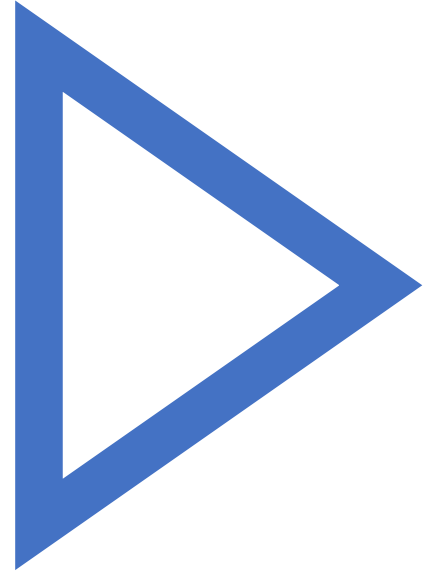


Unity

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 Serileştirilmesi)

- 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.

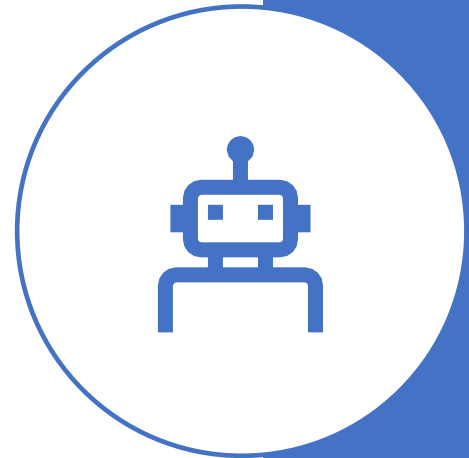
Unity tüm komut dosyası bileşenlerinizi serileştirir, yeni derlemeleri yeniden yükler ve komut dosyası bileşenlerinizi serileştirilmiş sürümlerden yeniden oluşturur. Bu serileştirme işlemi .NET'in serileştirme işlevinin yerine dahili Unity serileştirme sistemi ile yapılır.

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 ögesinden 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

Serializefield
Attribute
Instantiate
Metodu

Fueled By Our Awesome Patrons

- 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;

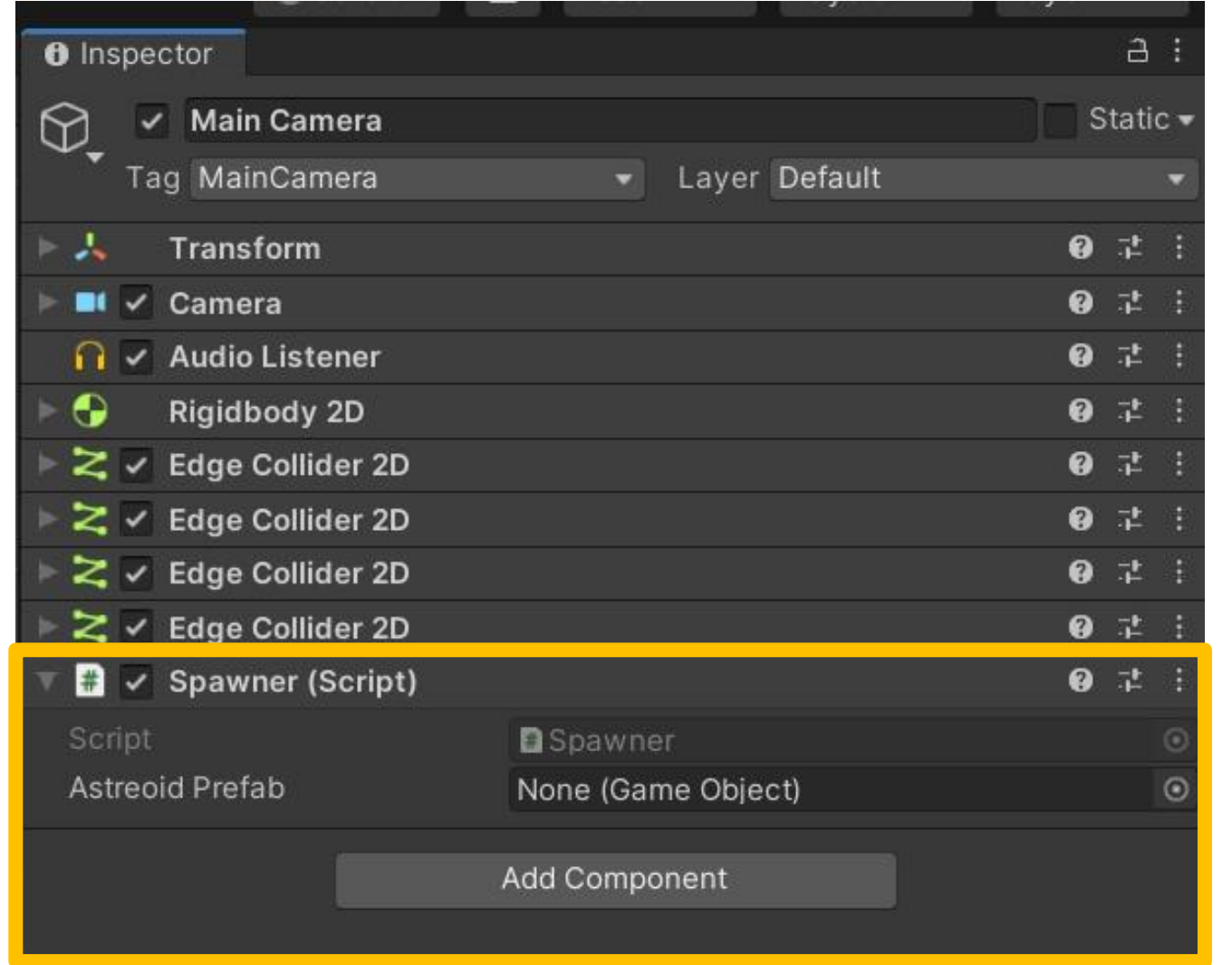


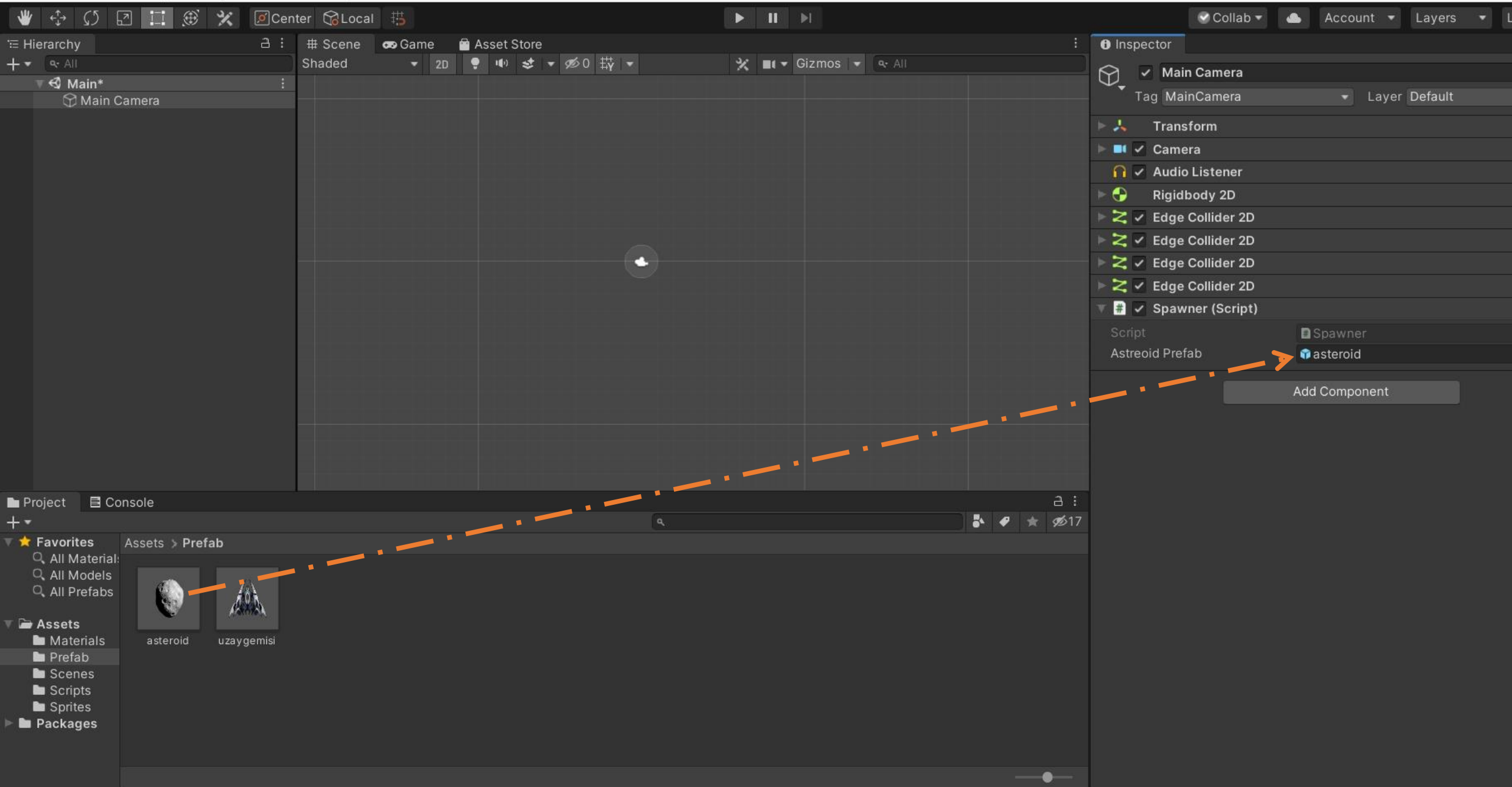
Spawner.cs

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

SerializeField ile Objeye Editör İerisinden Mdahale Etmek Mmkn

- Oyun objemiz Public olmamasına raėmen, SerializeField ile Editr ierisindeki Inspector penceresinden objemize mdahale edebiliyoruz.





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 Çağırılması

- 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();  
    }  
}
```



Hand Rotate Lock Center Local Grid Play Pause Step Forward

Hierarchy Scene Game Asset Store

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

Main

- Main Camera
- asteroid(Clone)
- asteroid(Clone)



Project Console

Clear Collapse Clear on Play Clear on Build Error Pause Editor

[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 9

Main Camera tag'li nesne ile çarpışmayı bıraktı

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

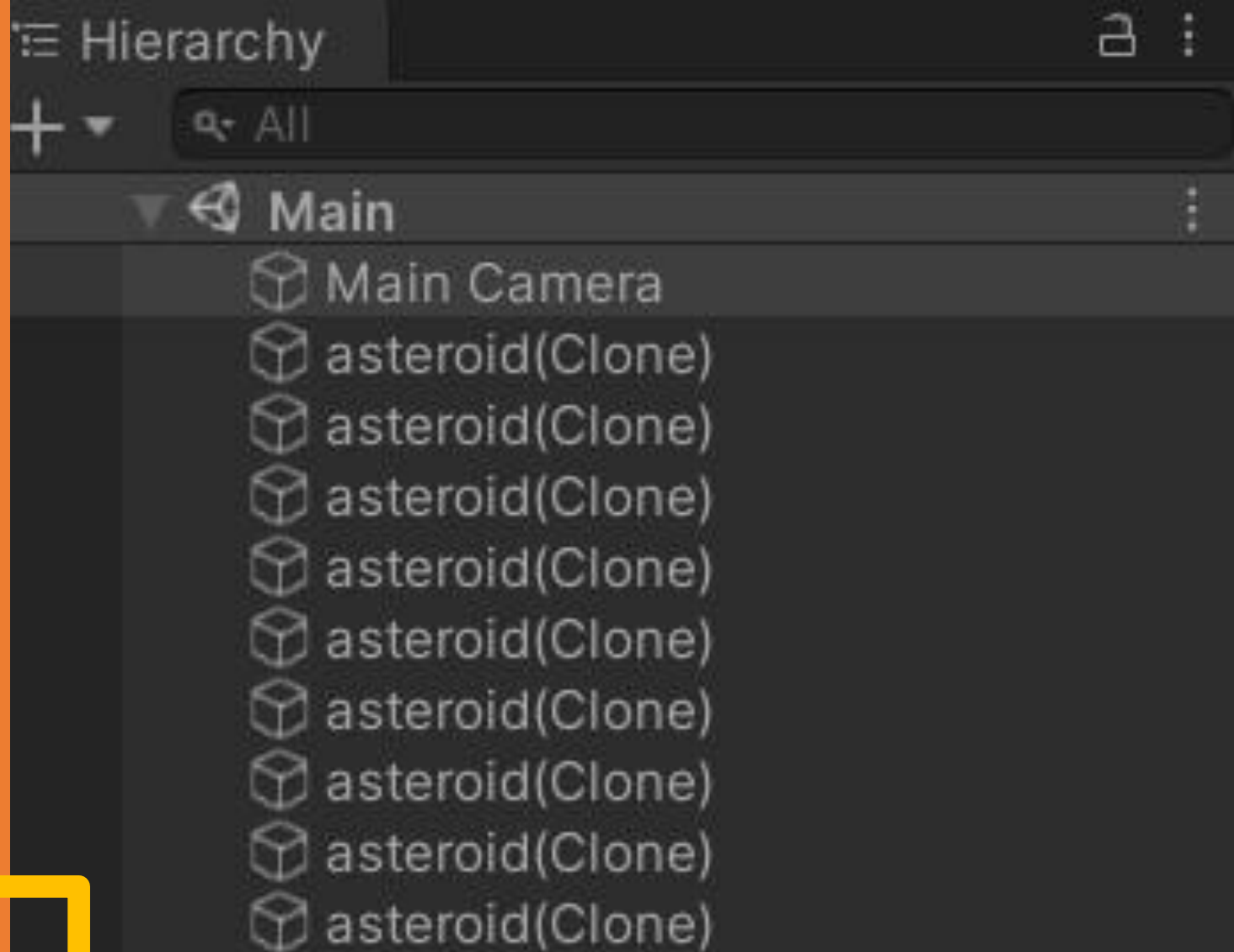
Geri Sayim Sayaci (Script)

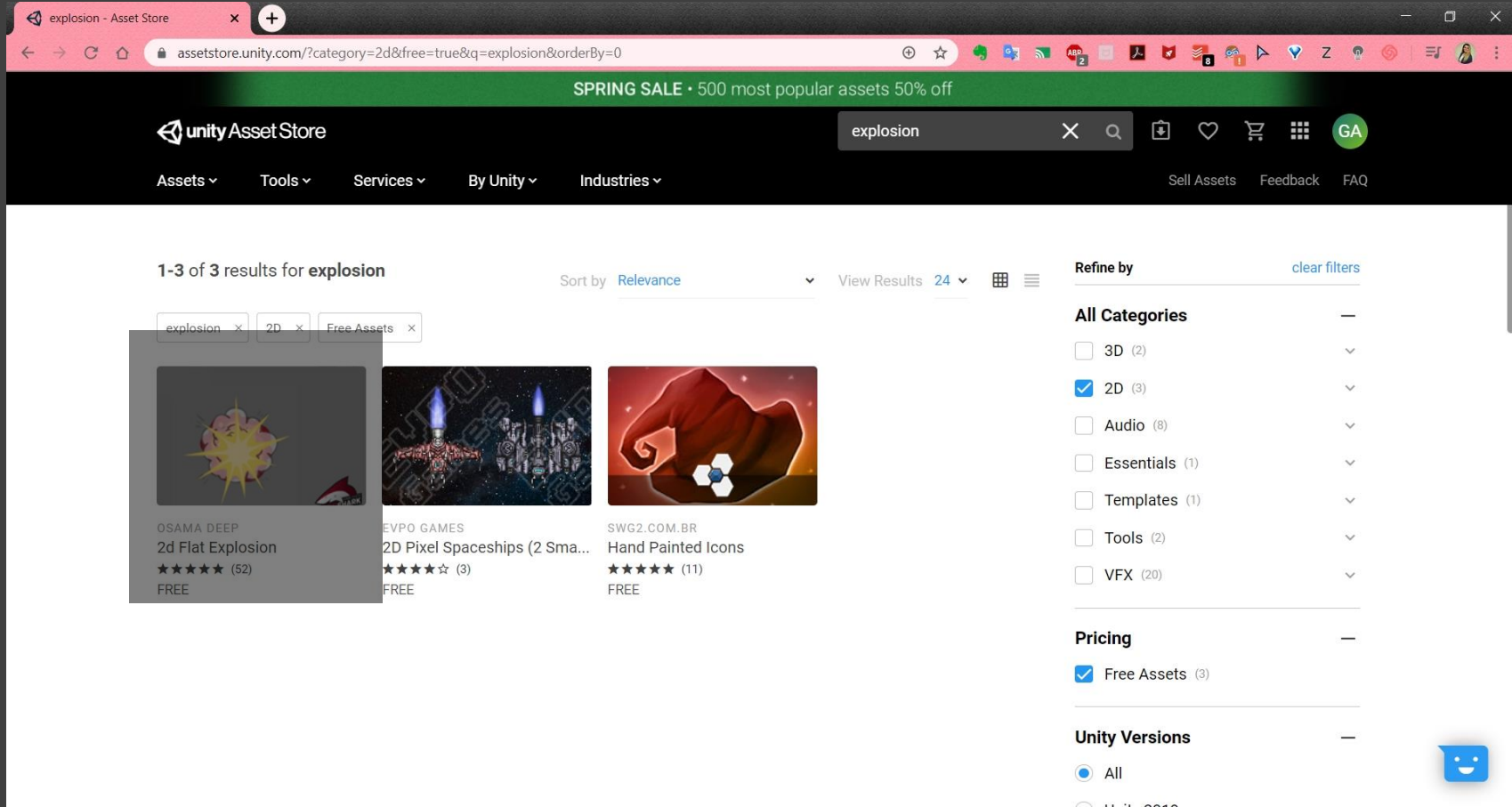
Script

GeriSayimSayaci

Add Component

astreoid

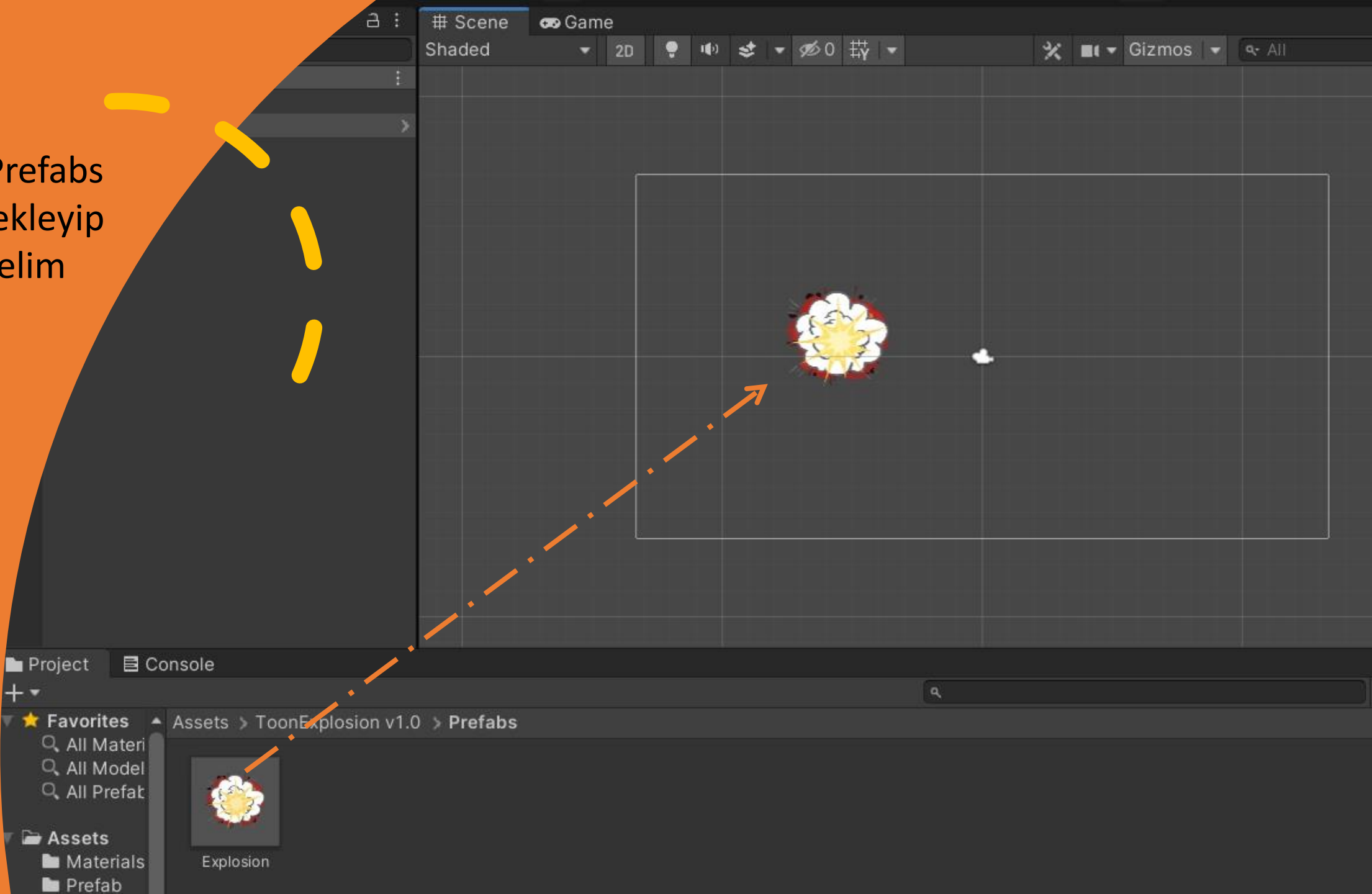




Asset Store üzerinden
indirdiğimiz
Explosion (Patlama)
efektini projemize dahil
edelim

- 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



GERİSAYİMSAYACI
SCRIPT'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();
}
```

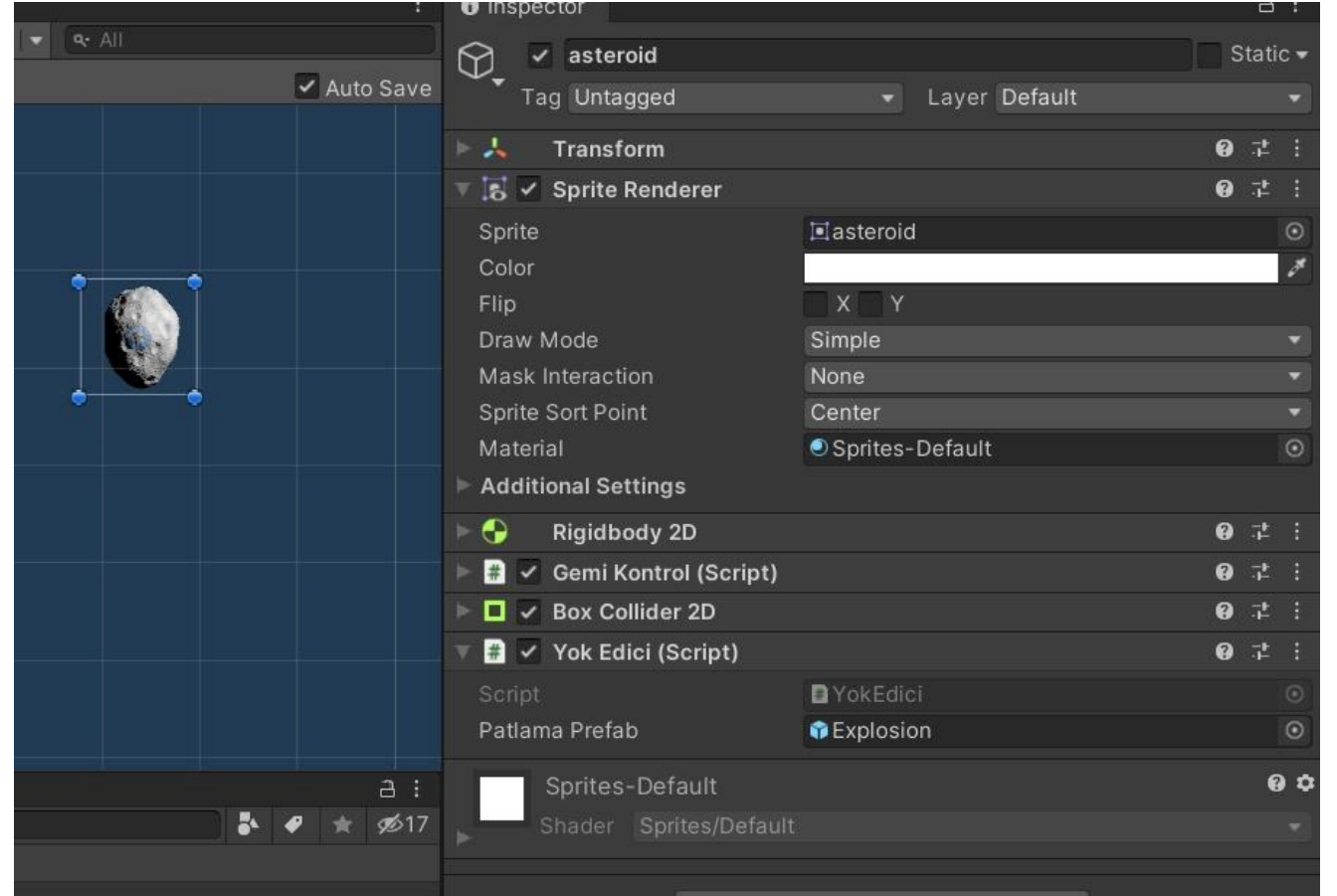
YokEdici.cs


```
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.

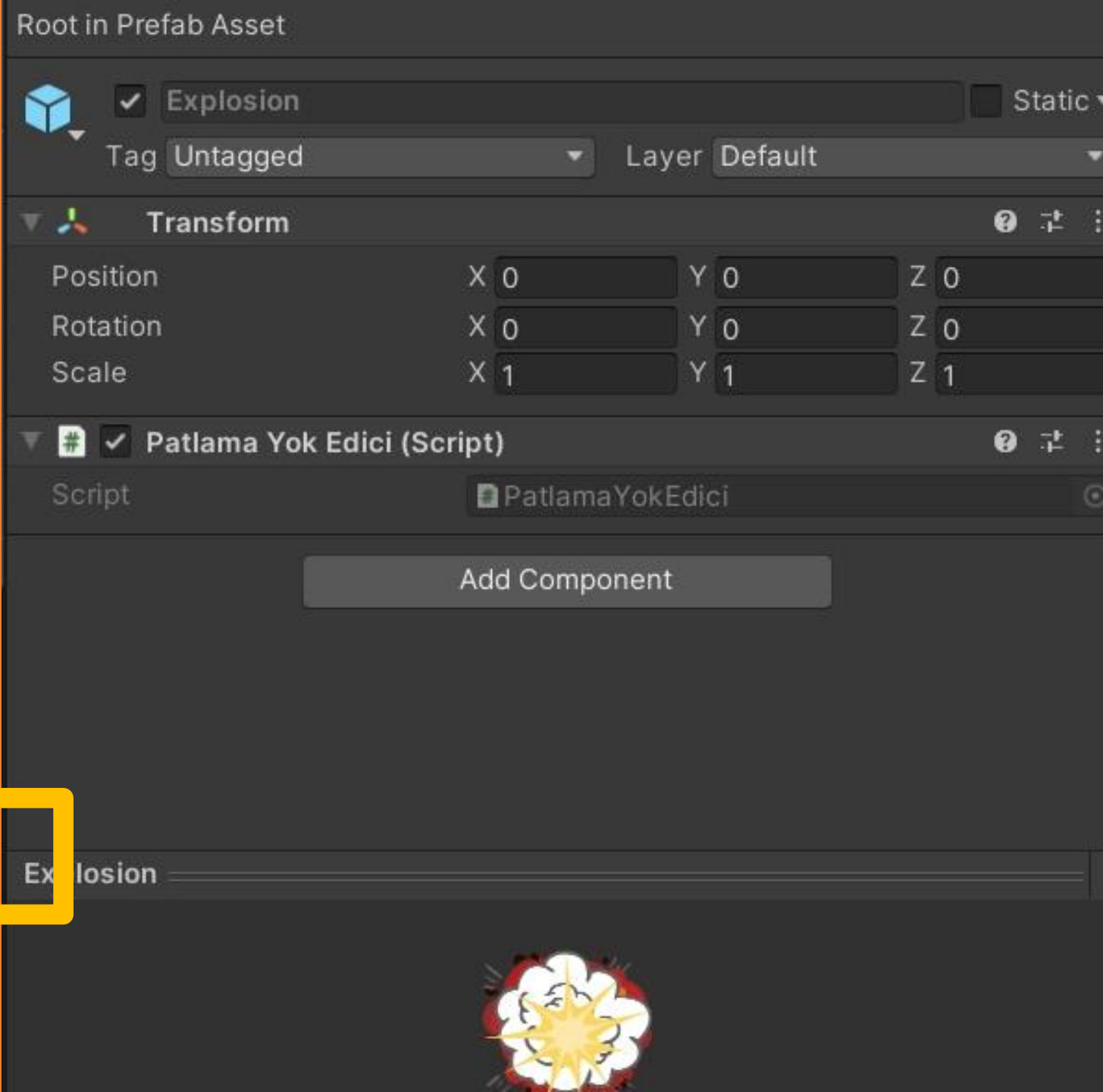
```
void Update()  
{  
    if (yokEdiciGeriSayim.Bitti)  
    {  
        Instantiate(patlamaPrefab,  
gameObject.transform.position, Quaternion.identity);  
        //gameObject.transform.position : patlamaPrefab  
position değerini objemizin position değeri olarak belirledik.  
        //Quaternion.identity : 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<GeriSayi
mSayaci>();

    //gameObject bu script'i
eklediğimiz objeyi ifade eder.


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

    yokEdiciGeriSayim.Calistir();
}
```

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

Hierarchy

Main*
Main Camera
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
asteroid(Clone)
Explosion(Clone)
asteroid(Clone)

Scene

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

Project Console

Favorites
All Material:
All Models
All Prefabs

Assets
Materials
Prefab
Scenes
Scripts
Sprites
ToonExplos v1.0

3.007179

Oyuncu Etkileşimi





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.

▼ Fire1	
Name	Fire1
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	left ctrl
Alt Negative Button	
Alt Positive Button	mouse 0
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button ▼
Axis	X axis ▼
Joy Num	Get Motion from all Joysticks ▼

▼ Fire1	
Name	Fire1
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	joystick button 0
Alt Negative Button	
Alt Positive Button	
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button ▼
Axis	X axis ▼
Joy Num	Get Motion from all Joysticks ▼

Fire2

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

▼ Fire2	
Name	Fire2
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	left alt
Alt Negative Button	
Alt Positive Button	mouse 1
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button ▼
Axis	X axis ▼
Joy Num	Get Motion from all Joysticks ▼

▼ Fire2	
Name	Fire2
Descriptive Name	
Descriptive Negative Name	
Negative Button	
Positive Button	joystick button 1
Alt Negative Button	
Alt Positive Button	
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>
Type	Key or Mouse Button ▼
Axis	X axis ▼
Joy Num	Get Motion from all Joysticks ▼

Version: 2019.3▼

Input

- IntegratedSubsystem
- IntegratedSubsystemDescriptor
- Joint
- Joint2D
- JointAngleLimits2D
- JointDrive
- JointLimits
- JointMotor
- JointMotor2D
- JointSpring
- JointSuspension2D

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour
{
    public void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            Debug.Log(Input.mousePosition);
        }
    }
}
```

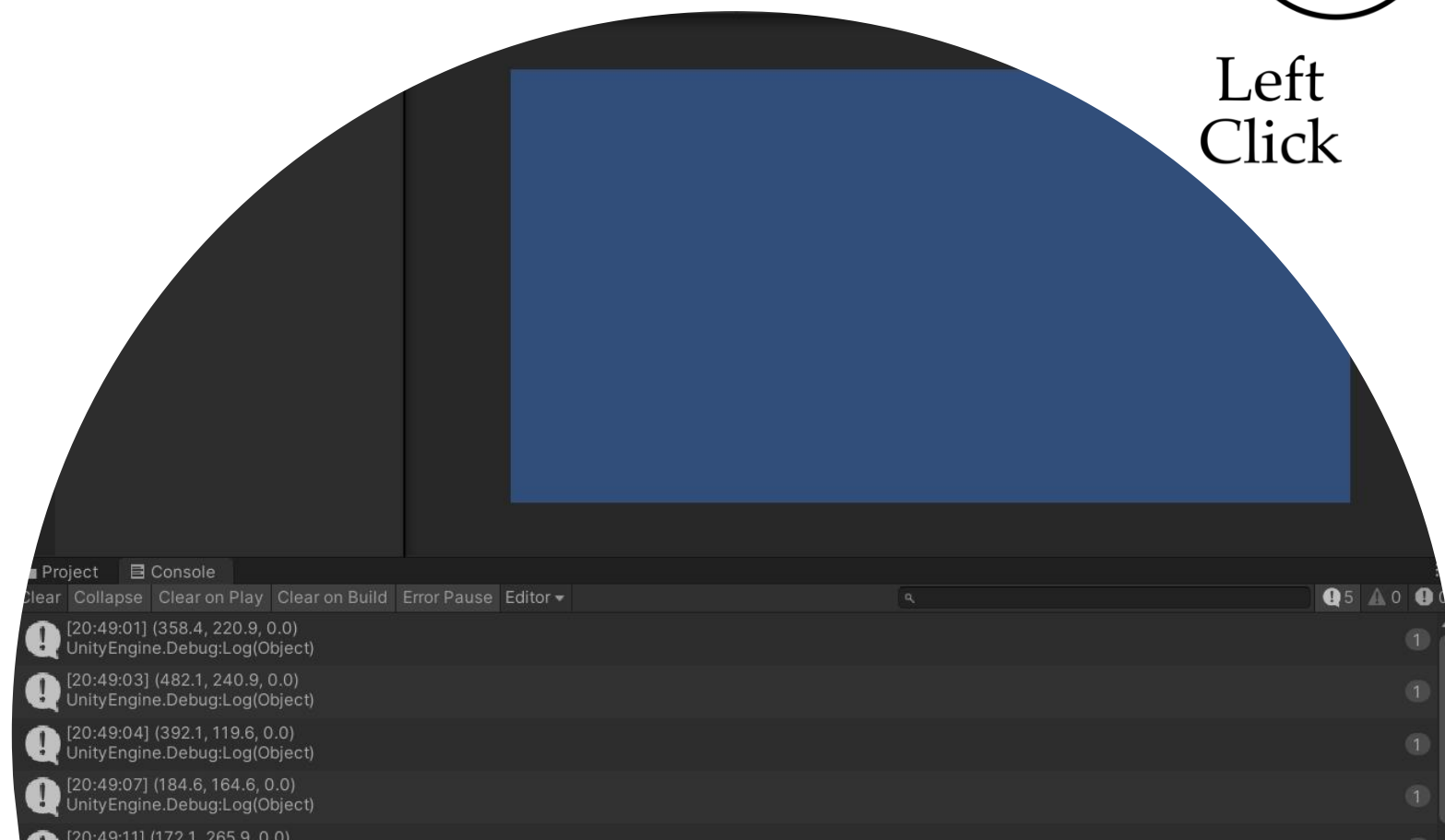


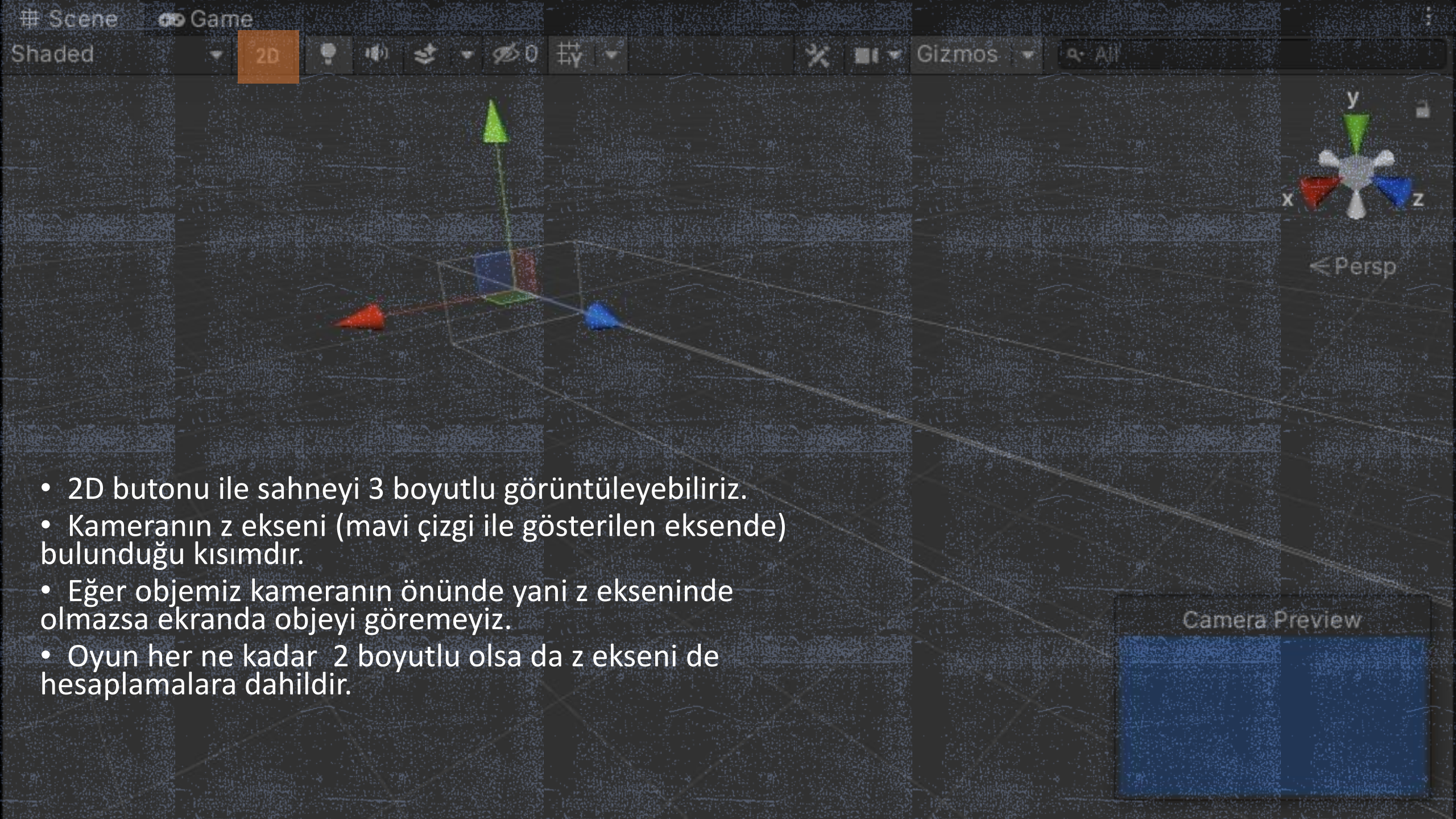
InputKontrol.cs

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



Left
Click



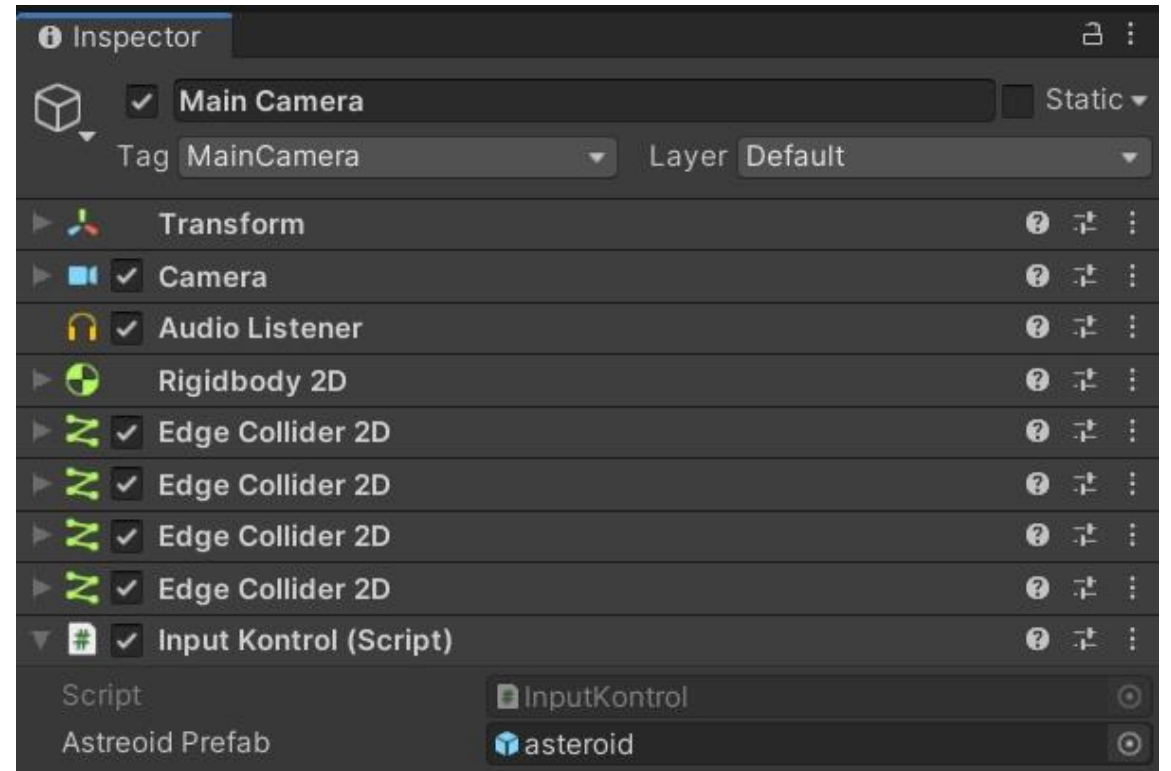


- 2D butonu ile sahneyi 3 boyutlu görüntüleyebiliriz.
- Kameranın z ekseni (mavi çizgi ile gösterilen ekseninde) bulunduğu kısımdır.
- Eğer objemiz kameranın önünde yani z ekseninde olmazsa ekranda objeyi göremeyiz.
- Oyun her ne kadar 2 boyutlu olsa da z eksenini de hesaplamalara dahildir.

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);
```

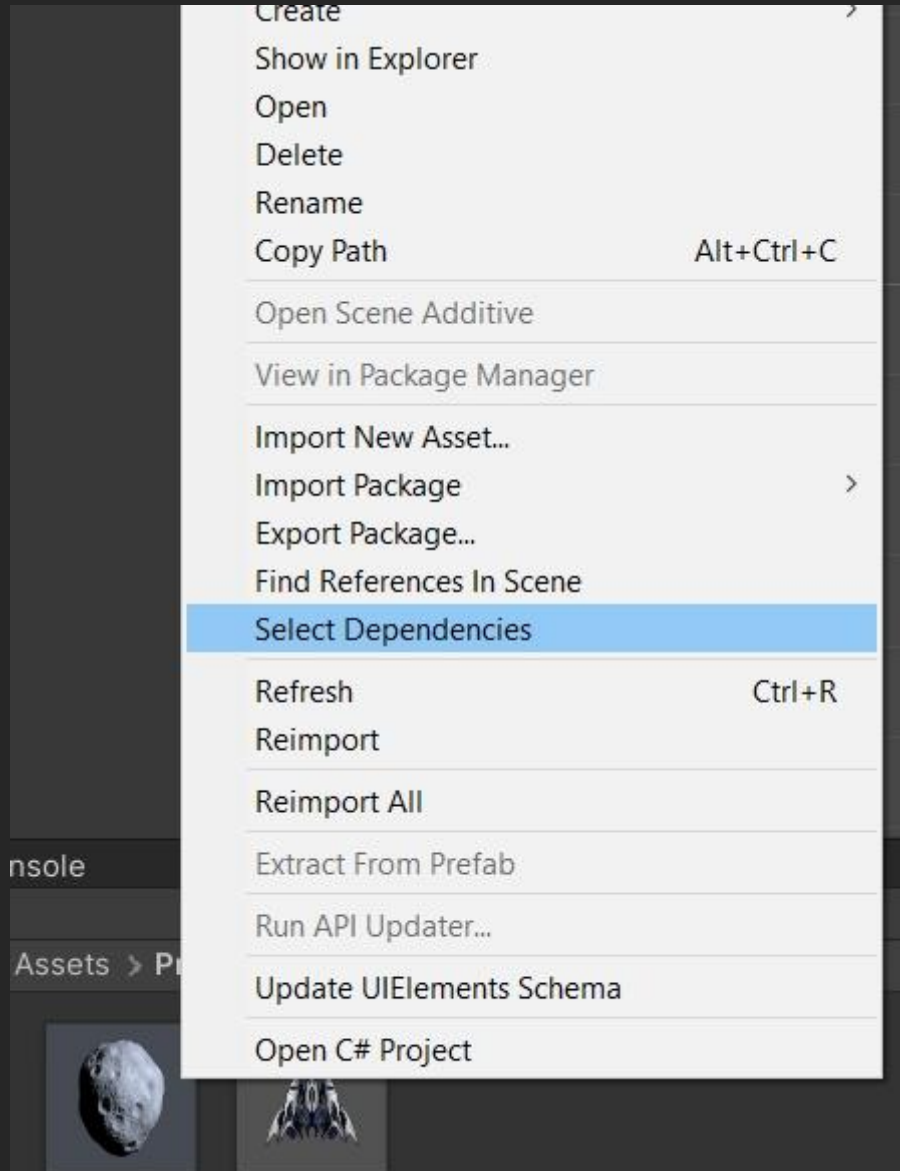
- ScreenToWorldPoint -> piksel değerlerimizi oyun dünyasındaki değerlere dönüştürür.
- Mouse position değeri olarak Unity'nin ekran piksellerine göre verdiği değeri oyun dünyasının kendi koordinatlarına çevirdik.
- Kamerayı referans alıyoruz. Çünkü kameranın yerini değiştirsek bile bu durum objemizin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

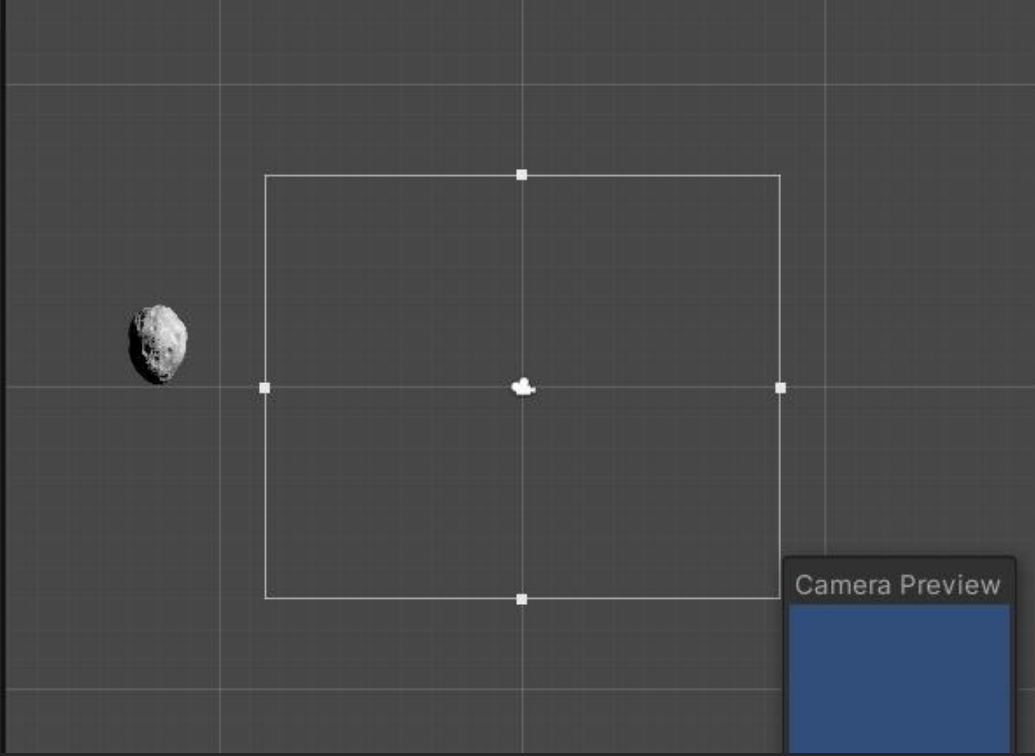
—

A stylized sun graphic on the left side of the slide. It features a solid yellow circle at the bottom left, with several short, curved yellow lines above it, suggesting rays. The background is split into an orange upper half and a white lower half, separated by a curved line.

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 kameranin gördüğü alanların dışına çıkmaması için belirli bir alan içerisinde hareket etmesine izin verilmesi gerekir.

Vector3(x,y,z)

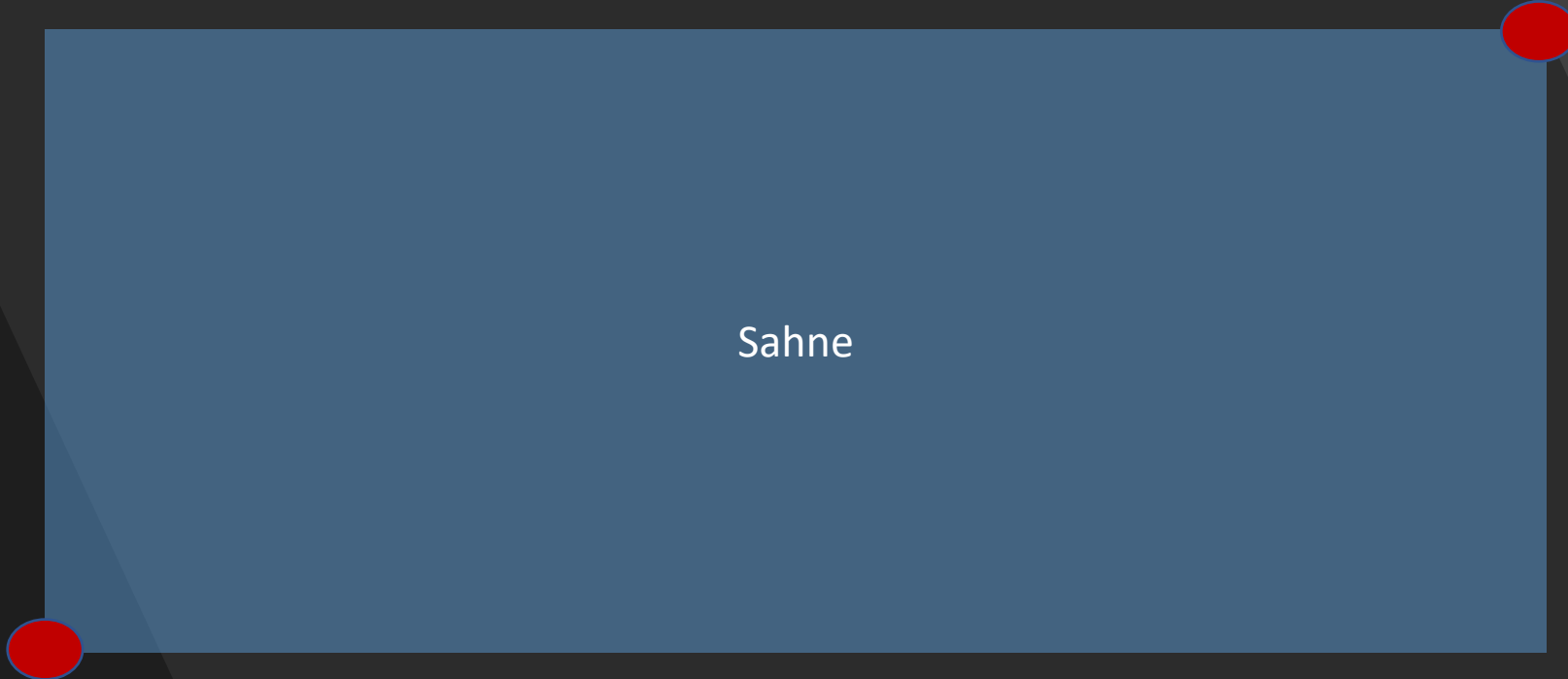
sagUstKose

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

Sahne

Vector3(0, 0, zEkseni)

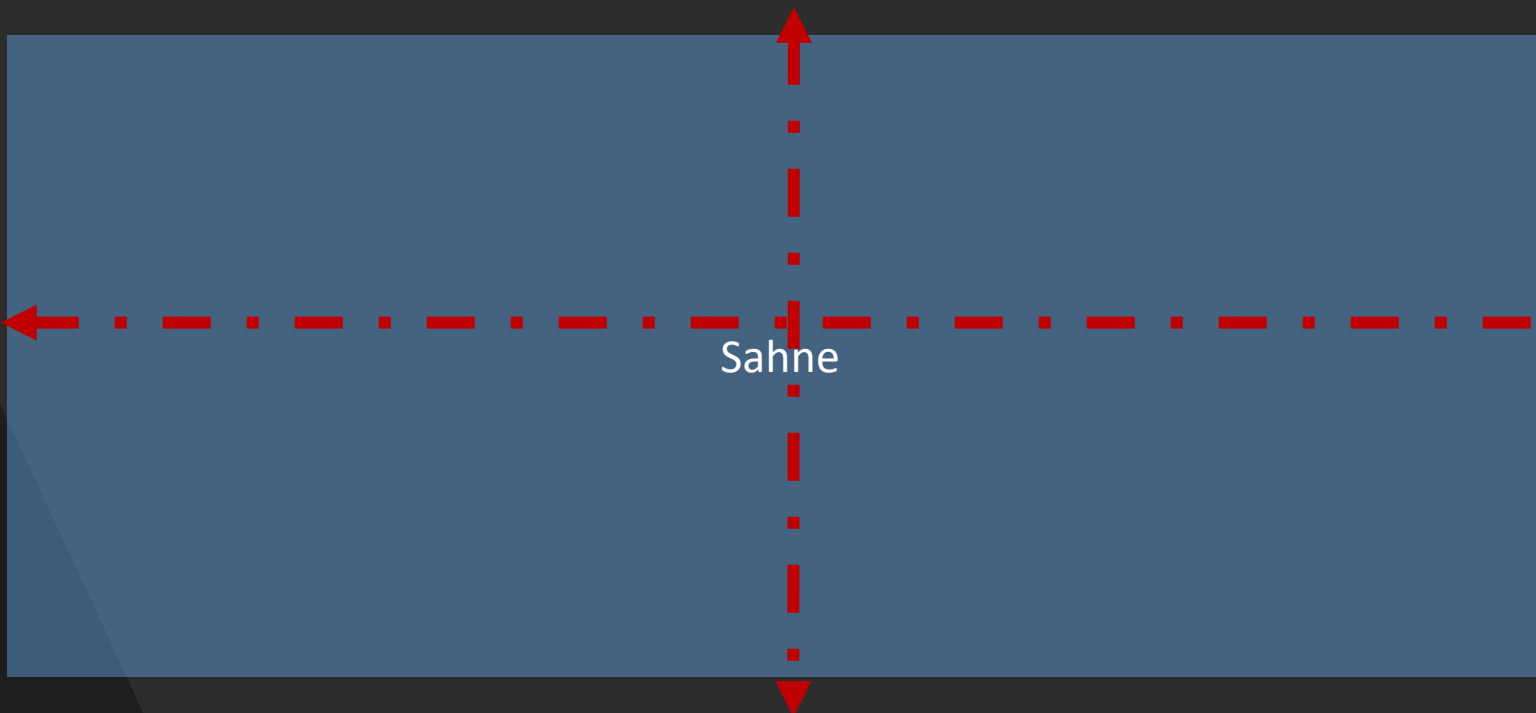
solAltKose



Vector3(x,y,z)

ust = sagUstKose.y

sol = solAltKose.x



Sahne

sag = sagUstKose.x

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 zEksenini = -Camera.main.transform.position.z;
    Vector3 solAltKose = new Vector3(0, 0, zEksenini);
    Vector3 sagUstKose = new Vector3(Screen.width,
Screen.height, zEksenini);

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

    sol = solAltKoseOyunDunyasi.x;
    sag = sagUstKoseOyunDunyasi.x;
    ust = solAltKoseOyunDunyasi.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.