24360859210 Elif Nur Beycan

# Oyun Programlama Hafta 1

Bu raporda kullanılan kodlara erişmek için github linki:
https://github.com/elifnurbeycan/unity-ile-oyun-programlama/tree/main/hafta1/scripts

## 1-Yeni proje oluşturma

İlk olarak **Unity Hub** uygulaması açılarak projelerin ve Unity sürümlerinin yönetildiği ana ekran görüntülenir. **Projects** sekmesinde daha önce oluşturulmuş projeler listelenir. Yeni bir proje başlatmak için sağ üstteki **"New Project"** butonuna tıklanır.

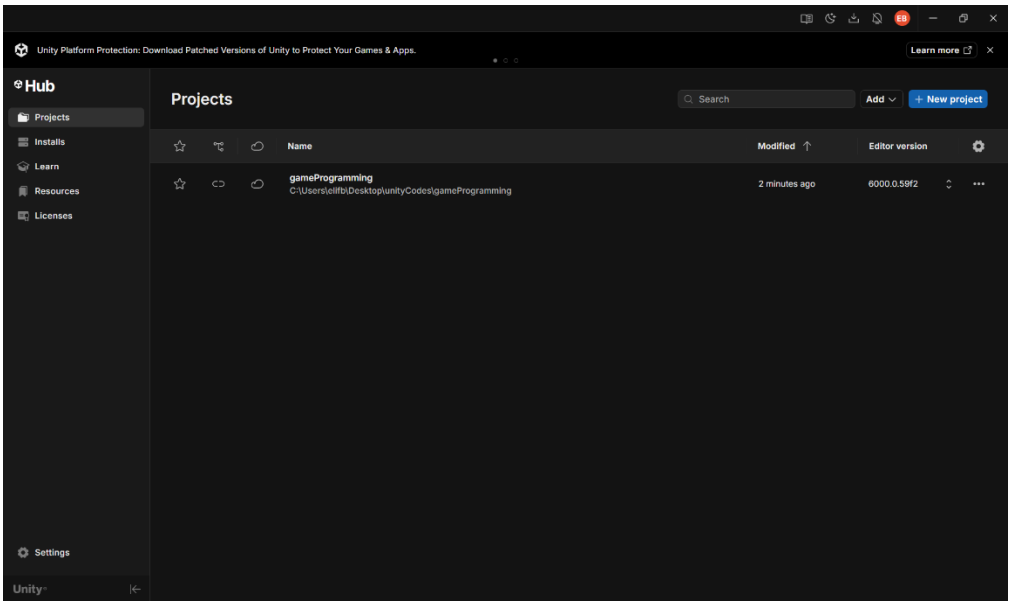
Açılan proje oluşturma penceresinde:

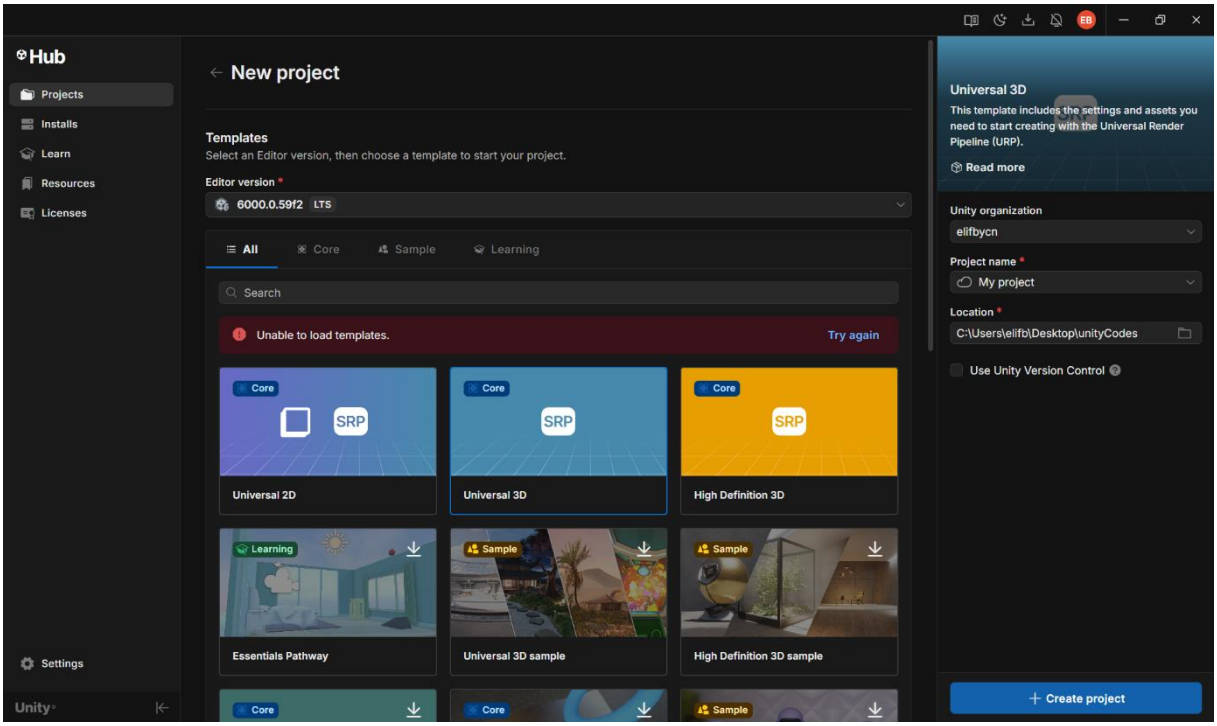
- **Editor version** kısmından kullanılacak Unity sürümü seçilir (örnekte: *Unity 6.0.0f1 LTS*).
- **Template (şablon)** kısmında proje türü belirlenir. Bu örnekte **Universal 3D (URP)** seçilmiştir çünkü 3D nesneler, ışıklar ve materyaller üzerinde çalışmak için uygundur.
- Sağ tarafta **Project name** alanına proje ismi (*gameProgramming*), **Location** kısmına ise projenin kaydedileceği dizin (*C:\Users\elifb\Desktop\UnityCodes*) girilir.
- Tek kişilik projelerde **Use Unity Version Control** kutucuğu işaretlenmez.

Tüm ayarlar tamamlandıktan sonra **"Create Project"** butonuna basılarak proje oluşturulur. Unity, gerekli dosyaları hazırlayıp varsayılan bir sahne (**SampleScene**) ile çalışma ortamını açar. Açılan arayüzde;

- **Hierarchy** paneli sahnedeki nesneleri,
- **Scene** bölümü 3D çalışma alanını,
- **Inspector** paneli ise seçilen nesnenin özelliklerini gösterir.

Bu aşamadan sonra sahneye nesne ekleme, materyal atama ve script yazma adımlarına geçilir.

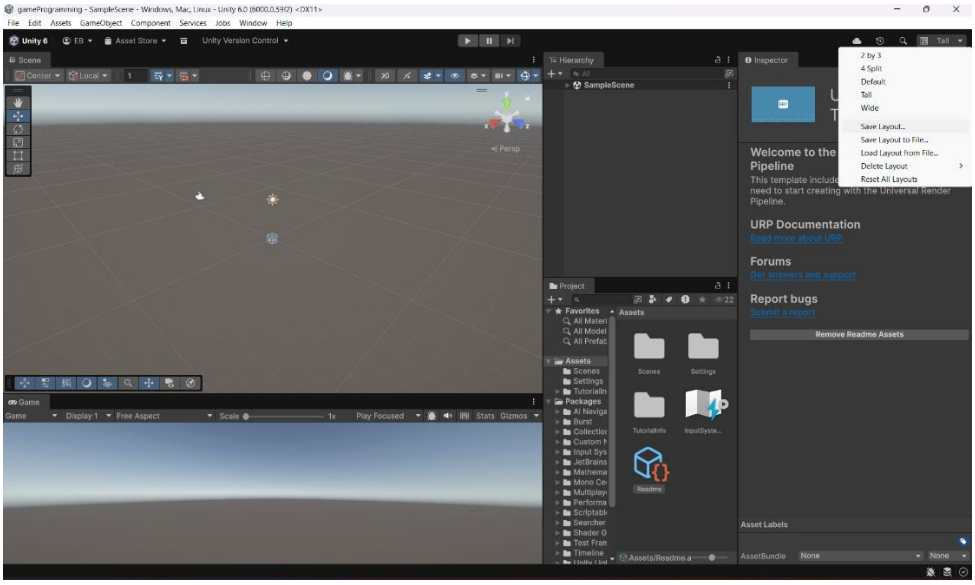



## 2-Kendi özel layout’unu oluşturma

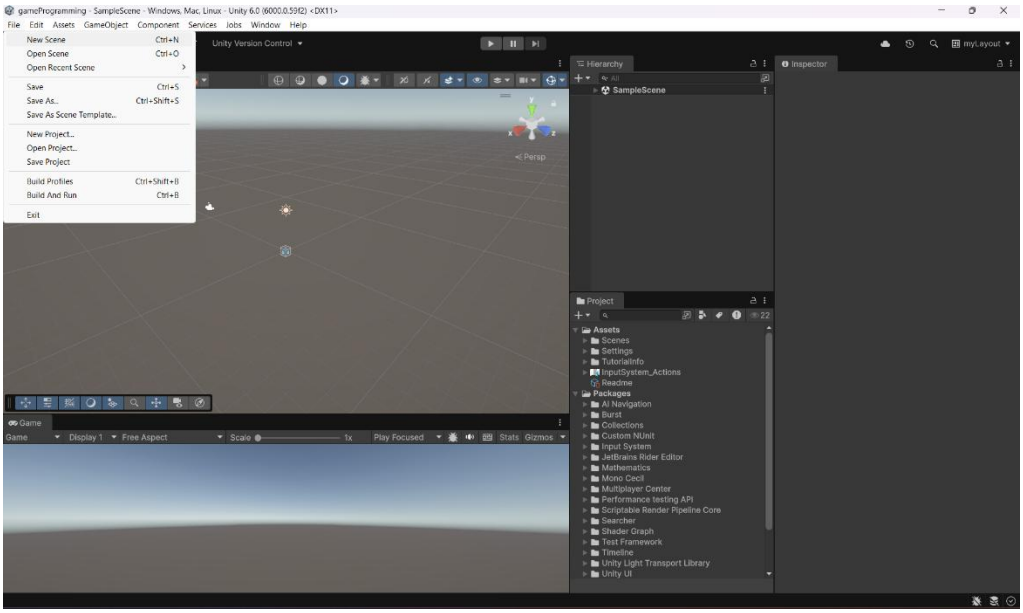
Unity arayüzü; **Scene**, **Game**, **Hierarchy**, **Inspector** ve **Project** gibi panellerden oluşur.
Bu panellerin yerleşimini (düzenini) kendi çalışma alışkanlıklarına göre değiştirebilir ve bu düzeni kalıcı hale getirebilirsin.
Bu işlem, proje geliştirirken verimliliği artırmak ve sık kullanılan panellere kolay erişim sağlamak için oldukça faydalıdır.

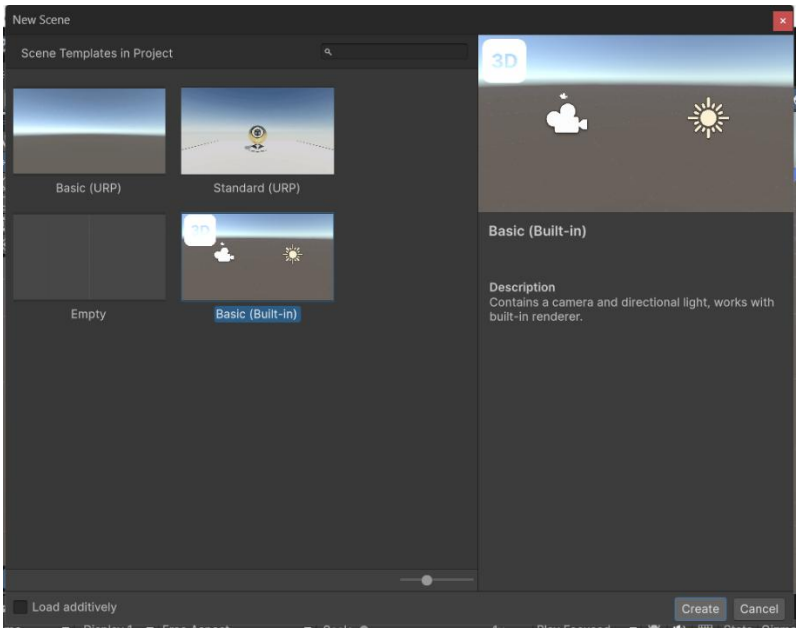
Panelleri (Scene, Game, Inspector vb.) sürükleyip istediğim gibi yerleştirdim.
Sağ üstteki **Layout** menüsünden **Save Layout…** seçeneğiyle düzeni kaydedip isim verdim.
Artık kendi özel görünümüm kayıtlı; istersem **Reset All Layouts** ile varsayılan hâle dönebilirim.

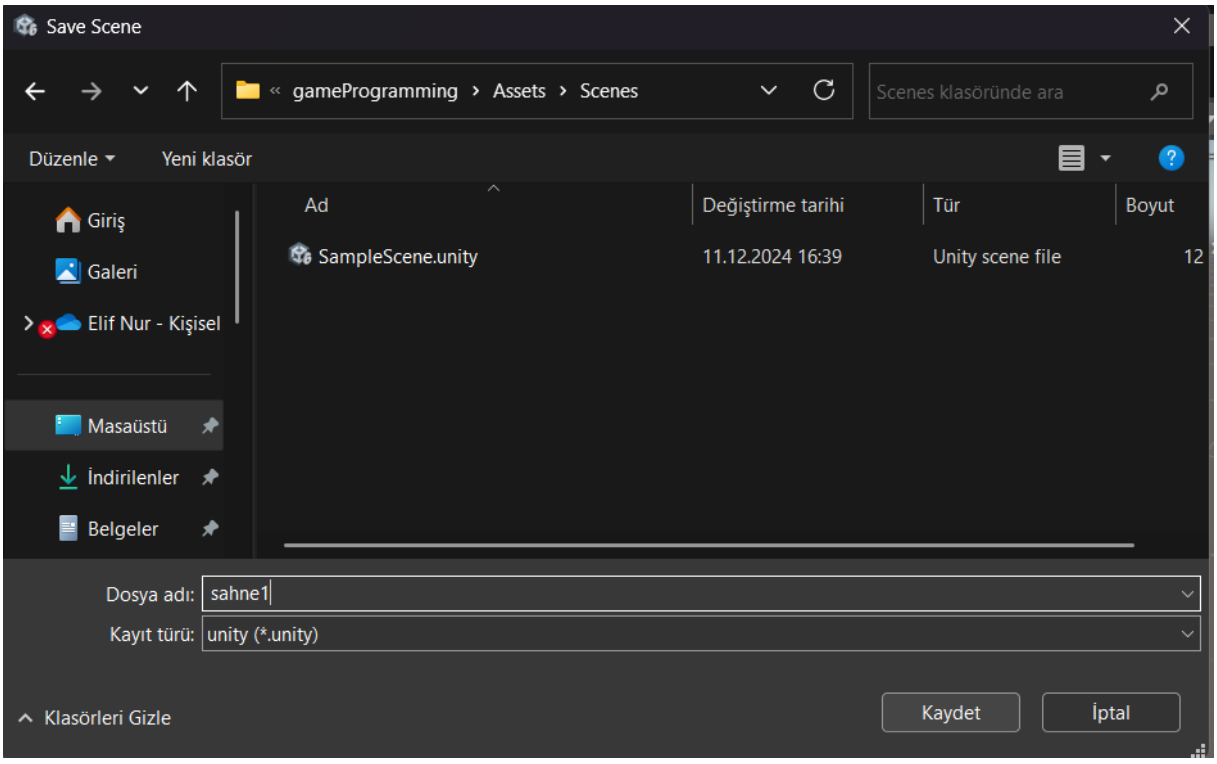


## 3-Sahne ekleme

Unity’de sahneler (Scenes), oyunun veya uygulamanın farklı bölümlerini temsil eder.
Yeni bir sahne eklemek için üst menüden **File → New Scene** seçeneğini kullandım.
Açılan pencerede **Basic (Built-in)** şablonunu seçip **Create** butonuna bastım.
Yeni sahne oluşturulduktan sonra **File → Save As** diyerek sahneyi kaydettim ve adını **“sahne1”** olarak belirledim.

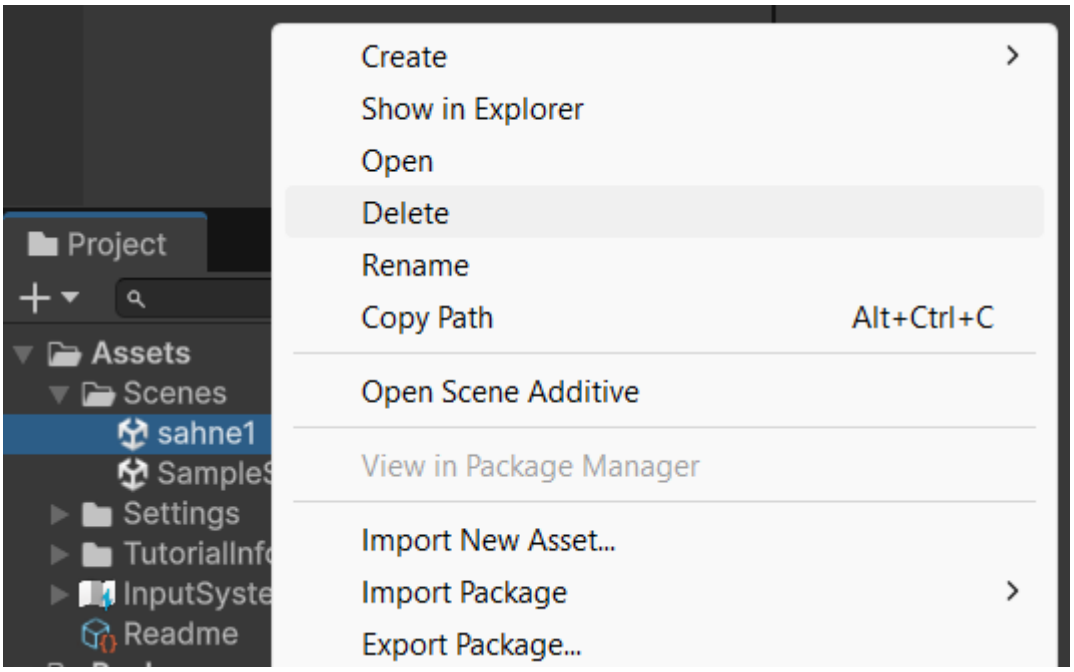




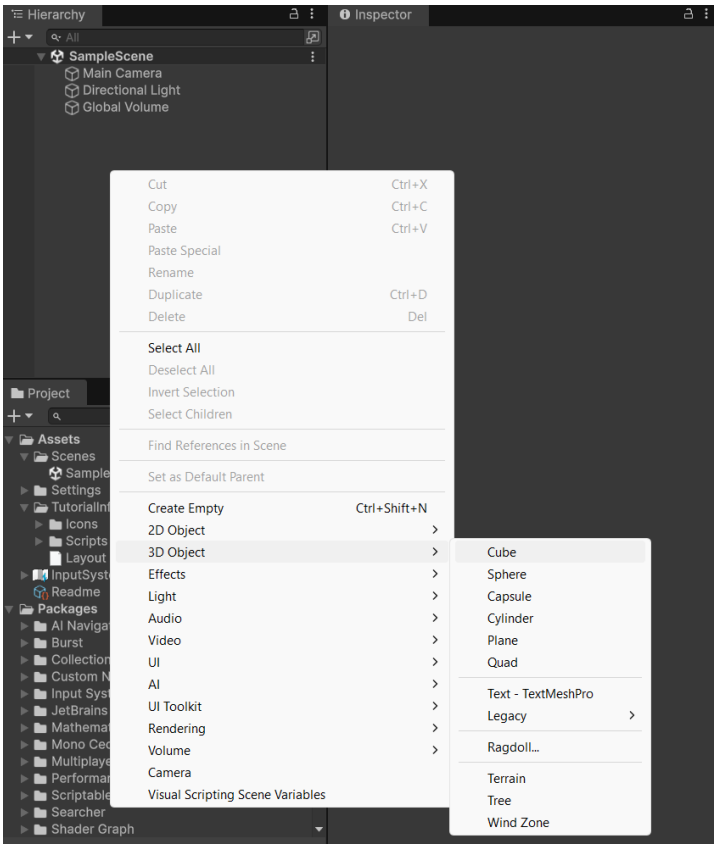

## 4-Sahne silme

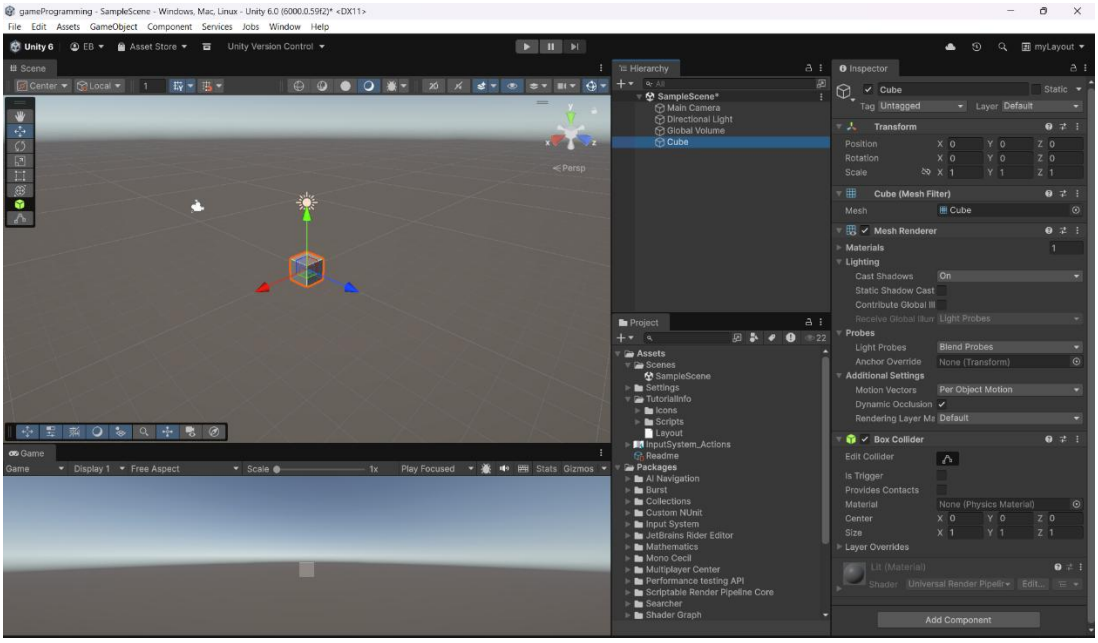
Projede yer alan bir sahneyi silmek için **Project** panelindeki **Scenes** klasörünü açtım.
Silmek istediğim sahne dosyasına (**sahne1**) sağ tıklayıp açılan menüden **Delete** seçeneğini seçtim.
Bu işlemle sahne proje dosyalarından kaldırıldı.




## 5-Sahneye nesne ekleme

Sahneye bir nesne eklemek için **Hierarchy** panelinde sağ tıklayıp **3D Object → Cube** seçeneğini seçtim.
Bu işlemle sahneye bir küp nesnesi eklendi.
Küp, sahne merkezinde (0,0,0) konumunda oluşturuldu ve **Inspector** panelinde konum, boyut ve materyal gibi özellikleri görüntülenebiliyor.



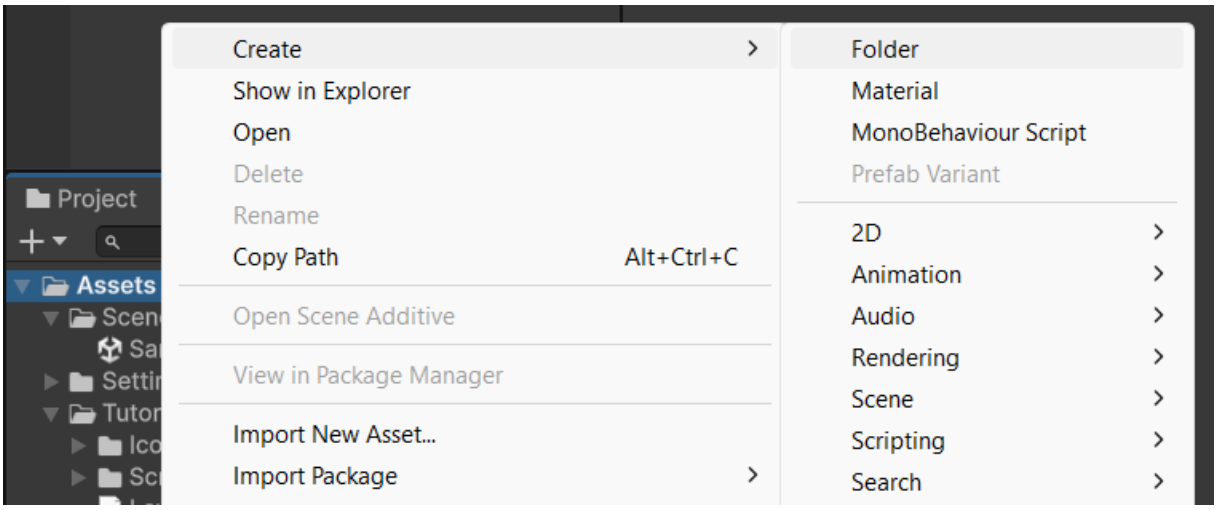


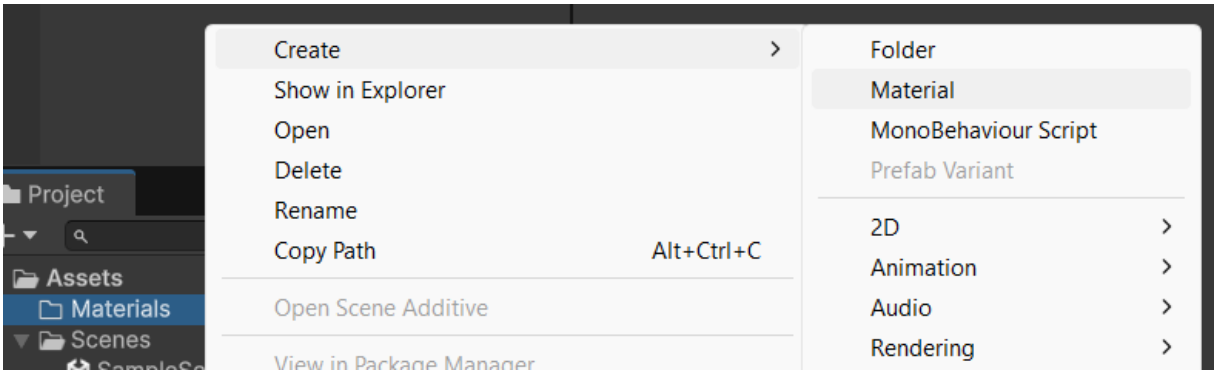

## 6-Sahnedeki nesneye materyal ekleme

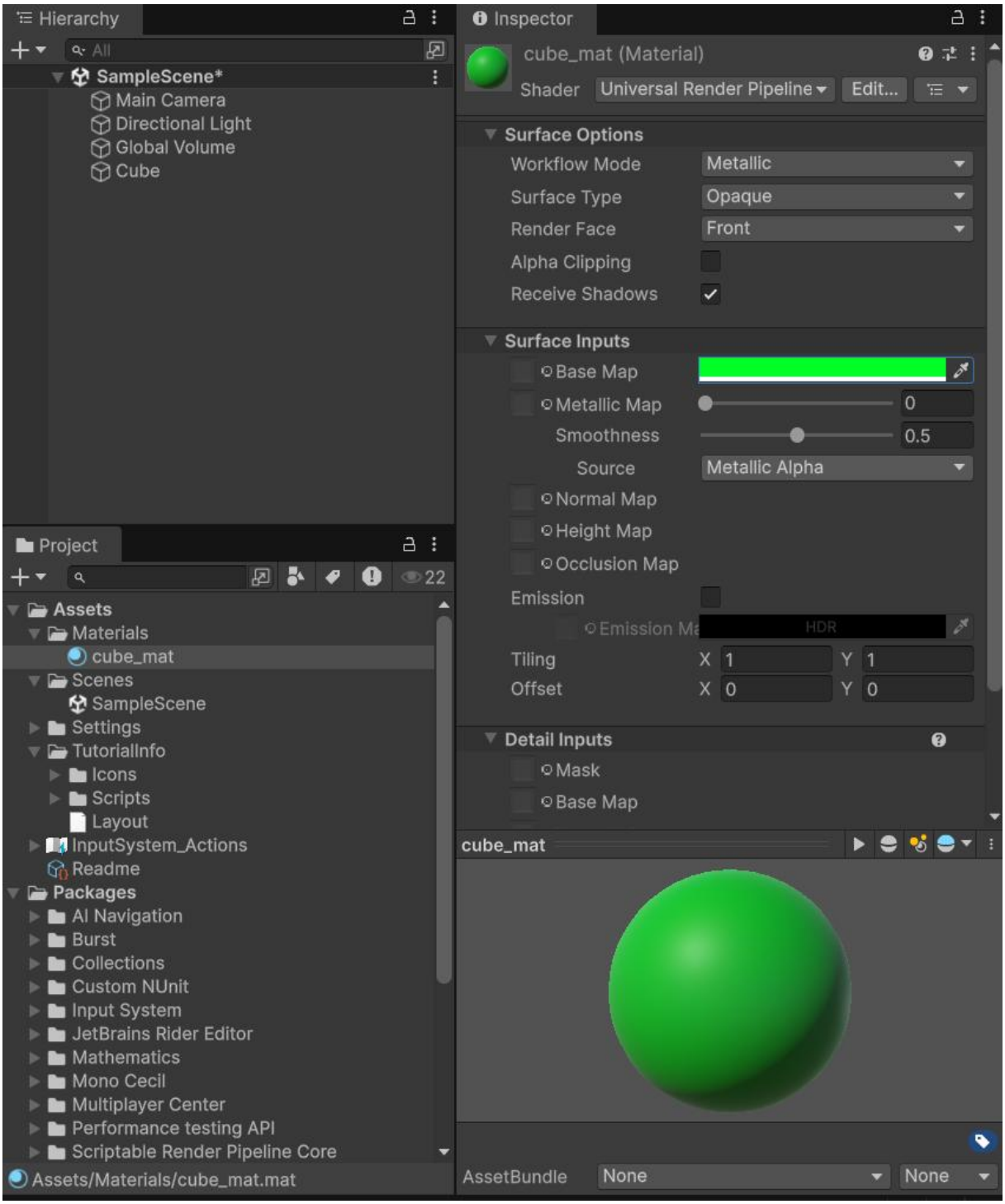
Öncelikle **Assets** klasöründe sağ tıklayarak **Create → Folder** seçeneğiyle "**Materials**" adında bir klasör oluşturdum.
Daha sonra bu klasör içinde tekrar sağ tıklayıp **Create → Material** seçeneğiyle yeni bir materyal (**cube_mat**) oluşturdum.

Materyali seçince **Inspector** panelinde özellikleri görüntülendi.
Rengi değiştirmek için **Surface Inputs** başlığı altındaki **Base Map** kutusuna tıklayıp renk paletinden **yeşil** tonunu seçtim.
Bu işlem materyalin temel yüzey rengini belirledi.

Son olarak **cube_mat** materyalini **Hierarchy** panelindeki **Cube** nesnesinin üzerine sürükleyip bırakarak uyguladım.
Böylece küp nesnesi sahnede yeşil renkle görünecek şekilde materyal kazandı.

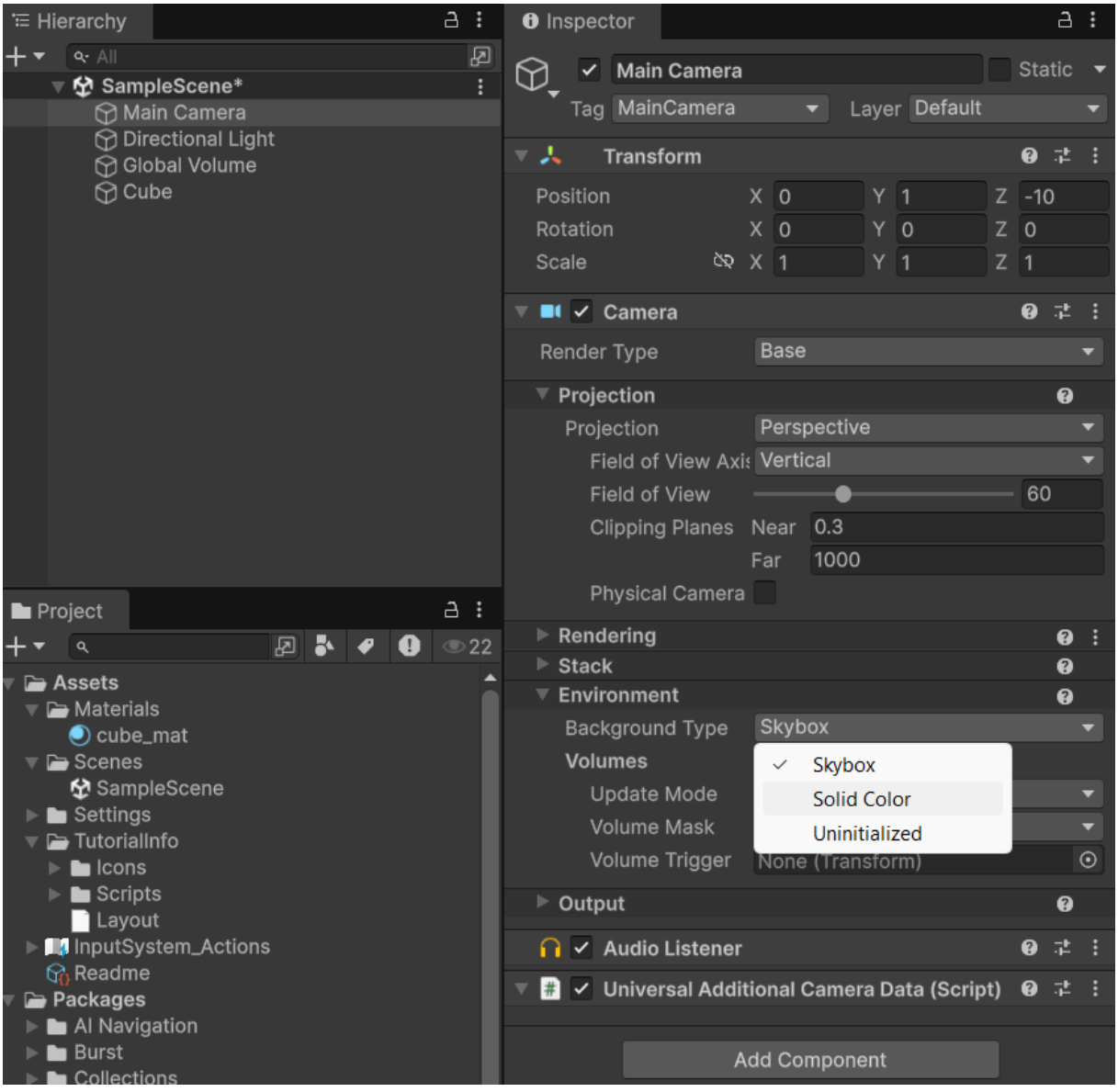

Create
Show in Explorer
Open
Delete
Rename
Copy Path Alt+Ctrl+C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Folder
Material
MonoBehaviour Script
Prefab Variant
2D
Animation
Audio
Rendering
Scene
Scripting
Search
Project
Assets


Create
Show in Explorer
Open
Delete
Rename
Copy Path Alt+Ctrl+C
Open Scene Additive
Folder
Material
MonoBehaviour Script
Prefab Variant
2D
Animation
Audio
Rendering
Project
Assets
Materials
Scenes


Hierarchy
All
SampleScene*
Main Camera
Directional Light
Global Volume
Cube
Project
22
Assets
Materials
cube_mat
Scenes
SampleScene
Settings
TutorialInfo
Icons
Scripts
Layout
InputSystem_Actions
Readme
Packages
AI Navigation
Burst
Collections
Custom NUnit
Input System
JetBrains Rider Editor
Mathematics
Mono Cecil
Multiplayer Center
Performance testing API
Scriptable Render Pipeline Core
Assets/Materials/cube_mat.mat
Inspector
cube_mat (Material)
Shader
Universal Render Pipeline
Edit...
Surface Options
Workflow Mode
Metallic
Surface Type
Opaque
Render Face
Front
Alpha Clipping
Receive Shadows
Surface Inputs
Base Map
Metallic Map
0
Smoothness
0.5
Source
Metallic Alpha
Normal Map
Height Map
Occlusion Map
Emission
HDR
Tiling
X 1
Y 1
Offset
X 0
Y 0
Detail Inputs
Mask
Base Map
cube_mat
AssetBundle
None
None

## 7-Arka Plan Rengini Değiştirme

Sahnenin arka plan rengini değiştirmek için **Hierarchy** panelinden **Main Camera** nesnesini seçtim.
**Inspector** panelinde **Environment** başlığı altındaki **Background Type** seçeneğini "**Skybox**" yerine "**Solid Color**" olarak değiştirdim.
Bu sayede arka plan, düz bir renk olacak şekilde ayarlandı.
İstenilen renge ulaşmak için hemen altındaki renk kutusuna tıklanarak renk paletinden seçim yapılabiliyor.

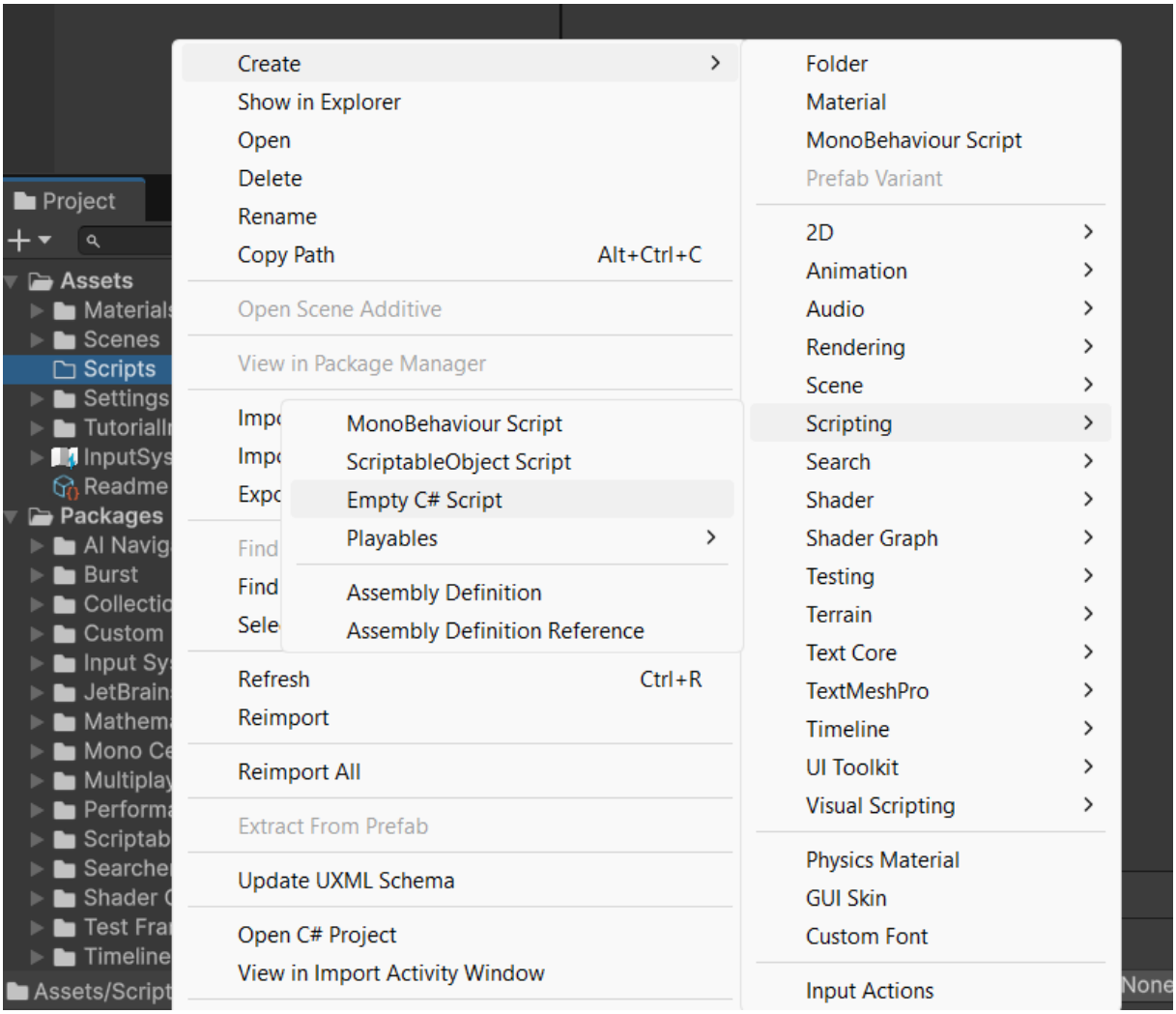


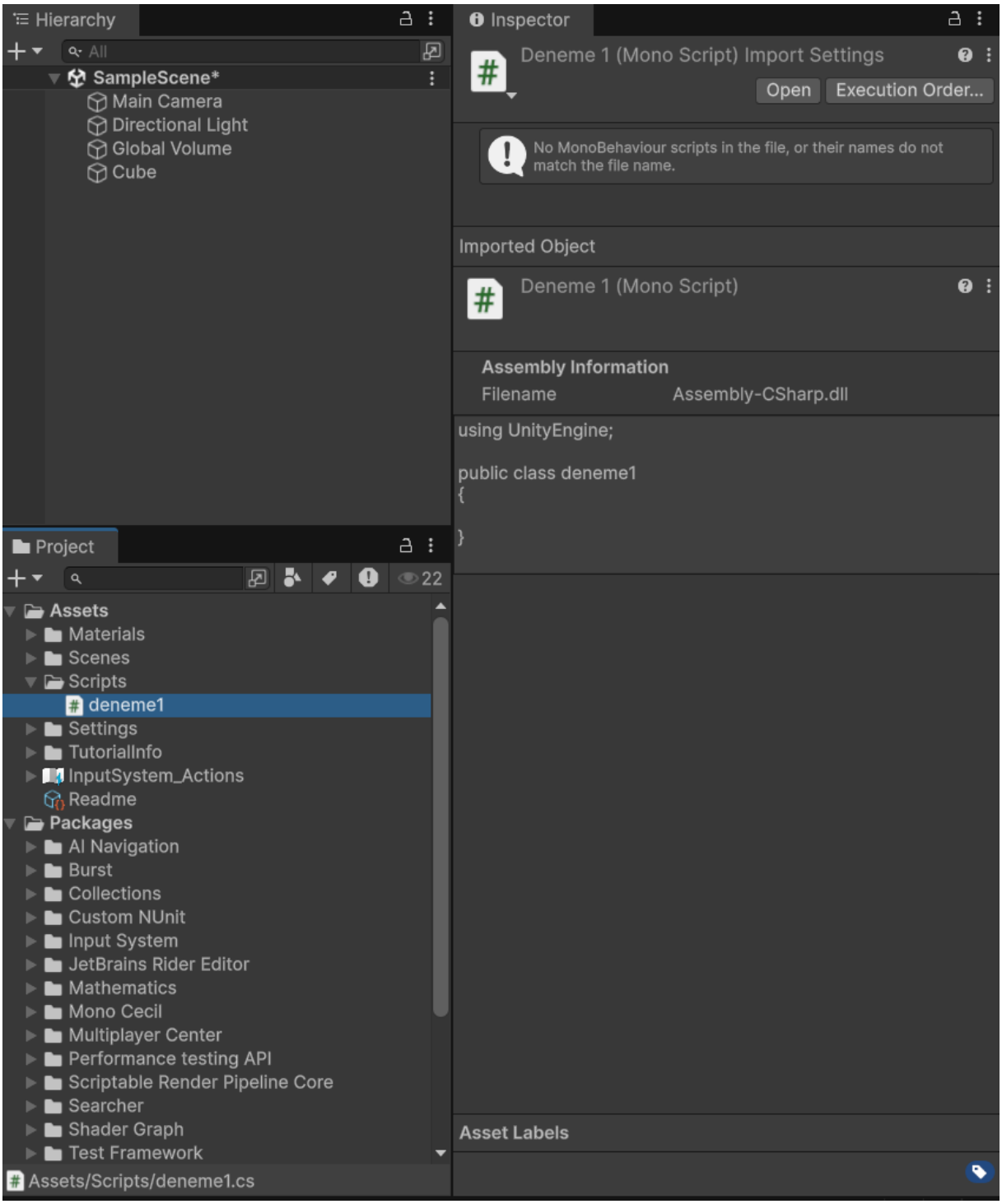
## 8-Sahnedeki nesneye script ekleme

Öncelikle **Assets** klasöründe sağ tıklayarak **Create → Folder** seçeneğiyle "**Scripts**" adında bir klasör oluşturdum.
Ardından bu klasör içinde tekrar sağ tıklayıp **Create → C# Script** seçeneğiyle yeni bir script dosyası (**deneme1**) oluşturdum.

Oluşturduğum script dosyasına çift tıklayarak **Visual Studio Code** üzerinde açtım ve gerekli düzenlemeleri yaptım.
Daha sonra scripti **Hierarchy** panelindeki **Cube** nesnesinin üzerine **sürükleyip bırakarak** sahnedeki nesneye ekledim.

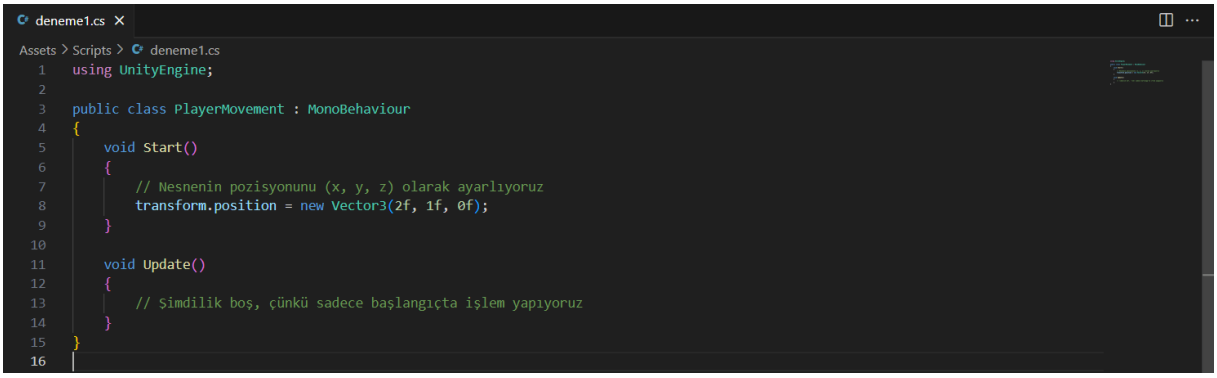
Bu sayede küp nesnesine davranış eklenmiş ve scriptte yazılan kodlar artık bu nesne üzerinde çalışabilir hale gelmiştir.

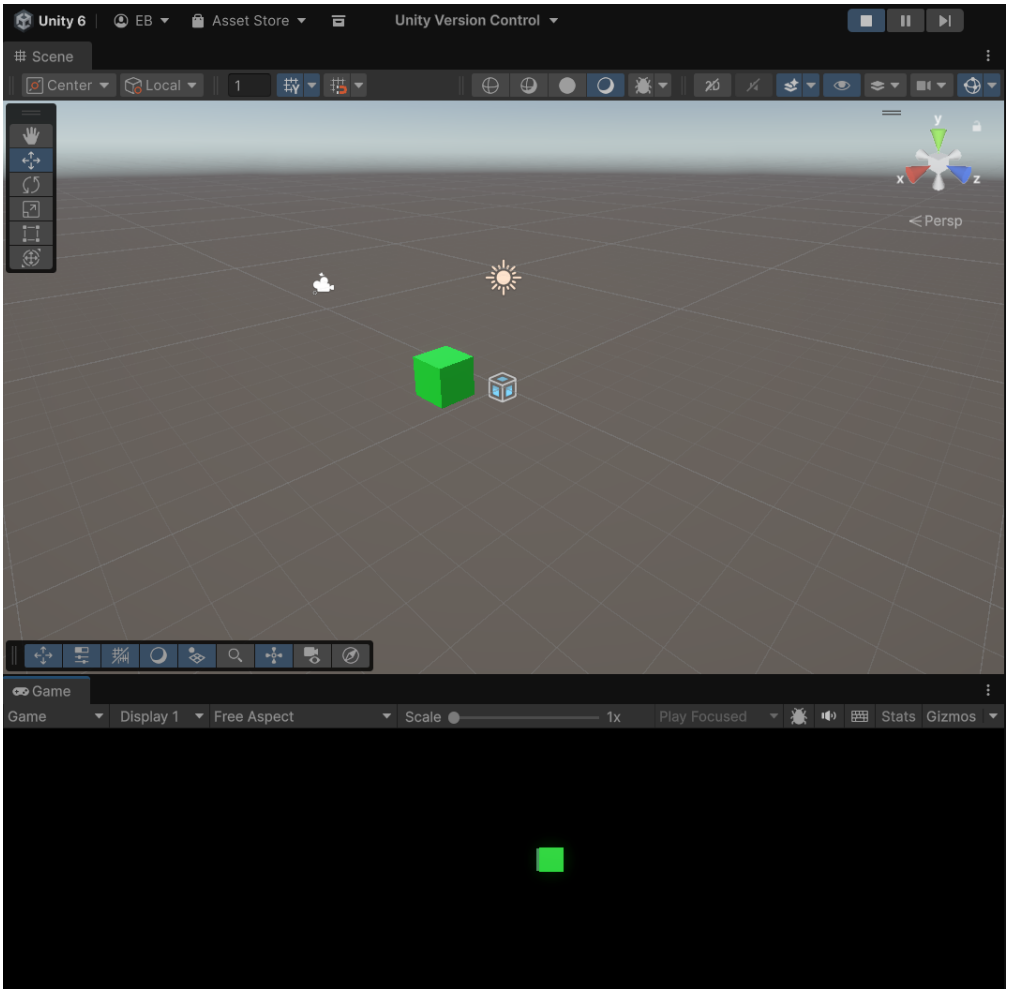



Hierarchy
All
SampleScene*
Main Camera
Directional Light
Global Volume
Cube
Project
22
Assets
Materials
Scenes
Scripts
deneme1
Settings
TutorialInfo
InputSystem_Actions
Readme
Packages
AI Navigation
Burst
Collections
Custom NUnit
Input System
JetBrains Rider Editor
Mathematics
Mono Cecil
Multiplayer Center
Performance testing API
Scriptable Render Pipeline Core
Searcher
Shader Graph
Test Framework
Assets/Scripts/deneme1.cs
Inspector
Deneme 1 (Mono Script) Import Settings
Open
Execution Order...
No MonoBehaviour scripts in the file, or their names do not match the file name.
Imported Object
Deneme 1 (Mono Script)
Assembly Information
Filename
Assembly-CSharp.dll
using UnityEngine;
public class deneme1
{
}
Asset Labels

## 9-Script ile nesnenin konumunu bir kez değiştirme (start fonksiyonu)

Bu adımda sahnedeki nesnenin konumunu, oyun başladığında sadece **bir kez** değiştirecek bir script yazdım.
**Scripts** klasöründeki "**deneme1.cs**" dosyasını **Visual Studio Code** ile açarak aşağıdaki kodu ekledim. Bu kod, oyun başladığında nesnenin pozisyonunu sahnede belirtilen koordinatlara (2, 1, 0) taşıyor.
**Start()** fonksiyonu yalnızca oyun ilk çalıştığında bir kez çağrıldığı için nesnenin konumu sadece başlangıçta değiştiriliyor.



## 10-Script ile nesnenin konumunu sürekli değiştirme (update fonksiyonu ile tek yönde ilerleme)

Bu adımda nesnenin sahnede **sürekli olarak hareket etmesini** sağlamak için **Update()** fonksiyonunu kullandım.
Script dosyasında aşağıdaki kodu yazarak nesnenin z ekseni boyunca ileri yönde sürekli hareket etmesini sağladım.

**Update()** fonksiyonu her karede (frame) çalıştığı için, bu kod nesnenin sahne boyunca sürekli ilerlemesini sağlar.
Bu sayede nesne tek bir yönde (Z ekseninde) sabit hızla hareket eder.


## 11-Zamanın normalizasyonu (nesnenin saniyede bir birim ilerlemesi)

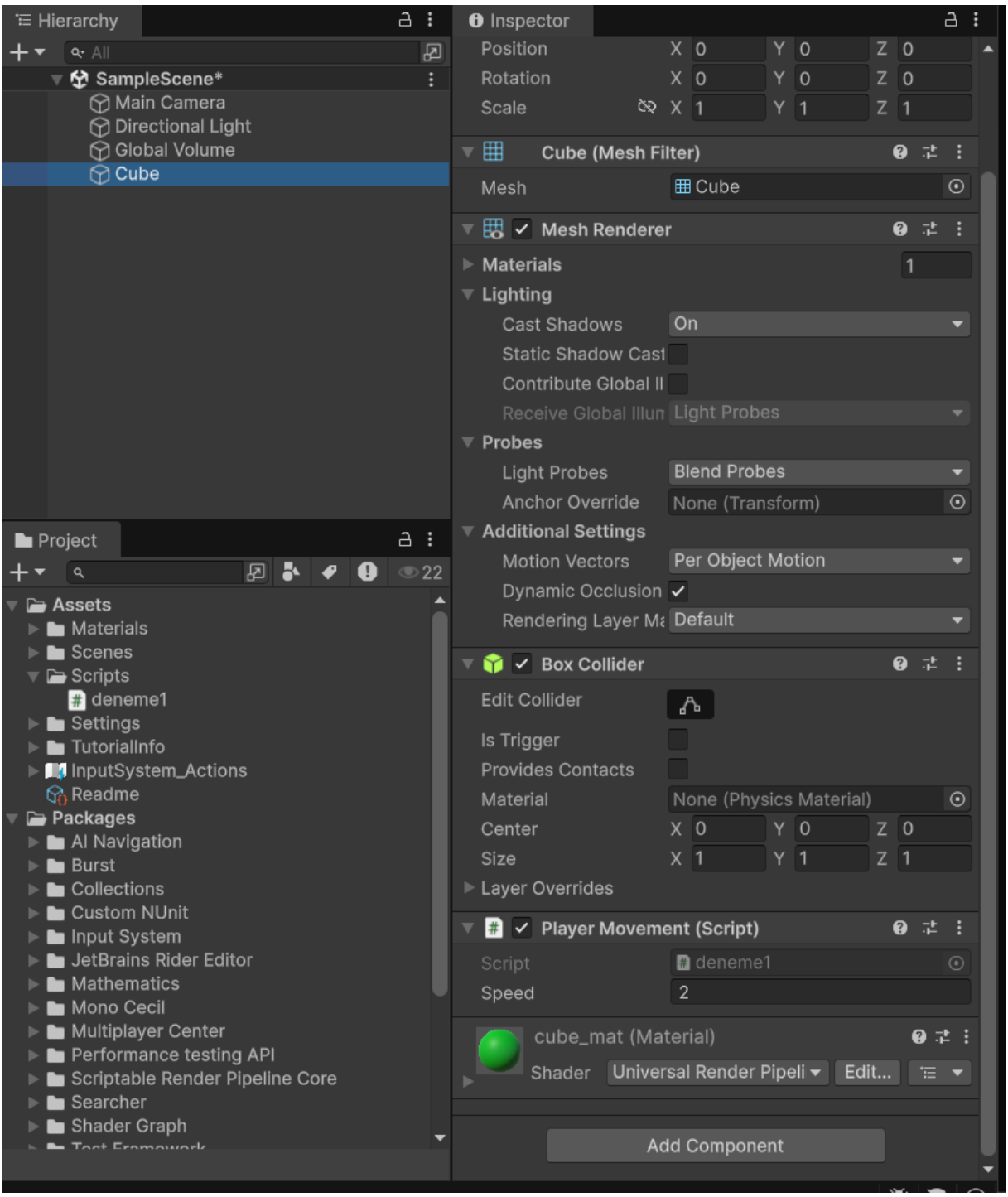
Önceki adımda **zaman normalizasyonu yoktu**, bu yüzden hareket **FPS'e bağlı**ydı. Bilgisayar hızlıysa nesne çok hızlı ilerleyip ekrandan çabucak kayboldu. Bunu düzeltmek için Time.deltaTime kullandım; böylece hareket **saniye bazında** hesaplandı ve tüm cihazlarda aynı hız elde edildi. Nesne artık **saniyede 1 birim** sabit hızla ilerliyor; FPS değişse bile hız değişmiyor.


## 12-Speed değişkeni tanımlama (public, private farkı)

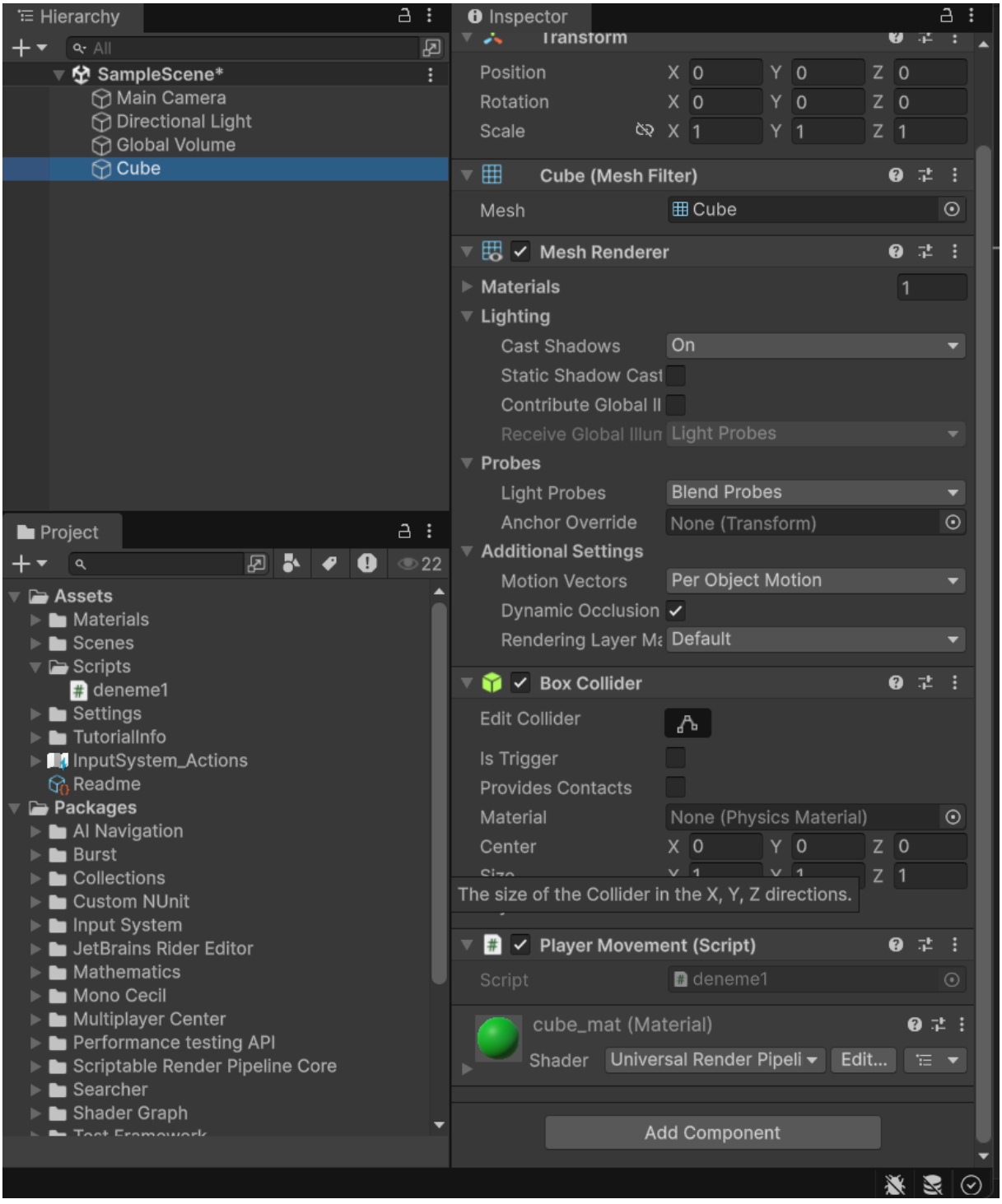
Bu adımda nesnenin hareket hızını kontrol etmek için **speed** adında bir değişken tanımladım. Bu değişkeni **public** olarak tanımladığımda, Unity'nin **Inspector** panelinde en altta yer alan **Scripts** (Player Movement) kısmında otomatik olarak görünür hale gelir. Böylece hızı doğrudan bu panelden değiştirebilir, örneğin "Speed" alanına farklı bir değer yazarak nesnenin daha hızlı veya yavaş hareket etmesini sağlayabilirim. Buna karşılık, değişken

**private** olarak tanımlandığında sadece kod içinden erişilebilir ve **Inspector panelinde görünmez**, dolayısıyla dışarıdan düzenleme yapılamaz. Kısaca, *public değişkenler Inspector'da görünür ve dışarıdan değiştirilebilirken, private değişkenler yalnızca kod içinde erişilebilir.*

Public hali:



Private hali:



## 13-Klavyeden yön tuşları ile nesnenin hareketinin kontrolü (dikey eksen)

Bu adımda nesnenin klavyedeki yön tuşlarıyla hareket etmesini sağladım. Bunun için Unity'nin **Input System** fonksiyonlarından biri olan Input.GetAxis("Vertical") ifadesini kullandım. Bu komut, **yukarı ok (↑)** tuşuna basıldığında +1, **aşağı ok (↓)** tuşuna basıldığında ise -1 değeri döndürür. Bu değerleri kullanarak nesnenin **dikey eksende (Y ekseni)** hareket etmesini sağladım.


## 14-Yatay eksenin hareket kontrolüne dahil edilmesi

Bu adımda, önceki dikey eksen kontrolüne ek olarak nesnenin **yatay eksende (X ekseni)** de hareket edebilmesini sağladım. Böylece nesne artık klavyedeki hem **yön tuşları (↑ ↓ ← →)** hem de **W, A, S, D** tuşlarıyla iki eksende kontrol edilebiliyor.

Bunu yapmak için Unity'nin **Input.GetAxis()** fonksiyonunu hem "Horizontal" hem de "Vertical" parametreleriyle kullandım.

- "Horizontal" değeri, **sağ ok (→)** veya **D** tuşuna basıldığında +1, **sol ok (←)** veya **A** tuşuna basıldığında -1 değerini döndürür.
- "Vertical" değeri, **yukarı ok (↑)** veya **W** tuşuna basıldığında +1, **aşağı ok (↓)** veya **S** tuşuna basıldığında -1 döndürür.