```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;

namespace UnityStandardAssets.Characters.ThirdPerson
{
    [RequireComponent(typeof (ThirdPersonCharacter))]
    public class ThirdPersonUserControl : MonoBehaviour
    {
        private ThirdPersonCharacter m_Character; // A reference to the ThirdPersonCharacter on the object
        private Transform m_Cam;                  // A reference to the main camera in the scenes transform
        private Vector3 m_CamForward;             // The current forward direction of the camera
        private Vector3 m_Move;
        private bool m_Jump;                      // the world-relative desired move direction, calculated from the camForward and user input.


        private void Start()
        {
            // get the transform of the main camera
            if (Camera.main ≠ null)
            {
                m_Cam = Camera.main.transform;
            }
            else
            {
                Debug.LogWarning(
                    "Warning: no main camera found. Third person character needs a Camera tagged \"MainCamera\", for camera-relative controls.", gameObject);
                // we use self-relative controls in this case, which probably isn't what the user wants, but hey, we warned them!
            }

            // get the third person character ( this should never be null due to require component )
            m_Character = GetComponent<ThirdPersonCharacter>();
        }


        private void Update()
        {
            if (!m_Jump)
            {
                m_Jump = CrossPlatformInputManager.GetButtonDown("Jump");
            }
        }


        // Fixed update is called in sync with physics
        private void FixedUpdate()
        {
            // read inputs
            float h = CrossPlatformInputManager.GetAxis("Horizontal");
            float v = CrossPlatformInputManager.GetAxis("Vertical");
            bool crouch = Input.GetKey(KeyCode.C);

```

```
            // calculate move direction to pass to character
            if (m_Cam != null)
            {
                // calculate camera relative direction to move:
                m_CamForward = Vector3.Scale(m_Cam.forward, new Vector3(1, 0, 1)).normalized;
                m_Move = v*m_CamForward + h*m_Cam.right;
            }
            else
            {
                // we use world-relative directions in the case of no main camera
                m_Move = v*Vector3.forward + h*Vector3.right;
            }
#if !MOBILE_INPUT
      // walk speed multiplier
           if (Input.GetKey(KeyCode.LeftShift)) m_Move *= 0.5f;
#endif
•
            // pass all parameters to the character control script
            m_Character.Move(m_Move, crouch, m_Jump);
            m_Jump = false;
        }
    }
}
```