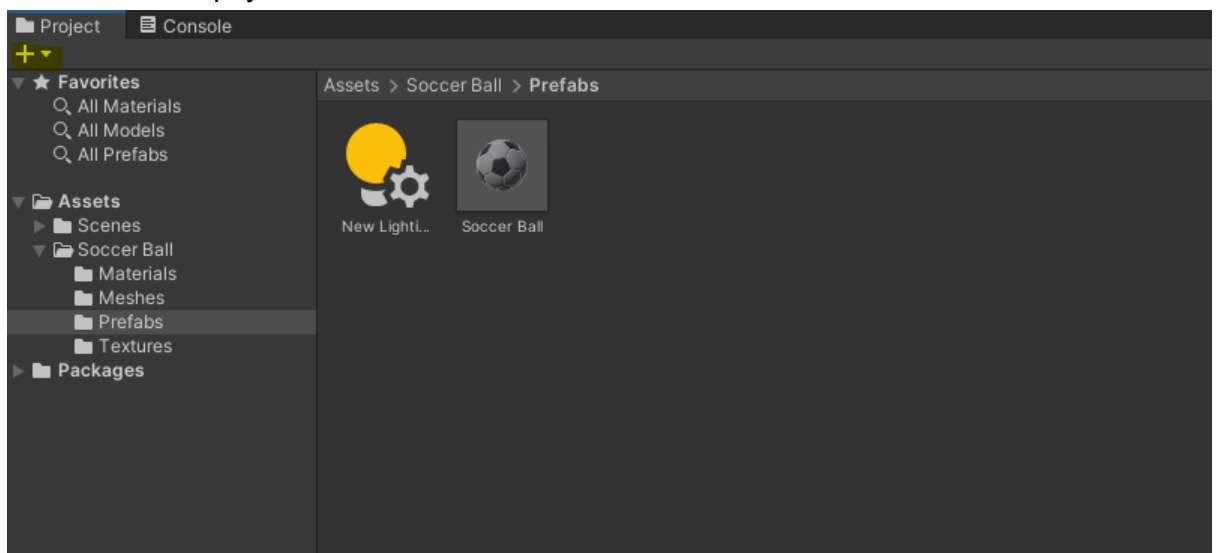


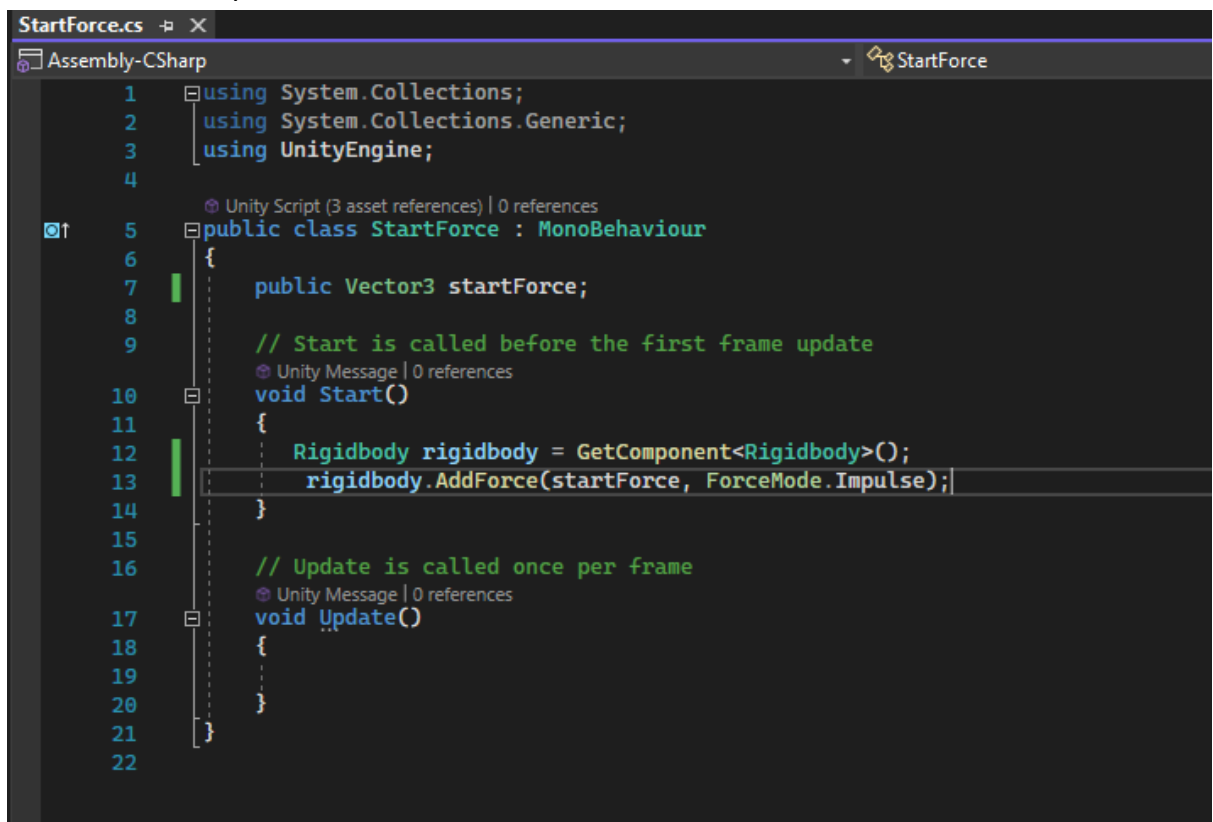
Infinite Bouncing Ball

1. Start off by creating a cube in your 3D scene, change its scale by 5 on the X and Z axis;
2. Now let's create a new material, the cube right now looks grey and it's kinda boring. Right mouse click in the Assets folder and select "Material" and name it "Green";
3. Select the material by clicking on it and on the inspector on right go over "Albedo" and change the colour from white to green;
4. Now drag the "Green" material from the Assets folder onto the cube in the scene;
5. Now let's get a ball, we can download a soccer one from the assets store for free. Go to Window > Asset Store and in the search bar type "Low polygon soccer ball". If the Asset Store doesn't work in Unity you can open it in your browser, just make sure you are logged in with your Unity account. Just click on "Add to my assets" and "Import".
6. Open the "Soccer Ball" > "Prefabs" and drag the Soccer Ball prefab from onto the scene on top of the cube. Make sure the position on the x axis is 1, y axis is 2 and z axis is 0. Make sure to also add a rigidbody by scrolling down in the inspector > clicking add component. If you don't add a rigid body our ball will fall through the cube when we start the scene;
7. Now let's just our Main Camera to have a better view. Select the Main Camera in the Hierarchy on the left hand side of the screen. Change the z axis to -3 and on Clear Flags change from Skybox to Solid Color;
8. If we now start out game we can see that the ball will fall on the cube, but now bounce;
9. To fix this we shall create a Physics Material by clicking on the plus sign and selecting Physics Material. A Physics Material allows us to change the values of friction and bouncing to the colliding object (the soccer ball) it will be added to. We will rename this physics material to Ball.



10. Let's select this material and on the inspector change its Bounciness value from 0 to. If we now start our scene we can see that the ball will bounce about 2 times before settling in on the floor.

11. However we want our ball to bounce and never stop. If we look at the inspector we can see that the Bounce Combine is set to Average, this means the Bounciness of the cube (0) and the Bounciness of the ball (1) will be averaged to 0.5. If we change the Bounce Combine to Maximum it will use the highest amount for the bounciness which in this case is 1 (the ball). Let's also set the Dynamic Friction to 0 if we want our ball to bounce forever and set the Friction Combine value to Minimum;
12. If we play our scene we can see that the ball will keep bouncing forever but it gets higher and higher as it goes. To avoid this let's change the value of the Bounciness from 1 to 0.965, this value is the one that will keep the ball at a constant height.
13. Let's make things more interesting by duplicating our cube and changing its position x axis to 3, y to 2, and the scale x axis to 1 and y axis to 5;
14. Duplicate the cube you've duplicated before and change the position x axis to -3;
15. Let's create a script called "StartForce" and add it to our ball:



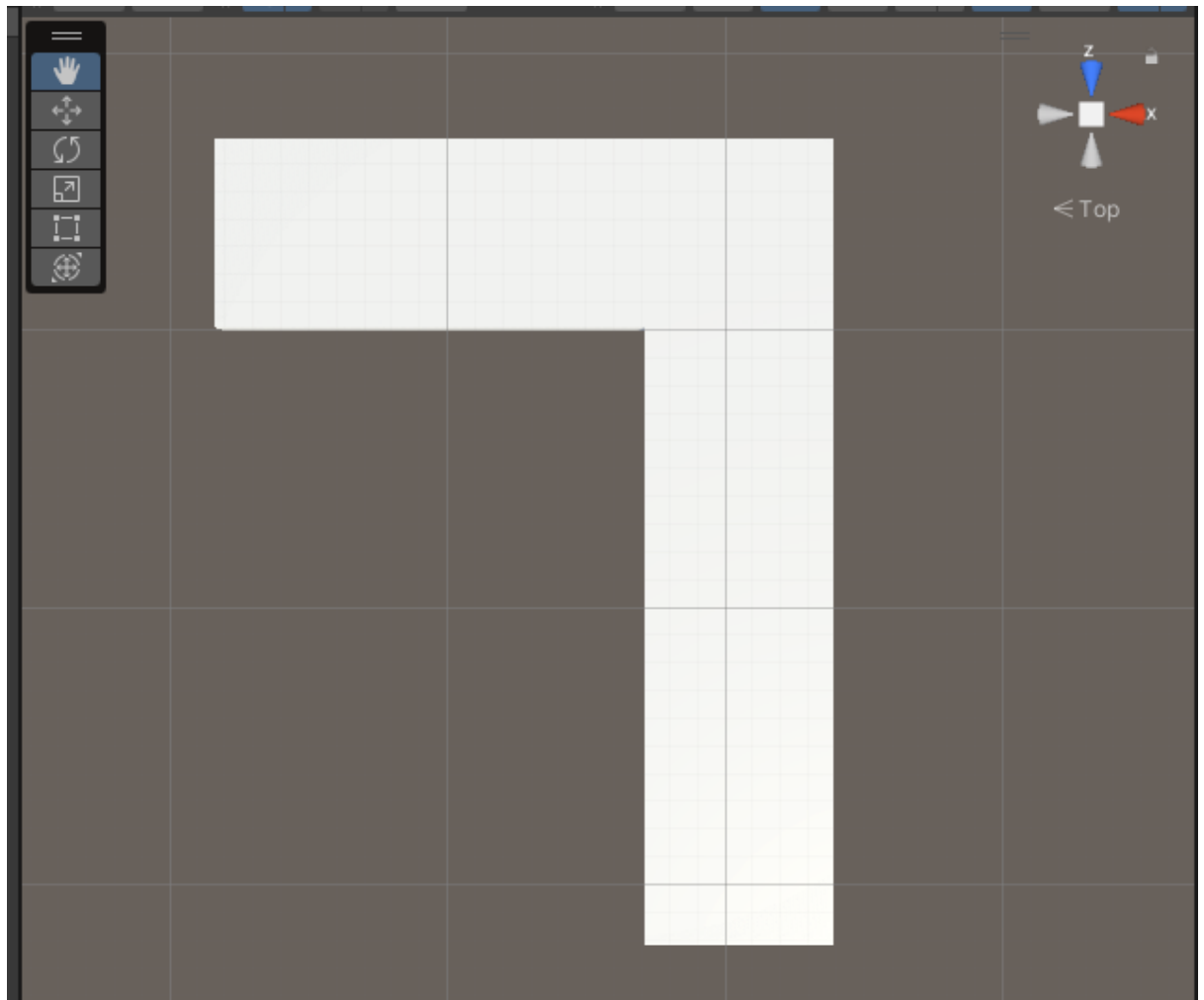
```

StartForce.cs
Assembly-CSharp
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class StartForce : MonoBehaviour
6 {
7     public Vector3 startForce;
8
9     // Start is called before the first frame update
10    void Start()
11    {
12        Rigidbody rigidbody = GetComponent<Rigidbody>();
13        rigidbody.AddForce(startForce, ForceMode.Impulse);
14    }
15
16    // Update is called once per frame
17    void Update()
18    {
19    }
20
21 }
22
  
```

16. Now select the ball in the hierarchy and go into the inspector to change its start force value to 2 in the x axis;
17. The ball will bounce into the wall but not bounce back, let's fix this;
18. Edit > Project Settings > Physics change the Bounce Threshold to 0;

Resident Evil Fixed Camera Style

1. Install cinemachine package;
2. Create two simple corridors in the scene view, something that from the Top Down view will look like this:



- 3.
4. Then add a Box Collider to each corridor, name them Trigger Zone 1 and Trigger Zone 2. Make sure to tick the Is Trigger box;
5. Now click on Cinemachine in the top bar and click on Create Virtual Camera, create two them and position them to face each respective corridor;
6. Create a script called CamSwitcher

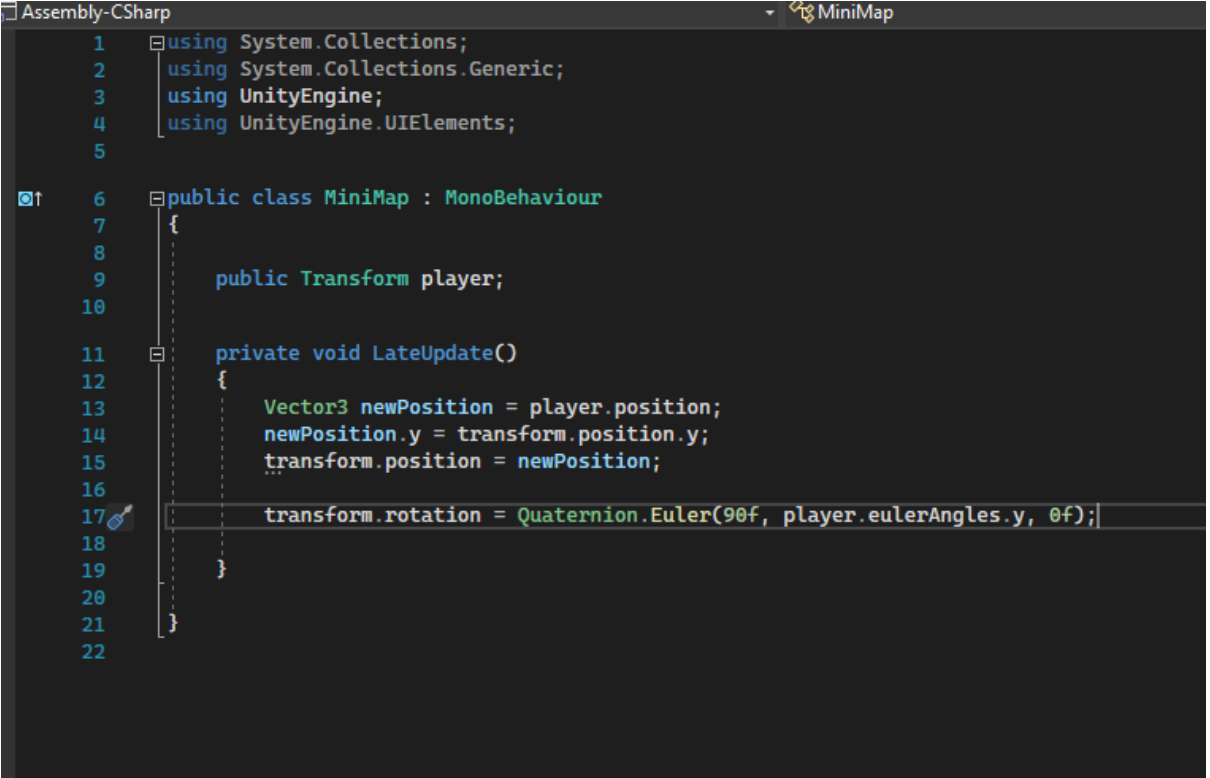
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Cinemachine;
5
6  public class CamSwitcher : MonoBehaviour
7  {
8
9      public Transform Player;
10     public CinemachineVirtualCamera activeCam;
11     private void OnTriggerEnter(Collider other)
12     {
13         if (other.CompareTag("Player"))
14         {
15             activeCam.Priority = 1;
16         }
17     }
18
19     private void OnTriggerExit(Collider other)
20     {
21         if (other.CompareTag("Player"))
22         {
23             activeCam.Priority = 0;
24         }
25     }
26 }
27
```

- 7.
8. Make sure your Player has the tag set to "Player";
9. Now add this script to your Trigger Zones, drag the Player into Player and the respective virtual camera to the Active Cam.

Mini Map

1. Create another Camera, name it MinimapCamera and rearrange it to face it downwards on the player
2. Click on the Main Camera and in the inspector left click and select copy component. Select MinimapCamera, left click in the inspector and click paste component.
3. Change the projection of MinimapCamera from Perspective to Orthographic.
4. Now create a new object in the hierarchy, go to UI and click Raw Image, rename this to Minimap.
5. In the inspector of Minimap change the width to 170 and height to the same value. Change Pos X to -110 and Pos Y to the same value.

6. In the Assets folder left click and select Render Texture, rename it to MinimapRT. In the inspector change the size to 170 x 170, and change the Depth Buffer to No depth buffer.
7. Now click on MinimapCamera in the hierarchy and in the inspector where it says Target Texture drag the MinimapRT there.
8. Now click on Minimap in the hierarchy and drag the MinimapRT into the Texture.
9. Create a new script called Minimap and assign it to the MinimapCamera.



```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UIElements;
5
6  public class MiniMap : MonoBehaviour
7  {
8
9      public Transform player;
10
11     private void LateUpdate()
12     {
13         Vector3 newPosition = player.position;
14         newPosition.y = transform.position.y;
15         transform.position = newPosition;
16
17         transform.rotation = Quaternion.Euler(90f, player.eulerAngles.y, 0f);
18     }
19
20
21
22

```

10.

AI

1. Let's start by creating a script called EnemyAI.
2. In the heading of the script, where you can see all the "using System.Collections", "using UnityEngine" etc... make sure to add "using UnityEngine.AI" this will allow the script to use the navmesh agent, which is necessary for the AI to move around the scene.
3. Now create the following variables

```

public NavMeshAgent navMeshAgent;
public float startWaitTime = 4;
public float timeToRotate = 2;
public float speedWalk = 6;
public float speedRun = 9;

public float viewRadius = 15;
public float viewAngle = 90;
public LayerMask playerMask;
public LayerMask obstacleMask;
public float meshResolution = 1f;
public int edgeIterations = 4;
public float edgeDistance = 0.5f;

public Transform[] waypoints;
int m_CurrentWaypointIndex;

Vector3 playerLastPosition = Vector3.zero;
Vector3 m_PlayerPosition;

float m_WaitTime;
float m_TimeToRotate;
bool m_PlayerInRange;
bool m_PlayerNear;
bool m_IsPatrol;
bool m_CaughtPlayer;

```

- 4.
5. They are pretty self explanatory, the first paragraph of variables specify at what speed the AI will walk, run, wait to move to another point etc... The other paragraph are variables for the enemy view angle and radius. The script also has an array for the waypoints a.k.a the points which the AI will follow when patrolling.

```

void Start()
{
    m_PlayerPosition = Vector3.zero;
    m_IsPatrol = true;
    m_CaughtPlayer = false;
    m_PlayerInRange = false;
    m_WaitTime = startWaitTime;
    m_TimeToRotate = timeToRotate;

    m_CurrentWaypointIndex = 0;
    navMeshAgent = GetComponent<NavMeshAgent>();

    navMeshAgent.isStopped = false;
    navMeshAgent.speed = speedWalk;
    navMeshAgent.SetDestination(waypoints[m_CurrentWaypointIndex].position);
}

```

- 6.

7.

```

Unity Message | 0 references
void Update()
{
    EnviromentView();

    if (!m_IsPatrol)
    {
        Chasing();
    }
    else
    {
        Patrolling();
    }
}

```

8.

```

private void Chasing()
{
    m_PlayerNear = false;
    playerLastPosition = Vector3.zero;

    if (!m_CaughtPlayer)
    {
        Move(speedRun);
        navMeshAgent.SetDestination(m_PlayerPosition);
    }
    if(navMeshAgent.remainingDistance <= navMeshAgent.stoppingDistance)
    {
        if(m_WaitTime <= 0 && !m_CaughtPlayer && Vector3.Distance(transform.position, GameObject.FindGameObjectWithTag("Player").transform.position)>= 6f)
        {
            m_IsPatrol=true;
            m_PlayerNear = false;
            Move(speedWalk);
            m_TimeToRotate = timeToRotate;
            m_WaitTime = startWaitTime;
            navMeshAgent.SetDestination(maypoints[m_CurrentWaypointIndex].position);
        }
        else
        {
            if(Vector3.Distance(transform.position, GameObject.FindGameObjectWithTag("Player").transform.position)>= 2.5f)
            {
                Stop();
                m_WaitTime -= Time.deltaTime;
            }
        }
    }
}
}

```

1 reference

```
private void Patrolling()
{
    if (m_PlayerNear)
    {
        if(m_TimeToRotate <= 0)
        {
            Move(speedWalk);
            LookingPlayer(playerLastPosition);
        }

        else
        {
            Stop();
            m_TimeToRotate -= Time.deltaTime;
        }
    }

    else
    {
        m_PlayerNear = false;
        playerLastPosition = Vector3.zero;
        navMeshAgent.SetDestination(waypoints[m_CurrentWaypointIndex].position);
        if(navMeshAgent.remainingDistance <= navMeshAgent.stoppingDistance)
        {
            if (m_WaitTime <= 0)
            {
                NextPoint();
                Move(speedWalk);
                m_WaitTime = startWaitTime;
            }
            else
            {
                Stop();
                m_WaitTime -= Time.deltaTime;
            }
        }
    }
}
```

5 references

9.


```

5 references
void Move(float speed)
{
    navMeshAgent.isStopped = false;
    navMeshAgent.speed = speed;
}

4 references
void Stop()
{
    navMeshAgent.isStopped = true;
    navMeshAgent.speed = 0;
}

1 reference
public void NextPoint()
{
    m_CurrentWaypointIndex = (m_CurrentWaypointIndex + 1) % waypoints.Length;
    navMeshAgent.SetDestination(waypoints[m_CurrentWaypointIndex].position);
}

0 references
void CaughtPlayer()
{
    m_CaughtPlayer = true;
}

1 reference
void LookingPlayer (Vector3 player)
{
    navMeshAgent.SetDestination(player);
    if(Vector3.Distance(transform.position, player) <= 0.3)
    {
        if(m_WaitTime <= 0)
        {
            m_PlayerNear = false;
            Move(speedWalk);
            navMeshAgent.SetDestination(waypoints[m_CurrentWaypointIndex].position);
            m_WaitTime = startWaitTime;
            m_TimeToRotate = timeToRotate;
        }

        else
        {
            Stop();
            m_WaitTime -= Time.deltaTime;
        }
    }
}

```

10.

11. The vector 3 player in LookingPlayer which represents the last known position of the player, the AI will go to that last known position and return to its Patrolling state.

```

1 reference
void EnviromentView()
{
    Collider[] playerInRange = Physics.OverlapSphere(transform.position, viewRadius, playerMask);

    for(int i = 0; i < playerInRange.Length; i++)
    {
        Transform player = playerInRange[i].transform;
        Vector3 dirToPlayer = (player.position - transform.position).normalized;
        if(Vector3.Angle(transform.forward, dirToPlayer) < viewAngle / 2)
        {
            float dstToPlayer = Vector3.Distance(transform.position, player.position);
            if(!Physics.Raycast(transform.position, dirToPlayer, dstToPlayer, obstacleMask))
            {
                m_PlayerInRange = true;
                m_IsPatrol = false;
            }

            else
            {
                m_PlayerInRange = false;
            }
        }

        if(Vector3.Distance(transform.position, player.position) > viewRadius)
        {
            m_PlayerInRange = false;
        }

        if (m_PlayerInRange)
        {
            m_PlayerPosition = player.transform.position;
        }
    }
}

```

12.

13. The EnviromentView method pretty much allows the enemy to be able to detect the player and the obstacle. The first bit of the method creates a sphere around the enemy which allows it to detect the player and if the player in the radius of the enemy view the player's position will be registered. The else statement checks if the player is behind an obstacle, if that's true the player's position will not be registered and the Enemy AI will not chase them.