Importing the CrossPlatformInput package adds a menu item to Unity, "CrossPlatformInput", which allows you to enable or disable the CrossPlatformInput in the editor. You must enable the CrossPlatformInput in order to see the control rigs in the editor, and to start using Unity Remote to control your game.

The CrossPlatformInput sample assets contains two main sections.

1) The folder of prefabs provide a variety of ready-to-use "MobileControlRigs". Each control rig is suitable for a different purpose, and each implements the touch or tilt-based equivalent of some of the default standalone axes or buttons. These are ready to drop into your scene, and to use them you simply need to read the axes via the CrossPlatformInput class, rather than Unity's regular Input class.

2) The set of scripts provided are the scripts we used to put together the control rigs prefabs. They provide a simplified way of reading basic mobile input, such as tilt, taps and swipe gestures. They are designed so that various mobile controls can be read in the same way as regular Unity axes and buttons. You can use these scripts to build your own MobileControlRigs.

For example the Car control rig feeds the tilt input of the mobile device to the "Horizontal" axis, and has an accelerator and brake touch button which are fed as a pair into the "Vertical" axis. These are virtual equivalents of the real "Horizontal" and "Vertical" axes defined in Unity's Input Manager.

Therefore when you read CrossPlatformInput.GetAxis("Horizontal"), you will either get the "real" input value - if your build target is non-mobile, or the value from the mobile control rig - if your build target is set to a mobile platform.

The CrossPlatformInput scripts and prefabs are provided together as an example of how you can implement a cross-platform control solution in Unity. They also allow us to provide our other sample scenes in a form that can be published as standalone or to mobile targets with no modification.

To use the CrossPlatformInput, you need to drop a "Mobile Control Rig" into your scene (or create your own), and then make calls to CrossPlatformInput functions, referring to the axes and buttons that the Rig implements.

When reading input from the CrossPlatformInput class, the values returned will be taken either from Unity's Input Manager settings, or from the mobile-specific controls set up, depending on which build target you have selected.

The CrossPlatformInput class is designed to be called instead of Unity's own Input class, and so mirrors certain parts of the Input API - specifically the functions relating to Axes and Buttons:

GetAxis, GetAxisRaw

GetButton, GetButtonDown, GetButtonUp

Notes for coders:

This package sets two compiler define symbols. One is always set automatically, the other is optionally set from a menu item.

Importing the "CrossPlatformInput" package will automatically add a compiler define symbol, "CROSS\_PLATFORM\_INPUT". This enables the CrossPlatformInput functions defined in some of the other Sample Asset packages (such as the Characters, Planes, etc). Without this symbol defined, those packages use Unity's regular Input class, which means they can be imported alone and still work without the CrossPlatformInput package.

The optional define (which is set by default, but can be disabled using the "Mobile Input" menu), is "MOBILE\_INPUT". This causes the MobileControlRigs to become active when a mobile build target is selected. It also enables certain mobile-specific control nuances in some of the packages, which make more sense when the character or vehicle is being controlled using mobile input (such as auto-leveling the character's look direction). This define is optional because some developers prefer to use standalone input methods instead of the Unity Remote app, when testing mobile apps in the editor's play mode.