v3.0

Learn

[download](#) | [install](#) | [documentation](#)

This short guide explains a few notions of C# that are used in Hinput, in simple terms.
If you are an experienced developer, feel free to skip to the black boxes of code to discover how to use the Hinput API.

Part 1 : Quick Start

Getting the state of a button

To determine whether a given button is pressed or not, you can use one of the following lines:

```
Hinput.gamepad[0].A
Hinput.gamepad[0].leftTrigger
Hinput.gamepad[0].rightStickClick
```

You start by calling Hinput, then you select the gamepad you would like to get, then a button. This will return true if the button is pressed, and false if it is released.

All you have to do next is add that line to the *Update* method of one of your scripts :

```
void Update () {
        if (Hinput.gamepad[0].A) {
            // Do something whenever A is pressed
        }
    }
```

justPressed and justReleased

To detect the exact frame when a button is pushed in Hinput, use the *justPressed* keyword.

```
void Update () {
    if (Hinput.gamepad[0].A.justPressed) {
        // Do something on the exact frame A is pressed
    }
}
```

Similarly, if you want to detect the exact frame when a button is released, you can use *justReleased.*

```
void Update () {
    if (Hinput.gamepad[0].A.justReleased) {
        // Do something on the exact frame A is released
    }
}
```

justPressed and justReleased are set to true for exactly one Update frame. Do NOT try to detect them in FixedUpdate, or you will risk missing them or getting the same one several times.

Getting the position of a stick

Here is how to get a stick's position as a **Vector2**:

```
Hinput.gamepad[0].leftStick
Hinput.gamepad[0].rightStick
Hinput.gamepad[0].dPad
```

The horizontal and vertical coordinates of a stick are always between -1 and 1, and the position of a released stick is always (0, 0).

Each stick also has 8 directions. They are virtual buttons, considered pressed if the stick is pushed in the right direction.

```
Hinput.gamepad[0].leftStick.left
Hinput.gamepad[0].rightStick.down
Hinput.gamepad[0].dPad.upRight
```

You can use them exactly like you would use any button, for instance you could do this:

```
void Update () {
```

```
    if (Hinput.gamepad[0].leftStick.left.justPressed) {
        // Do something on the exact frame the left stick is pushed left
    }
}
```

Vibrating a controller

Hinput integrates a plugin called XInputDotNet, which allows you to use gamepad vibration on Windows for 4 up to controllers.
Here's how to trigger a simple constant vibration:

```
Hinput.gamepad[0].Vibrate();
```

Most gamepads contain two vibration motors: a left-side motor, that feels like a low rumble, and a right-side motor, which is more of a sharp, higher-frequency buzz.
You can decide the exact intensity of each motor, as well as the duration of the vibration:

```
// Vibrate the left side at 20% intensity and the right side at 100%
// intensity for 0.5 seconds
Hinput.gamepad[0].Vibrate(0.2f, 1, 0.5f);
```

Vibrations can take a lot of time to tweak though. Hinput gives you access to vibration presets you can use to gain some time:

```
// A short and intense vibration, suitable for an impact.
Hinput.gamepad[0].Vibrate(VibrationPreset.Impact);

// A low and powerful vibration, suitable for an explosion.
Hinput.gamepad[0].Vibrate(VibrationPreset.Explosion);
```

Part 2 : Overview of the plugin

Here are some of the most useful features of Hinput, shown by class. The full list is available in the documentation.

Hinput

The main class of the Hinput package, from which you can access gamepads.

```
// This class is static, so you can access it from anywhere by typing:
Hinput
```

- *gamepad* (**List<Gamepad>**): A list of 8 gamepads, labelled 0 to 7.
- *anyGamepad* (**Gamepad**): A virtual gamepad that returns the inputs of every gamepad at once.

Gamepad

One of the 8 gamepads of Hinput.

```
// This is a Gamepad
Hinput.gamepad[0]
```

- *A, B, X, Y, leftBumper, rightBumper, back, start, leftStickClick, rightStickClick, leftTrigger, rightTrigger* (**Pressable**): The buttons of a gamepad.
- *leftStick, rightStick, dPad* (**Stick**): The sticks of a gamepad.
- *Vibrate* (no arguments): Vibrate a gamepad with a constant intensity.
- *Vibrate* (argument : vibrationPreset (**VibrationPreset**)): Vibrate a gamepad with an intensity based on a **VibrationPreset**.

Pressable

Everything that can be pressed and released. **Button**, **Trigger** and **StickDirection** are **Pressable**.

```
// This is a Pressable
Hinput.gamepad[0].A
```

- *pressed* (**bool**): Returns true if an input is pressed. Returns false otherwise.
- *released* (**bool**): Returns true if an input is released. Returns false otherwise.
- *justPressed* (**bool**): Returns true if an input has been pressed this frame. Returns false otherwise.

- *justReleased* (**bool**): Returns true if an input has been released this frame. Returns false otherwise.
- *simplePress* (**Press**): Considered pressed whenever an input is pressed.
- *doublePress* (**Press**): Considered pressed when an input has been pressed twice in a row.
- *longPress* (**Press**): Considered pressed when an input has been pressed for a long time.

Press

A specific way of pressing a **Pressable**, like a regular press, a double press or a long press.

```
// This is a Press
Hinput.gamepad[0].A.simplePress
```

- *pressed* (**bool**): Returns true if a press is pressed. Returns false otherwise.
- *released* (**bool**): Returns true if a press is released. Returns false otherwise.
- *justPressed* (**bool**): Returns true if a press has been pressed this frame. Returns false otherwise.
- *justReleased* (**bool**): Returns true if a press has been released this frame. Returns false otherwise.

Stick

A stick or a D-Pad.

```
// This is a Stick
Hinput.gamepad[0].leftStick
```

- *position* (**Vector2**): The coordinates of a stick.
- *horizontal, vertical* (**float**): The position of a stick along the horizontal and the vertical axes (between -1 and 1).
- *distance* (**float**): The distance from the current position of a stick to its origin (between 0 and 1).
- *angle* (**float**): The angle between the current position of a stick and the horizontal axis (In degrees : left=180, up=90, right=0, down=-90).
- *inPressedZone* (**Pressable**): Virtual button considered pressed if a stick is pushed in any direction.
- *up, down, left, right, upLeft, downLeft, upRight, downRight* (**Pressable**): Virtual buttons considered pressed if a stick is pushed in the right direction.

Part 3 : Best practices

Gamepad reference

If your game has a lot of different controls, it will quickly become tedious to write “Hinput.gamepad[0]” every two lines of code.

That’s why you might want to create a variable that represents your controller. The simplest way to do so is to create a private **Gamepad** variable, and assign it a value in your *Start* method.

After that, you just need to write “*gamepad*” instead of “*Hinput.gamepad[0]*” for the rest of the script.

```
using UnityEngine;
using HinputClasses;

public class MyCharacter : MonoBehaviour {
    private Gamepad gamepad;

    void Start () {
        gamepad = Hinput.gamepad[0];
    }

    void Update () {
        if (gamepad.A.pressed) { // This is much shorter
            // Do something
        }
    }
}
```

*Note that you will need to specify “using HinputClasses” at the top of your script in order to access the **Gamepad** class.*

Implicit conversions of Pressable

This expression:

```
Hinput.gamepad[0].A
```

is actually a shortcut for that one:

```
Hinput.gamepad[0].A.simplePress.pressed
```

“*simplePress*” and “*pressed*” are implicit, which means that Hinput will add them by default on every button.

You can replace “*simplePress*” with “*doublePress*” or “*longPress*”, and you can replace “*pressed*” with “*justPressed*”, “*released*” or “*justReleased*” to use different features of the package.

Here are more examples of how Hinput does implicit conversions:

//What you write	//How Hinput interprets it
gamepad.A	gamepad.A(.simplePress)(.pressed)
gamepad.A.justPressed	gamepad.A(.simplePress).justPressed
gamepad.A.released	gamepad.A(.simplePress).released
gamepad.A.justReleased	gamepad.A(.simplePress).justReleased
gamepad.A.doublePress	gamepad.A.doublePress(.pressed)
gamepad.A.longPress	gamepad.A.longPress(.pressed)
gamepad.A.simplePress.pressed	gamepad.A.simplePress.pressed
gamepad.A.doublePress.justPressed	gamepad.A.doublePress.justPressed
gamepad.A.longPress.justReleased	gamepad.A.longPress.justReleased

Implicit conversion of Stick

Similarly, the “*position*” property of a **Stick** is implicit, which means that Hinput will add it by default if the context suggests that it should be a **Vector2**.

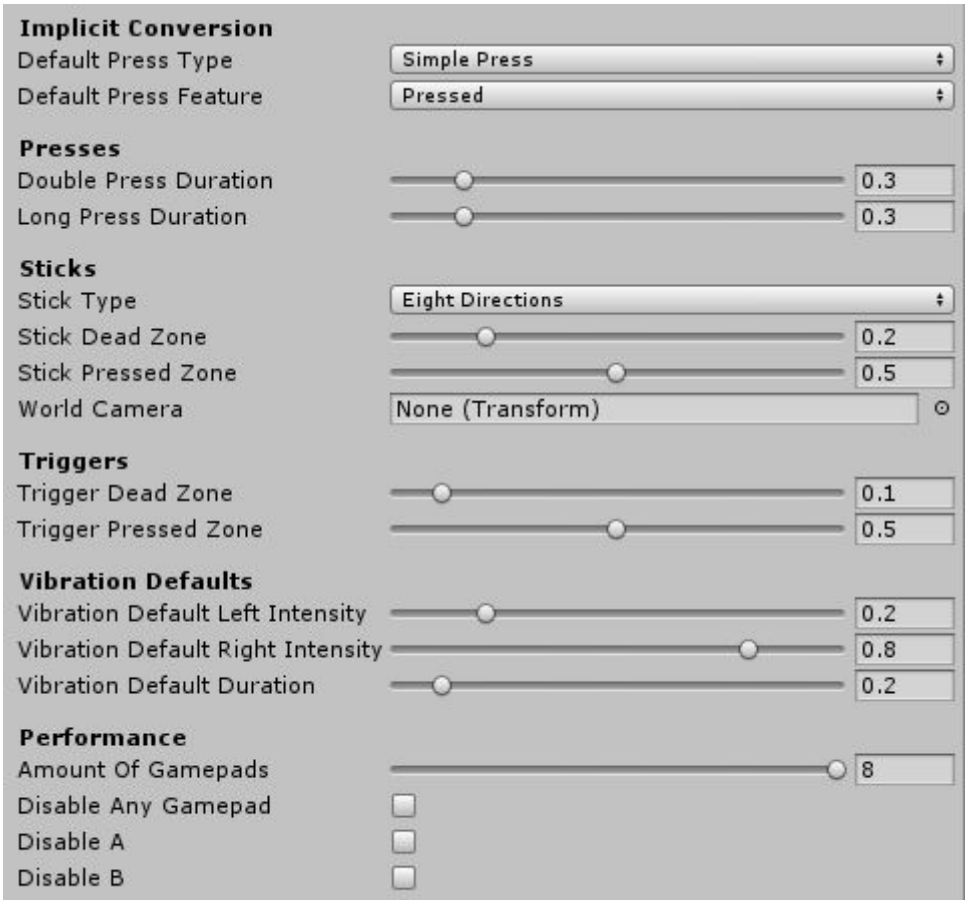
If you use a **Stick** as a **bool**, Hinput will instead use the property “*inPressedZone.simplePress.pressed*” of that stick, meaning it will return true if the stick is pushed in any direction, and false otherwise.

//What you write	//How Hinput interprets it
// As a Vector2 gamepad.leftStick	gamepad.leftStick.position
// As a bool gamepad.leftStick	gamepad.leftStick.inPressedZone.simplePress.pressed

Settings

Hinput has many parameters that you can use to adapt gamepad controls to your own game.

If you want to use the default settings of Hinput, you don't have to do anything. Otherwise, you can simply instantiate the HinputSettings prefab from the Hinput folder, or create a new gameobject with the **Settings** script on it. You can then edit its variables.



Among other things, the settings panel allows you to:

- Change the implicit conversion of **Pressable** (for instance if you want *justPressed* to be the default, instead of *pressed*).
- Decide exactly how fast (or slow) the player should be to trigger a double press or a long press.
- Make your sticks behave as 4-directional or 8-directional sticks, by changing the width of stick directions.
- Set default vibrations duration and intensity, to allow you to omit parameters in the **Vibrate** method.
- Disable the gamepads, buttons and features that you don't use, to improve the performance of your game.

You can have a look at the detailed **Documentation** for the full range of options offered by the plugin, or the **Hinput Example Scene** included in the plugin for a more hands-on approach.

If you have questions, feel free to email **hello@hinput.co**.