BURSA TEKNİK ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ (T) BÖLÜMÜ

BİLGİSAYAR OYUNLARDA YAPAY ZEKA

Ödev-3 Raporu

Github: https://github.com/MoussaBane/BOYZ-Genetik-Programlama-II

MOUSSA BANE

24435004029

Proje Tanımı

Bu proje, karakterlerin hayatta kalma süresine göre genetik kod aktarımı yaparak hareket yeteneklerini geliştirmeyi amaçlıyor. Başlangıçta oluşturulan 3D sahnede Ethan karakteri, belirli hareket şablonlarıyla yönlendirilecek ve kırmızı platform üzerinde başlayacak. Sahneye eklenen beyaz platform (DeadGround), "ölüm alanı" olarak tanımlanmış durumda ve etiketi “dead” olarak ayarlandı.

Karakterler, genetik algoritmalarla belirlenen hareket kalıplarını uygular ve en uzun süre hayatta kalanlar, genetik kodlarını bir sonraki nesle aktarır. Bu süreç, karakterlerin her nesilde daha yetenekli ve dayanıklı hale gelmesini sağlıyor. Böylece, proje sonunda hayatta kalma şansını en üst düzeye çıkaran hareket stratejilerini keşfetmiş oluyoruz.

Gereksinimler

- **Unity**: Projeyi geliştirmek için Unity oyun motoru.
- **C# Scriptleri**: Botların hareketi ve DNA yönetimi için C# dilinde yazılmış scriptler.
- **3D Model**: Bot prefab'ı olarak kullanılacak bir 3D model (önceden sağlanmış).

1. Yeni Proje Kurulumu ve Gereken Assetlerin İçe Aktarılması

- **Yeni 3D Proje Oluşturma**: Unity’yi açın ve yeni bir 3D proje oluşturun.
- **Ethan Asset’i Eklemek**: Unity Asset Store’a gidip Ethan karakterini projenize ekleyin. Ethan’ı, projeye dahil etmek için “Assets” klasöründen sahneye sürükleyin.

2. Sahne Düzeni – Platformların Eklenmesi

- **Beyaz ve Kırmızı Platformların Eklenmesi**:

- “GameObject” menüsünden yeni bir Cube oluşturun ve buna “DeadGround” adını verin. Renk seçiciden beyaz renge boyayın ve **etiketini** “dead” olarak ayarlayın.
- Aynı işlemi kırmızı platform için de yapın ancak rengini kırmızı yapın ve ismini değiştirmeyin.

- **Karakterlerin Başlangıç Pozisyonu**: Kırmızı platforma yerleştirerek karakterlerin başlangıç konumunu belirleyin.

3. DNA ve Brain Scriptleri ile Karakter Hareketi Kontrolü

- **DNA_sc Script’i**: DNA’yı oluşturmak için “DNA_sc” adında bir script dosyası oluşturun. Kodda, hareket seçenekleri (ileri, geri, sola, sağa, zıplama ve çömelme) genetik kod olarak belirlenmiştir. Kodları yukarıdaki gibi ekleyin ve her bir fonksiyonun üzerine yorumlar ekleyin.

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

Assets > Scripts > DNA_sc.cs > ...

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
   9 references
5  public class DNA_sc
6  {
       12 references
7      public List<int> genes = new List<int>();
       7 references
8      public int dnaLength;
       4 references
9      public int maxValue;
10
       2 references
11     public DNA_sc(int length, int max)
12     {
13         dnaLength = length;
14         maxValue = max;
15         SetRandom();
16     }
17
18     // Set random values for DNA genes
       1 reference
19     public void SetRandom()
20     {
21         genes.Clear();
22         for (int i = 0; i < dnaLength; i++)
23         {
24             genes.Add(Random.Range(0, maxValue));
25         }
26     }
```

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

Assets > Scripts > DNA_sc.cs > ...

```
5   public class DNA_sc
        0 references
29      public void SetInt(int pos, int value)
30      {
31          genes[pos] = value;
32      }
33
34      // Retrieve a specific gene value
        6 references
35      public int GetInt(int pos)
36      {
37          return genes[pos];
38      }
39
40      // Combine two DNA sets from two parents
        1 reference
41      public void Combine(DNA_sc d1, DNA_sc d2)
42      {
43          for (int i = 0; i < dnaLength; i++)
44          {
45              // Combine half genes from each parent
46              genes[i] = i < dnaLength / 2 ? d1.genes[i] : d2.genes[i];
47          }
48      }
49
50      // Mutation method (note: add mutation rate as parameter)
        1 reference
51      public void Mutate(float mutationRate)
52      {
53          for (int i = 0; i < genes.Count; i++)
54          {
55              if (Random.Range(0f, 1f) < mutationRate)
56              {
57                  genes[i] = Random.Range(0, maxValue);
58              }
59          }
60      }
```

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

Assets > Scripts > DNA_sc.cs > ...

```
5   public class DNA_sc
62      // Cross over DNA with a partner
        0 references
63      public DNA_sc Crossover(DNA_sc partner)
64      {
65          DNA_sc child = new DNA_sc(dnaLength, maxValue);
66          int midpoint = Random.Range(0, dnaLength);
67
68          // Genes from both parents up to midpoint
69          for (int i = 0; i < dnaLength; i++)
70          {
71              child.genes[i] = (i > midpoint) ? genes[i] : partner.genes[i];
72          }
73
74          return child;
75      }
76  }
```

- **Brain_sc Script'i**: Karakterin hareketlerini kontrol etmek için "Brain_sc" adlı bir script dosyası oluşturun. Bu kod, karakterin hareketini genetik koda göre yönlendirmektedir.
- **Movement Script'i Eklemek**: Ethan üzerinde bulunan Third Person Character component'ine alternatif bir hareket script'i ekleyin.

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

Assets > Scripts > Brain_sc.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityStandardAssets.Characters.ThirdPerson;
5
6   [RequireComponent(typeof(ThirdPersonCharacter))]
    6 references
7   public class Brain_sc : MonoBehaviour
8   {
        1 reference
9       public int DNALength = 1;
        2 references
10      public float timeAlive;
        11 references
11      public DNA_sc dna_sc;
        2 references
12      private ThirdPersonCharacter m_Character;
        2 references
13      private Vector3 m_Move;
        3 references
14      private bool m_Jump;
        3 references
15      bool alive = true;
16
        2 references
17      public float distanceTravelled;
        2 references
18      Vector3 startPosition;
19
20      // Collision detection for death
        0 references
21      void OnCollisionEnter(Collision obj)
22      {
23          if (obj.gameObject.tag == "dead")
24          {
25              alive = false;
26          }
27      }
```

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

Assets > Scripts > Brain_sc.cs > ...

```
7   public class Brain_sc : MonoBehaviour
29      // Initialize DNA and components
        2 references
30      public void Init()
31      {
32          dna_sc = new DNA_sc(DNALength, 6);
33          m_Character = GetComponent<ThirdPersonCharacter>();
34          timeAlive = 0;
35          alive = true;
36          startPosition = this.transform.position;
37      }
38
        0 references
39      private void FixedUpdate()
40      {
41          float h = 0;
42          float v = 0;
43          bool crouch = false;
44
45          // Interpret DNA values for movement
46          if (dna_sc.GetInt(0) == 0) v = 1;
47          else if (dna_sc.GetInt(0) == 1) v = -1;
48          else if (dna_sc.GetInt(0) == 2) h = -1;
49          else if (dna_sc.GetInt(0) == 3) h = 1;
50          else if (dna_sc.GetInt(0) == 4) m_Jump = true;
51          else if (dna_sc.GetInt(0) == 5) crouch = true;
52
53          m_Move = v * Vector3.forward + h * Vector3.right;
54          m_Character.Move(m_Move, crouch, m_Jump);
55          m_Jump = false;
56
57          if (alive)
58          {
59              timeAlive += Time.deltaTime;
60              distanceTravelled = Vector3.Distance(this.transform.position, startPosition);
61          }
62      }
```

4. PopulationManager ile Nesil Yönetimi

- **PopulationManager_sc Script'i**: "PopulationManager_sc" adında yeni bir script oluşturun. Kodları yukarıdaki gibi ekleyin. Bu script, karakterlerin yaşam süresine göre yeni bir nesil oluşturarak genetik kodları birleştirir ve popülasyon yönetimini sağlar.
- **Popülasyon Boyutu ve Nesil Oluşumu**: Başlangıç popülasyon boyutunu ve her nesil için karakterlerin özelliklerini PopulationManager üzerinde tanımlayın.

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

Assets > Scripts > PopulationManager_sc.cs > ...

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using System.Linq;
5
    0 references
6   public class PopulationManager_sc : MonoBehaviour
7   {
        2 references
8       public GameObject botPrefab;
        1 reference
9       public int populationSize = 50;
        6 references
10      List<GameObject> population = new List<GameObject>();
        4 references
11      public static float elapsed = 0;
        1 reference
12      public float trialTime = 5;
        2 references
13      int generation = 1;
        6 references
14      GUIStyle guiStyle = new GUIStyle();
15
        0 references
16      void OnGUI()
17      {
18          guiStyle.fontSize = 25;
19          guiStyle.normal.textColor = Color.white;
20
21          GUI.BeginGroup(new Rect(10, 10, 250, 150));
22          GUI.Box(new Rect(0, 0, 140, 140), "Stats", guiStyle);
23          GUI.Label(new Rect(10, 25, 200, 30), "Gen: " + generation, guiStyle);
24          GUI.Label(new Rect(10, 50, 200, 30), string.Format("Time: {0:0.00}", elapsed), guiStyle);
25          GUI.Label(new Rect(10, 75, 200, 30), "Population: " + population.Count, guiStyle);
26          GUI.EndGroup();
27      }
28
```

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

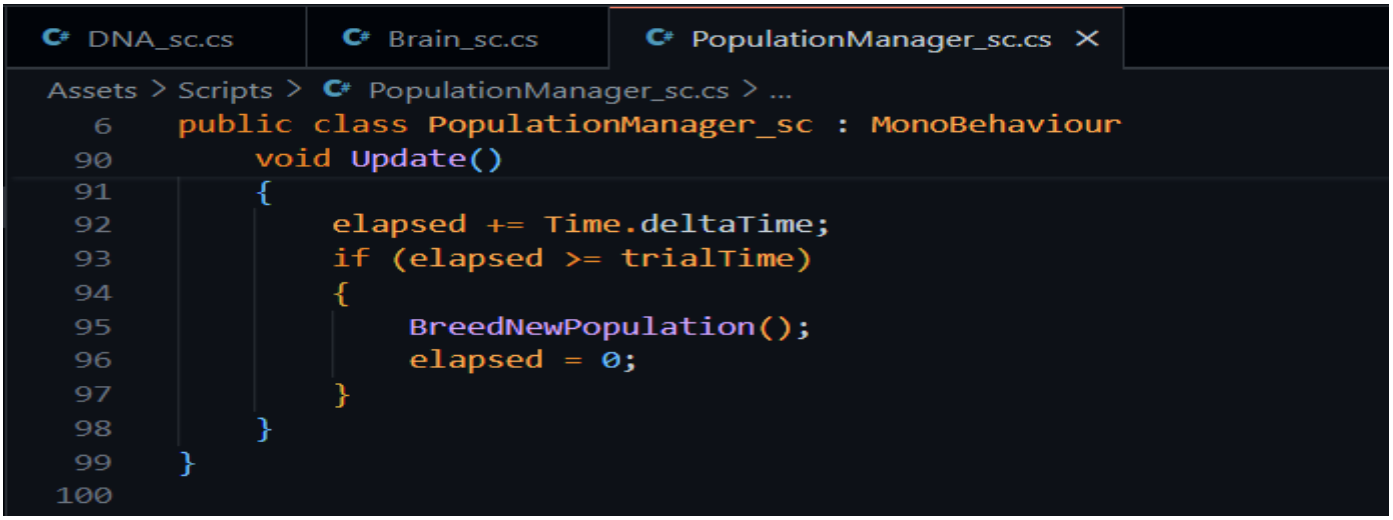
Assets > Scripts > PopulationManager_sc.cs > ...

```
6   public class PopulationManager_sc : MonoBehaviour
29      void Start()
30      {
31          // Instantiate population
32          for (int i = 0; i < populationSize; i++)
33          {
34              Vector3 startingPos = new Vector3(
35                  this.transform.position.x + Random.Range(-2, 2),
36                  this.transform.position.y,
37                  this.transform.position.z + Random.Range(-2, 2)
38              );
39
40              GameObject b = Instantiate(botPrefab, startingPos, this.transform.rotation);
41              b.GetComponent<Brain_sc>().Init();
42              population.Add(b);
43          }
44      }
45
        2 references
46      GameObject Breed(GameObject parent1, GameObject parent2)
47      {
48          Vector3 startingPos = new Vector3(
49              this.transform.position.x + Random.Range(-2, 2),
50              this.transform.position.y,
51              this.transform.position.z + Random.Range(-2, 2)
52          );
53
54          GameObject offspring = Instantiate(botPrefab, startingPos, this.transform.rotation);
55          Brain_sc b = offspring.GetComponent<Brain_sc>();
56
57          // Mutate or combine genes
58          b.Init();
59          if (Random.Range(0, 100) < 1) // mutate with a small probability
60          {
61              b.dna_sc.Mutate(0.01f); // Example mutation rate
62          }
63          else
```

DNA_sc.cs | Brain_sc.cs | PopulationManager_sc.cs

Assets > Scripts > PopulationManager_sc.cs > ...

```
6   public class PopulationManager_sc : MonoBehaviour
46      GameObject Breed(GameObject parent1, GameObject parent2)
63          else
64          {
65              b.dna_sc.Combine(parent1.GetComponent<Brain_sc>().dna_sc, parent2.GetComponent<Brain_sc>().dna_sc);
66          }
67
68          return offspring;
69      }
70
        1 reference
71      void BreedNewPopulation()
72      {
73          List<GameObject> sortedList = population.OrderByDescending(o => o.GetComponent<Brain_sc>().distanceTravelled).ToList();
74          population.Clear();
75
76          // Keep top performers
77          for (int i = (int)(sortedList.Count / 2.0f) - 1; i < sortedList.Count - 1; i++)
78          {
79              population.Add(Breed(sortedList[i], sortedList[i + 1]));
80              population.Add(Breed(sortedList[i + 1], sortedList[i]));
81          }
82
83          for (int i = 0; i < sortedList.Count; i++)
84          {
85              Destroy(sortedList[i]);
86          }
87          generation++;
88      }
```

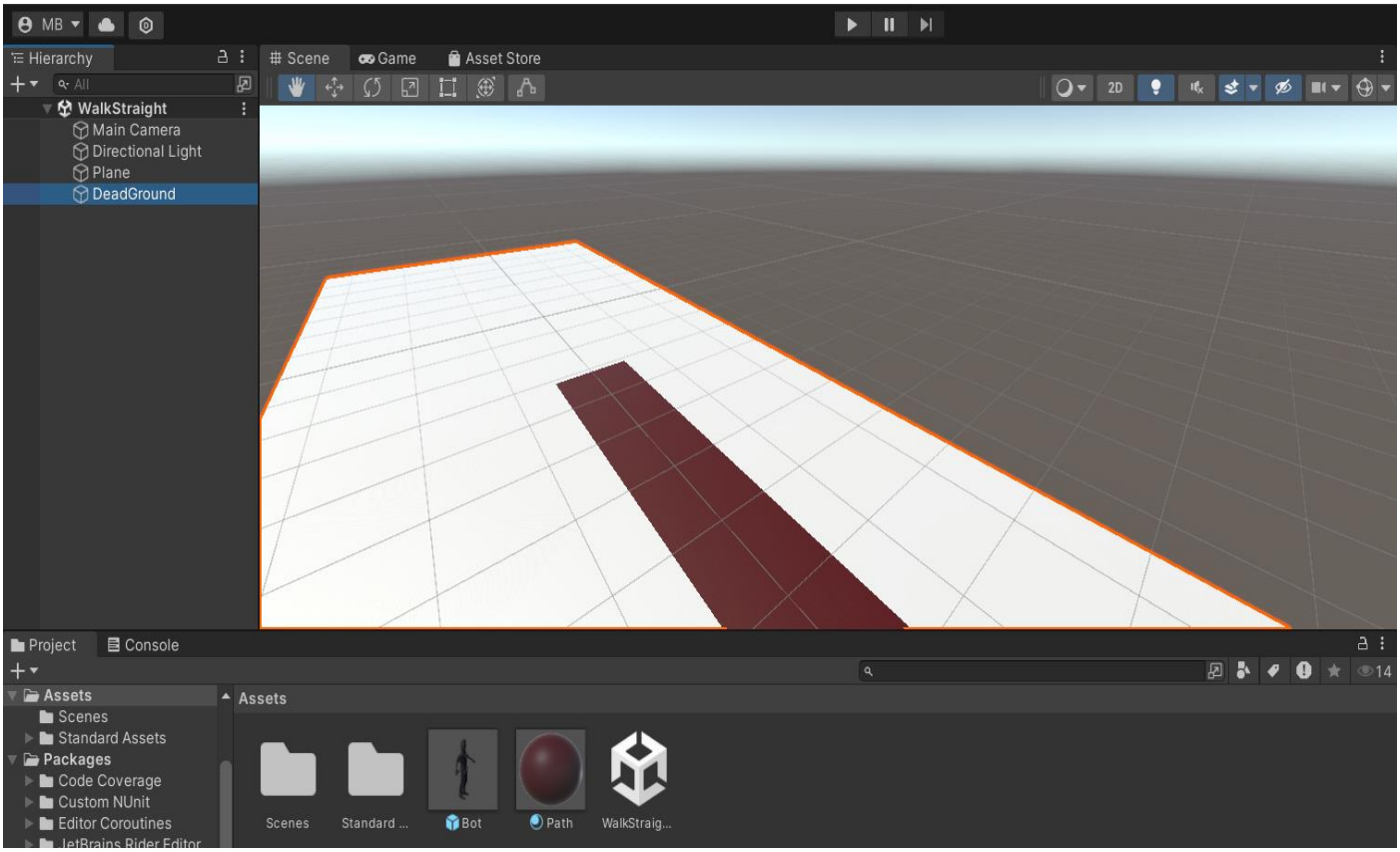


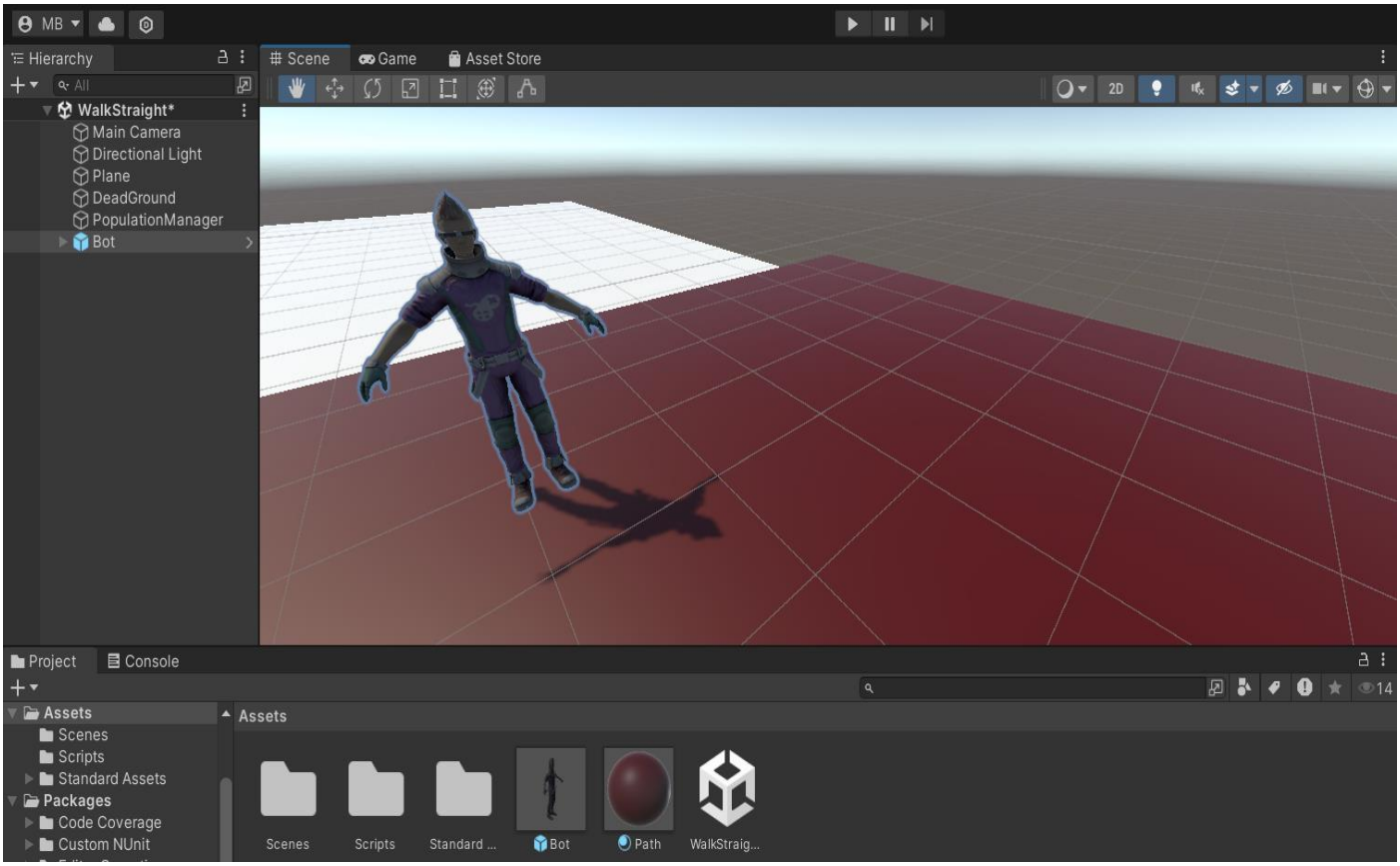
5. Simülasyonu Başlatmak ve İzlemek

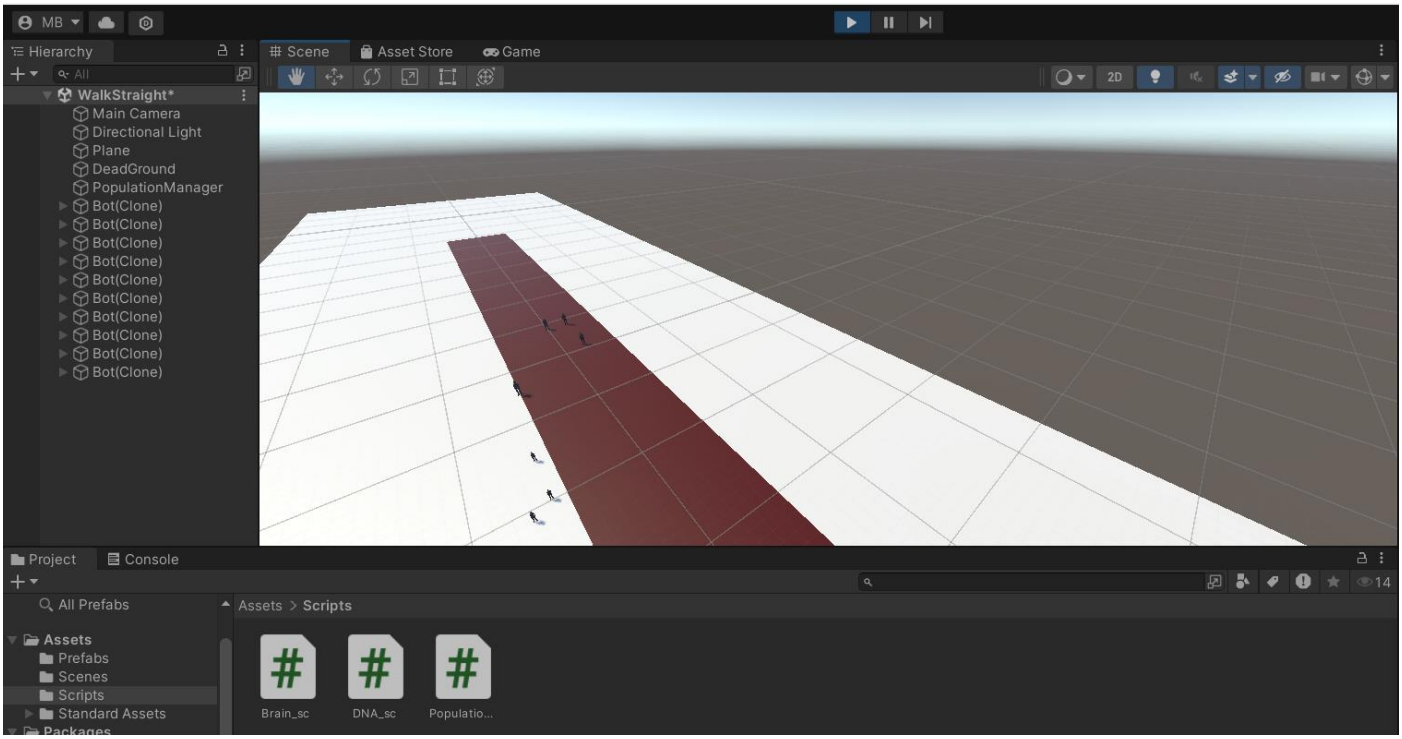
- **Simülasyonu Başlatma**: “Play” tuşuna basarak simülasyonu başlatın. Karakterler kırmızı platform üzerinde olacak ve zamanla en uzun süre kalanlar genetik kodlarını sonraki nesle aktaracaktır.
- **GUI ile Nesil ve Zaman Takibi**: PopulationManager’daki GUI bölümünde, nesil sayısı, geçen süre ve popülasyon boyutunu ekranda görebilirsiniz.

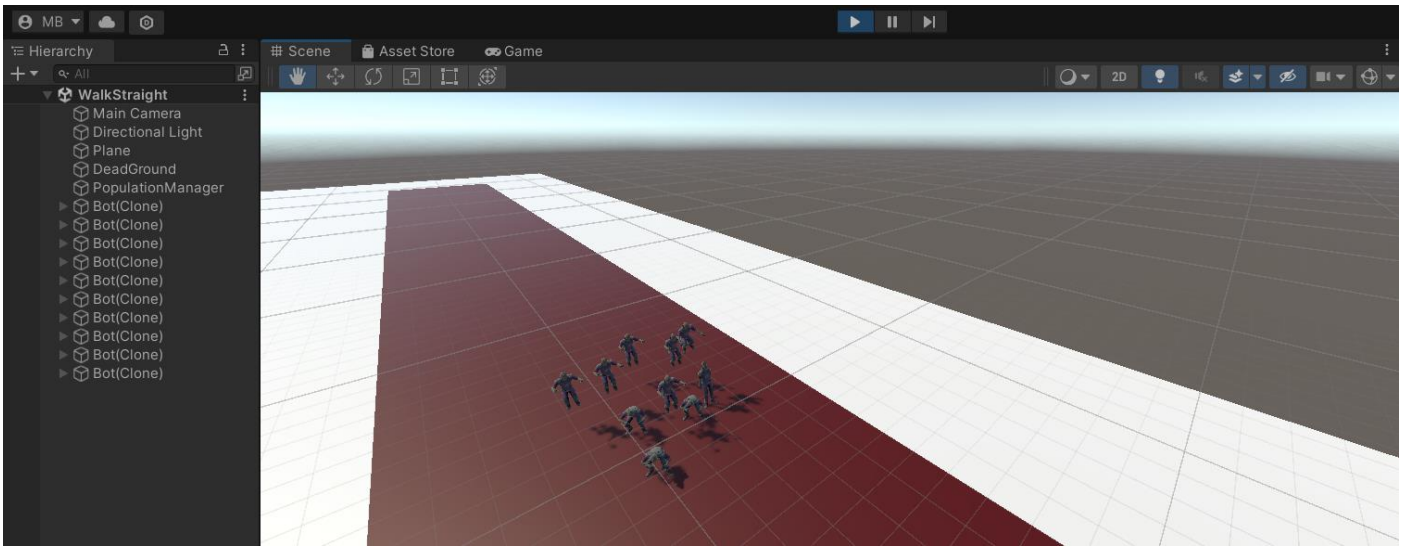
Sonuç

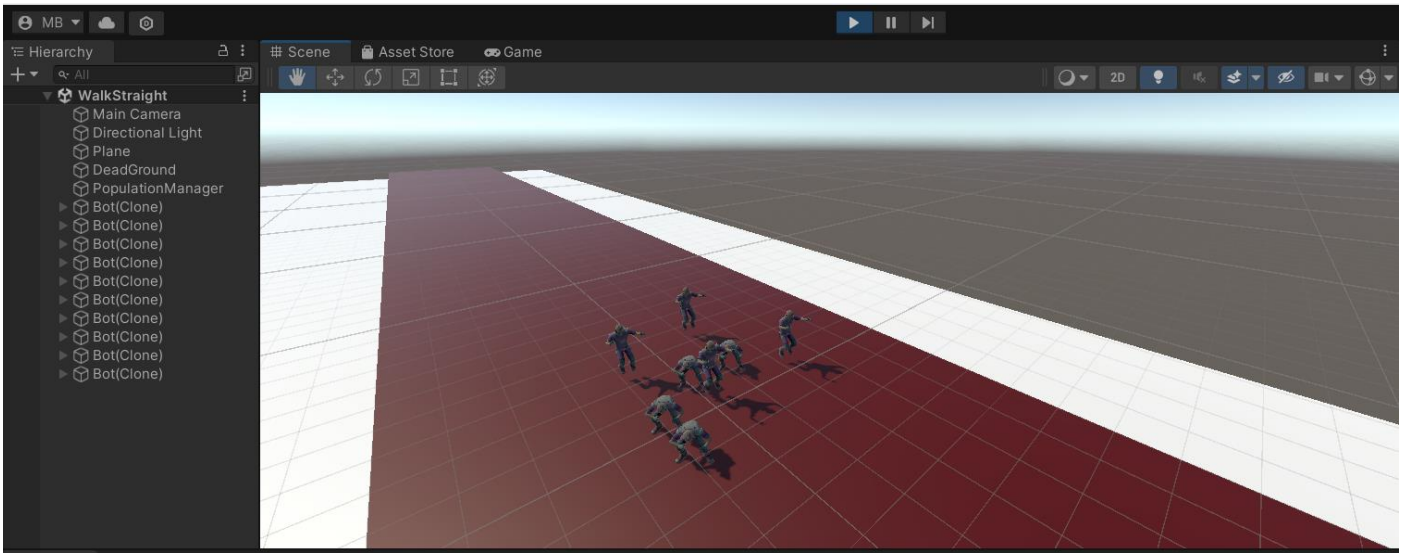
Bu proje, genetik algoritmalar kullanarak bir bot popülasyonunun nasıl oluşturulup geliştirileceğini göstermektedir. Scriptler, botların hareketlerini ve DNA yönetimini sağlar. Her yeni nesil, en iyi botların genetik özelliklerini taşır ve böylece gelişim sağlanır.


Hierarchy
Scene
Game
Asset Store
WalkStraight
Main Camera
Directional Light
Plane
DeadGround
Project
Console
Assets
Scenes
Standard Assets
Packages
Code Coverage
Custom NUnit
Editor Coroutines
Scenes
Standard ...
Bot
Path
WalkStraig...


Hierarchy
Scene
Game
Asset Store
WalkStraight*
Main Camera
Directional Light
Plane
DeadGround
PopulationManager
Bot
Project
Console
Assets
Scenes
Scripts
Standard Assets
Packages
Code Coverage
Custom NUnit
Scenes
Scripts
Standard ...
Bot
Path
WalkStraig...


Hierarchy
Scene
Asset Store
Game
WalkStraight*
Main Camera
Directional Light
Plane
DeadGround
PopulationManager
Bot(Clone)
Project
Console
All Prefabs
Assets > Scripts
Assets
Prefabs
Scenes
Scripts
Standard Assets
Brain_sc
DNA_sc
Populatio...


Hierarchy
Scene
Asset Store
Game
WalkStraight
Main Camera
Directional Light
Plane
DeadGround
PopulationManager
Bot(Clone)


Hierarchy
Scene
Asset Store
Game
WalkStraight
Main Camera
Directional Light
Plane
DeadGround
PopulationManager
Bot(Clone)