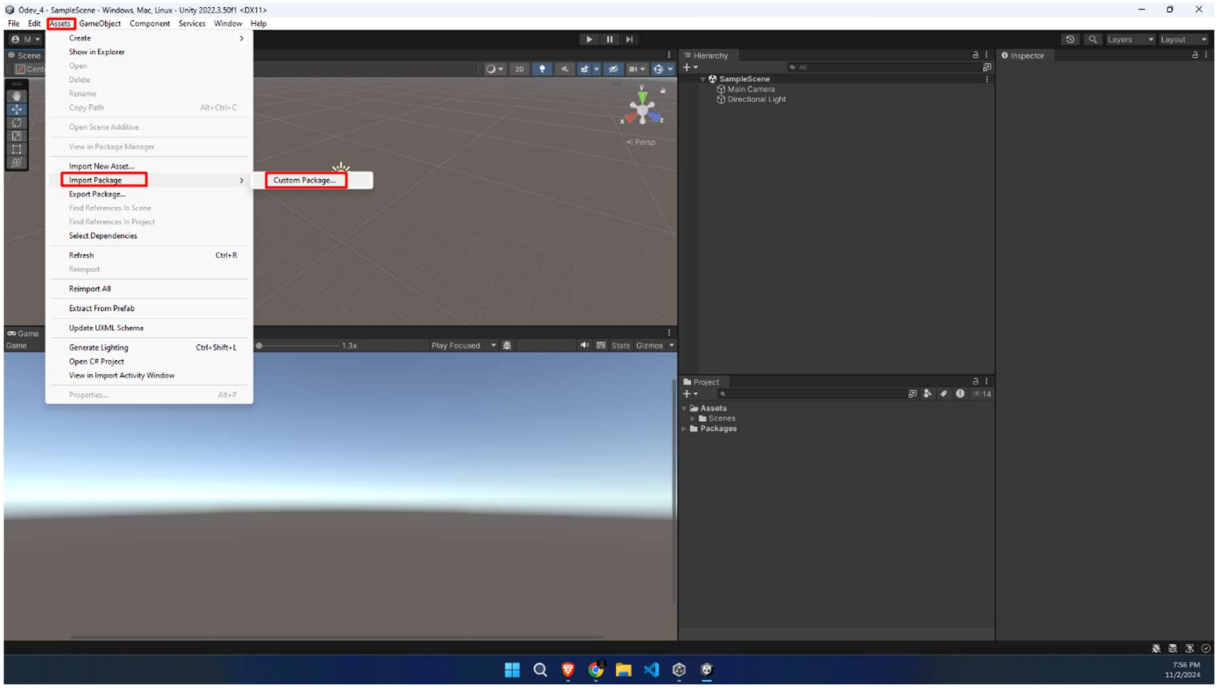
Bilgisayar Oyunlarında Yapay Zekâ - Ödev 4

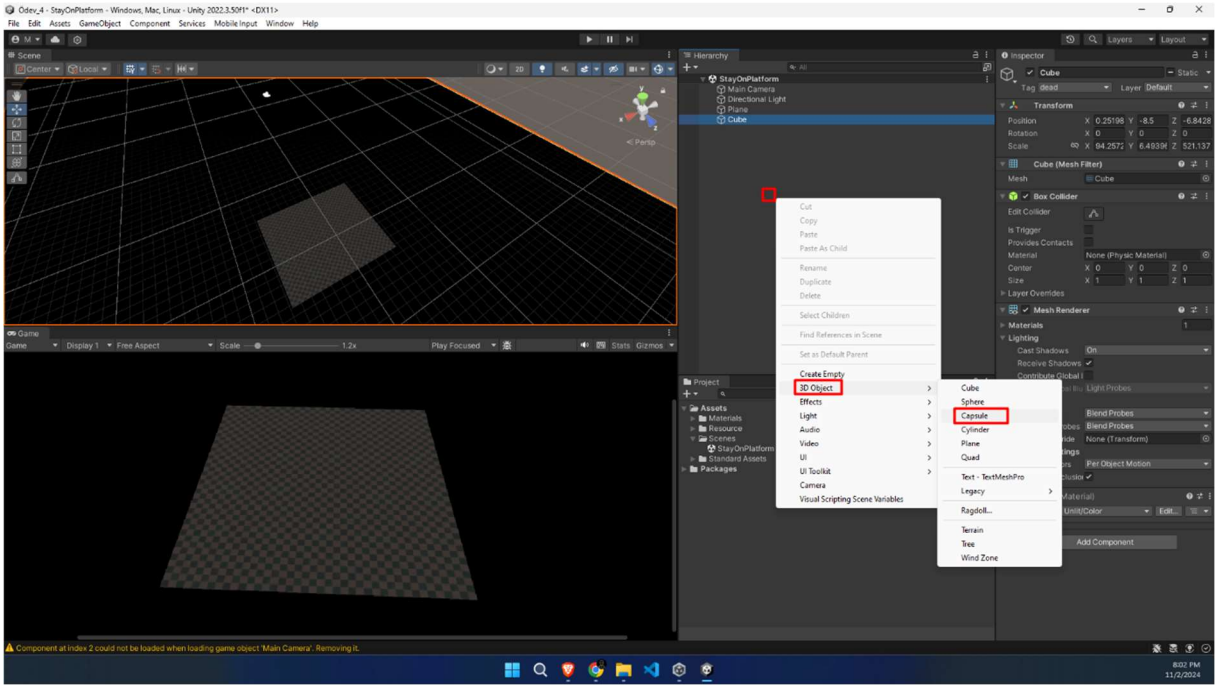
1 – Asset Import etme

1.1 - Üst menüdeki Assets->Import Package->Custom Package kısmından “StayOnPlatformStarter.unitypackage” adlı dosyayı seçiyoruz.

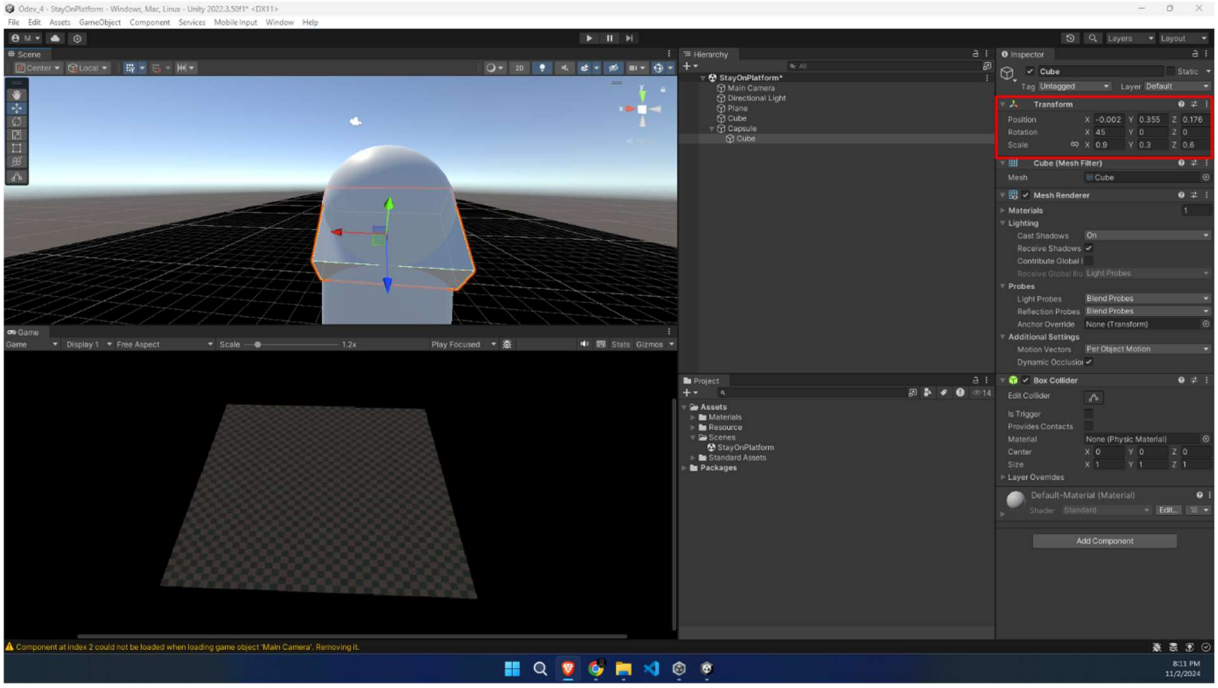


2 – Oyuncu oluşturma

2.1 – Hierarchy kısmında sağ tıklayarak capsule ekliyoruz. Ardından capsule üzerinde sağ tıklayarak cube ekliyoruz. Cube ismini “Eyes” olarak değiştiriyoruz.



2.2 – Eyes nesnesinin rotasyonunu gözler yere bakacak şekilde ayarlıyoruz.



3 – Scriptleri oluşturuyoruz.

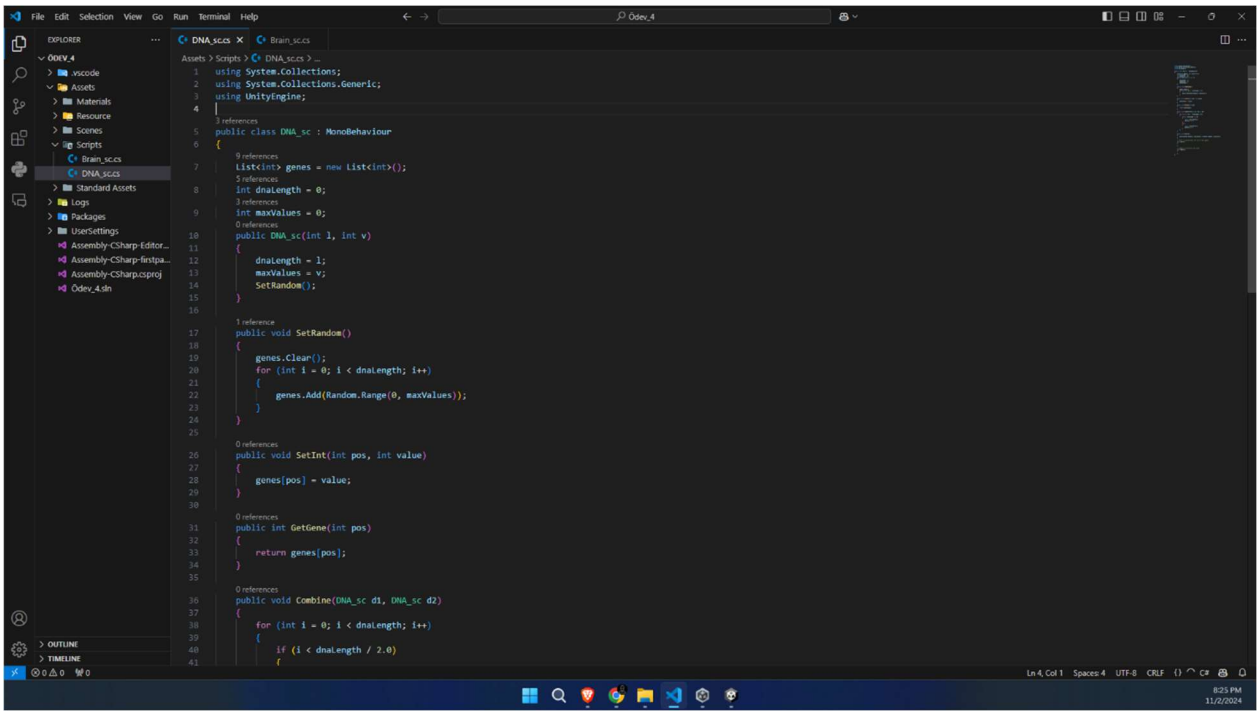
3.1 – DNA_sc scriptini aşağıdaki gibi oluşturuyoruz.

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class DNA_sc : MonoBehaviour

{

    // Genleri tutan liste

    List<int> genes = new List<int>();

    // DNA uzunluğu

    int dnaLength = 0;

    // Maksimum değerler

    int maxValues = 0;


    // Yapıcı metot, DNA uzunluğu ve maksimum değerleri alır

    public DNA_sc(int l, int v)
```

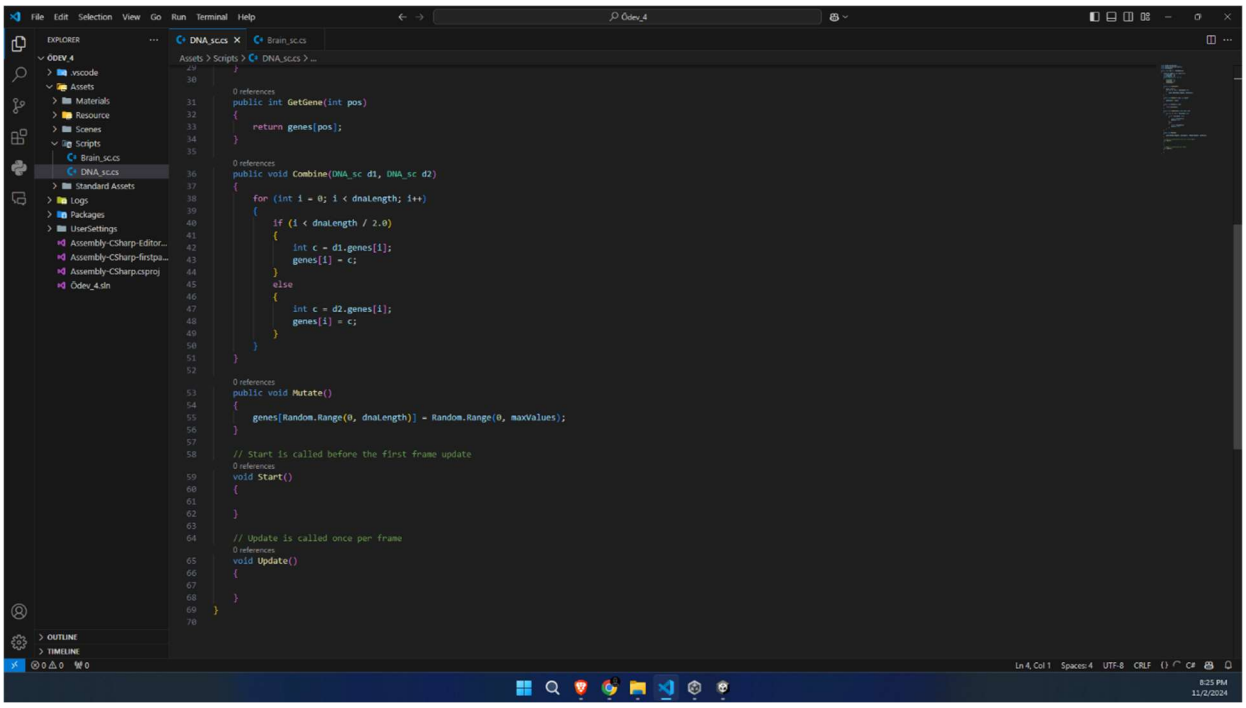
```
{
    dnaLength = l;
    maxValues = v;
    SetRandom();
}

// Genleri rastgele değerlerle doldurur
public void SetRandom()
{
    genes.Clear();
    for (int i = 0; i < dnaLength; i++)
    {
        genes.Add(Random.Range(0, maxValues));
    }
}

// Belirtilen pozisyondaki geni belirli bir değerle ayarlar
public void SetInt(int pos, int value)
{
    genes[pos] = value;
}

// Belirtilen pozisyondaki geni döndürür
public int GetGene(int pos)
{
    return genes[pos];
}

// İki DNA'yı birleştirir
public void Combine(DNA_sc d1, DNA_sc d2)
{
```

```
    for (int i = 0; i < dnaLength; i++)
    {
        if (i < dnaLength / 2.0)
        {
            int c = d1.genes[i];
            genes[i] = c;
        }
        else
        {
            int c = d2.genes[i];
            genes[i] = c;
        }
    }
}

// Rastgele bir geni mutasyona uğratır
public void Mutate()
{
    genes[Random.Range(0, dnaLength)] = Random.Range(0, maxValues);
}

// Start metodu, oyun başladığında bir kez çağrılır
void Start()
{

}

// Update metodu, her karede bir kez çağrılır
void Update()
{
```

```
    }

}
```



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

3 references
public class DNA_sc : MonoBehaviour
{
    9 references
    List<int> genes = new List<int>();
    5 references
    int dnaLength = 0;
    3 references
    int maxValues = 0;
    0 references
    public DNA_sc(int l, int v)
    {
        dnaLength = l;
        maxValues = v;
        SetRandom();
    }

    1 reference
    public void SetRandom()
    {
        genes.Clear();
        for (int i = 0; i < dnaLength; i++)
        {
            genes.Add(Random.Range(0, maxValues));
        }
    }

    0 references
    public void SetInt(int pos, int value)
    {
        genes[pos] = value;
    }

    0 references
    public int GetGene(int pos)
    {
        return genes[pos];
    }

    0 references
    public void Combine(DNA_sc d1, DNA_sc d2)
    {
        for (int i = 0; i < dnaLength; i++)
        {
            if (i < dnaLength / 2.0)
            {
```



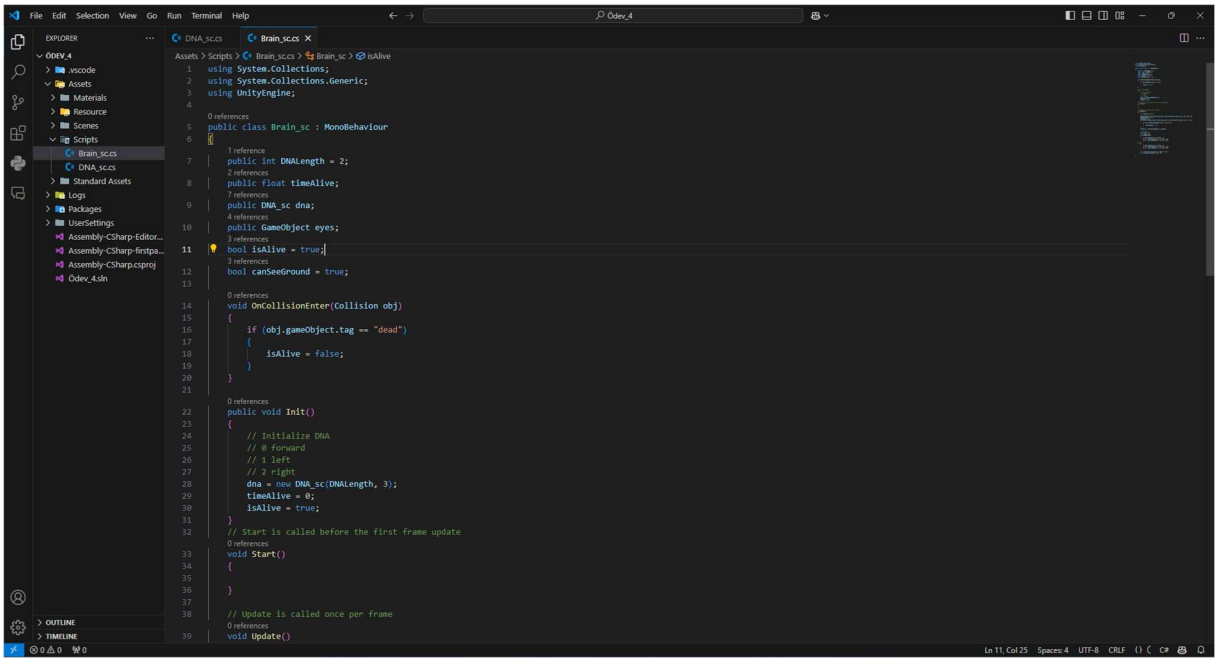
```
    }

    0 references
    public int GetGene(int pos)
    {
        return genes[pos];
    }

    0 references
    public void Combine(DNA_sc d1, DNA_sc d2)
    {
        for (int i = 0; i < dnaLength; i++)
        {
            if (i < dnaLength / 2.0)
            {
                int c = d1.genes[i];
                genes[i] = c;
            }
            else
            {
                int c = d2.genes[i];
                genes[i] = c;
            }
        }
    }

    0 references
    public void Mutate()
    {
        genes[Random.Range(0, dnaLength)] = Random.Range(0, maxValues);
    }

    // Start is called before the first frame update
    0 references
    void Start()
    {

    }

    // Update is called once per frame
    0 references
    void Update()
    {

    }
}
```

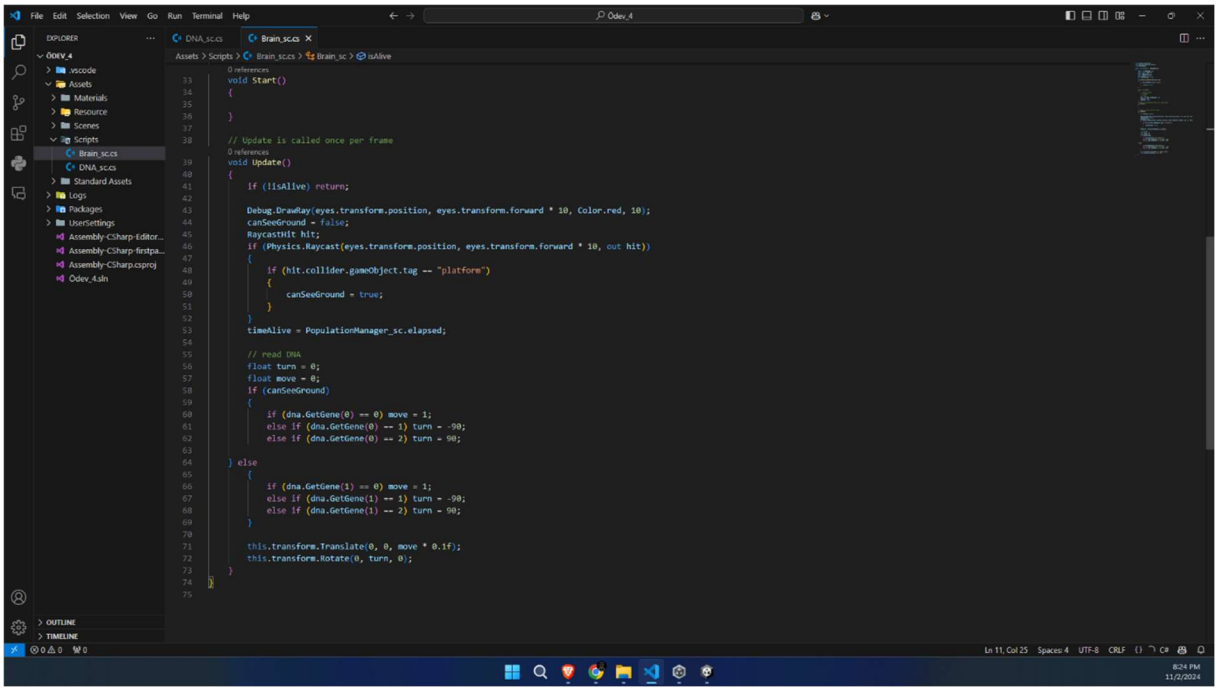
3.2 - Brain_sc scripti

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Brain_sc : MonoBehaviour
{
    public int DNALength = 2;
    public float timeAlive;
    public DNA_sc dna_sc;
    public GameObject eyes;
    bool isAlive = true;
    bool canSeeGround = true;

    void OnCollisionEnter(Collision obj)
    {
        if (obj.gameObject.tag == "dead")
        {
            isAlive = false;
        }
    }

    public void Init()
    {
        // Initialize DNA
        // 0 forward
        // 1 left
        // 2 right
```

```
    dna_sc = new DNA_sc(DNALength, 3);
    timeAlive = 0;
    isAlive = true;
}
// Start is called before the first frame update
void Start()
{

}

// Update is called once per frame
void Update()
{
    if (!isAlive) return;

    Debug.DrawRay(eyes.transform.position, eyes.transform.forward * 10, Color.red, 10);
    canSeeGround = false;
    RaycastHit hit;
    if (Physics.Raycast(eyes.transform.position, eyes.transform.forward * 10, out hit))
    {
        if (hit.collider.gameObject.tag == "platform")
        {
            canSeeGround = true;
        }
    }
    timeAlive = PopulationManager_sc.elapsed;

    // read DNA
    float turn = 0;
    float move = 0;
    if (canSeeGround)
```

```
        {
            if (dna_sc.GetGene(0) == 0) move = 1;
            else if (dna_sc.GetGene(0) == 1) turn = -90;
            else if (dna_sc.GetGene(0) == 2) turn = 90;

        }
        else
        {
            if (dna_sc.GetGene(1) == 0) move = 1;
            else if (dna_sc.GetGene(1) == 1) turn = -90;
            else if (dna_sc.GetGene(1) == 2) turn = 90;
        }

        this.transform.Translate(0, 0, move * 0.1f);
        this.transform.Rotate(0, turn, 0);
    }
}
```





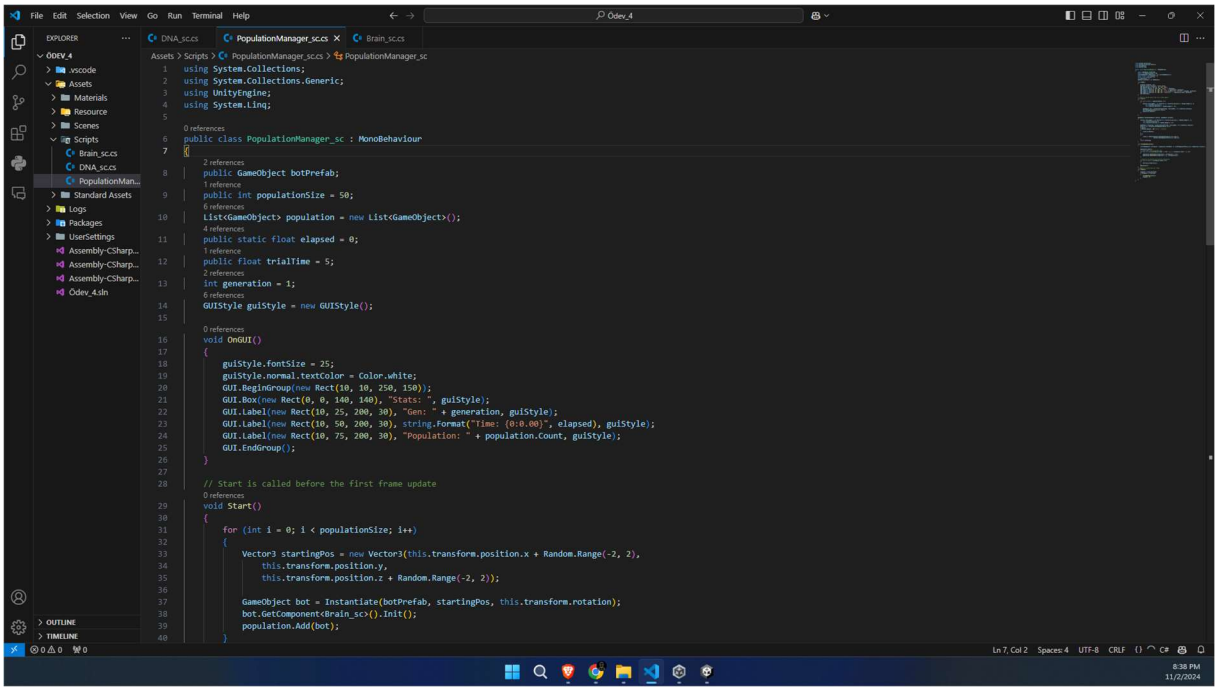
3.3 – PopulationManager_sc scripti

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;

public class PopulationManager_sc : MonoBehaviour
{
    public GameObject botPrefab;
    public int populationSize = 50;
    List<GameObject> population = new List<GameObject>();
    public static float elapsed = 0;
    public float trialTime = 5;
    int generation = 1;
    GUIStyle guiStyle = new GUIStyle();

    void OnGUI()
```

```
{
    guiStyle.fontSize = 25;
    guiStyle.normal.textColor = Color.white;
    GUI.BeginGroup(new Rect(10, 10, 250, 150));
    GUI.Box(new Rect(0, 0, 140, 140), "Stats: ", guiStyle);
    GUI.Label(new Rect(10, 25, 200, 30), "Gen: " + generation, guiStyle);
    GUI.Label(new Rect(10, 50, 200, 30), string.Format("Time: {0:0.00}", elapsed), guiStyle);
    GUI.Label(new Rect(10, 75, 200, 30), "Population: " + population.Count, guiStyle);
    GUI.EndGroup();
}

// Start is called before the first frame update
void Start()
{
    for (int i = 0; i < populationSize; i++)
    {
        Vector3 startingPos = new Vector3(this.transform.position.x + Random.Range(-2, 2),
            this.transform.position.y,
            this.transform.position.z + Random.Range(-2, 2));

        GameObject bot = Instantiate(botPrefab, startingPos, this.transform.rotation);
        bot.GetComponent<Brain_sc>().Init();
        population.Add(bot);
    }

}

GameObject Breed(GameObject parent1, GameObject parent2)
{
    Vector3 startingPos = new Vector3(this.transform.position.x + Random.Range(-2, 2),
        this.transform.position.y,
```

```
        this.transform.position.z + Random.Range(-2, 2));

    GameObject offspring = Instantiate(botPrefab, startingPos, this.transform.rotation);
    Brain_sc b = offspring.GetComponent<Brain_sc>();
    b.Init();
    if(Random.Range(0, 100) == 1) // mutation
    {
        b.dna_sc.Mutate();
    }
    else
    {
        b.dna_sc.Combine(parent1.GetComponent<Brain_sc>().dna_sc,
                    parent2.GetComponent<Brain_sc>().dna_sc);
    }
    return offspring;
}

void BreedNewPopulation()
{
    List<GameObject> sortedList = population.OrderBy(o =>
(o.GetComponent<Brain_sc>().timeAlive)).ToList();

    population.Clear();
    // breed upper half of sorted list
    for (int i = (int)(sortedList.Count / 2.0f) - 1; i < sortedList.Count - 1; i++)
    {
        population.Add(Breed(sortedList[i], sortedList[i + 1]));
        population.Add(Breed(sortedList[i + 1], sortedList[i]));
    }

    // destroy all parents and previous population
```

```
        for (int i = 0; i < sortedList.Count; i++)
        {
            Destroy(sortedList[i]);
        }
        generation++;
    }
    // Update is called once per frame
    void Update()
    {
        elapsed += Time.deltaTime;
        if(elapsed >= trialTime)
        {
            BreedNewPopulation();
            elapsed = 0;
        }
    }
}
```



File Edit Selection View Go Run Terminal Help

Ödev_4

EXPLORER

ÖDEV_4
- .vscode
- Assets
  - Materials
  - Resource
  - Scenes
  - Scripts
    - Brain_sc.cs
    - DNA_sc.cs
    - PopulationMan...
  - Standard Assets
  - Logs
  - Packages
  - UserSettings
  - Assembly-CSharp...
  - Assembly-CSharp...
  - Assembly-CSharp...
  - Ödev_4.sln

OUTLINE

TIMELINE

DNA_sc.cs | PopulationManager_sc.cs | Brain_sc.cs

Assets > Scripts > PopulationManager_sc.cs > PopulationManager_sc

```
    // Start is called before the first frame update
    0 references
    void Start()
    {
        for (int i = 0; i < populationSize; i++)
        {
            Vector3 startingPos = new Vector3(this.transform.position.x + Random.Range(-2, 2),
                this.transform.position.y,
                this.transform.position.z + Random.Range(-2, 2));

            GameObject bot = Instantiate(botPrefab, startingPos, this.transform.rotation);
            bot.GetComponent<Brain_sc>().Init();
            population.Add(bot);
        }

    }

    2 references
    GameObject Breed(GameObject parent1, GameObject parent2)
    {
        Vector3 startingPos = new Vector3(this.transform.position.x + Random.Range(-2, 2),
            this.transform.position.y,
            this.transform.position.z + Random.Range(-2, 2));

        GameObject offspring = Instantiate(botPrefab, startingPos, this.transform.rotation);
        Brain_sc b = offspring.GetComponent<Brain_sc>();
        b.Init();
        if(Random.Range(0, 100) == 1) // mutation
        {
            b.dna_sc.Mutate();
        }
        else
        {
            b.dna_sc.Combine(parent1.GetComponent<Brain_sc>().dna_sc,
                             parent2.GetComponent<Brain_sc>().dna_sc);
        }
        return offspring;
    }

    1 reference
    void BreedNewPopulation()
    {
        List<GameObject> sortedList = population.OrderBy(o => (o.GetComponent<Brain_sc>().timeAlive)).ToList();

        population.Clear();
        // breed upper half of sorted list
        for (int i = (int)(sortedList.Count / 2.0f) - 1; i < sortedList.Count - 1; i++)
        {
```

0 0 0 0

Ln 7, Col 2 Spaces: 4 UTF-8 CRLF {} C#

8:39 PM
11/2/2024

File Edit Selection View Go Run Terminal Help

Ödev_4

EXPLORER

ÖDEV_4
- .vscode
- Assets
  - Materials
  - Resource
  - Scenes
  - Scripts
    - Brain_sc.cs
    - DNA_sc.cs
    - PopulationMan...
  - Standard Assets
  - Logs
  - Packages
  - UserSettings
  - Assembly-CSharp...
  - Assembly-CSharp...
  - Assembly-CSharp...
  - Ödev_4.sln

OUTLINE

TIMELINE

DNA_sc.cs | PopulationManager_sc.cs | Brain_sc.cs

Assets > Scripts > PopulationManager_sc.cs > PopulationManager_sc

```
    }

    1 reference
    void BreedNewPopulation()
    {
        List<GameObject> sortedList = population.OrderBy(o => (o.GetComponent<Brain_sc>().timeAlive)).ToList();

        population.Clear();
        // breed upper half of sorted list
        for (int i = (int)(sortedList.Count / 2.0f) - 1; i < sortedList.Count - 1; i++)
        {
            population.Add(Breed(sortedList[i], sortedList[i + 1]));
            population.Add(Breed(sortedList[i + 1], sortedList[i]));
        }

        // destroy all parents and previous population
        for (int i = 0; i < sortedList.Count; i++)
        {
            Destroy(sortedList[i]);
        }
        generation++;
    }
    // Update is called once per frame
    0 references
    void Update()
    {
        elapsed += Time.deltaTime;
        if(elapsed >= trialTime)
        {
            BreedNewPopulation();
            elapsed = 0;
        }
    }
}
```

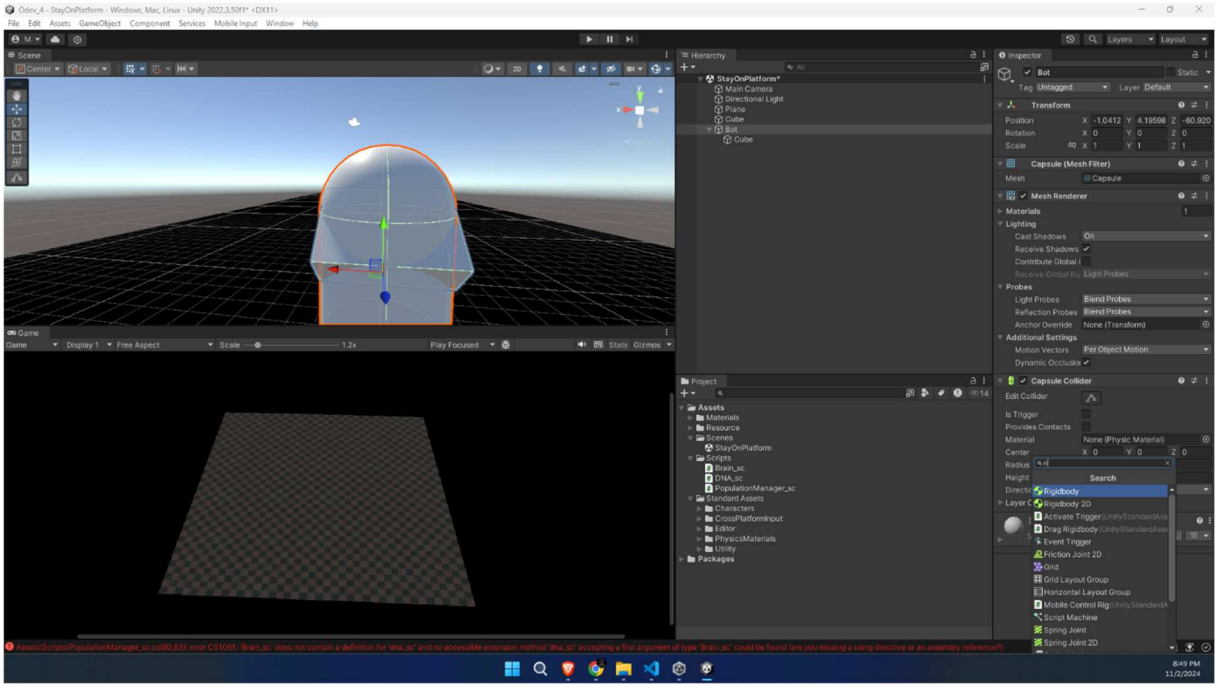
0 0 0 0

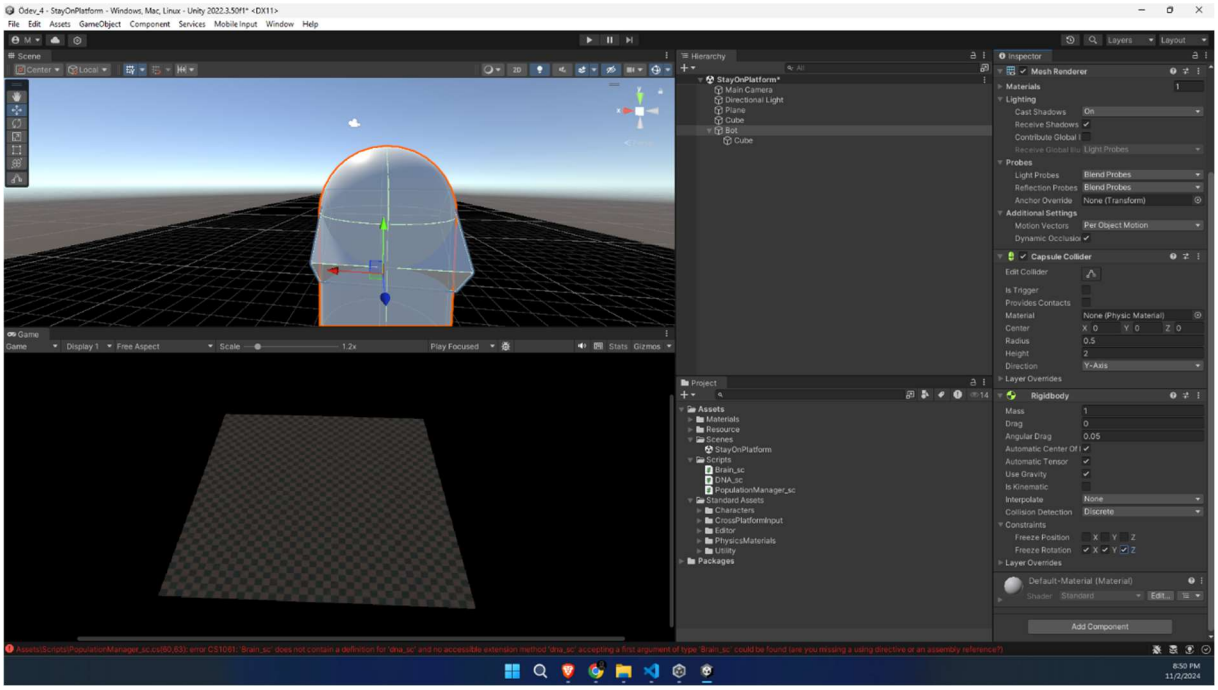
Ln 7, Col 2 Spaces: 4 UTF-8 CRLF {} C#

8:39 PM
11/2/2024

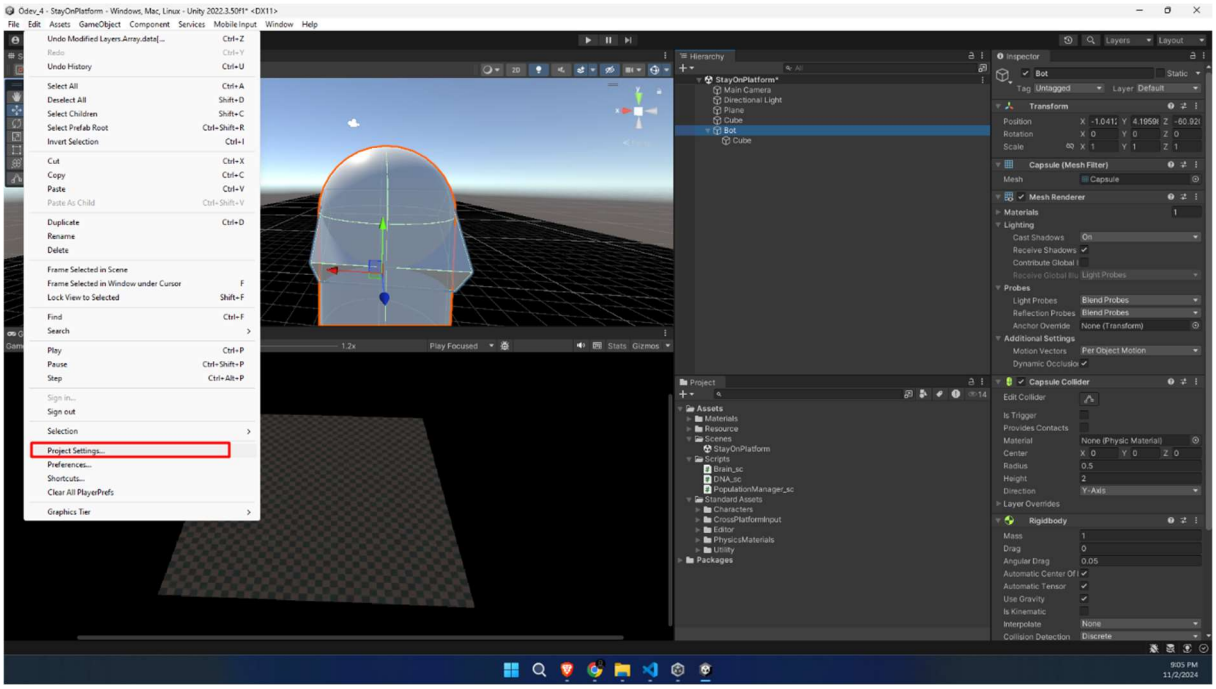
4 – Son adımlar

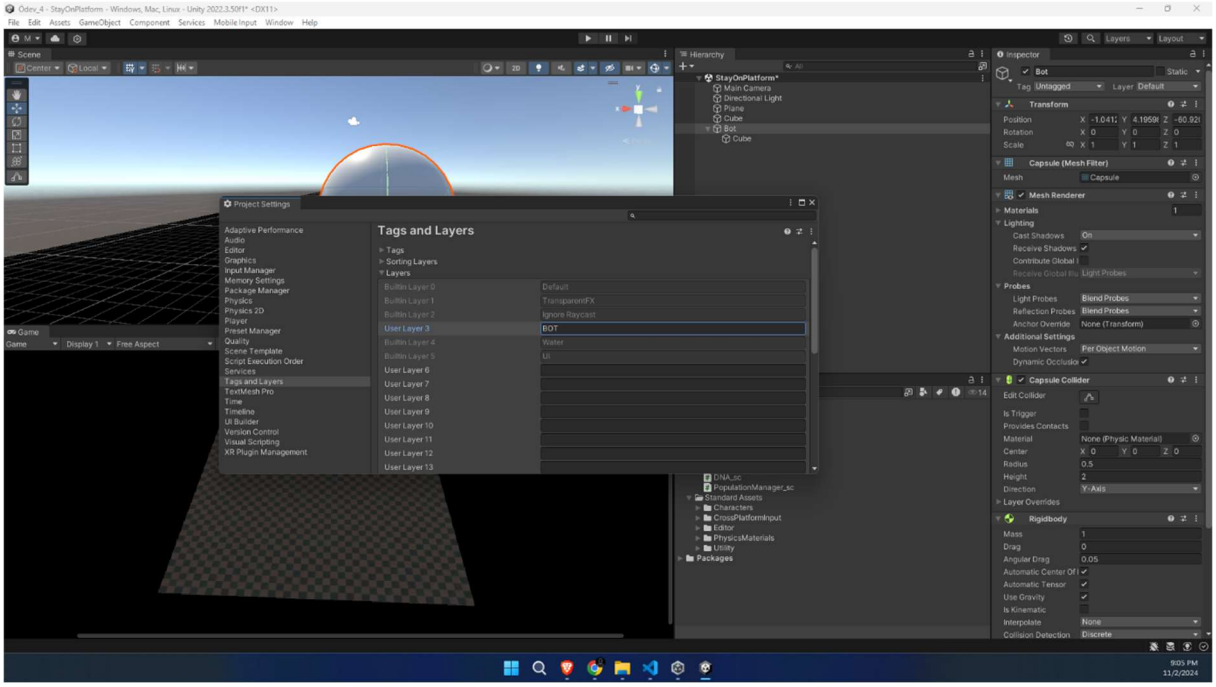
4.1 – Capsule nesnesinin adını Bot yapıyoruz. Bot nesnesine RigidBody ekleyip Constraint ekliyoruz: Rotation: x, y, z.

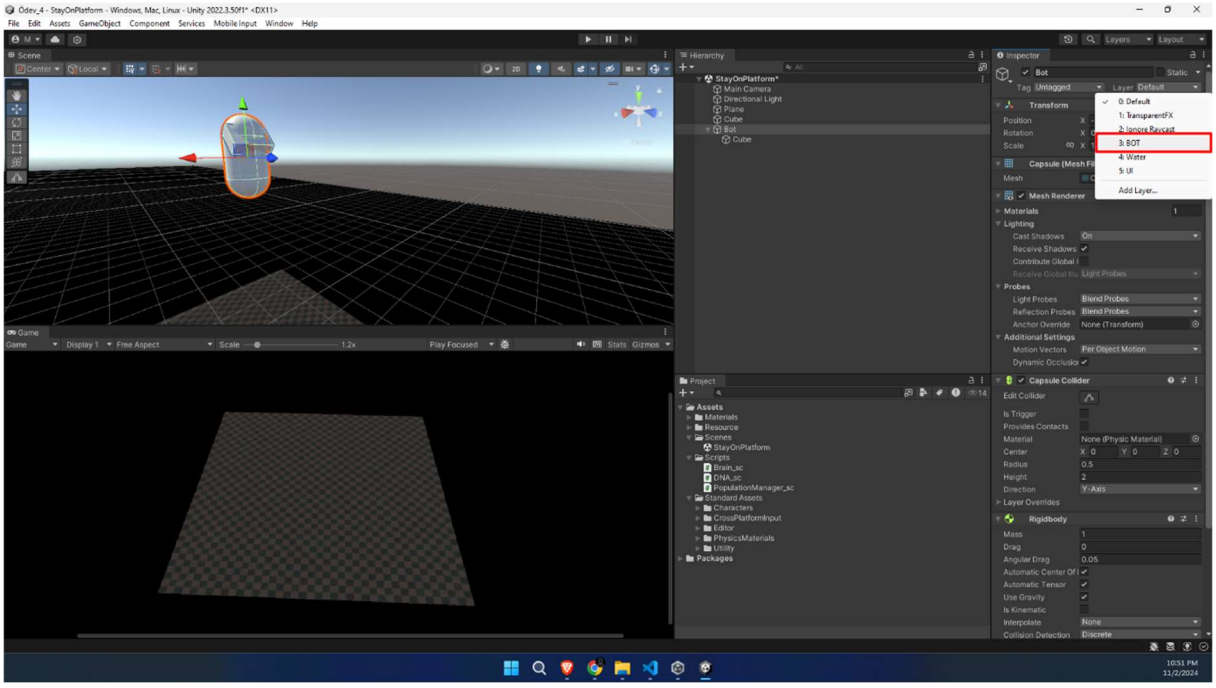


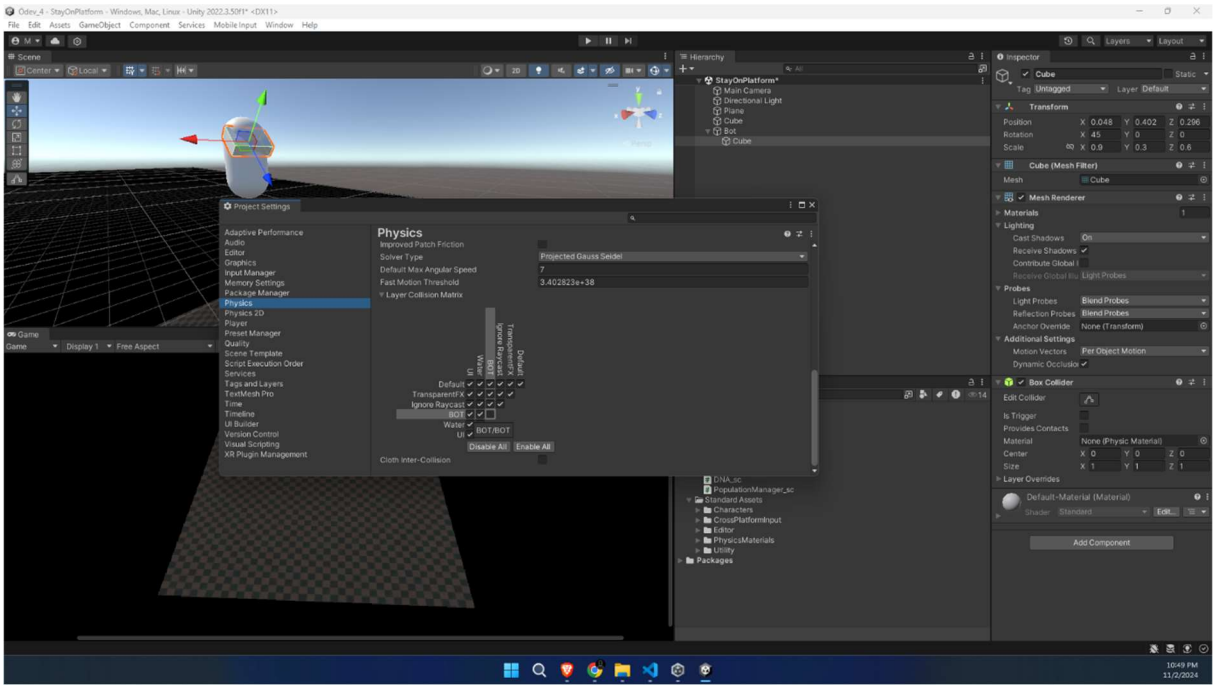


4.2 – Yeni bir katman ekliyoruz ve BOT-BOT çarpışmasını kapatıyoruz. Bot nesnesi için layeri BOT olarak seçiyoruz.

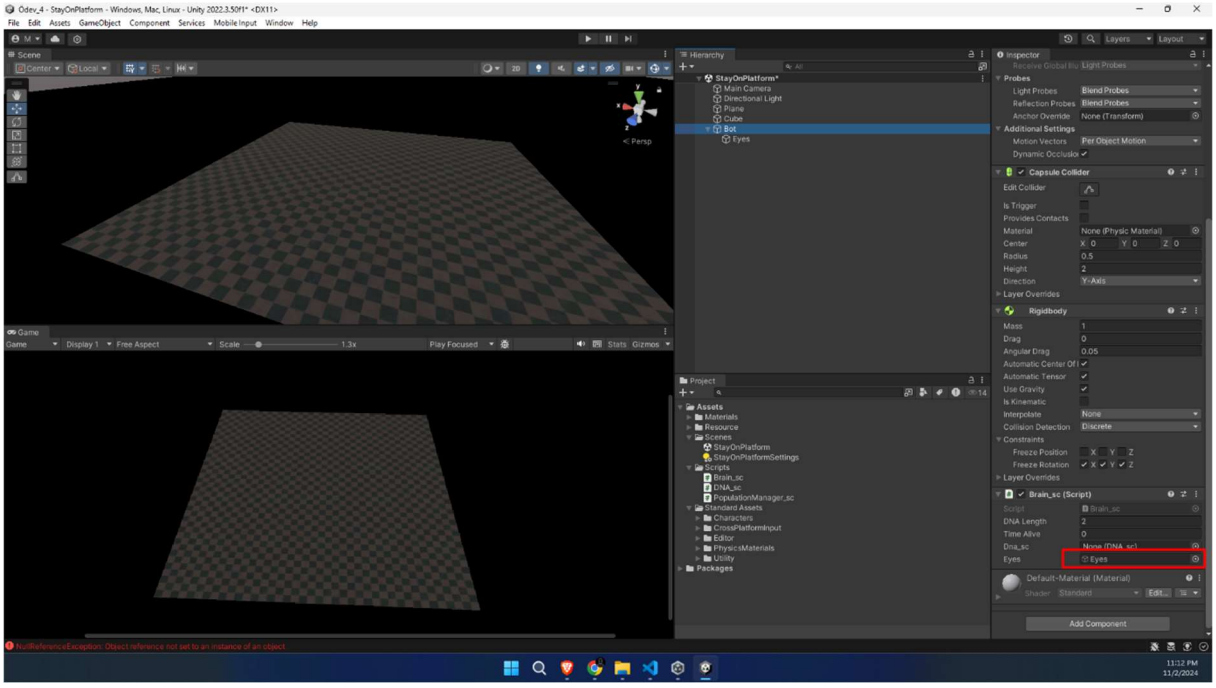





Odev_4 - StayOnPlatform - Windows, Mac, Linux - Unity 2022.3.50f1* <DX11>
File Edit Assets GameObject Component Services Mobile Input Window Help
Scene
Hierarchy
Inspector
Game
Project
StayOnPlatform*
Main Camera
Directional Light
Plane
Cube
Bot
Cube
Assets
Materials
Resource
Scenes
StayOnPlatform
Scripts
Brain_sc
DNA_sc
PopulationManager_sc
Standard Assets
Characters
CrossPlatformInput
Editor
PhysicsMaterials
Utility
Packages
Bot
Tag Untagged
Layer Default
0: Default
1: TransparentFX
2: Ignore Raycast
3: BOT
4: Water
5: UI
Add Layer...
Transform
Capsule (Mesh Filter)
Mesh Renderer
Capsule Collider
Rigidbody

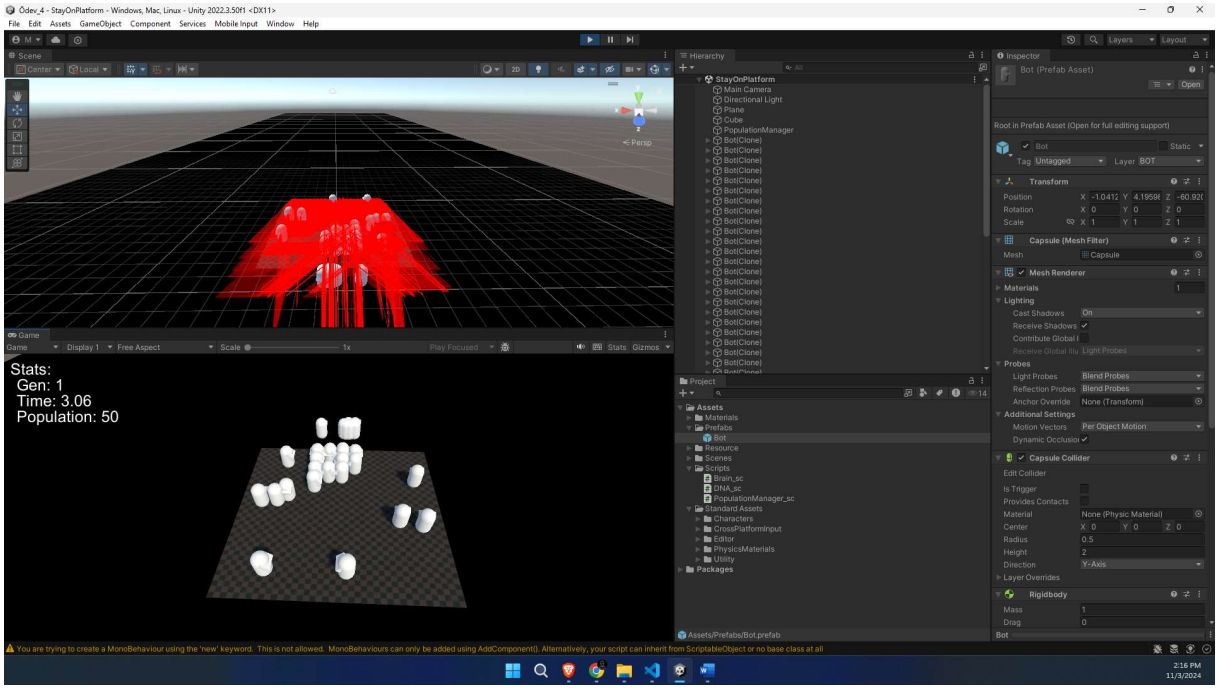

Odev_4 - StayOnPlatform - Windows, Mac, Linux - Unity 2022.3.50f1* <DX11>
File Edit Assets GameObject Component Services Mobile Input Window Help
Scene
Hierarchy
Inspector
Game
Project Settings
Adaptive Performance
Audio
Editor
Graphics
Input Manager
Memory Settings
Package Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Tags and Layers
TextMesh Pro
Time
Timeline
UI Builder
Version Control
Visual Scripting
XR Plugin Management
Physics
Improved Patch Friction
Solver Type
Projected Gauss Seidel
Default Max Angular Speed
7
Fast Motion Threshold
3.402823e+38
Layer Collision Matrix
Default
TransparentFX
Ignore Raycast
BOT
Water
UI
BOT/BOT
Disable All
Enable All
Cloth Inter-Collision
StayOnPlatform*
Main Camera
Directional Light
Plane
Cube
Bot
Cube
Cube
Transform
Position X 0.048 Y 0.402 Z 0.296
Rotation X 45 Y 0 Z 0
Scale X 0.9 Y 0.3 Z 0.6
Cube (Mesh Filter)
Mesh Renderer
Box Collider
Size X 1 Y 1 Z 1
Default-Material (Material)
Shader Standard
Add Component

4.3 – Brain scriptini Bot nesnesine ekliyoruz. Bot nesnesine tıklayarak eklenen brain scriptine eyes nesnesini ekliyoruz.



4.4 – Boş bir nesne ekleyerek adını PopulationManager yapıp PopulationManager_sc scriptini ekliyoruz.

4.5 – Project altında Prefabs klasörü oluşturarak Bot nesnesini prefabs altına ekliyoruz. Hierarchy altındaki Bot nesnesini silerek oyunu başlatıyoruz.



GitHub linki:
https://github.com/mehmetgencdal/BilgisayarOyunlarindaYapayZeka/tree/main/%C3%96dev_4