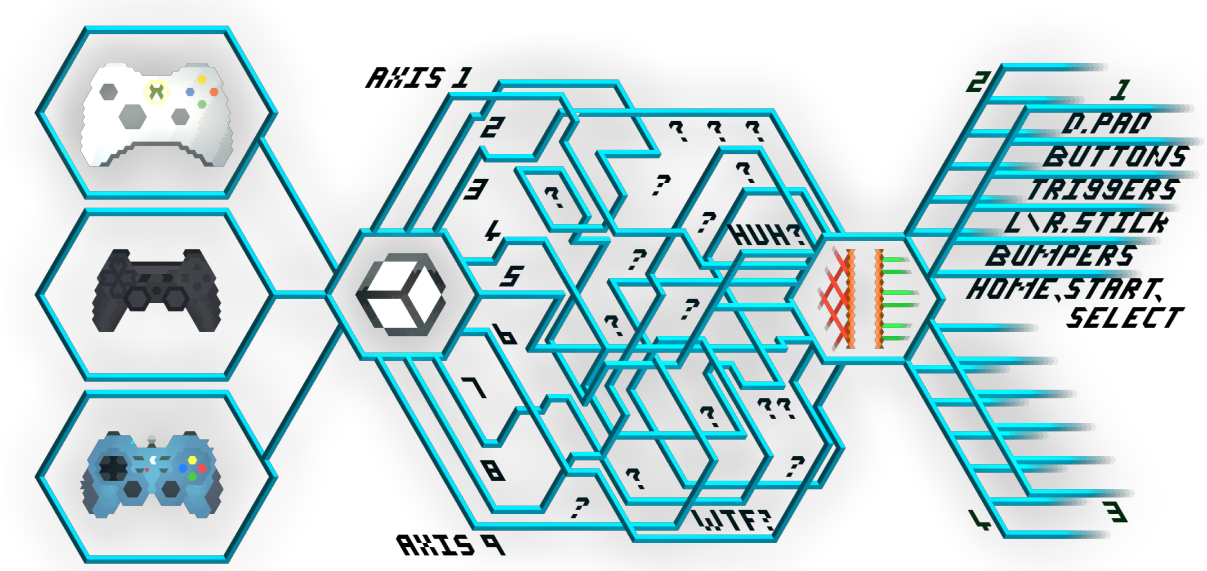
Cross-Platform Cross-Controller
Gamepad Input Manager



Open Beta v0.1

http://raxterworks.blogspot.com/
raxterworks@gmail.com

What is the problem?

Although Unity exposes 20 buttons and 10 axes per input device (a controller or gamepad), the configuration of which axis or button relates to what actual button on the device is never clear without experimentation. To confuse this matter not all platforms use the same drivers, even with the same controller. For instance the XBOX controller has a unique configuration each for PC, Mac, and Linux.

What is the solution?

This plugin aims to abstract all of the mess and expose a neat, easy to understand and easy to use interface. The most popular controllers come with configurations already available (XBOX, PS, and Logitech/Generic) for PC, Mac, and Linux. With *NO* DLLs or external libraries, only C# scripts!

What if I have my own controller

Included with the plugin is are configuration helpers (with a usable example) that allows the developer to configure any device they plug in. This will be added to a ScriptableObject that will be built into your game when you deploy.

What do I need to do to set up?

If you just want the standard XBOX, PS, Generic controllers supported you just need to follow a few simple clicks to setup the project's input axes (see below). If you are concerned about having to set up 20+ axes manually in your input controller, do not worry! The plugin comes with a tool that will automatically set it up for you in the click of a button. Already have inputs axes setup that you don't want overwritten? If you have a pro licence the plugin can append to the settings, leaving your painstakingly set up axes as you left them!

What do I do if it doesn't work properly?

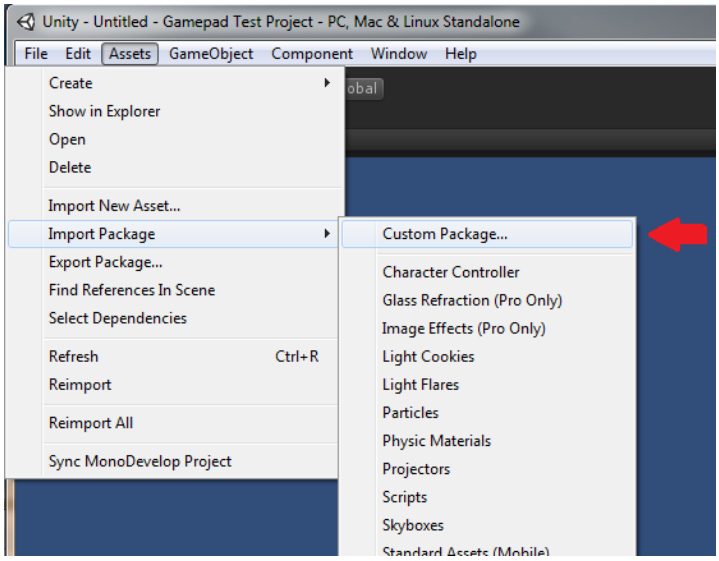
If something breaks or you have a suggestion or feature request, email raxterworks@gmail.com or leave a comment at

Who made this? How can I give you money? (:D)

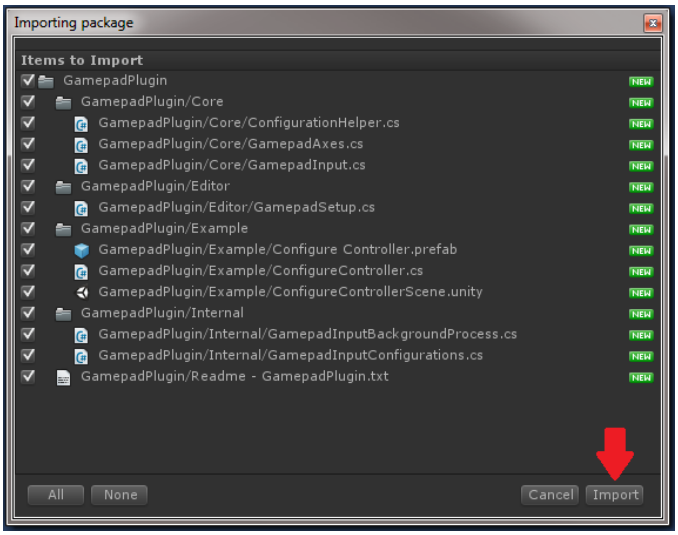
This plugin is made by one developer in his spare time, it is in open beta right now, so bug reports, suggested features, or any advice or critisicm are greatly appriciated.If you found this plugin useful at all and wish to show your support, visit
http://raxterworks.blogspot.com/p/cross-platform-cross-controller-gamepad.html and donate! With enough support I will be able to support more controllers, more platforms (namely iOS and Android), and add more features!

Enjoy!

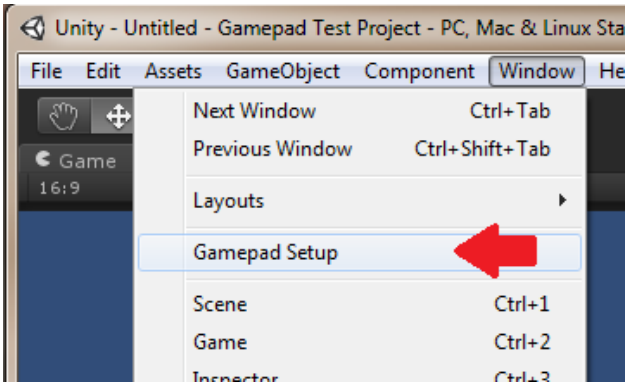
First Time Setup



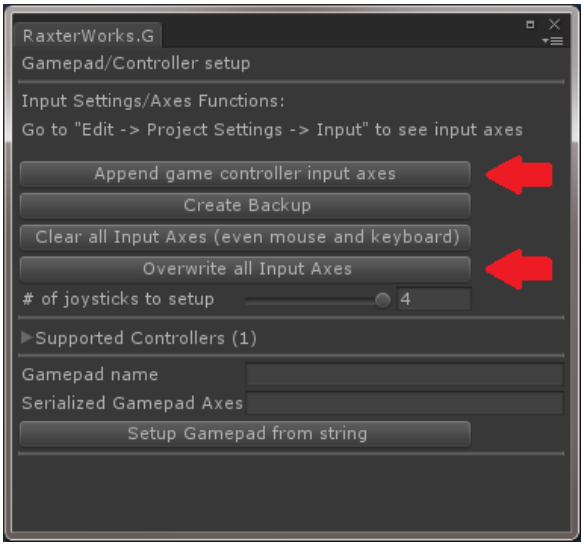
1. Open your project and drag “GamepadInputManager_v#.#.unityplugin” into your “Project” window or go to Assets -> Import Package -> Custom Package… , navigate to GamepadInputManager_v#.#.unityplugin.



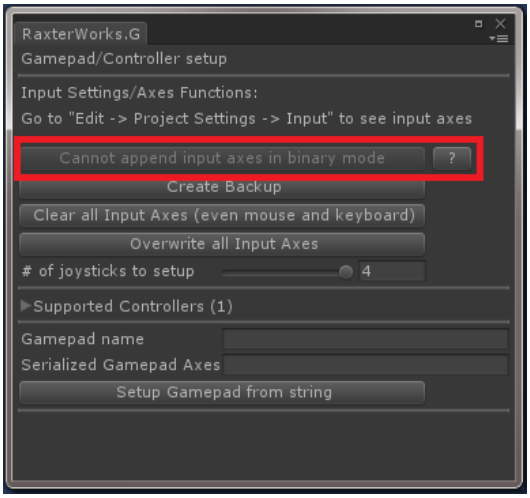
2. Click “Import” to extract the files to your project



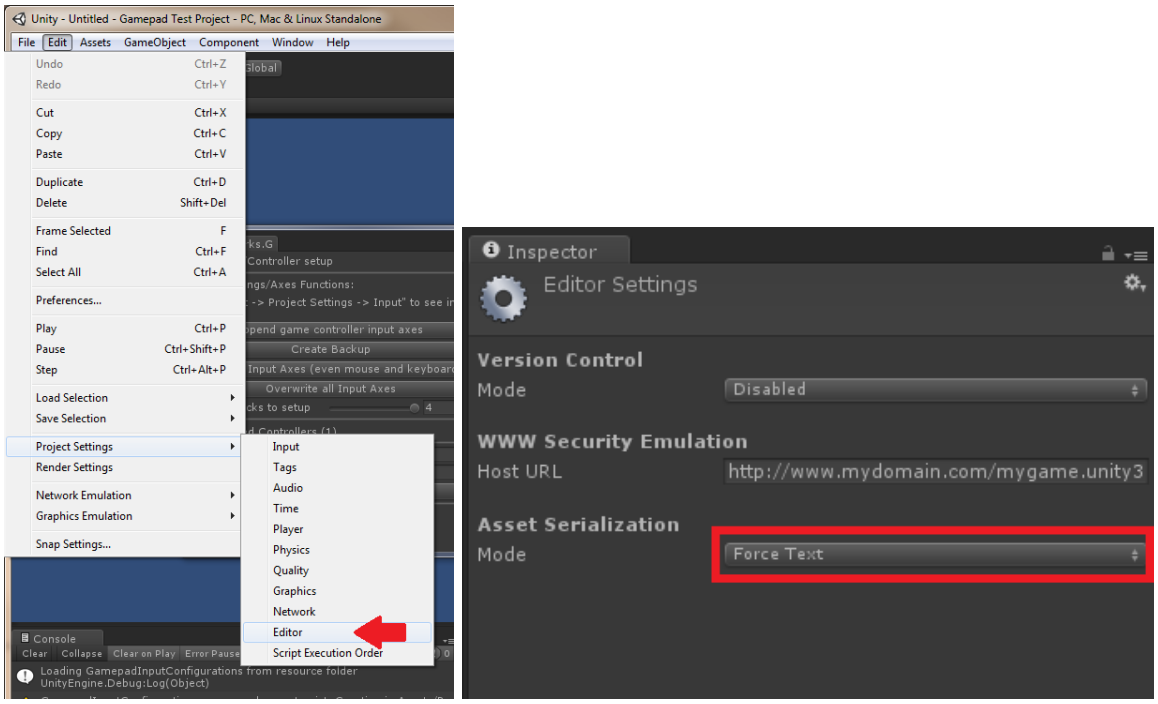
3. Go to Window -> Gamepad Setup to open up the editor



4. Overwrite or append the Input Axes
    a. If you want to overwrite the input settings, click “Overwrite all input Axes”
    b. If you want to append input settings click “Append game controller input axes”



5. * Optional* If you have the free version you won’t be able to append input axes (it will be greyed out). To enable this option the Pro version of Unity is required. If you don’t require the append functionality, ignore this step



a. Open the Editor Setting by going to Edit -> Project Settings -> Editor Settings
b. Ensure that “Asset Serialization” is set to “Force Text”
c. Go back to step 4

Example Usage

Documentation is usually boring, so here are a whole smash of examples (rigorous documentation below)

```
//Gets whether the left bumper has been released
GamepadInput.GetButtonUp(Button.LeftBumper);

//Was the dpad's up-direction pressed
GamepadInput.GetButtonDown(Button.DPadUp);

//Is the X button held down (i.e. the lower action button)
GamepadInput.GetButton(XBOXButton.X);
GamepadInput.GetButton(Button.ActionDown); // This is the same command

//Is Controller 2's Circle button held down (i.e. the right action button)
GamepadInput.GetButton(PSButton.Circle, 2));
GamepadInput.GetButton(Button.ActionRight, 2));

//Is the left action button held down (X for XBOX, Square for PS, U for OUYA)
GamepadInput.GetButton(Button.ActionLeft));

//Is the start button was pressed
GamepadInput.GetButtonDown(Button.Start));

//Gets the left analog stick's horizontal value between -1 and 1
GamepadInput.GetAxis(Axis.LeftAnalogX);

//Gets the right trigger's pressed value between 0 and 1
GamepadInput.GetAxis(Axis.RightTrigger);

//Gets the right analog's XY values between (-1,-1) and (1,1)
GamepadInput.GetXYAxis(XYAxis.RightAnalogX);

//Gets the left analogs polar coordinates (angle, magnitude)
GamepadInput.GetPolarAxis(XYAxis.LeftAnalog));

//Gets the DPad as a Vector2
GamepadInput.GetXYAxis(XYAxis.DPad));
```

API Specification

```
// Is a button currently held down
bool        GetButton (Button button);
bool        GetButton (Button button, int controllerIndex);

// Was a button pressed down
bool        GetButtonDown (Button button);
bool        GetButtonDown (Button button, int controllerIndex);

// Was a button released
bool        GetButtonUp (Button button);
bool        GetButtonUp (Button button, int controllerIndex);

// List of Buttons
enum Button {
      ActionDown, ActionUp, ActionLeft, ActionRight,
      LeftBumper, RightBumper,
      LeftAnalog, RightAnalog,
      Start, Select, Home,
      DPadUp, DPadDown, DPadLeft, DPadRight,
      LeftTrigger, RightTrigger,
};
```

```
// Gets a 2D vector representing a D-Pad or Analog Stick ***
Vector2     GetXYAxis (XYAxis axis);
Vector2     GetXYAxis (XYAxis axis, int controllerIndex);

// List of 2D
enum XYAxis {
      LeftAnalog,
      RightAnalog,
      DPad,
      ActionButtons // ***
};

/*** XYAxis.ActionButtons simulates a 2D axis using the four action buttons
(A, B, X, Y). This allows one to use the action buttons as a 'D-Pad' which can
be used for left hand modes if so desired
*/
```

```
// Gets a float representing an axis
float       GetAxis (Axis axis);
float       GetAxis (Axis axis, int controllerIndex);

enum Axis {
      LeftAnalogX, LeftAnalogY,
      RightAnalogX, RightAnalogY,
      DPadX, DPadY,
      ActionButtonsX, ActionButtonsY,
      LeftTrigger, RightTrigger,
};

/* LeftTrigger and RightTrigger are the two most useful of this enum, the rest
are the individual components of the 2D XYAxis values
*/
```

```
/* ----------------------
   BUTTON AND AXIS OVERLAP
   ----------------------

You will have noticed that the Left/Right Triggers, D-Pad, and Action Buttons
can each be accessed as either a Button or an XYAxis (or an Axis)

This is because some controllers recognise some of the inputs as buttons, but
on another controller, the same input is recognised as an Axis. For this
reason both are supported.

If the specific controller uses Buttons for, say, the D-Pad, then the D-Pad
Axis is simulated. If the controller recognises the D-Pad as an Axis, the
Button presses are simulated. Similarly the Triggers also can be accessed as
either a Button or an Axis.

*/
```
