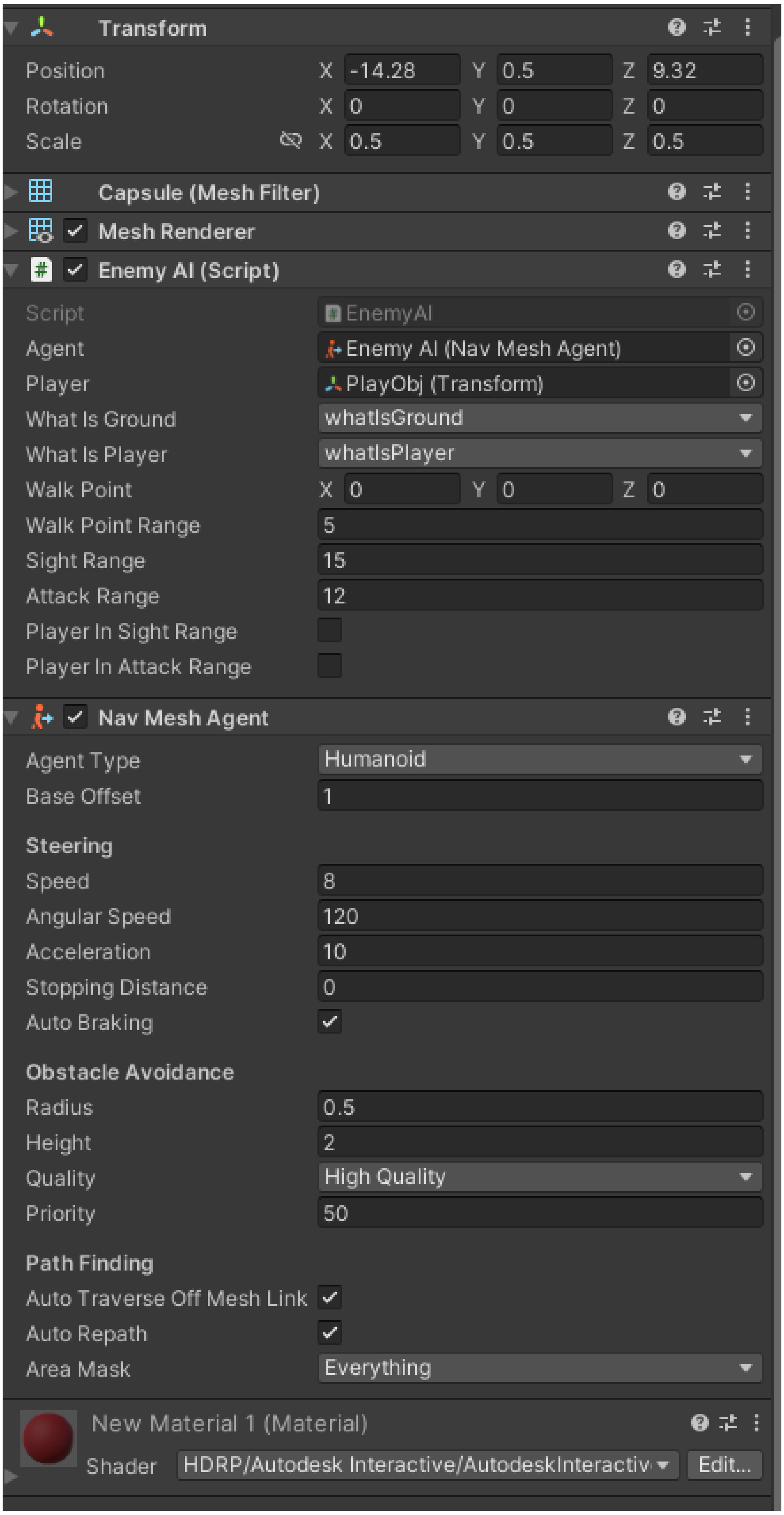
3D Enemy AI

1. First add a plane to the scene, rename it ‘Ground’ and scale it up to 5x5x5. Then add a 3D object capsule to the scene.
2. Name the 3D capsule ‘EnemyAI’ and delete the capsule collider because you wont need it. Add a new nav mesh agent component and increase the speed to 11 and the acceleration to 12. A nav mesh agent helps identify the areas a player caan walk or avoid in the environment.
3. Make a new script called ‘EnemyAI’. At the top add “Using UnityEngine.UI;” because we are using a nav mesh agent.
4. Make 3 public variables, one being a NavMeshAgent named ‘agent’ to get reference to the nav mesh on the player, another being a Transform called ‘ player’ to get reference to the player object and the final one being a 2 LayerMask labelled ‘whatIsGround’ and ‘whatIsPlayer’. This is so the objects and script knows how to interact with one another.
5. Then make a public Vector3 called ‘walkPoint’, a public float called ‘walkPointRange’ and a bool called ‘walkPointSet’; These will be used for enemy patrolling.
6. For the enemy states, you need to make a public float named ‘sightRange, attackRange’ and a public bool called ‘playerInSightRange, playerInAttackRange’.
7. Delete the ‘void start’ because you wont need it, and create a new private void awake, and inside type ‘player = GameObject.Find("Player").transform;’ because we need to get a reference to the player's position in the world. Then do ‘agent = GetComponent<NavMeshAgent>();’ to get a reference to the navmesh of the enemy.
8. Make the void update private, then inside make ‘ playerInSightRange = Physics.CheckSphere(transform.position, sightRange, whatIsPlayer);’ and ‘playerInAttackRange = Physics.CheckSphere(transform.position, attackRange, whatIsPlayer);’. This is to check if the player is in sight or attack range by using the physics function in Unity.
9. After this type these 3 if statements; ‘if (!playerInSightRange && !playerInAttackRange) Patrolling();’ Meaning if the player isn't in sight or attack range, continue patrolling the area. ‘if (playerInSightRange && !playerInAttackRange) ChasePlayer();’ Meaning if the player is in sight range, but not in attack range, chase the player. ‘if (playerInSightRange && playerInAttackRange) AttackPlayer();’ meaning if the player is in sight and attack range, attack the player.
10. Patrolling (), ChasePlayer() and AttackPlayer() will be new functions we make, so make a private void for each of these.
11. Inside ‘private void Patrolling’ make 2 if statements, ‘if (!walkPointSet) SearchWalkPoint();’ Meaning if the walk point is not set for the AI enemy, it will just search the area. ‘ if (walkPointSet) agent.SetDestination(walkPoint);’ meaning if the walk point is set the enemy AI will walk to it.
12. Create a new private void function called ‘SearchWalkPoint’ and then make 2 floats, ‘float randomZ = Random.Range(-walkPointRange, walkPointRange);’ This will return a random value depending on high your walk point value is; and ‘ float randomX = Random.Range(-walkPointRange, walkPointRange); Then make the ‘walkPoint = new Vector3(transform.position.x + randomZ, transform.position.y, transform.position.z + randomZ); This is to calculate a random point and direction in range for the AI enemy to patrol.

13. Finally, add a if statement that states, ‘if(Physics.Raycast(walkPoint, -transform.up, 2f, whatIsGround)) walkPointSet = true;’ This is to make sure to walk point formulated is actually on the ground and within the area. If not it would be equal to false a new point will be made for the enemy AI to walk to.
14. Back to the Patrolling () function, type Vector3 distanceToWalkPoint = transform.positon - walkPoint; to calculate the distance to each walk point. Then type if (distanceToWalkPoint.magnitude < 1f) walkPointSet = false; This means if the distance is less than 1 unity unit, make it equal to false to calculate a new walk point.
15. In the ‘private void ChasePlayer()’ function type, ‘agent.SetDestination(player.position);’ to make sure the enemy follows the player when it sees it.
16. In the ‘private void AttackPlayer()’ , do ‘agent.SetDestination(player.position);’ and then below that type, ‘transform.LookAt(player);’ to make the enemy AI stop moving to the set destination point and go towards the players position and also look at the player when it is attacking them.
17. Now back in Unity, add the script to the EnemyAI object. From the hierarchy, drag the Enemy AI to the agent slot on the script, drag the player object to the ‘player obj’.
18. Next make 2 new layers and call them ‘whatIsGround’ and ‘whatIsPlayer’ and then select them in the aligned places in the Enemy AI script. Give the walk point range a value of 5, sight range a value of 20 and attack range the value 12 - These will all be unity units.
19. Finally, open the navigation window and then select bake, then click bake to bake the area and to make walkable paths for the enemy AI. When you play the game, your Enemy AI should be working.



Transform

Position X -14.28 Y 0.5 Z 9.32

Rotation X 0 Y 0 Z 0

Scale X 0.5 Y 0.5 Z 0.5

Capsule (Mesh Filter)

☑ Mesh Renderer

☑ Enemy AI (Script)

Script: EnemyAI

Agent: Enemy AI (Nav Mesh Agent)

Player: PlayObj (Transform)

What Is Ground: whatIsGround

What Is Player: whatIsPlayer

Walk Point X 0 Y 0 Z 0

Walk Point Range: 5

Sight Range: 15

Attack Range: 12

Player In Sight Range: ☐

Player In Attack Range: ☐

☑ Nav Mesh Agent

Agent Type: Humanoid

Base Offset: 1

Steering

Speed: 8

Angular Speed: 120

Acceleration: 10

Stopping Distance: 0

Auto Braking: ☑

Obstacle Avoidance

Radius: 0.5

Height: 2

Quality: High Quality

Priority: 50

Path Finding

Auto Traverse Off Mesh Link: ☑

Auto Repath: ☑

Area Mask: Everything

New Material 1 (Material)

Shader: HDRP/Autodesk Interactive/AutodeskInteractiv