
Rigidbody

sleeping
Rigidbody

simulating
Rigidbody



Colliders3D



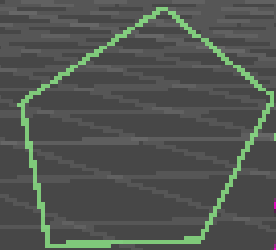
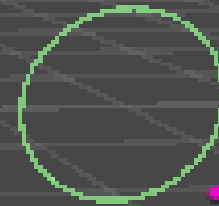
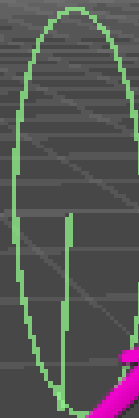
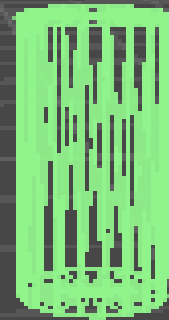
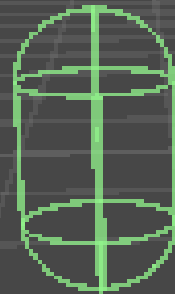
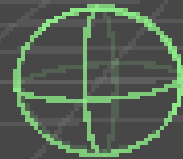
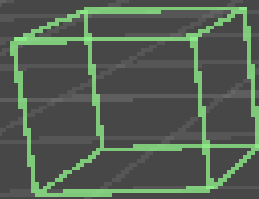
SphereCollider

MeshCollider

BoxCollider

CapsuleCollider

WheelCollider



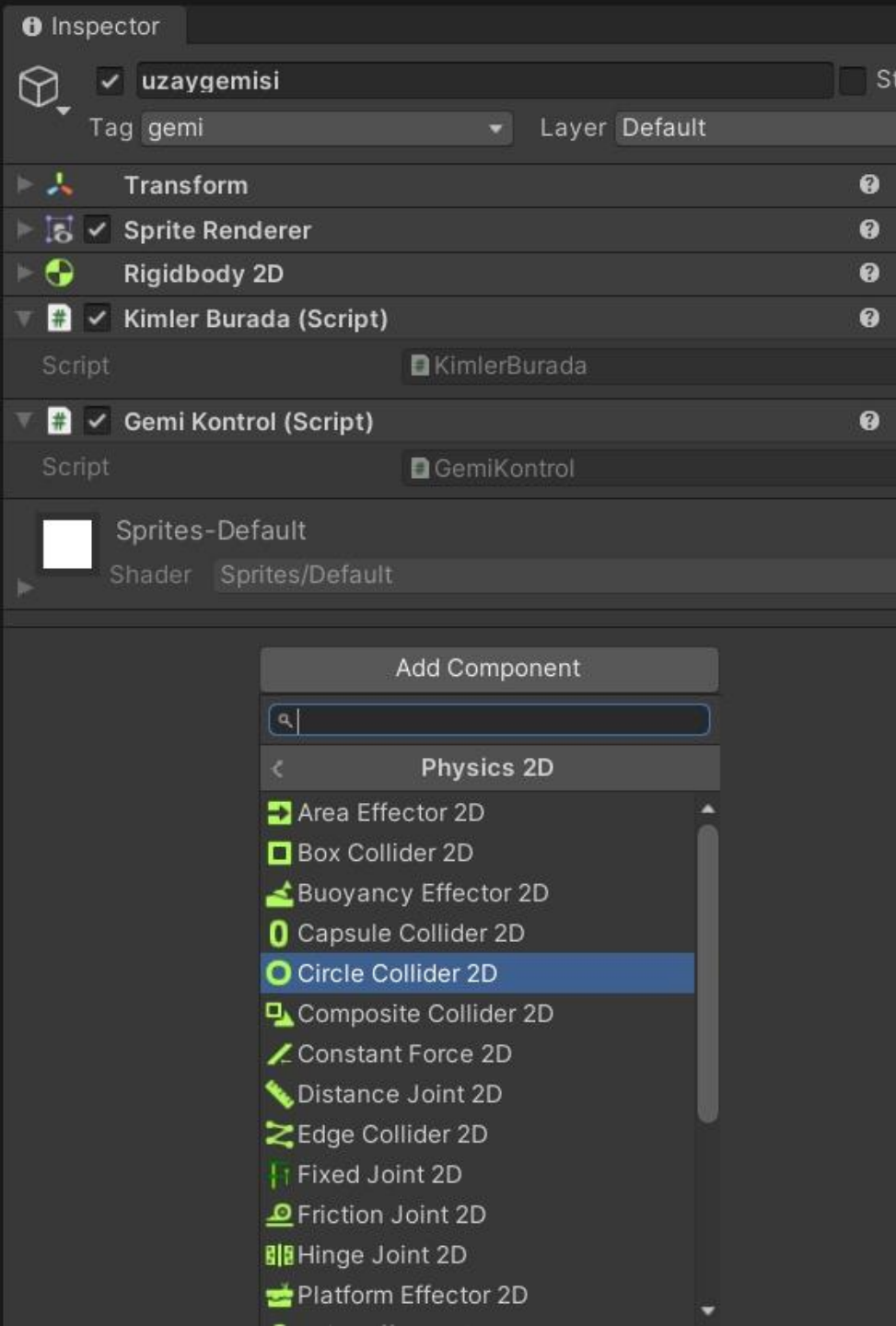
BoxCollider2D

EdgeCollider2D

CircleCollider2D

PolygonCollider2D

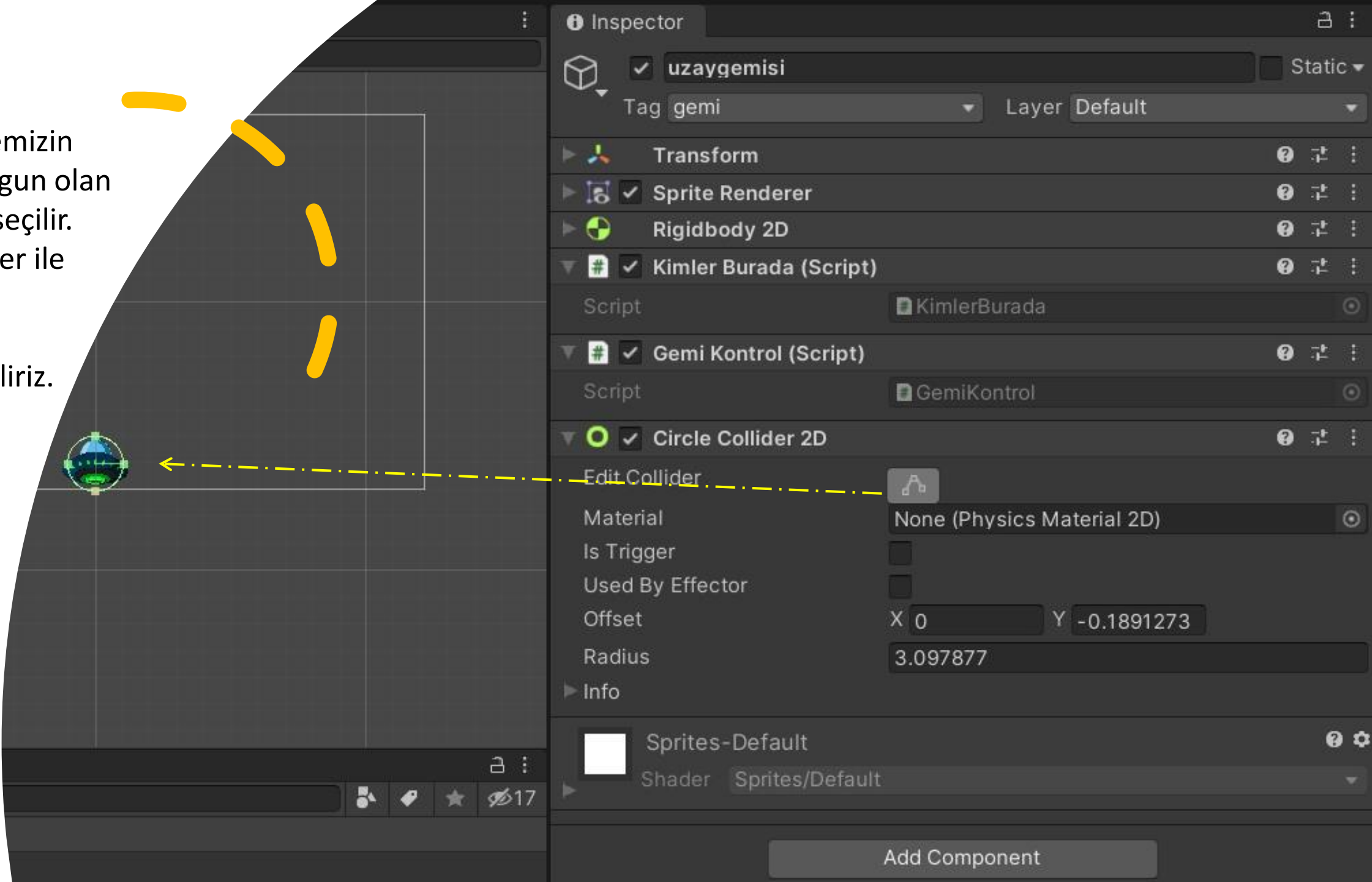
Colliders2D

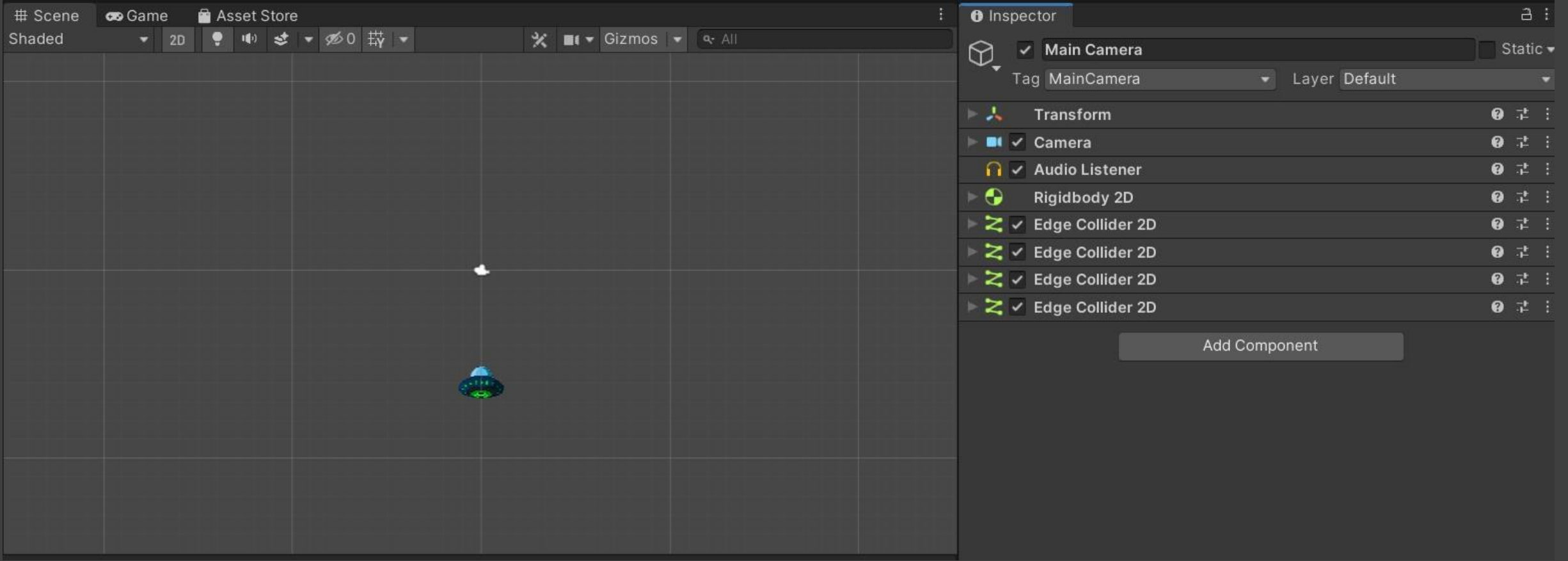


GameObject'e Collider Eklemek

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

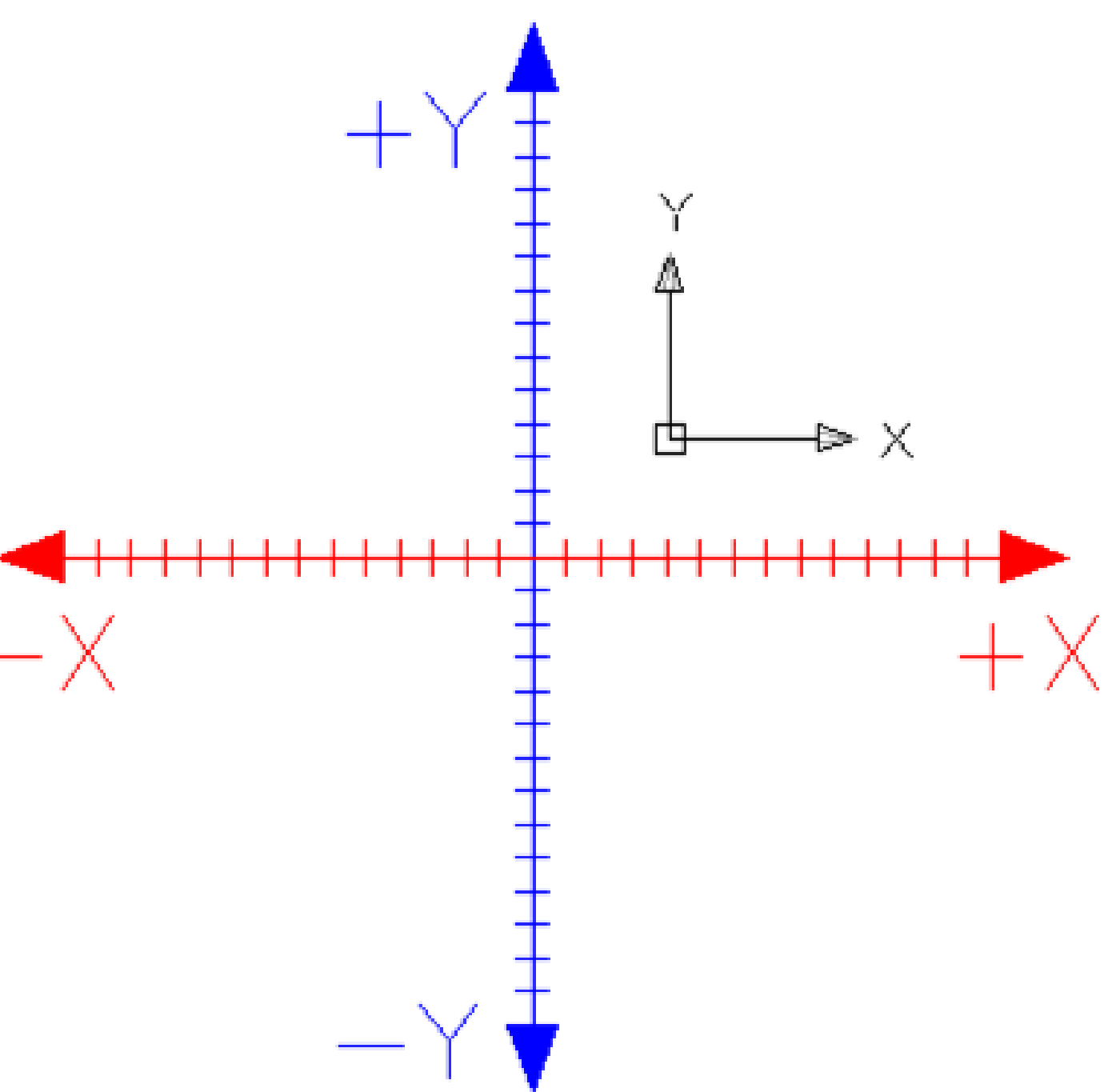
- Oyun objemizin şekline uygun olan Collider'ı seçilir.
- Edit Collider ile Collider'ın sınırlarını ayarlayabiliriz.



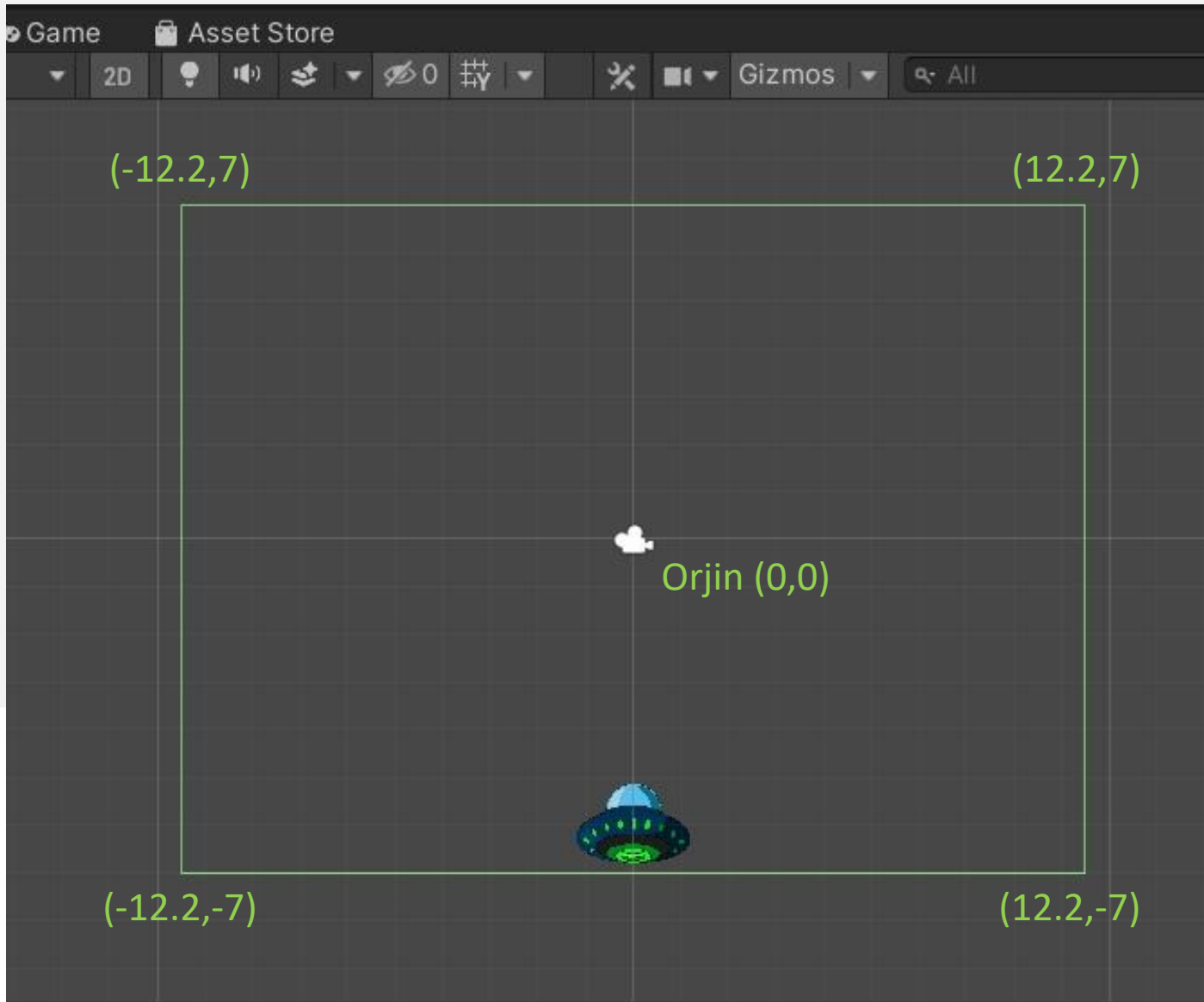


MainCamera'ya Collider Ekleme

- Tüm sınırlarını çizebilmek adına 4 tane Edge 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

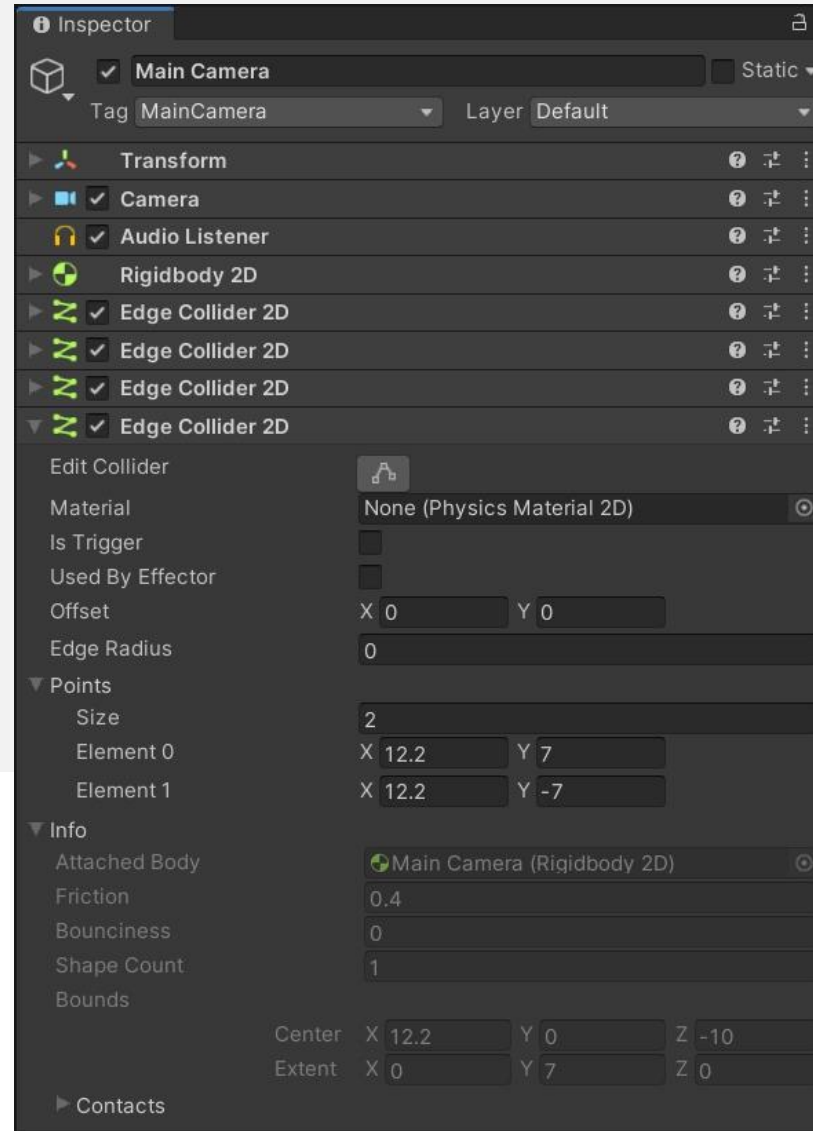


Üst ve Alt
Sınırlara
Edge Collider
Oluşturulması

Edge Collider - Sol Sınır



Edge Collider - Sağ Sınır



Sol ve Sağ
Sınırlara
Edge Collider
Oluşturulması



GemiKontrol Script-1

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

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



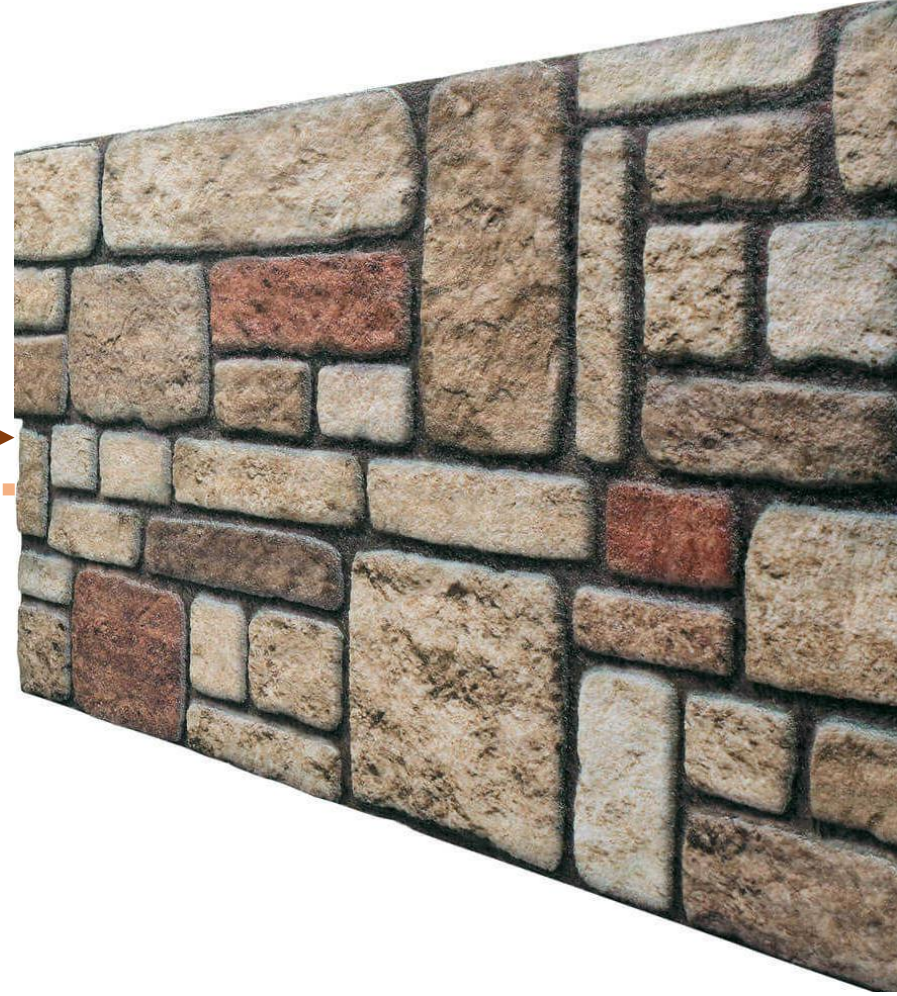
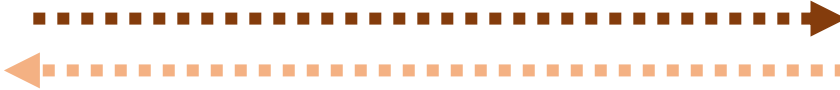
GemiKontrol Script-2

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

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

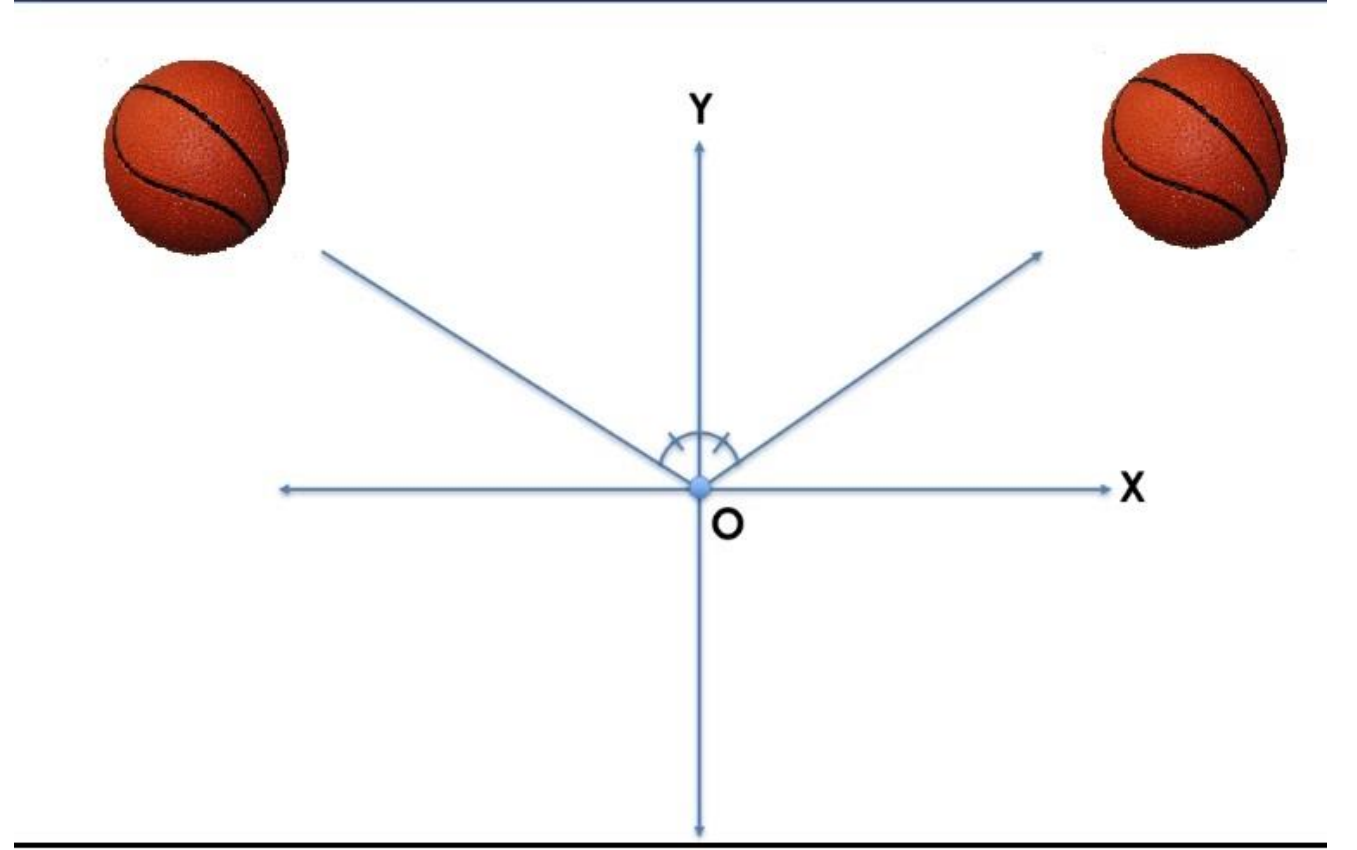

Fizik Materyalleri

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



Fizik Materyalleri

- Topu açısai 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.

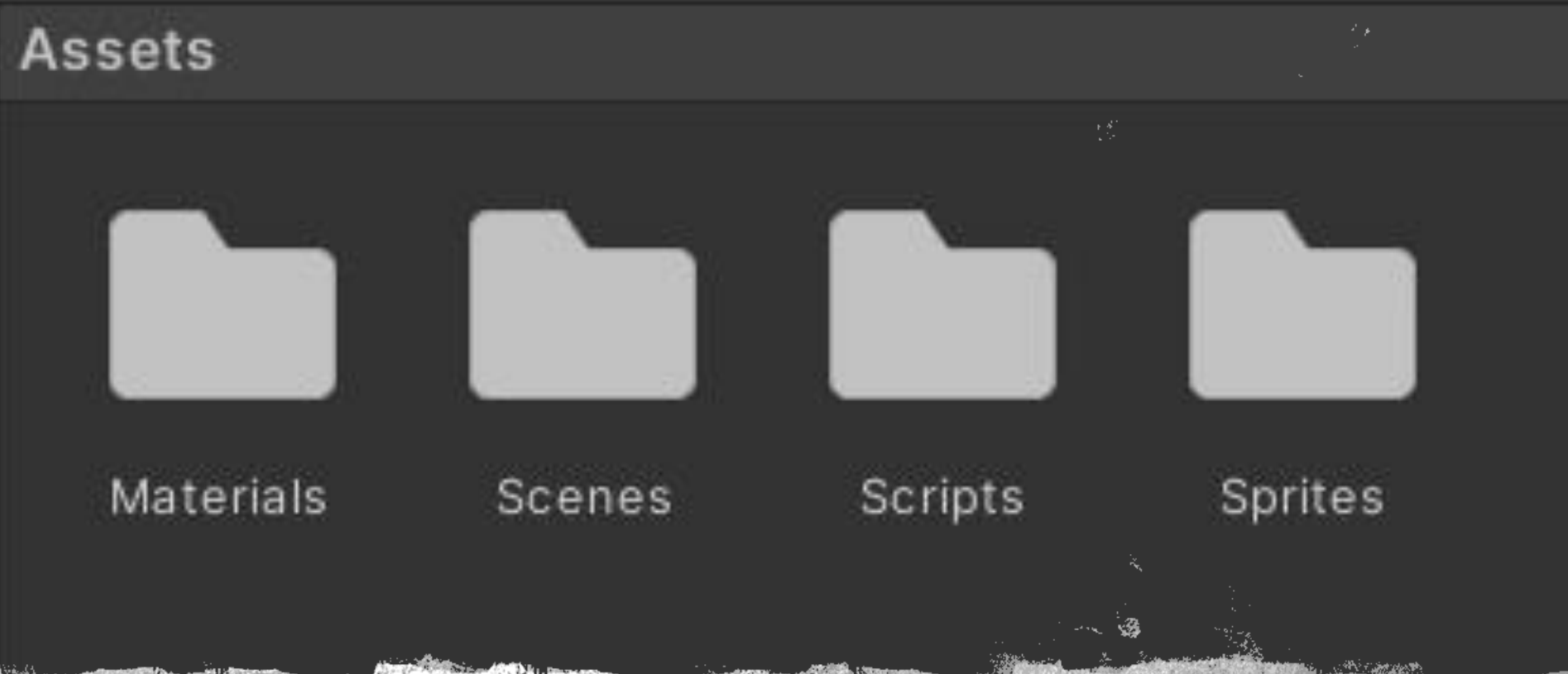
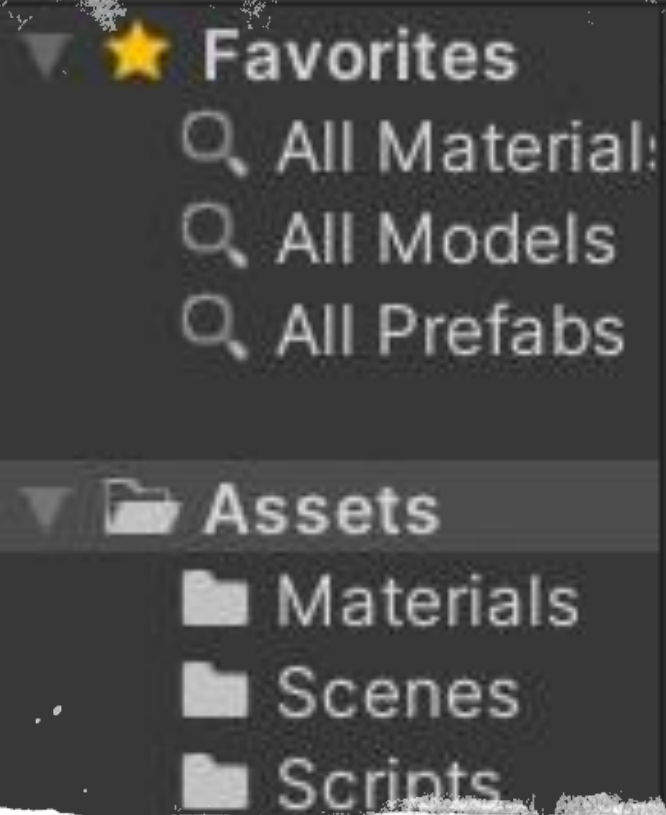


Fizik Materyalleri

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

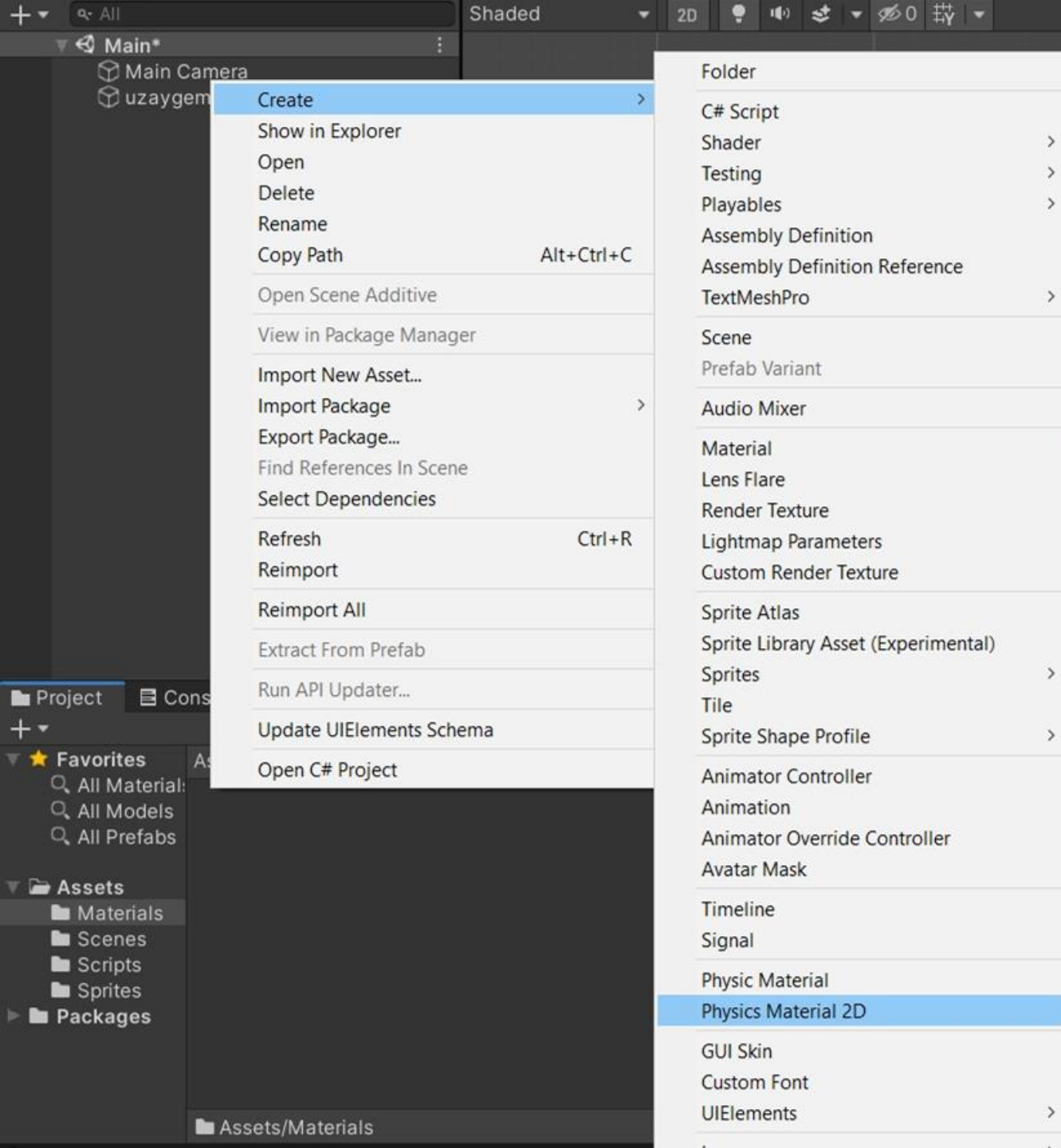
Fizik Materyalleri

- 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.

Assets > Materials



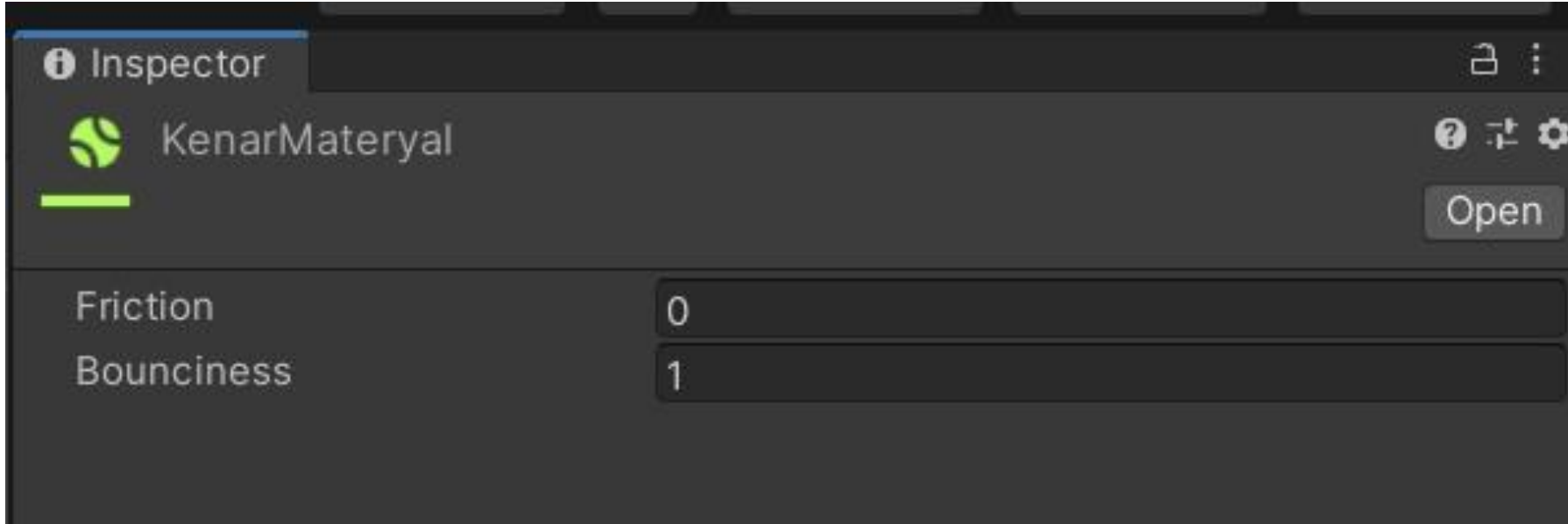
KenarMate...

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

Kenar Materyal

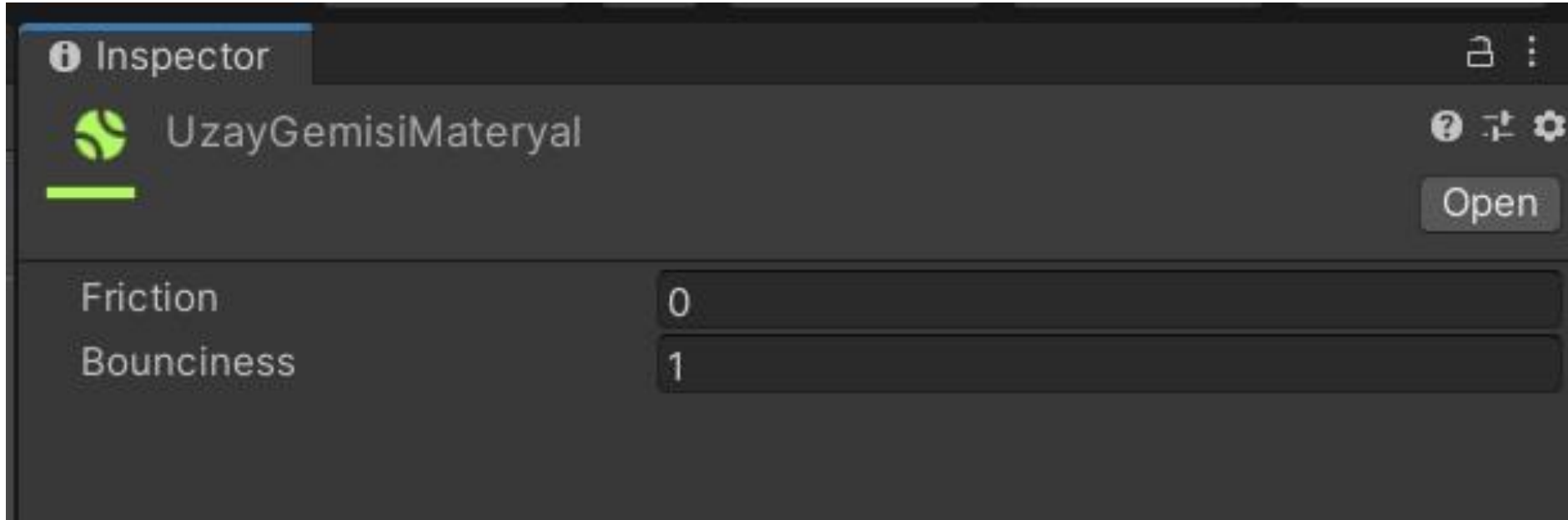


Assets/Materials/KenarMateryal



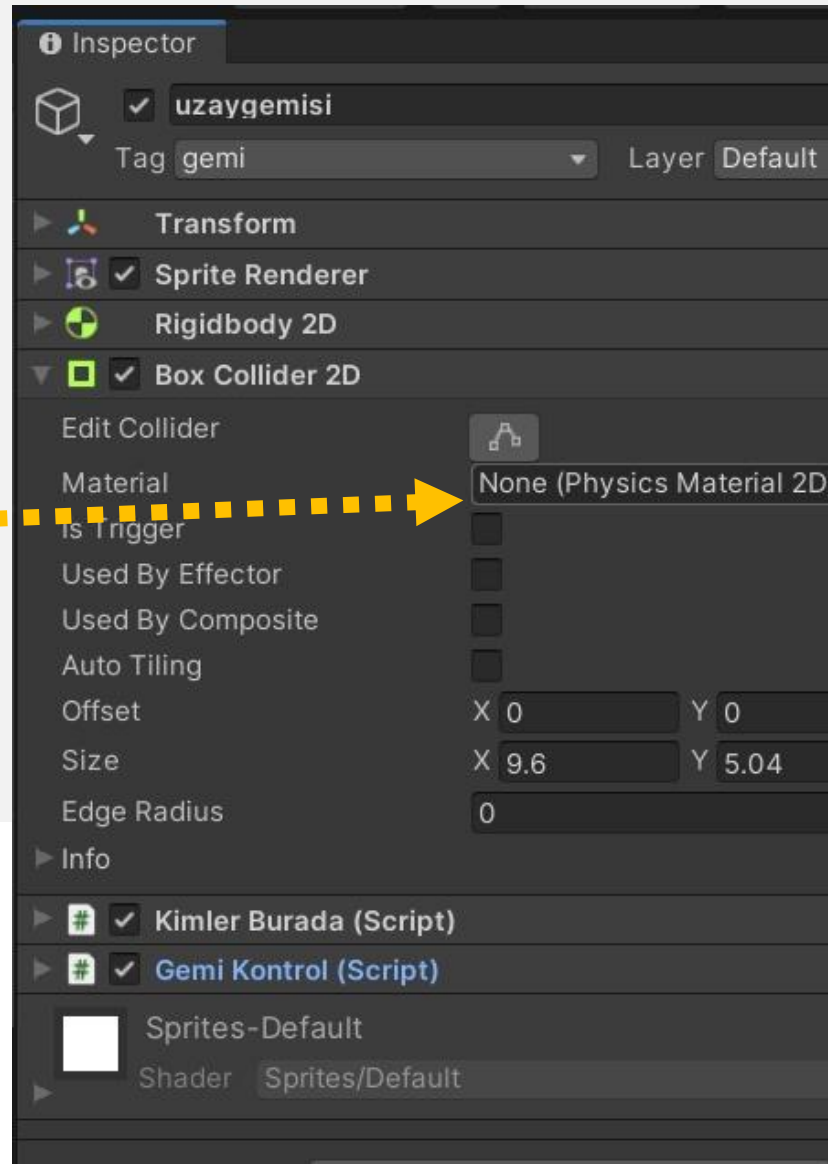
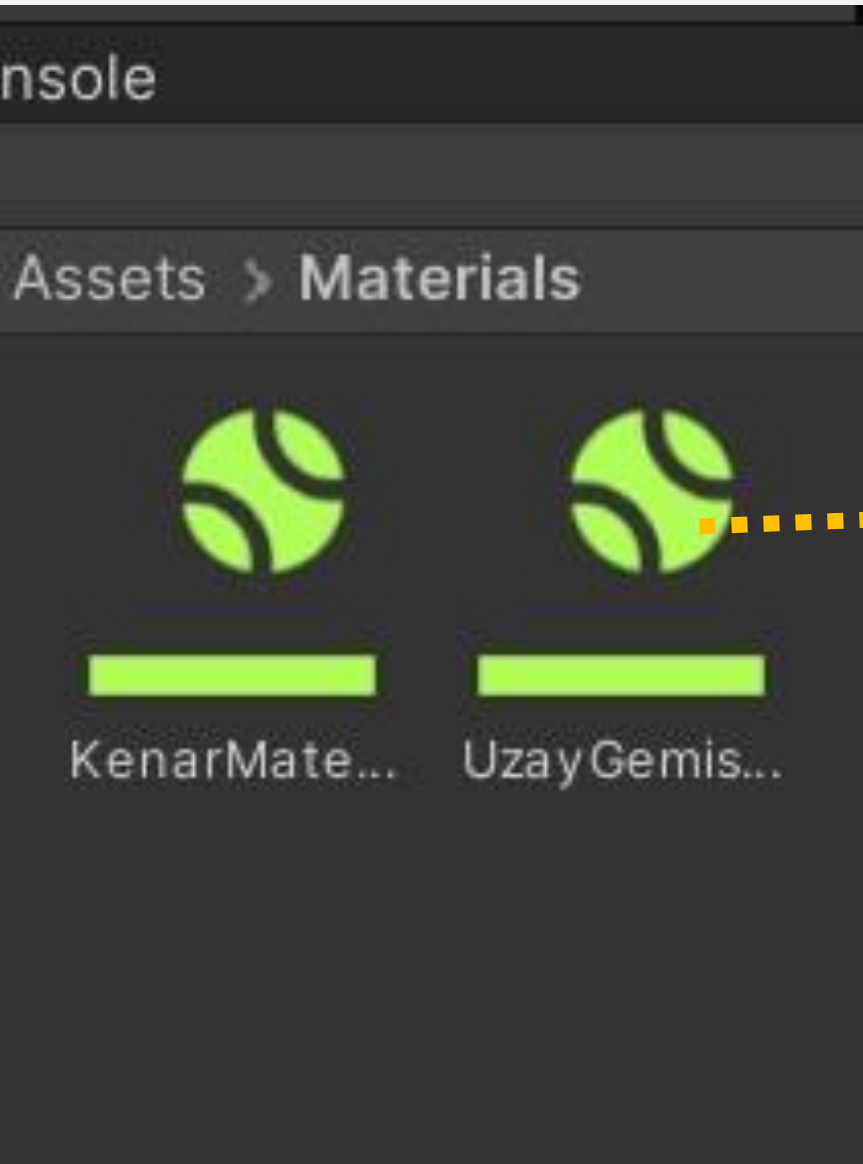
Kenar Materyal Inspector

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



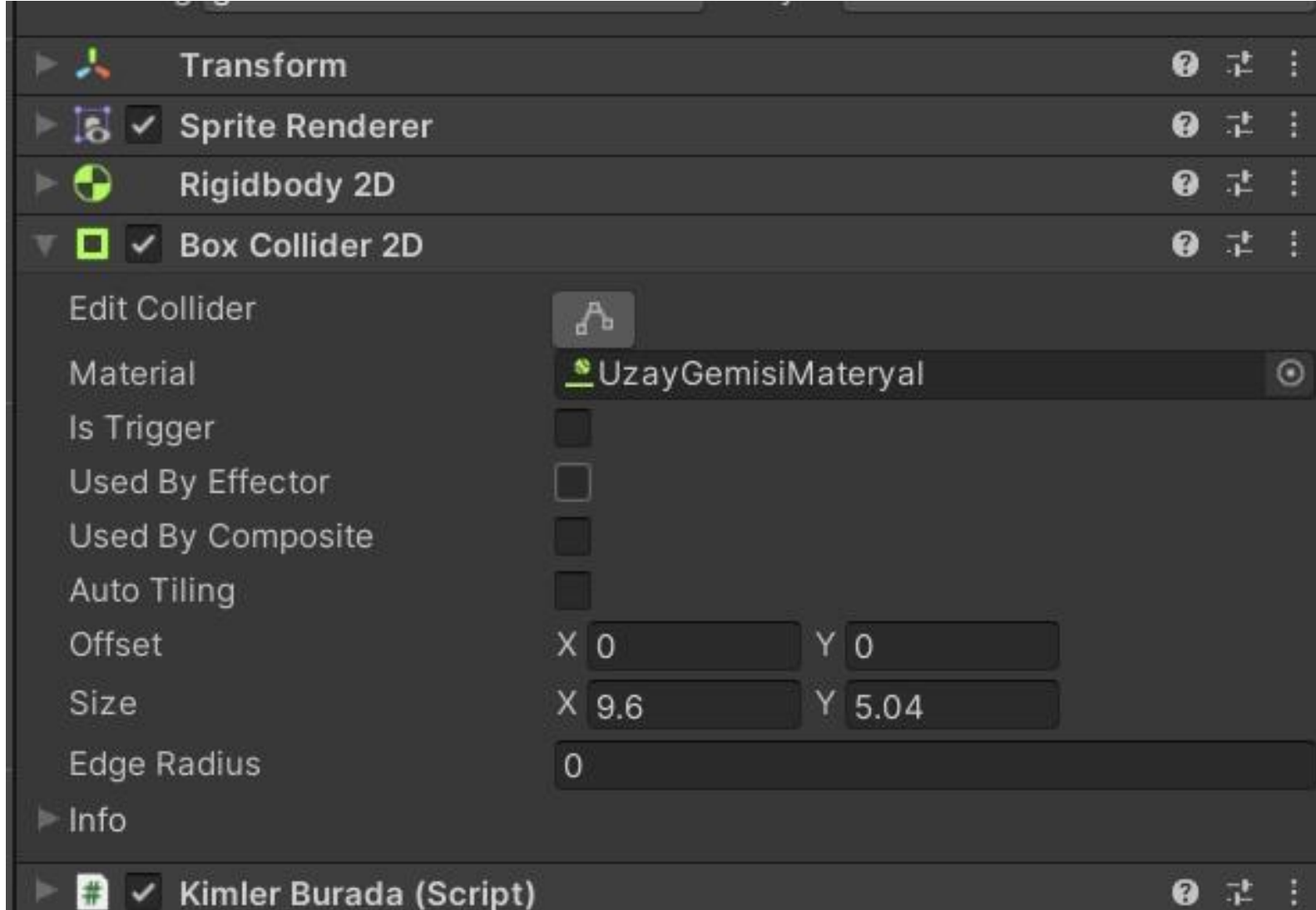
Uzay Gemisi Materyal Inspector

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

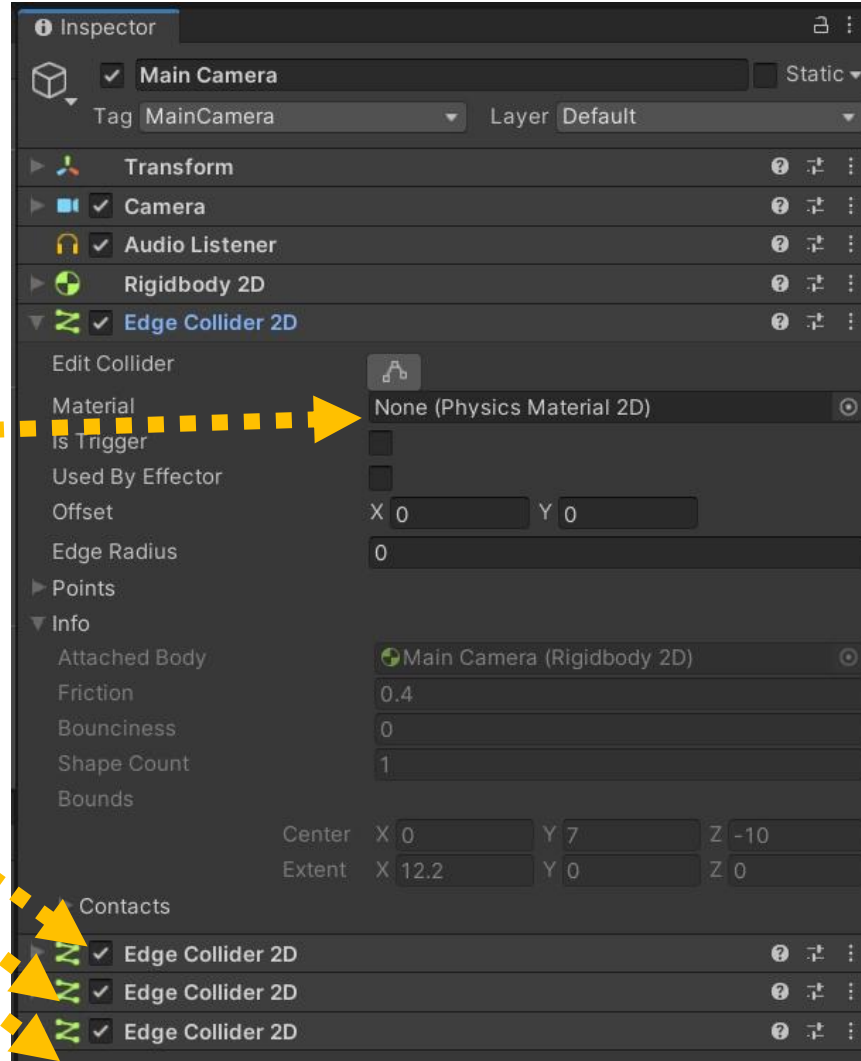
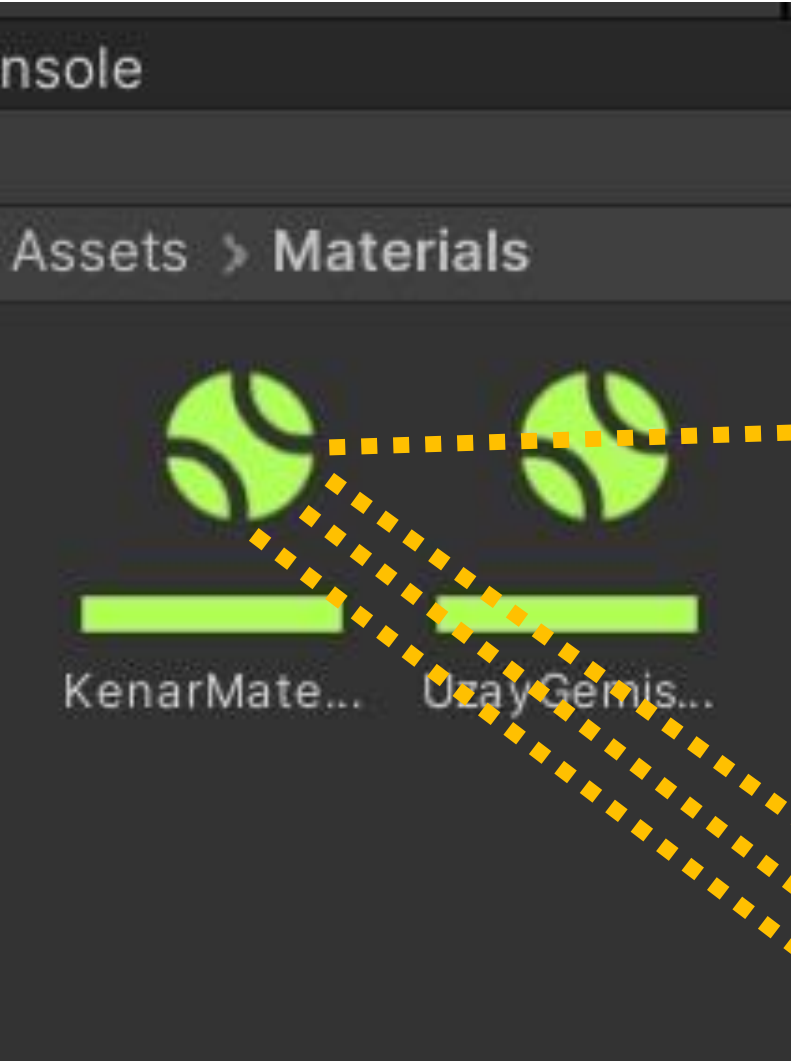


Uzay Gemisi Material

Materyalimizi sürükleyerek uzaygemisi objemizin Material kısmına bırakırız.



Materyali
ekledik.



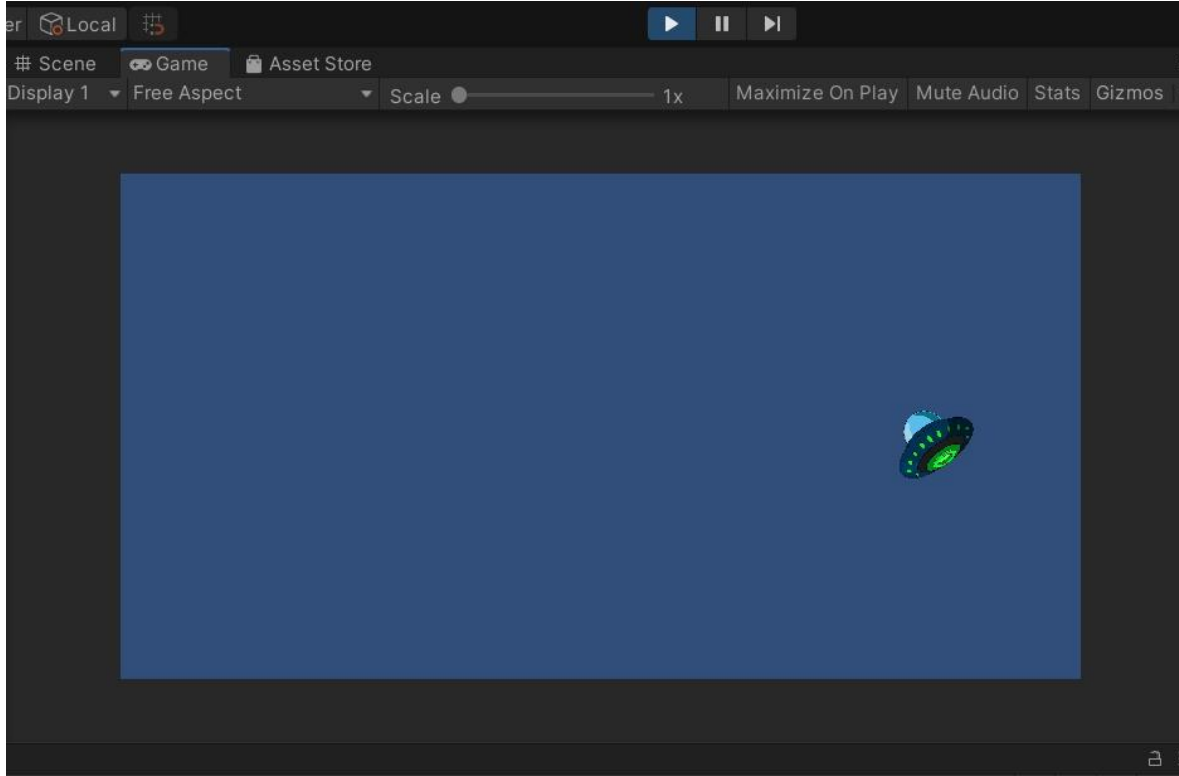
Main Camera Material

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



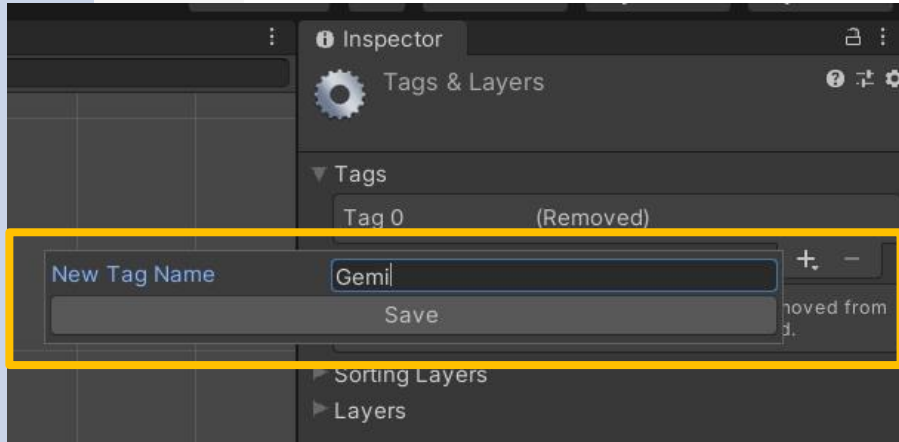
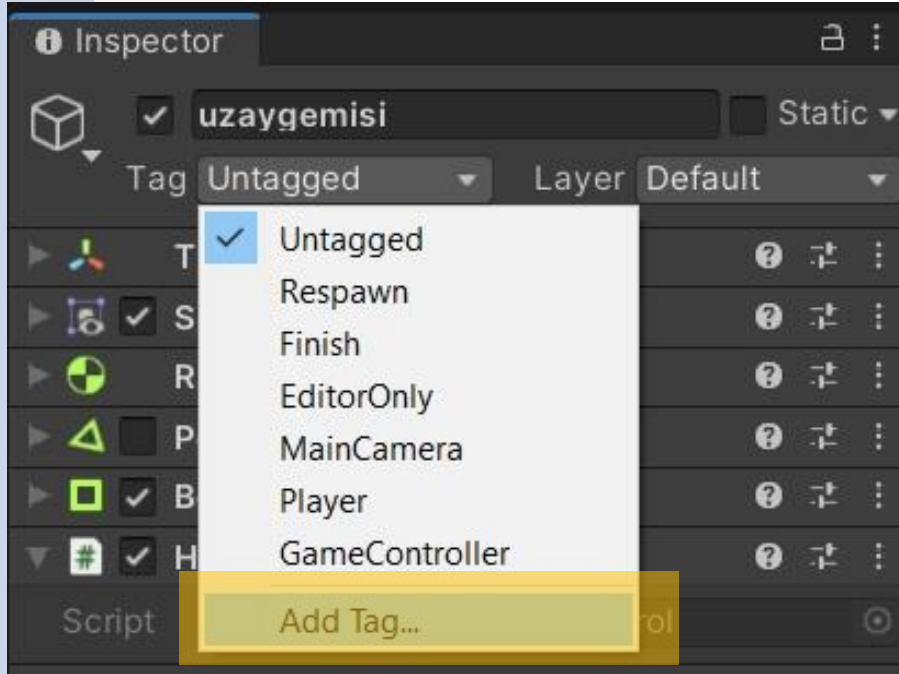
Materyali
ekledik.





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.

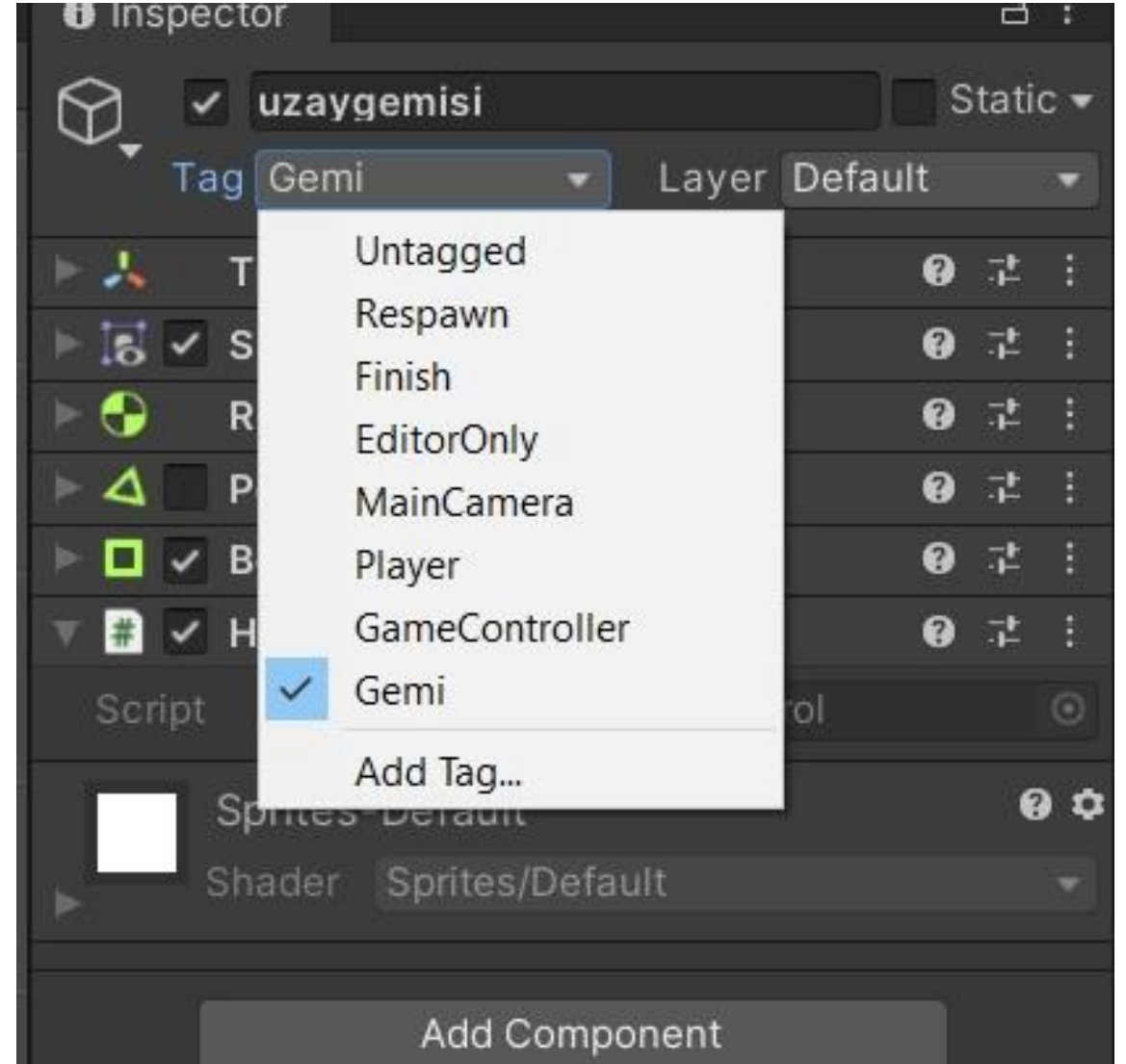


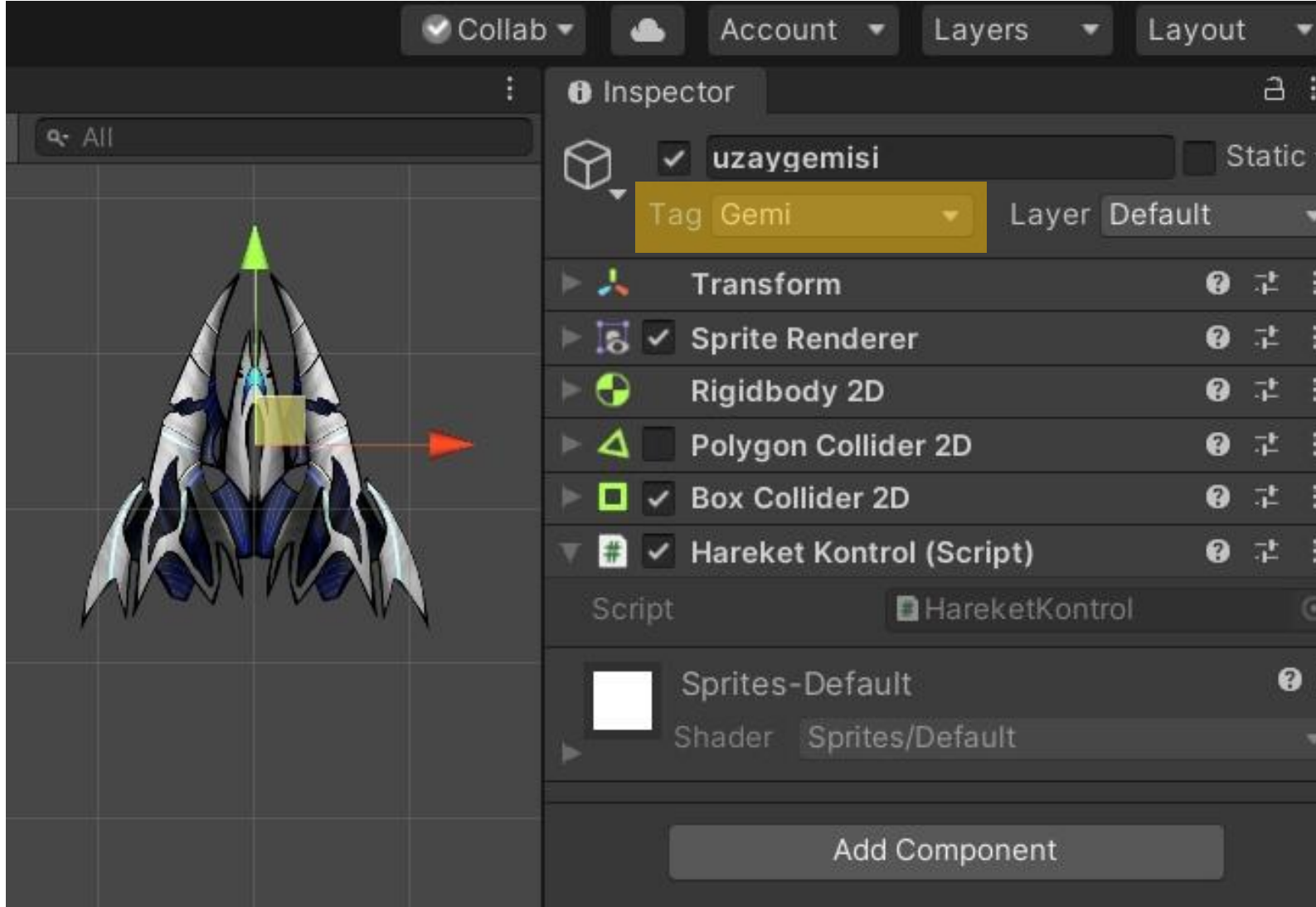
Tag Değeri Ekleme

- 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 Deęeri Atamak

- Inspector penceresinde yer alan Tag seçenekleri arasından objeye atamak istedięimiz Tag deęeri seçilir.





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

Version: 2019.3

Collider2D

- ColliderDistance2D
- Collision
- Collision2D
- Color
- Color32
- ColorUtility
- CombineInstance
- Compass
- Component
- + CompositeCollider2D

Messages

OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).

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.

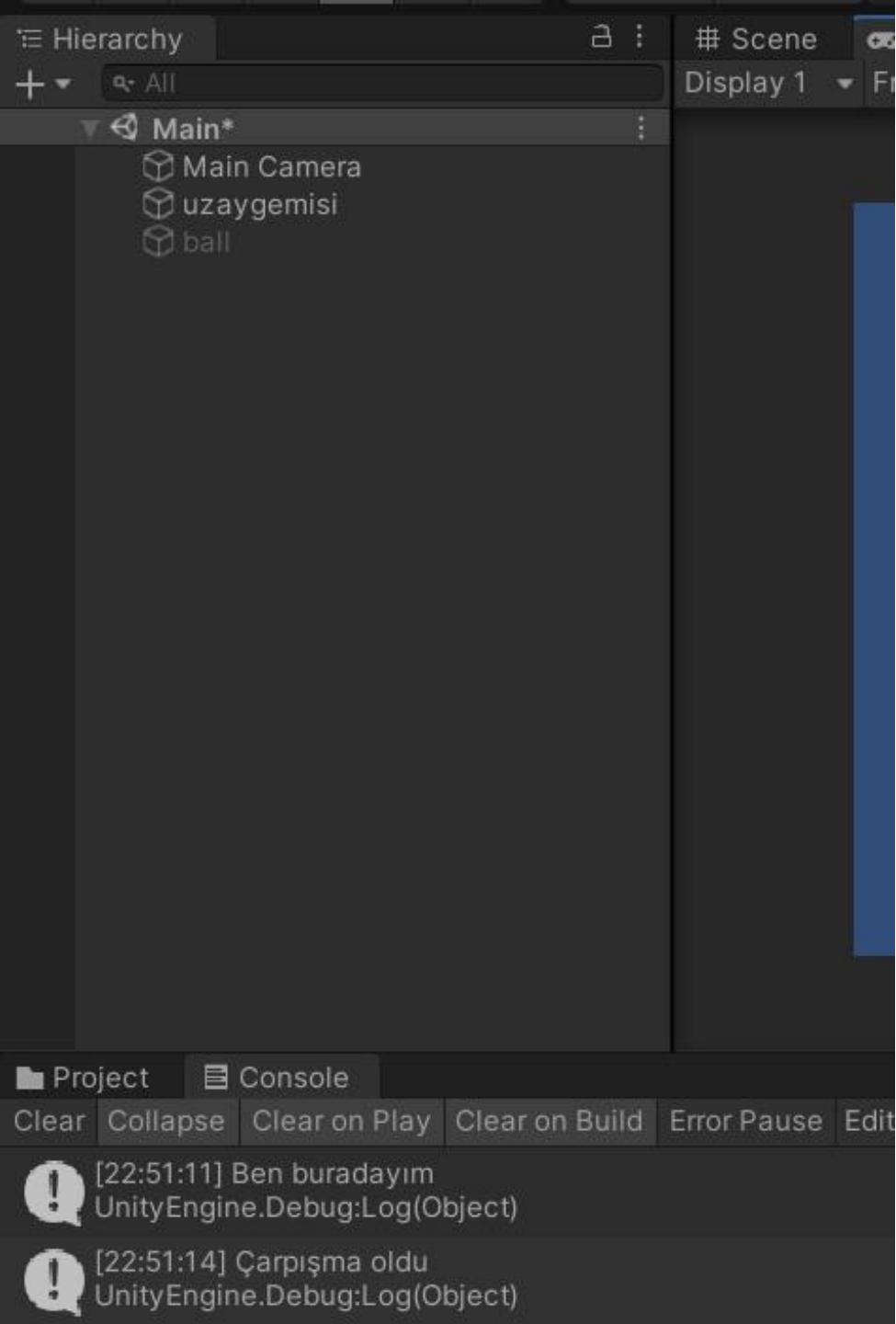
Notes: Collision events will be sent to disabled MonoBehaviours, to allow enabling Behaviours in response to collisions.

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.

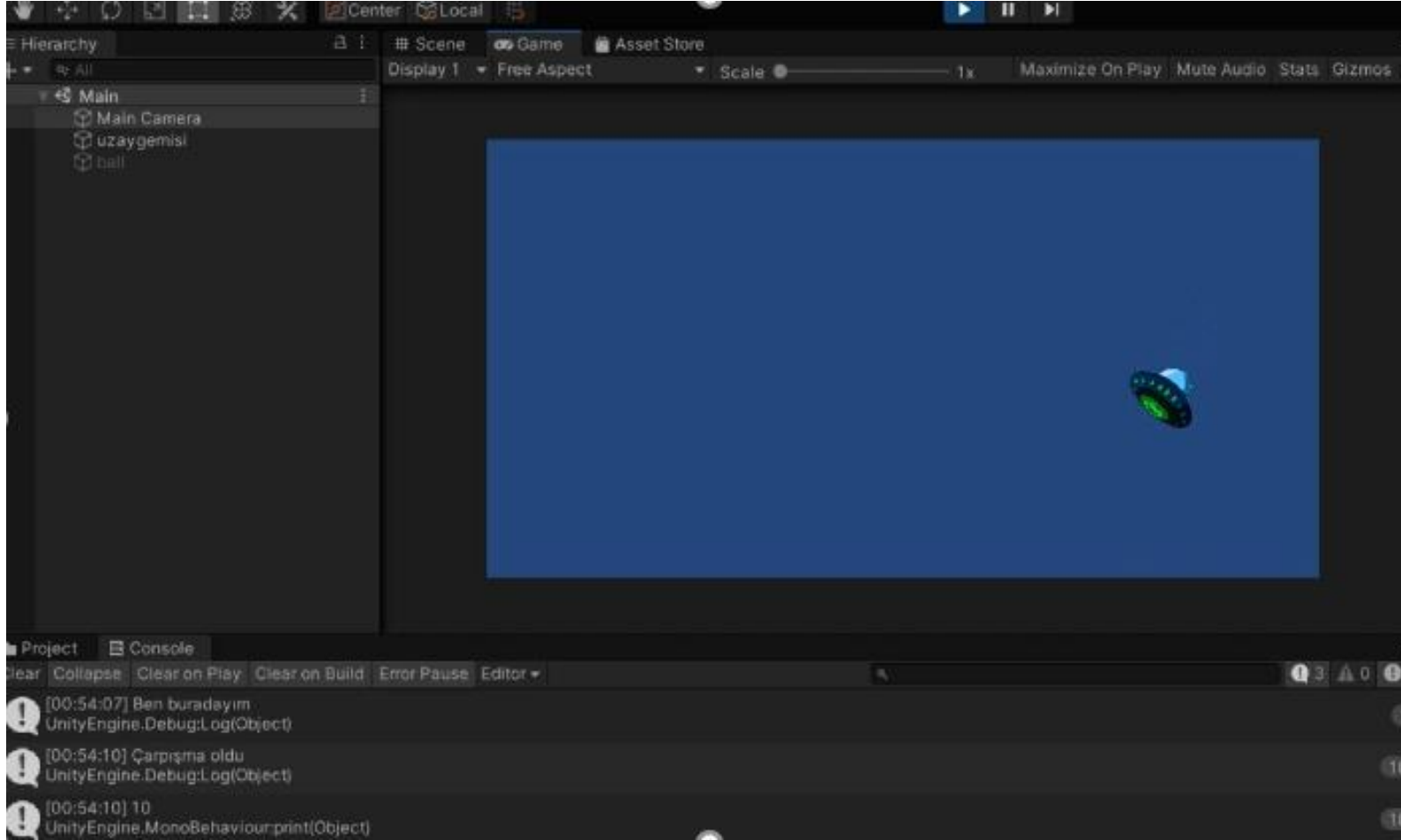
```
private void  
OnCollisionEnter2D(Collision2D collision)  
{  
    Debug.Log("Çarpışma oldu");  
}
```

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);
}
```

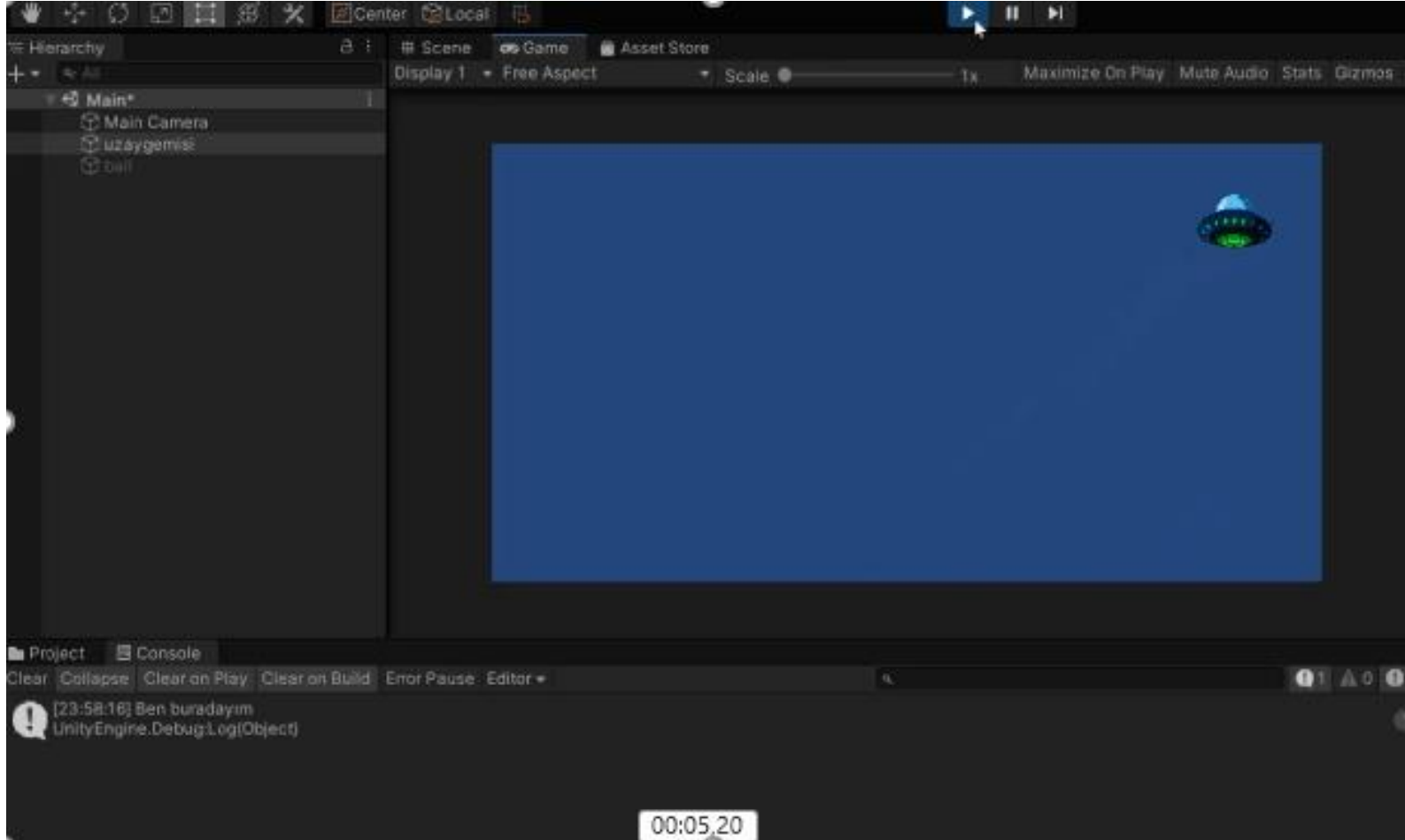


Çarpışma
Hasar
Uygula
Script Play

Çarpışma Collider Component Pasif Yapma Script

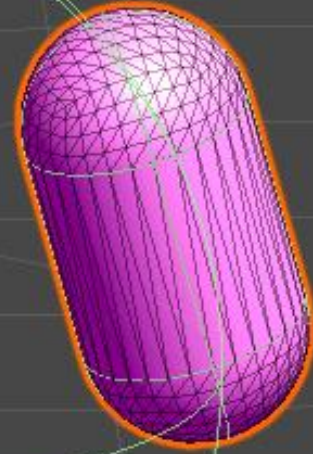
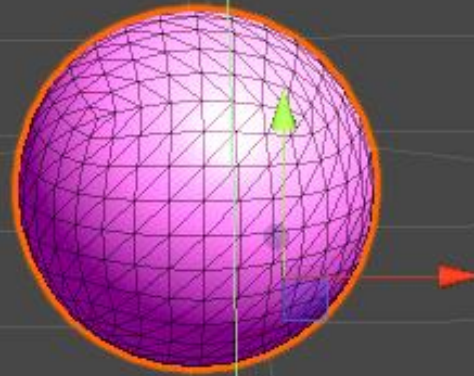
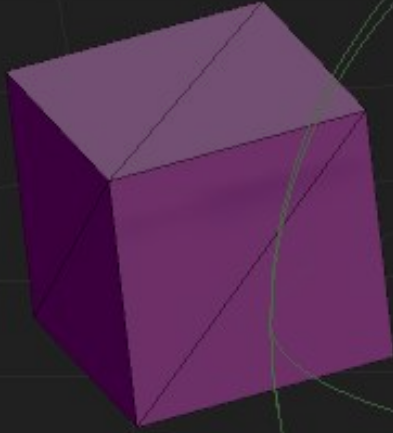
- Çarpışma olduğunda Main Camera collider componenti aktif ise pasif yapalım.

```
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 çarpıştırıcısına 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");  
    }  
}
```

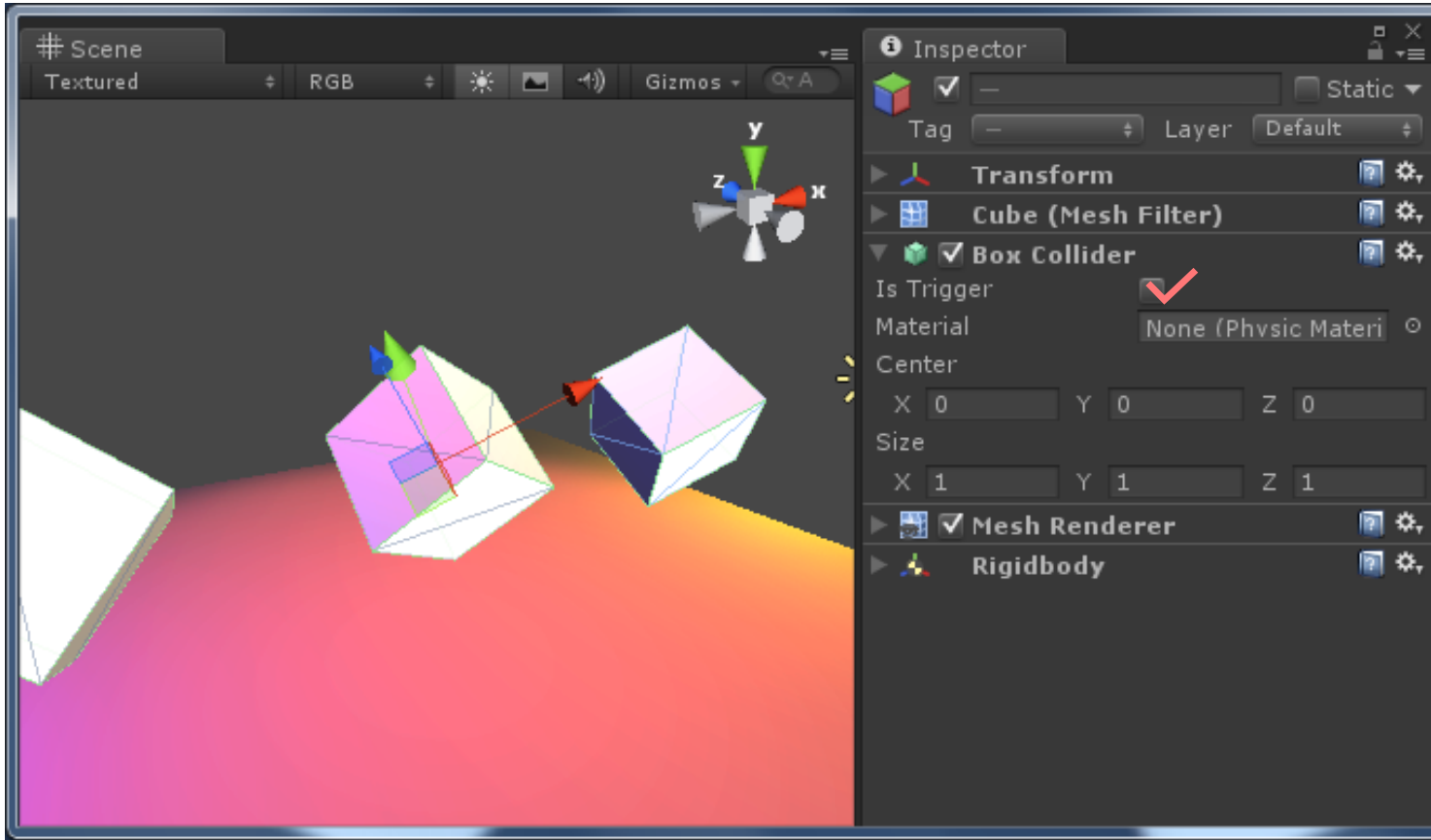
OnCollisionExit 2D

- 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 ");
    }
}
```

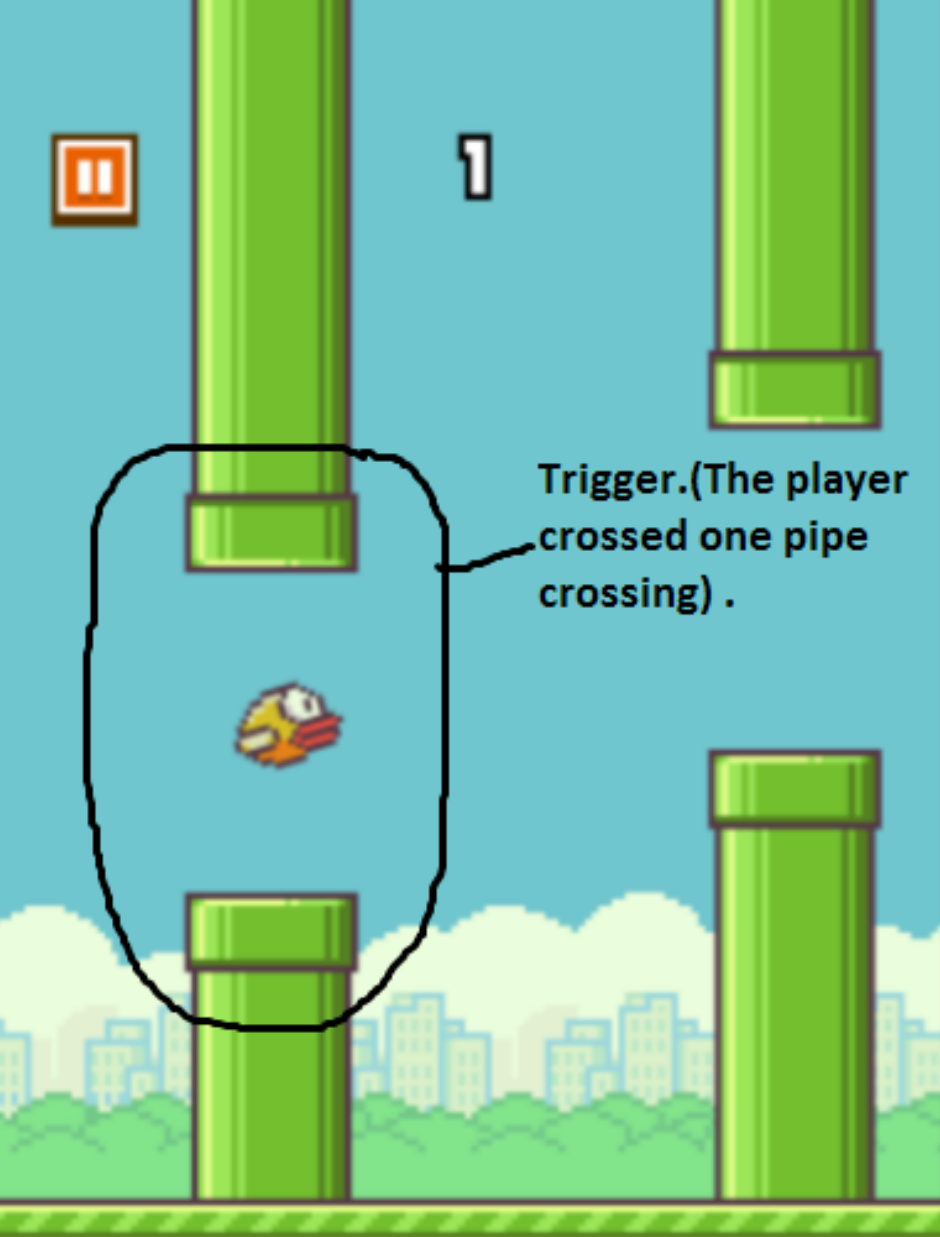
The image shows the Unity 2019.3.7f1* <DX11> interface. The Hierarchy panel on the left shows the scene structure: Main* (Main Camera, uzaygemisi, ball). The Inspector panel on the right shows the properties of the selected object, uzaygemisi, which has a Box Collider 2D component. The console at the bottom shows the following logs:

```
UnityEngine.MonoBehaviour.print(Object)
[18:56:11] Main Camera tag'li nesne ile çarpışmayı bıraktı
UnityEngine.Debug.Log(Object)
[18:56:11] Main Camera tag'li nesne ile çarpışmayı bıraktı
UnityEngine.Debug.Log(Object)
[18:56:11] Main Camera tag'li nesne ile çarpışıyor
UnityEngine.Debug.Log(Object)
[18:56:11] Main Camera tag'li nesne ile çarpışıyor
UnityEngine.Debug.Log(Object)
Main Camera tag'li nesne ile çarpışıyor
UnityEngine.Debug.Log(Object)
GemiKontrol:OnCollisionStay2D(Collision2D) (at Assets/Scripts/GemiKontrol.cs:86)
```



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

Trigger



Trigger

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

OnCollisionEnter

- OnTriggerEnter bir objenin çarpma durumunda 1 kere çalışır.

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

OnCollisionStay

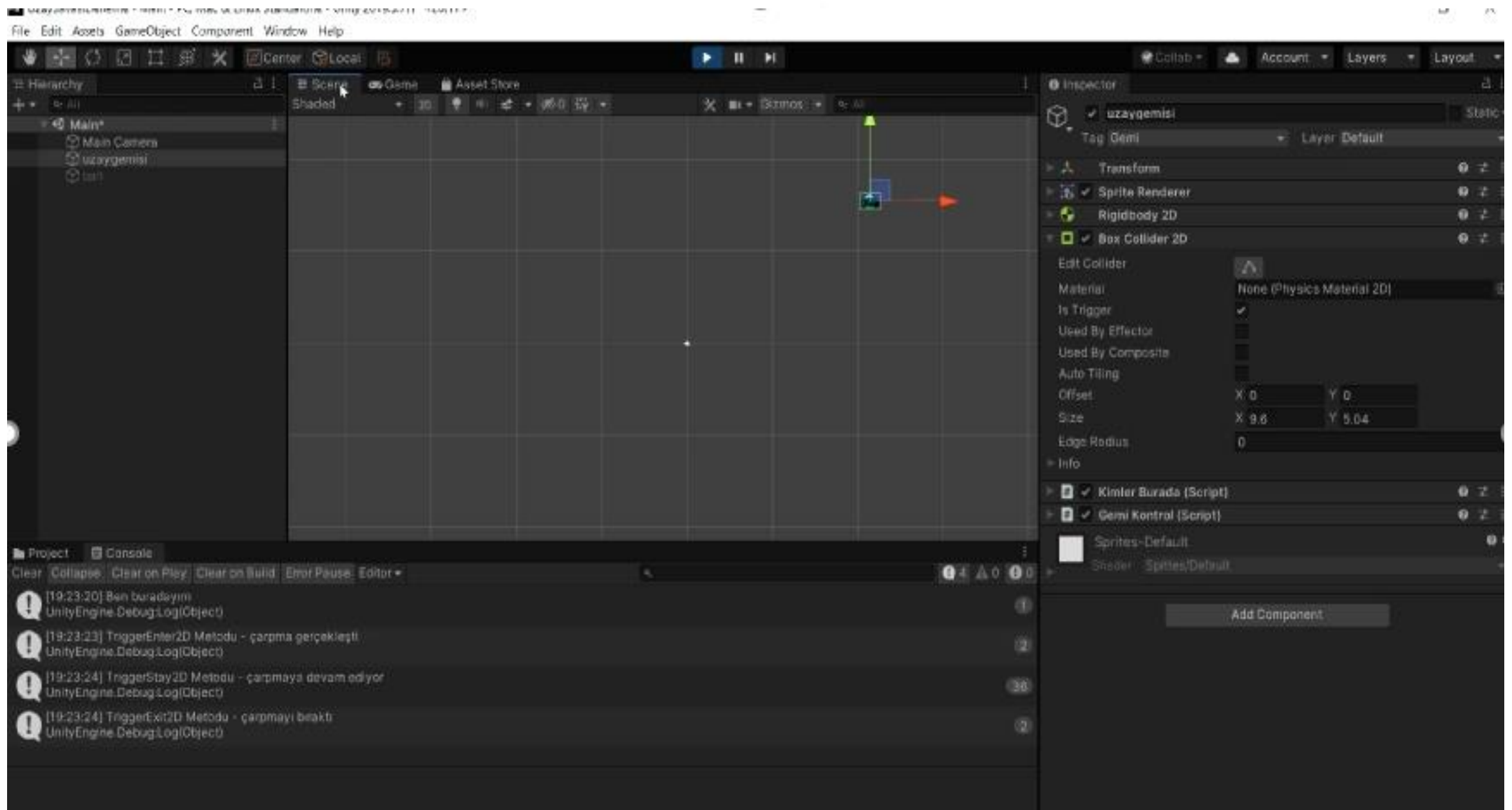
- OnTriggerStay bir objenin çarpma durumu sürdüğü sürece çalışır.

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

OnCollisionExit

- OnTriggerExit bir objenin çarpma durumu bittiğinde 1 kere çalışır.

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

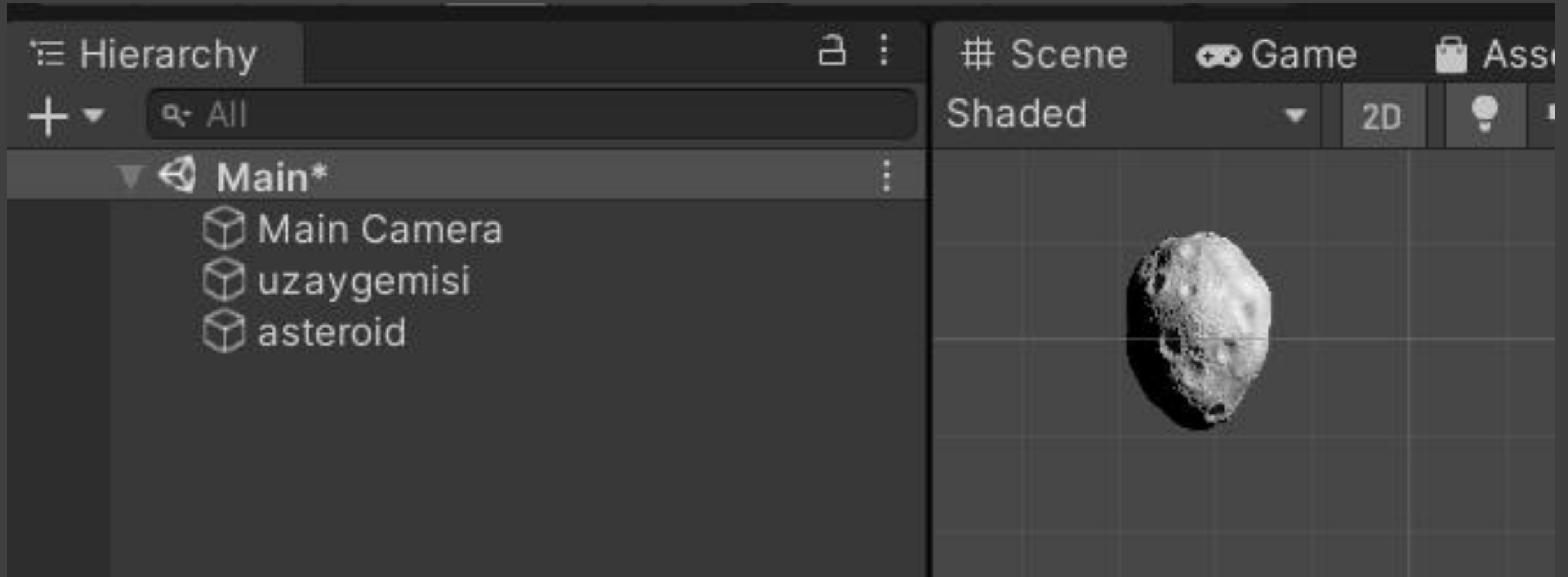


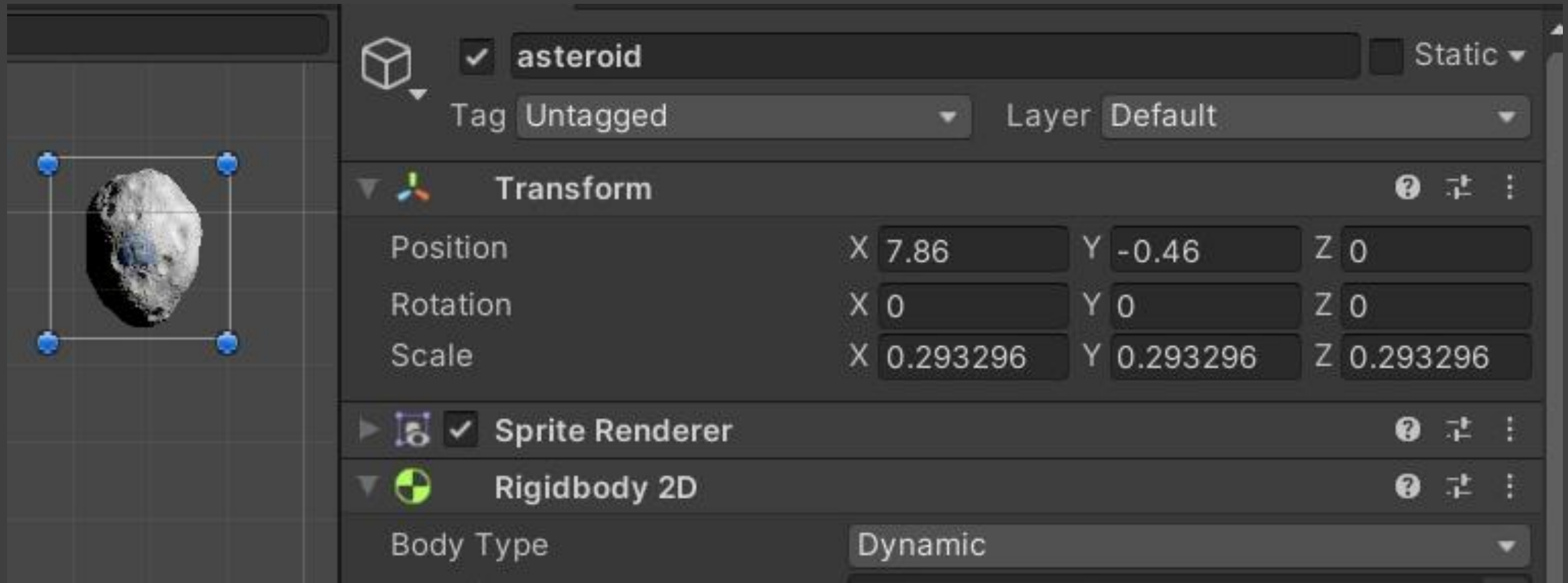
Çoklu Oyun Objeleri *Prefab*



Oyun Objesi

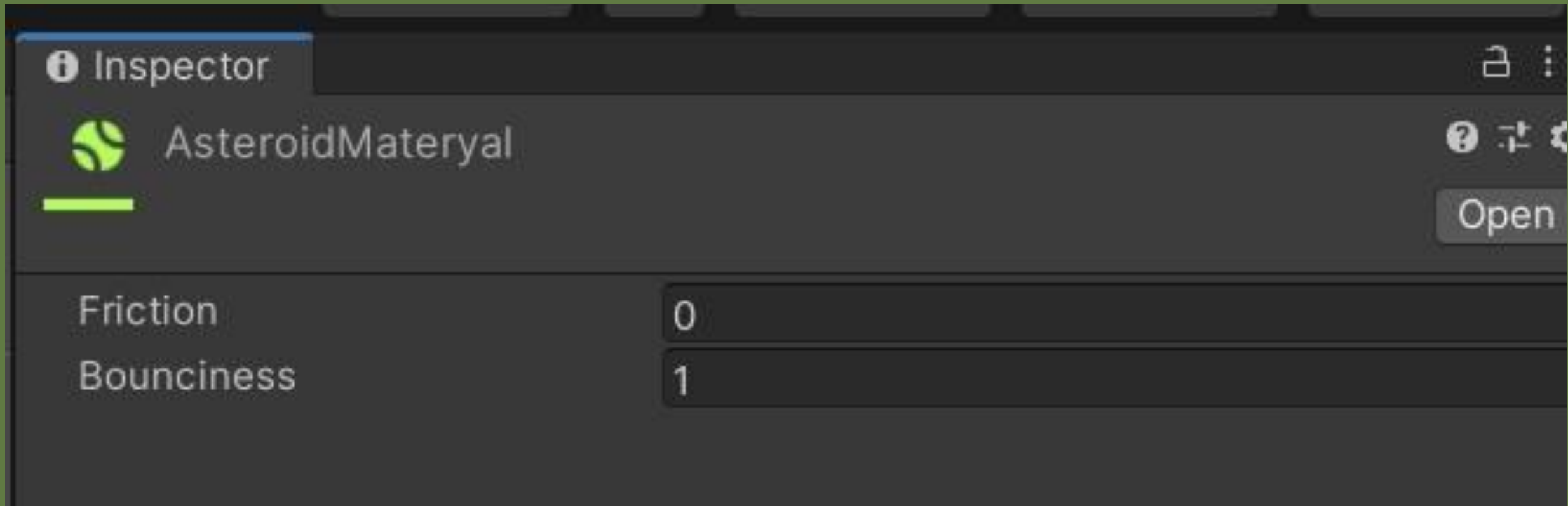
- Projeye bir Sprite oyun objesi ekleriz.



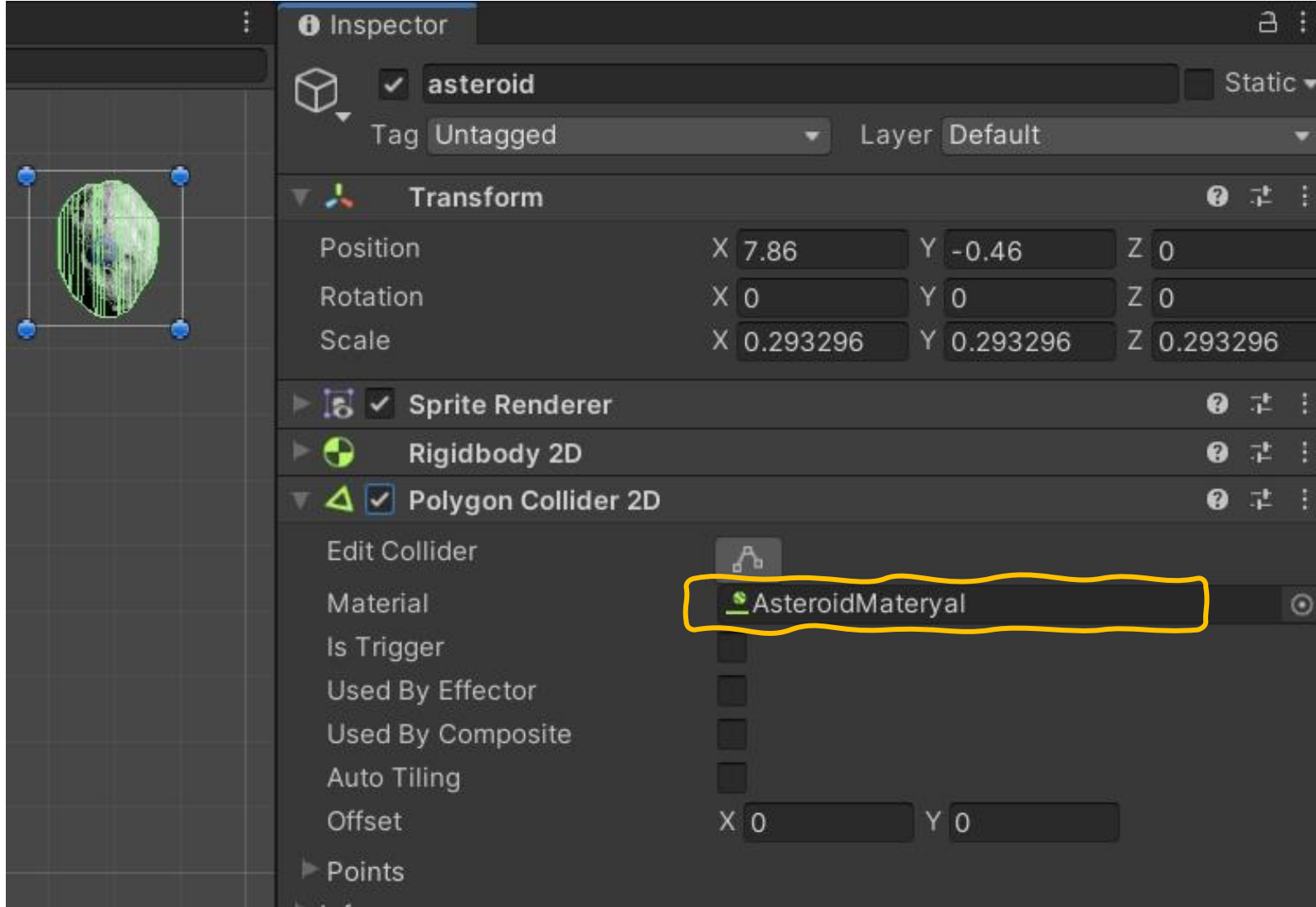


Rigidbody 2D
Component

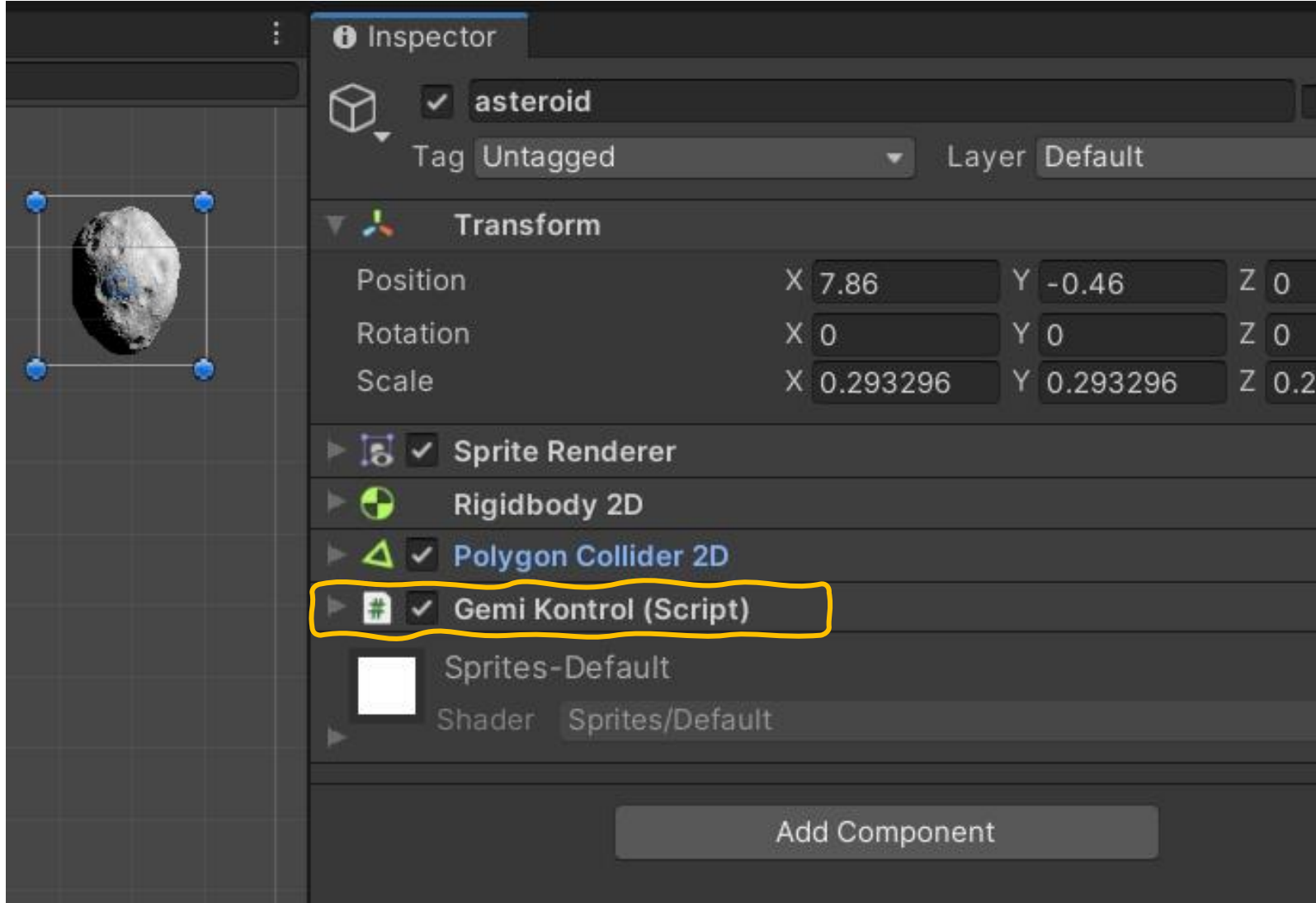
Oyun objemize Rigidbody 2D Component'ini ekledik.



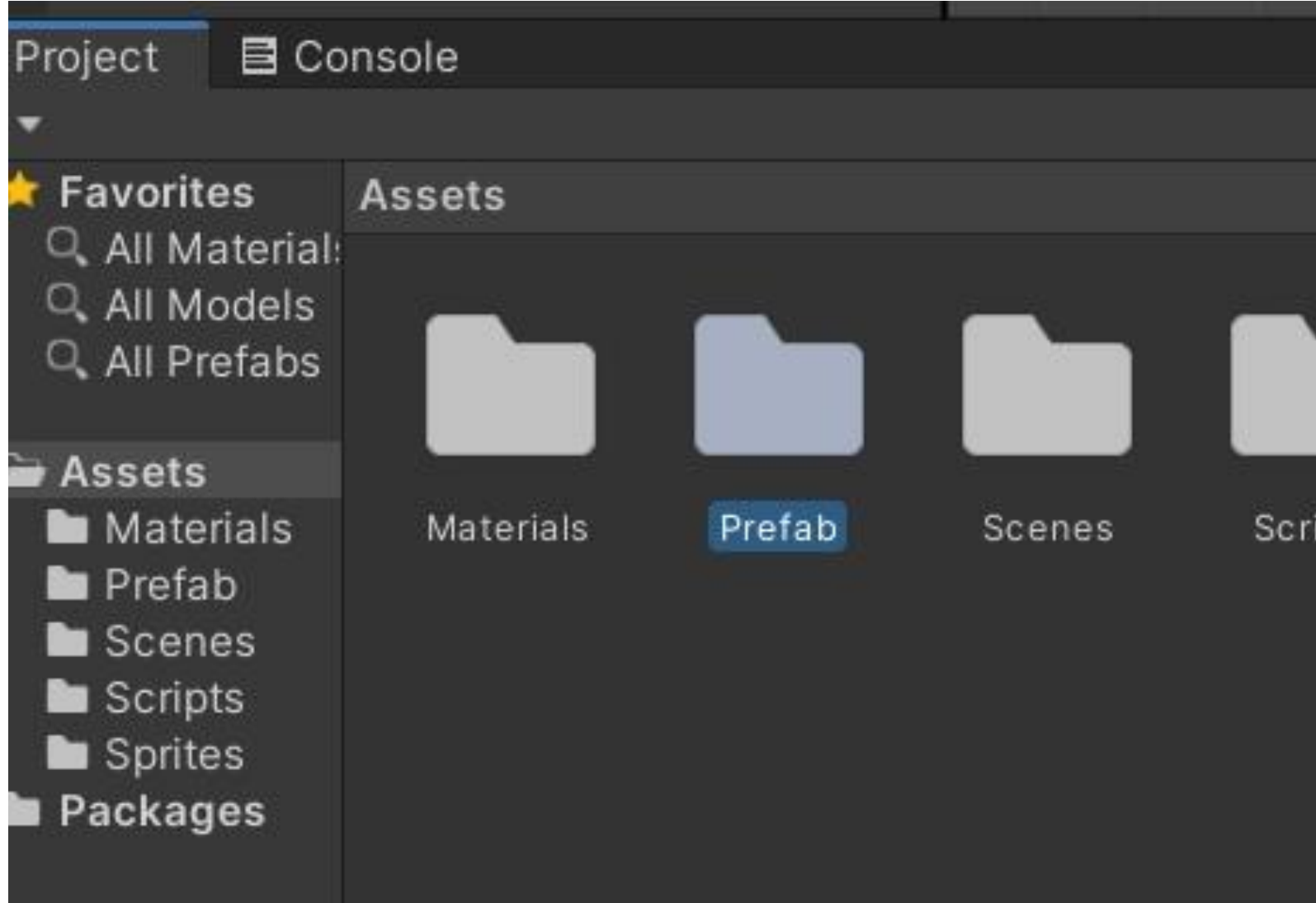
AsteroidMaterial ekledik



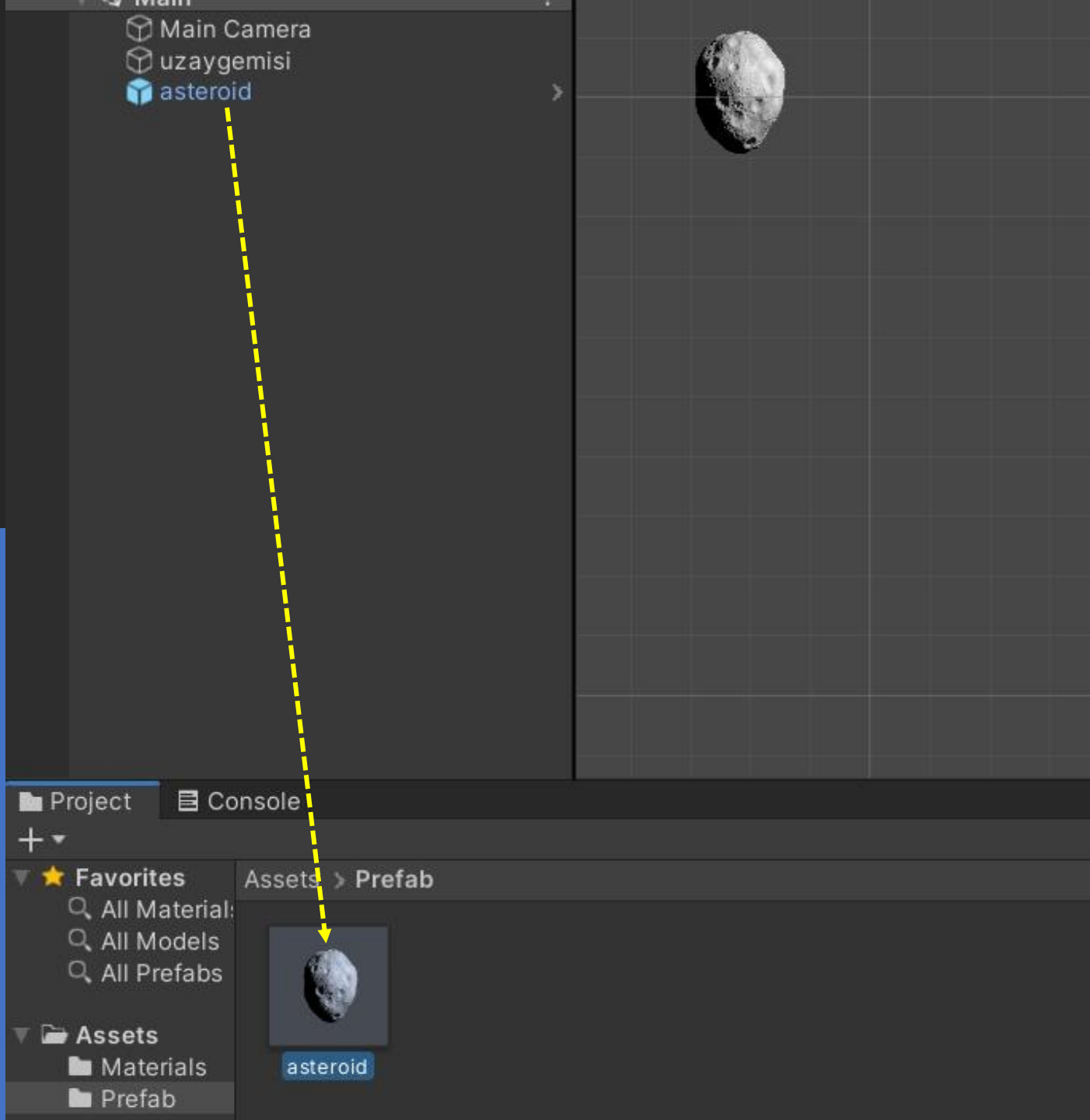
Objemize uygun olan Collider'ı oyun objemize ekleyerek, Material kısmından bu obje için oluşturmuş olduğumuz materyali seçeriz.



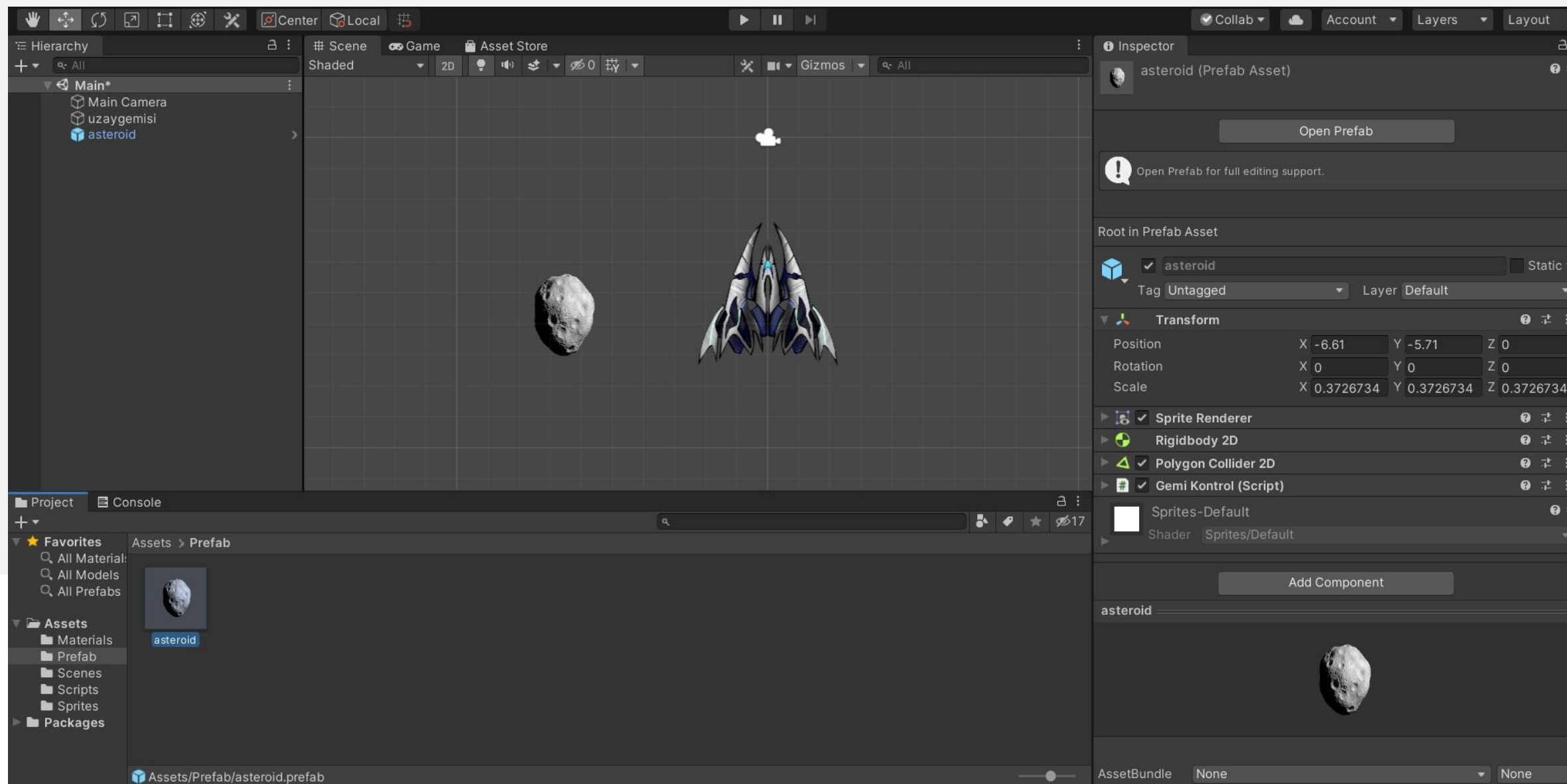
Oyun
Objesi
Hareket
Script

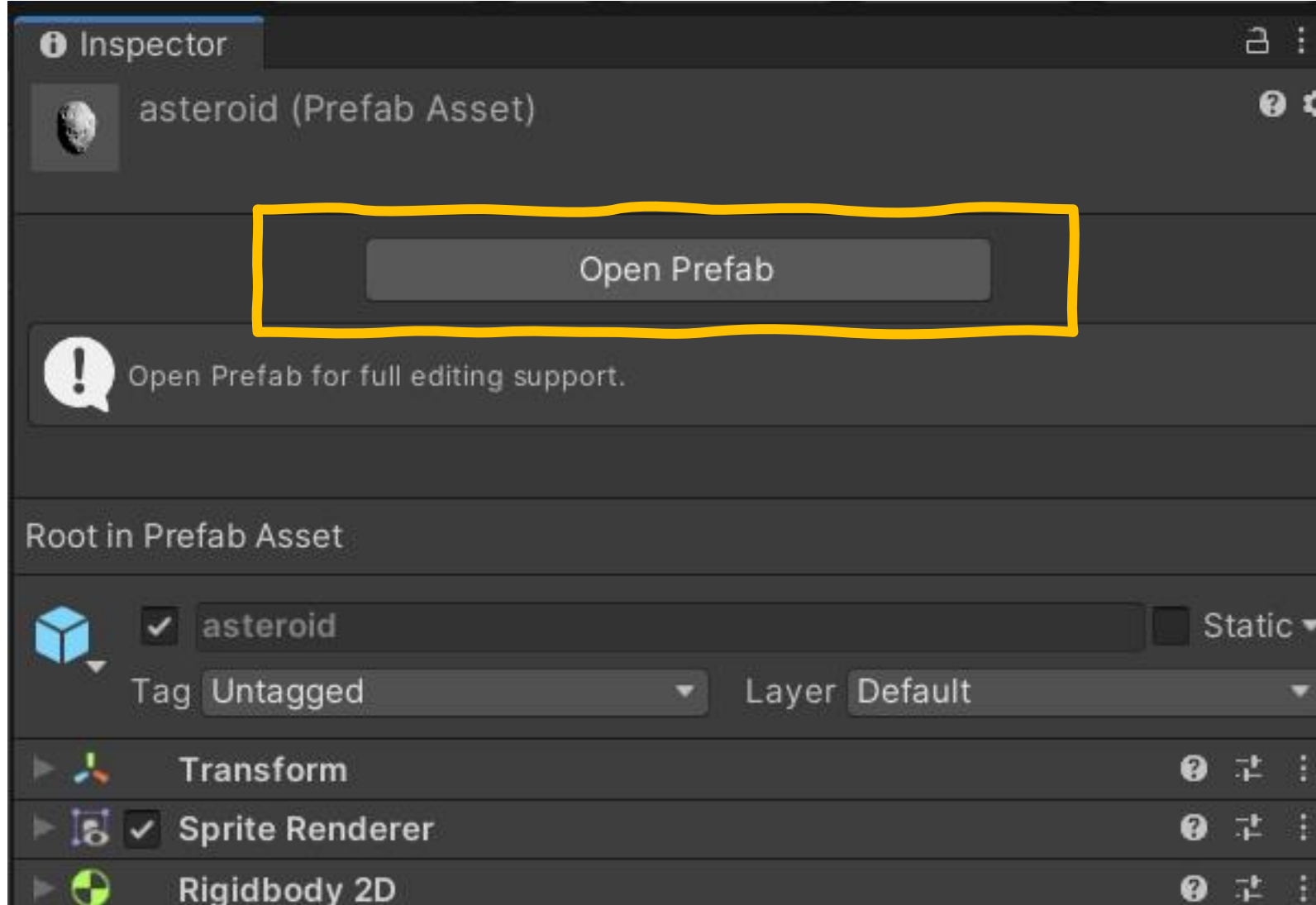


Prefab
klasörünü
oluştururuz

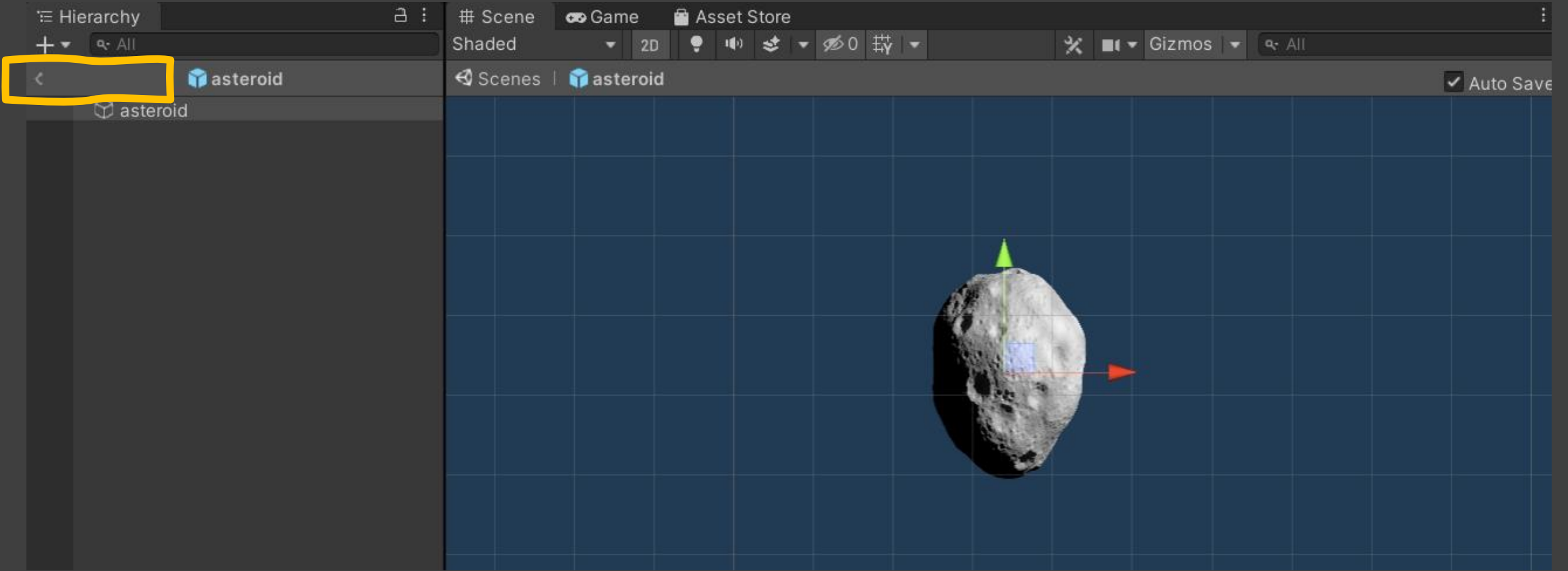


Oyun objemizi
Prefab
klasörüne
sürükleyerek
bırakırız



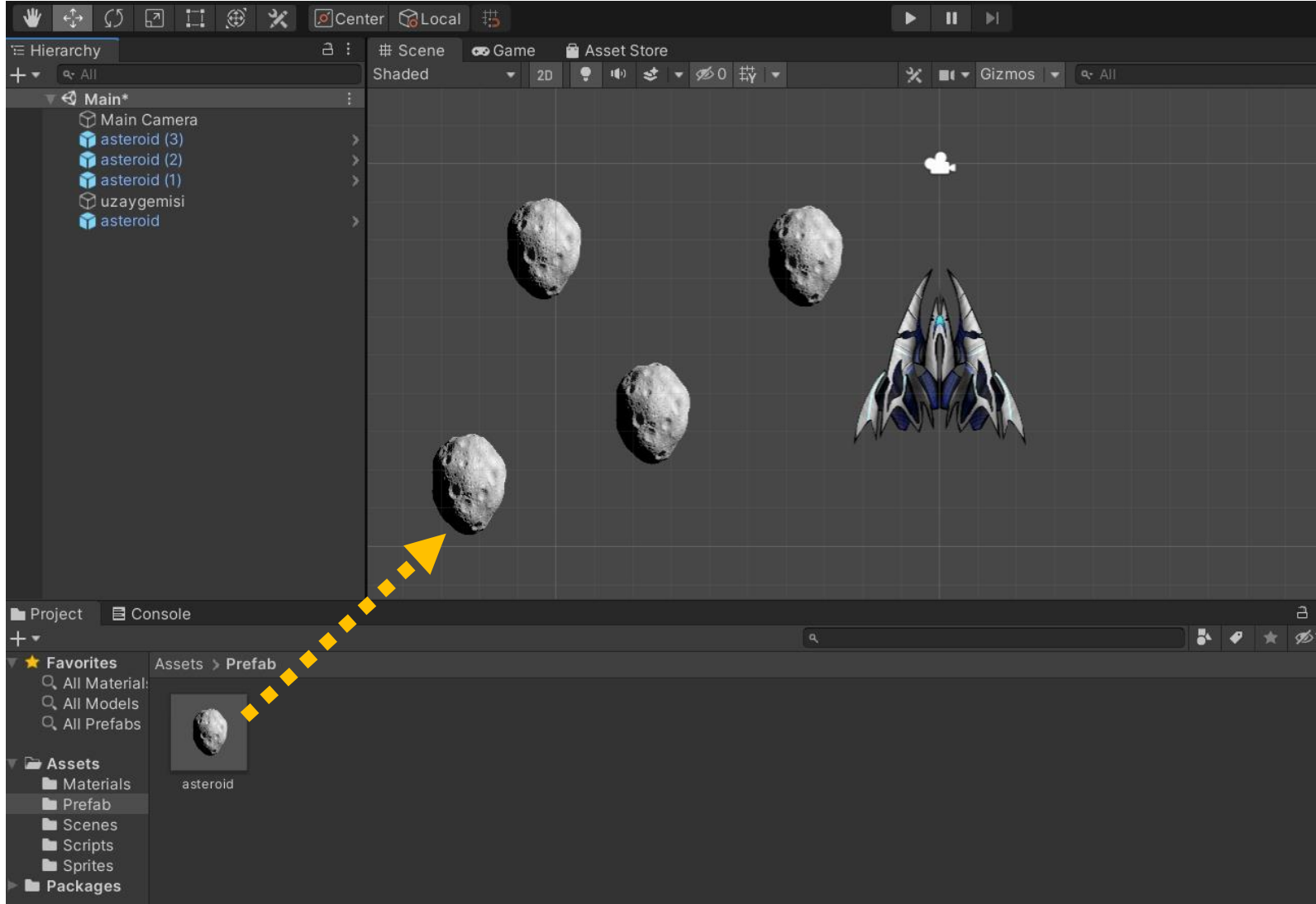


Prefab
üzerinde bir
değişiklik
yapmak için
Open Prefab
düğmesine
tıklarız



Prefab Modundan Çıkış

Prefab değişiklik modundan çıkmak için Hierarchy kısmındaki ok işaretine tıklarız



Prefab
objemizi
projemize
dahil etmek
için
nesnemizi
sürükleyerek
sahneye
bırakırız

