using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyHandler : MonoBehaviour

{

private List<GameObject> Enemies = new List<GameObject>();

// Use this for initialization

public void AddToHandler(GameObject enem)

{

//Debug.Log("added to handler: " + enem.name);

Enemies.Add(enem);

}

public void AlertOthers(GameObject enem, float alertDistance)

{

//Debug.Log("Allerting others of disturbance at: "+ enem.transform.position);

Enemies.Remove(enem);

foreach (GameObject obj in Enemies)

{

if ((enem.transform.position - obj.transform.position).magnitude < alertDistance)

{

//Debug.Log(obj.name + "has been alerted");

obj.GetComponent<EnemyMovement>().SetWaypoint(enem);

obj.GetComponent<EnemyScript>().SetDisturbedLocation(enem.transform.position,true);

}

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AI;

public class EnemyMovement : MonoBehaviour

{

public bool IsRanged;

public enum State

{

none,

patroling,

walk,

stop,

shoot,

look,

knife

};

public AudioClip ShootSound;

public AudioClip KnifeSound;

private float \_volume;

private State \_state;

// change below to public if not NavMesh

public GameObject target;

public float Speed;

private NavMeshAgent navigator;

public int Wait;

public int LookWait;

public int InitialDelay;

public float MeleeDistance;

public float RangeDistance;

private GameObject \_waypoint;

private GameObject \_tempWaypoint;

private int \_wait;

private int \_lookWait;

private float \_distanceToTarget;

public bool \_inVision;

private List<GameObject> Waypoints;

Vector3 \_speed;

Vector3 \_rayDirection;

Vector3 \_moveDirection;

Vector3 \_lastKnownTargetPosition;

RigidbodyConstraints \_normalConstraints;

RigidbodyConstraints \_stoppedConstraints;

//[SerializeField] private GameObject[] \_weapons;

[SerializeField] private int \_rangedDamage;

[SerializeField] private int \_meleeDamage;

public State CurrentState

{

get { return \_state; }

}

// Use this for initialization

void Start()

{

\_tempWaypoint = new GameObject();

\_volume = Utils.EffectVolume();

//Debug.Log("effect volume = "+\_volume);

\_wait = InitialDelay;

\_stoppedConstraints = RigidbodyConstraints.FreezePosition;

\_normalConstraints = gameObject.GetComponent<Rigidbody>().constraints;

Waypoints = gameObject.GetComponent<EnemyScript>().Waypoints;

navigator = GetComponent<NavMeshAgent>();

\_speed = new Vector3(Speed \* 3, Speed \* 3, Speed \* 3);

if (Waypoints.Count <= 1)

gameObject.GetComponent<EnemyScript>().StartOffset = true;

\_waypoint = Waypoints[0];

if (!gameObject.GetComponent<EnemyScript>().StartOffset)

Patrol();

}

//private void Awake() {

// if (!IsRanged) {

// \_weapons[0].SetActive(true);

// } else {

// \_weapons[1].SetActive(true);

// }

//}

// Update is called once per frame

void Update()

{

if(Input.GetKeyDown(KeyCode.Z))

{

Debug.Log("My position is: " + gameObject.transform.position + " and I am heading to: " + \_waypoint.transform.position + " with the last known target position at: " + \_lastKnownTargetPosition);

}

//Debug.Log(navigator.isStopped);

if (target != null)

{

//Debug.Log(\_state +" with target in vision: "+\_inVision);

\_distanceToTarget = (target.transform.position - gameObject.transform.position).magnitude;

if (\_distanceToTarget < RangeDistance)

{

Utils.ChangeGameObjectColor(gameObject, Color.white);

CheckVision();

}

else

{

\_inVision = false;

}

if (\_inVision)

{

//Debug.Log("looking");

EnemyAttack();

if (\_lookWait >= LookWait)

{

navigator.isStopped = false;

//Debug.Log("stopped looking");

}

else

{

navigator.isStopped = true;

transform.LookAt(target.transform);

SetLastPositionToTarget();

\_lookWait++;

}

if (IsRanged)

{

\_wait = InitialDelay;

}

}

else

{

//Debug.Log(\_waypoint.transform.position + " si the waypoint with the last position = " + \_lastKnownTargetPosition + " | obj pos = " +transform.position);

if (\_waypoint == null)

{

Debug.Log(gameObject.name + " has null waypoint");

}

else

if ((transform.position - \_lastKnownTargetPosition).magnitude < 0.5f)

{

Debug.Log("at last known position");

//Debug.Log(\_wait);

if (\_wait <= 0)

{

//Debug.Log("Patrolling");

Patrol();

}

\_wait--;

}

else

{

//Debug.Log("walking");

\_wait = Wait;

\_state = State.walk;

}

}

if (\_state != State.shoot)

{

navigator.isStopped = false;

//gameObject.GetComponent<EnemyScript>().OnCheckpoint(target, true);

}

else if (IsRanged)

{

//Debug.Log("setting waypoiny to: " + target.transform.position);

transform.LookAt(target.transform);

SetWaypoint(target);

navigator.isStopped = true;

}

}

//Debug.Log(\_state + " is the state, with the player in vision: " + \_inVision + " also with the enemy being stopped: " + navigator.isStopped);

}

public void SetWaypoint(GameObject waypoint)

{

if (navigator == null)

{

//Debug.Log("NAVIGATOR IS NULL");

}

else

{

//Debug.Log("Set Waypoint to: " + waypoint.transform.position);

\_waypoint = waypoint;

//\_state = State.patroling;

if (navigator != null && navigator.isActiveAndEnabled)

{

navigator.SetDestination(\_waypoint.transform.position);

}

}

}

private void Patrol()

{

//Debug.Log("On patrol");

//Debug.Log(navigator==null);

if (navigator != null && navigator.isActiveAndEnabled)

{

//Debug.Log("SHIT WENT DOWN");

if (\_waypoint != null)

{

//Debug.Log("set destination to: "+\_waypoint.transform.position);

navigator.SetDestination(\_waypoint.transform.position);

gameObject.GetComponent<EnemyScript>().OnCheckpoint(\_waypoint, false);

}

}

//\_state = State.patroling;

}

private void SetTragetDestinationToPPosition(Vector3 pos)

{

//Debug.Log("setting target location to go to is = " + pos);

navigator.SetDestination(pos);

\_waypoint.transform.position = pos;

\_lastKnownTargetPosition = pos;

gameObject.GetComponent<EnemyScript>().SetDisturbedLocation(target.transform.position, false);

/\*\*

if (gameObject.GetComponent<Rigidbody>().velocity.magnitude < Speed)

{

moveDirection = target.transform.position - transform.position;

moveDirection = moveDirection.normalized;

moveDirection.Scale(\_speed);

GetComponent<Rigidbody>().AddForce(moveDirection);

}

/\*\*/

}

private void CheckVision()

{

Vector3 directionToTarget = transform.position - target.transform.position;

float angle = Vector3.Angle(transform.forward, directionToTarget);

if (Mathf.Abs(angle) > 90 && Mathf.Abs(angle) < 270)

{

\_inVision = true;

if (\_distanceToTarget < MeleeDistance)

{

StopMovement();

transform.LookAt(target.transform);

//Debug.Log("In melee range");

\_state = State.knife;

//Debug.Log("About to attack (Melee)");

return;

}

RaycastHit hit = new RaycastHit();

if (Physics.Linecast(transform.position, target.transform.position, out hit))

{

if (hit.transform.tag == "Player")

{

SetLastPositionToTarget();

\_state = State.shoot;

gameObject.GetComponent<EnemyScript>().SetDisturbedLocation(target.transform.position,true);

//navigator.SetDestination(transform.position);

//StopMovement();

}

else

{

\_inVision = false;

\_lookWait = 0;

navigator.isStopped = false;

\_state = State.none;

//Debug.Log("no vision");

}

}

else

{

//if (hit.transform.tag == "Player")

//{

if (hit.transform != null)

{

Debug.Log("not sure");

SetLastPositionToTarget();

\_state = State.shoot;

}

//gameObject.GetComponent<EnemyScript>().OnCheckpoint(gameObject, true);

//navigator.SetDestination(transform.position);

//StopMovement();

//}

//else

// Debug.Log("no vision");

}

}

}

private void StopMovement()

{

//\_state = State.knife;

//Debug.Log("stopping");

\_lastKnownTargetPosition = transform.position;

gameObject.GetComponent<EnemyScript>().SetDisturbedLocation(transform.position, false);

//navigator.isStopped = true;

//navigator.SetDestination(transform.position);

}

private void EnemyAttack()

{

//edit here

if (IsRanged || !\_inVision)

{

//Debug.Log("should stop");

//gameObject.GetComponent<Rigidbody>().constraints = \_stoppedConstraints;

//SetTragetDestinationToPPosition(transform.position);

//gameObject.GetComponent<EnemyScript>().OnCheckpoint(gameObject,true);

}

else

{

//Debug.Log("should move");

//gameObject.isStatic = false;

//gameObject.GetComponent<Rigidbody>().constraints = \_normalConstraints;

SetTragetDestinationToPPosition(\_lastKnownTargetPosition);

}

if (\_wait <= 0)

{

if (\_state == State.shoot && IsRanged)

{

if (target.GetComponent<CombatControls>().Health > 0)

{

Utils.ChangeGameObjectColor(gameObject, Color.red);

target.GetComponent<CombatControls>().DecreaseHealth(\_rangedDamage);

gameObject.GetComponent<AudioSource>().PlayOneShot(ShootSound, \_volume);

//Debug.Log("shoot shoot");

}

else

{

//Debug.Log("not shot");

}

}

if (\_state == State.knife && !IsRanged)

{

if (target.GetComponent<CombatControls>().Health > 0)

{

Utils.ChangeGameObjectColor(gameObject, Color.blue);

target.GetComponent<CombatControls>().DecreaseHealth(\_meleeDamage);

gameObject.GetComponent<AudioSource>().PlayOneShot(KnifeSound, \_volume);

//Debug.Log("Knify knify");

StopMovement();

}

else

{

//Debug.Log("not knife");

}

}

\_wait = Wait;

}

\_wait--;

}

public void SetLastPositionToTarget()

{

//Debug.Log("setting last pos to target");

\_lastKnownTargetPosition = target.transform.position;

\_tempWaypoint.transform.position = target.transform.position;

if (\_waypoint == null)

{

\_waypoint = \_tempWaypoint;

}

else

{

\_waypoint = \_tempWaypoint;

}

gameObject.GetComponent<EnemyScript>().SetDisturbedLocation(target.transform.position, false);

}

public void GiveTarget(GameObject ptarget)

{

//Debug.Log("Giving Target");

target = ptarget;

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyScript : MonoBehaviour

{

public int LookAngle;

public List<GameObject> Waypoints;

public float MinDistanceToWaypoint;

public bool StartOffset;

public bool ChooseRandomWaypoint;

public float AlertDistance;

public float WaitTimeAtWaypoint;

public GameObject Handler;

private GameObject \_currentWaypoint;

private float \_distanceToWaypoint;

private int \_disturbWait;

private int \_waypointIndex;

public GameObject DisturbWaypoint;

private bool \_onDisturbance;

[SerializeField] private int \_health;

private int \_left;

private int \_right;

private bool \_lookedLeft;

private bool \_lookedRight;

private bool \_notSetLooks;

public int Health

{

get { return \_health; }

}

public bool IsDead

{

get { return \_health == 0; }

}

// Use this for initialization

void Start()

{

//Debug.Log("first waypoint = "+Waypoints[0]);

if (Waypoints.Count <= 1)

StartOffset = true;

\_currentWaypoint = Waypoints[0];

\_waypointIndex = 0;

if (!StartOffset)

gameObject.GetComponent<EnemyMovement>().SetWaypoint(\_currentWaypoint);

if (Handler == null)

{

//Debug.Log("ERROR NO HANDLER");

}

else

{

Handler.GetComponent<EnemyHandler>().AddToHandler(gameObject);

}

}

public void OnCheckpoint(GameObject checkpoint, bool forPlayerSight)

{

if(\_onDisturbance)

{

//Debug.Log("on disturbance");

if (Look())

\_onDisturbance = false;

}

else

{

//Debug.Log("on checkpoint");

if (\_disturbWait >= WaitTimeAtWaypoint)

{

if (!forPlayerSight)

{

if (ChooseRandomWaypoint)

{

//Debug.Log("Next checkpoint random");

\_currentWaypoint = Waypoints[(int)Random.Range(0, Waypoints.Count)];

}

else

{

//Debug.Log("Next checkpoint");

\_currentWaypoint = Waypoints[(++\_waypointIndex) % Waypoints.Count];

}

}

gameObject.GetComponent<EnemyMovement>().SetWaypoint(\_currentWaypoint);

\_disturbWait = 0;

}

if (gameObject.GetComponent<Rigidbody>().velocity.magnitude <= 1)

\_disturbWait++;

}

}

public bool Look()

{

//Debug.Log("looking");

//\_left = gameObject.transform.localRotation;

//\_left.SetLookRotation(-gameObject.transform.right);

//\_right = gameObject.transform.localRotation;

//\_right.SetLookRotation(gameObject.transform.right);

if (!\_lookedLeft)

{

//Debug.Log("looking left");

gameObject.transform.Rotate(0, -1, 0);

\_left++;

//gameObject.transform.localRotation = Quaternion.Lerp(gameObject.transform.localRotation, \_left, LookSpeed);

if (\_left == LookAngle)

\_lookedLeft = true;

}

else if (!\_lookedRight)

{

//Debug.Log("looking right");

gameObject.transform.Rotate(0, 1, 0);

\_right++;

//gameObject.transform.localRotation = Quaternion.Lerp(gameObject.transform.localRotation, \_right, LookSpeed);

if (\_right == 2 \* LookAngle)

\_lookedRight = true;

}

else

{

\_notSetLooks = false;

\_lookedLeft = false;

\_lookedRight = false;

\_left = 0;

\_right = 0;

//Debug.Log("done looking");

return true;

}

return false;

}

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown(KeyCode.Z))

{

Debug.Log("My position is: " + gameObject.transform.position + " and I am heading to: " + \_currentWaypoint.transform.position);

}

//Debug.Log("Current waypoint = "+\_currentWaypoint);

\_distanceToWaypoint = (gameObject.transform.position - \_currentWaypoint.transform.position).magnitude;

//Debug.Log(\_distanceToWaypoint + " with the stop at " + (MinDistanceToWaypoint ));

if (\_distanceToWaypoint <= MinDistanceToWaypoint + 1.3f && !StartOffset)

{

//Debug.Log(\_distanceToWaypoint + " with the stop at " + (MinDistanceToWaypoint + 0.5f));

OnCheckpoint(\_currentWaypoint, false);

}

else

{

//Debug.Log("i am not on currentwaypoint");

}

}

public void SetDisturbedLocation(Vector3 pos, bool forEnemyDeath)

{

if(forEnemyDeath)

{

\_onDisturbance = true;

}

DisturbWaypoint.transform.position = pos;

\_currentWaypoint = DisturbWaypoint;

//\_waypointIndex--;

}

public void ResetDisturbWait()

{

\_disturbWait = 0;

}

private void OnDestroy()

{

if (Handler != null)

{

Handler.GetComponent<EnemyHandler>().AlertOthers(gameObject, AlertDistance);

}

}

public void DecreaseHealth(int pAmount)

{

if (\_health > 0)

{

\_health -= pAmount;

if (\_health < 0)

{

\_health = 0;

}

}

}

}