JANNE SILLANPÄÄ

```
public class EnemyAI : MonoBehaviour
{
    private GameObject player;
    private GameObject[] wps;

    private bool playerIsBeingSeen;
    private bool inCombat;
    private bool alliesInCombat;

    private List<EnemyAI> nearbyUnits;
    private EnemyAI closestAllyInCombat;
    private Vector3 playerLastSightedLocation;

    private float attackDistance;
    private float movementSpeed;
    private float fovAngle;
    private float fovDistance;

    private string enemyRole;

    //SHOOTING
    private float fireRate;
    private float nextFire;
    private float rotationLeft = 360;
    private GameObject projectile;
    private Transform projectileSpawn;

    private NavMeshAgent navMeshAgent;
    private Vector3 currentDestination;
    private Quaternion rot;

    // FSM
    private FSM stateMachine;
    private FSM.IFSMStates idleState, pursueState, patrolState, attackState, flankingState,
supportState;

    private MasterAIController masterAIController;
    private EnemyInfo enemyInfo;
```

```
void Start()
    {

        masterAIController =
        GameObject.Find("MasterAIController").GetComponent<MasterAIController>();

        enemyInfo = GetComponent<EnemyInfo>();


        stateMachine = new FSM();
        navMeshAgent = gameObject.GetComponent<NavMeshAgent>();

        // Assign values from Player Info class
        player = enemyInfo.player;
        fovAngle = enemyInfo.fovAngle;
        fovDistance =   enemyInfo.fovDistance;
        wps = enemyInfo.wps;
        fireRate = enemyInfo.fireRate;
        projectile = enemyInfo.projectile;
        projectileSpawn = enemyInfo.projectileSpawn;
        attackDistance = enemyInfo.attackDistance;
        movementSpeed = enemyInfo.movementSpeed;

        // Initialize
        EnemyGroup();
        CreateState();
        SetRole("");
        masterAIController.AddEnemy(gameObject.GetComponent<EnemyAI>());
        stateMachine.Push(idleState);
    }






void Update()
    {
        FOV();
        stateMachine.Update(gameObject);

        // Update enemy group
        if (Random.Range(0, 100) < 20)
        {
            EnemyGroup();
        }


    }

// Field of View update function
    private void FOV()
    {
        Vector3 direction = player.transform.position - transform.position;
        float angle = Vector3.Angle(direction, transform.forward);
        playerIsBeingSeen = false;

        if(angle < fovAngle / 2)
        {
            Debug.DrawRay(transform.position, transform.forward * fovDistance, Color.green);
            if (Physics.Raycast(transform.position, direction.normalized, out RaycastHit hit,
fovDistance))
            {
                if (hit.collider.gameObject == player)
                {
                    playerIsBeingSeen = true;
                    playerLastSightedLocation = player.transform.position;
                    Debug.Log("Player is being seen");
                }
            }
        }

    }
```

```
// All the states
    private void CreateState()
    {

        // IDLE STATE
        idleState = (fsm, gameObj) =>
        {
            Debug.Log("In idle state");

              // The Enemy sees the player so move into doing correct action depending on the
              current role
            if (playerIsBeingSeen)
            {
                inCombat = true;
                fsm.Pop();
                if (enemyRole == "Attacker")
                {
                    fsm.Push(pursueState);
                }
                else if (enemyRole == "Support")
                {
                    fsm.Push(supportState);
                }
                else if (enemyRole == "Flanker")
                {
                    currentDestination = transform.position;
                    fsm.Push(flankingState);
                }
                return;
            } // We dont see the enemy so by default go into patrol mode
            else
            {
                currentDestination = wps[Random.Range(0, wps.Length)].transform.position;
                navMeshAgent.SetDestination(currentDestination);
                inCombat = false;
                fsm.Pop();
                fsm.Push(patrolState);
            }

        };


        // ATTACK STATE
        attackState = (fsm, gameObj) =>
        {
            Debug.Log("In attack state");

            // If we are in attack range then attack
            if (Vector3.Distance(transform.position, player.transform.position) < attackDistance)
            {
                transform.rotation = Quaternion.Slerp(transform.rotation,
Quaternion.LookRotation(player.transform.position - transform.position), Time.deltaTime *
movementSpeed);
                if (Time.time > nextFire)
                {
                    Fire();
                }
            } // Pursue the enemy if we are not in attack range
            else
            {
                fsm.Pop();
                fsm.Push(pursueState);
            }
        };




       // PURSUE STATE
        pursueState = (fsm, gameObj) =>
        {
            Debug.Log("In pursue state");

            // If the player is in sight then constantly move towards player location
            if (Vector3.Distance(transform.position, player.transform.position) > attackDistance &
playerIsBeingSeen)
            {
                navMeshAgent.SetDestination(player.transform.position);
            } // We are in range to the enemy so either attack or start flanking
            else if (playerIsBeingSeen)
            {
                inCombat = true;
                navMeshAgent.ResetPath();
                fsm.Pop();
                if (enemyRole == "Flanker")
                {
                    fsm.Push(flankingState);
                }
                else
                {
                    fsm.Push(attackState);
                }
                return;
            }

            // We have lost sight to the player
            if (!playerIsBeingSeen)
            {
                inCombat = false;
                CheckAlliesInCombat();

                // If nearby enemies are currently not in combat then move to the last sighted
                location of the player
                if (Vector3.Distance(transform.position, playerLastSightedLocation) > attackDistance
&& !alliesInCombat)
                {
                    navMeshAgent.SetDestination(playerLastSightedLocation);
                    Debug.Log("MOVE TO INVESTIGATE");
                } // If nearby enemies are in combat then move to player's current location
                else if (alliesInCombat)
                {
                    navMeshAgent.SetDestination(player.transform.position);
                }
                else // If we have reached the players last location and no allies are in   combat
nearby then move to idle state
                {
                    navMeshAgent.ResetPath();

                    // Make the enemy rotate 360° to look for enemies
                    Debug.Log("Rotating");
                    float rotation = 200f * Time.deltaTime;

                    if (rotationLeft > rotation)
                    {
                        rotationLeft -= rotation;
                    }
                    else
                    {
                        rotation = rotationLeft;
                        rotationLeft = 360;
                        inCombat = false;
                        fsm.Pop();
                        fsm.Push(idleState);
                        return;
                    }

                    transform.Rotate(0, rotation, 0);
                }

            }
        };



        // PATROL STATE
        patrolState = (fsm, gameObj) =>
        {
            Debug.Log("In patrol state");

            CheckAlliesInCombat();

            // The enemy has spotted the player so move into to the correct state
            if (playerIsBeingSeen || alliesInCombat)
            {
                fsm.Pop();
                inCombat = true;

                if (enemyRole == "Attacker")
                {
                    fsm.Push(pursueState);
                }
                else if (enemyRole == "Support")
                {
                    fsm.Push(supportState);
                }
                else if (enemyRole == "Flanker")
                {
                    fsm.Push(flankingState);
                }
                return;
            }

            // Continue normal patrol
            if (Vector3.Distance(transform.position, currentDestination) <= 1f)
            {
                currentDestination = wps[Random.Range(0, wps.Length)].transform.position;
                navMeshAgent.SetDestination(currentDestination);
            }

        };


        // FLANKING STATE
        flankingState = (fsm, gameObj) =>
        {
            Debug.Log("In flanking state");

            CheckAlliesInCombat();
            transform.LookAt(player.transform.position);
            navMeshAgent.angularSpeed = 0;

            // Pick a random location near while we are in attack range
            if (Vector3.Distance(transform.position, currentDestination) <= 1f ||
Vector3.Distance(transform.position, player.transform.position) > attackDistance)
            {
                Vector3 targetPlace = Random.insideUnitSphere * 20;

                // Check the target location is not near another enemy and is still close to the
player
                if (Vector3.Distance(targetPlace, closestAllyInCombat.transform.position) >= 2f &&
Vector3.Distance(targetPlace, player.transform.position) < attackDistance)
```

```
                {
                    currentDestination = targetPlace;
                    navMeshAgent.SetDestination(currentDestination);

                }
            }

            // Keep attacking if we are in range
            if (Vector3.Distance(transform.position, player.transform.position) < attackDistance &&
playerIsBeingSeen)
            {

                if (Time.time > nextFire)
                {
                    Fire();
                }
            } // We are no longe in range so move to pursue
            else
            {
                navMeshAgent.angularSpeed = 120f;
                fsm.Pop();
                fsm.Push(pursueState);
            }

        };


        // SUPPORT STATE
        supportState = (fsm, gameObj) =>
        {
            Debug.Log("In support state");

            CheckAlliesInCombat();

            // If allies are in combat then move to support the nearest enemy
            if (alliesInCombat)
            {
                if (Vector3.Distance(transform.position, closestAllyInCombat.transform.position) >
5f)
                {
                    navMeshAgent.SetDestination(closestAllyInCombat.transform.position -
closestAllyInCombat.transform.forward);
                }
            } // No longer in combat, go to idle
            else
            {
                inCombat = false;
                fsm.Pop();
                fsm.Push(pursueState);
            }

        };

    }




// Create a group based on nearby enemies
    private void EnemyGroup()
    {
        List<EnemyAI> allEnemies;
        allEnemies = masterAIController.GetEnemies();

        float distance;

        List<EnemyAI> nearby = new List<EnemyAI>();
        foreach (EnemyAI enemy in allEnemies)
        {
            if (enemy.gameObject != gameObject)
            {
                distance = Vector3.Distance(enemy.gameObject.transform.position, 
transform.position);

                // All enemies that are in range are in this group
                if (distance <= 10f)
                {
                    nearby.Add(enemy);
                }
            }
        }


        nearbyUnits = nearby;

    }

    // Check if allies in our group are currently in combat
    private void CheckAlliesInCombat()
    {
        bool alliedCombat = false;
        foreach (EnemyAI enemy in nearbyUnits)
        {
            if (enemy.GetIsInCombat())
            {
                closestAllyInCombat = enemy;
                alliedCombat = true;
            }
        }

        alliesInCombat = alliedCombat;
    }





 // Simple Move and Fire functions
    private void Move(Vector3 towards)
    {
        transform.rotation = Quaternion.Slerp(transform.rotation, Quaternion.LookRotation(towards - 
transform.position), Time.deltaTime * movementSpeed);
        transform.Translate(0, 0, Time.deltaTime * movementSpeed);
    }

    private void Fire()
    {
        nextFire = Time.time + fireRate;
        Instantiate(projectile, projectileSpawn.position, projectileSpawn.rotation);
    }

    // Get and Set role for MasterAIController
    public void SetRole(string role) { enemyRole = role; }
    public string GetRole() { return enemyRole; }


    public bool GetIsInCombat() { return inCombat; }

}
```