InputPlus

5/4/2014
Message:
Thanks for purchasing InputPlus. I want to do what I can to bring some sort of standardization to gamepads on computer systems. Almost every gamepad I've tried out works in its own slightly unique way and a uniform way to view all these gamepads in code would be a great help. I also want games on computers to be easier to play and I hope this helps.

While I'm confident I've covered a majority of controller behaviors, considering how many controllers performed in unique manners I suspect there will be controllers that act in ways I didn't quite think of. Please understand that without a huge collection of gamepads, I can't promise every single one will work. Please tell me about any problems you have so I can get them fixed for everyone.

Contact:
inputplusforunity@gmail.com

Quick Run-down:
What is InputPlus:

- Allows players to teach the game about their gamepad controller
- By knowing about the player's gamepad, universal gamepad support is obtained*
- Simplifies game-to-player communication about the players' controllers

What InputPlus isn't:

- A pre-built user interface with rebinding of controls (Although InputPlus will allow you to build that type of interface)

Features:

- Supports the unity maximum of controllers, currently 11.
- Provides a learning sequence through the whole gamepad to teach your game about the controller
- Also provides a single control learning ability for you to completely customize controller learning in your game
- Also works as a controller diagnostic tool, display raw input values from all your controllers!
- Pixel based images to guide players through controller setup.

What is included:

- InputPlus script
- GuiExample - A simple GUI used to set up controller programming and diagnostics output
- ControlExample - A simple example of InputPlus in use
- Pixel drawn controller images used by the examples

The ideal support for gamepads on the PC would be for the user to be able to plug one in an just have it work exactly as expected, like you can experience on console system, not only at a hardware level, but in functionality as well. Unfortunately it seems unlikely that this can be possible any time soon, however we can work to make this process as smoothly as possible. Since there doesn't seem to be much standardization in the way gamepads work on PC, it makes the task a little difficult.

Let's consider the buttons for a moment. Some have labels that are letters, numbers, words, and some even just have symbols, but to the OS these buttons are assigned numbers (and they might not even be the same numbers you see written on the button!). As computer users we trade off some consistencies for flexibility and

customizability, but we want to make games as easy as possible to play. So this solution creates support to resemble the modern console controller.

Imagine playing a game, you just found a new jetpack item, and on screen text says “press button 7 to activate the jetpack!” Right there you’re pulled out of your game! Where’s button 7?! Do you pause the game to try to find out through the interface which button might be 7, it might just say “Use Jetpack: Button 7” and that does you no good. Maybe you could remap your controls so you know where the Use Jetpack button will be. Another option is you could just mash some buttons hoping to find it, but maybe you’ll accidentally use that medkit you picked up a few minutes before, that you were planning on saving for the boss. You can see it’s quite confusing. For some, learning to play a game can be challenging enough, having to fumble with this kind of control might cause them to just quit your game.

So this is where InputPlus comes in. With InputPlus the user teaches the game about their controller(s) and as a developer you refer to the controller in terms common to modern consoles. This way, if you want to display some information on how the player can use their new jetpack, you can display an image of a standard controller with the button highlighted. Now the user knows exactly what to do! Not only that, but you can create an interface that shows their bindings with practical names such as ‘dpad up’ or ‘right top shoulder button’. Now it’s easy to teach someone how to play your game.

Important Info + notes:
As of writing this, for windows 7 (and possibly others) Unity itself isn’t supporting connecting/reconnecting controllers in stand-alone builds. What this means is the player starts with controllers already connected, and if a controller becomes disconnected without restarting the game, they can’t use it anymore. As unsettling as this is, it’s something that needs to be fixed on Unity’s end and would be the case regardless of using InputPlus or not. It’s recommended that developers plan their code to make this the least painful as possible (pausing and saving the game) or to at least warn players not to disconnect their controllers.

Xbox 360 wireless controllers in windows also have a slight problem, when you connect them (before starting the game) they somehow queue themselves incorrectly, this may be a driver issue or even another problem with Unity. Using a single controller won’t reveal this problem, but multiple controllers do. Please instruct the player to connect all their xbox wireless controllers and then turn off the highest numbered controller and reconnect it. Repeat this in a descending order through all the controllers, then they will be queued properly and Unity will see controller 1 as 1, 2 as 2, and so forth.

dPad - Axis vs. Buttons
Because some gamepads use buttons for the dPad and some use axis representation, the best way to have universal support is for InputPlus to handle both and only report as one type. So InputPlus is set up to report the four dpad directions separately as values of either 0 or 1.

Documentation:

Setting up a project

1. **InputManager.asset**
    a. You should note any input settings you may have made in Edit>Project Settings>Input. Make a backup of this file, because you will overwrite it, clearing your settings, and if you changed it you may want to view those settings again.
    b. Unzip the InputManager.zip file.
    c. Copy the resulting InputManager.asset file into the ProjectSettings folder and overwrite it when asked.

2. **InputPlus.cs**
    a. InputPlus contains the code that reads and interprets from the InputManager. It should be in your project's Assets folder, and can be placed anywhere inside it.
3. **using InputPlusControl;**
    a. use the InputPlusControl namespace in files where you'll want to access InputPlus functions (typically in your scripts handling control)
4. **Examples**
    a. Please look at the GuiExample and then ControlExample to see how to set up controller learning and to see an example of how InputPlus is used.

Glossary and Terms

- **Thumb Sticks** - ThumbLeft_x, ThumbLeft_y, ThumbLeft, ThumbRight_x, ThumbRight_y, ThumbRight - the analog sticks _x and _y denote those each axis for the stick, and just the name indicates pressing the stick button
- **Interface Buttons** - Interface_left, Interface_right - Commonly seen as "Select" and "Start" or "Back" and "Start," these are the interface buttons, commonly associated with accessing a game's interface.
- **FP Buttons** - FP_top, FP_bottom, FP_left, FP_right - these Front Panel buttons are the typical buttons on the right side of the controller, commonly references and A, B, X, and Y, as symbols, or even as numbers on some gamepads.
- **dPad** - dpad_up, dpad_down, dpad_left, dpad_right - this is the plus symbol shaped controller which has been a staple of gamepads since their inception.
- **Shoulder Buttons** - ShoulderTop_left, ShoulderTop_right, ShoulderBottom_left, ShoulderBottom_right - the four buttons on the far edge of the controller when held in hands.
- **Modern Gamepad** - A gamepad with all the common elements listed above that are found on today's console controllers and many gamepads
- **Raw Values** - these values represent the actual information Unity receives from the controller. InputPlus will manipulate these values to a common usage across controllers
- I**nputPlus Values** - these are the values InputPlus reports, all gamepads should behave similarly when seen this way.
-

Data Structures

- **ControllerVarEnum**
    - ThumbLeft_x, ThumbLeft_y, ThumbLeft, ThumbRight_x, ThumbRight_y, ThumbRight,
    - dpad_up, dpad_down, dpad_left, dpad_right,
    - FP_top, FP_bottom, FP_left, FP_right,
    - Interface_left, Interface_right,
    - ShoulderTop_left, ShoulderTop_right, ShoulderBottom_left, ShoulderBottom_right
- **On_EVENT_Disconnect** - subscribe to this event to handle disconnecting. Currently since reconnecting controllers in Unity is not available in windows standalone builds, **reconnecting is not supported** (This is a current issue with Unity, and not something that can currently be fixed). Please consider a graceful (saving the game, maybe) strategy for controller disconnects, your players will be happier for it.

Classes

- **InputPlus** - This contains all the functions you'll need to use InputPlus. When using the InputPlusControl namespace you can access InputPlus functions with this class.
- **Controller** - The Controller class is used solely by InputPlus. Normal InputPlus useage shouldn't need to use the Controller class. It contains functions, data, and settings for a controller which InputPlus will utilize.

InputPlus Constant Variables

- **MAX_CTRL_SIZE** - the max number of controllers the default is 11, which is Unity's current max, feel free to change it to 4, for example, if your game only has up to four players. However, you don't need to change it.
- **DEAD_ZONE** - This is the analog stick dead-zone, the range at which the analog controls report 0 instead of smaller floating point values.

InputPlus Public Functions

- **Initialize -** Starts up InputPlus
- **SetDebugText -** Turn on some debug text from InputPlus
- **GetData -** Reads the value of a control from a controller
- **LearnController -** Starts the learning sequence. A player is prompted to manipulate a controller in a specific manner, teaching InputPlus about the controller. This sequence must be completed to save the controller.
- **LearnControllerSingle -** Currently an experimental feature, it is not recommended to use this yet.
- **ToggleDataView -** Switch between InputPlus's various data displays and off.
- **GetProgrammingStatus -** Determine if InputPlus is currently learning a controller.
- **CancelProgramming -** Stops programming a controller, the changes made will stay, but are not saved.
- **GetControllerName -** Get the name of a controller
- **GetListeningFor -** Get the number of the controller that is currently being listened to during programming.

Notifications

The photo image of the modular synthesizer used in the background for InputPlus on the Unity Asset Store is from the Wikimedia Commons and by Nina Richards (who can be contacted via userame ZoeB on Wikipedia). It is used under the Creative Commons Attribution 3.0 Unported license. It has been altered from the original image.