

Öğr. Gör. Gözde Mihran ALTINSOY

Unity



Döngüler

- For
 - İşlemin kaç kere yapılacağı biliniyorsa kullanılır.
- ForEach
 - Array, list<> gibi nesnelerle kullanılır.
- While
 - Bir durum gerçekleştiği sürece yapılacak olan işlemlerde kullanılır.
- Do / While



For Döngüsü

Tekrarlanan işlemler için kullanırız.

Bir kod bloğunda kaç kez döngü yapmak istediğimizi bildiğimizde, while döngüsü yerine for döngüsünü kullanırız.



Syntax

```
for (ifade 1; ifade 2; ifade 3)
{
    // yürütülecek kod bloğu
}
```

İfade 1 : kod bloğunun yürütülmesinden önce bir kez yürütülür. Burada döngünün başlangıç değeri tanımlanabilir.

İfade 2 : kod bloğunun yürütülmesi için koşul tanımlanır.

İfade 3 : kod bloğu yürütüldükten sonra (her seferinde) yürütülür. Burada değerin artış miktarı tanımlanabilir.

Ör: 0 ile 4
arasındaki
sayıların Console
ekranına Log'unun
yazdırmasını
sağlayalım

```
for (int i = 0; i < 5; i++)  
{  
    Debug.Log(i);  
}
```

İfade 1 : Döngü başlamadan önce bir değişken ayarlar (int i=0)

İfade 2 : Döngünün çalışması için koşulu tanımlar (i 5'ten küçük olmalıdır). Koşul doğruysa, döngü yeniden başlar, yanlıssa döngü sona erer.

İfade 3 : Döngüdeki kod bloğu her yürütüldüğünde değeri (i++) artırır.
i++ (i=i+1 veya i+=1) 1 artır anlamı taşımaktadır.

Dikkat: Sonsuz Döngü

- Unity içerisinde sonsuz döngüyü aşağıdaki gibi tanımlayabiliriz.

```
for ( ; ; )  
{  
    Debug.Log("Sonsuz Döngü");  
}
```

- Ama bu beraberinde problemlere yol açacaktır.





Dikkat : Sonsuz Döngü

- Unity içerisinde sonsuz döngü tanımladığımızda Unity sonsuz döngüye girdiği anda o frame'de kalır.
- Görev yöneticisinden kapatılmadığı sürece Unity hep açık kalacaktır.
- Bu yüzden **Unity'de sonsuz döngüden kaçınmalıyız.**

Dikkat : Mantıksal Hatalar

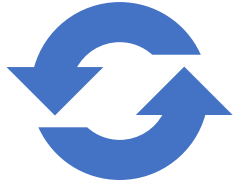
- Mantıksal hatalar nedeniyle döngümüz sonsuza kadar çalışacak bir sonsuz döngü haline gelebilir.
- Bu yüzden mantıksal hatalardan da kaçınmalıyız.

```
for (int i = 0; i >= 0; i++)  
{  
    Debug.Log("Bu döngüde mantıksal hata vardır. Döngü sonsuza kadar çalışır.");  
}
```


| Ad | Durum | CPU | Bellek | Disk | Ağ | GPU | GPU altyapısı |
|--------------------------------------|-------|------------------|------------|-----------|-----------|-----|---------------|
| Uygulamalar (6) | | | | | | | |
| > Google Chrome (140) | | %8,2 | 2.179,8 MB | 0,1 MB/sn | 0,1 Mb/sn | %0 | GPU 1 - 3D |
| > Görev Yöneticisi | | %1,2 | 35,7 MB | 0 MB/sn | 0 Mb/sn | %0 | |
| > Microsoft PowerPoint (2) | | %0,2 | 250,3 MB | 0 MB/sn | 0 Mb/sn | %0 | GPU 1 - 3D |
| > Microsoft Visual Studio 2019 (3... | | %0,1 | 342,1 MB | 0 MB/sn | 0 Mb/sn | %0 | |
| > Unity Editor (8) | | %0,1 | 224,4 MB | 0 MB/sn | 0 Mb/sn | %0 | |
| > Windows Gezgini (3) | | %1,5 | 92,9 MB | 0 MB/sn | 0 Mb/sn | %0 | |
| Daha az ayrıntı | | Görevi sonlandır | | | | | |

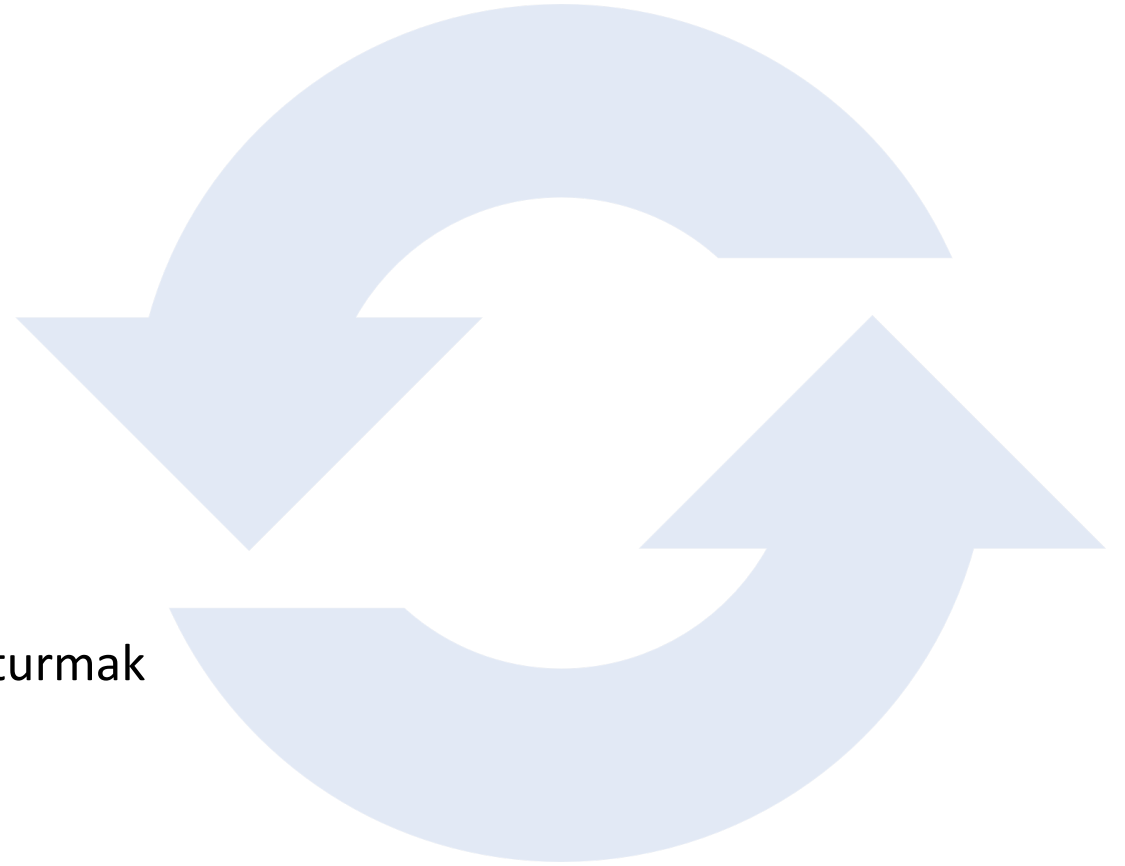
Windows: Ctrl + Alt + Del
Mac: Cmd + Alt + Esc

- Unity'de projemiz sonsuz döngü ile kilitlendiğinde görev yöneticisi ile projemizi sonlandırmak durumundayız.
- Kaydetmemiş olduğumuz tüm değişiklikler programı görev yöneticisi ile kapatmaya zorladığımız için kaybolacaktır.



Foreach Döngüsü

- Bir dizideki öğeler arasında döngü oluşturmak için kullanılır.



Syntax

```
foreach (var item in collection)
{
    // yürütülecek kod bloğu
}
```

Ör: skorlar
dizisinin içerisinde
saklanan
değerlerin
Console ekranına
Log'unun
yazdırmasını
sağlayalım

```
int[] skorlar = {10, 20, 15, 30};  
foreach (int i in skorlar)  
{  
    Debug.Log(i);  
}
```



While Döngüsü

Belirtilen bir koşul gerçekleştiği sürece (true olduğu sürece) kod bloğunu çalıştıran döngülerdir.

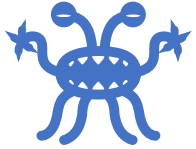
Syntax

```
while (koşul)
{
    // yürütülecek kod bloğu
}
```



Ör: sure değerinin
10'dan küçük olduğu
sürece sure değerinin
Console ekranına
Log'unun yazdırmasını
sağlayalım

```
int sure = 0;  
while (sure < 10)  
{  
    Debug.Log(sure);  
    sure++;  
}
```



While Döngüsü Örnek:

```
public class Donguler : MonoBehaviour
{
    void Start()
    {
        int saldiranDusman = 5;
        bool saldiriDevam = true;
        while (saldiriDevam)
        {
            saldiranDusman--;
            if (saldiranDusman < 1)
                saldiriDevam = false;
            Debug.Log("Saldırı altındayız. Düşman sayısı: " +
saldiranDusman);
        }
    }
}
```


Project Console

Clear Collapse Clear on Play Clear on Build Error Paused

!

[00:22:56] Saldırı altındayız. Düşman sayısı: 4
UnityEngine.Debug:Log(Object)

!

[00:22:56] Saldırı altındayız. Düşman sayısı: 3
UnityEngine.Debug:Log(Object)

!

[00:22:56] Saldırı altındayız. Düşman sayısı: 2
UnityEngine.Debug:Log(Object)

!


[00:22:56] Saldırı altındayız. Düşman sayısı: 1
UnityEngine.Debug:Log(Object)

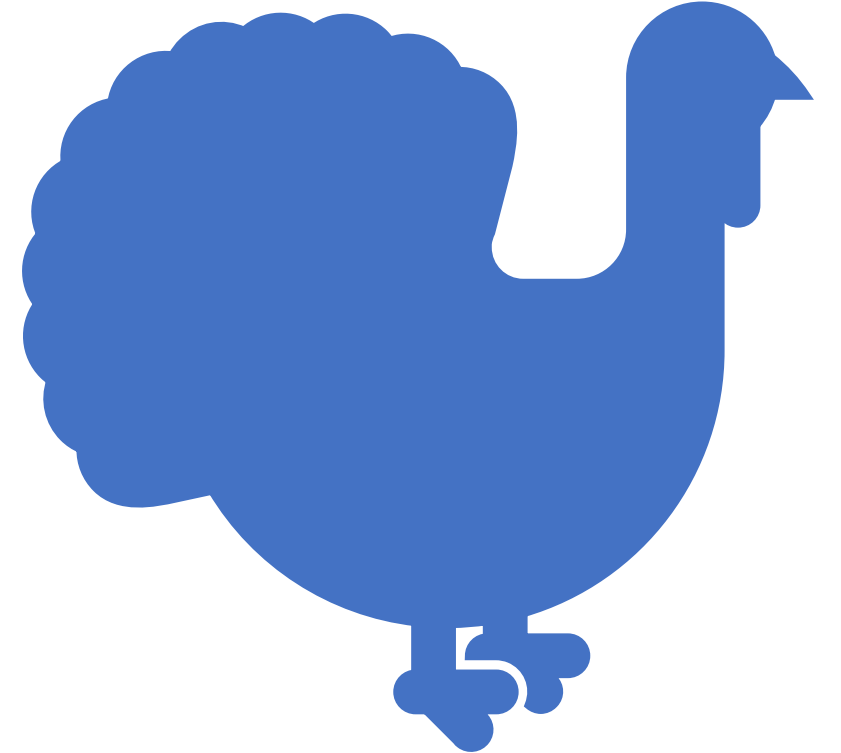
!

[00:22:56] Saldırı altındayız. Düşman sayısı: 0
UnityEngine.Debug:Log(Object)

While
Örnek

Do / While Döngüsü


| | |
|---|---|
|  | While döngüsünün bir çeşididir. |
| | Bu döngü, koşulun doğru olup olmadığını kontrol etmeden önce kod bloğunu bir kez yürütür, ardından koşul doğru olduğu sürece döngüyü tekrarlar. |
| | Kod bloğu, koşul test edilmeden önce yürütüldüğünden, koşul yanlış olsa bile her zaman en az bir kez yürütülür. |
| | |





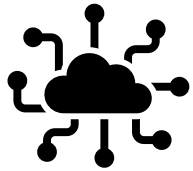
Syntax

```
do  
{  
    // yürütülecek kod bloğu  
} while (koşul);
```


A stack of several rolled-up teal towels, showing the texture of the fabric and the spiral pattern of the rolls.

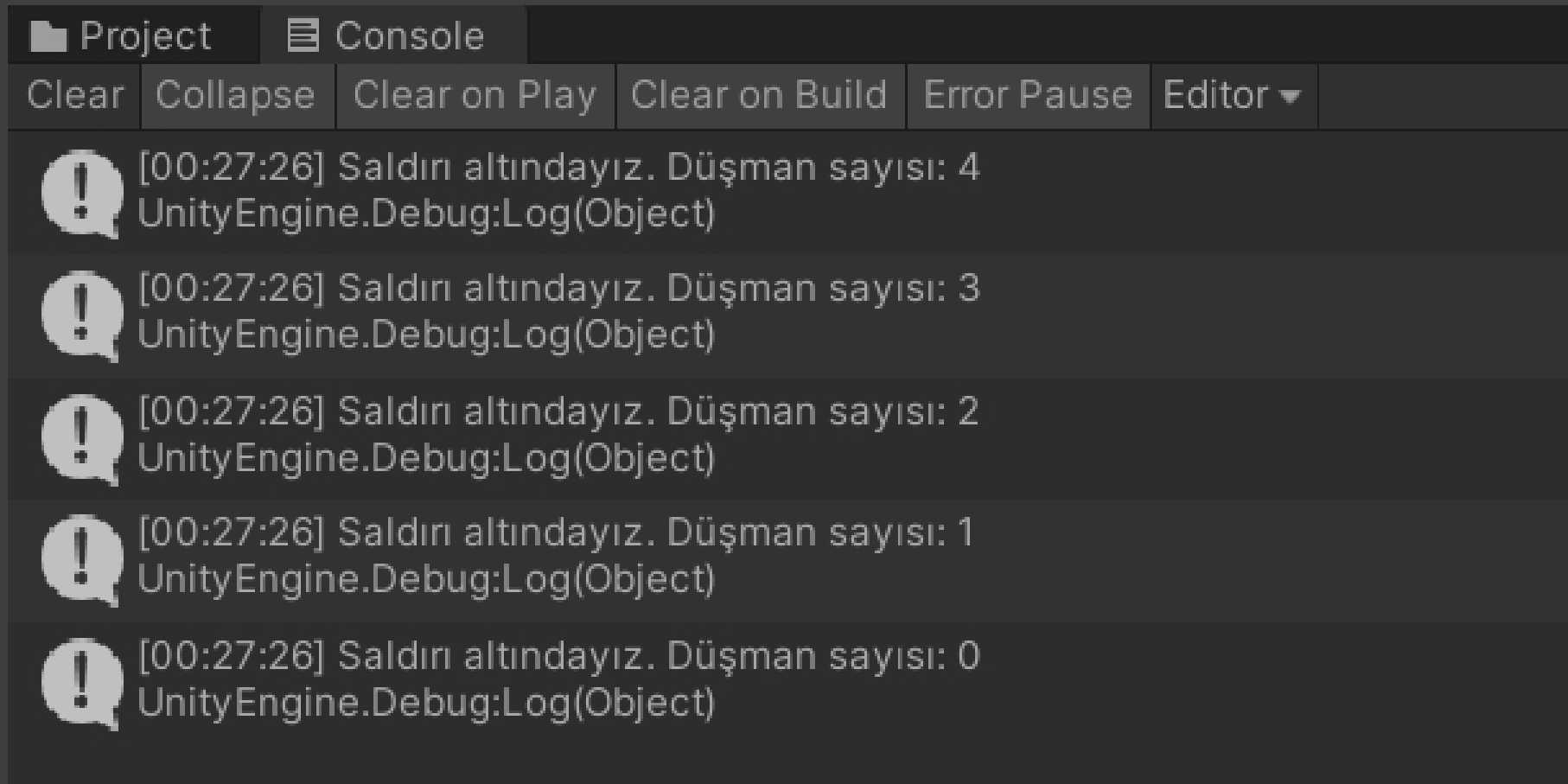
Ör: sure değerinin 10'dan küçük olduğu sürece sure değerinin Console ekranına Log'unun yazdırmasını sağlayalım

```
int sure = 0;  
do  
{  
    Debug.Log(sure);  
    sure++;  
}  
while (sure < 10);
```



Do / While Döngüsü Örnek:

```
public class Donguler : MonoBehaviour
{
    void Start()
    {
        int saldiranDusman = 5;
        bool saldiriDevam = false;
        do
        {
            saldiranDusman--;
            if (saldiranDusman < 1)
                saldiriDevam = false;
            else
                saldiriDevam = true;
            Debug.Log("Saldırı altındayız. Düşman sayısı: " +
            saldiranDusman);
        }
        while (saldiriDevam);
    }
}
```



Do / While Örnek

- Do/while döngüsünde while (saldiriDevam); koşulu ilk döngü başladığında sağlanmamış olsa bile döngü çalışır. Çünkü döngüye ilk aşamada bakmadan girer.
- Do/while döngüsü her koşulda en az 1 kere çalışmış olur.

Uzay Savaşı Oyunu Bileşenleri



Array & List



Döngü



Metot ve
Class



Bileşenler



Arayüz



Sesler



Uzay Savaşı Oyunu

Sayısal Değerler



Gemi hızı



Astreoid Sayısı



Skor

Array (Dizi)



Aynı türde çoklu verileri kaydetmek için kullanılır.



Başlangıç indis (index) değeri sıfırdır.



Projede astreoid'leri array'de saklayacağız.



Array Örnekleri

- int veri türüyle Array tanımlamak:
 - `int[] sayilar=new int[5]`
- GameObject türüyle Array tanımlamak:
 - `GameObject[] obje=new GameObject[5]`

sayilar[6]
(Array)

| Indeks | Adres | İçerik |
|--------|--------|--------|
| 0 | 992210 | 5 |
| 1 | 992214 | 6 |
| 2 | 992218 | 7 |
| 3 | 992222 | 8 |
| 4 | 992226 | 9 |
| 5 | 992230 | 10 |

```
int[] sayilar = new int[6]  
(Array)
```



Değişken[index]

Adres

İçerik

sayilar[0]

99221110

5

sayilar[1]

99221114

6

sayilar[2]

99221118

7

sayilar[3]

99221122

8

sayilar[4]

99221126

9

sayilar[5]

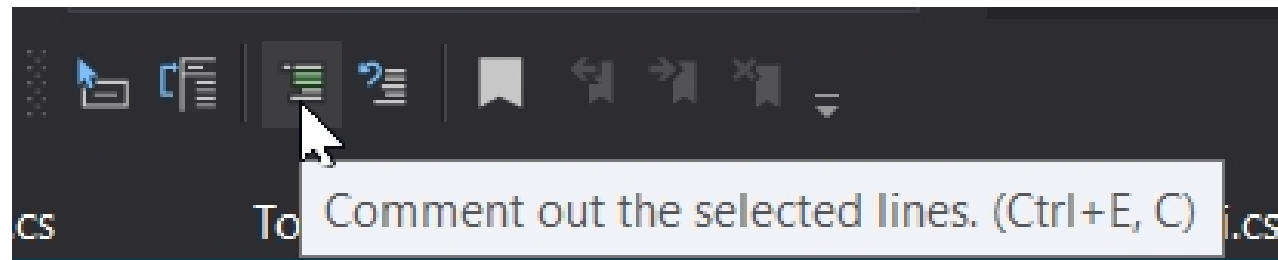
99221130

10

- int değişken türü hafızada 4 byte yer kaplar.
- Bu yüzden ardışık olan iki dizi elemanının adresleri arasındaki fark 4'tür.
- Bir dizinin (array) adres bilgisine ulaşırken dizinin ilk elemanının adres bilgisi referans alınarak hesaplama yapılır.
- Ör: sayilar[2] dizi elemanının adres bilgisine ulaşmak için aşağıdaki formül uygulanır.
 - $\text{sayilar}[2] \text{ index'i} * 4 + \text{sayilar}[0] \text{'ın adresi}$
 - $2*4+99221110 = 99221118$
- Yani adres hesaplamada index değeri çarpan durumunda olduğu için sıfır değerinden başlar.

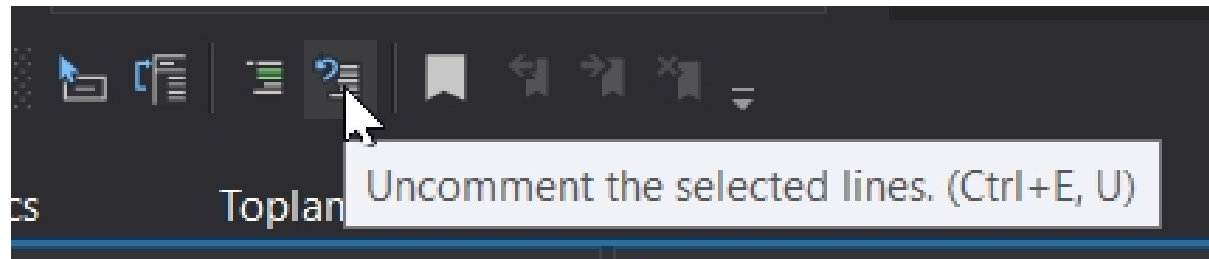
Satırları Yorum Satırı Haline Getirmek

- 1.Yöntem:
 - Yorum satırı yapılacak satırları seçiniz.
 - Sırasıyla CTRL + E → CTRL + C tuşlarına basınız.
- 2.Yöntem:



Satırlardaki Yorum Satırını Kaldırmak

- 1.Yöntem:
 - Yorum satırı yapılacak satırları seçiniz.
 - Sırasıyla CTRL + E → CTRL + U tuşlarına basınız.
- 2.Yöntem:




Rastgele Bir
Kuvvet ile
Oyun Objesini
Hareket
Ettirelim

```
public class HareketKontrol :  
MonoBehaviour  
{  
    private void Start()  
    {  
        //Oyun objesini rastgele bir  
        kuvvetle hareket ettiriyoruz  
  
        GetComponent<Rigidbody2D>().AddForce(  
            new Vector2(Random.Range(-5, 5),  
                Random.Range(-5, 5)),  
            ForceMode2D.Impulse);  
    }  
}
```

4 Elemanlı GameObject Türünde Array Tanımlamak

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

    GameObject[] astreoids = new GameObject[4];
}
```

Mouse ile Sol Click Yapılan Noktaya 4 Tane Astreoid'in Oluşmasını Sağlamak

```
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);

        for (int i = 0; i < 4; i++)
        {
            astreoids[i] = Instantiate(astreoidPrefab,
position, Quaternion.identity);
        }
    }
}
```

Mouse Sol Click ile Oyunda
En Son Oluşturmuş
Olduğumuz Astreoid'leri
Sağ Click ile Yok Edelim

```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        Debug.Log(Input.mousePosition);
        Vector3 position = Input.mousePosition;
        position.z = -Camera.main.transform.position.z;
        position = Camera.main.ScreenToWorldPoint(position);

        for (int i = 0; i < 4; i++)
        {
            astreoids[i] = Instantiate(astreoidPrefab, position, Quaternion.identity);
        }
    }
    if (Input.GetMouseButtonDown(1))
    {
        Debug.Log(astreoids.Length);
        for (int i = 0; i < astreoids.Length; i++)
        {
            Destroy(astreoids[i]);
        }
    }
}
```

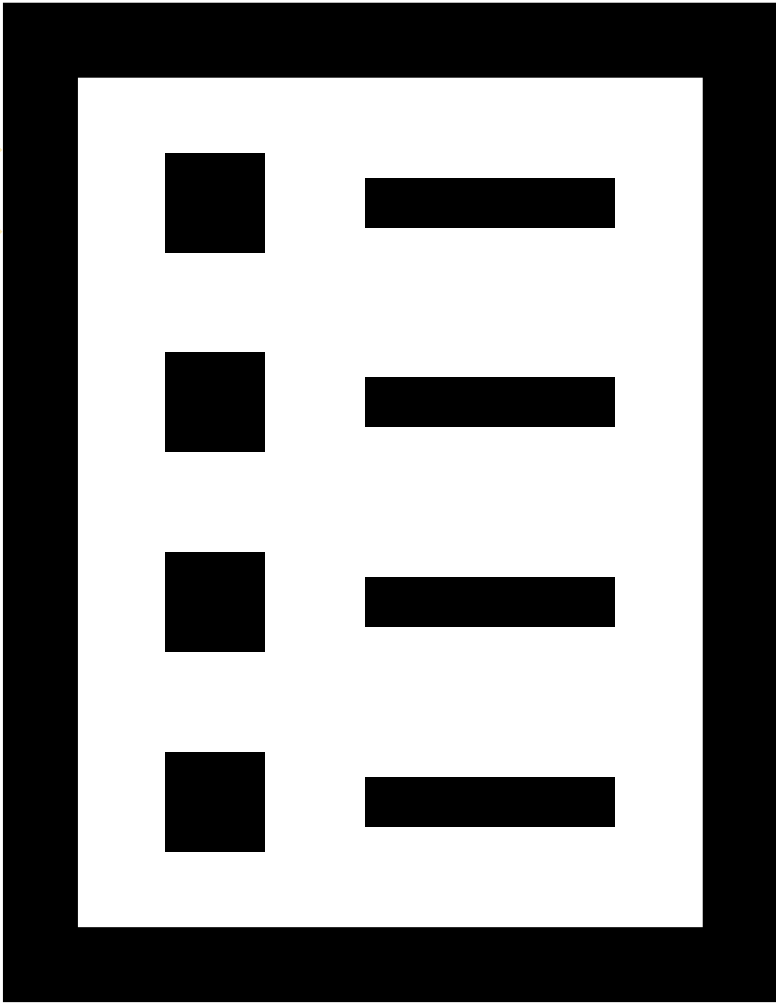


List (Liste)

- List içinde tutulacak olan değerlerin eleman sayısını önceden belirtmek zorunda değiliz.
- İçerisinde saklanacak olan verilerin veri türü aynı olmak zorundadır.
- `using System.Collections.Generic;` alan adının içerisinde yer alırlar.
- Listelerin kapasitesi (içerisinde tuttukları değer sayısı) ihtiyaç olduğunda arttırılabilir.

List Örnekleri

- `List<int> sayiliste=new List<int>()`



InputKontrol.cs UzakGemisiKontrol.cs HareketKontrol.cs Spawner.cs

Assembly-CSharp

InputKontrol

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UIElements;
5
```

using System.Collections.Generic;

Henüz bu kütüphaneyi kullanmamış olduğumuz için rengi bu şekilde görünmektedir.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UIElements;
5
6 public class InputKontrol : MonoBehaviour
7 {
8     [SerializeField]
9     GameObject astreoidPrefab;
10
11     List<GameObject> asteroidList = new List<GameObject>();
```

List objesi oluşturulduktan sonra Generic kütüphanesi aktif olur ve bu yüzden kütüphanenin rengi değişir.

List Oyun Objesi Tanımlamak

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UIElements;

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

    List<GameObject> asteroidList = new
    List<GameObject>();
}
```


Her Sol Click'te 10 tane List Oyun Objesini Projeye Dahil Etmek

```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        Debug.Log(Input.mousePosition);
        Vector3 position = Input.mousePosition;
        position.z = -Camera.main.transform.position.z;
        position = Camera.main.ScreenToWorldPoint(position);
        for (int i = 0; i < 10; i++) {
            asteroidList.Add(Instantiate(astreoidPrefab,
position, Quaternion.identity));
        }
    }
}
```

Sağ Click ile Tüm Objeleri Silmek

1. Yöntem : For

```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        Debug.Log(Input.mousePosition);
        Vector3 position = Input.mousePosition;
        position.z = -Camera.main.transform.position.z;
        position = Camera.main.ScreenToWorldPoint(position);
        for (int i = 0; i < 10; i++) {
            asteroidList.Add(Instantiate(astreoidPrefab, position,
Quaternion.identity));
        }
    }
    if (Input.GetMouseButtonDown(1))
        for (int i = 0; i < asteroidList.Count ; i++)
            Destroy(asteroidList[i]);
        asteroidList.Clear();
}
```

Sağ Click ile Tüm Objeleri Silmek

2. Yöntem : For Each

```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        Debug.Log(Input.mousePosition);
        Vector3 position = Input.mousePosition;
        position.z = -Camera.main.transform.position.z;
        position = Camera.main.ScreenToWorldPoint(position);
        for (int i = 0; i < 10; i++) {
            asteroidList.Add(Instantiate(astreoidPrefab, position,
Quaternion.identity));
        }
    }
    if (Input.GetMouseButtonDown(1))
    {
        Debug.Log(asteroidList.Count);
        foreach (GameObject astreoid in asteroidList)
            Destroy(astreoid);
        asteroidList.Clear();
    }
}
```

List Metotları



```
Debug.Log(asteroidList.Count);
```



Count ile astreoidList listesinin eleman sayısı ekrana yazdırılır.



```
asteroidList.Clear();
```



Clear() metodu ile listenin (astreoidList) içerisinde tutulan elemanların tümü silinir.



Eğer Clear() metodu ile listenin içerisinde saklanan elemanları silmemiş olsaydık; her çalıştırdığımızda eleman sayısı 10 artacağı için ekrandan silinen objeleri burada tutmaya devam edecektik.

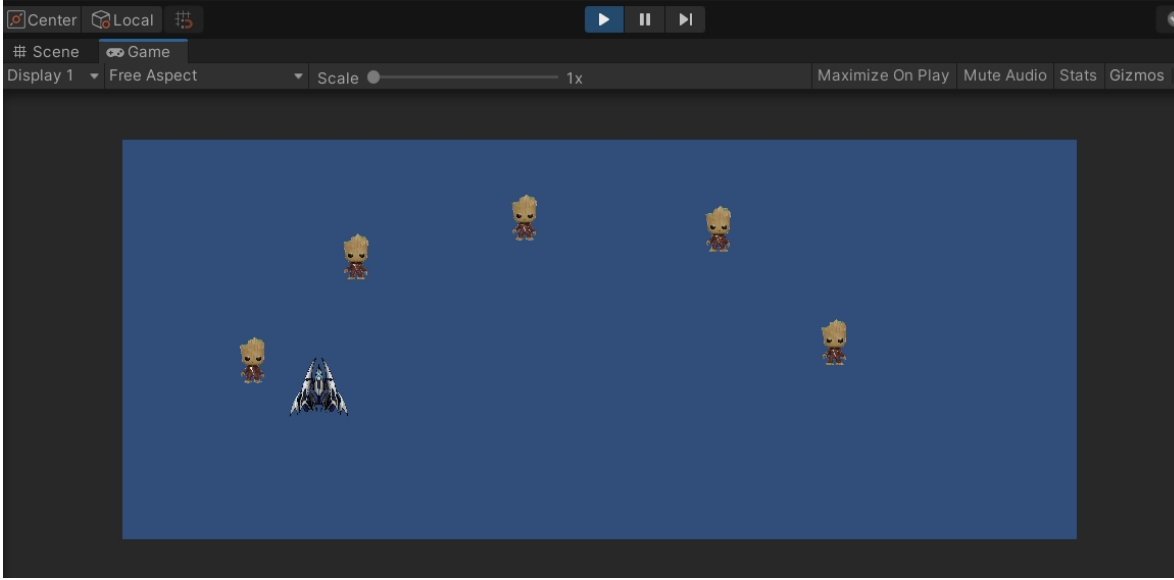


Listenin içinde tutulan elemanların Clear() metodu ile silinmemesi sonucu liste eleman sayısı artacaktır. Bu durum oyunumuzda teknik aksaklıklara ve sistemin gereğinden fazla çalışmasına yol açabilir.



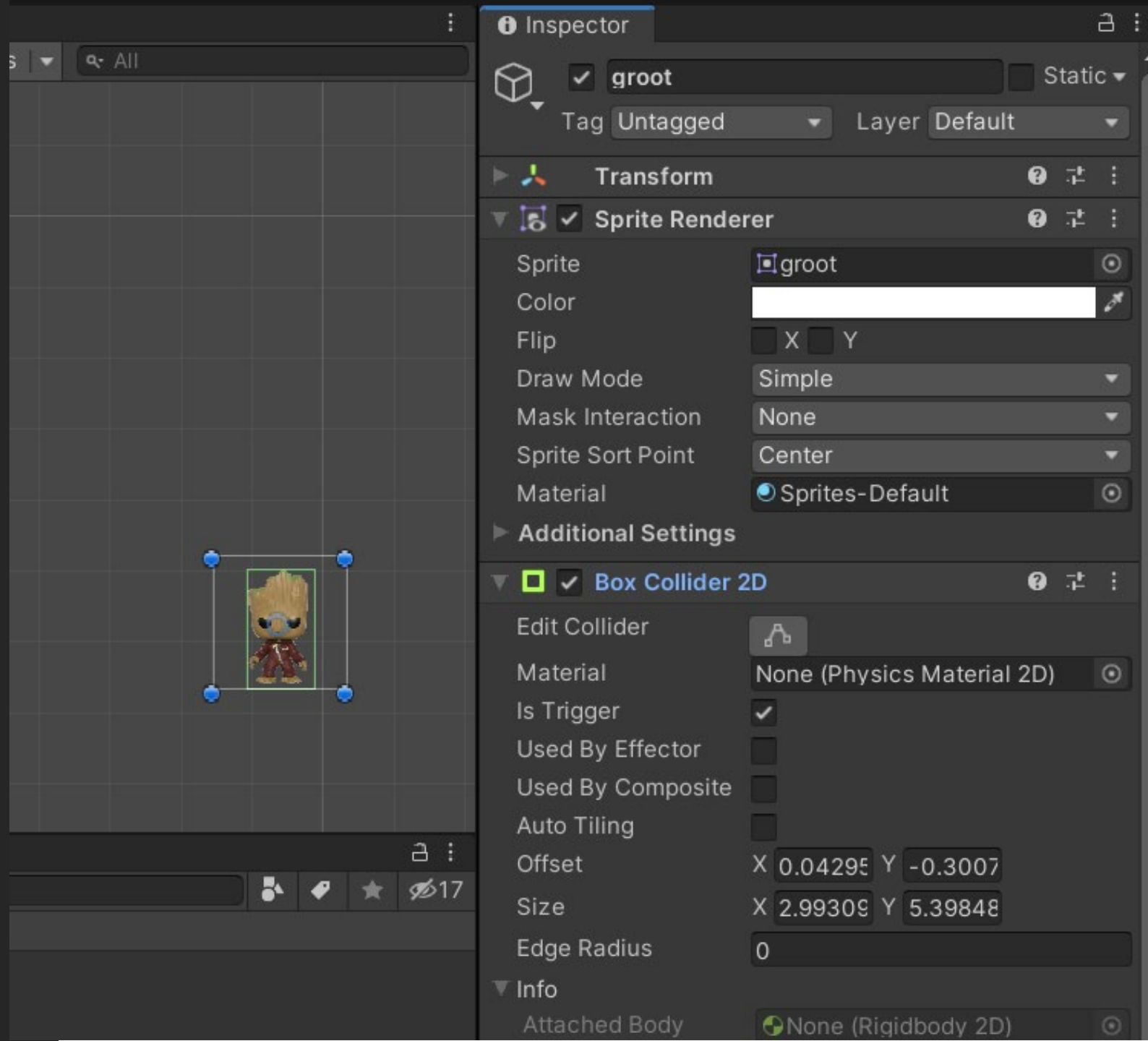
Oyun Objelerimizin
Birbiri ile Etkileşimini
Sağlamak

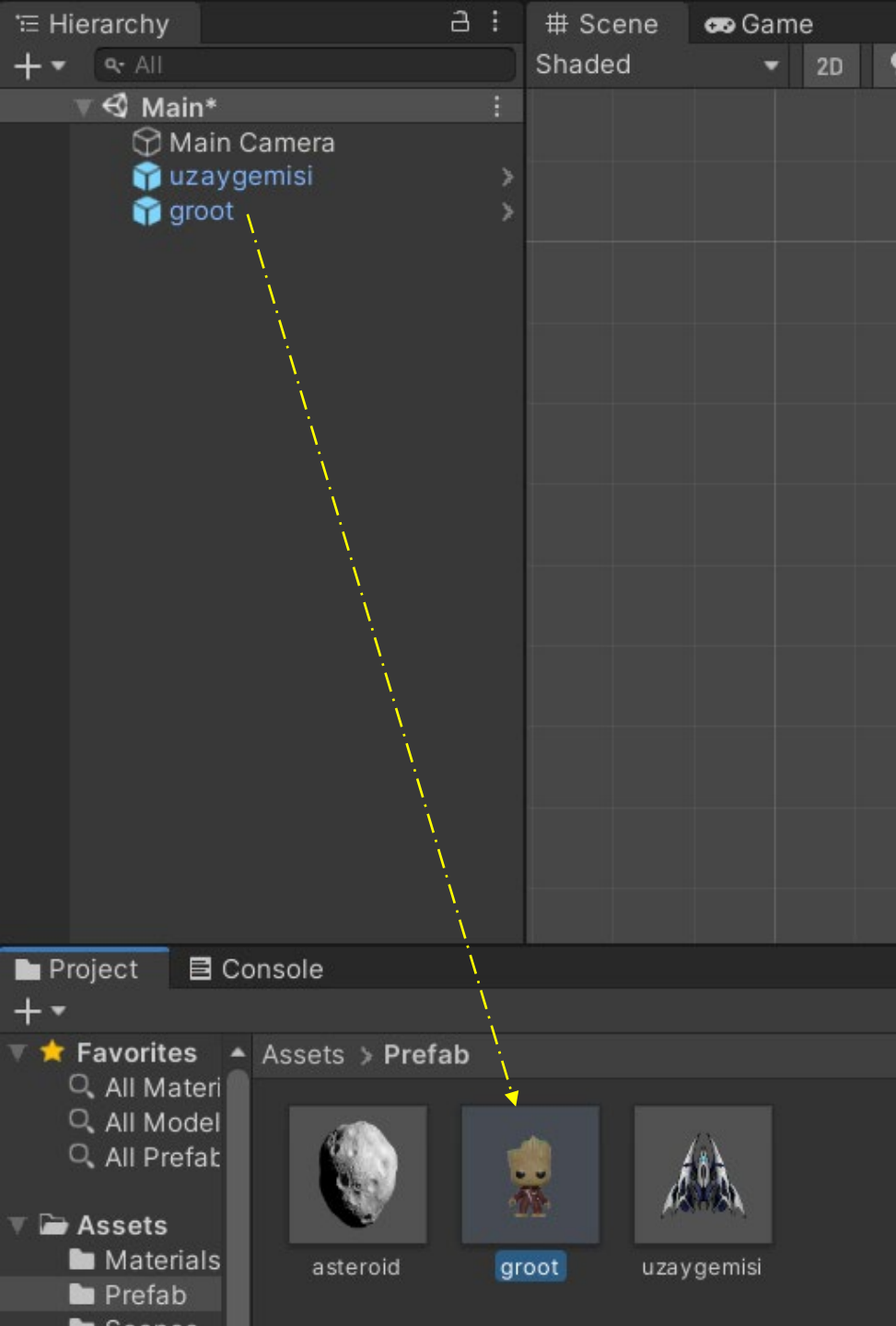
Uzay Gemisi Oyunu



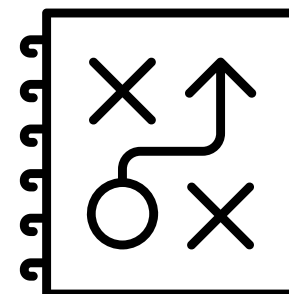
- Oyun ekranına mouse sağ tuş ile tıkladığımız (sağ click) konumlara Groot objesinin gelmesini sağlayalım.
- Uzay gemisine sol tuş ile tıkladığımızda (sol click) Groot nesnelerini ekrana eklediğimiz sırayla toplamasını sağlayalım.

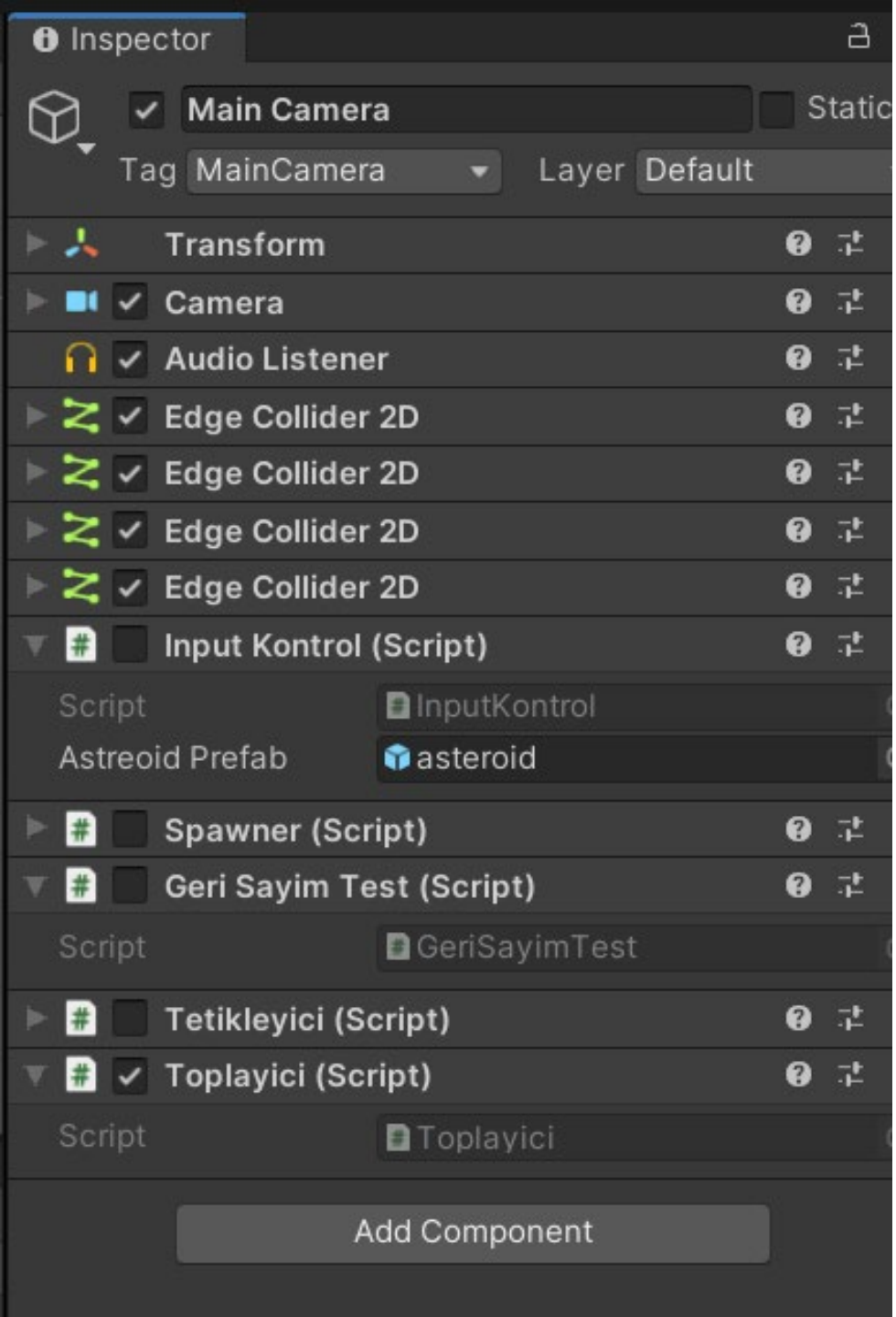
Oyunumuza bir
Sprite objesi
ekleyelim.
Sprite objemize
Box Collider 2D
ekleyerek
«Is Trigger»
özelliğini aktif
hale getirelim



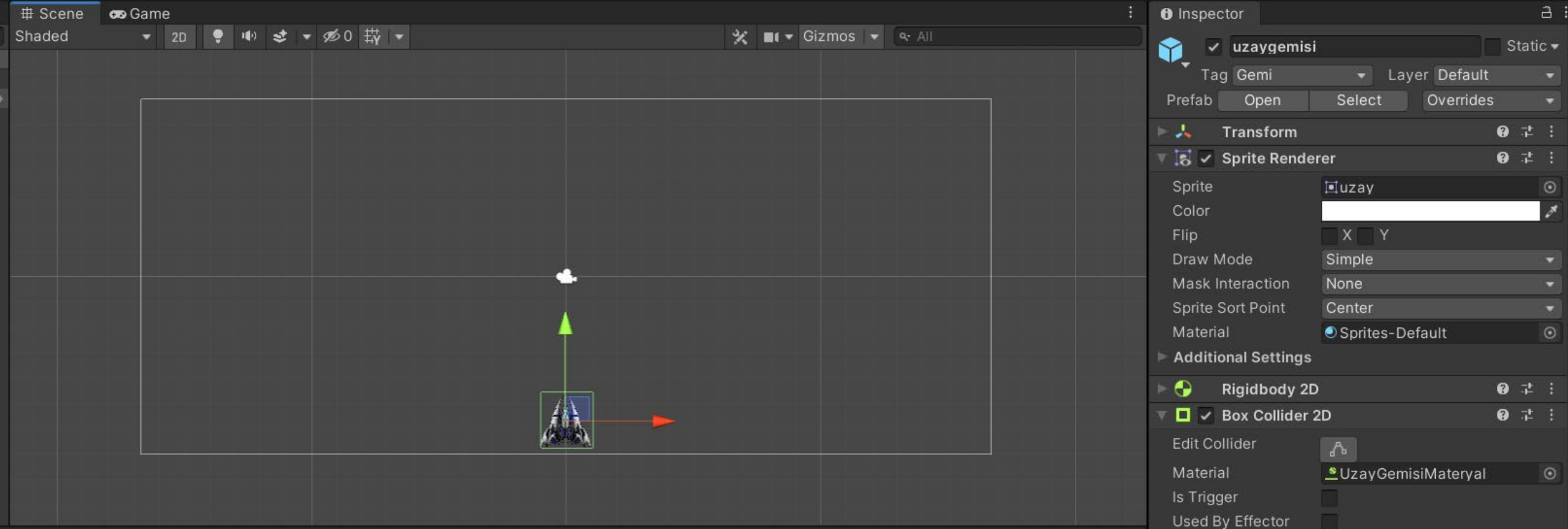


Oyun objemizi Prefab haline getirip, Hierarchy'den objemizi kaldıralım.





Toplayici.cs Script'i
oluşturulup,
Main Camera'ya
Component olarak
eklenir.



Sahneye Uzay Gemisi Prefab objesini ekleyelim

Toplayici.cs

```
public class Toplayici : MonoBehaviour
{

    [SerializeField]
    GameObject grootPrefab;

    List<GameObject> groot = new List<GameObject>();
}
```



Select GameObject

Assets

Scene

17

Explosion

groot

Edge Collider 2D

Input Kontrol (Script)

Script: InputKontrol

Astreoid Prefab: asteroid

Spawner (Script)

Geri Sayim Test (Script)

Script: GeriSayimTest

Tetikleyici (Script)

Toplayici (Script)

Script: Toplayici

Groot Prefab: groot

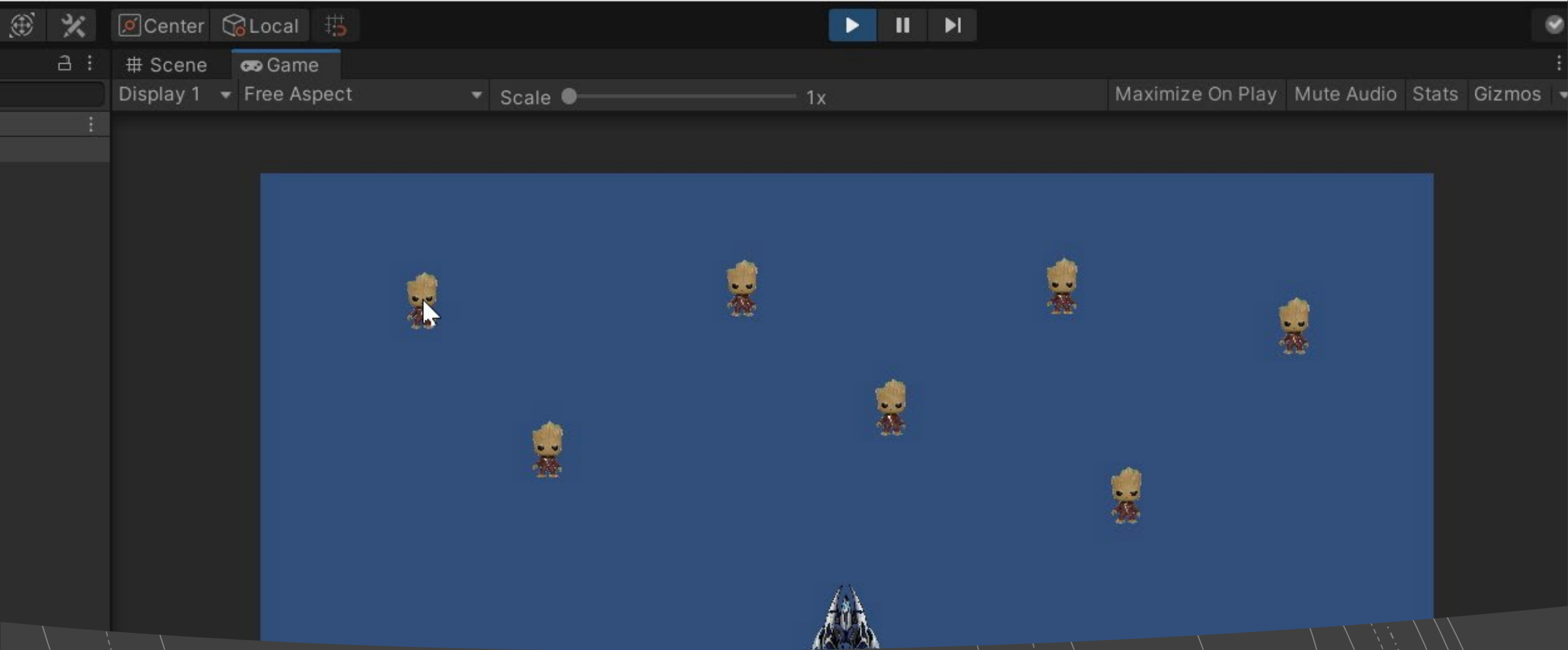
Add Component

Mouse Sol Click
Yapınca Oyun
Ekranındaki
Tıklanılan Pozisyona
Prefab Objemizin
Gelmesini
Sağlayalım

```
void Update()
{
    if (Input.GetMouseButtonDown(1))
    {
        Vector3 position = Input.mousePosition;
        position.z = -
Camera.main.transform.position.z;
        Vector3 oyunPosition =
Camera.main.ScreenToWorldPoint(position);
        groot.Add(Instantiate(grootPrefab,
oyunPosition, Quaternion.identity));
    }
}
```

Ekрана Eklenen
Groot Objesini
Eklendiği Sırayla
Yok Etmemizi
Sağlamak için
Hedefi
Belirleyelim

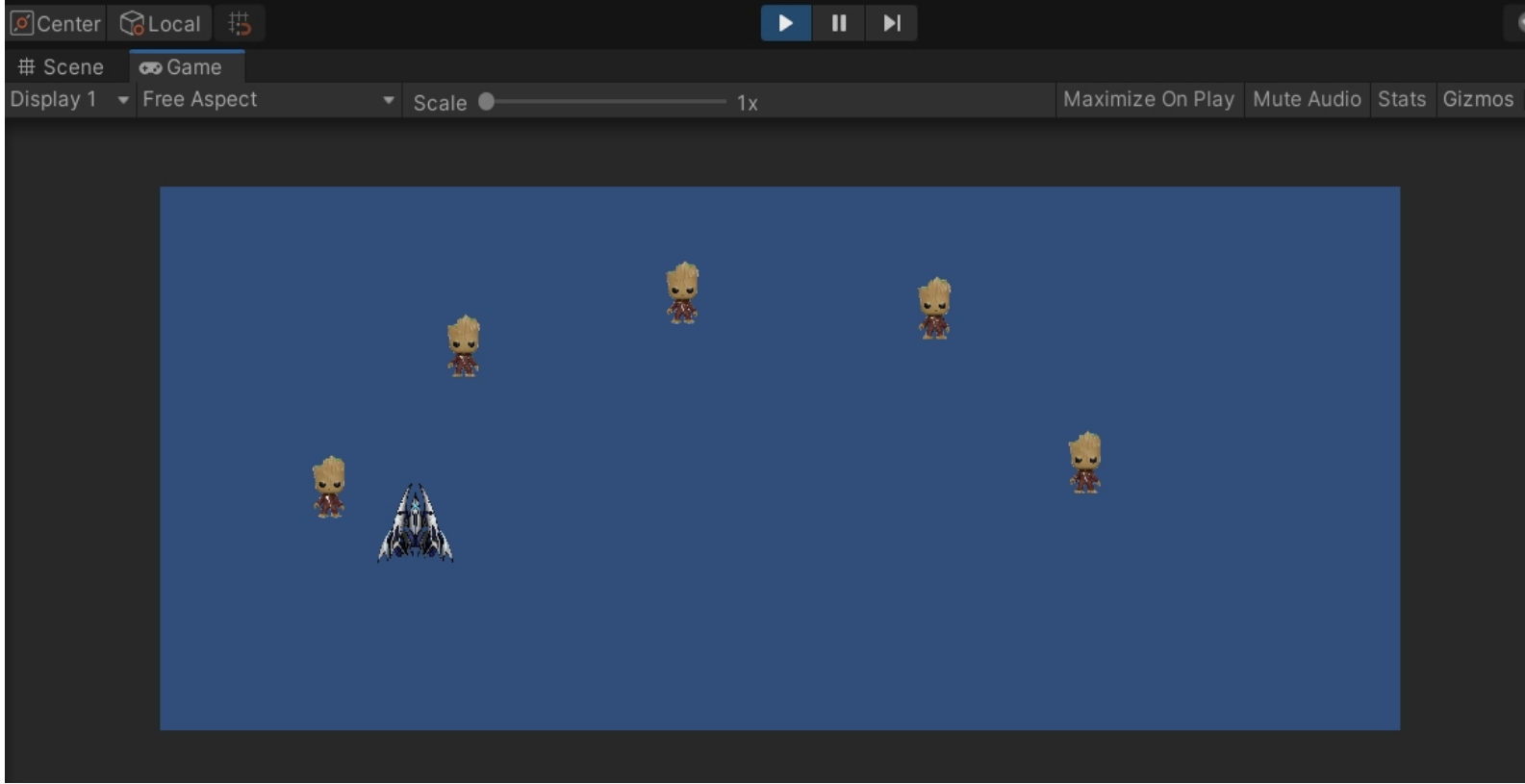
```
/// <summary>  
/// Hedefteki groot nesnesini söyler  
/// </summary>  
public GameObject HedefGroot  
{  
    get  
    {  
        if (groot.Count > 0)  
            return groot[0]; //İlk groot'u döndür.  
        else  
            return null; //Boş cevap döndür  
    }  
}
```



Oyunu alıřtırıp, oyun ekranındaki her saė click iřleminde oyun ekranımıza Groot objesi eklenir.

Hedefteki Groot Objesini Yok Edelim

```
/// <summary>
/// Parametre olarak gönderilen Groot'u yok
eder
/// </summary>
/// <param name="yokEdilecekGroot"></param>
public void GrootYokEt(GameObject
yokEdilecekGroot)
{
    groot.Remove(yokEdilecekGroot);
    //Nesne listeden silinir
    Destroy(yokEdilecekGroot);
    //Nesne oyun ekranından yok edilir
}
```

Sağ Click ile oyun ekranına tıkladığımızda Groot objesi oluşur. Uzun gemisine Sol Click yaptığımızda uzay gemisi hareket eder ve Groot'ları eklediğimiz sırayla yok eder.



UzayGemisiKontrol.cs

```
public class UzayGemisiKontrol :  
MonoBehaviour  
{  
    const float hareketgucu = 5;  
    GameObject hedef;  
    Rigidbody2D myRigidBody2d;  
    Toplayici toplayici;  
}
```



UzayGemisiKontrol.cs

```
void Start()  
{  
    myRigidBody2d =  
GetComponent<Rigidbody2D>();  
    toplayici =  
Camera.main.GetComponent<Toplayici>();  
}
```

UzayGemisiKontrol.cs

GitVeTopla 1.Yöntem: AddForce()

```
void GitVeTopla()
{
    hedef = toplayici.HedefGroot;
    if (hedef != null)
    {
        Vector2 gidilecekYer = new Vector2(hedef.transform.position.x -
transform.position.x, hedef.transform.position.y - transform.position.y);
        gidilecekYer.Normalize();
        myRigidBody2d.AddForce(gidilecekYer * hareketgucu, ForceMode2D.Impulse);
    }
}
```

UzayGemisiKontrol.cs

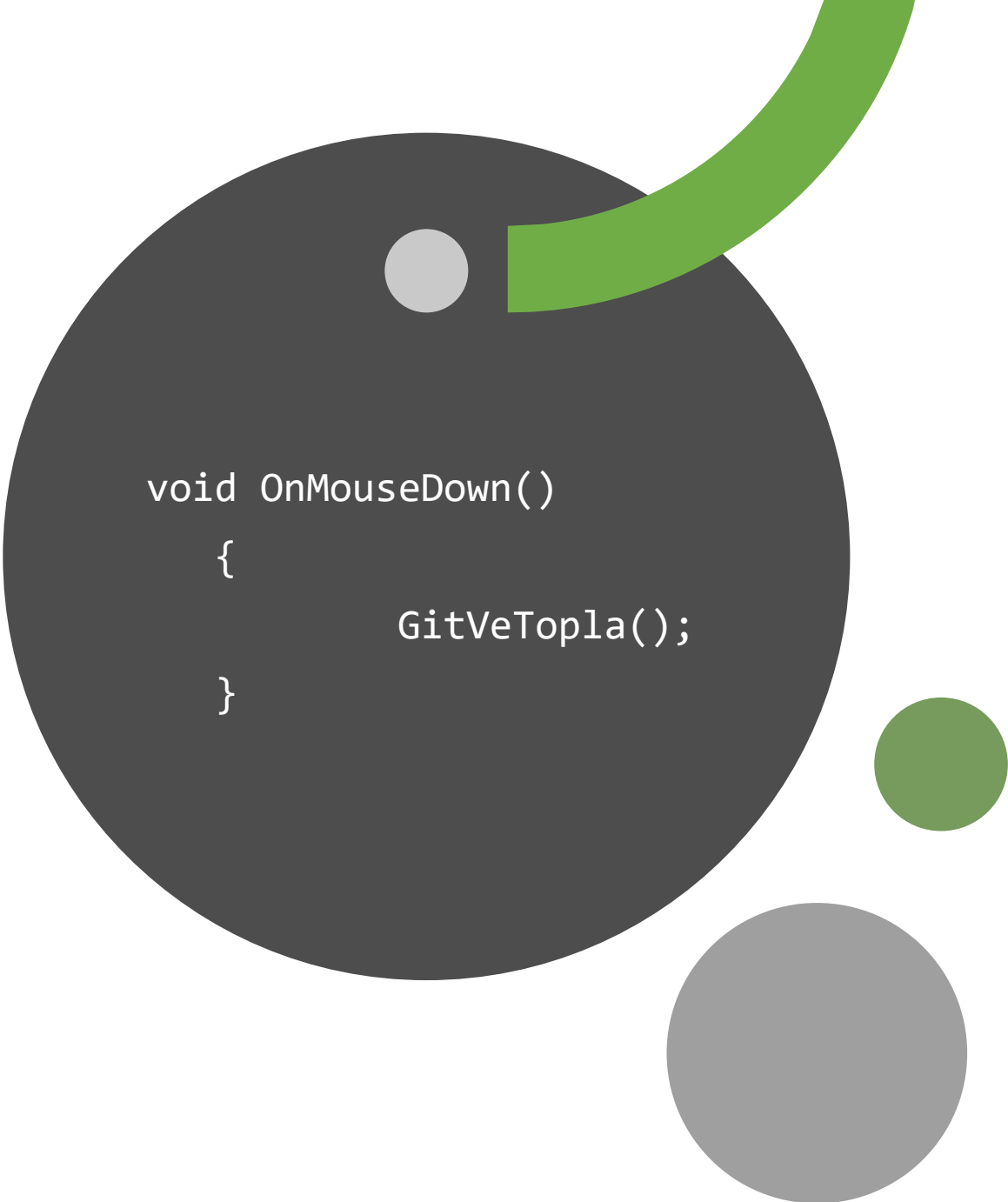
GitVeTopla 2.Yöntem: velocity

```
void GitVeTopla()
{
    hedef = toplayici.HedefGroot;
    if (hedef != null)
    {
        Vector2 gidilecekYer = new Vector2(hedef.transform.position.x -
transform.position.x, hedef.transform.position.y - transform.position.y);
        gidilecekYer.Normalize();
        myRigidBody2d.velocity = gidilecekYer * hareketgucu;
    }
}
```

UzayGemisiKontrol.cs

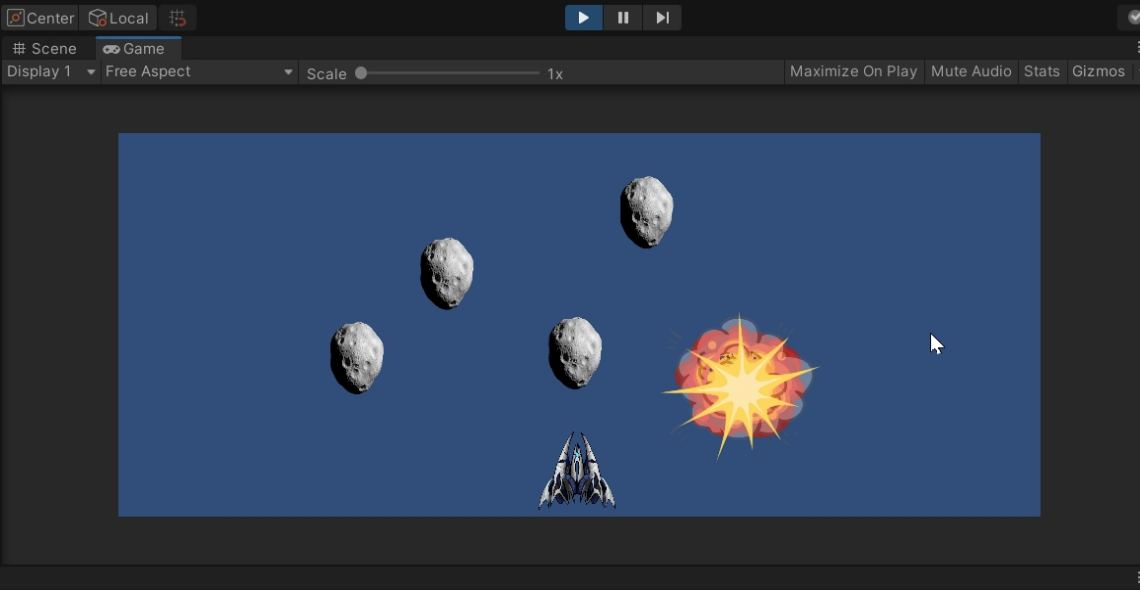
```
void OnTriggerStay2D(Collider2D other)
{
    if (other.gameObject == hedef)
    {
        toplayici.GrootYokEt(hedef);
        myRigidBody2d.velocity = Vector2.zero;
        //velocity = hiz
        GitVeTopla();
    }
}
```

UzayGemisiKontrol.cs



```
void OnMouseDown()  
{  
    GitVeTopla();  
}
```

Uzay Gemisi Oyunu



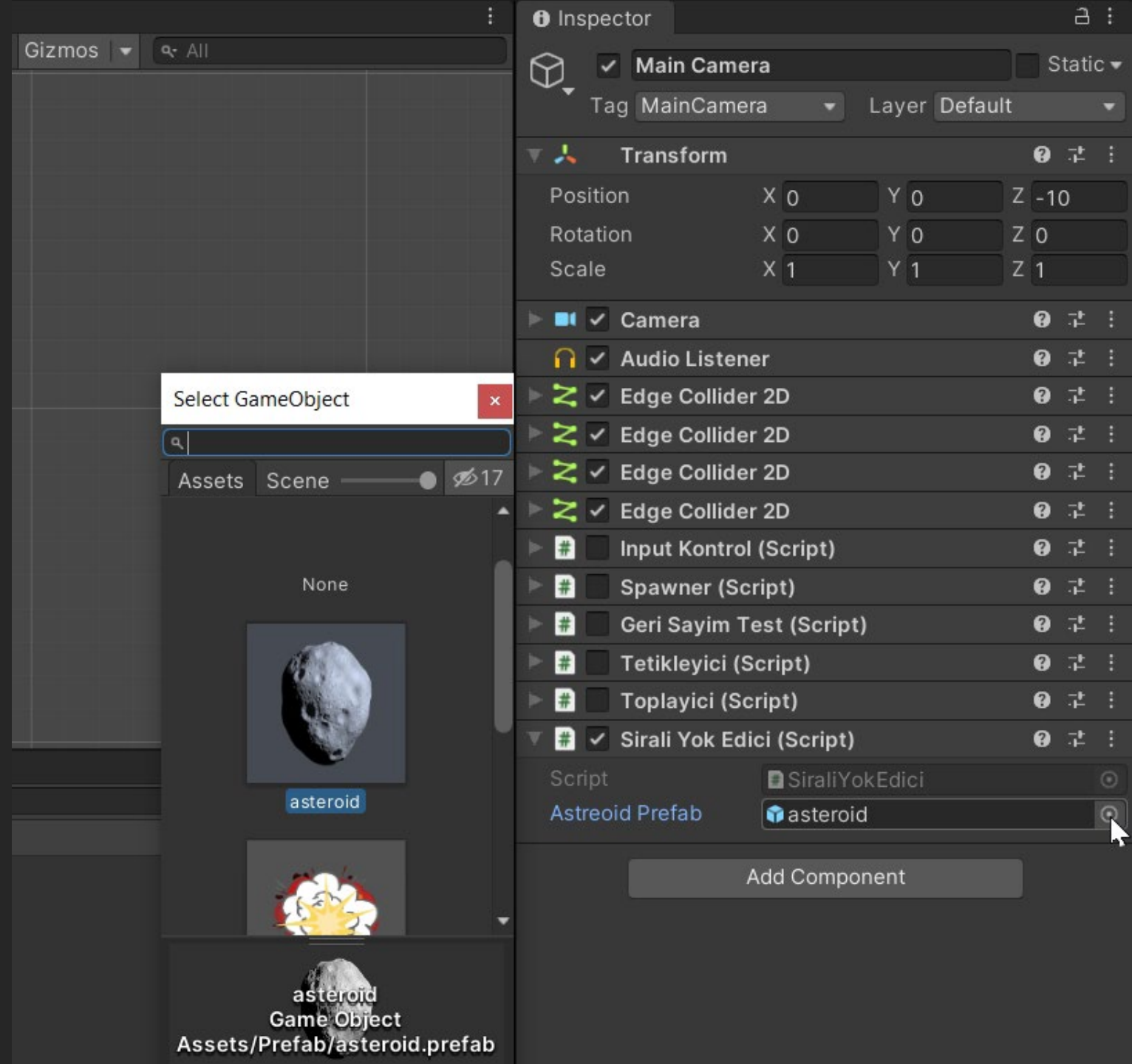
- Oyun ekranına mouse sol tuş ile tıkladığımız (sol click) konumlara Groot objesinin gelmesini sağlayalım.
- Ekranda herhangi bir yere sağ tuş ile tıkladığımızda (sağ click) her 2 saniyede bir uzay gemisine en yakın astreoid objesinden başlamak üzere astreoid objelerimizin oyundan yok edilmesini sağlayalım.
- 1 sn'de bir objenin yok olması, 1 sn'de patlama efekti ile toplam 2 sn'de objemiz oyundan yok edilecektir.
- Bu yok etme işlemi devam ederken sol tuş ile uzay gemisine yakın olacak şekilde yeni astreoid objesi eklediğimizde bir sonraki adımda yine en yakın astreoid objesini yok etmesini sağlayalım.



SiraliYokEdici.cs

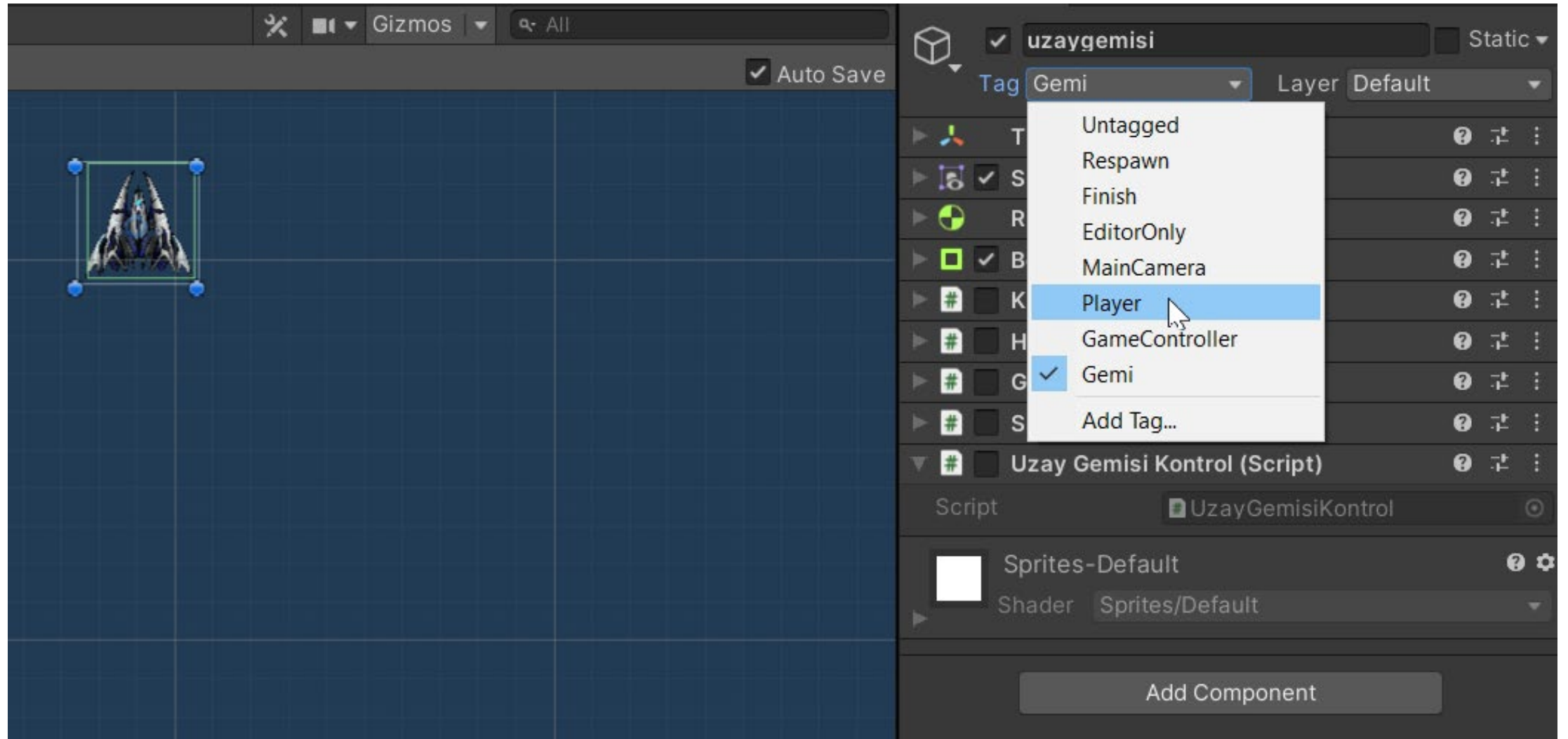
```
public class SiraliYokEdici : MonoBehaviour
{
    [SerializeField]
    GameObject astreoidPrefab;
}
```

Main Camera
Objesine
SiralıYokEdici
Component'ini
Ekleyelim be
Astreoid Prefab için
astreoid Prefab
objesini seçelim

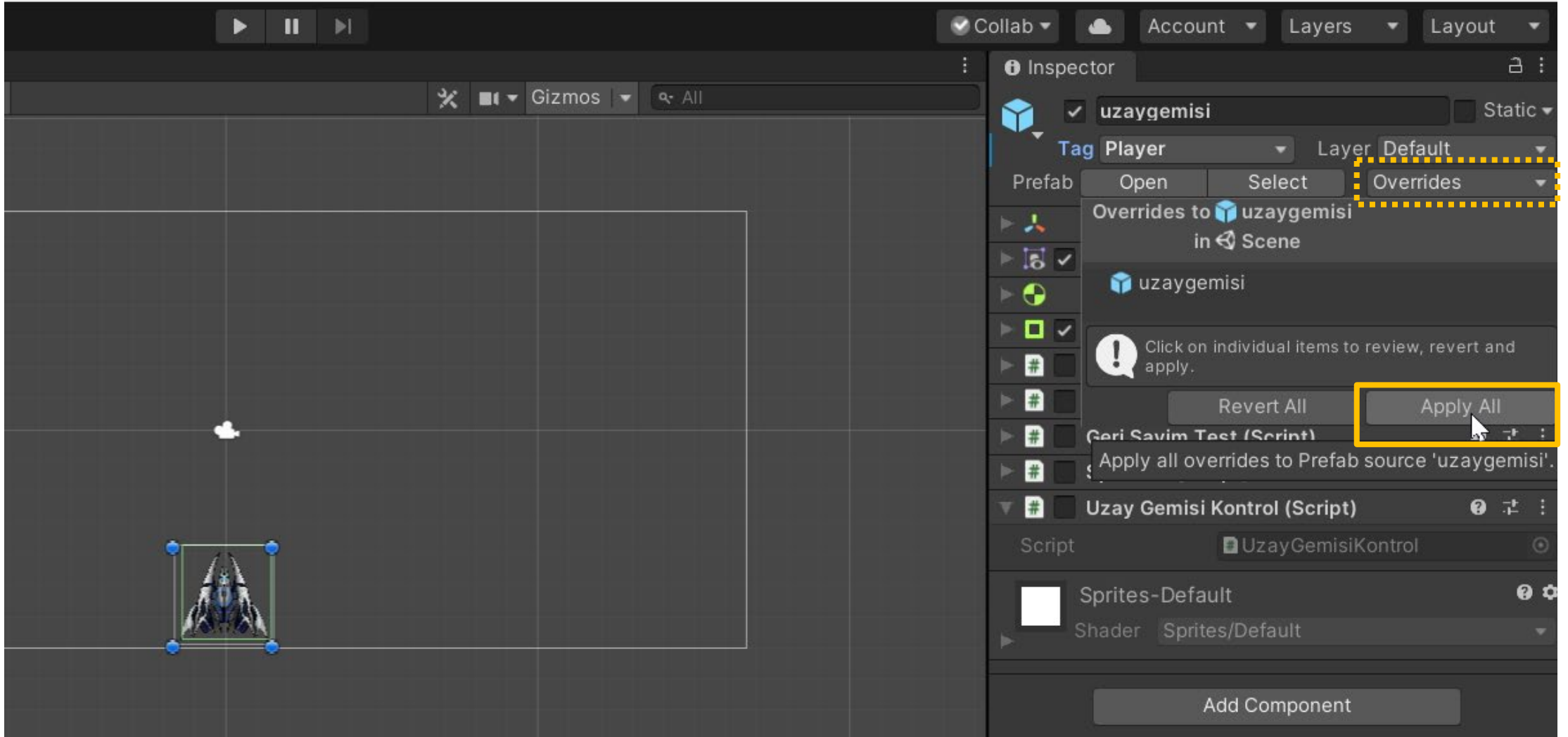


Prefab Oyun Objesinin Tag Değerini Değiştirmek



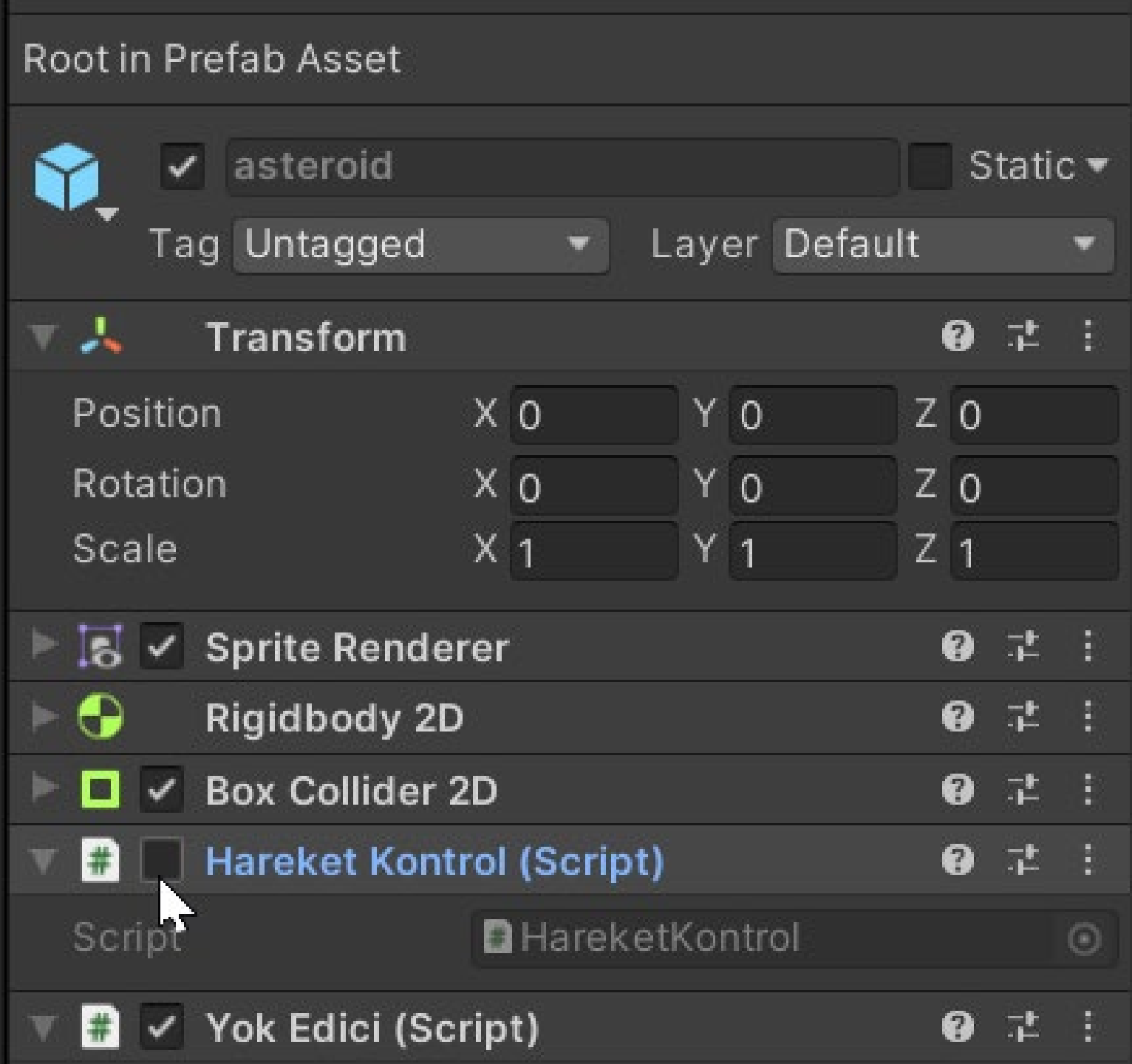


1.Yöntem Prefab objemiz seçili iken Inspector penceresinden Tag değerini seçerek bu seçimin tüm Prefab objeleri için geçerli olmasını sağlarız.



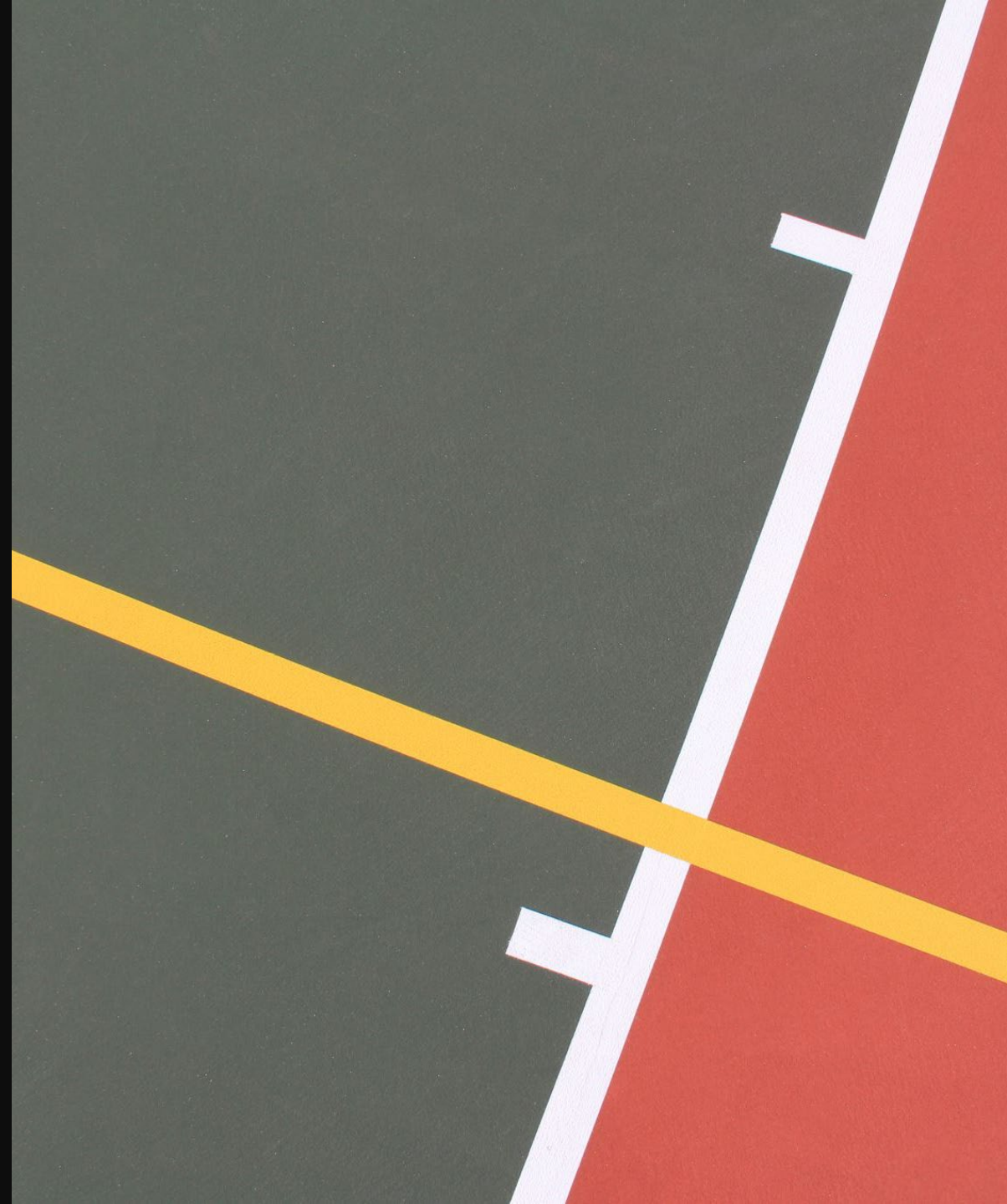
2.Yöntem sahnedeki oyun objemiz seçili durumda iken Inspector penceresinden Tag değerini ayarlayıp, Overrides seçeneklerindeki Apply All ile bu seçimin tüm Prefab objeleri için geçerli olmasını sağlarız.

Astreoid
Prefab
Üzerindeki
Force Kuvvetini
Kaldıralım.



YokEdici.cs

```
public class YokEdici :  
MonoBehaviour  
{  
    [SerializeField]  
    GameObject patlamaPrefab;  
    GeriSayimSayaci  
yokEdiciGeriSayim;  
}
```



YokEdici.cs

```
void Start()
{
    yokEdiciGeriSayim = gameObject.AddComponent<GeriSayimSayaci>();
    //yokEdiciGeriSayim.ToplamSure = Random.Range(1,20);
    //yokEdiciGeriSayim.Calistir();
}
```

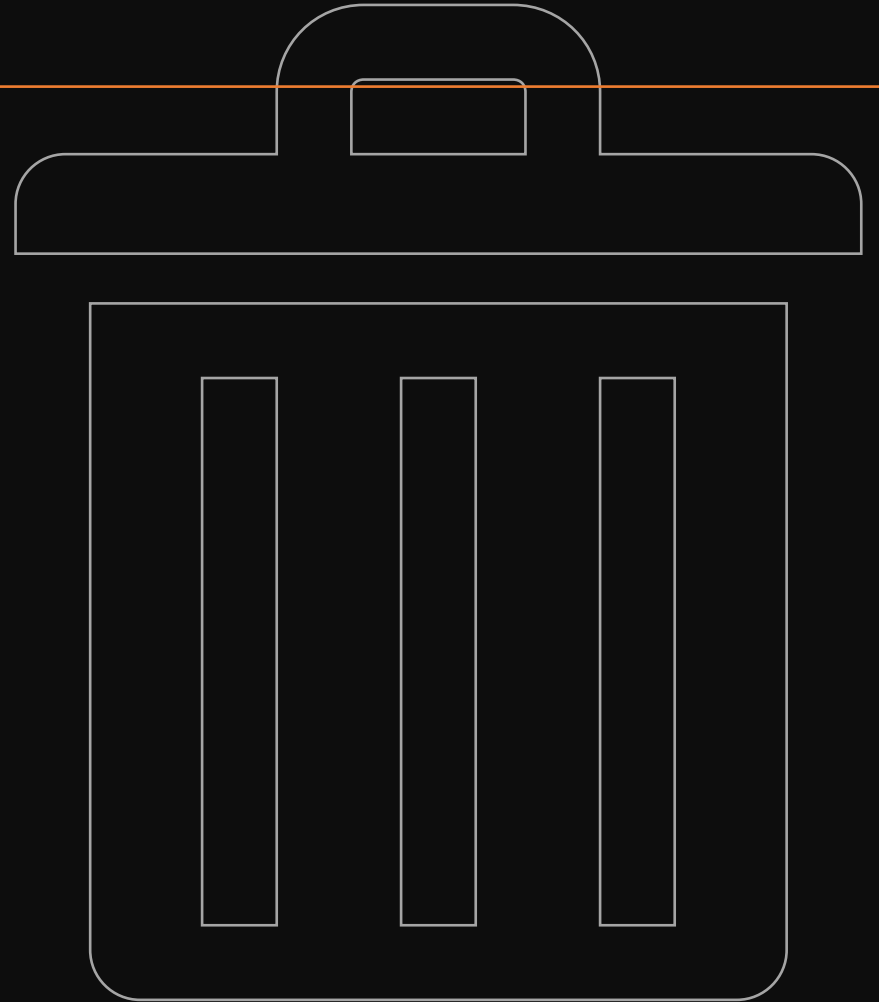

YokEdici.cs

```
void Update()
{
    if (yokEdiciGeriSayim.Bitti)
    {
        Instantiate(patlamaPrefab,
gameObject.transform.position,
Quaternion.identity);
        Destroy(gameObject);
    }
}
```



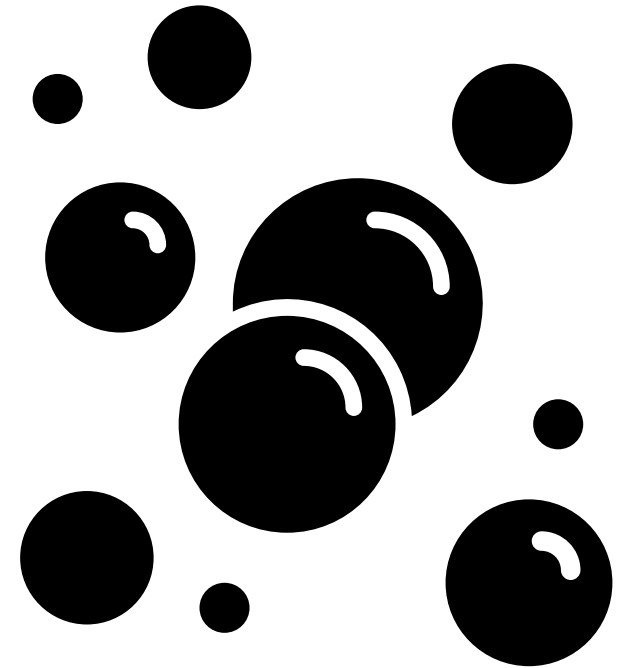
YokEdici.cs

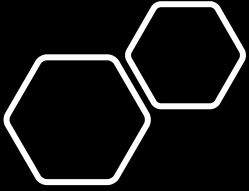
```
public void AsteroidYokEdici(int  
sure)  
{  
  
yokEdiciGeriSayim.ToplamSure =  
sure;  
  
yokEdiciGeriSayim.Calistir();  
}
```



PatlamaYokEdici.cs

```
public class PatlamaYokEdici :  
MonoBehaviour  
{  
    GeriSayimSayaci  
    gerisayimsayaci;  
  
    SiraliYokEdici  
    siraliYokEdici;  
}
```

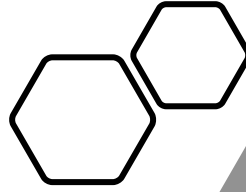




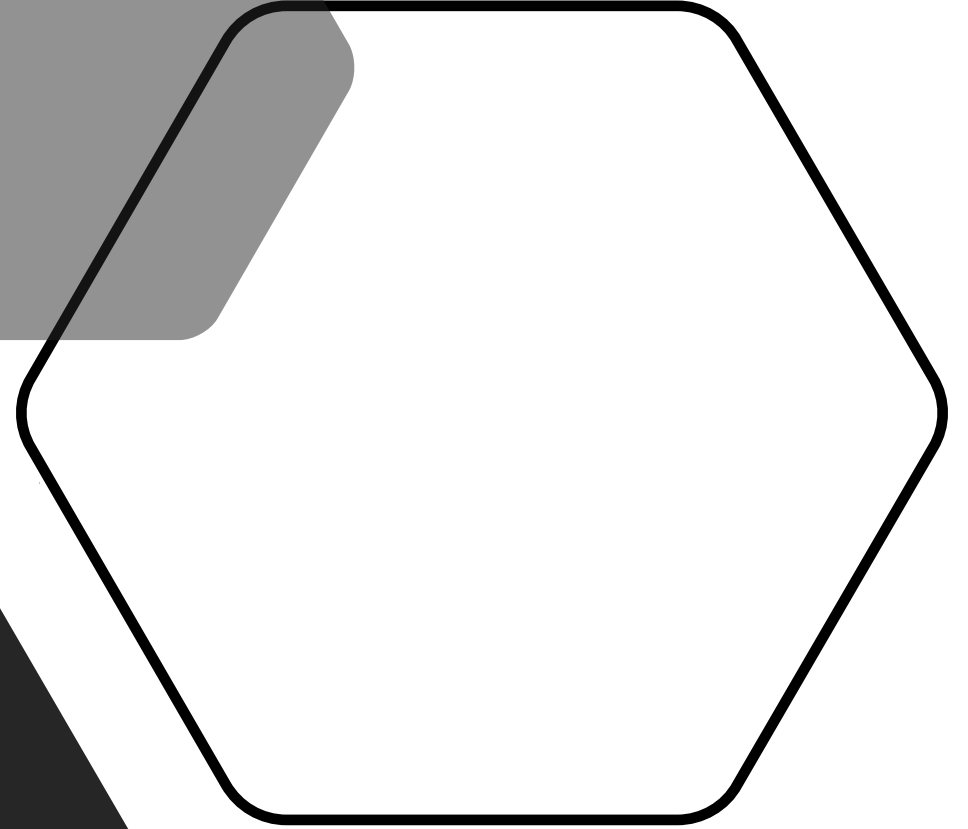
PatlamaYokEdici.cs

```
void Start()  
{  
    gerisayimsayaci =  
gameObject.AddComponent<GeriSayimSayaci>();  
    siraliYokEdici =  
Camera.main.GetComponent<SiraliYokEdici>();  
    gerisayimsayaci.ToplamSure = 1;  
    gerisayimsayaci.Calistir();  
}
```

PatlamaYokEdici.cs



```
void Update()
{
    if (gerisayimsayaci.Bitti)
    {
        siraliYokEdici.HedefiYokEt();
        Destroy(gameObject);
    }
}
```



SiraliYokEdici.cs

```
public class SiraliYokEdici :  
MonoBehaviour  
{  
    [SerializeField]  
    GameObject astreoidPrefab;  
    GameObject uzayGemisi;  
    List<GameObject> astreoidList;  
    GameObject hedefAstreoid;  
}
```



SiraliYokEdici.cs

```
void Start()
{
    uzayGemisi =
GameObject.FindGameObjectWithTag("Player");
    //FindGameObjectWithTag : Çoğul olmayan
haliyle fonksiyon seçimini yapıyoruz.
    astreoidList = new List<GameObject>();
}
```



SiraliYokEdici.cs

```
GameObject EnYakinAstreoid()  
{  
    GameObject enYakinAstreoid;  
    float enYakinMesafe;  
    if(astreoidList.Count==0)  
        return null;  
    else  
    {  
        enYakinAstreoid =  
astreoidList[0];  
        enYakinMesafe =  
MesafeOlcer(enYakinAstreoid);  
    }
```

```
foreach(GameObject astreoid in  
astreoidList)  
{  
    float mesafe =  
MesafeOlcer(astreoid);  
    if (mesafe<enYakinMesafe)  
    {  
        enYakinMesafe = mesafe;  
        enYakinAstreoid = astreoid;  
    }  
}  
return enYakinAstreoid;  
}
```


Kod Açıklamaları

```
enYakinAstreoid = astreoidList[0];
```

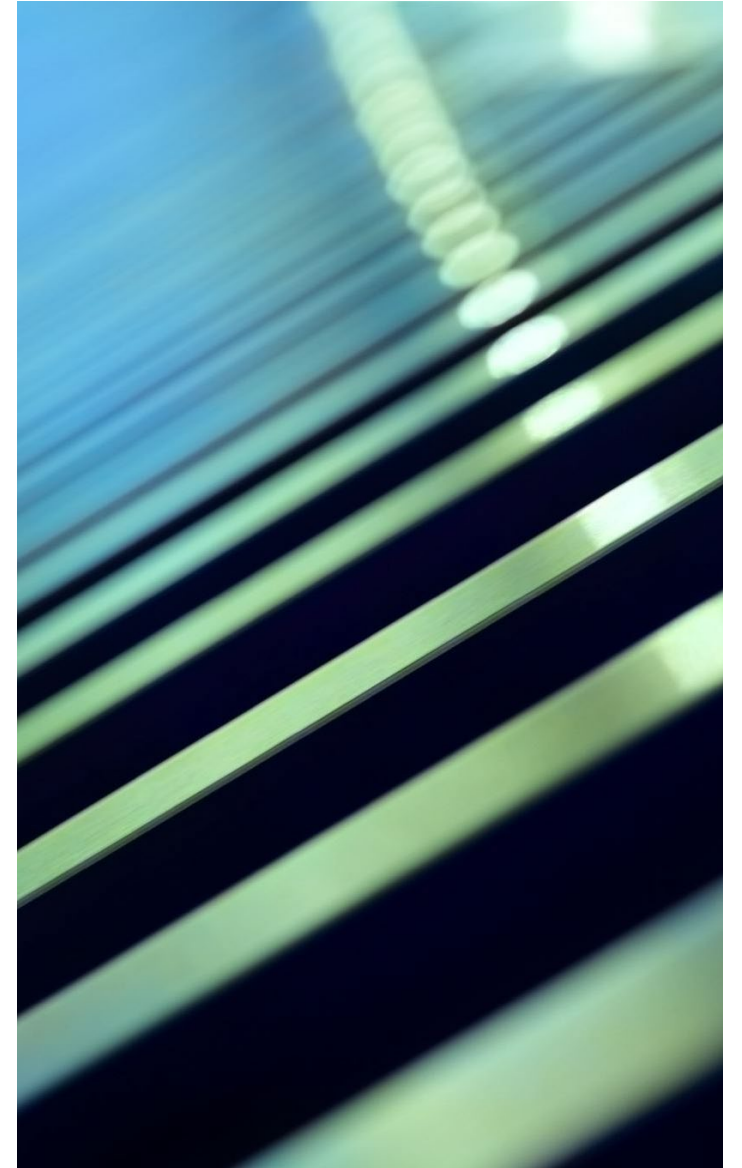
Listenin ilk elemanını
enYakinAstreoid olarak varsayalım

```
float mesafe = MesafeOlcer(astreoid);
```

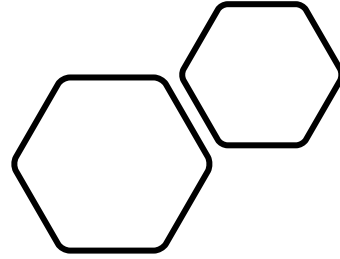
Uzay gemimize olan mesafeyi float
cinsinden döndürür

```
if (mesafe < enYakinMesafe)
```

Gelen mesafe en yakın mesafeden
daha küçükse en yakın mesafeyi bu
mesafe yapıyoruz



SiraliYokEdici.cs



```
float MesafeOlcer(GameObject hedef)
{
    return
    Vector3.Distance(uzayGemisi.transform.position,
    hedef.transform.position);
    //İki pozisyon arasındaki mesafeyi
    hesaplamamızı sağlar
    //Hedefimiz ile uzay gemimiz arasındaki
    mesafeyi geri döndürür
}
```



SiraliYokEdici.cs

```
public void HedefiYokEt()
```

```
{
```

```
    hedefAstreoid = EnYakinAstreoid();
```

```
    if (hedefAstreoid!=null)
```

```
{
```

```
        hedefAstreoid.GetComponent<YokEdici>().AsteroidYokEdici(1);
```

```
        astreoidList.Remove(hedefAstreoid);
```

```
    }
```

```
}
```

SiraliYokEdici.cs

```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        Vector3 position = Input.mousePosition;
        position.z = -Camera.main.transform.position.z;
        position = Camera.main.ScreenToWorldPoint(position);
        GameObject astreoid = Instantiate(astreoidPrefab, position, Quaternion.identity);
        astreoidList.Add(astreoid);
    }
    if (Input.GetMouseButtonDown(1))
        HedefiYokEt();
}
```

