using System;

using UnityEngine;

using Random = UnityEngine.Random;

namespace UnityStandardAssets.Vehicles.Aeroplane

{

[RequireComponent(typeof (AeroplaneController))]

public class AeroplaneAiControl : MonoBehaviour

{

// This script represents an AI 'pilot' capable of flying the plane towards a designated target.

// It sends the equivalent of the inputs that a user would send to the Aeroplane controller.

[SerializeField] private float m\_RollSensitivity = .2f; // How sensitively the AI applies the roll controls

[SerializeField] private float m\_PitchSensitivity = .5f; // How sensitively the AI applies the pitch controls

[SerializeField] private float m\_LateralWanderDistance = 5; // The amount that the plane can wander by when heading for a target

[SerializeField] private float m\_LateralWanderSpeed = 0.11f; // The speed at which the plane will wander laterally

[SerializeField] private float m\_MaxClimbAngle = 45; // The maximum angle that the AI will attempt to make plane can climb at

[SerializeField] private float m\_MaxRollAngle = 45; // The maximum angle that the AI will attempt to u

[SerializeField] private float m\_SpeedEffect = 0.01f; // This increases the effect of the controls based on the plane's speed.

[SerializeField] private float m\_TakeoffHeight = 20; // the AI will fly straight and only pitch upwards until reaching this height

[SerializeField] private Transform m\_Target; // the target to fly towards

private AeroplaneController m\_AeroplaneController; // The aeroplane controller that is used to move the plane

private float m\_RandomPerlin; // Used for generating random point on perlin noise so that the plane will wander off path slightly

private bool m\_TakenOff; // Has the plane taken off yet

// setup script properties

private void Awake()

{

// get the reference to the aeroplane controller, so we can send move input to it and read its current state.

m\_AeroplaneController = GetComponent<AeroplaneController>();

// pick a random perlin starting point for lateral wandering

m\_RandomPerlin = Random.Range(0f, 100f);

}

// reset the object to sensible values

public void Reset()

{

m\_TakenOff = false;

}

// fixed update is called in time with the physics system update

private void FixedUpdate()

{

if (m\_Target != null)

{

// make the plane wander from the path, useful for making the AI seem more human, less robotic.

Vector3 targetPos = m\_Target.position +

transform.right\*

(Mathf.PerlinNoise(Time.time\*m\_LateralWanderSpeed, m\_RandomPerlin)\*2 - 1)\*

m\_LateralWanderDistance;

// adjust the yaw and pitch towards the target

Vector3 localTarget = transform.InverseTransformPoint(targetPos);

float targetAngleYaw = Mathf.Atan2(localTarget.x, localTarget.z);

float targetAnglePitch = -Mathf.Atan2(localTarget.y, localTarget.z);

// Set the target for the planes pitch, we check later that this has not passed the maximum threshold

targetAnglePitch = Mathf.Clamp(targetAnglePitch, -m\_MaxClimbAngle\*Mathf.Deg2Rad,

m\_MaxClimbAngle\*Mathf.Deg2Rad);

// calculate the difference between current pitch and desired pitch

float changePitch = targetAnglePitch - m\_AeroplaneController.PitchAngle;

// AI always applies gentle forward throttle

const float throttleInput = 0.5f;

// AI applies elevator control (pitch, rotation around x) to reach the target angle

float pitchInput = changePitch\*m\_PitchSensitivity;

// clamp the planes roll

float desiredRoll = Mathf.Clamp(targetAngleYaw, -m\_MaxRollAngle\*Mathf.Deg2Rad, m\_MaxRollAngle\*Mathf.Deg2Rad);

float yawInput = 0;

float rollInput = 0;

if (!m\_TakenOff)

{

// If the planes altitude is above m\_TakeoffHeight we class this as taken off

if (m\_AeroplaneController.Altitude > m\_TakeoffHeight)

{

m\_TakenOff = true;

}

}

else

{

// now we have taken off to a safe height, we can use the rudder and ailerons to yaw and roll

yawInput = targetAngleYaw;

rollInput = -(m\_AeroplaneController.RollAngle - desiredRoll)\*m\_RollSensitivity;

}

// adjust how fast the AI is changing the controls based on the speed. Faster speed = faster on the controls.

float currentSpeedEffect = 1 + (m\_AeroplaneController.ForwardSpeed\*m\_SpeedEffect);

rollInput \*= currentSpeedEffect;

pitchInput \*= currentSpeedEffect;

yawInput \*= currentSpeedEffect;

// pass the current input to the plane (false = because AI never uses air brakes!)

m\_AeroplaneController.Move(rollInput, pitchInput, yawInput, throttleInput, false);

}

else

{

// no target set, send zeroed input to the planeW

m\_AeroplaneController.Move(0, 0, 0, 0, false);

}

}

// allows other scripts to set the plane's target

public void SetTarget(Transform target)

{

m\_Target = target;

}

}

}