

Unity

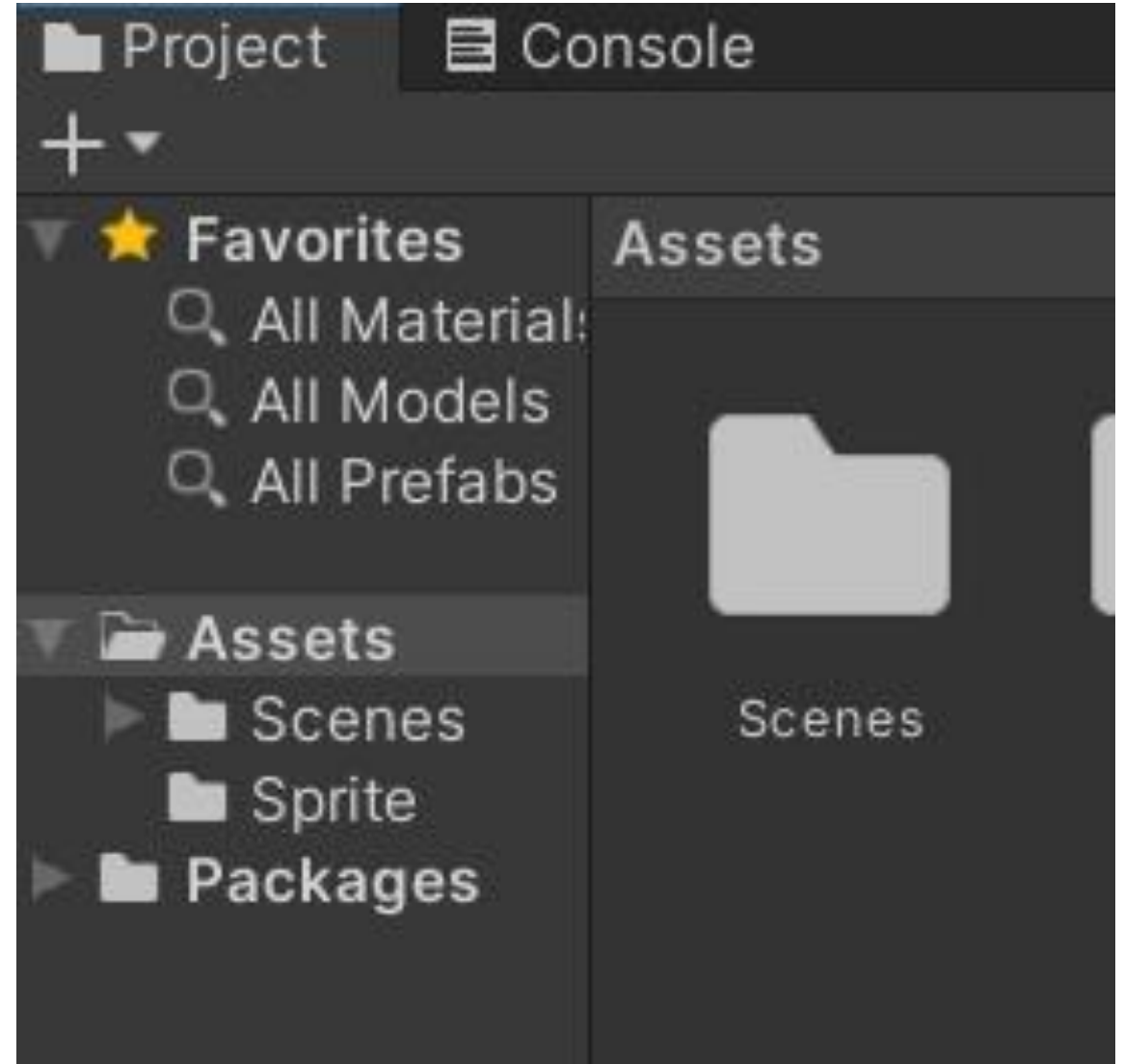
Öğr. Gör. Gözde Mihran ALTINSOY

Sprite Nesnesini Projemize Ekleyelim

1. Öncelikle Sprite'larımız için Sprite klasörünü oluşturalım.

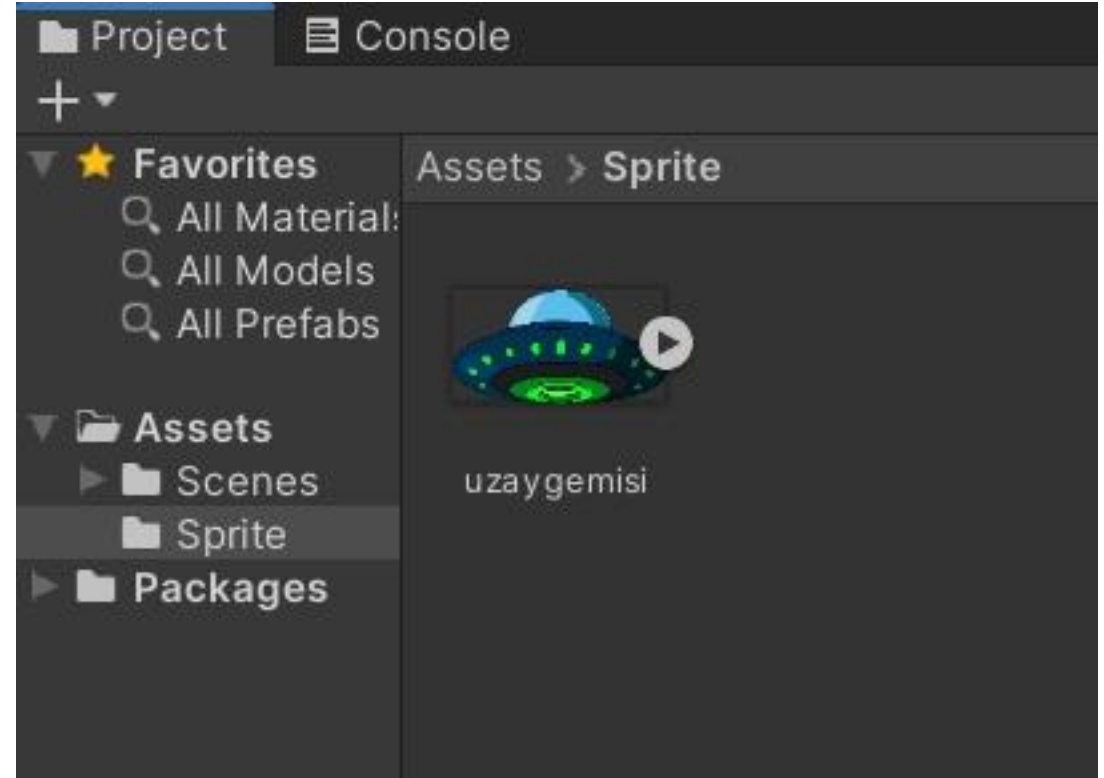
Bunun için;

- 1. yöntem Project yerleşimindeki Assets klasöründe sağ tıklayıp Create → Folder seçeneğini seçmektir.
- 2. yöntem Assets menüsünde yer alan Create → Folder seçeneğini seçmektir.



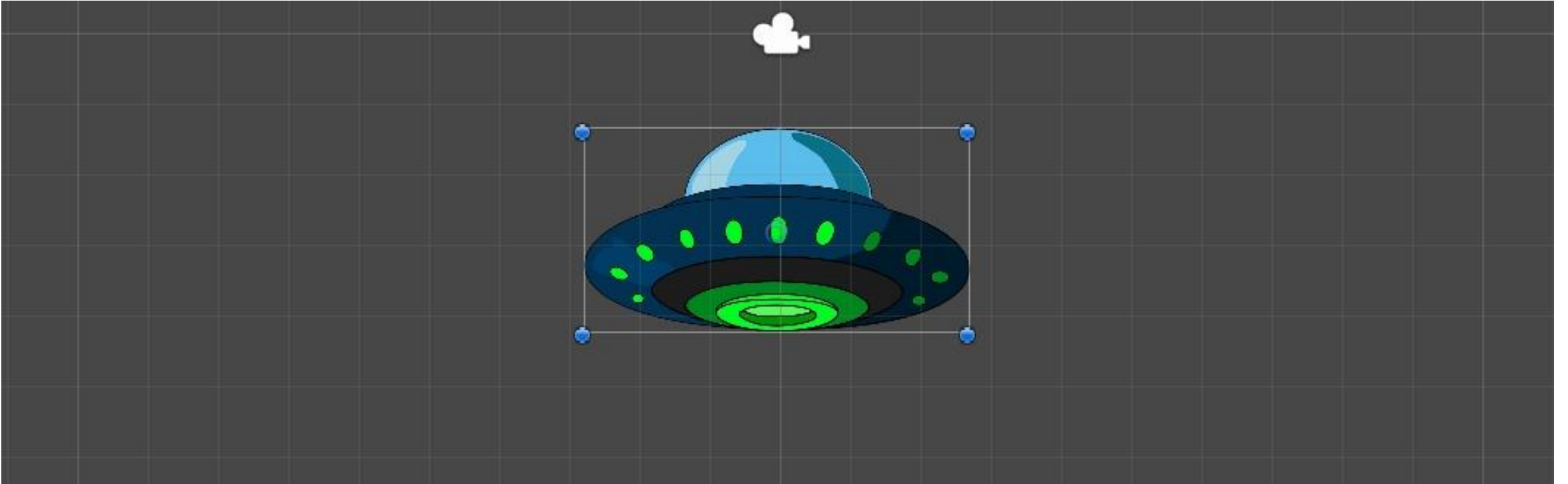
Sprite Nesnesini Projemize Ekleyelim

2. Sprite klasörüne çift tıklayarak klasörün içerisine girelim
3. Daha önce bilgisayarımıza kaydetmiş olduğumuz Sprite (görsel materyali) sürükleyerek Sprite içerisine bırakalım.

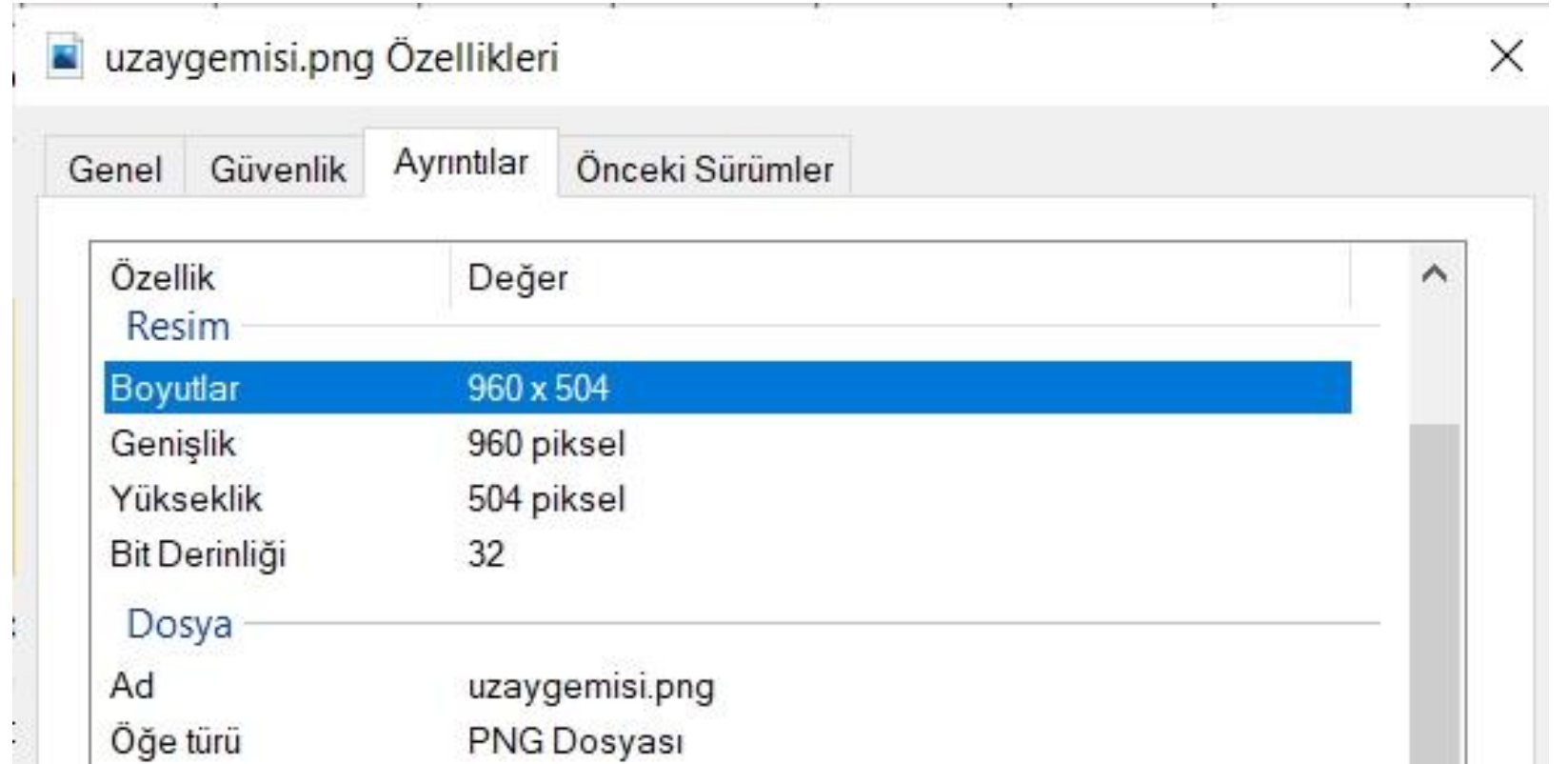


Sprite Nesnesini Projemize Ekleyelim

- Sprite nesnemizi sahneye eklemek için;
 - Sprite nesnemizi sürükleyerek sahneye bırakalım.
 - Hierarchy'de 2D Object → Sprite nesnesi ekleyelim. Sprite Renderer ile Sprite seçeneğinde nesnemizi seçelim.



Sprite Boyutları



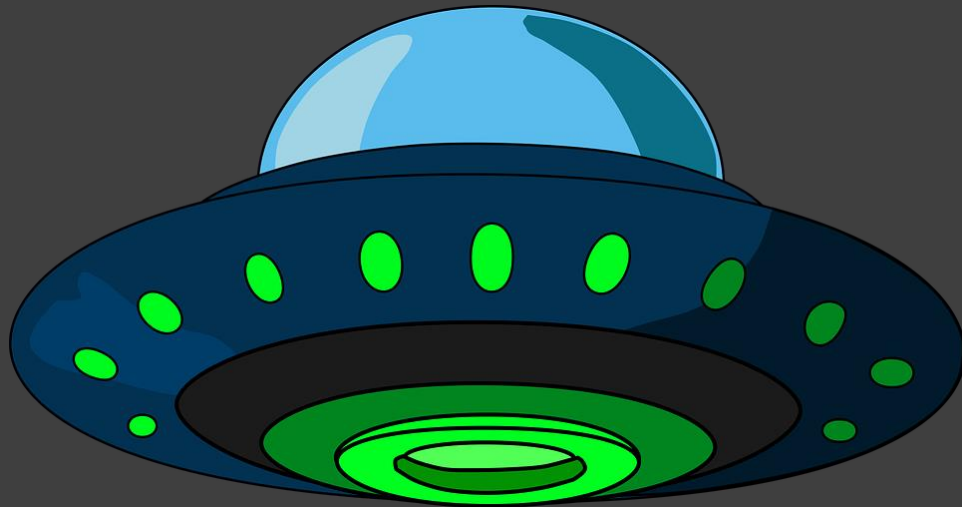
- Sprite boyutu 2'nin kuvveti olmalıdır. Render işleminin yapıldığı grafik kartlarının olabildiğince verimli kullanılabilmesi bu değerlerinin 2'nin kuvveti olması ile ilgili bir durumdur.

Component

- Sahnedeki oyun objelerimizin karakteristiğini tamamıyla o objeye ekli olan bileşenler yani Component'ler belirler.

Unity Bileşenleri (Component)

Sprite



Inspector



uzaygemisi

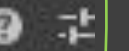
Stat

Tag Untagged

Layer Default



Transform



Position

X 0.09

Y -4.94

Z 0

Rotation

X 0

Y 0

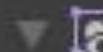
Z 0

Scale

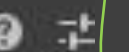
X 0.431697

Y 0.431697

Z 0.431



Sprite Renderer



Sprite

uzaygemisi2

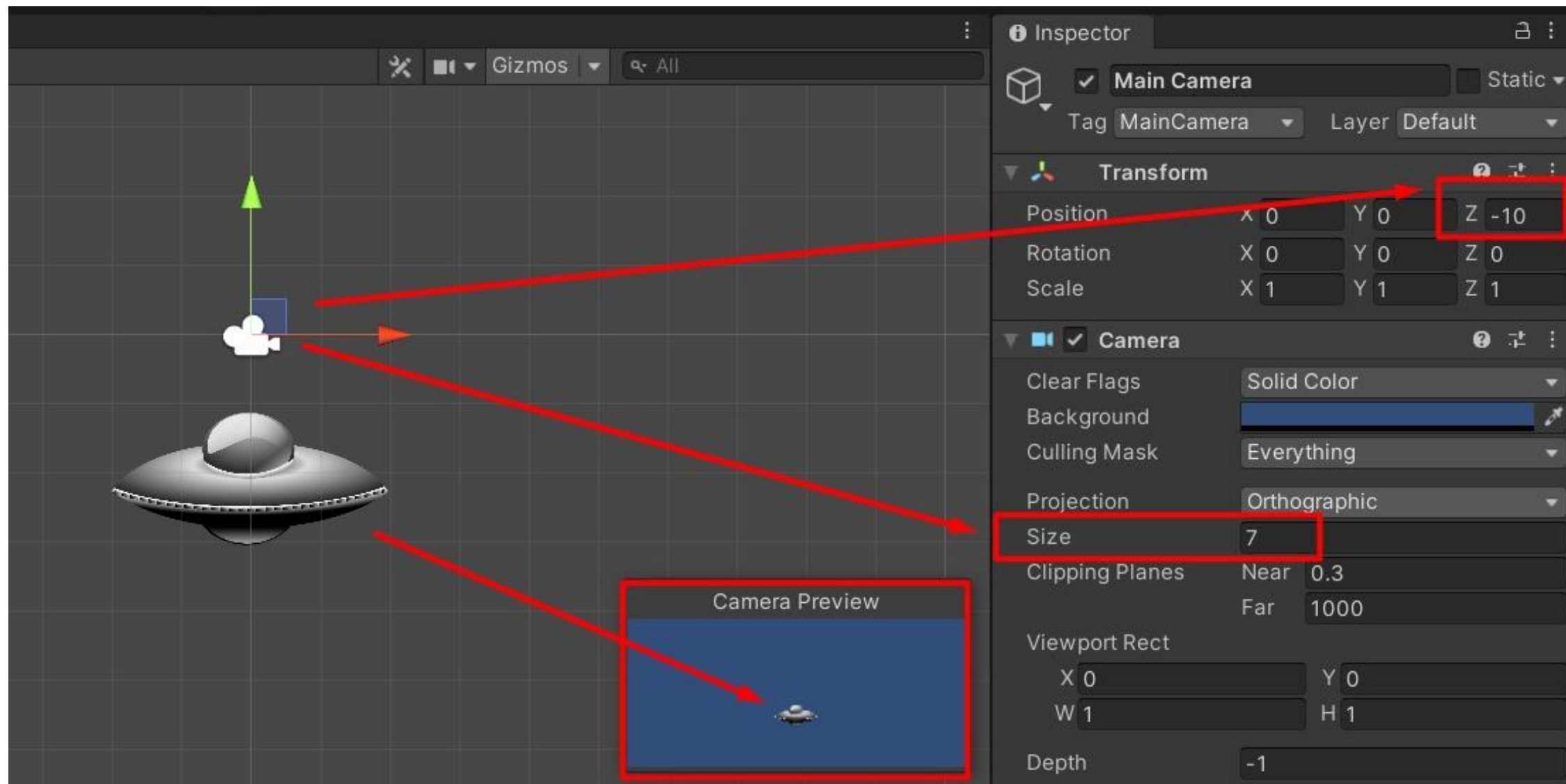
Color

Game Object

Camera Preview

Kameramızı nesnemizi görebilecek açıya getirmek istiyorsak Inspector panelindeki Position Z değeri negatif değer yapılmalıdır.

Aynı zamanda sahne büyüklüğü Size değeri ile ayarlanabilir.





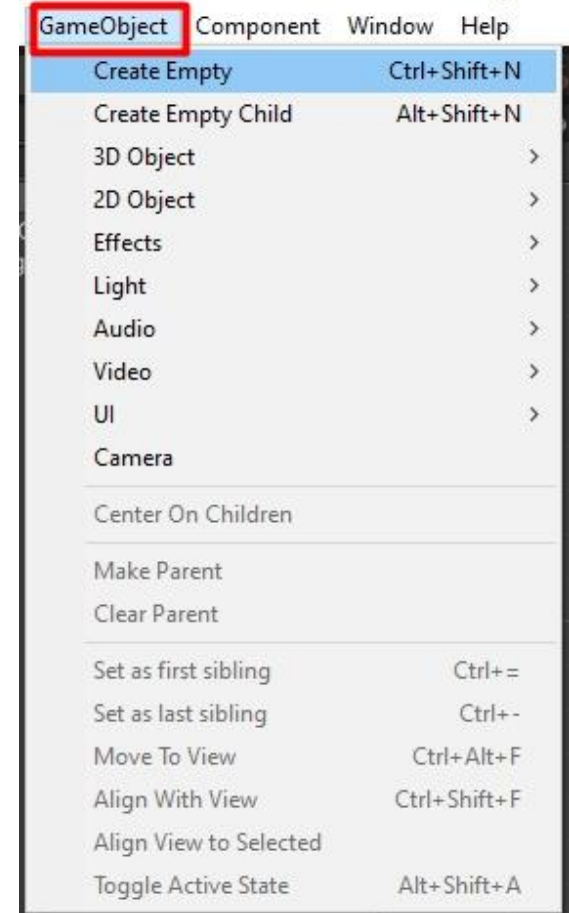
Audio Listener Component

- Camera nesnesinde olur.
- Oyunumuzdaki nesnelerin seslerinin oynatabilmesi için bir yerden dinleniyor olması gerekiyor ki oynatılabilsin yani Play işlemi gerçekleştirilebilsin.

GameObject Ekleme (Ör.Create Empty)

1. Yöntem:

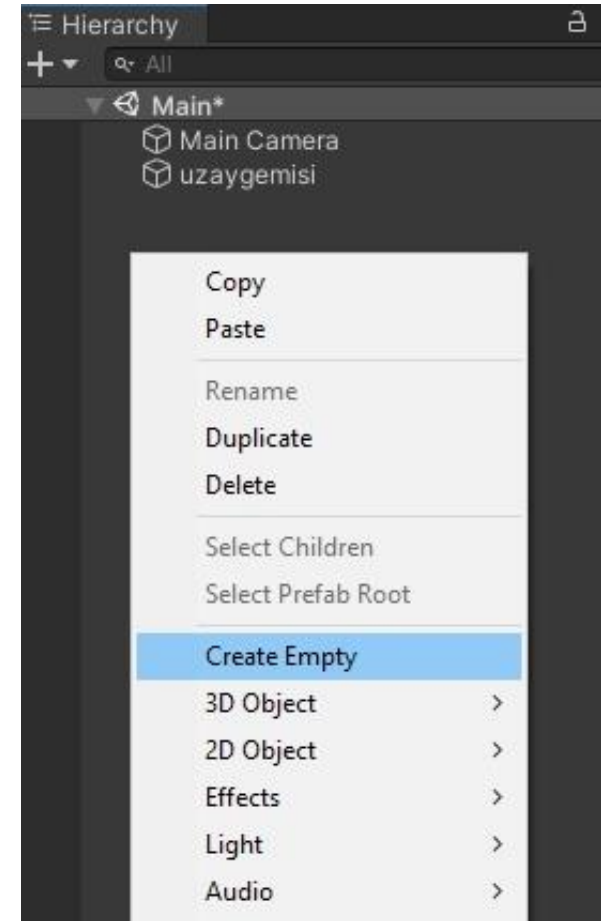
- GameObject sekmesine tıklandığında gelen seçeneklerden Create Empty nesnesi seçilir.
- Diğer nesnelerde bu yöntemle eklenebilir.



GameObject Ekleme (Ör.Create Empty)

2. Yöntem:

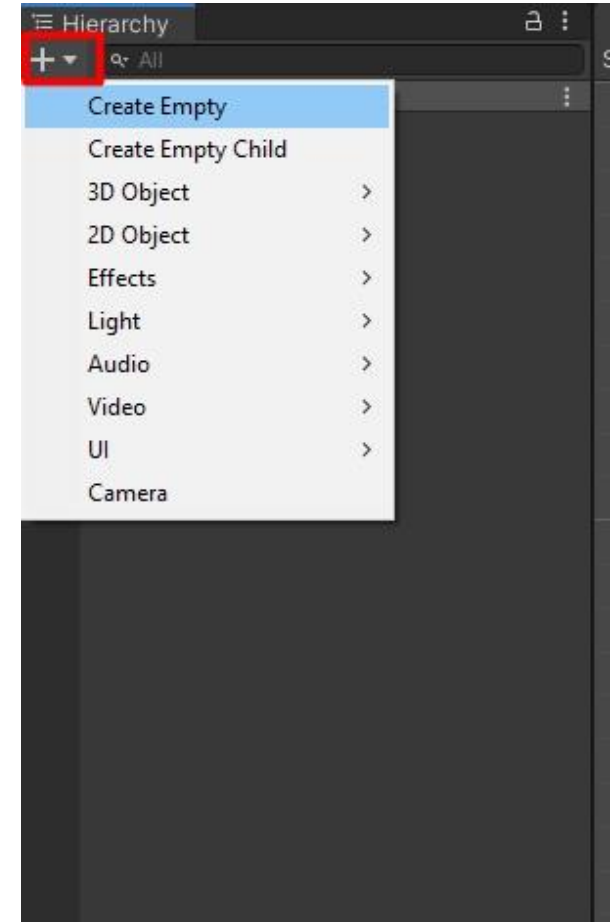
- Hierarchy penceresinde boş bir yere sağ click yaparak Create Empty nesnesi seçilir.
- Diğer GameObject'ler de bu yöntemle eklenebilir.



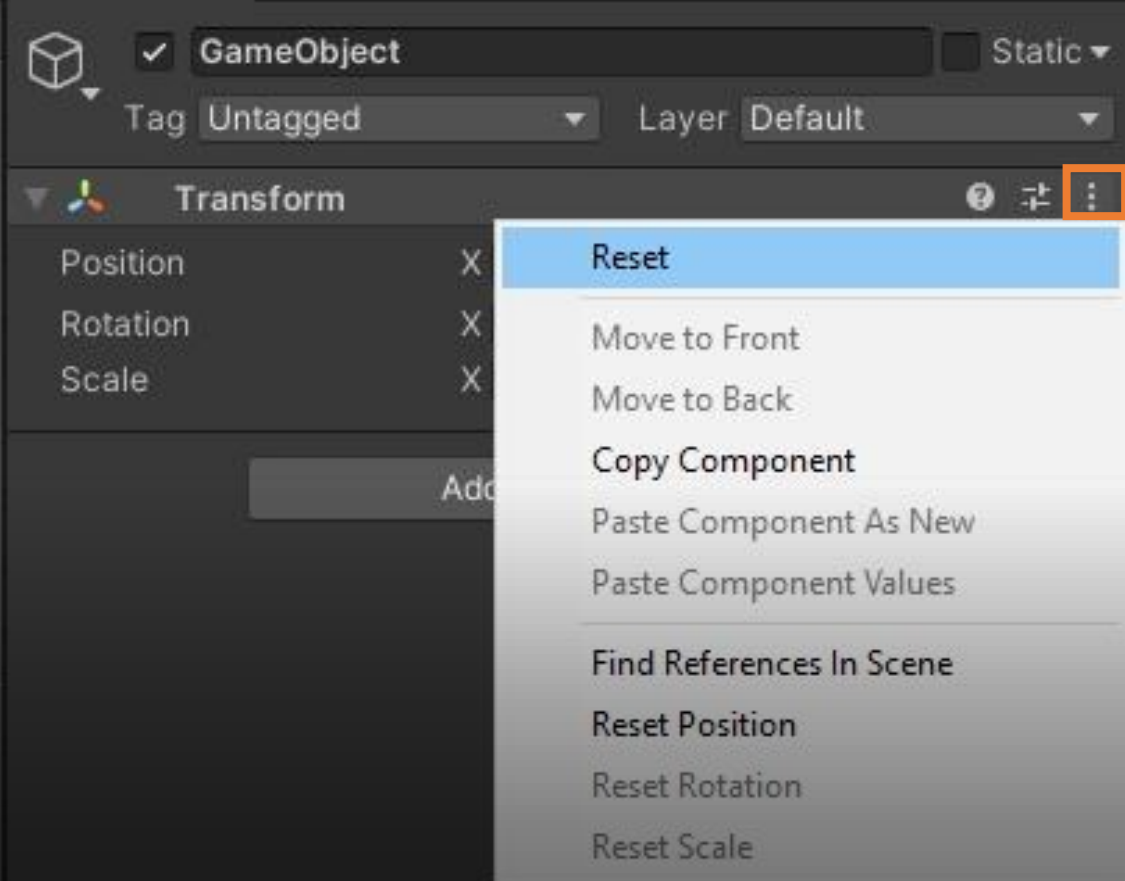
GameObject Ekleme (Ör. Create Empty)

3. Yöntem:

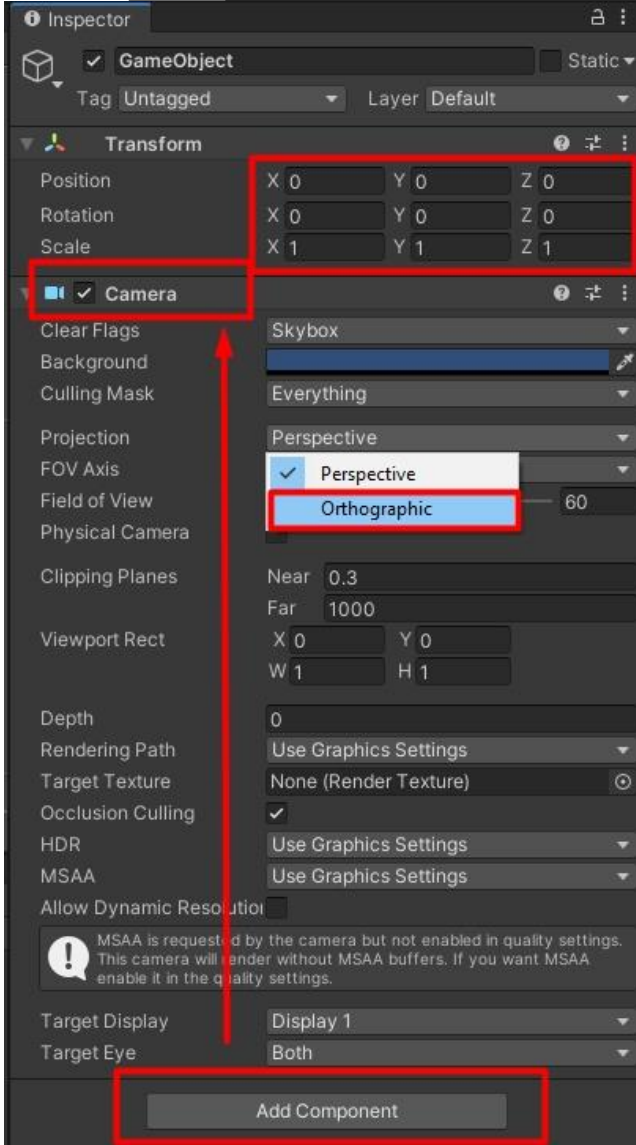
- Hierarchy penceresinde + seçeneğine tıklayarak gelen seçeneklerden Create Empty nesnesi seçilir.
- Diğer nesnelerde bu yöntemle eklenebilir.



GameObject
Transform → Reset



GameObject Özelleştirmek



1. Transform ayarlarını Reset ile sıfırlayarak nesnenin Orjine gelmesini sağladık.
2. Add Component ile Camera Component'ini nesneye ekledik.
3. MainCamera ile özellikleri benzer olsun diye Projection ayarındaki Perspective seçeneğini Orthographic yaptık.

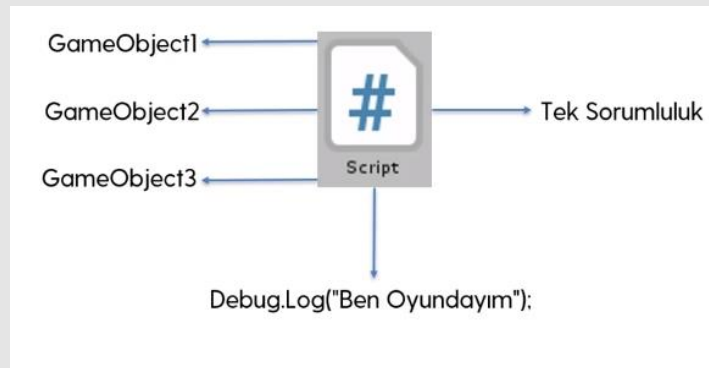
Buradaki ayarlar gerçek kameralarda olan ayarlarla birebir örtüşmektedir.

Scripting (Betik Programlama)

- Script → Programming Language (Programlama Dili)
- Belli görevleri yerine getirmeleri için Script'ler kullanılır.
- Belirli bir görevi yapması için hazırladığımız bileşenleri Component mantığı ile birçok yerde tekrar tekrar kullanabiliriz. Yani bir Component sadece bir oyun objesine ait olmak zorunda değildir. Diğer nesnelere de eklenebilir.

Single Responsibility Principle (SPR)

-Tek Sorumluluk İlkesi-



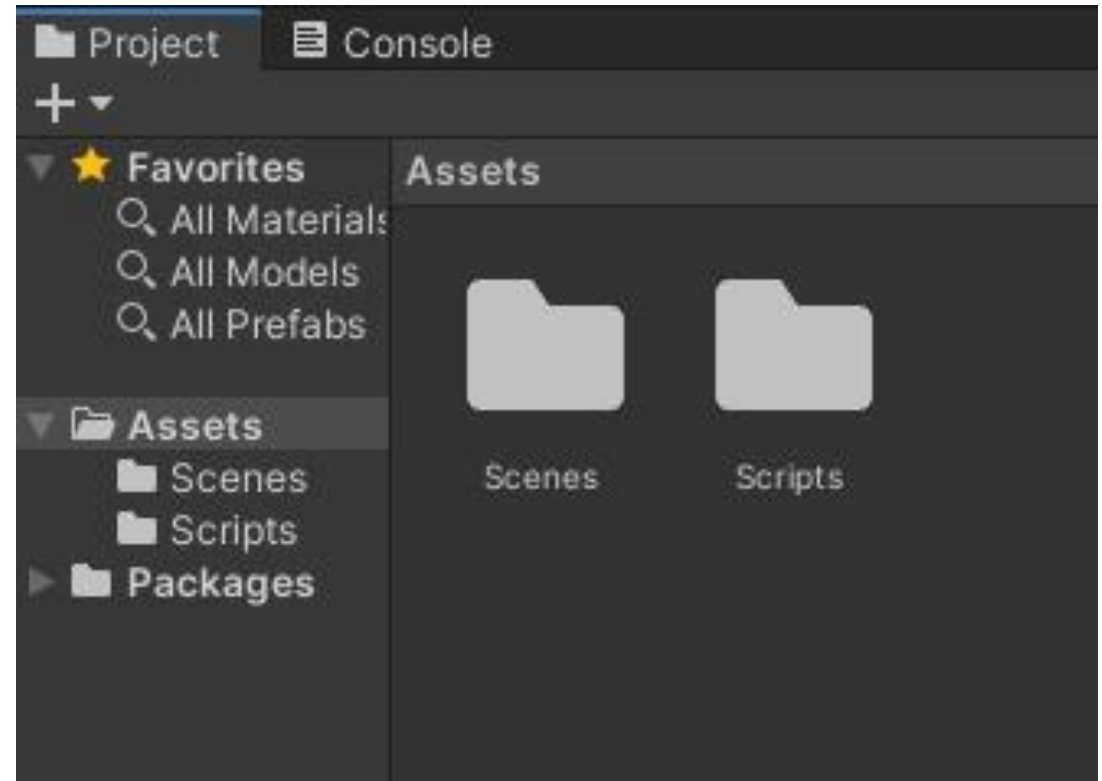
- Hazırladığımız ve Component olarak kullandığımız bir Script'in sadece bir işi yapmasını hedeflemeliyiz.
- Örneğin sahnede olan oyun objelerinin her biri için Console'a «Ben Oyundayım» Log mesajının yazdırılmasını sadece bir Script hazırlayıp, bu Component'i tüm oyun objelerine ekleyerek sağlayabiliriz.
- Aynı şekilde bu sorumluluğa sahip olan Script'e başka bir sorumluluk vermemeliyiz.

«Ben buradayım» Log Mesajını Veren Script'i Oluşturalım

1. Öncelikle Script'lerimiz için Scripts klasörünü oluşturalım.

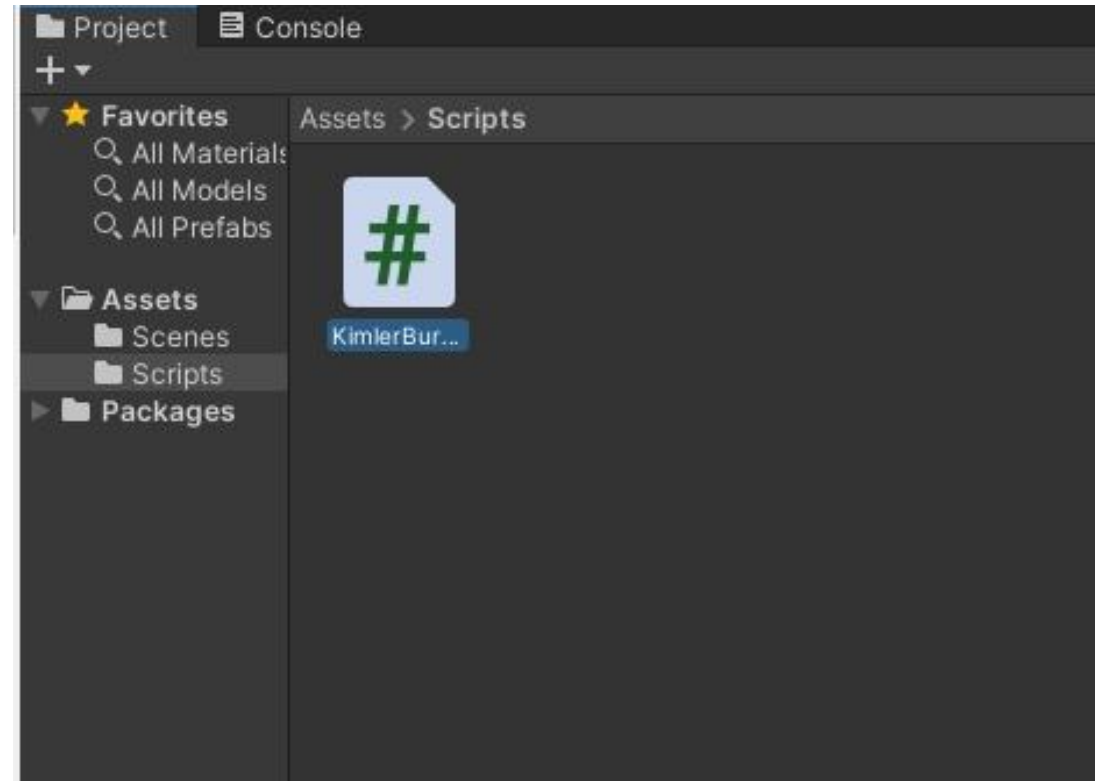
Bunun için;

- 1. yöntem Project yerleşimindeki Assets klasöründe sağ tıklayıp Create → Folder seçeneğini seçmektir.
- 2. yöntem Assets menüsünde yer alan Create → Folder seçeneğini seçmektir.



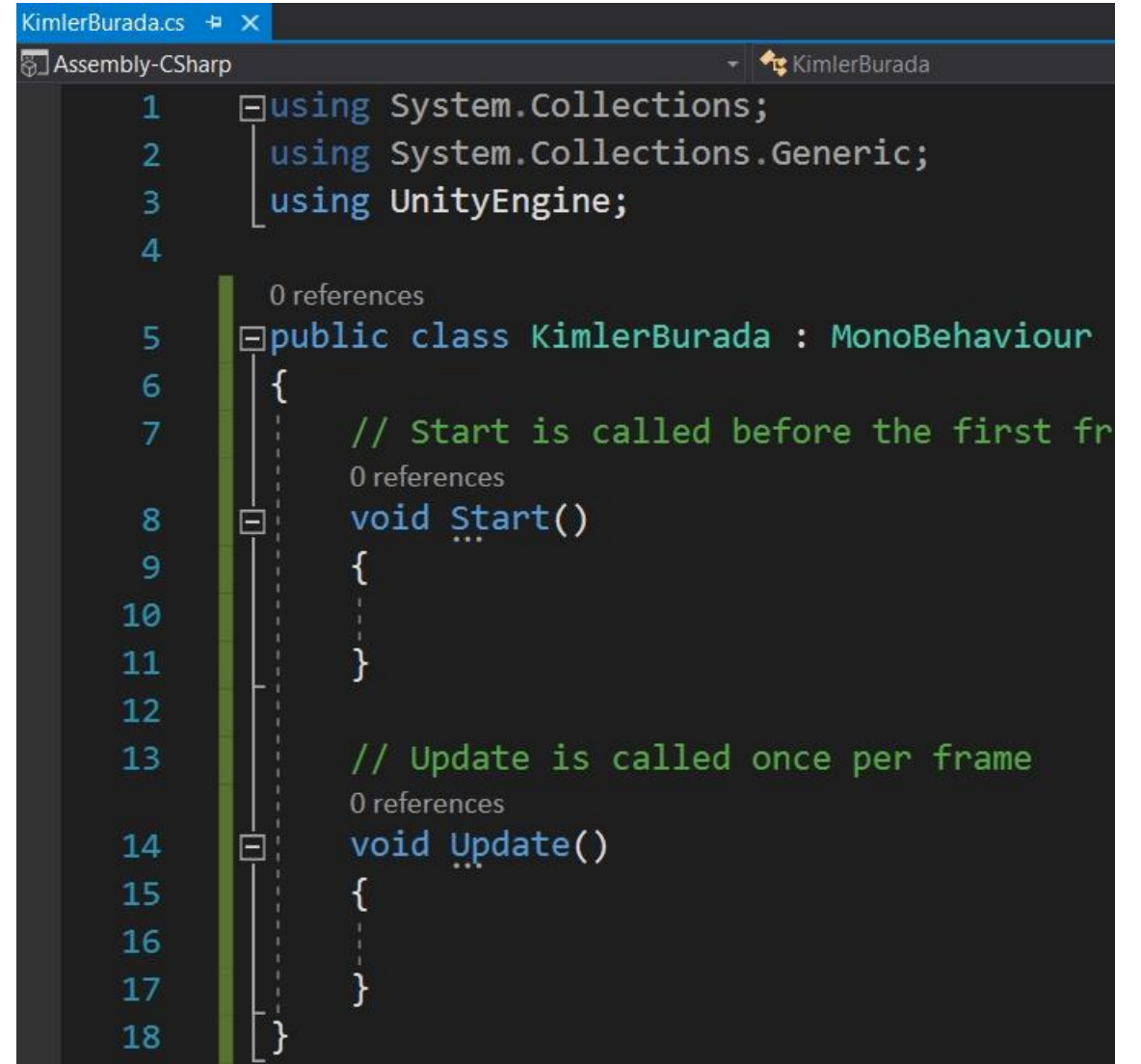
«Ben buradayım» Log Mesajını Veren Script'i Oluşturalım

2. Scripts klasörüne çift tıklayarak klasörün içerisine girelim.
3. Assets klasöründe sağ tıklayıp Create → C# Script seçeneğini seçelim. Script'e bir isim verelim. Ör. KimlerBurada



«Ben buradayım» Log Mesajını Veren Script'i Oluşturalım

4. Script'e çift tıklayarak Visual Studio'nun açılmasını sağlayalım.



```
KimlerBurada.cs
Assembly-CSharp
KimlerBurada

1  using System.Collections;
2      using System.Collections.Generic;
3      using UnityEngine;
4
5  public class KimlerBurada : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10
11      }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
```

«Ben buradayım» Log Mesajını Veren Script'i Oluşturalım

5. Gelen Visual Studio ekranında Start metoduna aşağıdaki kodu ekleyelim.

```
void Start()  
{  
    //Sahneде var olan oyun  
objeleri Ben Buradayım desin  
    Debug.Log("Ben buradayım");  
}
```

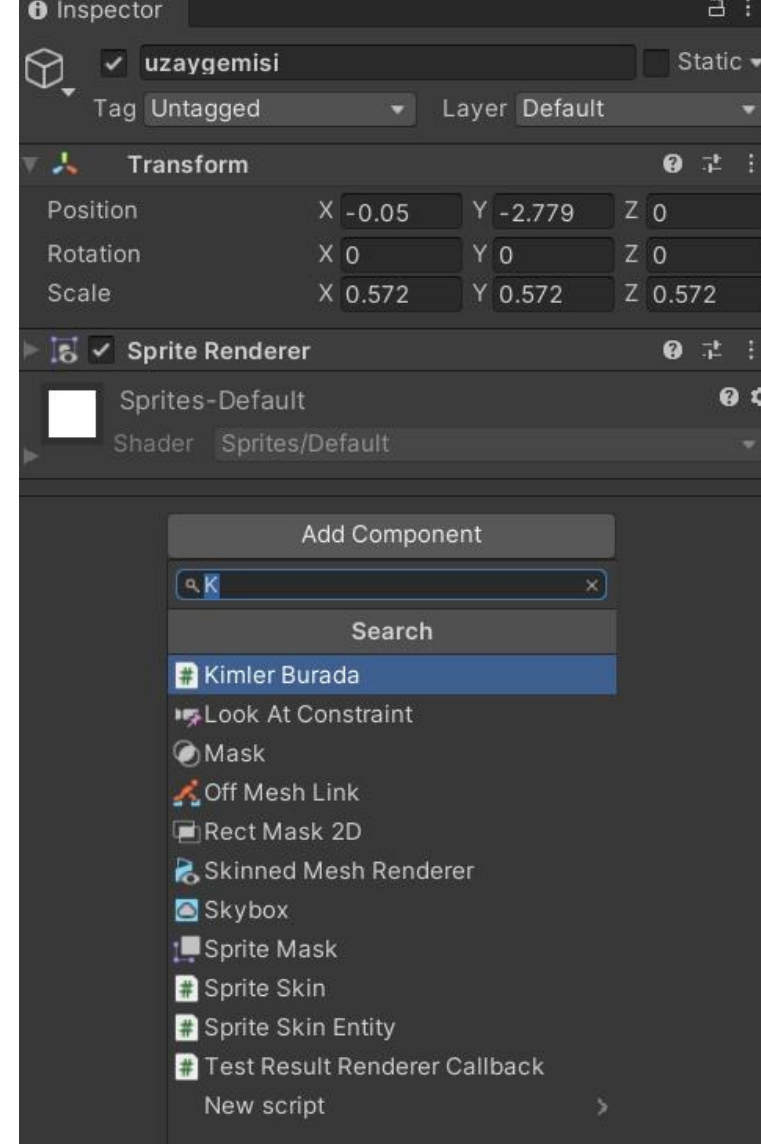
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class KimlerBurada : MonoBehaviour
6 {
7     void Start()
8     {
9         //Sahnedeki var olan oyun objeleri Ben Buradayım desin
10        Debug.Log("Ben buradayım");
11    }
12 }
```

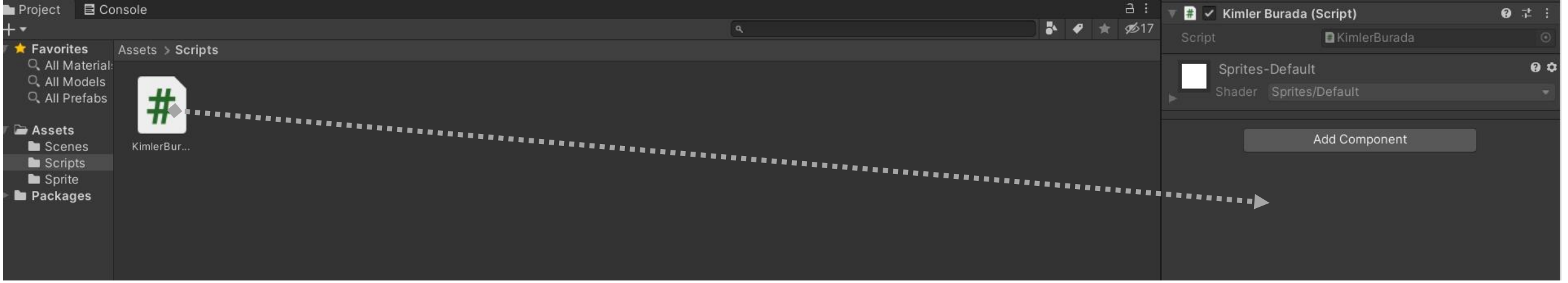
«Ben Oyundayım»
Log Mesajını Veren
Script'i Oluşturalım

Oluşturmuş olduğumuz Script'in son halini kaydedelim.
(CTRL + S)

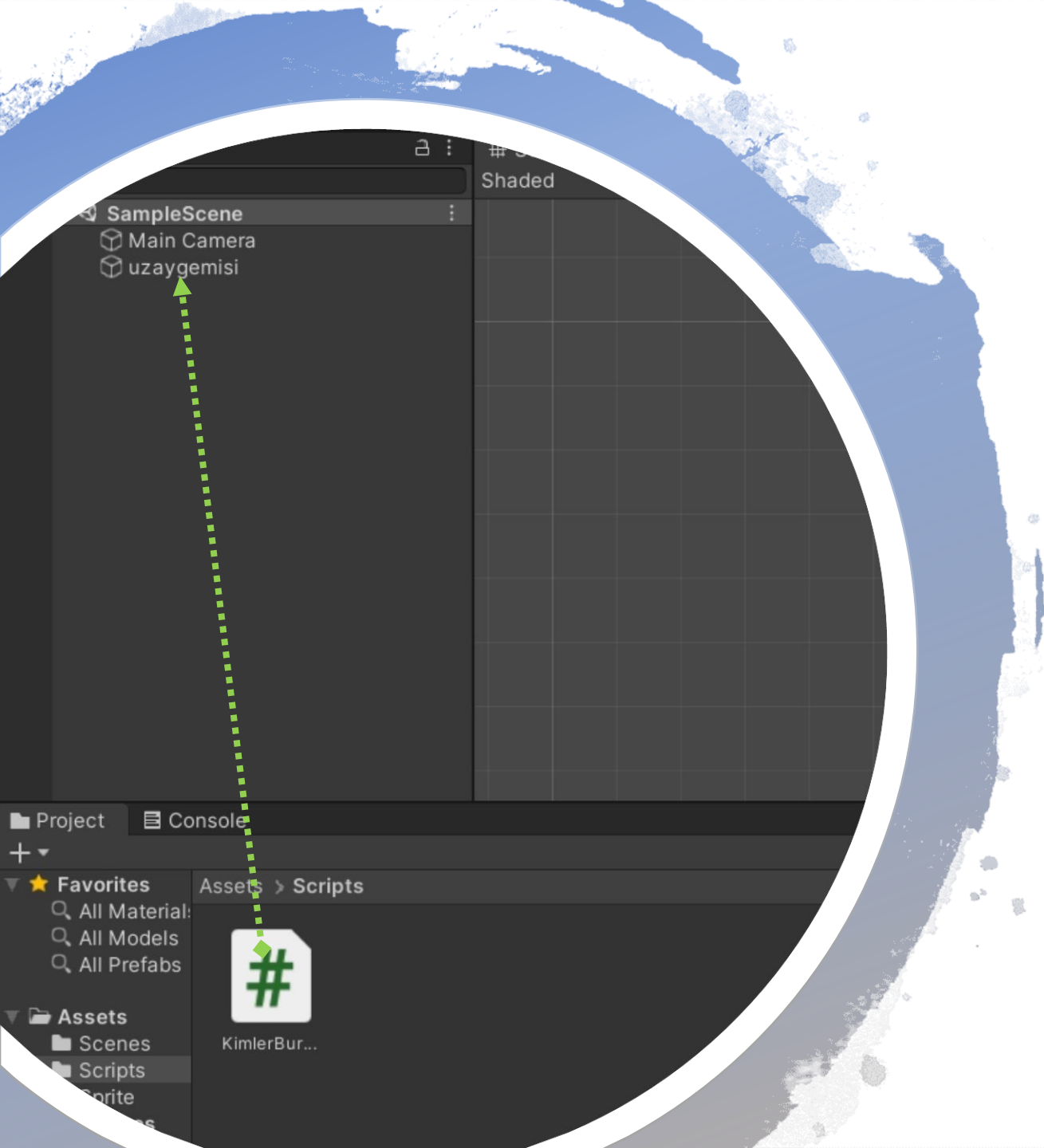
Oyun objemize Component Ekleme ÖR: Sprite objemize «Ben Oyundayım» Log Mesajını Veren Script'i Ekleyelim

Yöntem-1: Sprite nesnemiz aktif
iken Add Component ile Script'imiz
üzerine tıklanır.



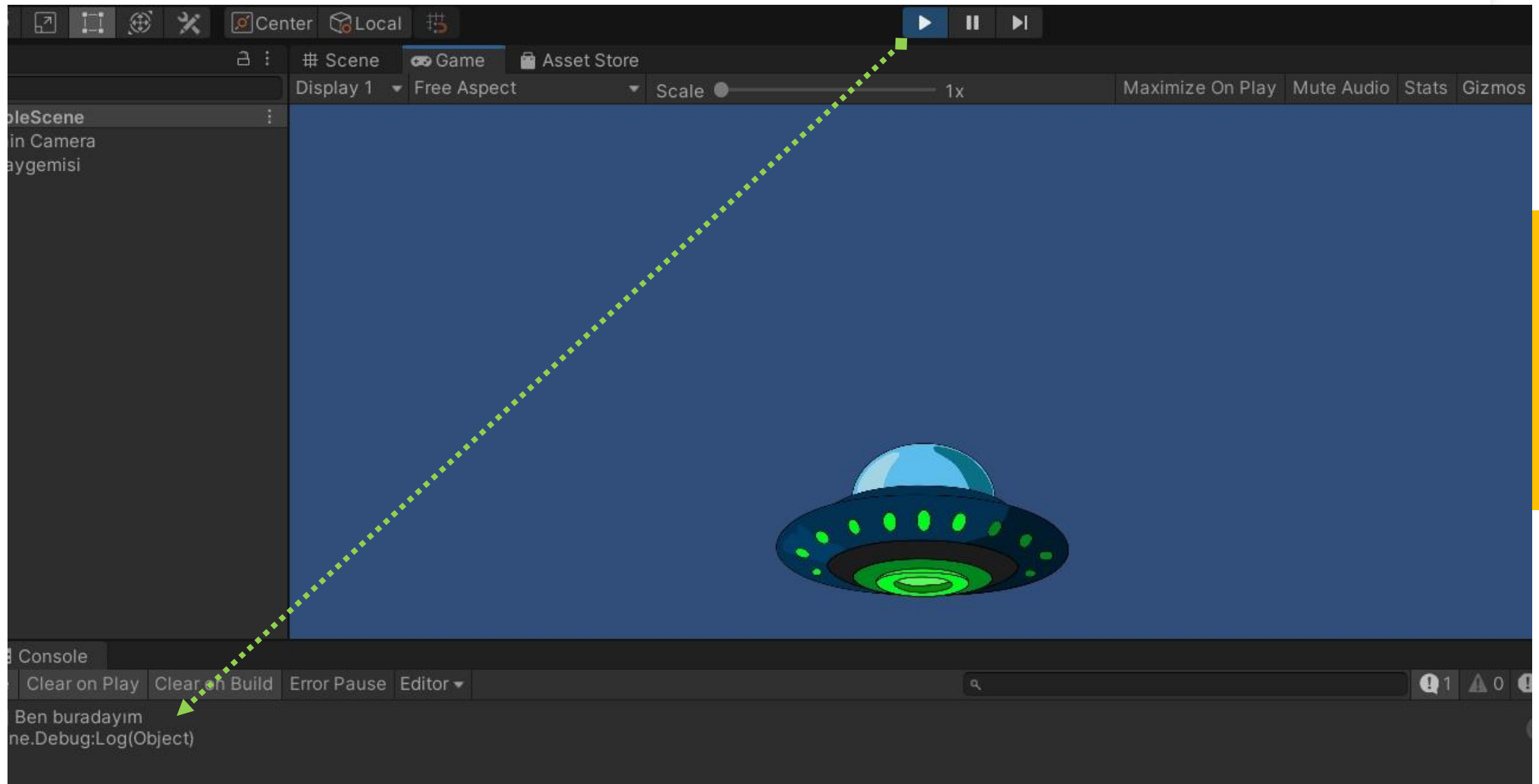


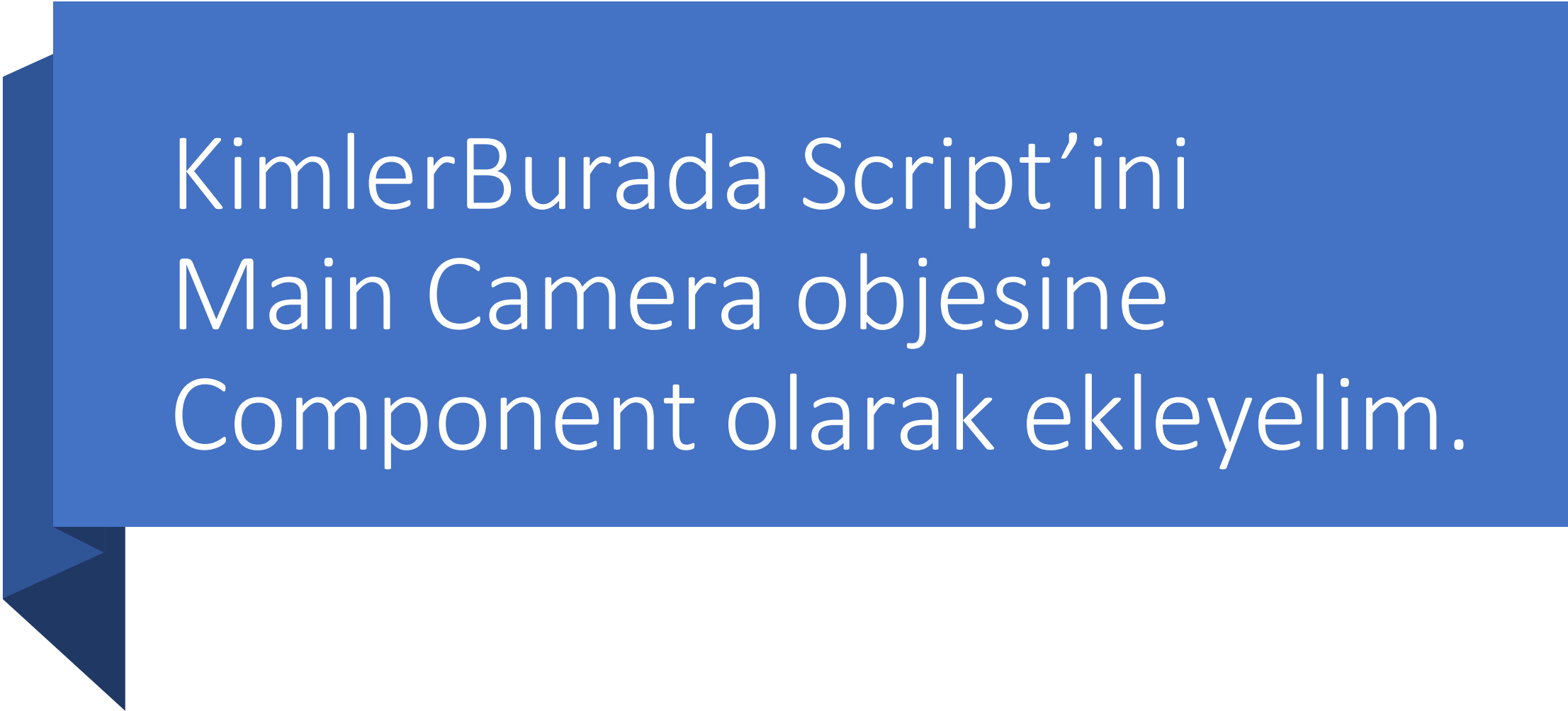
Oyun objemize Component Eklemek
ÖR: Sprite objemize «Ben Oyundayım» Log Mesajını Veren Scripti Ekleyelim
Yöntem-2: Sprite nesnemiz aktif iken Script'imiz sürüklenerek Inspector kısmına bırakılır.



Yöntem-3: Script'imiz sürüklenerek Hierarchy'deki oyun objemizin üzerine bırakılır.

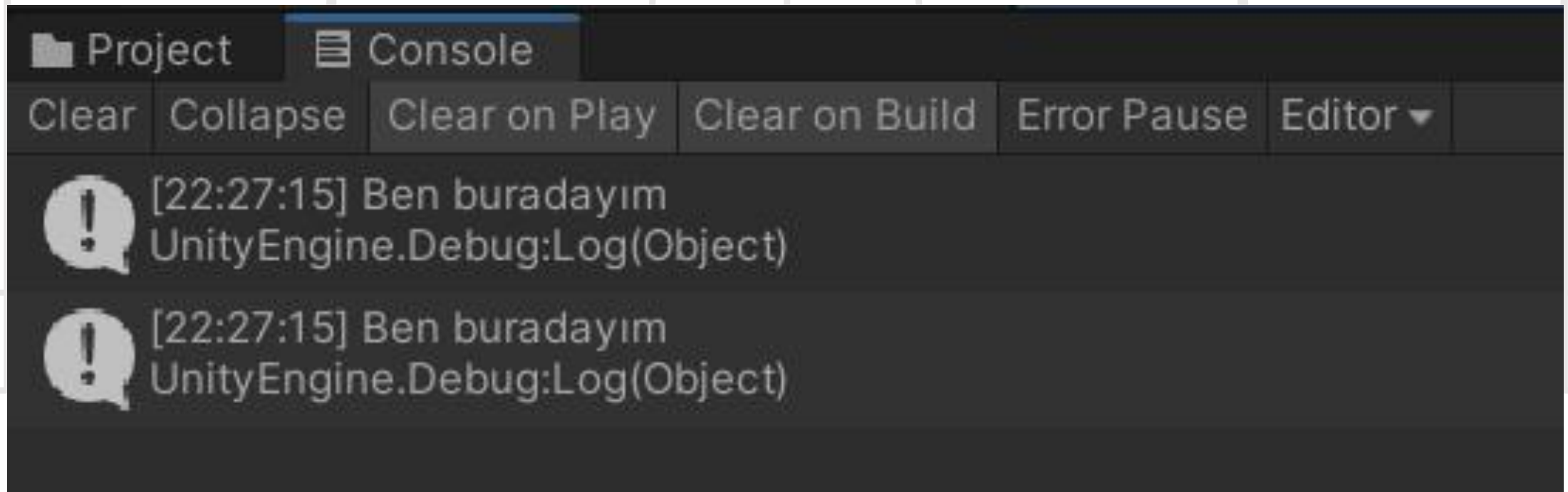
Oyun objemize Component Ekleme
ÖR: Sprite objemize «Ben Oyundayım» Log Mesajını Veren Scripti Ekleyelim





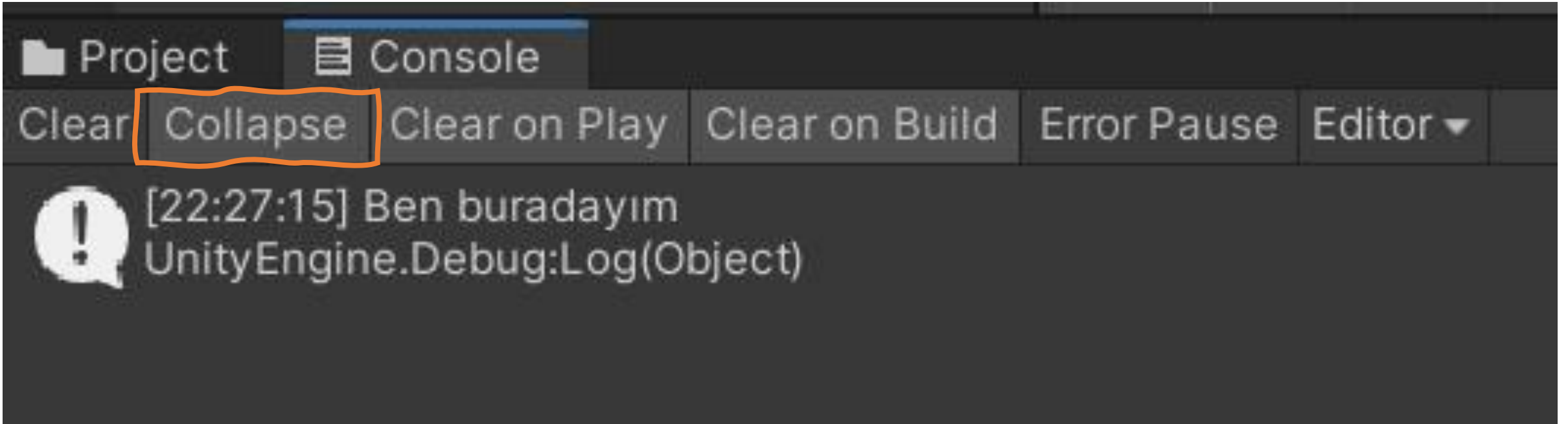
KimlerBurada Script'ini
Main Camera objesine
Component olarak ekleyelim.

Oyun objelerimizin ikisi için «Ben buradayım» Log mesajı oluştu. Bu Log mesajlarının üzerine tıklayarak ilgili Script'imizi görebiliriz. Log üzerine çift tıklayarak Visual Studio'da Script'in ilgili kod satırına gidebiliriz.



Collapse

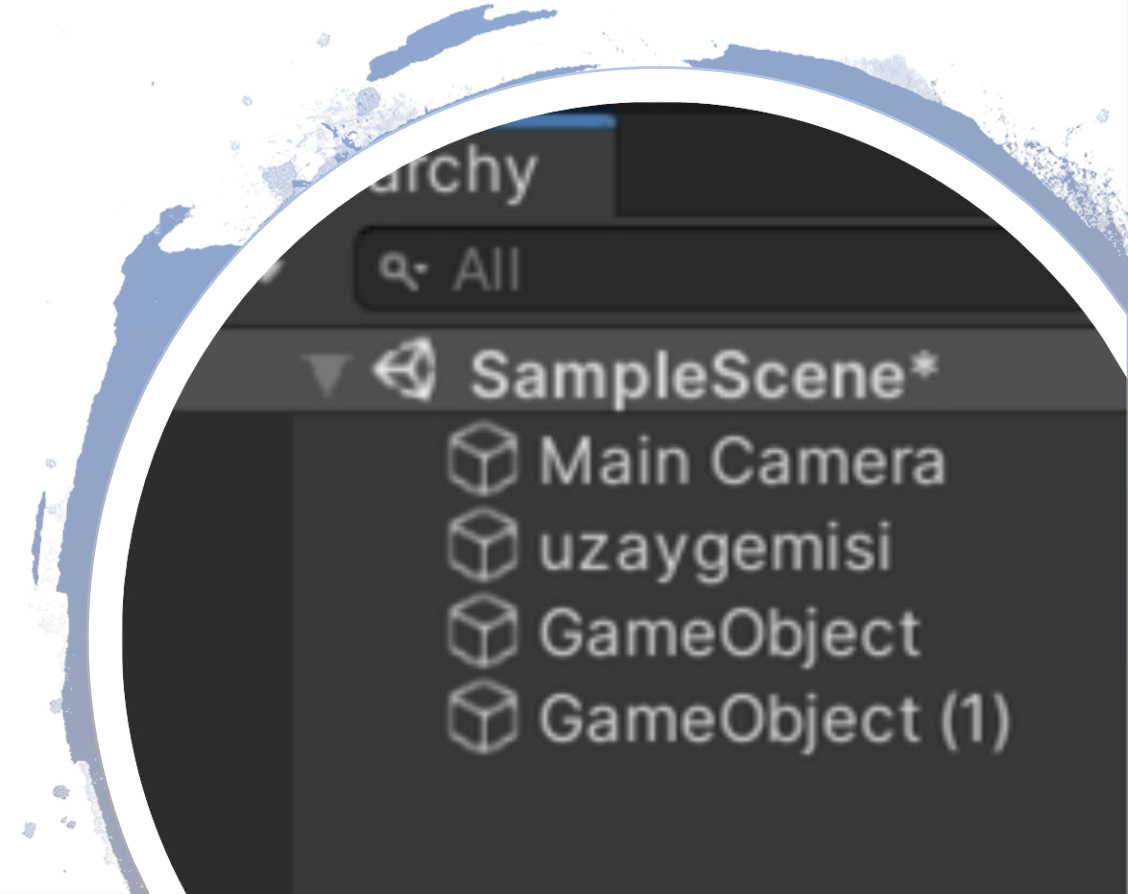
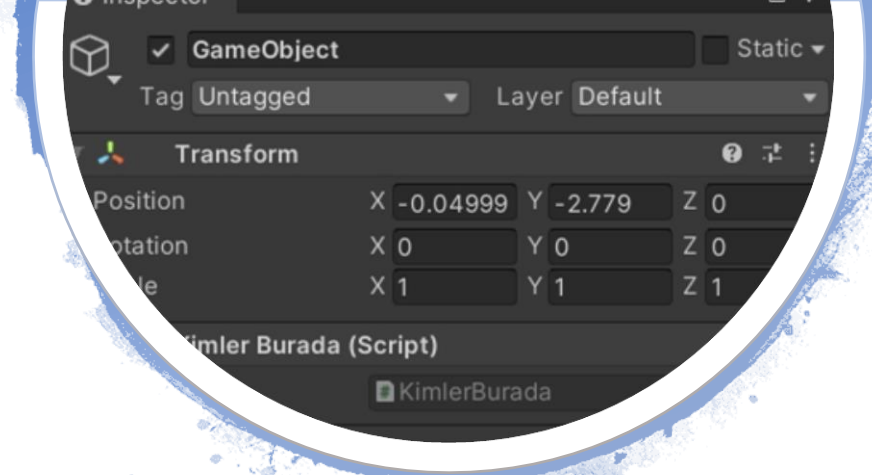
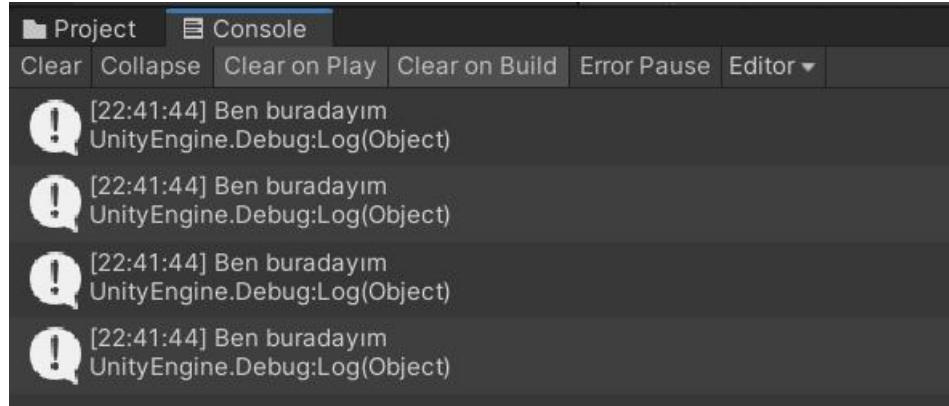
Collapse seçeneği aktif yapılarak aynı mesajlar tek satırda gösterilebilir.

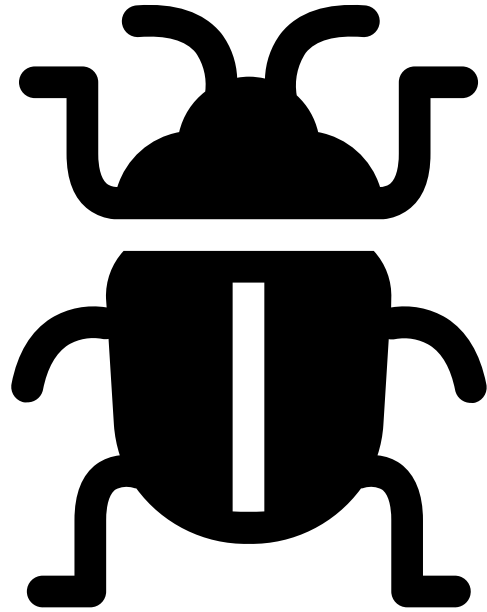


Uzay Savaşı isimli projede sahneye 2 adet boş Oyun
Objesi ekleyip KimlerBurada isimli Script'i
Component olarak boş oyun objelerine ekleyiniz ve
konsolda 4 defa "Ben buradayım" yazmasını
sağlayınız.

Egzersiz

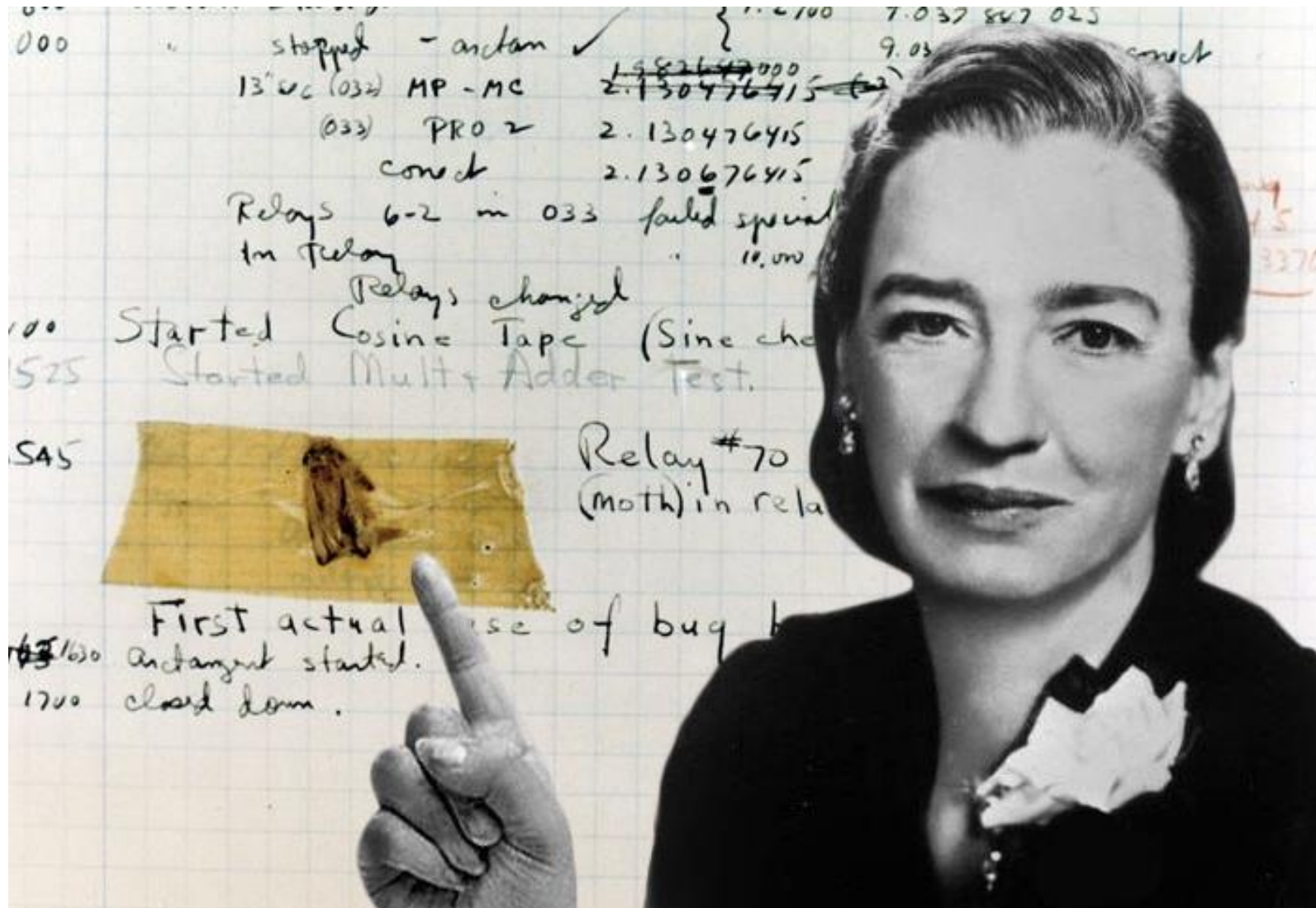
Egzersiz - Çözüm





BUG

- Bug = Böcek
- Debugging = Böcekten arınmak
- Yazılımdaki hatalara Bug denir.
- Neden hatalara bug yani böcek denildiğini düşündünüz mü?



Grace Hopper
9 Eylül 1974
15:45


1974'te MARK II bilgisayarında F panelinin 70 nolu rölesinde sıkışan gerçek bir böcek ile hata oluşur.

Bu olaydan sonra bilgisayarda oluşan tüm hatalara Bug (böcek) denilmektedir.

9/9

0800 Antan started { 1.27
1000 " stopped - antan ✓
1300 (032) MP - MC 1.58267000
2.13047641
(033) PRO 2 2.130476415
conv 2.130676415
Relays 6-2 in 033 failed spec
in relay 10.000
Relays changed

1100 Started Cosine Tape (Sine ch.
1525 Started Mult + Adder Test.

1545  Relay #70
(moth) in rel

First actual case of bug be

1630 Antan started.
1700 closed down.



Debugging (Hata Ayıklama)

- Programımızda çalışma esnasında oluşan hatalar (Syntax Error) dışında da hatalar olabilir.
- Yani programımız çalışma anında hata vermiyorsa ama elde etmek istediğimizi sonucu da getirmiyorsa oluşan bu hatalara mantıksal hatalar diyebiliriz.
- Örneğin 7 ile 3 sayısının toplamını hesaplattırmaya çalışırken bize farkları olan 4 değerini vermesi gibi.

Debugging (Hata Ayıklama)

- Hatanın nerede olabileceği tespit edilir.
- Hatanın gerçekleşebileceği bölüm minimuma indirgenir.



Kendi yazılımımızda bu
böceklerden nasıl kurtuluruz?

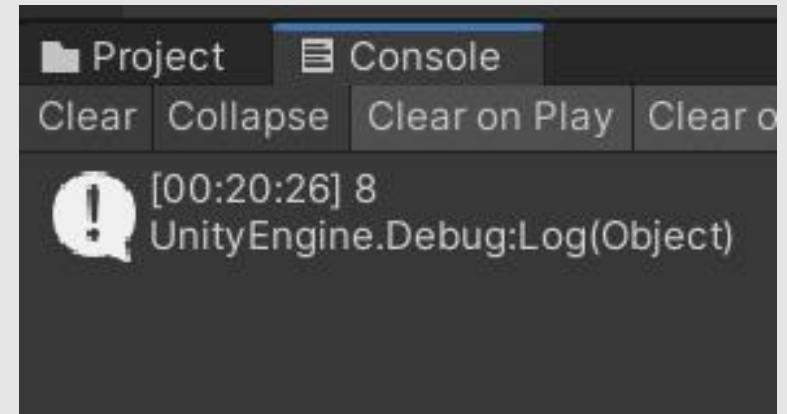
Script
içerisine
yandaki kodu
yazalım.

```
void Start()  
{  
    //7 ile 2 sayısının ortalamasını  
    hesaplattırmaya çalışalım  
    //Sonucu Log ile Console ekranına yazdıralım  
    double sonuc = 7 + 2 / 2;  
    Debug.Log(sonuc.ToString());  
}
```

Aslında bu kodda elde etmeye çalıştığımız 7 ile 2 sayılarının ortalamalarını hesaplattırmaktı.

Ama bu işlem bize sonuç olarak 4.5 yerine 8 değerini verecektir.

Şimdi bu hatanın nerede olduğunu bulurken hata yakalama yöntemlerini uygulayacağız.



Breakpoint



Location: Hesaplama.cs, line 12 character 9 ('Hesaplama.Start()')

```
// Start is called before the first frame update
```

```
0 references
```

```
void Start()
```

```
{
```

```
//7 ile 2 sayısının ortalamasını hesaplattırmaya çalışalım
```

```
//Sonucu Log ile Console ekranına yazdıralım
```

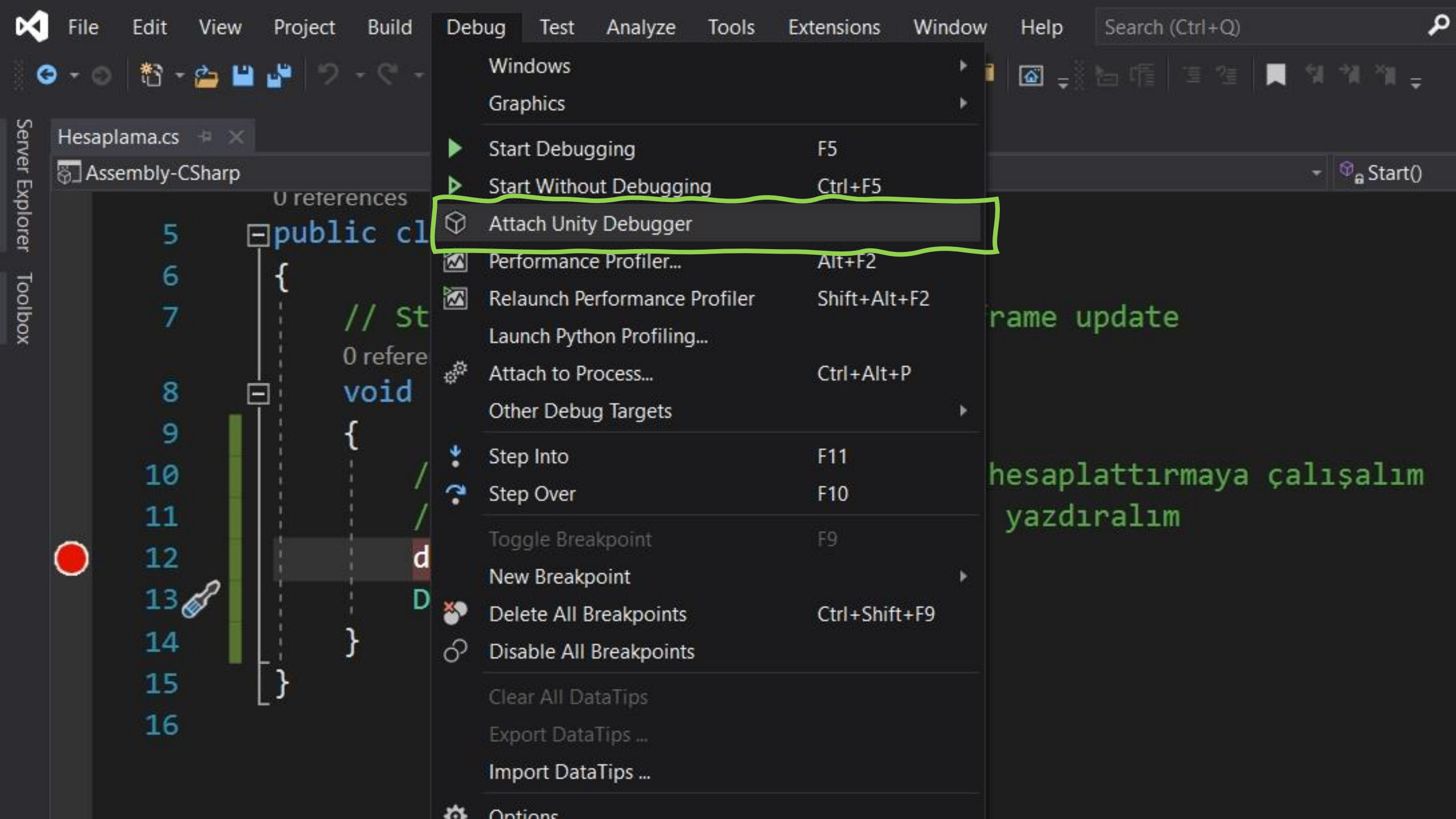
```
double sonuc = 7 + 2 / 2;
```

```
sonuc.ToString());
```

```
}
```

Debugging (Hata Ayıklama)

- Hatanın hangi Script'te olduğunu biliyoruz.
- Hatanın olduğunu düşündüğümüz Script'i Visual Studio ile açıyoruz.
- Hatanın olabileceği satırın başına gelip Mouse sol tuşu ile bir kere tıklayarak Breakpoint ekliyoruz.



Windows

Graphics

▶ Start Debugging F5

▶ Start Without Debugging Ctrl+F5

📦 Attach Unity Debugger

📊 Performance Profiler... Alt+F2

📊 Relaunch Performance Profiler Shift+Alt+F2

⚙️ Launch Python Profiling...

⚙️ Attach to Process... Ctrl+Alt+P

Other Debug Targets ▶

⏏ Step Into F11

⏏ Step Over F10

🚩 Toggle Breakpoint F9

New Breakpoint ▶

🚫 Delete All Breakpoints Ctrl+Shift+F9

🔒 Disable All Breakpoints

Clear All DataTips

Export DataTips ...

Import DataTips ...

⚙️ Options

Hesaplama.cs

Assembly-CSharp

0 references

5 public class

6 {

7 // static

8 void

9 {

10 //

11 //

12 //

13 //

14 }

15 }

16 }

frame update

hesaplattırmaya çalışalım
yazdıralım

Hesaplama.cs Assembly-CSharp Hesaplama Start()

0 references

```
5 public class Hesaplama : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10         //7 ile 2 sayısının ortalamasını hesaplatırmaya çalışalım
11         //Sonucu Log ile Console ekrana yazalım
12         double sonuc = 7 + 2 / 2;
13         Debug.Log(sonuc.ToString());
14     }
15 }
16
```

Select Unity Instance

Filter...

Project	Machine	Type	Port	Information
Unity2DOrneK	GOZDEMIHRAN	Editor	56276	PID:39276

Input IP Refresh OK Cancel

Asset Store

Play

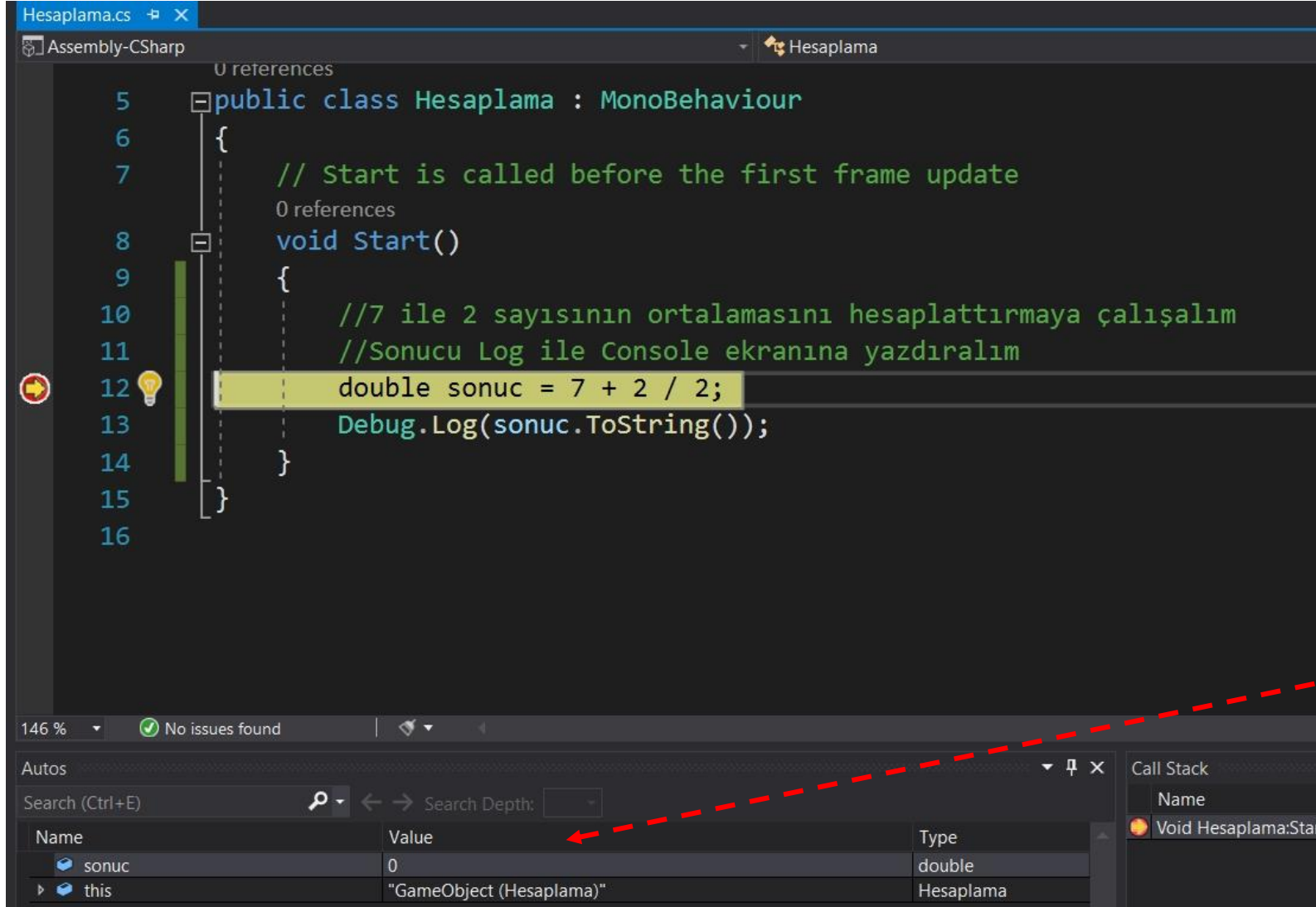


Gizmos

All



Unity üzerinden Play ile
oyun çalıştırılır.



```
5 public class Hesaplama : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10         //7 ile 2 sayısının ortalamasını hesaplattırmaya çalışalım
11         //Sonucu Log ile Console ekranına yazdıralım
12         double sonuc = 7 + 2 / 2;
13         Debug.Log(sonuc.ToString());
14     }
15 }
16
```

Autos

Search (Ctrl+E) Search Depth: [dropdown]

Name	Value	Type
sonuc	0	double
this	"GameObject (Hesaplama)"	Hesaplama

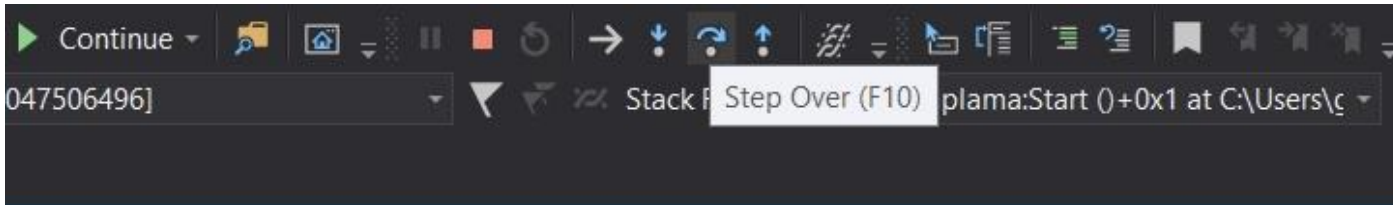
Call Stack

Name
Void Hesaplama:Start

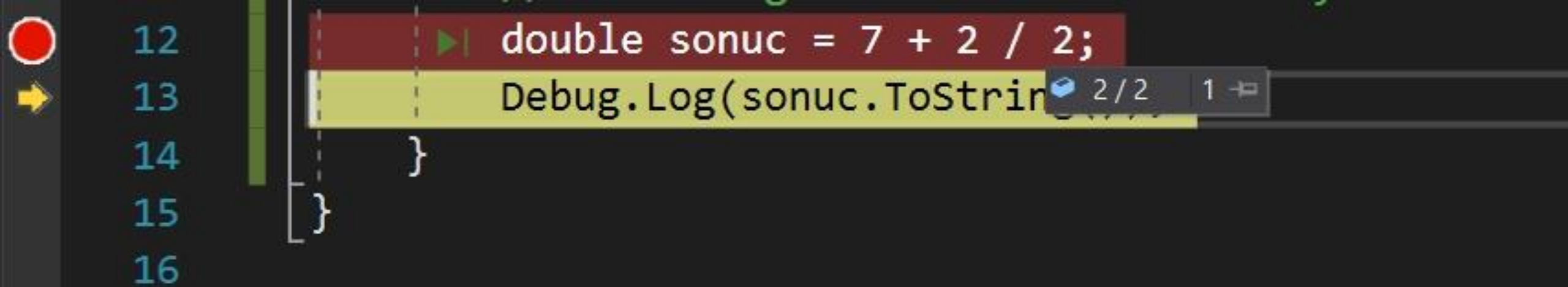
Program
Breakpoint
koyduğumuz
satıra kadar
çalıştı ve bu
satırda
durdu.

Aşağıdaki
penceredeki
değerleri
inceleyelim.

Debugging (Hata Ayıklama)



- Step Over (F10) ile programın bir sonraki adıma geçmesi sağlanır. Bunun için 3 seçenek vardır.
 - Debug → Step Over
 - Debugging (Hata Ayıkla) → Step Over (Üzerinden Adımla)
 - F10
- Değişkenlerin aldığı değerler kontrol edilir.
- Böylece hata tespit edilmeye çalışılır.



Visual Studio code editor showing a C# snippet. Line 12: `double sonuc = 7 + 2 / 2;` (highlighted in red). Line 13: `Debug.Log(sonuc.ToString());` (highlighted in yellow). A breakpoint is set on line 12. The debug console on the right shows the output: `2/2 1`.

146 % No issues found

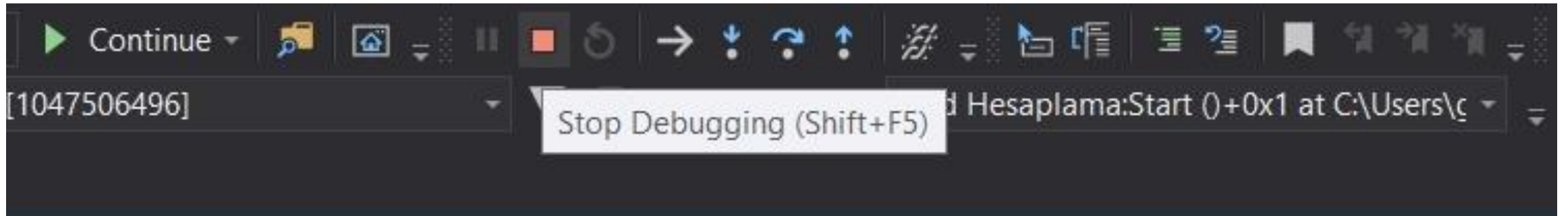
Debugging (Hata Ayıklama)

Search (Ctrl+E) Search Depth:

Name	Value	Type
sonuc	8	double
this	"GameObject (Hesaplama)"	Hesaplama

Debugging (Hata Ayıklama)

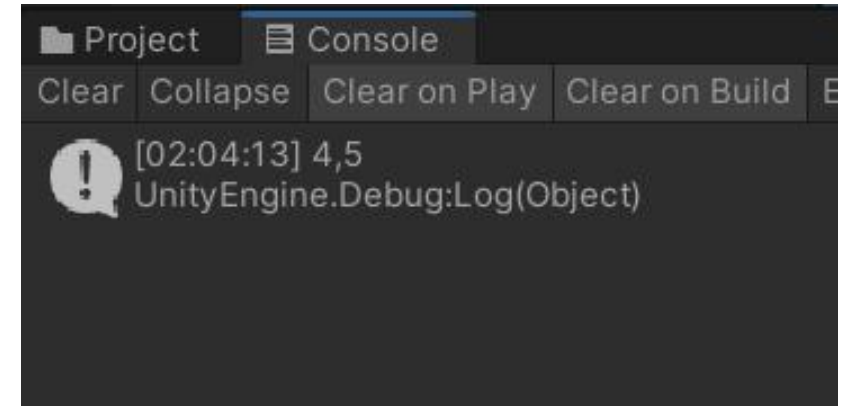
Hata Ayıklamayı sonlandırmak için Stop Debugging (Shift + F5) yapılmalıdır.



Script
içerisine
yandaki kodu
yazalım.

```
void Start()
{
    //7 ile 2 sayısının ortalamasını
    hesaplattırmaya çalışalım
    //Sonucu Log ile Console ekranına yazdıralım
    double sonuc = Convert.ToDouble(7 + 2) / 2;
    Debug.Log(sonuc.ToString());
}
```

Yapmış olduğumuz mantık hatasını düzelterek tekrar projeyi kaydedip çalıştırdığımızda karşımıza doğru sonuç olarak 4.5 değeri gelmektedir.



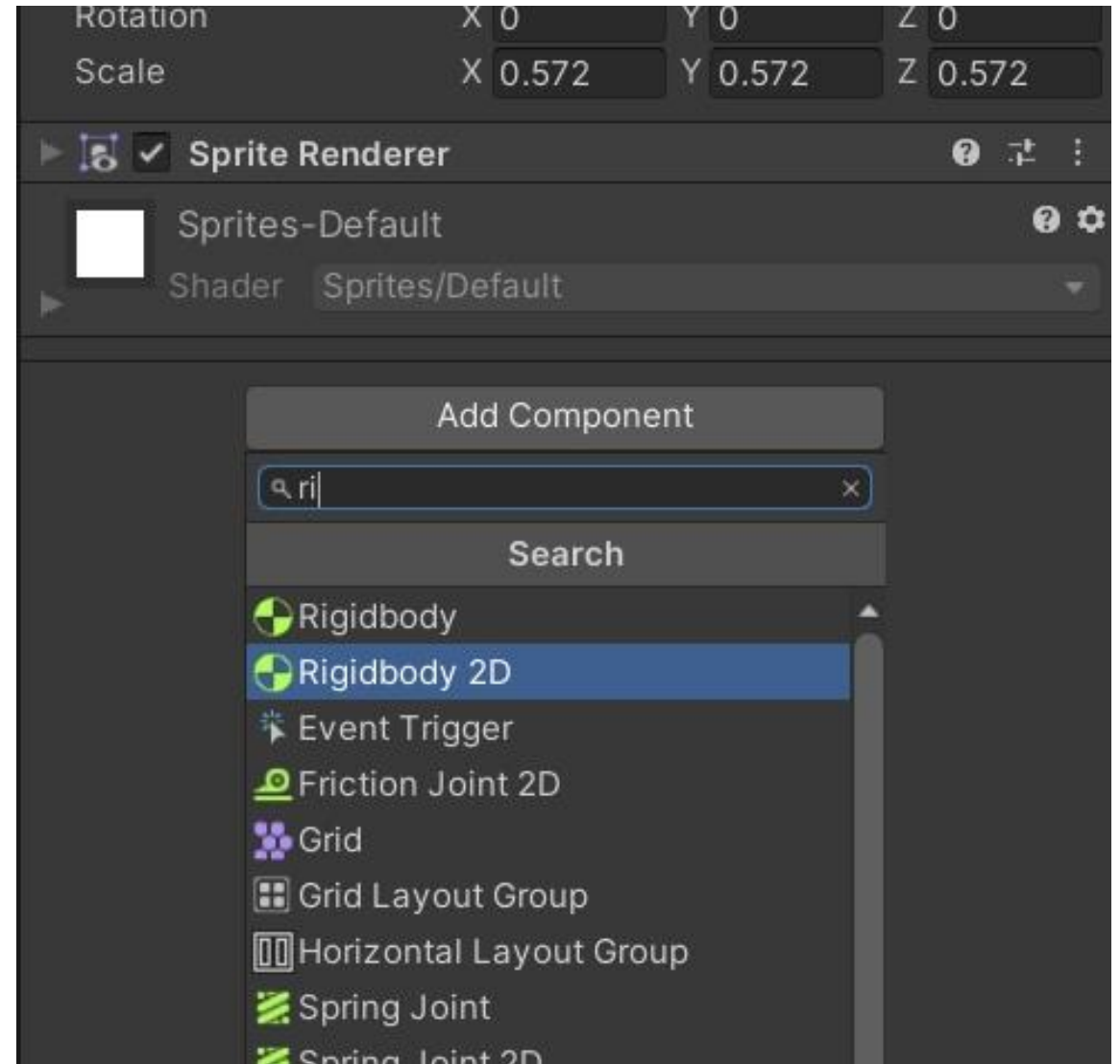
Fizik Bileşenleri Rigidbody

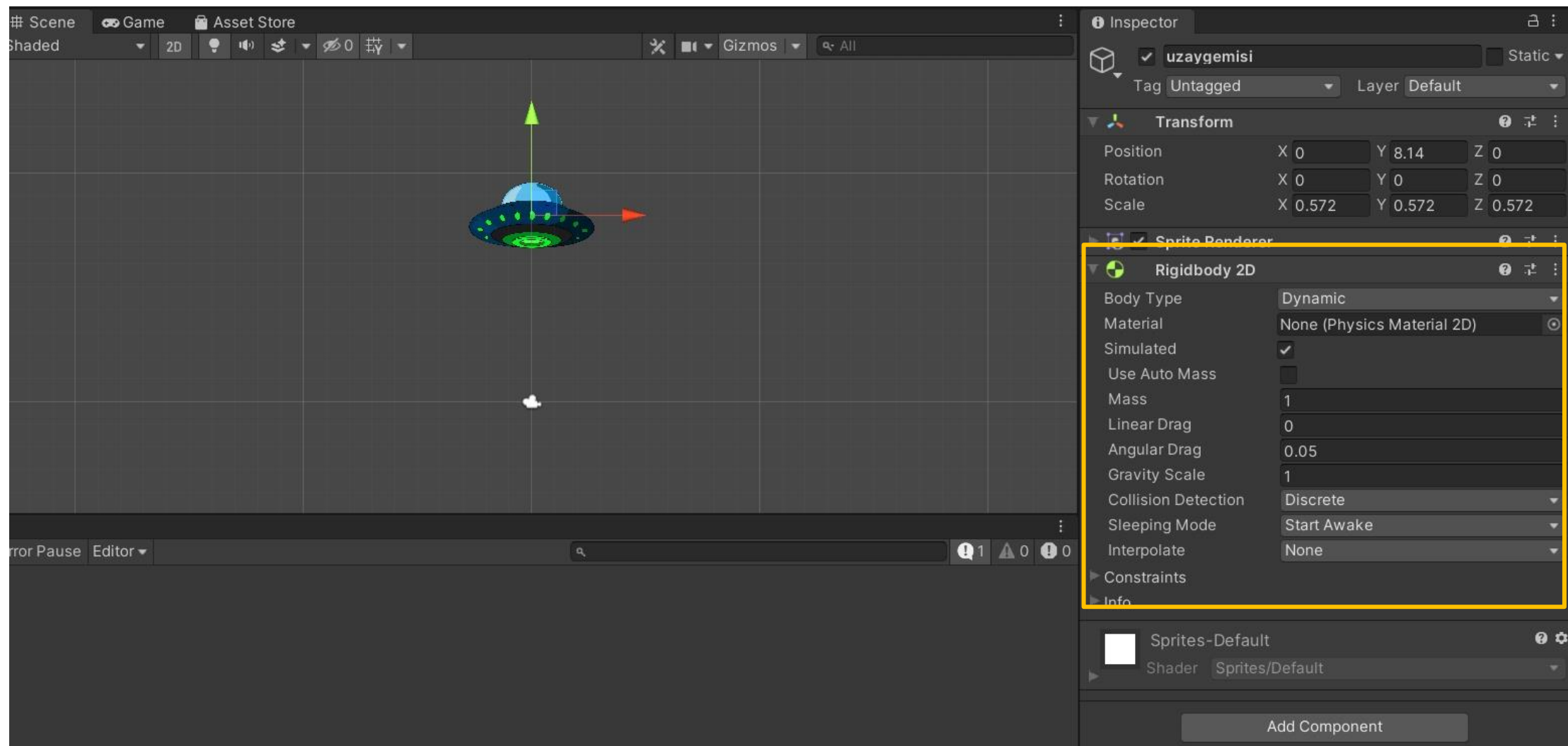
- Unity ortamında 2D ve 3D ortamlar için hazırlanmış ve objelerimize Component olarak ekleyebileceğimiz bir takım fizik bileşenleri vardır.



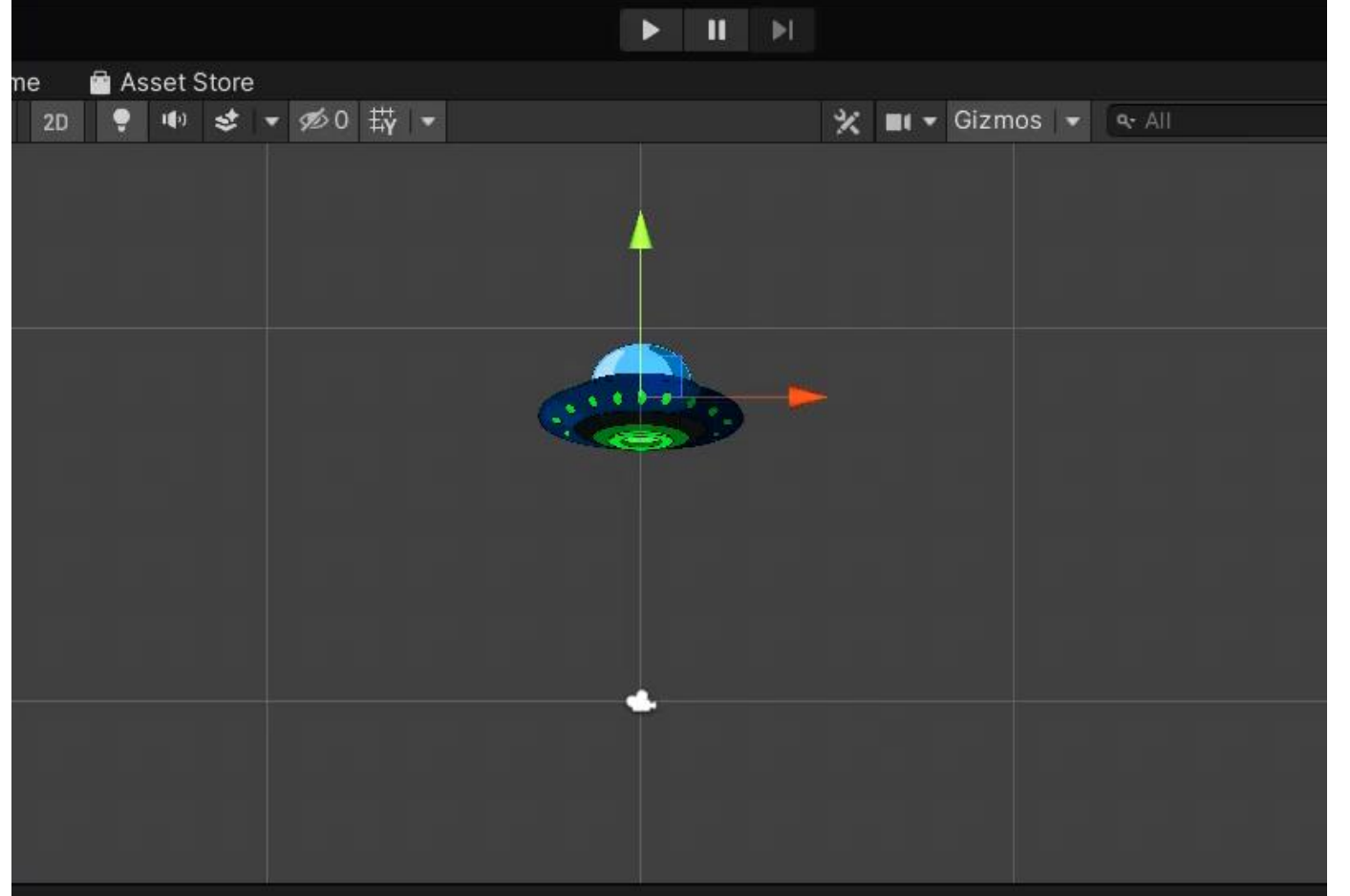
Rigidbody

- Oyun dünyasında bir oyun objesine fizik kurallarını işletebilmemiz için ihtiyacımız olan Component Rigidbody'dir.
- 2D projemizde Oyun objemize (GameObject) fizik eklemek için;
- Add Component → Physics 2D → Rigidbody 2D seçeneği seçilir.





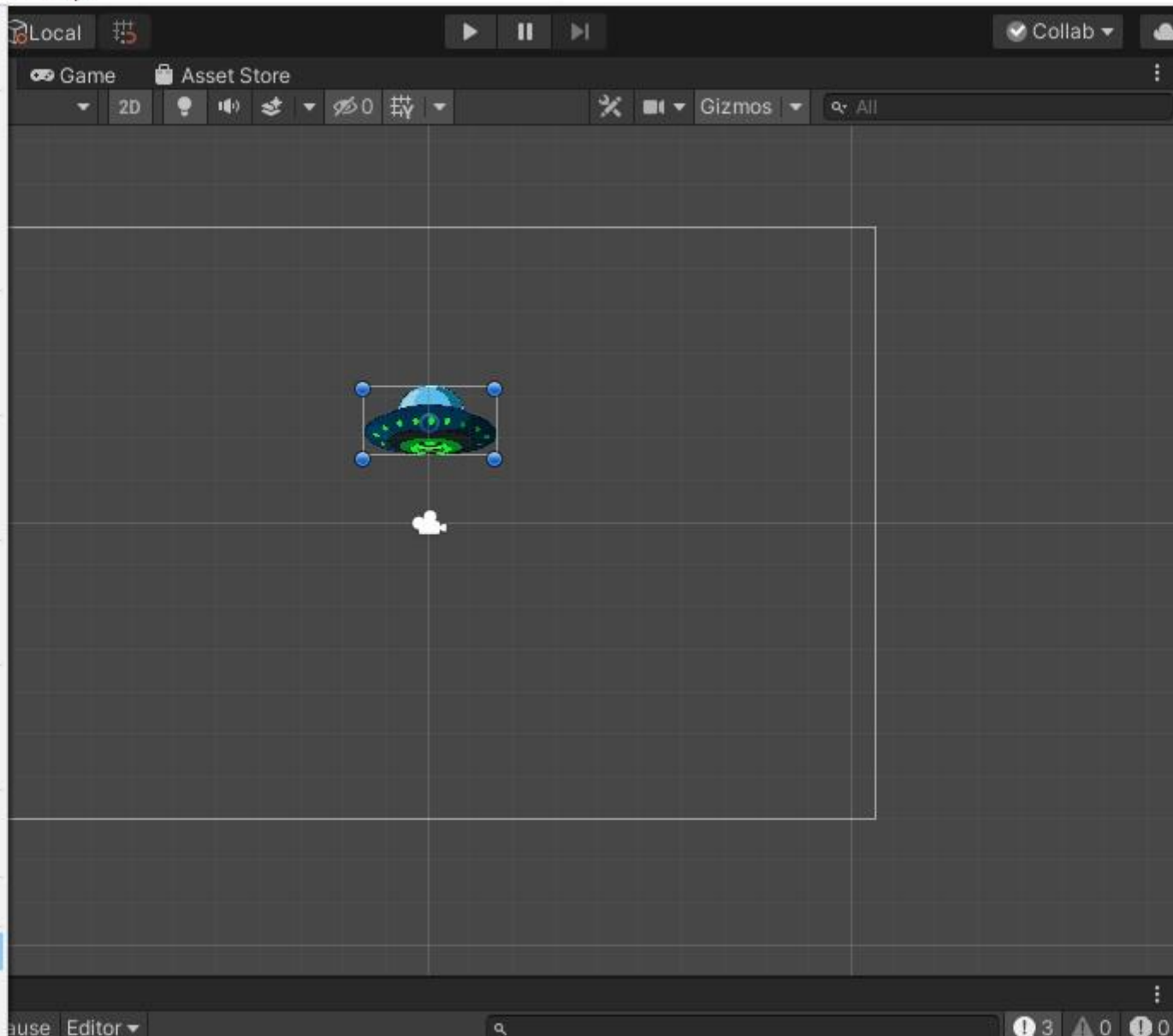
Projemizi
alıřtırdığımızda
Sprite objemiz
Y Ekseninde negatif
yönde hareket etti.
Yerçekimi (Gravity)
kuvvetiyle aşağıya
doğru düşmeye
başladı.




Projedeki Yerçekimini Kaldırmak



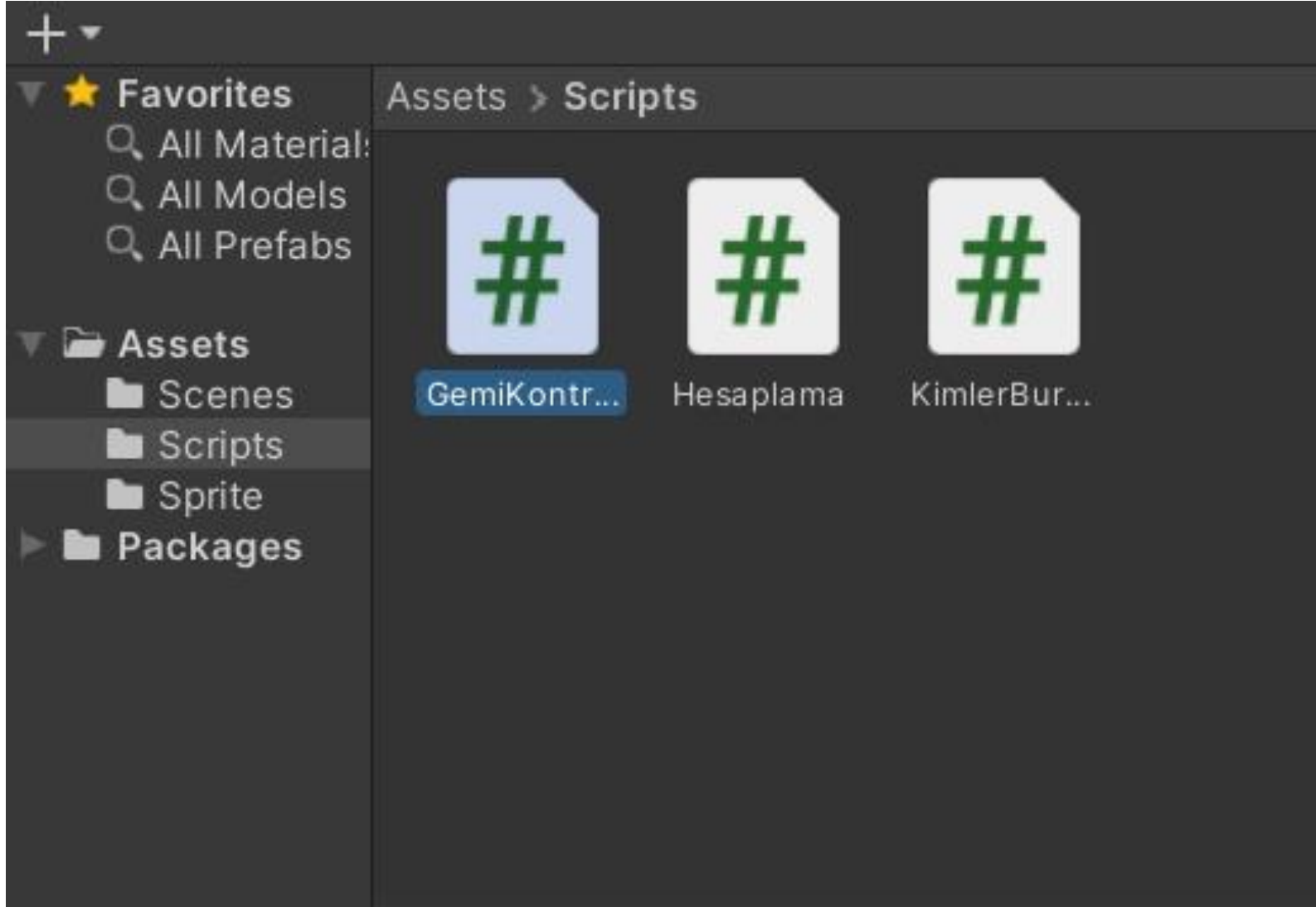
Undo Inspector	Ctrl+Z
Redo	Ctrl+Y
Select All	Ctrl+A
Deselect All	Shift+D
Select Children	Shift+C
Select Prefab Root	Ctrl+Shift+R
Invert Selection	Ctrl+I
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Duplicate	Ctrl+D
Rename	
Delete	
Frame Selected	F
Lock View to Selected	Shift+F
Find	Ctrl+F
Play	Ctrl+P
Pause	Ctrl+Shift+P
Step	Ctrl+Alt+P
Sign in...	
Sign out	
Selection	>
Project Settings...	
Preferences...	
Shortcuts...	
Clear All PlayerPrefs	
Graphics Tier	>
Grid and Snap Settings	



Inspector	
uzaygemisi2	Static
Tag	Untagged
Layer	Default
Transform	
Position	X 0.06 Y 2.42 Z 0
Rotation	X 0 Y 0 Z 0
Scale	X 0.3267 Y 0.3267 Z 0.3267
Sprite Renderer	
Sprite	uzaygemisi2
Color	
Flip	X Y
Draw Mode	Simple
Mask Interaction	None
Sprite Sort Point	Center
Material	Sprites-Default
Additional Settings	
Sorting Layer	Default
Order in Layer	0
Rigidbody 2D	
Body Type	Dynamic
Material	None (Physics Mate)
Simulated	<input checked="" type="checkbox"/>
Use Auto Mass	<input type="checkbox"/>
Mass	1
Linear Drag	0
Angular Drag	0.05
Gravity Scale	1
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None



Graphics	Gravity	X	0	Y	0	-9.81
Input Manager	Default Material					None (Physics Material 2D) 
Physics	Velocity Iterations					8
Physics 2D	Position Iterations					3
Player	Velocity Threshold					1
Preset Manager	Max Linear Correction					0.2
Quality	Max Angular Correction					8
Script Execution Order	Max Translation Speed					100
Tags and Layers	Max Rotation Speed					360
TextMesh Pro	Baumgarte Scale					0.2
Time	Baumgarte Time Of Impact Scale					0.75
VFX	Time To Sleep					0.5
XR Plugin Management	Linear Sleep Tolerance					0.01
	Angular Sleep Tolerance					2
	Default Contact Offset					0.01



Oyun Objelerinin Vektörel Hareketi

GemiKontrol isminde yeni
bir Script oluşturalım.

Unity İçerisinden Dokümanı Offline Açmak

Unity Manual

Scripting Reference

Premium Expert Help - Beta

Unity Services

Unity Forum

Unity Answers

Unity Feedback

Check for Updates

Download Beta...

Manage License

Release Notes

Software Licenses

Report a Bug...

Reset Packages to defaults

Troubleshoot Issue...

Rigidbody2D.AddForce


[SWITCH TO MANUAL](#)

```
public void AddForce(Vector2 force, ForceMode2D mode = ForceMode2D.Force);
```

Parameters

force	Components of the force in the X and Y axes.
mode	The method used to apply the specified force.

Oyun
Objelerinin
Vektörel
Hareketi

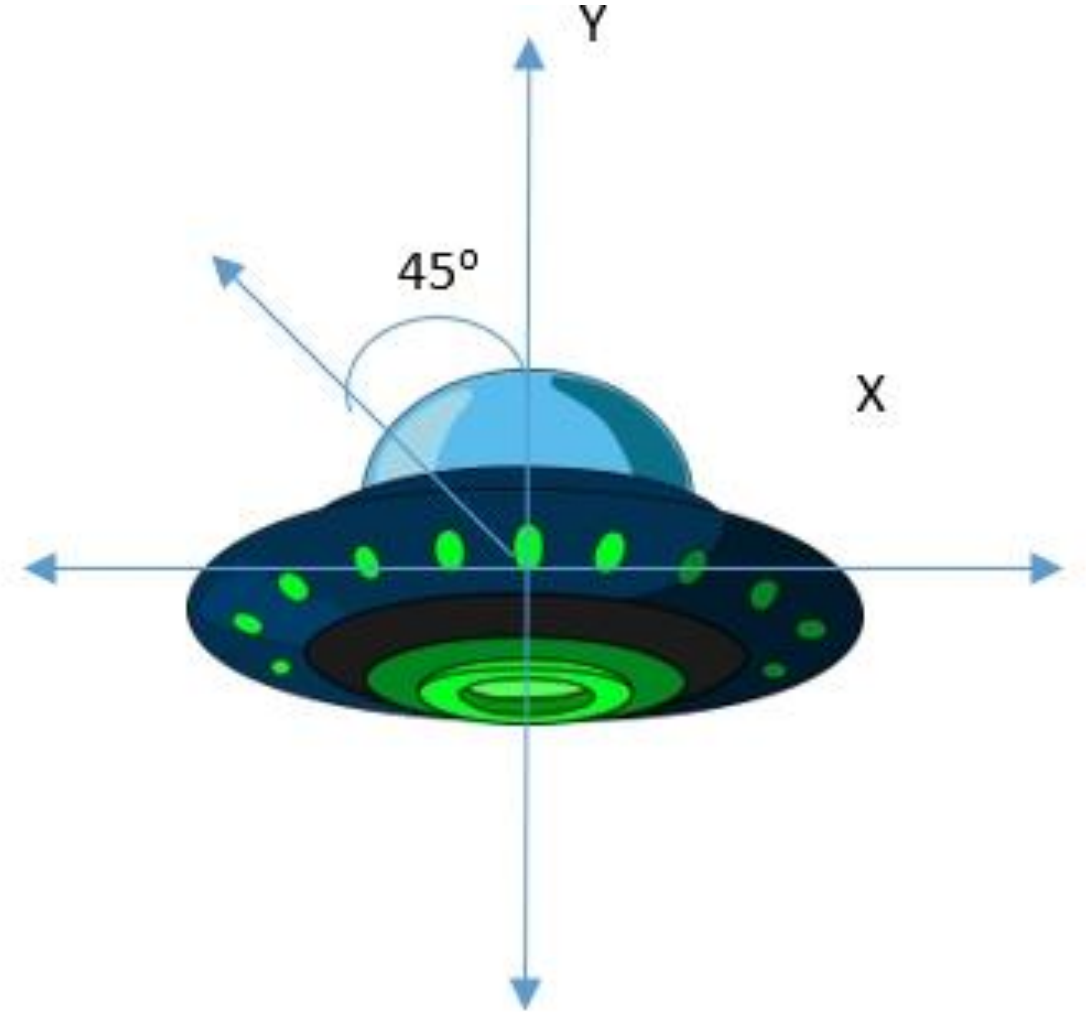


Oyun Objelerinin Vektörel Hareketi

- 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);`

Egzersiz

- Uzay Gemisi oyun objesine etki eden kuvveti, gemi sol yukarıya doğru 45 derecelik açı ile hareket edecek duruma getiriniz. (Kuvvetin X ve Y değerlerini istediğiniz değer yapabilirsiniz)



Egzersiz - Çözüm

```
void Start()
{
    GetComponent<Rigidbody2D>().AddForce(new Vector2(-5, 5),
    ForceMode2D.Impulse);
}
```

