BURSA TEKNİK ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ (T) BÖLÜMÜ

BİLGİSAYAR OYUNLARDA YAPAY ZEKA

Ödev-7 Raporu

Github: https://github.com/MoussaBane/BOYZ-Ping-Pong-ANN

MOUSSA BANE

24435004029

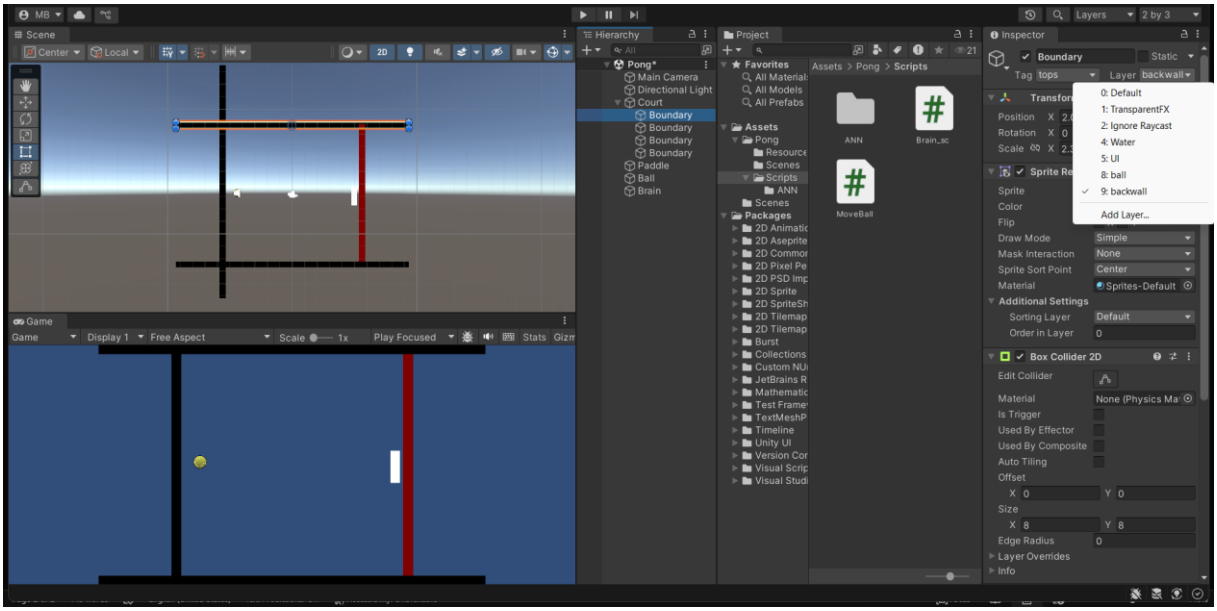
Giriş:

Bu rapor, "**Yapay Zekâ Destekli Pong Oyunu**" projesinde gerçekleştirdiğimiz adımları ve kullanılan kodları detaylı bir şekilde açıklamak için hazırlanmıştır. Unity'yi ilk kez kullanan bir kişinin de projeyi kolayca tekrarlayabilmesi için adım adım ve detaylı açıklamalar içermektedir.

Proje Hazırlıkları:

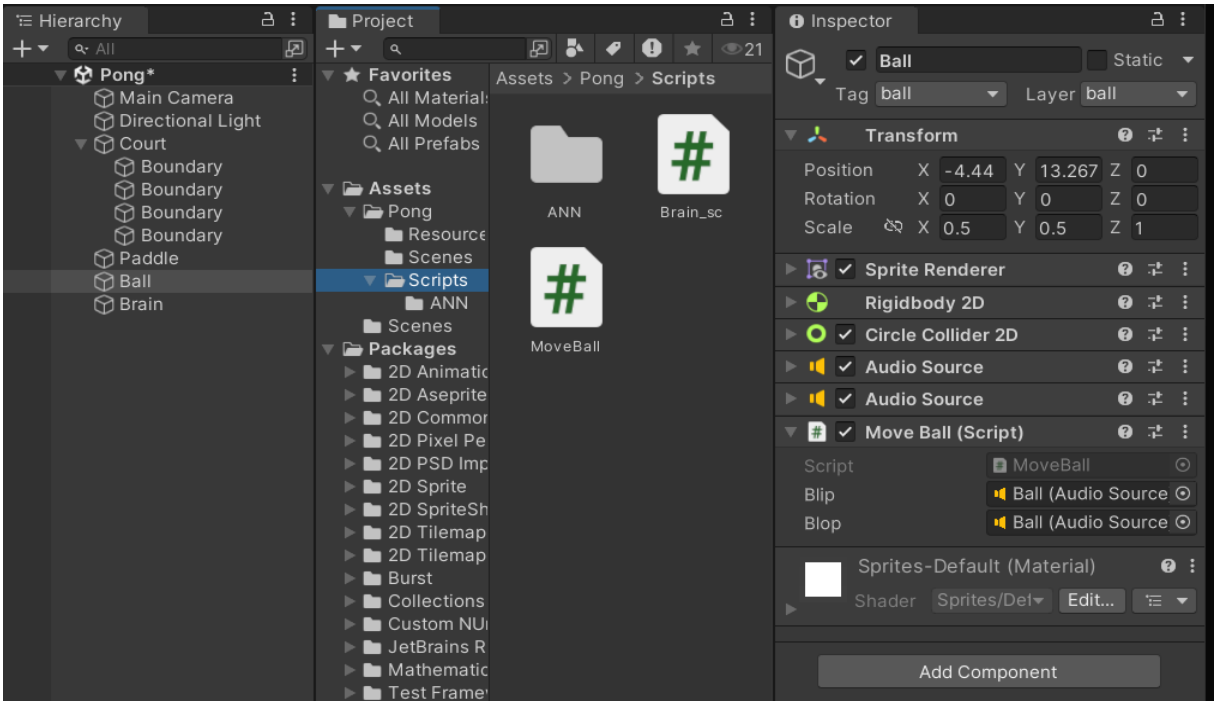
1. **Yeni 2D Proje Oluşturma :**
   - ✓ Unity'yi açarak yeni bir 2D proje oluşturun.
   - ✓ **PongStarter2022** dosyalarını Unity'ye içe aktarın.
   - ✓ **Pong** sahnesini açın.
2. **Sahne Düzeni :**
   - ✓ İki yatay (üst ve alt) ve iki dikey (sol ve sağ) oyun nesnesi ekleyin. Bu nesneler, topun hareket edebileceği sınırları belirler.

- ✓ Yatay nesnelere BoxCollider2D ve RigidBody2D bileşenleri mevcuttur.
- ✓ Dikey nesnelere aynı şekilde bileşenleri mevcut ve kırmızı renkli olan nesnenin etiketi backwall olarak ayarlandı.
- ✓ **Katman Ayarları**: Eğer mevcut değilse, 8 ve 9 numaralı katmanları ekleyin ve nesneleri uygun katmanlara atayın.



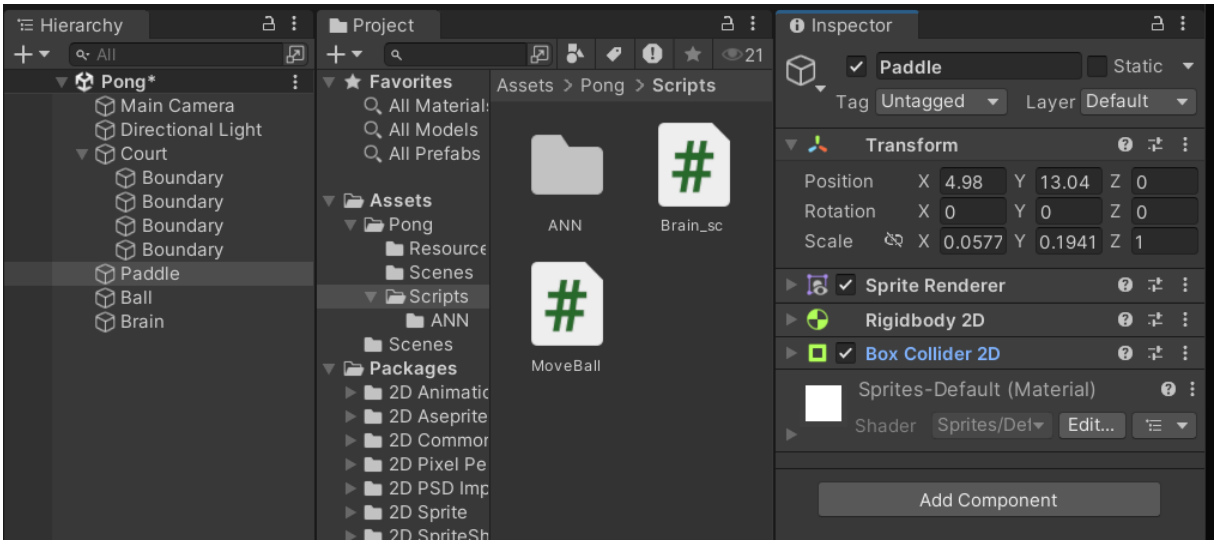
Top ve Paddle Ayarları :

1. **Top Nesnesi :**
   - ✓ Top nesnesine RigidBody2D ve CircleCollider2D bileşenlerini mevcut.
   - ✓ CircleCollider2D bileşenine Bounce isimli fizik materyali atandı. Bu, topun çarpışmalardan sekmesini sağlar.



2. **Paddle (Raket) :**

   ✓ Paddle nesnesine BoxCollider2D ve RigidBody2D bileşenleri mevcut. Hareket yalnızca Y ekseninde sınırlandırıldı.



MoveBall Scripti:

Bu script, topun hareketini ve çarpışmalarını kontrol eder.

✓ Top başlangıç pozisyonuna dönmesi için ResetBall fonksiyonu kullanılır.

✓ Topun backwall ve diğer nesnelere çarpması durumunda ses efektleri (blip ve blop) çalınır.

✓ space tuşuna basıldığında top yeniden başlatılır.

Kod Parçası:

```
0 references
public class MoveBall : MonoBehaviour
{
    // Stores the initial position of the ball
    2 references
    Vector3 ballStartPosition;

    // Reference to the Rigidbody2D component of the ball
    3 references
    Rigidbody2D rb;

    // Speed of the ball when launched
    1 reference
    float speed = 400;

    // Audio sources for sound effects when the ball hits different objects
    1 reference
    public AudioSource blip; // Played when hitting anything other than the backwall
    1 reference
    public AudioSource blop; // Played when hitting the backwall

    // Called when the script instance is loaded
    0 references
    void Start()
    {
        // Get the Rigidbody2D component attached to this GameObject
        rb = this.GetComponent<Rigidbody2D>();

        // Save the initial position of the ball for resetting purposes
        ballStartPosition = this.transform.position;

        // Initialize the ball's state
        ResetBall();
    }
```

```
33      // Called when the ball collides with another GameObject
        0 references
34      void OnCollisionEnter2D(Collision2D col)
35      {
36          // Check if the ball collided with the object tagged as "backwall"
37          if (col.gameObject.tag == "backwall")
38              blop.Play(); // Play the "blop" sound effect
39          else
40              blip.Play(); // Play the "blip" sound effect for other collisions
41      }
42
43      // Resets the ball's position and launches it in a random direction
        2 references
44      public void ResetBall()
45      {
46          // Reset the ball's position to its initial position
47          this.transform.position = ballStartPosition;
48
49          // Stop any current movement of the ball
50          rb.velocity = Vector3.zero;
51
52          // Generate a random direction for the ball
53          Vector3 dir = new Vector3(Random.Range(100, 300), Random.Range(-100, 100), 0).normalized;
54
55          // Apply a force to the ball to launch it
56          rb.AddForce(dir * speed);
57      }
```

```
59      // Called once per frame
        0 references
60      void Update()
61      {
62          // Check if the "space" key is pressed
63          if (Input.GetKeyDown("space"))
64          {
65              // Reset the ball when the space key is pressed
66              ResetBall();
67          }
68      }
69  }
```

Brain Scripti :

Yapay zekâ (ANN) ile paddle hareketlerini kontrol etmek için geliştirilmiştir.

1. **Girdi Değerleri (Input):**
   - ✓ Topun pozisyonu ve hızı (bx, by, bvx, bvy).
   - ✓ Paddle pozisyonu (px, py).
2. **Çıktı Değeri (Output):**
   - ✓ Paddle'ın Y eksenindeki hızı (pv).
3. **Raycast Kullanımı:**
   - ✓ Topun yörüngesini analiz etmek ve paddle ile çarpışma olasılığını hesaplamak için kullanılır.
4. **Yapay Sinir Ağı (ANN) Ayarları:**

- ✓ Gizli katman: 1
- ✓ Nöron sayısı: 4
- ✓ Aktivasyon fonksiyonu: tanh (-1 ile 1 arası değer döner).

Kod Parçası:

```
5   public class Brain_sc : MonoBehaviour
6   {
        7 references
7       public GameObject paddle; // Reference to the paddle GameObject
        4 references
8       public GameObject ball;   // Reference to the ball GameObject
        5 references
9       Rigidbody2D brb;          // Rigidbody2D component of the ball
10
        3 references
11      float yvel;               // Current y-axis velocity for paddle movement
        1 reference
12      float paddleMinY = 8.8f;  // Minimum y-position for the paddle
        1 reference
13      float paddleMaxY = 17.4f; // Maximum y-position for the paddle
        1 reference
14      float paddleMaxSpeed = 15; // Maximum speed at which the paddle can move
15
        0 references
16      public float numSaved = 0; // Counter for the number of balls successfully intercepted
        0 references
17      public float numMissed = 0; // Counter for the number of balls missed
18
        3 references
19      ANN ann; // Artificial Neural Network instance
20
21      // Called when the script is initialized
        0 references
22      void Start()
23      {
24          // Initialize the neural network with 6 input neurons, 1 output neuron,
25          // 1 hidden layer, 4 neurons in the hidden layer, and a learning rate of 0.11
26          ann = new ANN(6, 1, 1, 4, 0.11);
27
28          // Get the Rigidbody2D component of the ball
29          brb = ball.GetComponent<Rigidbody2D>();
30      }
```

```
5   public class Brain_sc : MonoBehaviour
31
32      // Runs the neural network with inputs and expected outputs
33      // Parameters:
34      // - bx, by: Ball's x and y positions
35      // - bvx, bvy: Ball's x and y velocities
36      // - px, py: Paddle's x and y positions
37      // - pv: Expected paddle velocity
38      // - train: Whether the network should train on this input/output pair
        1 reference
39      List<double> Run(double bx, double by, double bvx, double bvy, double px, double py, double pv, bool train)
40      {
41          List<double> inputs = new List<double>();  // Neural network inputs
42          List<double> outputs = new List<double>(); // Neural network expected outputs
43
44          // Add inputs
45          inputs.Add(bx);
46          inputs.Add(by);
47          inputs.Add(bvx);
48          inputs.Add(bvy);
49          inputs.Add(px);
50          inputs.Add(py);
51
52          // Add expected output
53          outputs.Add(pv);
54
55          // Train the network if specified, otherwise calculate the output
56          if (train)
57              return ann.Train(inputs, outputs); // Train and return error
58          else
59              return ann.CalcOutput(inputs, outputs); // Calculate the output
60      }
```

```
5   public class Brain_sc : MonoBehaviour
63      void Update()
64      {
65          // Calculate the new y-position of the paddle based on the velocity
66          float posy = Mathf.Clamp(
67              paddle.transform.position.y + (yvel * Time.deltaTime * paddleMaxSpeed),
68              paddleMinY,
69              paddleMaxY);
70
71          // Update the paddle's position
72          paddle.transform.position = new Vector3(
73              paddle.transform.position.x,
74              posy,
75              paddle.transform.position.z);
76
77          // List to hold the neural network output
78          List<double> output = new List<double>();
79
80          // Raycasting to predict the ball's trajectory
81          int layerMask = 1 << 9; // Only detect objects in layer 9
82          RaycastHit2D hit = Physics2D.Raycast(ball.transform.position, brb.velocity, 1000, layerMask);
83
84          // If the raycast hits something
85          if (hit.collider != null)
86          {
87              // If it hits the "tops" tag, calculate the reflection
88              if (hit.collider.gameObject.tag == "tops")
89              {
90                  Vector3 reflection = Vector3.Reflect(brb.velocity, hit.normal);
91                  hit = Physics2D.Raycast(hit.point, reflection, 1000, layerMask);
92              }
```

```
5   public class Brain_sc : MonoBehaviour
63      void Update()
94              // If it hits the "backwall" tag
95              if (hit.collider != null && hit.collider.gameObject.tag == "backwall")
96              {
97                  // Calculate the vertical difference between the paddle and the predicted hit point
98                  float dy = (hit.point.y - paddle.transform.position.y);
99
100                 // Run the neural network to determine the paddle's velocity
101                 output = Run(
102                     ball.transform.position.x,
103                     ball.transform.position.y,
104                     brb.velocity.x,
105                     brb.velocity.y,
106                     paddle.transform.position.x,
107                     paddle.transform.position.y,
108                     dy,
109                     true);
110
111                 // Set the paddle's y-velocity to the neural network's output
112                 yvel = (float)output[0];
113             }
114         }
115         else
116         {
117             // If no object is hit, stop paddle movement
118             yvel = 0;
119         }
120     }
121 }
```

Test ve İyileştirme:

1. **Eğitim Seti Zenginleştirme:**

   ✓ Topun sınır nesnelerine çarpması durumunda raycast ile sonraki hareketleri analiz edilerek eğitim setine ek veri sağlandı.

2. **Öğrenme Hızının İyileştirilmesi:**

   ✓ Öğrenme hızı, farklı değerlerde test edilerek (ör. 0.001) en iyi sonuç gözlemlendi.

Sonuç :

Proje başarıyla tamamlanmış ve paddle yapay zekâ tarafından kontrol edilebilir hale getirilmiştir. Eğitim seti genişletildiğinde ve daha fazla veri sağlandığında, yapay zekâ performansının iyileştirilebileceği gözlemlenmiştir.

Çalıştırılmış Hali :