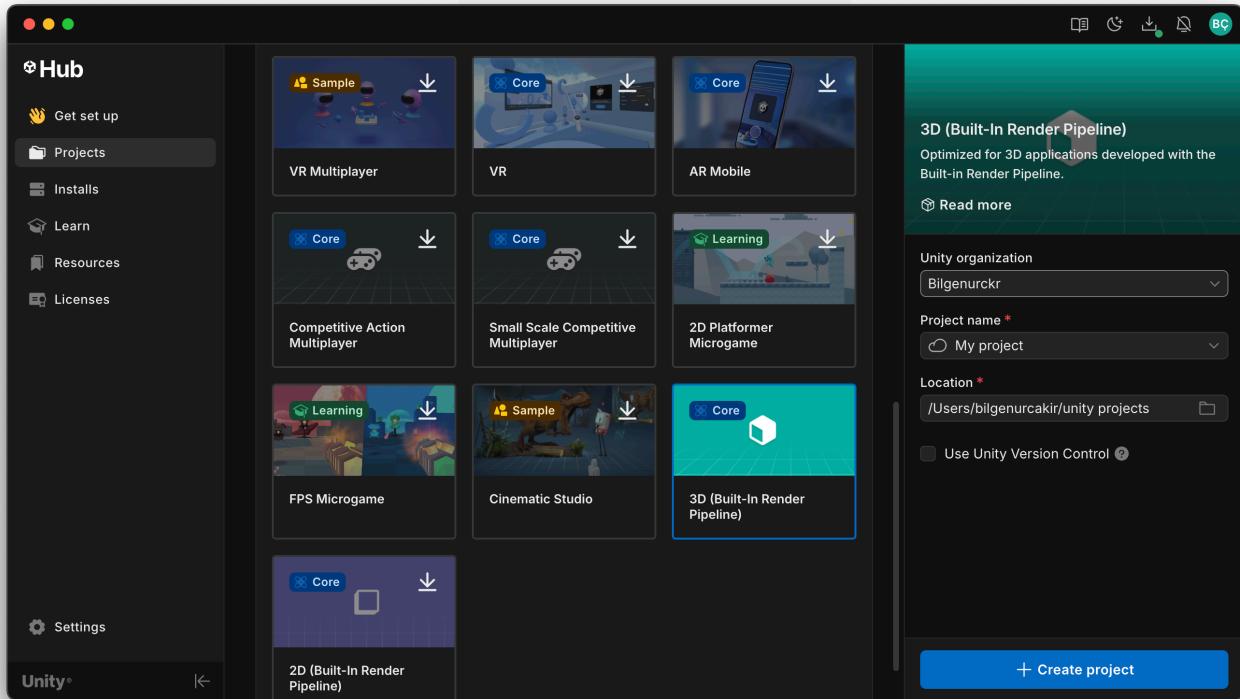
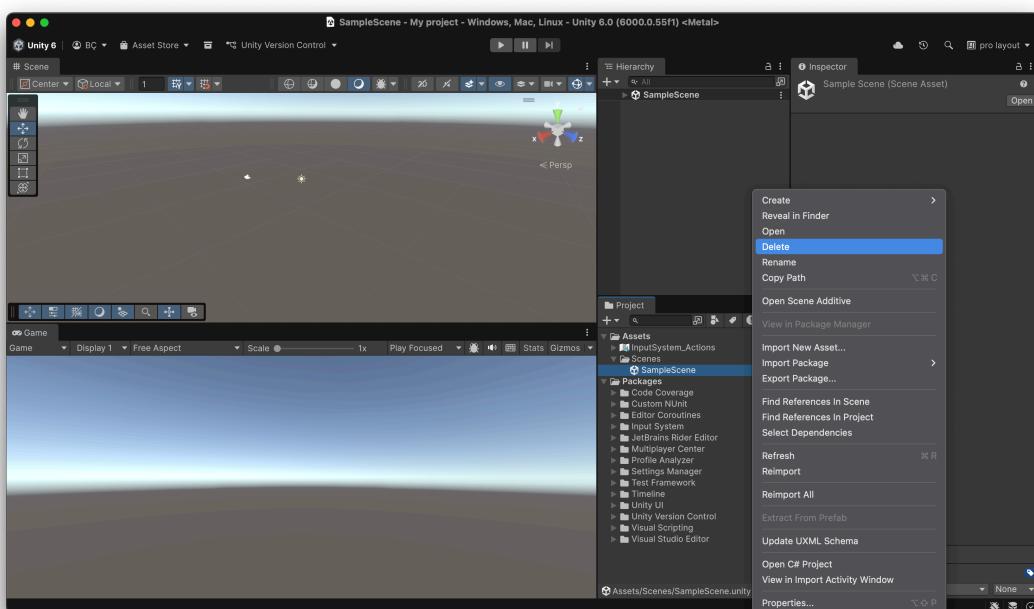


Yeni proje oluşturma

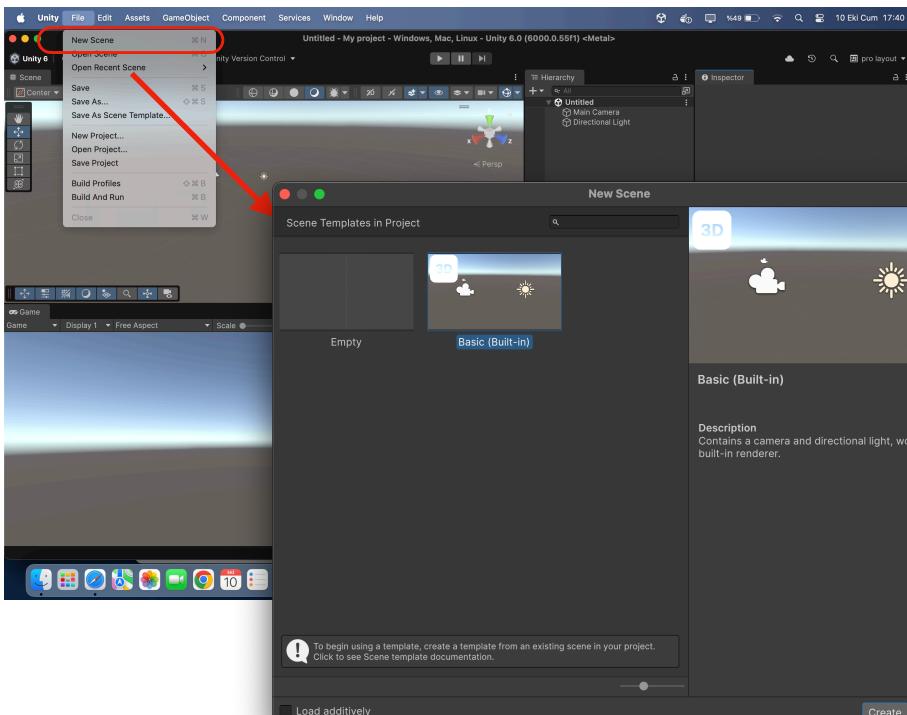


Unity Hub üzerinde Projects sayfası açılır. 3D(build-in render pipeline) seçeneğini seçilir, isim ve kurulacağı ortam belirlenerek proje oluşturulur.

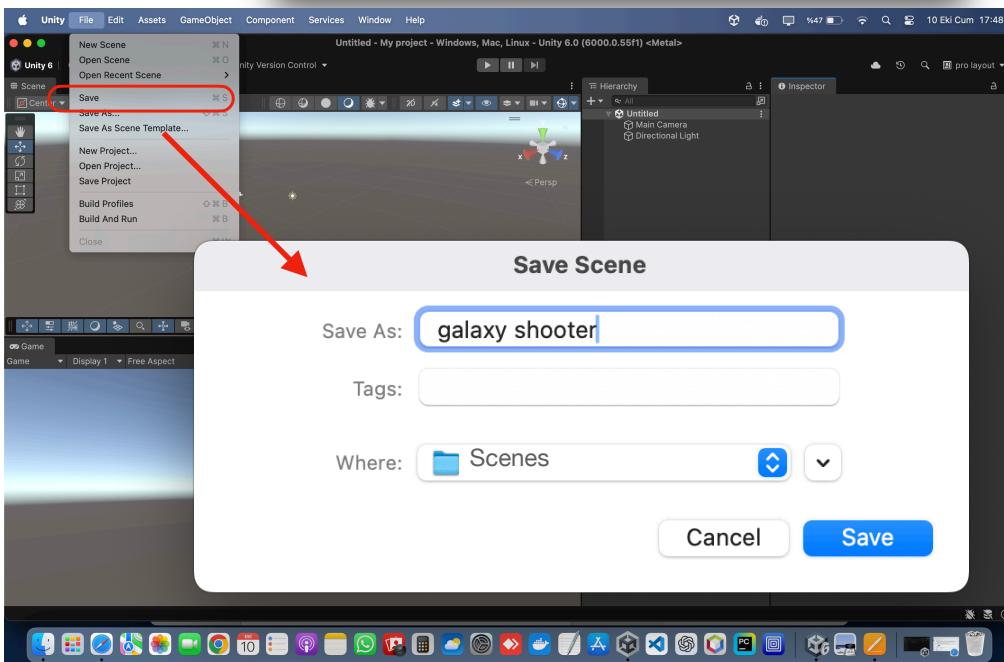
Sahne silme ve ekleme



Mevcut sahneyi Assets/ Scenes klasörleri altında bulabilir ve sağ tıklayıp silebiliriz.

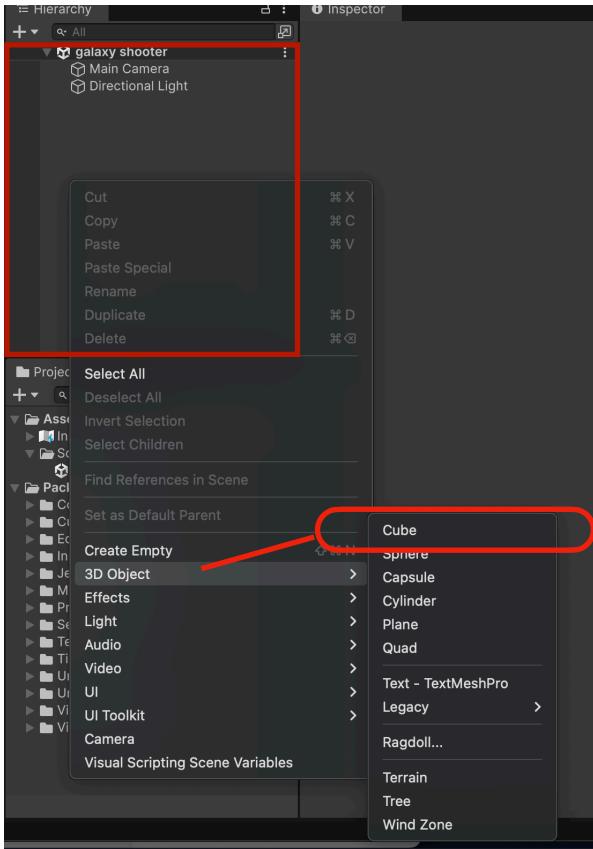


yeni sahneyi üst sekmlerden file-> new scenes seçeneğiyle ekleyebiliriz. Eklerken açılan pencereden basic(built-in) sekmesini seçip create tuşuna basıyoruz.



Ancak şu anda sahnemiz bellekte olduğundan projeye eklemek için tekrardan file sekmesine geliyoruz ve save tuşuna basıyoruz. Ardından assets klasörünün altındaki scenes klasörüne kaydediyoruz.

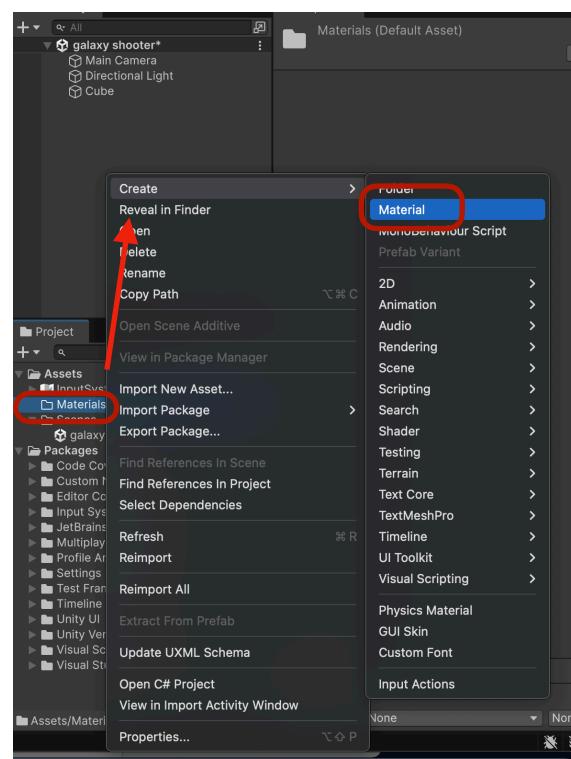
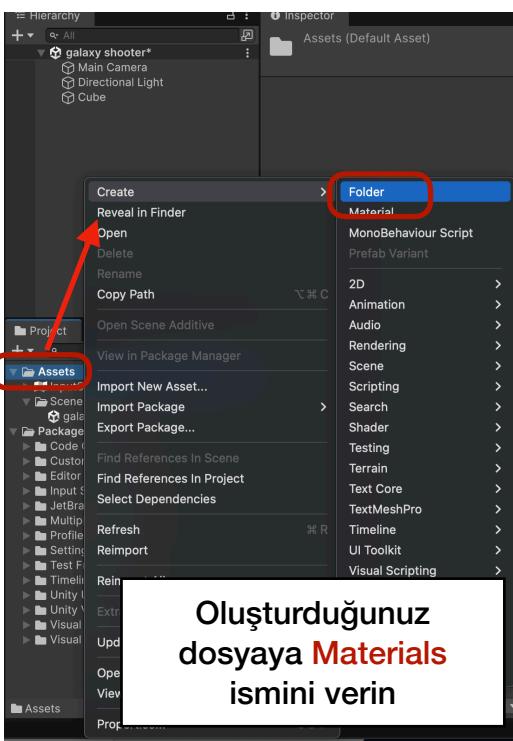
sahneye nesne ekleme (küp)



Hierarchy penceresinde galaxy shooter sahnesi içerisinde sağ tık ile nesneler ekleyebiliriz.

Örneğin küp için 3D objects->Cube seçilir, Scene içine gelecektir.

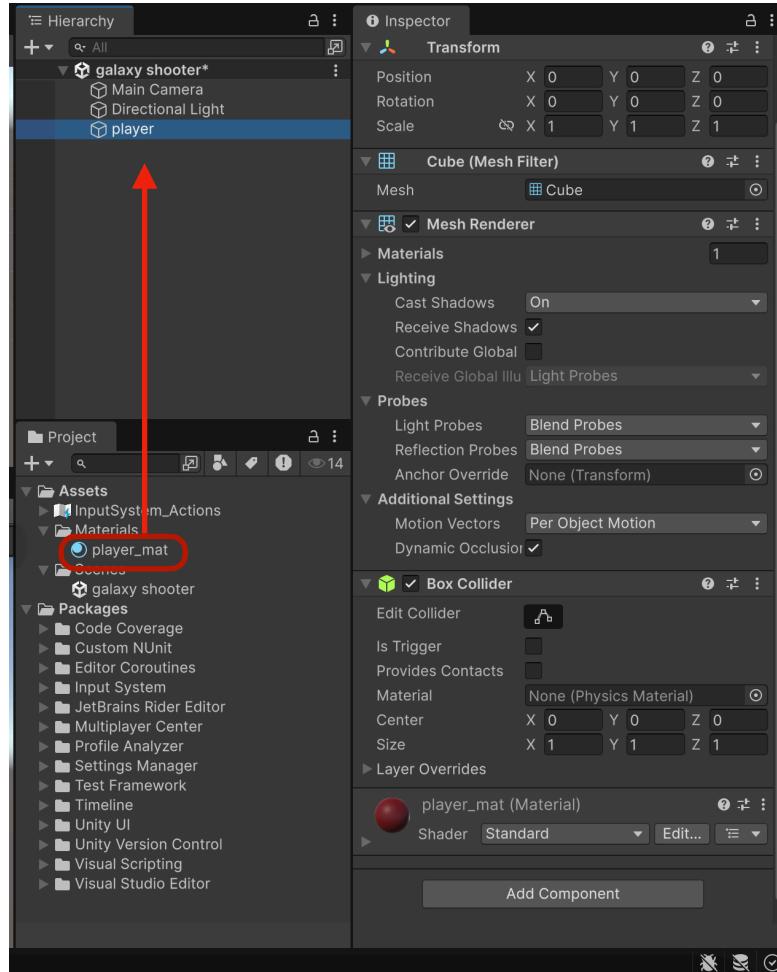
sahnedeki nesneye materyal ekleme



Assets klasörü içinde yeni bir klasör açın ve ismini Materials koyun. Daha sonra klasör içerisinde sağ tık ile Create->material diyerek yeni bir material ekleyin. Materyalleri isimlendirirken example_mat şeklinde koymaya özen gösterin.

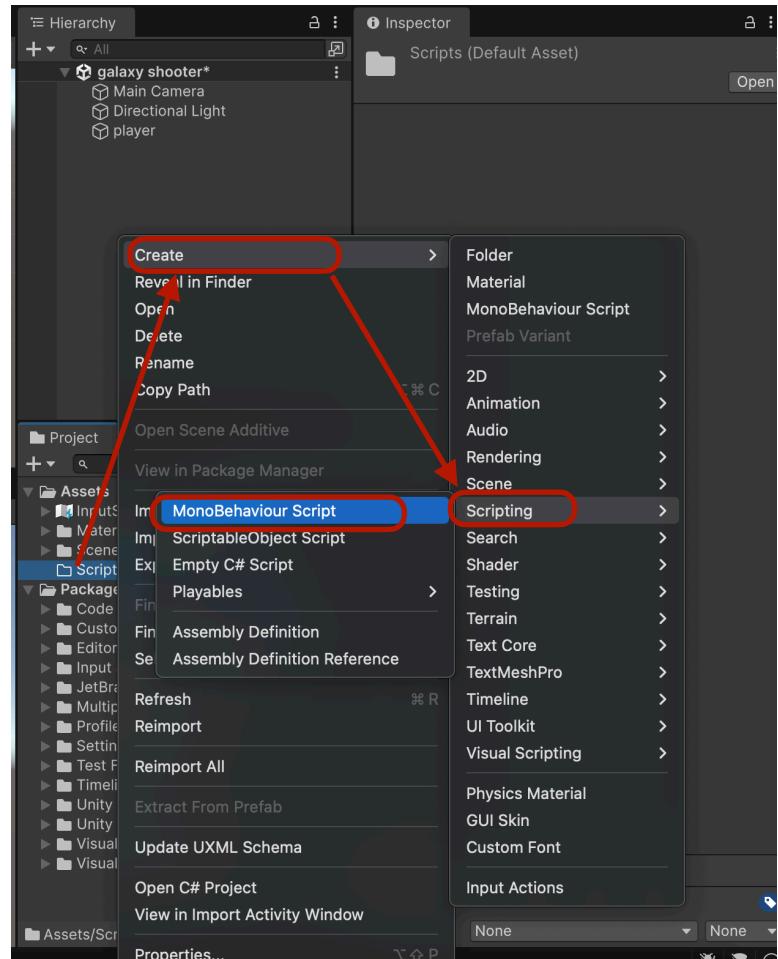
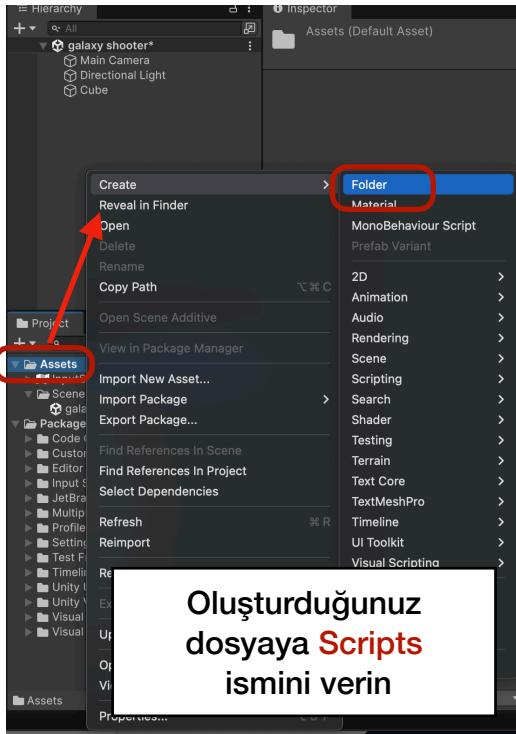


Materyalin özelliklerini inspector kısmında görebilir, Albedo yardımıyla rengini değiştirebilirsiniz.



Daha sonrasında materyali nesnenin üzerine sürükle bırak yaparak nesneye uygulayabilirsiniz.

sahnedeki nesneye script ekleme



Asset klasörü içine Script adında yeni bir dosya oluşturun ve bu dosya içerisine.Create->Scripting->monoBehaviorScript ile yeni bir script ekleyin.

Scriptlerin adını example_sc olarak koymaya özen gösterin.

Daha sonrasında scripti sürükle bırak yöntemi ile nesnenin üzerine bırakın. (Aksi halde scripte yazdığımız hareketler nesne üzerinde görünmeyecektir.)

script ile nesnenin konumunu bir kez değiştirme (start fonksiyonu)

The screenshot shows the Unity Editor's code editor with a file named 'player_sc.cs'. The code is as follows:

```
using UnityEngine;
public class NewMonoBehaviourScript : MonoBehaviour
{
    void Start() // sadece ilk frame'de bir kez çalışır
    {
        transform.position = new Vector3(-2, 0, 0); // nesnenin pozisyon bilgisini değiştirir.
        // Not: 3 boyutlu bir oyun olduğundan dolayı Vector3 kullanılır.
    }
}
```

`transform.position` ile nesnenin yeri `x` ekseninde -2 birim sola değiştirilir.
(`transform:` nesnenin sahnedeki dönüşüm bilgilerini tutar,
`position :nesnenin uzaydaki 3d durumu , new vector3: yeni konum)`

script ile nesnenin konumunu sürekli değiştirme (update fonksiyonu ile tek yönde ilerleme)

zamanın normalizasyonu (nesnenin saniyede bir birim ilerlemesi)

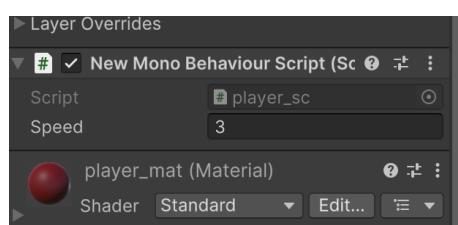
```
12 // Update is called once per frame
13 // 0 references
14 void Update()// her frame'de çalışır
15 {
16     transform.Translate(Vector3.right); // transform.Translate( new Vector3(1, 0, 0)) ile aynı işlevdedir.
17     // nesneyi x ekseninde +1 birim sağa kaydırır. Ancak bu çok hızlı gerçekleşir. Bu yüzden de alttaki kod kullanılabilir.
18
19     transform.Translate(Vector3.right * Time.deltaTime); // hareketi frame hızından bağımsız hale getirir. saniye bazında hareketi sağlar.
20
21
22
23
24
25 }
26
27 }
```

Fonksiyonumuz update fonksiyonu içine yazıldığı için sürekli çalışacak.

(`transform.Translate(Vector3.right):`Nesneyi `x` ekseninde 1 birim sağa kaydırır. `Vector3.right`, `(1, 0, 0)` yön vektörünü temsil eder,
`transform.Translate(Vector3.right * Time.deltaTime):` Nesneyi her frame'de sağa doğru hareket ettirir. `Time.deltaTime` kullanıldığı için hareket, frame hızından bağımsız olarak sabit hızda gerçekleşir.)

speed değişkeni tanımlama (public, private farkı)

```
1 using UnityEngine;
2
3 public class NewMonoBehaviourScript : MonoBehaviour
4 {
5
6     public int speed = 3; // public tanımlandığı için her yerden erişilebilir, değiştirilebilir.
7     private int speed2 = 3; // private tanımlandığı için sınıf içinde kullanılır, inspector'de gözükmey.
8
9 }
```



Public tanımlanan değişkenler inspector penceresinde gözükmek ve değiştirilebilirken , private tanımlananlar sadece kendi sınıfında gözükmek.

klavyeden yön tuşları ile nesnenin hareketinin kontrolü (dikey eksen) ve

Bonus: yatay eksenin hareket kontrolüne dahil edilmesi

```
0 references
void Update()// her frame'de çalışır
{
    transform.Translate(Vector3.right); // transform.Translate( new Vector3(1, 0, 0)) ile aynı işlevdedir.
    // nesneyi x ekseninde +1 birim sağa kaydırır. Ancak bu çok hızlı gerçekleşir. Bu yüzden de alttaki kod kullanılabilir.
    transform.Translate(Vector3.right * Time.deltaTime); // hareketi frame hızından bağımsız hale getirir. saniye bazında hareketi sağlar.

    transform.Translate(new Vector3(Input.GetAxis("Horizontal"), Input.GetAxis("Vertical"), 0) * Time.deltaTime * speed);
    // klavyeden alınan bilgilerle nesnenin hareketini sağlar.

}
```

klavyeden alınan kullanıcı girişine (Input.GetAxis("Horizontal")) sağ/sol tuşları ve Input.GetAxis("Vertical") yukarı/aşağı tuşlarına göre nesneyi hareket ettirir. time.deltaTime ile hareketin saniye bazında olmasını sağlar ve speed değişkeni ile hareket hızını ayarlar.

(Not: Input.GetAxis("") içine yazılacak tuş isimlerinin doğru olduğundan emin olmalıyız.)