using System;

using UnityEngine;

using UnityEngine.EventSystems;

using UnityEngine.UI;

namespace UnityStandardAssets.CrossPlatformInput

{

[RequireComponent(typeof(Image))]

public class TouchPad : MonoBehaviour, IPointerDownHandler, IPointerUpHandler

{

// Options for which axes to use

public enum AxisOption

{

Both, // Use both

OnlyHorizontal, // Only horizontal

OnlyVertical // Only vertical

}

public enum ControlStyle

{

Absolute, // operates from teh center of the image

Relative, // operates from the center of the initial touch

Swipe, // swipe to touch touch no maintained center

}

public AxisOption axesToUse = AxisOption.Both; // The options for the axes that the still will use

public ControlStyle controlStyle = ControlStyle.Absolute; // control style to use

public string horizontalAxisName = "Horizontal"; // The name given to the horizontal axis for the cross platform input

public string verticalAxisName = "Vertical"; // The name given to the vertical axis for the cross platform input

public float Xsensitivity = 1f;

public float Ysensitivity = 1f;

Vector3 m\_StartPos;

Vector2 m\_PreviousDelta;

Vector3 m\_JoytickOutput;

bool m\_UseX; // Toggle for using the x axis

bool m\_UseY; // Toggle for using the Y axis

CrossPlatformInputManager.VirtualAxis m\_HorizontalVirtualAxis; // Reference to the joystick in the cross platform input

CrossPlatformInputManager.VirtualAxis m\_VerticalVirtualAxis; // Reference to the joystick in the cross platform input

bool m\_Dragging;

int m\_Id = -1;

Vector2 m\_PreviousTouchPos; // swipe style control touch

#if !UNITY\_EDITOR

private Vector3 m\_Center;

private Image m\_Image;

#else

Vector3 m\_PreviousMouse;

#endif

void OnEnable()

{

CreateVirtualAxes();

#if !UNITY\_EDITOR

m\_Image = GetComponent<Image>();

m\_Center = m\_Image.transform.position;

#endif

}

void CreateVirtualAxes()

{

// set axes to use

m\_UseX = (axesToUse == AxisOption.Both || axesToUse == AxisOption.OnlyHorizontal);

m\_UseY = (axesToUse == AxisOption.Both || axesToUse == AxisOption.OnlyVertical);

// create new axes based on axes to use

if (m\_UseX)

{

m\_HorizontalVirtualAxis = new CrossPlatformInputManager.VirtualAxis(horizontalAxisName);

CrossPlatformInputManager.RegisterVirtualAxis(m\_HorizontalVirtualAxis);

}

if (m\_UseY)

{

m\_VerticalVirtualAxis = new CrossPlatformInputManager.VirtualAxis(verticalAxisName);

CrossPlatformInputManager.RegisterVirtualAxis(m\_VerticalVirtualAxis);

}

}

void UpdateVirtualAxes(Vector3 value)

{

value = value.normalized;

if (m\_UseX)

{

m\_HorizontalVirtualAxis.Update(value.x);

}

if (m\_UseY)

{

m\_VerticalVirtualAxis.Update(value.y);

}

}

public void OnPointerDown(PointerEventData data)

{

m\_Dragging = true;

m\_Id = data.pointerId;

#if !UNITY\_EDITOR

if (controlStyle != ControlStyle.Absolute )

m\_Center = data.position;

#endif

}

void Update()

{

if (!m\_Dragging)

{

return;

}

if (Input.touchCount >= m\_Id + 1 && m\_Id != -1)

{

#if !UNITY\_EDITOR

if (controlStyle == ControlStyle.Swipe)

{

m\_Center = m\_PreviousTouchPos;

m\_PreviousTouchPos = Input.touches[m\_Id].position;

}

Vector2 pointerDelta = new Vector2(Input.touches[m\_Id].position.x - m\_Center.x , Input.touches[m\_Id].position.y - m\_Center.y).normalized;

pointerDelta.x \*= Xsensitivity;

pointerDelta.y \*= Ysensitivity;

#else

Vector2 pointerDelta;

pointerDelta.x = Input.mousePosition.x - m\_PreviousMouse.x;

pointerDelta.y = Input.mousePosition.y - m\_PreviousMouse.y;

m\_PreviousMouse = new Vector3(Input.mousePosition.x, Input.mousePosition.y, 0f);

#endif

UpdateVirtualAxes(new Vector3(pointerDelta.x, pointerDelta.y, 0));

}

}

public void OnPointerUp(PointerEventData data)

{

m\_Dragging = false;

m\_Id = -1;

UpdateVirtualAxes(Vector3.zero);

}

void OnDisable()

{

if (CrossPlatformInputManager.AxisExists(horizontalAxisName))

CrossPlatformInputManager.UnRegisterVirtualAxis(horizontalAxisName);

if (CrossPlatformInputManager.AxisExists(verticalAxisName))

CrossPlatformInputManager.UnRegisterVirtualAxis(verticalAxisName);

}

}

}