Enemy AI

1. **Create a new scene**

Start by creating a new scene called “AI Enemy”
Download and add the NaveMesh Components from the Unity Github.
Add a 3D cube named “Floor”.
Make a new blank material and assign it to the floor.
Create a layer and call it “whatIsGround” and assign it to the floor.
Finally add a NavMeshSurface script to the floor and select the layers you dont want to include by deselecting them from the drop down menu and bake the map.

Create two capsules one will be the enemy and the other will be the player.
Create and assign two materials one for the enemy and the other for the player.
On the enemy capsule youll add a NavMeshAgent component and leave it by default.
On the player we will add a player controller script and character controller component.
Finally creating a layer called “whatIsPlayer” and adding it to the player.

2. **Create the AI script.**

Create a new script called “AIEnemy”.

First thing to do is to add the reference to the UnityEngine reference at the top of the script.

```
using UnityEngine.AI;
```

Secondly we add the references, which will help us determine several options later in the Unity editor. We can start referencing the NavMeshAgent, the Player’sTransform(positions,scale,rotation of the player) and reference the layers that we have created at the beginning. These three references will be added later on inside of the Unity editor.

```
public NavMeshAgent agent;

public Transform player;

public LayerMask whatIsGround, whatIsPlayer;
```

```
private void Awake()
{
    player = GameObject.Find("Capsule").transform;
    agent = GetComponent<NavMeshAgent>();
}
```

3. **Add Spheres**

Next we will create two spheres to see if the player in on sight and in attacking range by creating a bool and a float that will check for the ranges. The float will check for the enemies range and the bool will check to see if the player is within the float's range.

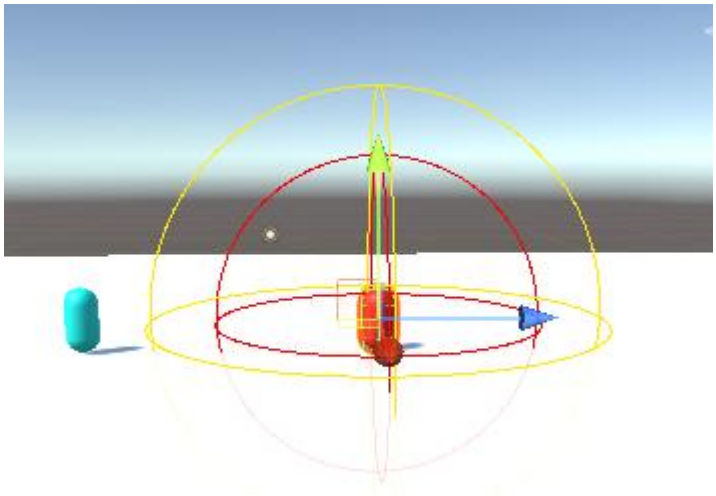
```
[Header("States")]
public float sightRange, attackRange;
public bool playerInSightRange, playerInAttackRange;
```

With the previous references we will be able check for the player position within the sight range and also within the attack range by checking the players position the enemy will know how to react.

```
void Update()
{
    //Check for the player to be in sight and attacking range
    playerInSightRange = Physics.CheckSphere(transform.position, sightRange, whatIsPlayer);
    playerInAttackRange = Physics.CheckSphere(transform.position, attackRange, whatIsPlayer);
```

Finally, if you want to show the sphere colliders within the unity editor you can use the function called "OnDrawGizmos()".

```
private void OnDrawGizmosSelected()
{
    Gizmos.color = Color.red;
    Gizmos.DrawWireSphere(transform.position, attackRange);
    Gizmos.color = Color.yellow;
    Gizmos.DrawWireSphere(transform.position, sightRange);
}
```



4. **<u>State what the enemy will do when he is within each range</u>**

Finally we are going to state the behaviour of the enemy as the player enters within the two ranges created before. First of all we are going to make some references that we will use later on and can be manipulated in the unity editor too.

```
[Header("Patroling")]
public Vector3 walkPoint;
bool walkPointSet;
public float walkPointRange;

[Header("Attacking")]
public float timeBetweenAttacks;
bool alreadyAttacked;
```

With these two sections we will be able to determine what the enemies will do when they get near the player. First of all we are going to set what will happen to the AI in different situations and we will do the following in void Update:

```
    //If the player is not on sight or in attack range
    if (!playerInSightRange && !playerInAttackRange)
    {
        Patrolling();
    }

    //If the player is on sight but not in attack range
    if (playerInSightRange && !playerInAttackRange)
    {
        ChasePlayer();
    }

    //If the player is on sight and in attack range
    if (playerInSightRange && playerInAttackRange)
    {
        AttackPlayer();
    }
}
```

With the functions we have created we will be righting up what will happen in each occasion, if the player have been seen but it is not in attack range the AI will chase the player where he was last seen but if he doesn't find him patrols and it does does with the following code in its respective voids:

```
60      private void Patrolling()
61      {
62          if (!walkPointSet)
63          {
64              SearchWalkPoint();
65          }
66
67          if (walkPointSet)
68          {
69              agent.SetDestination(walkPoint);
70          }
71
72          //Calculate point distance
73          Vector3 distanceToWalkPoint = transform.position - walkPoint;
74
75          //if walk point reached
76
77          if (distanceToWalkPoint.magnitude <1f)
78          {
79              walkPointSet = false;
80          }
81      }
82      private void SearchWalkPoint()
83      {
84          //Calculating a random point within range to patrol
85          float randomZ = Random.Range(-walkPointRange, walkPointRange);
86          float randomX = Random.Range(-walkPointRange, walkPointRange);
87
88          walkPoint = new Vector3(transform.position.x + randomX, transform.position.y, transform.position.z + randomZ);
89
90          //Checking if the point is on the ground
91          if (Physics.Raycast(walkPoint, -transform.up, 2f, whatIsGround))
92          {
93              walkPointSet = true;
94          }
95      }
```

And this other section will chase the player if seen:

```
private void ChasePlayer()
{
    //Chase player position
    agent.SetDestination(player.position);
}
```

Finally if the player has been see and its in attack range the AI will attack the player by going where they are with the following code:

```
103     private void AttackPlayer()
104     {
105         agent.SetDestination(player.position);
106
107         transform.LookAt(player);
108     }
109
```

5. Testing

After finishing the code we will assign it to the Enemy and give values to each property as well as assign the slots properly. To finalise we will test if it works.