Getting started

hinput is a simple multi-OS gamepad system for Unity

a hilogo project from henri

Get hinput | **How to install** | **Documentation**

The basics

The most obvious thing you might want to do with hinput will be to determine whether a given button is pressed or not. Using any one of the following lines will return true or false, depending on which buttons are pressed down :

```
hinput.gamepad[0].A.pressed
hinput.gamepad[6].leftTrigger.pressed
hinput.anyGamepad.rightStick.left.pressed
```

A good way to attach an action to a button is simply to check if it is pressed in the Update method of one of your scripts. Something like this :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class MyCharacter : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (hinput.gamepad[0].A.pressed) {
            // Do something
        }
    }
}
```

How to make a simple character controller

Let's say you decided to make a game where something (usually a player character) is being moved by the stick of a gamepad. To do so, start by attaching a script to the gameobject you want to move.

In this script, you're going to use one of the features of the **hStick** class that translates a stick position to a world movement. There are two such features :

- *worldPositionFlat* : Translates a stick's position into a flat horizontal movement. Use this if you're making a 3D game or a top-down 2D game.
- *worldPositionCamera* : Translates a stick's position into a movement that faces the game's camera. Use this if you're making a side scrolling 2D game, or to move the cursor of a RTS game, for instance.

Here is how to use these features (don't use both in the same script !) :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MyCharacter : MonoBehaviour {
    // This is a "speed" variable that you can change in the inspector
    public float speed = 5;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        // To make a simple character controller, you can do this
        // (for a 3D game or a top-down 2D game)
        transform.position +=
            hinput.gamepad[0].leftStick.worldPositionFlat * speed * Time.deltaTime;

        // OR you can do this
        // (for a side scrolling 2D game, or to move a cursor)
        transform.position +=
            hinput.gamepad[0].leftStick.worldPositionCamera * speed * Time.deltaTime;
    }
}
```

You will notice that this code only makes your character move with the left stick of the gamepad 0, but I'm sure you can guess how to change that by this point.

A few more things…

You learned how to check if a button is pressed, and how to make a simple character controller, but there are a few more features of hinput that you might want to try out.

For instance, let's say you are making a platformer game, and you want your character to jump with A. You might want to write something like this :

```
// Update is called once per frame
void Update () {
    if (hinput.gamepad[0].A.pressed) {
        // Jump
    }
}
```

But if you try it out, you will notice that pressing A will make you character jump EVERY SINGLE FRAME.

Instead of the *pressed* property of the A button, you might want to use *justPressed*. It is similar to *pressed*, but it returns true only if A has started being pressed this exact frame.

There are many other button properties that you can use, here are a few examples of the most common ones :

```
    if (hinput.gamepad[0].A.justPressed) {
        // Jump
    }

    if (hinput.anyGamepad.dPad.up.justPressed) {
        // Emote
    }

    if (hinput.anyGamepad.A.released) {
        // Fall
    }

    if (hinput.anyGamepad.rightStickClick.justReleased) {
        // Crouch
    }

    if (hinput.gamepad[7].X.doublePress) {
        // Tumble
    }
```

```
if (hinput.gamepad[4].Y.longPress) {
    // Heal
}

if (hinput.gamepad[0].rightStick.vertical < 0) {
    // Look down
}

if (hinput.gamepad[6].leftStick.distance > 0.8f) {
    // Dash
}
```

Feel free to explore the different possibilities, and have a look at the full documentation for more !

Shortcut - Creating a gamepad variable

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

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 **hGamepad** variable, and assign it a value in your Start method like this :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MyCharacter : MonoBehaviour {
    private hGamepad gamepad;

    // Use this for initialization
    void Start () {
        gamepad = hinput.gamepad[0];
    }

    // Update is called once per frame
    void Update () {
        if (gamepad.A.pressed) {
            // Do something
        }
    }
}
```

Notice how you just need to write “gamepad” instead of “hinput.gamepad[0]” for the rest of the script.

This will work just fine in most situations. However, sometimes you might need your gamepad to be initialized right away, even before your Start method.

If that’s the case you can use this technique instead : (I won’t go into too many details because it’s a bit more complicated)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MyCharacter : MonoBehaviour {
    private hGamepad _gamepad;
    private hGamepad gamepad {
        get  {
            if (_gamepad == null) {
                _gamepad = hinput.gamepad[0];
            }

            return _gamepad;
        }
    }

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (gamepad.A.pressed) {
            // Do something
        }
    }
}
```

The end result is the same : you can write “gamepad” instead of “hinput.gamepad[0]” for the rest of the script.

Shortcuts : Implicit casts

This one will sound a bit technical, but it's really simple. Here it comes :

A **hPressable** can be implicitly cast to a boolean with the value *pressed*.

It means that you actually don't need to type ".pressed" after the name of a button to know whether it is pressed or not.

In other words, these two lines are exactly equivalent :

```
if (hinput.gamepad[0].A) { //...Some code
if (hinput.gamepad[0].A.pressed) { //...Some code
```

The same way, a **hStick** can be implicitly cast to a Vector2 with the value *position*. This means that the following two lines are also equivalent :

```
if (hinput.gamepad[0].leftStick == //...A Vector2
if (hinput.gamepad[0].leftStick.position == //...A Vector2
```

Settings

hinput has many settings that you can use to adapt gamepad controls to your own game.

If you want to use hinput's default settings, you don't have anything to do. At runtime, a hinput gameobject will be created automatically and handle all gamepad inputs.

However, you can also instantiate the hinputSettings prefab manually (you will find it at the root of the hinput folder, in your Project tab). It will expose some useful settings that you might want to tweak, such as the duration of a double press, the width of the stick's directions, or the size of the dead zone of the sticks and triggers.

Feel free to check out the detailed documentation for the full range of options offered by hinput !