EmbeddedInputModule

A module built with Unity Engine's new InputSystem to mimic the legacy UnityEngine.Input module.

This REQUIRES Unity's new InputSystem package to be in your project.
Instructions on how to find this package are at the top of the EmbeddedInputModule.cs file.

Introduction

EmbeddedInputModule is a single C# file which takes as much of the structure and names of members within the now legacy UnityEngine.Input class (henceforth Legacy Input Module), building functionality of each member within back up using Unity Technologies' new InputSystem. It includes all the documentation and is (hopefully) a direct drag-and-drop replacement for the old system in most cases. This module seeks to provide a quick, fast and well-documented means of still using the same members and methods that those developers are used to, while leveraging all the benefits of the new system where possible.

This module started life during my time at University, where I was working on one of my Advanced Games Programming tasks. It exists because the though of "wouldn't it be cool if I could just switch controllers right in the middle of gameplay?" had occured during one of several tests of the camera system. During which, there were attempts to use the InputManager, to no avail. Upon seeing that InputSystem had controller support and some hotplugging features for at the very least switching controllers, that was attempted.

However, during attempts to use InputSystem, the InputActions, InputMaps and Editor Window for managing the InputSystem package felt rather disjointed compared to the C# approach which I had been much more accustomed to and longed to return to. For a week straight afterwards in November, this module saw an extremely fast development cycle to be what it is now with the goal of facilitating the ability to quickly identify and hot-swap controllers even when multiple are plugged in.

Features

EmbeddedInputModule mostly consists of the names of members and methods which mostly mimic the behaviour of the Legacy Input Module. However, it also features a few extra things which may help with Quality-of-Life when working on making your project work with controllers (Xbox, PS4/PS5 and Switch controllers at the time of writing).

After this page, you can find an absolute HEAP of information about what's available. 3 pages of comparing the input systems, 3 pages just for the new API stuff and a page just for the extra configuration options at the top of the EmbeddedInputModule.

Installation

(What you probably care most about)

To make use of EmbeddedInputModule, download the EmbeddedInputModule.cs file within this repository and copy it into Your Unity project's **Assets** folder. It's *suggested* that **Assets/Input** or **Assets/Resources** are created and/or used so that this module can be easily found (and modified to your heart's content).

How to call through "Input" rather than "EmbeddedInputModule"

(Alternatively, how I learned to love aliasing)

If you want to use EmbeddedInputModule as if you were directly using the Input class itself, add the following line below the other "using" lines at topof the script you are looking to use this module within:

```
using Input = EmbeddedInputModule;
```

As long as you are not using the old *UnityEnigne.Input* class nor the new *InputSystem* within your script, then this will work fine. If not, check if your script uses either of those and amend according to your needs.

Compatibility with Legacy Input Module

An ongoing process for EmbeddedInputModule (**EIM**) is the compatibility with the Legacy Input Module (**LIM**). With the use of the new InputSystem (**IS**) package there are still API features not available yet, which **EIM** is affected by as well. If a member, method or class from the **LIM** is marked as available within **EIM** in the table below, it has been translated to be used exactly as the legacy UnityEngine.Input API describes in the exact same way.

Available under **InputSystem** means that Legacy Input Module behaviour exists but usually requires accesing InputDevice or a sub-class. Available with caveats means it has to be rewritten or manually added and handled (i.e. KeyCode support).

KEY: ✓ → Available ✓ * → Available with caveats X → Not Available ? → Absent in Documentation

Member Compatibility

	LIM	IS	EIM	NOTES
Input.acceleration	✓	✓	✓	
Input.accelerationEventCount	✓	X	X	No API.
Input.accelerationEvents	✓	X	X	No API.
Input.anyKey	✓	✓ *	✓	***A setting can be enabled to include Gamepad and Mouse Inputs.***
Input.anyKeyDown	✓	✓ *	✓	***A setting can be enabled to include Gamepad and Mouse Inputs.***
Input.backButtonLeavesApp	✓	X	X	No API.
Input.compass	✓	X	X	No API.
Input.compensateSensors	✓	✓	✓	
Input.compositionCursorPos	✓	✓ *	✓ *	InputSystem only has a method to set Cursor Position. ***EIM tries to track this. Report issues on GitHub with example projects.***
Input.compositionString	✓	✓ *	✓ *	***EIM** handles the event subscription and conversion noted in InputSystem.*
Input.deviceOrientation	✓	X	X	No API.
Input.gyro	✓	✓ *	✓ *	Input.rotationRateUnbiased has no API. *Gyro is now a public static class.*
Input.imeCompositionMode	✓	X	X	No API.
Input.imeIsSelected	✓	✓	✓ *	A get-set property is created for this. Needs testing so **Asterisk'd for now.**
Input.inputString	✓	✓ *	✓ *	A get property is created for this. Needs testing so **Asterisk'd for now.**
Input.location	✓	X	X	No API.
Input.mousePosition	✓	✓	✓	
Input.mousePresent	✓	✓	✓	
Input.mouseScrollDelta	✓	?	X	Not documented or referenced in any version of InputSystem.
Input.multiTouchEnabled	✓	X	X	No API.
Input.penEventCount	✓	?	X	Not documented or referenced in any version of InputSystem.
Input.simulateMouseWithTouches	✓	X	X	No API.
Input.stylusTouchSupported	✓	X	X	No API.
Input.touchCount	✓	✓	✓	
Input.touches	✓	✓	✓	Touches are of the InputSystem Touch class, not UnityEngine.Touch class.
Input.touchPressureSupported	✓	X	X	No API.
Input.touchSupported	✓	✓	✓	

Every attempt has been made to provide, find alternatives to or replace missing API features.

Input.gyro does not work exactly as the original member does, being replaced by a static class which mimics the underlying members of the original **UnityEngine.Gyroscope** class. This class directly hooks onto any current Gysoscope, GravitySensVor, AttitudeSensor, Accelerometer and LinearAccelerationSensor in the device the application is running on and attempts to retrieve data from them. In the event the device is not found, the default for the object type to be returned is given.

Input.anyKey and **Input.anyKeyDown** makes use of the current keyboard device's synthetic **AnyKey** button. Instead of Unity's suggestion for wasUpdatedThisFrame, the use of Keyboard.isPressed and Keyboard.wasPressedThisFrame are used as they behave as expected of the original input system. **wasUpdatedThisFrame** may still work.

KEY: ✓ → Available ✓ * → Available with caveats X → Not Available ? → Absent in Documentation

Method Compatibility

	LIM	IS	EIM	NOTES
Input.ClearLastPenContactEvent	✓	?	X	Not documented or referenced in any version of InputSystem.
Input.GetAccelerationEvent	✓	X	X	No API.
Input.GetAxis	✓	X	X	*In-Development as part of an add-on to EmbeddedInputModule.*
Input.GetAxisRaw	✓	X	X	No current use-case, so was skipped for the time-being.
Input.GetButton	✓	X	X	*In-Development as part of an add-on to EmbeddedInputModule.*
Input.GetButtonDown	✓	X	X	*In-Development as part of an add-on to EmbeddedInputModule.*
Input.GetButtonUp	✓	X	X	*In-Development as part of an add-on to EmbeddedInputModule.*
Input.GetJoystickNames	✓	✓ *	✓ *	Returns a list of discovered devices by InputSystem. ***A setting can be enabled to only return names of Gamepads.***
Input.GetKey	✓	✓ *	✓	*Custom ported*. Some KeyCodes aren't supported.
Input.GetKeyDown	✓	✓ *	✓	*Custom ported*. Some KeyCodes aren't supported.
Input.GetKeyUp	✓	✓ *	✓	*Custom ported*. Some KeyCodes aren't supported.
Input.GetLastPenContactEvent	✓	?	X	Not documented or referenced in any version of InputSystem.
Input.GetMouseButton	✓	✓ *	✓	*Support for **Mouse4** and **Mouse5** as Ids [**3**] and [**4**] added.*
Input.GetMouseButtonDown	✓	✓ *	✓	*Support for **Mouse4** and **Mouse5** as Ids [**3**] and [**4**] added.*
Input.GetMouseButtonUp	✓	✓ *	✓	*Support for **Mouse4** and **Mouse5** as Ids [**3**] and [**4**] added.*
Input.GetPenEvent	✓	X	X	No API.
Input.GetTouch()	✓	✓	✓	Polymorphic method. GetTouch() retrieves the last touch. GetTouch(int) retrieves the touch at the given index, if it's in range.
Input.IsJoystickPreconfigured	✓	X	X	*Not needed due to Gamepad auto-mapping controls.*
Input.ResetInputAxes	✓	X	X	*In-Development as part of an add-on to EmbeddedInputModule.*
Input.ResetPenEvents	✓	?	X	Not documented or referenced in any version of InputSystem.

Every attempt has been made to provide, find alternatives to or replace missing API features.

Input.GetKey, **Input.GetKeyDown** and **Input.GetKeyUp** were ported over with the ability to use UnityEngine's old KeyCode enum. Each value was, to the best ability possible, carefully compared and translated over where possible. However, there are still keycodes which at this time currently do not have an equivalent *Keyboard* key. Additionally some regions have keyboard keys (e.g. Tilde) in various locations. For missing KeyCodes falling into the latter case, a KeyboardRegionLayout setting is under consideration to provide an adequate means for developers to handle these.

Methods for Pen Events within the Legacy Input Module currently do not have any documentation or point of reference in InputSystem. Although they are present in UnityEngine.Input, it is possible that the InputSystem either already has included the intended behaviours within places, such as the PenDevice. Until more is known about these specific methods and how to best re-incorporate them into EmbeddedInputModule, these have been skipped.

On the topic of the Virtual Axes featured within LegacyInputModule's InputManager, it was originally considered to not be possible, however a 2015 forum post has since highlighted otherwise. Upon further investigation, it was deemed that while GetAxis is still of use (though GetAxisRaw may not be), any such work is best kept as an optional add-on as custom inspectors and extra classes to facilitate the re-implementation of Virtual Axes would result in extra files. In this future VirtualAxes addon module,the goal is to offer the support (as best as possible) for any axes-related methods which are currently not included.

As to why Virtual Axes will not appear in the main module, that in the goal of this module. EmbeddedInputModule is about providing a single file which contains everything necessary to make use of all of the benefits of Unity's new InputSystem while offering the documentation, members and methods of the Legacy Input Module. Virtual Axes creating additional classes and inspectors means there are extra requirements the end-user must perform to get the drag-and-drop replacement of that same functionality. It also means that additional documentation is required specifically to help developers set the add-on up.

Additions

EmbeddedInputModule also includes some extra members and methods as quality-of-life additions to remove some of the extra steps required to get some Inputs. On top of this, a class, a method used internally and some enums exist which are exposed so that developers can independently add in any additional input devices or extend to more device identifiers. Lastly, it was decided to separate the *Pressed*, *Released* and *Held* controller input members for the sake of legibility.

The following lists *assume* that there is a directive at the top of a developer's code giving the alias of **Input** to this module.

New Members (Unique)

	Description
Input.PlatformIcons	Which platform-specific icons should UI elements be displaying.
Input.CurrentGamepad	The DeviceIdentifier object for the current controller device being used for input. This changes based on the last controller used for input, if there are various connected.

New Members (Device *or* Sensor Present)

	Description
Input.gamepadPresent	Returns true if a gamepad has been detected and set as the current gamepad.
Input.keyboardPresent	Returns true if Keyboard.Current isn't null.
Input.accelerometerPresent	Returns true if there is an accelerometer on the device.
Input.gyroscopePresent	Returns true if there is a gyroscope on the device.
Input.altitudeSensorPresent	Returns true if there is an altitude sensor on the device.
Input.gravitySensorPresent	Returns true if there is a gravity sensor on the device.
Input.linearAccelerationSensorPresent	Returns true if there is a linear acceleration sensor on the device.

Some are used by the new static class mimicing **Input.gyro**. Others are available because **Input.mousePresent** is.

New Members (Controller Input Pressed)

	Description
Input.LeftStickButton	Returns true if the Left Joystick Button on the currnet Controller was pressed this frame.
Input.RightStickButton	Returns true if the Right Joystick Button on the currnet Controller was pressed this frame.
Input.LeftShoulder	Returns true if the Left Shoulder on the currnet Controller was pressed this frame.
Input.RightShoulder	Returns true if the Right Shoulder on the currnet Controller was pressed this frame.
Input.LeftTrigger	Returns true if the Left Trigger on the currnet Controller was pressed this frame.
Input.RightTrigger	Returns true if the Right Trigger on the currnet Controller was pressed this frame.
Input.ButtonNorth	Returns true if the North Facing button on the currnet Controller was pressed this frame. [**Y** - Xbox | **Triangle** - Playstation | **X** - Switch]
Input.ButtonSouth	Returns true if the South Facing button on the currnet Controller was pressed this frame. [**A** - Xbox | **Cross** - Playstation | **B** - Switch]
Input.ButtonEast	Returns true if the East Facing button on the currnet Controller was pressed this frame. [**B** - Xbox | **Circle** - Playstation | **A** - Switch]
Input.ButtonWest	Returns true if the West Facing button on the currnet Controller was pressed this frame. [**X** - Xbox | **Square** - Playstation | **Y** - Switch]
Input.DpadUp	Returns true if D-Pad Up on the currnet Controller was pressed this frame.
Input.DpadDown	Returns true if D-Pad Down on the currnet Controller was pressed this frame.
Input.DpadLeft	Returns true if D-Pad Left on the currnet Controller was pressed this frame.
Input.DpadRight	Returns true if D-Pad Right on the currnet Controller was pressed this frame.
Input.Select	Returns true if the Select button on the currnet Controller was pressed this frame. [**View** - Xbox | **Share** - Playstation | **Minus** - Switch]
Input.Start	Returns true if the Select button on the currnet Controller was pressed this frame. [**Menu** - Xbox | **Options** - Playstation | **Plus** - Switch]

New Members (Controller Input Released)

	Description
Input.LeftStickButtonReleased	Returns true if the Left Joystick Button on the currnet Controller was released this frame.
Input.RightStickButtonReleased	Returns true if the Right Joystick Button on the currnet Controller was released this frame.
Input.LeftShoulderReleased	Returns true if the Left Shoulder on the currnet Controller was released this frame.
Input.RightShoulderReleased	Returns true if the Right Shoulder on the currnet Controller was released this frame.
Input.LeftTriggerReleased	Returns true if the Left Trigger on the currnet Controller was released this frame.
Input.RightTriggerReleased	Returns true if the Right Trigger on the currnet Controller was released this frame.
Input.ButtonNorthReleased	Returns true if the North Facing button on the currnet Controller was released this frame. [**Y** - Xbox | **Triangle** - Playstation | **X** - Switch]
Input.ButtonSouthReleased	Returns true if the South Facing button on the currnet Controller was released this frame. [**A** - Xbox | **Cross** - Playstation | **B** - Switch]
Input.ButtonEastReleased	Returns true if the East Facing button on the currnet Controller was released this frame. [**B** - Xbox | **Circle** - Playstation | **A** - Switch]
Input.ButtonWestReleased	Returns true if the West Facing button on the currnet Controller was released this frame. [**X** - Xbox | **Square** - Playstation | **Y** - Switch]
Input.DpadUpReleased	Returns true if D-Pad Up on the currnet Controller was released this frame.
Input.DpadDownReleased	Returns true if D-Pad Down on the currnet Controller was released this frame.
Input.DpadLeftReleased	Returns true if D-Pad Left on the currnet Controller was released this frame.
Input.DpadRightReleased	Returns true if D-Pad Right on the currnet Controller was released this frame.
Input.SelectReleased	Returns true if the Select button on the currnet Controller was released this frame. [**View** - Xbox | **Share** - Playstation | **Minus** - Switch]
Input.StartReleased	Returns true if the Select button on the currnet Controller was released this frame. [**Menu** - Xbox | **Options** - Playstation | **Plus** - Switch]

New Members (Controller Input Held)

	Description
Input.LeftStickButtonHeld	Returns true if the Left Joystick Button on the currnet Controller is held down.
Input.RightStickButtonHeld	Returns true if the Right Joystick Button on the currnet Controller is held down.
Input.LeftShoulderHeld	Returns true if the Left Shoulder on the currnet Controller is held down.
Input.RightShoulderHeld	Returns true if the Right Shoulder on the currnet Controller is held down.
Input.LeftTriggerHeld	Returns true if the Left Trigger on the currnet Controller is held down.
Input.RightTriggerHeld	Returns true if the Right Trigger on the currnet Controller is held down.
Input.ButtonNorthHeld	Returns true if the North Facing button on the currnet Controller is held down. [**Y** - Xbox | **Triangle** - Playstation | **X** - Switch]
Input.ButtonSouthHeld	Returns true if the South Facing button on the currnet Controller is held down. [**A** - Xbox | **Cross** - Playstation | **B** - Switch]
Input.ButtonEastHeld	Returns true if the East Facing button on the currnet Controller is held down. [**B** - Xbox | **Circle** - Playstation | **A** - Switch]
Input.ButtonWestHeld	Returns true if the West Facing button on the currnet Controller is held down. [**X** - Xbox | **Square** - Playstation | **Y** - Switch]
Input.DpadUpHeld	Returns true if D-Pad Up on the currnet Controller is held down.
Input.DpadDownHeld	Returns true if D-Pad Down on the currnet Controller is held down.
Input.DpadLeftHeld	Returns true if D-Pad Left on the currnet Controller is held down.
Input.DpadRightHeld	Returns true if D-Pad Right on the currnet Controller is held down.
Input.SelectHeld	Returns true if the Select button on the currnet Controller is held down. [**View** - Xbox | **Share** - Playstation | **Minus** - Switch]
Input.StartHeld	Returns true if the Select button on the currnet Controller is held down. [**Menu** - Xbox | **Options** - Playstation | **Plus** - Switch]

New Members (Controller Input Axes - Vector2)

	Description
Input.LeftStick	Get the Vector2 axis representing the current controller’s Left Joystick inputs.
Input.RightStick	Get the Vector2 axis representing the current controller’s Right Joystick inputs.
Input.Dpad	Get the Vector2 axis representing the current controller’s D-Pad inputs.

New Members (Controller Input - Other)

	Description
Input.LeftTriggerActuation	Get how far down the Left Trigger is pressed.
Input.RightTriggerActuation	Get how far down the Right Trigger is pressed.

New Methods

	Description
Input.ConvertKeycode	Convert a **UnityEngine.KeyCode** enum value to it’s corresponding ButtonControl value. Returns *true* or *false* depending on if a ButtonControl was found for the given KeyCode. [**Parameters**] **KeyCode key** → The desired key *out* **ButtonControl control** → The resulting control.

New Enums

	Description
Input.DeviceType	An enum as a list of **InputDevice** types that can be identified, if a device has been found. These are: • Keyboard • Mouse • Gamepad • Pen • Unknown
Input.GamepadType	An enum as a list of **Gamepad** types that can be identified, if a device has been identified as a gamepad. These are: • Xbox • Playstation • Switch • Generic
Input.InputIconDisplayType	An enum as a list of possible types that can be used to identify which icons should be appearing for input hints, button prompts or other context clues on user interfaces. • PC • Xbox • PlayStation • Switch • Generic It is at the developers discretion to interpret the Generic value however they deem fit. It is advised that for the other categories they display icons for those respective platforms.

Although it could be possible to combine all three enumerators, or at least condense them into two enumerators, the choice to create three separate ones was based on the context of use. Despite similarities, it was easier to create, understand and utilise enumerators which were explicitly named differently. This can be modified by developers at their own leisure.

DeviceType and GamepadType also include the value “None” as the default, first value

New Class - DeviceIdentifier

Although there already existed InputDevice and it's inheriting classes, the detection of some controllers was found to be inconsistent in testing. There were also concerns that third-party controllers which present icons for a specific platform may not be correctly identified if the HID information was slightly off. DeviceIdentifer ***can't*** rectify this immediately, however it and the current values within the enumerators previously mentioned are a good enough starting point for an initial release.

In terms of what DeviceIdentifier does, it is most simply a wrapper around InputDevice with extra members that handle the identification of the the device through it's name (as of right now).

If you have a controller which has yet to be identified, please create an Issue and submit the controller's name and HID information. *Images of the controller may also help, especially in cases where the controller is not for a specific platform.*

Public Members

	Description
DeviceIdentifier.type	The current type of device. *(DeviceType enum value)*
DeviceIdentifier.gamepadType	The current type of gamepad, if the device is a gamepad. *(GamepadType enum value)*
DeviceIdentifier.device	The InputDevice object which this identifier wraps around.
DeviceIdentifier.IsKeyboard	Is this DeviceIdentifier a Keyboard?
DeviceIdentifier.IsMouse	Is this DeviceIdentifier a Mouse?
DeviceIdentifier.IsPen	Is this DeviceIdentifier a Pen?
DeviceIdentifier.IsGamepad	Is this DeviceIdentifier a Gamepad?
DeviceIdentifier.IsDualSense	Is this DeviceIdentifier a PS5 Gamepad?
DeviceIdentifier.IsXInput	Is this DeviceIdentifier an Xbox Gamepad?
DeviceIdentifier.IsSwitch	Is this DeviceIdentifier a Switch Gamepad?
DeviceIdentifier.name	Get the name of the underlying InputDevice.
DeviceIdentifier.deviceId	Get the device ID of the underlying InputDevice.
DeviceIdentifier.Null	Is the underlying InputDevice null?

Public Methods

	Description
DeviceIdentifier.ConvertKeycode	Check if the underlying InputDevice of this DeviceIdentifier matches the comparative InputDevice provided. [**Parameters**] **InputDevice comparative** → The device to compare this one to.
DeviceIdentifier.ToString	Override of the generic ToString method provided by C#. Converts the DeviceIdentifier into a readable format. The following will be returned (with line-break characters). *Device: [InputDevice.name]* *Type: [DeviceIdentifier.type]* *Gamepad Type: [DeviceIdentifier.gamepadType]*

Constructor Variants

	Description
DeviceIdentifier(InputDevice)	Create a DeviceIdentifier from the provided InputDevice and attempt to discover what type of device it is, including the GamepadType if it is found to be a gamepad.
DeviceIdentifier(InputDevice, DeviceType)	Create a DeviceIdentifier from the provided InputDevice and DeviceType. Desired if the type of device is already known to prevent unnecessary identification attmepts.

Only these are valid constructors for the DeviceIdentifier class. The default DeviceIdentifier() will not be able to identify a device as none would be passed as a parameter in the constructor.