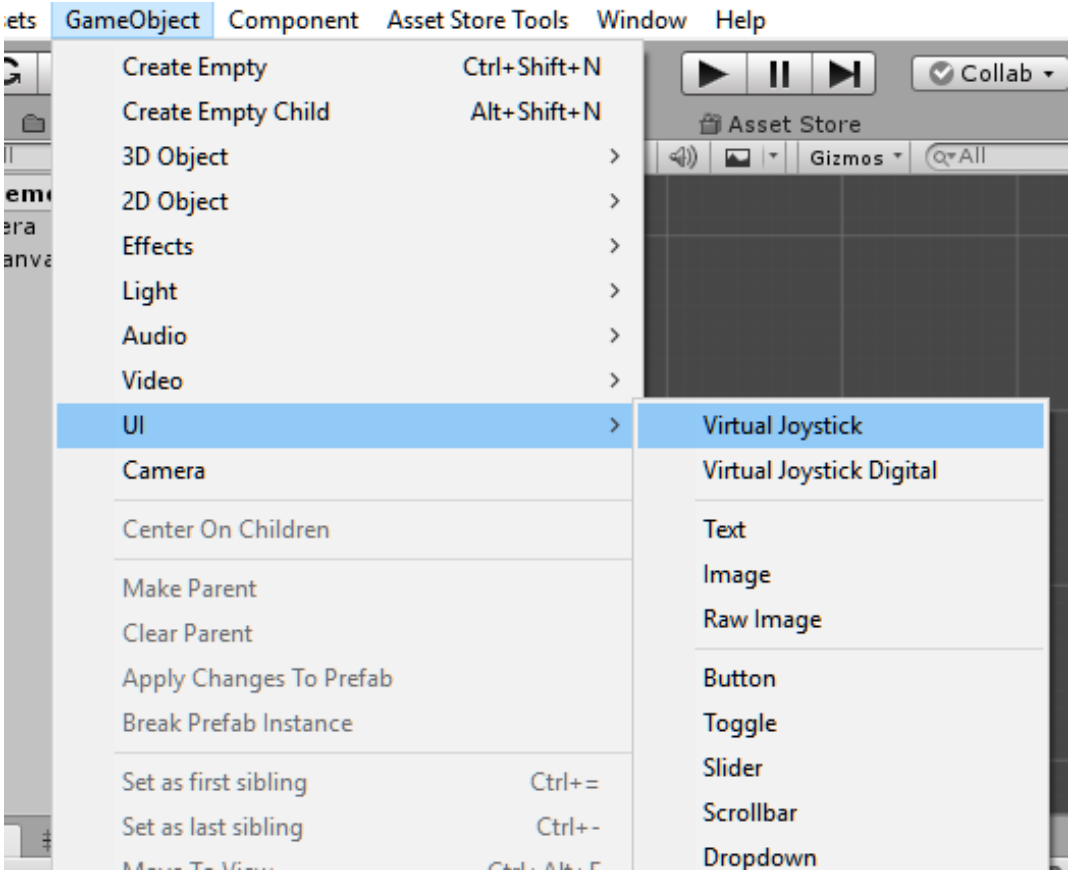
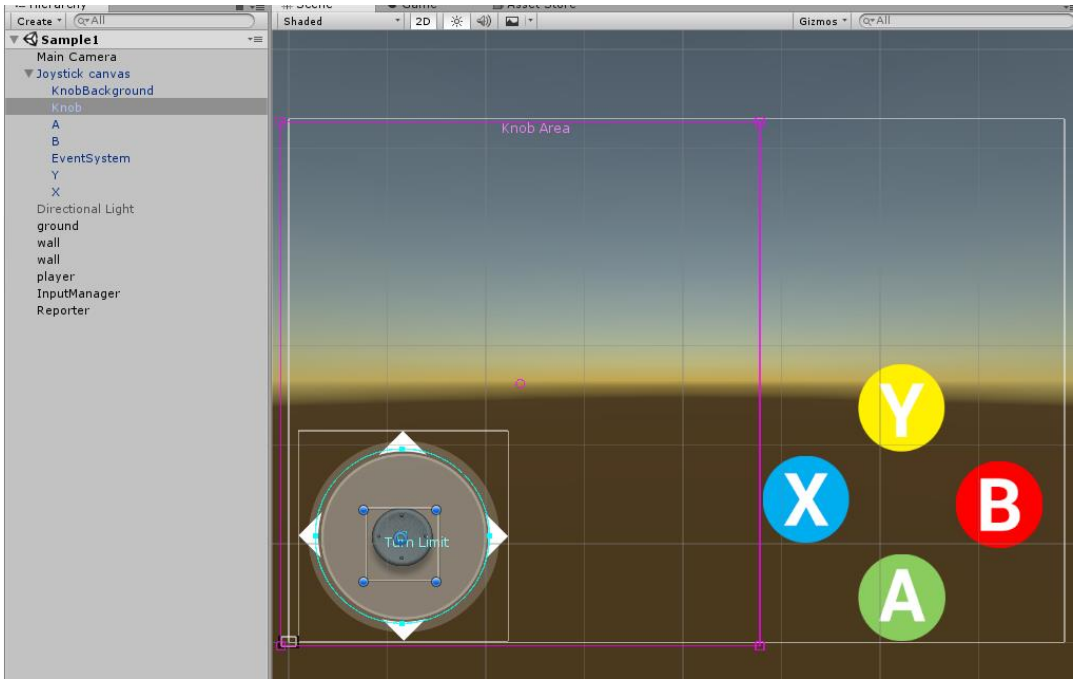
Plug And Play Joystick – Quick Start

1 - Add Joystick by going to the menu GameObject>UI>Virtual Joystick and clicking UI Joystick.

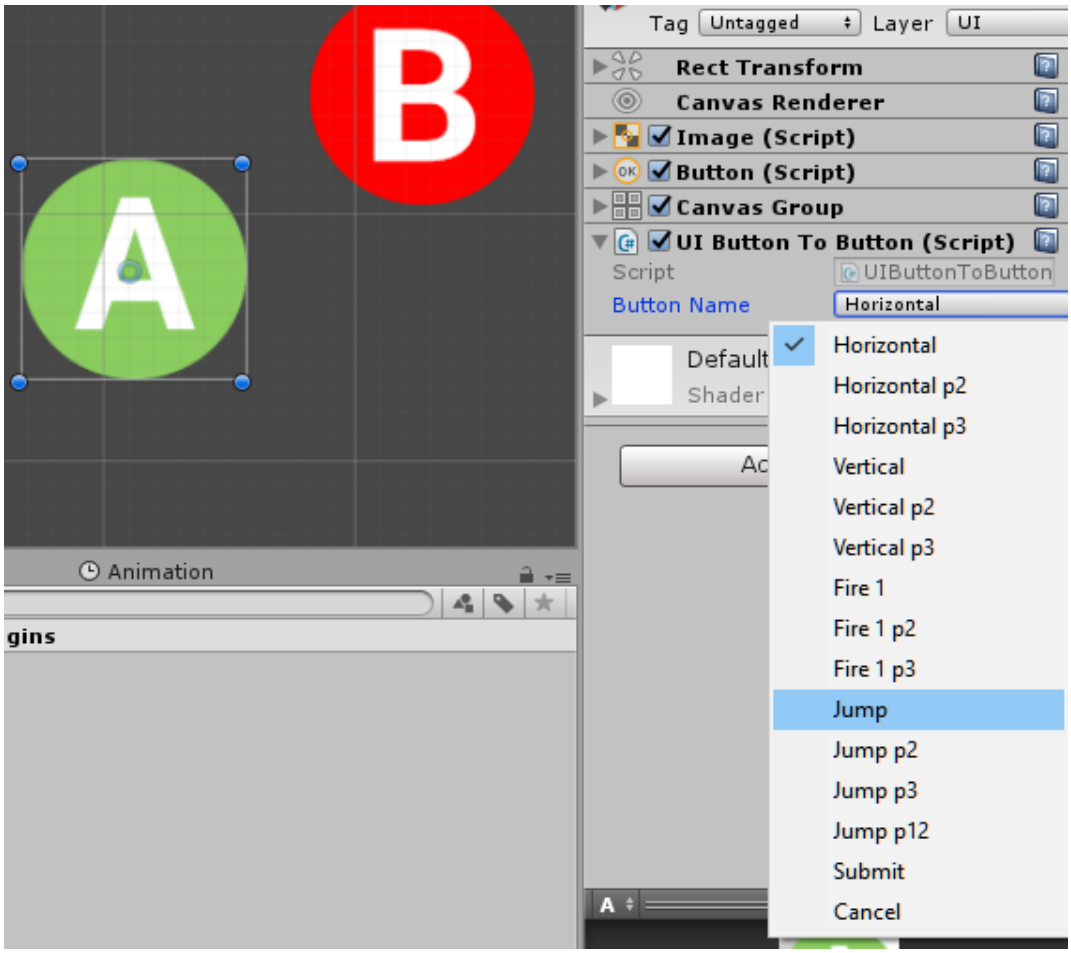


2 - A canvas joystick will be added in the scene, open it and note that there are some buttons.

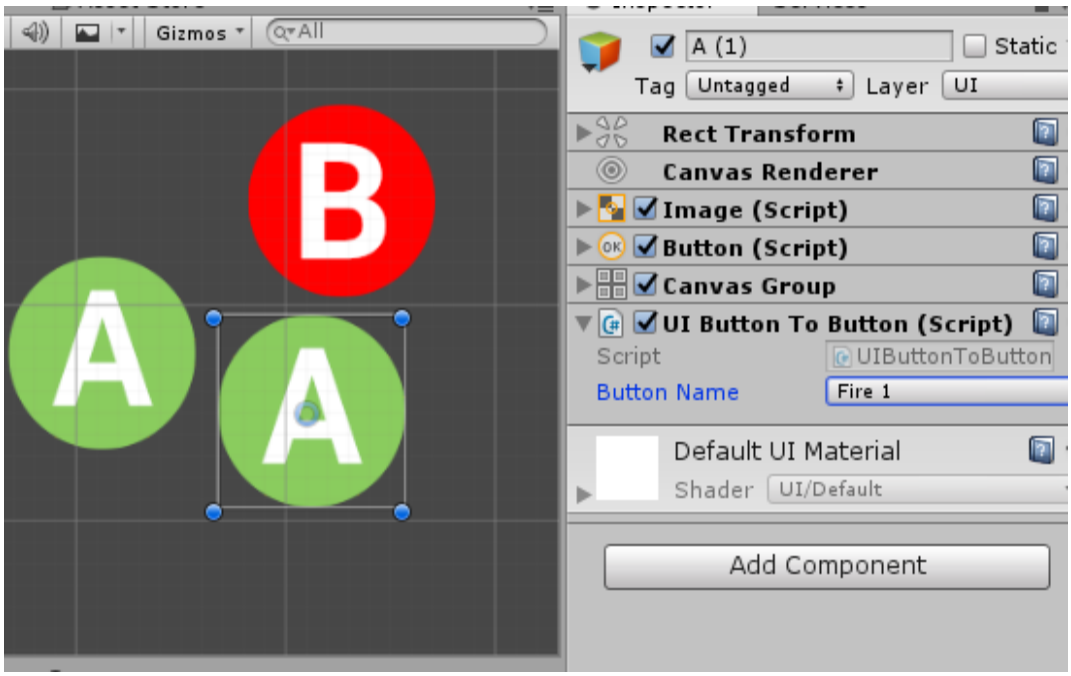


3 - Select the A or B button, and notice that there is a component called "UI Button to Button", in it you choose which standard button you want the button to press. If your game uses standard axes (Ex: Input.GetButtonDown ("Jump")), this

component will interfere and will detect this button being pressed as if it were on the keyboard or physical joystick.

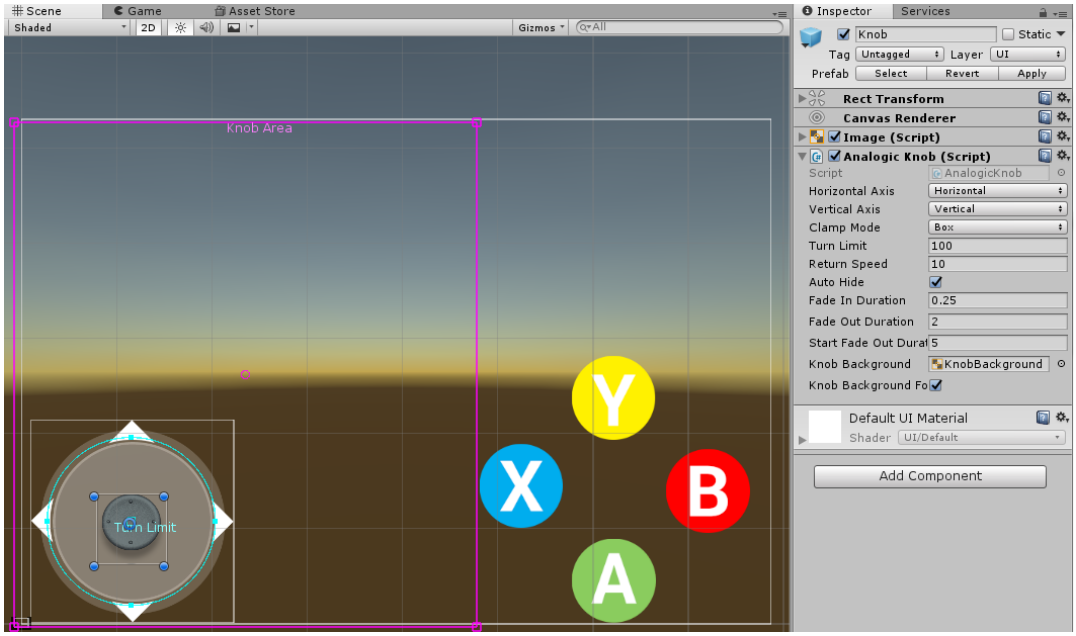


4 - You can duplicate any button, change the standard axes it matches, and swap the image without further effort. There are no limits to the number of buttons on the screen.

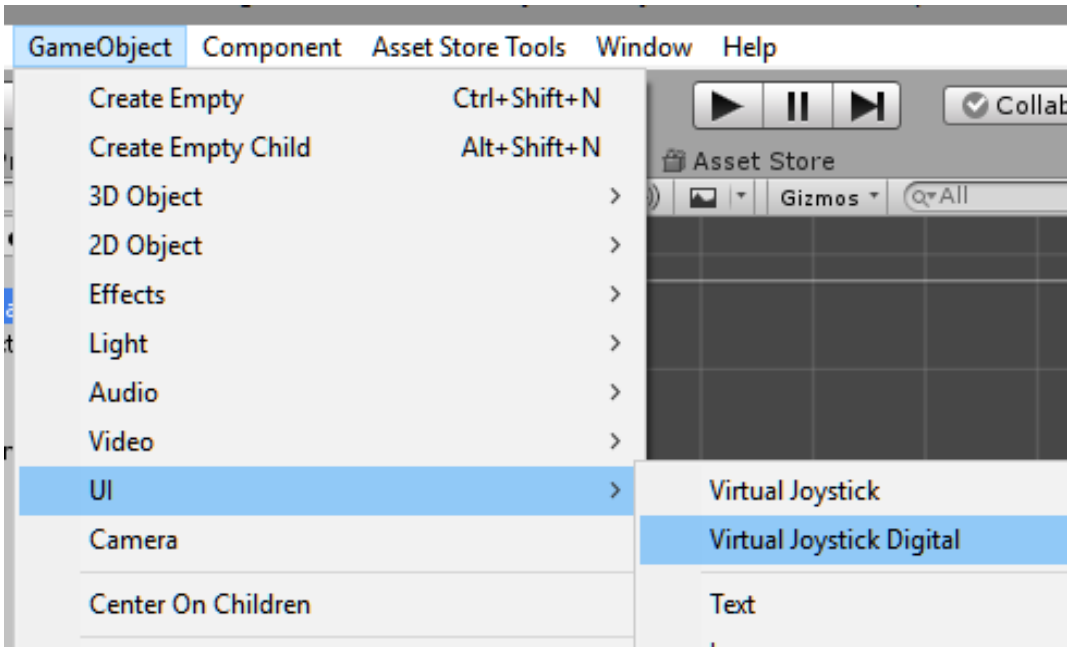


5 - Select "Joystick canvas" and note the "Analogic Knob" component. This component interferes with the directional Standard Axes, usually "Horizontal" and "Vertical", but you can modify it. This component interferes with checking

"Input.GetAxis (" horizontal ")". This way this will be detected by your game's scripts as if you were pressing the analog joystick directional or the arrow keys on the keyboard. The magenta area is the boundary where the knob can be dragged. There are nodes to you to resize it. By default this is ready for use, since almost all games will use the standard "Vertical" and "Horizontal". Drawings for the knob can be changed on the "Knob" and "KnobBackground" objects.

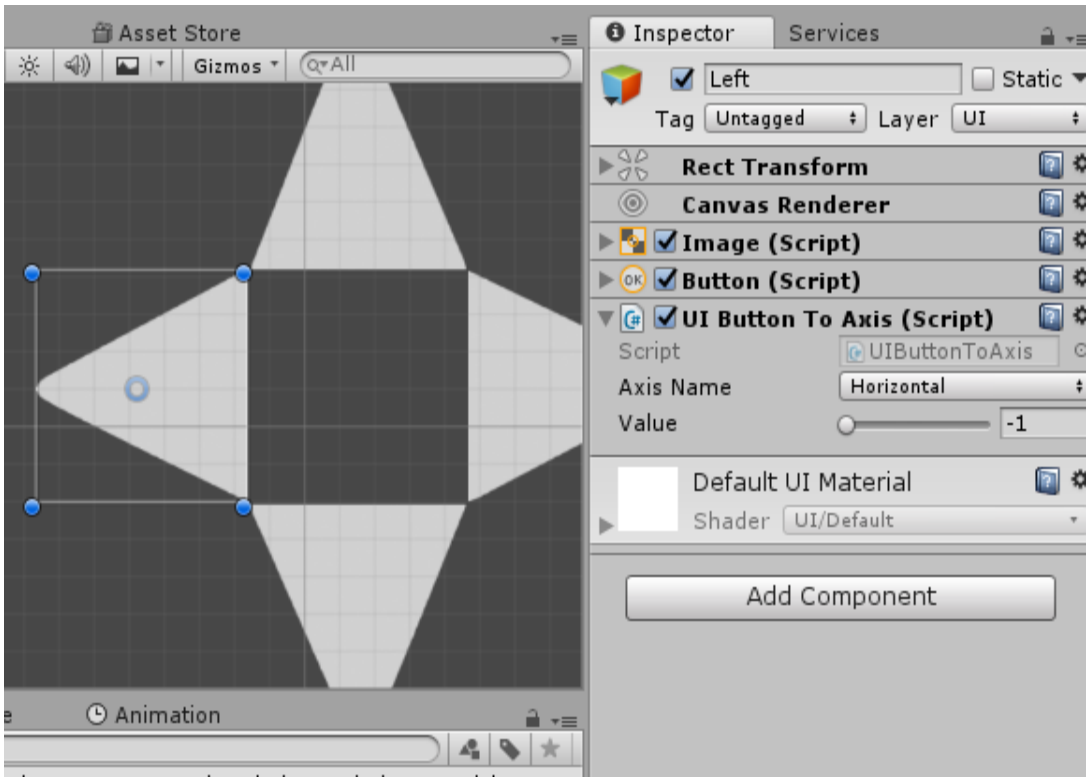


6 - There is a second type of joystick, similar, but with digital directional. You can add it in the menu GameObject>UI>Virtual Joystick Digital.



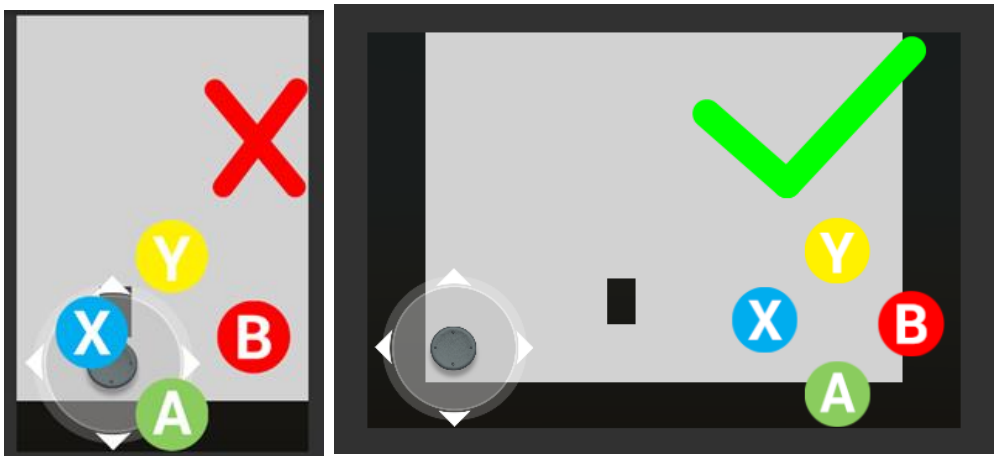
7 - The digital directional buttons have the UIButtonToAxis component that work similarly to the UIButtonToButton component, but this time they work with Input.GetAxis ("Direction") and you can specify the value that you want this button to match on the axis. For example, for the left arrow, I use Axis Name "Horizontal"

with a value of -1. You can delete some directions, no problems, there will be no errors.



8 - If your game uses "Input.GetButton()", "Input.GetButtonDown()", "Input.GetButtonUp()", "Input.GetAxis()" it is already perfectly matched to the virtual joystick, you do not need to do more nothing because the plugin pretends to be keyboard and joystick.

9 – IMPORTANT: **Lock the device orientation to make sure the buttons do not collide on the screen. If you want to use in portrait mode, set up the UI anchors in portrait mode. If you want to use it in landscape mode, set the anchors of the UI in landscape mode. If you do not lock the orientation, the user will come across the layout of the messy buttons.**



10 - Enjoy. Any questions you can ask me at leoluzprog@gmail.com

Note: Detections occur in FixedUpdate and Update. If you like to use input in fixed update to work with physics and avoid 1 frame of delay in physics, no problem.