using System;

using UnityEngine;

using UnityStandardAssets.CrossPlatformInput;

namespace UnityStandardAssets.Vehicles.Aeroplane

{

[RequireComponent(typeof (AeroplaneController))]

public class AeroplaneUserControl2Axis : MonoBehaviour

{

// these max angles are only used on mobile, due to the way pitch and roll input are handled

public float maxRollAngle = 80;

public float maxPitchAngle = 80;

// reference to the aeroplane that we're controlling

private AeroplaneController m\_Aeroplane;

private void Awake()

{

// Set up the reference to the aeroplane controller.

m\_Aeroplane = GetComponent<AeroplaneController>();

}

private void FixedUpdate()

{

// Read input for the pitch, yaw, roll and throttle of the aeroplane.

float roll = CrossPlatformInputManager.GetAxis("Horizontal");

float pitch = CrossPlatformInputManager.GetAxis("Vertical");

bool airBrakes = CrossPlatformInputManager.GetButton("Fire1");

// auto throttle up, or down if braking.

float throttle = airBrakes ? -1 : 1;

#if MOBILE\_INPUT

AdjustInputForMobileControls(ref roll, ref pitch, ref throttle);

#endif

// Pass the input to the aeroplane

m\_Aeroplane.Move(roll, pitch, 0, throttle, airBrakes);

}

private void AdjustInputForMobileControls(ref float roll, ref float pitch, ref float throttle)

{

// because mobile tilt is used for roll and pitch, we help out by

// assuming that a centered level device means the user

// wants to fly straight and level!

// this means on mobile, the input represents the \*desired\* roll angle of the aeroplane,

// and the roll input is calculated to achieve that.

// whereas on non-mobile, the input directly controls the roll of the aeroplane.

float intendedRollAngle = roll\*maxRollAngle\*Mathf.Deg2Rad;

float intendedPitchAngle = pitch\*maxPitchAngle\*Mathf.Deg2Rad;

roll = Mathf.Clamp((intendedRollAngle - m\_Aeroplane.RollAngle), -1, 1);

pitch = Mathf.Clamp((intendedPitchAngle - m\_Aeroplane.PitchAngle), -1, 1);

// similarly, the throttle axis input is considered to be the desired absolute value, not a relative change to current throttle.

float intendedThrottle = throttle\*0.5f + 0.5f;

throttle = Mathf.Clamp(intendedThrottle - m\_Aeroplane.Throttle, -1, 1);

}

}

}