Void Start is Responsible for Defining our variables and setting the origin point of our gameobjects.

void Start()

{

agent = GetComponentInParent<NavMeshAgent>();

transform.AddComponent<MeshRenderer>().material = VisionConeMaterial;

MeshFilter\_ = transform.AddComponent<MeshFilter>();

VisionConeMesh = new Mesh();

VisionAngle \*= Mathf.Deg2Rad;

target = GameObject.Find("Target").transform;

//centrePoint = GameObject.Find("cigSpawn").transform;

if (this.gameObject.tag == "CIG")

{

centrePoint = GameObject.Find("cigSpawn").transform;

}

if (this.gameObject.tag == "FLES")

{

centrePoint = GameObject.Find("flesSpawn").transform;

}

if (this.gameObject.tag == "Haak")

{

centrePoint = GameObject.Find("haakSpawn").transform;

}

}

In the update we call DrawVisionCone which is responsible for visualizing how far an enemy can see and which objects obstruct the enemies vision (the objects that obstruct an enemies vision are defined in a layermask) and for chasing the player whenever he is in the visioncone if the player is not inside the visioncone the enemy will instead walk around randomly within the room.

void Update()

{

VisionAngle = 2;

DrawVisionCone();//calling the vision cone function everyframe just so the cone is updated every frame

if (agent.remainingDistance <= agent.stoppingDistance) //done with path

{

Vector3 point;

if (RandomPoint(centrePoint.position, range, out point)) //pass in our centre point and radius of area

{

Debug.DrawRay(point, Vector3.up, Color.blue, 1.0f); //so you can see with gizmos

agent.SetDestination(point);

}

}

}

//Random Movement within sphere

bool RandomPoint(Vector3 center, float range, out Vector3 result)

{

Vector3 randomPoint = center + Random.insideUnitSphere \* range; //random point in a sphere

NavMeshHit hit;

if (NavMesh.SamplePosition(randomPoint, out hit, 1.0f, NavMesh.AllAreas)) //documentation: https://docs.unity3d.com/ScriptReference/AI.NavMesh.SamplePosition.html

{

//the 1.0f is the max distance from the random point to a point on the navmesh, might want to increase if range is big

//or add a for loop like in the documentation

result = hit.position;

return true;

}

result = Vector3.zero;

return false;

}

void DrawVisionCone()//this method creates the vision cone mesh

{

int[] triangles = new int[(VisionConeResolution - 1) \* 3];

Vector3[] Vertices = new Vector3[VisionConeResolution + 1];

Vertices[0] = Vector3.zero;

float Currentangle = -VisionAngle / 2;

float angleIcrement = VisionAngle / (VisionConeResolution - 1);

float Sine;

float Cosine;

for (int i = 0; i < VisionConeResolution; i++)

{

Sine = Mathf.Sin(Currentangle);

Cosine = Mathf.Cos(Currentangle);

Vector3 RaycastDirection = (transform.forward \* Cosine) + (transform.right \* Sine);

Vector3 VertForward = (Vector3.forward \* Cosine) + (Vector3.right \* Sine);

if (Physics.Raycast(transform.position, RaycastDirection, out RaycastHit hit, VisionRange, VisionObstructingLayer))

{

PlayerSeen = false;

Vertices[i + 1] = VertForward \* hit.distance;

}

else

{

Vertices[i + 1] = VertForward \* VisionRange;

}

if (Physics.Raycast(transform.position, RaycastDirection, out RaycastHit Enemy, VisionRange, EnemyLayer) && (!Physics.Raycast(transform.position, RaycastDirection, out RaycastHit \_hit, VisionRange, VisionObstructingLayer)))

{

PlayerSeen = true;

Debug.Log("Player seen");

if (Vector3.Distance(destination, target.position) > 1.0f && PlayerSeen == true)

{

destination = target.position;

agent.destination = destination;

}

}

else

{

PlayerSeen = false;

}

Currentangle += angleIcrement;

}

for (int i = 0, j = 0; i < triangles.Length; i += 3, j++)

{

triangles[i] = 0;

triangles[i + 1] = j + 1;

triangles[i + 2] = j + 2;

}

VisionConeMesh.Clear();

VisionConeMesh.vertices = Vertices;

VisionConeMesh.triangles = triangles;

MeshFilter\_.mesh = VisionConeMesh;

}

}