BURSA TEKNİK ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ (T) BÖLÜMÜ

BİLGİSAYAR OYUNLARDA YAPAY ZEKA

Ödev-9 Raporu

Github: https://github.com/MoussaBane/BOYZ-QLearning-BallBalance

MOUSSA BANE

24435004029

1. **Proje Kurulumu ve Başlangıç :**
   - Unity Hub üzerinden yeni bir 3D proje oluşturun ve adını BallBalanceProject olarak belirleyin.
   - BallBalanceStarter projesini indirin ve projenize içe aktarın.
   - Scenes klasöründen "BallBalance" sahnesini açın.
2. **Ajanın (Platformun) Davranışı :**
   - **Gözlem**: Ajan her iterasyonda topun mevcut durumunu (konum, hız, eğim) gözlemler.
   - **Karar**: Bu duruma göre, platformun sağa veya sola eğilmesi gibi bir aksiyon belirler.

- **Güncelleme**: Eğer top düşerse, ajan bu hatasından öğrenir ve gelecekteki aksiyonlarını bu doğrultuda düzenler (Q-learning tablosu veya yapay sinir ağı ile).

3. Oyun Mekanikleri :

Sahnedeki Elemanlar

- **Platform**: Topun dengede kalacağı alan.
- **Border**: Platformun sınırları (görünmez fakat çarpışma algılaması için etkin).
- **DroppedZone**: Top düştüğünde tespitin yapıldığı alan.

Border ve DroppedZone Özellikleri

- **Mesh Renderer**: Disabled (görünmez).
- **Tag**: "drop" (düşme olaylarını algılamak için etiket).

Eğer top platform sınırlarını aşar ve "DroppedZone" bölgesine düşerse, bu bir başarısızlık olarak algılanır. Oyun yeniden başlar ve ajan, önceki denemelerdeki bilgileri kullanarak daha iyi aksiyonlar almaya çalışır.

4. Scriptlerin Yazılması ve İşlevleri :

BallState Scripti

Topun düşüşünü algılamak için oluşturulan script, bu scripti topla ilişkilendirin:

BallState_sc.cs | Brain_sc.cs

Assets > BallState_sc.cs

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class BallState_sc : MonoBehaviour
6   {
7       public bool dropped = false;
8
9       void OnCollisionEnter(Collision col)
10      {
11          if (col.gameObject.tag == "drop")
12          {
13              dropped = true;
14          }
15      }
16  }
```

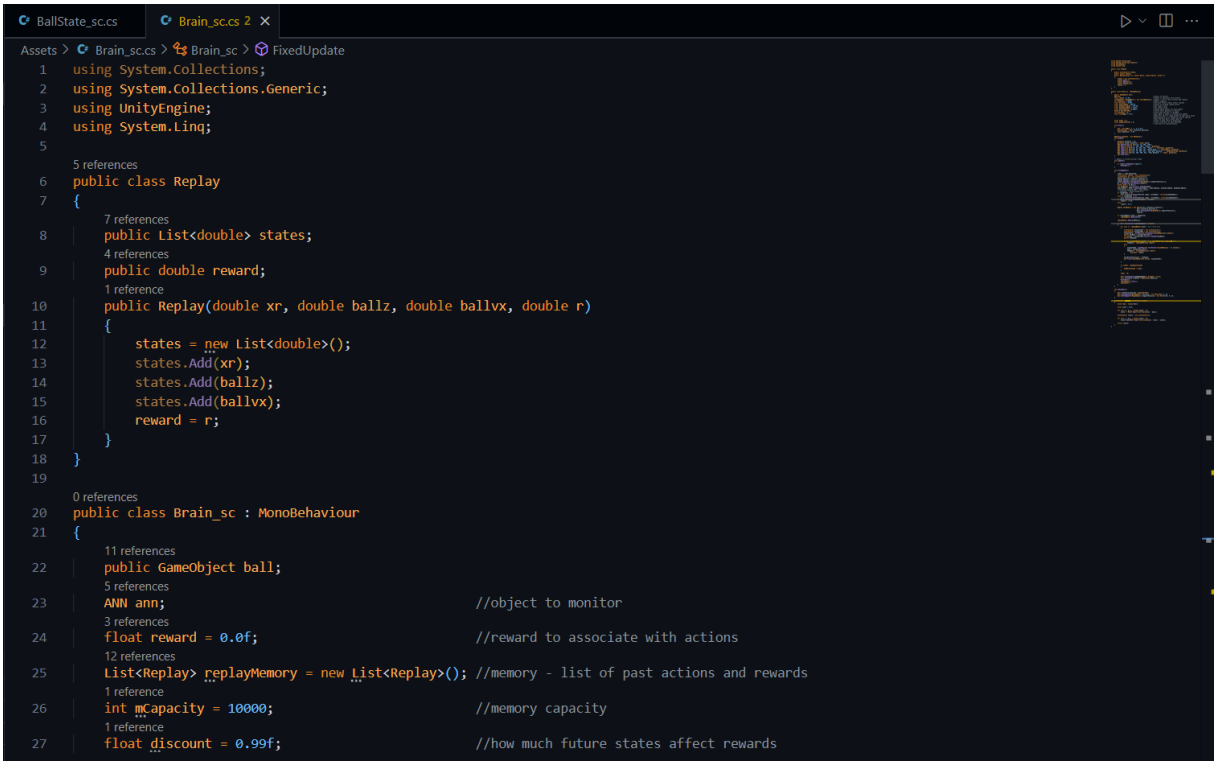
Bu script, "drop" etiketi ile temas eden topu tespit eder.

Brain Scripti

Platformun kontrolünü ve yapay sinir ağını yöneten script, bu scripti platformla ilişkilendirin:

Temel Özellikler:

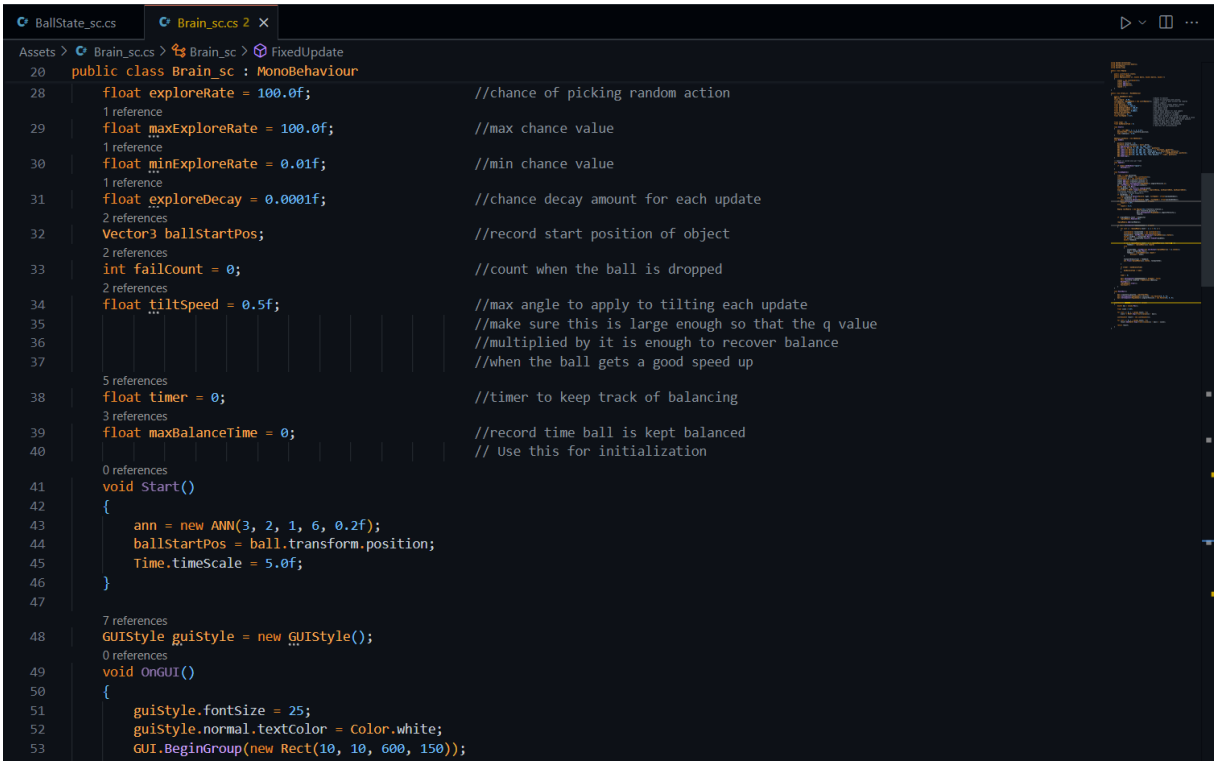
- **Girdiler:** Topun Z konumu, X hızı, platformun X rotasyonu.
- **Çıktılar:** Platformun sağa veya sola eğilmesi.



```
BallState_sc.cs | Brain_sc.cs 2
Assets > Brain_sc.cs > Brain_sc > FixedUpdate
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;

5 references
public class Replay
{
    7 references
    public List<double> states;
    4 references
    public double reward;
    1 reference
    public Replay(double xr, double ballz, double ballvx, double r)
    {
        states = new List<double>();
        states.Add(xr);
        states.Add(ballz);
        states.Add(ballvx);
        reward = r;
    }
}

0 references
public class Brain_sc : MonoBehaviour
{
    11 references
    public GameObject ball;
    5 references
    ANN ann;                                    //object to monitor
    3 references
    float reward = 0.0f;                        //reward to associate with actions
    12 references
    List<Replay> replayMemory = new List<Replay>(); //memory - list of past actions and rewards
    1 reference
    int mCapacity = 10000;                      //memory capacity
    1 reference
    float discount = 0.99f;                     //how much future states affect rewards
```



```
BallState_sc.cs | Brain_sc.cs 2
Assets > Brain_sc.cs > Brain_sc > FixedUpdate
public class Brain_sc : MonoBehaviour
    float exploreRate = 100.0f;                 //chance of picking random action
    1 reference
    float maxExploreRate = 100.0f;              //max chance value
    1 reference
    float minExploreRate = 0.01f;               //min chance value
    1 reference
    float exploreDecay = 0.0001f;               //chance decay amount for each update
    2 references
    Vector3 ballStartPos;                       //record start position of object
    2 references
    int failCount = 0;                          //count when the ball is dropped
    2 references
    float tiltSpeed = 0.5f;                     //max angle to apply to tilting each update
                                                //make sure this is large enough so that the q value
                                                //multiplied by it is enough to recover balance
                                                //when the ball gets a good speed up
    5 references
    float timer = 0;                            //timer to keep track of balancing
    3 references
    float maxBalanceTime = 0;                   //record time ball is kept balanced
                                                // Use this for initialization
    0 references
    void Start()
    {
        ann = new ANN(3, 2, 1, 6, 0.2f);
        ballStartPos = ball.transform.position;
        Time.timeScale = 5.0f;
    }

    7 references
    GUIStyle guiStyle = new GUIStyle();
    0 references
    void OnGUI()
    {
        guiStyle.fontSize = 25;
        guiStyle.normal.textColor = Color.white;
        GUI.BeginGroup(new Rect(10, 10, 600, 150));
```

BallState_sc.cs | Brain_sc.cs 2

Assets > Brain_sc.cs > Brain_sc > FixedUpdate

```
20  public class Brain_sc : MonoBehaviour
49      void OnGUI()
54          GUI.Box(new Rect(0, 0, 140, 140), "Stats", guiStyle);
55          GUI.Label(new Rect(10, 25, 500, 30), "Fails: " + failCount, guiStyle);
56          GUI.Label(new Rect(10, 50, 500, 30), "Decay Rate: " + exploreRate, guiStyle);
57          GUI.Label(new Rect(10, 75, 500, 30), "Last Best Balance: " + maxBalanceTime, guiStyle);
58          GUI.Label(new Rect(10, 100, 500, 30), "This Balance: " + timer, guiStyle);
59          GUI.EndGroup();
60      }
61
62      // Update is called once per frame
        0 references
63      void Update()
64      {
65          if (Input.GetKeyDown("space"))
66              ResetBall();
67      }
68
        0 references
69      void FixedUpdate()
70      {
71          timer += Time.deltaTime;
72          List<double> states = new List<double>();
73          List<double> qs = new List<double>();
74          states.Add(this.transform.rotation.x);
75          states.Add(ball.transform.position.z);
76          states.Add(ball.GetComponent<Rigidbody>().angularVelocity.x);
77          qs = SoftMax(ann.CalcOutput(states));
78          double maxQ = qs.Max();
79          int maxQIndex = qs.ToList().IndexOf(maxQ);
80          exploreRate = Mathf.Clamp(exploreRate - exploreDecay, minExploreRate, maxExploreRate);
81          //if(Random.Range(0,100) < exploreRate)
82          // maxQIndex = Random.Range(0,2);
83          if (maxQIndex == 0)
84              this.transform.Rotate(Vector3.right, tiltSpeed * (float)qs[maxQIndex]);
85          else if (maxQIndex == 1)
86              this.transform.Rotate(Vector3.right, -tiltSpeed * (float)qs[maxQIndex]);
```

BallState_sc.cs | Brain_sc.cs 2

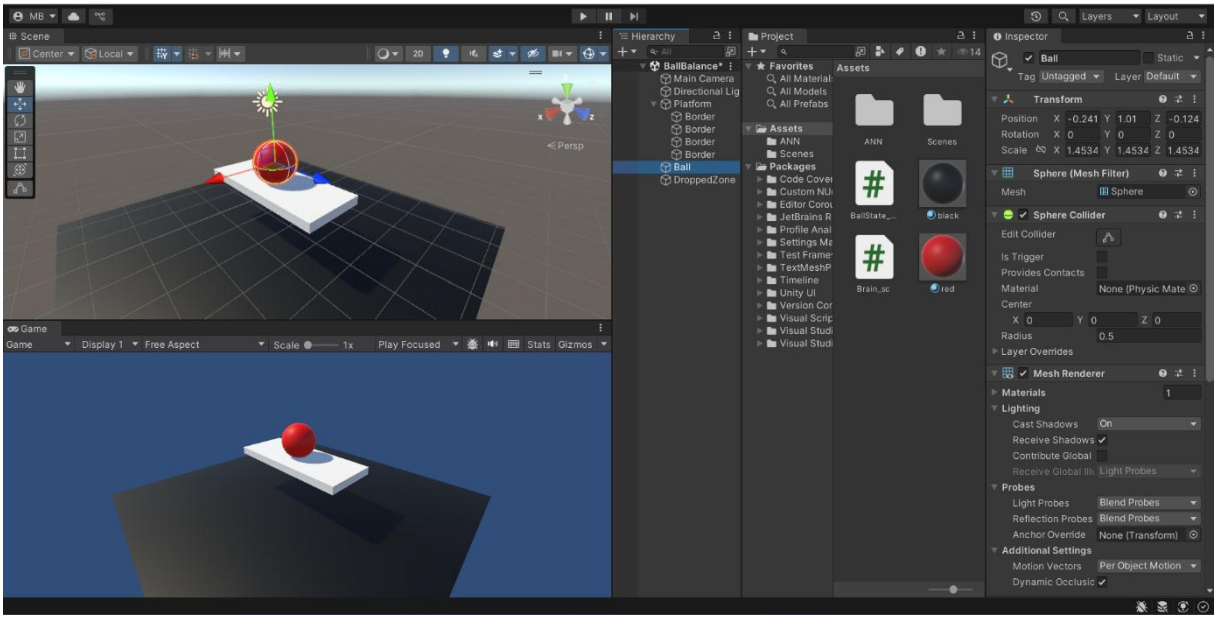
Assets > Brain_sc.cs > Brain_sc > FixedUpdate

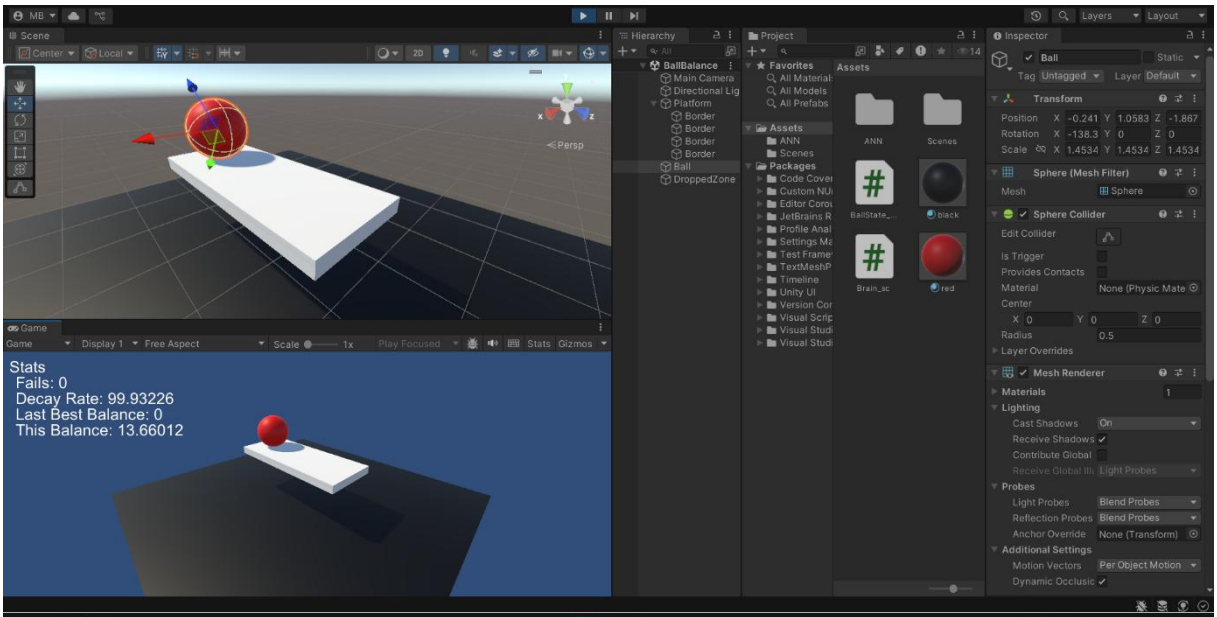
```
20  public class Brain_sc : MonoBehaviour
69      void FixedUpdate()
87          if (ball.GetComponent<BallState_sc>().dropped)
88              reward = -1.0f;
89          else
90              reward = 0.1f;
91
92          Replay lastMemory = new Replay(this.transform.rotation.x,
93                                         ball.transform.position.z,
94                                         ball.GetComponent<Rigidbody>().angularVelocity.x,
95                                         reward);
96
97          if (replayMemory.Count > mCapacity)
98              replayMemory.RemoveAt(0);
99
100         replayMemory.Add(lastMemory);
101
102         if (ball.GetComponent<BallState_sc>().dropped)
103         {
104             for (int i = replayMemory.Count - 1; i >= 0; i--)
105             {
106                 List<double> toutputsOld = new List<double>();
107                 List<double> toutputsNew = new List<double>();
108                 toutputsOld = SoftMax(ann.CalcOutput(replayMemory[i].states));
109                 double maxQOld = toutputsOld.Max();
110                 int action = toutputsOld.ToList().IndexOf(maxQOld);
111                 double feedback;
112
113                 if (i == replayMemory.Count - 1 || replayMemory[i].reward == -1)
114                     feedback = replayMemory[i].reward;
115                 else
116                 {
117                     toutputsNew = SoftMax(ann.CalcOutput(replayMemory[i + 1].states));
118                     maxQ = toutputsNew.Max();
119                     feedback = (replayMemory[i].reward +
120                         discount * maxQ);
121                 }
```

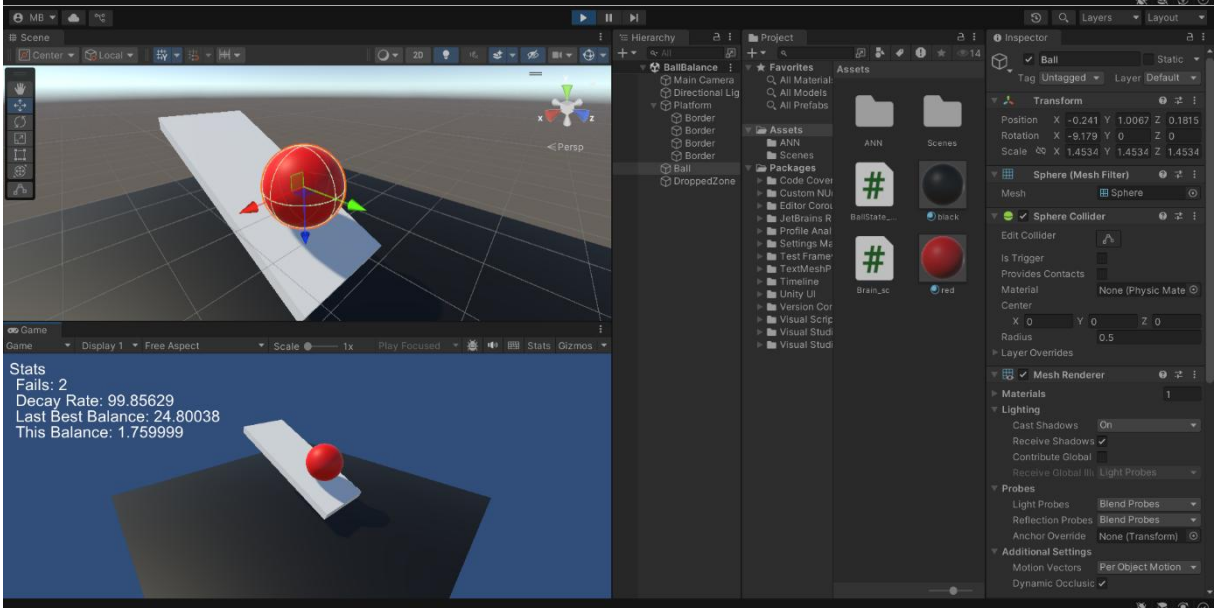
```
BallState_sc.cs | Brain_sc.cs 2
Assets > Brain_sc.cs > Brain_sc > FixedUpdate
20  public class Brain_sc : MonoBehaviour
69      void FixedUpdate()
122
123                 toutputsOld[action] = feedback;
124                 ann.Train(replayMemory[i].states, toutputsOld);
125
126             }
127
128             if (timer > maxBalanceTime)
129             {
130                 maxBalanceTime = timer;
131             }
132
133             timer = 0;
134
135             ball.GetComponent<BallState_sc>().dropped = false;
136             this.transform.rotation = Quaternion.identity;
137             ResetBall();
138             replayMemory.Clear();
139             failCount++;
140         }
141     }
142
        2 references
143     void ResetBall()
144     {
145         ball.transform.position = ballStartPos;
146         ball.GetComponent<Rigidbody>().velocity = new Vector3(0, 0, 0);
147         ball.GetComponent<Rigidbody>().angularVelocity = new Vector3(0, 0, 0);
148     }
149
        3 references
150     List<double> SoftMax(List<double> values)
151     {
152         double max = values.Max();
153
154         float scale = 0.0f;
```

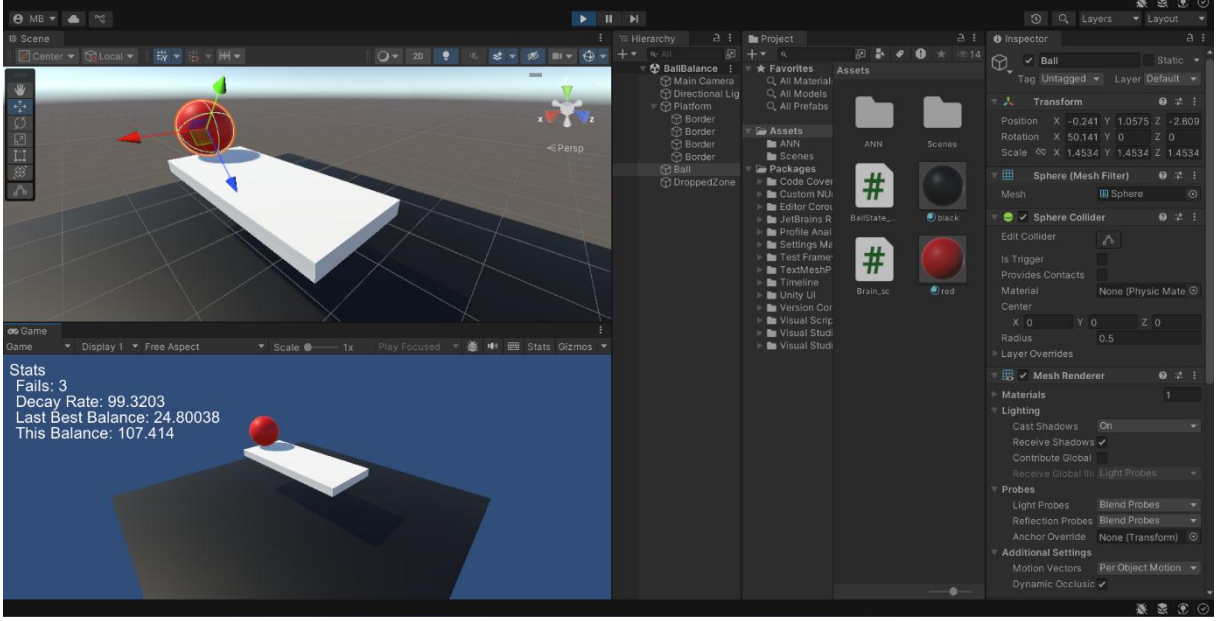
```
BallState_sc.cs | Brain_sc.cs 2
Assets > Brain_sc.cs > Brain_sc > FixedUpdate
20  public class Brain_sc : MonoBehaviour
150     List<double> SoftMax(List<double> values)
155
156         for (int i = 0; i < values.Count; ++i)
157             scale += Mathf.Exp((float)(values[i] - max));
158
159         List<double> result = new List<double>();
160
161         for (int i = 0; i < values.Count; ++i)
162             result.Add(Mathf.Exp((float)(values[i] - max)) / scale);
163
164         return result;
165     }
166 }
```

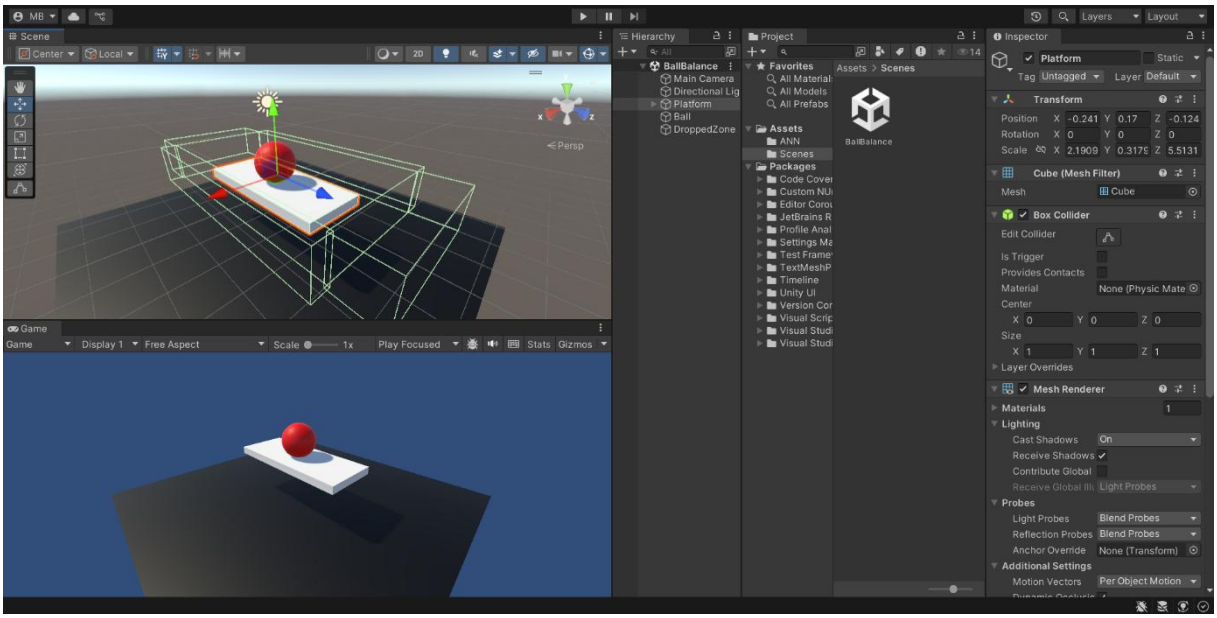
5. Oynun Çalıştırılmış Hali :




Stats
Fails: 0
Decay Rate: 99.93226
Last Best Balance: 0
This Balance: 13.66012


Stats
Fails: 2
Decay Rate: 99.85629
Last Best Balance: 24.80038
This Balance: 1.759999


Stats
Fails: 3
Decay Rate: 99.3203
Last Best Balance: 24.80038
This Balance: 107.414


Scene
Hierarchy
Project
Inspector
Game
BallBalance
Main Camera
Directional Lig
Platform
Ball
DroppedZone
Persp
Platform
Static
Tag Untagged
Layer Default
Transform
Position X -0.241 Y 0.17 Z -0.124
Rotation X 0 Y 0 Z 0
Scale X 2.1909 Y 0.3178 Z 5.5131
Cube (Mesh Filter)
Mesh Cube
Box Collider
Edit Collider
Is Trigger
Provides Contacts
Material None (Physic Mate
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1
Layer Overrides
Mesh Renderer
Materials 1
Lighting
Cast Shadows On
Receive Shadows
Contribute Global
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Motion Vectors Per Object Motion