using System;

using System.Collections.Generic;

using UnityEngine;

namespace UnityStandardAssets.CrossPlatformInput

{

public abstract class VirtualInput

{

public Vector3 virtualMousePosition { get; private set; }

protected Dictionary<string, CrossPlatformInputManager.VirtualAxis> m\_VirtualAxes =

new Dictionary<string, CrossPlatformInputManager.VirtualAxis>();

// Dictionary to store the name relating to the virtual axes

protected Dictionary<string, CrossPlatformInputManager.VirtualButton> m\_VirtualButtons =

new Dictionary<string, CrossPlatformInputManager.VirtualButton>();

protected List<string> m\_AlwaysUseVirtual = new List<string>();

// list of the axis and button names that have been flagged to always use a virtual axis or button

public bool AxisExists(string name)

{

return m\_VirtualAxes.ContainsKey(name);

}

public bool ButtonExists(string name)

{

return m\_VirtualButtons.ContainsKey(name);

}

public void RegisterVirtualAxis(CrossPlatformInputManager.VirtualAxis axis)

{

// check if we already have an axis with that name and log and error if we do

if (m\_VirtualAxes.ContainsKey(axis.name))

{

Debug.LogError("There is already a virtual axis named " + axis.name + " registered.");

}

else

{

// add any new axes

m\_VirtualAxes.Add(axis.name, axis);

// if we dont want to match with the input manager setting then revert to always using virtual

if (!axis.matchWithInputManager)

{

m\_AlwaysUseVirtual.Add(axis.name);

}

}

}

public void RegisterVirtualButton(CrossPlatformInputManager.VirtualButton button)

{

// check if already have a buttin with that name and log an error if we do

if (m\_VirtualButtons.ContainsKey(button.name))

{

Debug.LogError("There is already a virtual button named " + button.name + " registered.");

}

else

{

// add any new buttons

m\_VirtualButtons.Add(button.name, button);

// if we dont want to match to the input manager then always use a virtual axis

if (!button.matchWithInputManager)

{

m\_AlwaysUseVirtual.Add(button.name);

}

}

}

public void UnRegisterVirtualAxis(string name)

{

// if we have an axis with that name then remove it from our dictionary of registered axes

if (m\_VirtualAxes.ContainsKey(name))

{

m\_VirtualAxes.Remove(name);

}

}

public void UnRegisterVirtualButton(string name)

{

// if we have a button with this name then remove it from our dictionary of registered buttons

if (m\_VirtualButtons.ContainsKey(name))

{

m\_VirtualButtons.Remove(name);

}

}

// returns a reference to a named virtual axis if it exists otherwise null

public CrossPlatformInputManager.VirtualAxis VirtualAxisReference(string name)

{

return m\_VirtualAxes.ContainsKey(name) ? m\_VirtualAxes[name] : null;

}

public void SetVirtualMousePositionX(float f)

{

virtualMousePosition = new Vector3(f, virtualMousePosition.y, virtualMousePosition.z);

}

public void SetVirtualMousePositionY(float f)

{

virtualMousePosition = new Vector3(virtualMousePosition.x, f, virtualMousePosition.z);

}

public void SetVirtualMousePositionZ(float f)

{

virtualMousePosition = new Vector3(virtualMousePosition.x, virtualMousePosition.y, f);

}

public abstract float GetAxis(string name, bool raw);

public abstract bool GetButton(string name);

public abstract bool GetButtonDown(string name);

public abstract bool GetButtonUp(string name);

public abstract void SetButtonDown(string name);

public abstract void SetButtonUp(string name);

public abstract void SetAxisPositive(string name);

public abstract void SetAxisNegative(string name);

public abstract void SetAxisZero(string name);

public abstract void SetAxis(string name, float value);

public abstract Vector3 MousePosition();

}

}