using System;

using UnityEngine;

#if UNITY\_EDITOR

using UnityEditor;

#endif

namespace UnityStandardAssets.CrossPlatformInput

{

// helps with managing tilt input on mobile devices

public class TiltInput : MonoBehaviour

{

// options for the various orientations

public enum AxisOptions

{

ForwardAxis,

SidewaysAxis,

}

[Serializable]

public class AxisMapping

{

public enum MappingType

{

NamedAxis,

MousePositionX,

MousePositionY,

MousePositionZ

};

public MappingType type;

public string axisName;

}

public AxisMapping mapping;

public AxisOptions tiltAroundAxis = AxisOptions.ForwardAxis;

public float fullTiltAngle = 25;

public float centreAngleOffset = 0;

private CrossPlatformInputManager.VirtualAxis m\_SteerAxis;

private void OnEnable()

{

if (mapping.type == AxisMapping.MappingType.NamedAxis)

{

m\_SteerAxis = new CrossPlatformInputManager.VirtualAxis(mapping.axisName);

CrossPlatformInputManager.RegisterVirtualAxis(m\_SteerAxis);

}

}

private void Update()

{

float angle = 0;

if (Input.acceleration != Vector3.zero)

{

switch (tiltAroundAxis)

{

case AxisOptions.ForwardAxis:

angle = Mathf.Atan2(Input.acceleration.x, -Input.acceleration.y)\*Mathf.Rad2Deg +

centreAngleOffset;

break;

case AxisOptions.SidewaysAxis:

angle = Mathf.Atan2(Input.acceleration.z, -Input.acceleration.y)\*Mathf.Rad2Deg +

centreAngleOffset;

break;

}

}

float axisValue = Mathf.InverseLerp(-fullTiltAngle, fullTiltAngle, angle)\*2 - 1;

switch (mapping.type)

{

case AxisMapping.MappingType.NamedAxis:

m\_SteerAxis.Update(axisValue);

break;

case AxisMapping.MappingType.MousePositionX:

CrossPlatformInputManager.SetVirtualMousePositionX(axisValue\*Screen.width);

break;

case AxisMapping.MappingType.MousePositionY:

CrossPlatformInputManager.SetVirtualMousePositionY(axisValue\*Screen.width);

break;

case AxisMapping.MappingType.MousePositionZ:

CrossPlatformInputManager.SetVirtualMousePositionZ(axisValue\*Screen.width);

break;

}

}

private void OnDisable()

{

m\_SteerAxis.Remove();

}

}

}

namespace UnityStandardAssets.CrossPlatformInput.Inspector

{

#if UNITY\_EDITOR

[CustomPropertyDrawer(typeof (TiltInput.AxisMapping))]

public class TiltInputAxisStylePropertyDrawer : PropertyDrawer

{

public override void OnGUI(Rect position, SerializedProperty property, GUIContent label)

{

EditorGUI.BeginProperty(position, label, property);

float x = position.x;

float y = position.y;

float inspectorWidth = position.width;

// Don't make child fields be indented

var indent = EditorGUI.indentLevel;

EditorGUI.indentLevel = 0;

var props = new[] {"type", "axisName"};

var widths = new[] {.4f, .6f};

if (property.FindPropertyRelative("type").enumValueIndex > 0)

{

// hide name if not a named axis

props = new[] {"type"};

widths = new[] {1f};

}

const float lineHeight = 18;

for (int n = 0; n < props.Length; ++n)

{

float w = widths[n]\*inspectorWidth;

// Calculate rects

Rect rect = new Rect(x, y, w, lineHeight);

x += w;

EditorGUI.PropertyField(rect, property.FindPropertyRelative(props[n]), GUIContent.none);

}

// Set indent back to what it was

EditorGUI.indentLevel = indent;

EditorGUI.EndProperty();

}

}

#endif

}