user guide

Last updated: 02 June 2024

Foreword

If you encounter any problems in the package, have anything you would like to clarify, or would like to report a bug, please contact us at terresquall.com/contact.

Table of Contents

1. Version Changelogs

Version 1.0.3 (2 June 2024)

- **Removed the dependency on the Unity Event System and the Standalone Input Module.** The joystick will now work without an Event System on the Scene.
- **With the above change, the Joystick is no longer reliant on your Event System having a Standalone Input Module.** So if you replace that with the new `InputSystemUIInputModule` from Unity's new Input System, the Joystick will now continue working, as long as you have not disabled the old Input system.
- Added various notifications on the Inspector when the Virtual Joystick is not implemented correctly:
    - Added an error notification on the Virtual Joystick Inspector if the new Unity Input system is installed, and the old input system is turned off, as the joystick currently doesn't work with the new Unity Input system.
    - Fixed an issue in the Inspector for the Virtual Joystick where, if a **Control Stick** is not assigned for it, selecting the Virtual Joystick will constantly produce a `NullReferenceException`. This has been replaced with a note in the Inspector reminding the user to assign a **Control Stick** instead.
    - Added a warning on the Virtual Joystick Inspector if your Joystick is not assigned to a Canvas.
- The joystick will now automatically disable itself and print a message on the Console when:
    - It is not parented to a Canvas
    - The old Input system is not enabled
- A couple of existing methods and properties in the `VirtualJoystick` script has been changed, to make it easier for other components to interface with it:
    - A new function, `GetAxisDelta()`, has been added. The function returns a `Vector2` expressing the change in the axis's position from the last frame to the current frame.
    - The `GetAxis()` and `GetAxisRaw()` functions can now be called on an instance of a joystick. This will make it easier for you to code components that interface with it.
    - A couple of `private` and `protected` properties have had their access modifiers changed to `internal` instead. This makes them accessible to other components, allowing the possibility of coding extensions for the Virtual Joystick component.
- Various sections of the guide have been updated to reflect the new features and changes that have been added.

Version 1.0.2 (3 April 2024)

- Removed the **UI Text Print Axis** property, which showed the axis of a joystick if you assigned a `UnityEngine.UI.Text` object to it. The removal is because newer versions of Unity only support Text Mesh Pro, and the console debug option was improved.
- Improved on the **Console Print Axis** property. Now, it shows the name of the joystick that is producing the output; and the output only generates if you are using the joystick.
- Fixed an issue where, before it is first used, a joystick may point to (or away from) the Anchor on certain Canvases that have a scale factor that is not 1, until they are used for the first time.
- Fixed an issue where resizing the screen would cause the joystick to become deformed. Now, the joysticks will recalculate their positions whenever the screen size changes (e.g. phone rotation).
- Fixed the code blocks not showing properly on the user guide.
- Added some FAQs.

Version 1.0.1 (22 March 2024)

- Fixed an issue that caused the asset to throw errors when you try to build the project.
- Added dependency on UI and the old Text element to maintain compatibility with newer versions of Unity.

## Version 1.0.0 (3 December 2023)

- Limited initial release.

2. Setting up

Import the asset into your project. The asset should be unpacked into a folder called `VirtualJoystick` in your `Assets` folder.

To add and use a virtual joystick, drag any of the joystick prefabs from `VirtualJoystick/Prefabs` onto *any* Canvas GameObject in your Scene, and it should be ready to use.

Available joystick designs to choose from.

If you have Unity's new Input system installed on your project, you will also need to ensure that the **Active Input Handling** setting is set to **Both**. Otherwise, the original Unity Input system (which the Virtual Joystick uses) will be disabled, and the Joystick will not work.

The **Active Input Handling** field is in **Edit > Project Settings**, under **Player > Other Settings**.
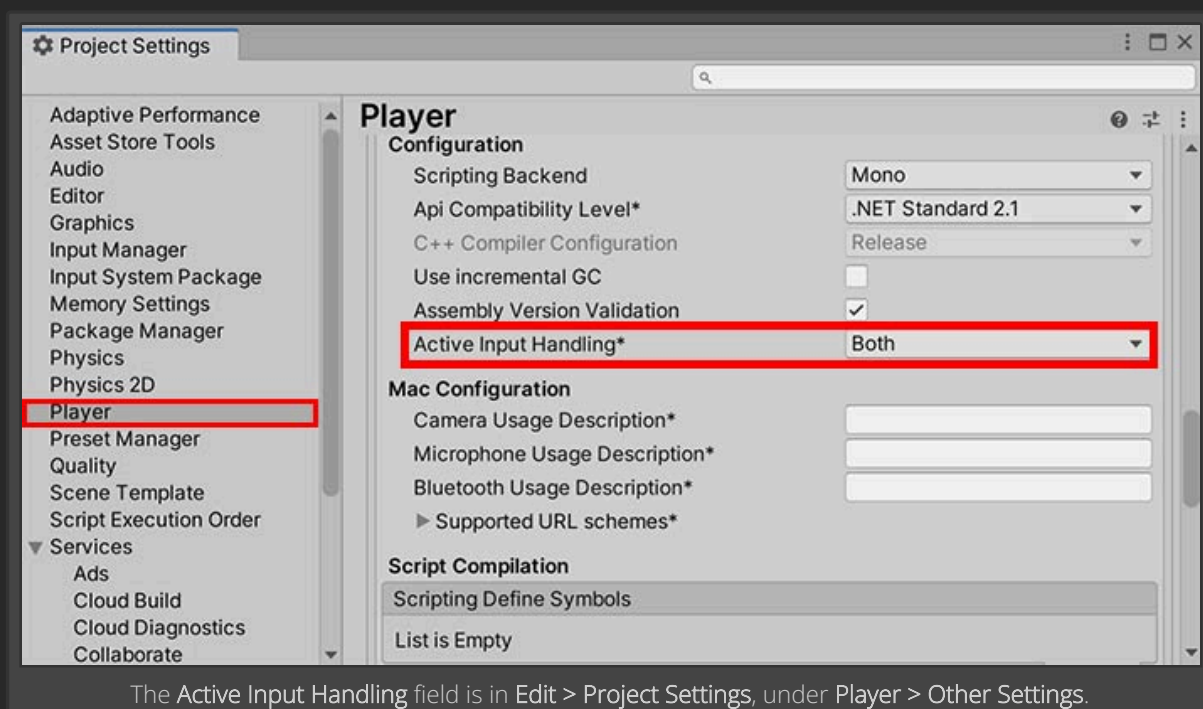
Having the wrong setting on your joystick will cause the component to print an error message informing you of how to fix the issue.

Follow the instructions above to rectify this.
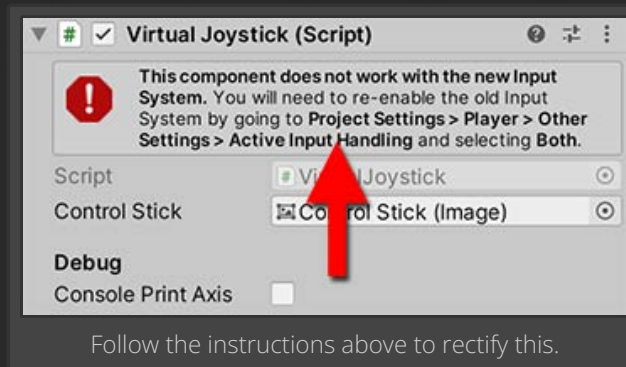
3. Reading joystick input

Once the joystick is set up, in every script where you want to read input from any of your virtual joysticks, you will need to add the following namespace to the top of your scripts:

```
using Terresquall;
```

Once that is done, you will be able to access the `VirtualJoystick` class.

a. Using `GetAxis()`

To read input from the joystick, use `VirtualJoystick.GetAxis("Horizontal")` to read horizontal offset, and `VirtualJoystick.GetAxis("Vertical")` to read vertical offset. For example, the following code moves the character in the horizontal direction the joystick is pushed:

```
// The value of x is between -1 and 1.
float x = VirtualJoystick.GetAxis("Horizontal");
transform.position += x * Time.deltaTime;
```

The function works similarly to Unity's own `Input.GetAxis()` method. Do note, however, that the `"Horizontal"` and `"Vertical"` prompts are hardcoded into the joystick and are unrelated to the values in Unity's Input Manager.

From Version 1.0.3 onwards, you can also call `GetAxis()` on an instance of a joystick.

b. Using `GetAxisRaw()`

If you want to snap the values to -1, 0 or 1, you can also use `VirtualJoystick.GetAxisRaw("Horizontal")` or `VirtualJoystick.GetAxisRaw("Vertical")`, which functions like Unity's own `Input.GetAxisRaw()`.

c. Using `GetAxis()` or `GetAxisRaw()` without arguments

If you don't like to retrieve each axis separately, you can also call `VirtualJoystick.GetAxis()` without any arguments to retrieve a `Vector2` containing the horizontal and vertical inputs.

```
Vector2 joyInput = VirtualJoystick.GetAxis();

// Moves the character with the joystick.
transform.position += joyInput.x * Time.deltaTime;
```

From Version 1.0.3 onwards, you can also call `GetAxisRaw()` on an instance of a joystick.

d. Reading multiple joysticks

If you have multiple virtual joysticks on the Scene, you will need to add an extra integer to your `GetAxis()` or `GetAxisRaw()` calls to read the 2nd virtual joystick and beyond.

```
// Reads the horizontal input of the 2nd joystick on the scene.
VirtualJoystick.GetAxis("Horizontal", 1);
// Reads the vertical input of the 3rd joystick on the scene.
VirtualJoystick.GetAxis("Vertical", 2);
```

If you want to retrieve input data for both axes, just pass the integer value of the joystick, like so:

```
// Gets the input data of the 3rd joystick.
VirtualJoystick.GetAxis(2);
```

e. Using `GetAxisDelta()`

If you want to find the change in the axis of the joystick from the last frame to the current frame, you can use the following code:

```
Vector2 delta = VirtualJoystick.GetAxisDelta();
```

If you have multiple joysticks on the screen, you can also pass an integer to the function to get your n^{th} joystick. The first joystick has an index of 0, so if you want the delta of the 2nd joystick:

```
Vector2 delta = VirtualJoystick.GetAxisDelta(1);
```

You can also call the function on an instance of your joystick:

```
VirtualJoystick vj = FindObjectOfType<VirtualJoystick>();
Vector2 delta = vj.GetAxisDelta();
```

Do note that, since the magnitude of `GetAxisDelta()` is affected by frame rate, if you want use it to judge how fast the joystick is moving, you should be dividing the resulting vector by `Time.deltaTime` first.

```
Vector2 delta = VirtualJoystick.GetAxisDelta() / Time.deltaTime;
```

4. Settings

On top of this, each virtual joystick also comes with a **Virtual Joystick** component, which has a variety of settings you can toggle.

The Virtual Joystick component.
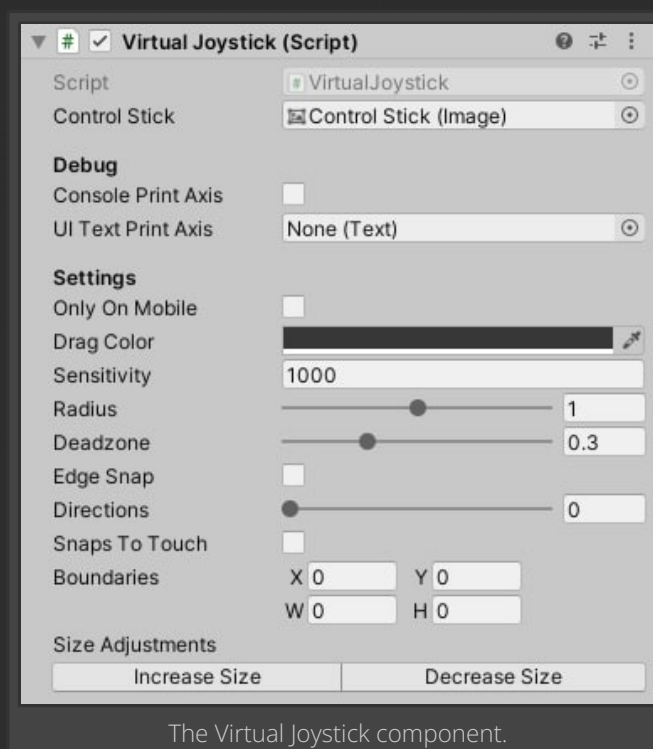
To adjust how the Virtual Joystick works, you will want to adjust the attributes under the **Settings** section. Below are a list of the properties, and what they do:

| Property	Description
Only On Mobile	Check this box if you want to hide the Virtual Joystick when the game is not being played on a mobile device. Only works on Unity 2020 and above. Works with the Device Simulator in Unity.
Drag Color	What the color of the joystick turns into when you are tapping on it. Used to provide feedback when using the joystick.
Sensitivity	This controls how responsive the joystick is.
Radius	This controls how far you can pull the control stick on the joystick away from the joystick base at the centre. When adjusting this, a red circle will be shown on the joystick, showing you how big this radius is.
Deadzone	A value between 0 and 1, representing a percentage of the maximum distance the joystick can travel. For example, if this value is 0.3, you will need to pull the joystick at least 30% away from the centre for the input to register.
Edge Snap	Only works if **Direction** is more than 0. When checked, this causes the joystick to only move along the axes of the directions it can snap to.
Direction	If more than 0, the joystick will snap to specific directions when outside the **Deadzone**. The useful values here are 4 and 8, although you are technically unrestricted to set the number here. If **Direction** is set to 4, the joystick snaps to ↑ ↓ ← → . Set it to 8, and it will snap to ↑ ↗ ← ↘ ↓ ↙ ← ↖ .
Snaps To Touch	This works together with the **Boundaries** attribute. When checked, the joystick will teleport to wherever your finger is, as long as they are within

Property	Description
	the boundaries set.
Boundaries	If a finger is tapped within the bounds denoted (in the Editor, this is a yellow box around the joystick), Snaps To Touch will occur.
Size Adjustments	For adjusting the size of the joystick, use the buttons here to make your life easier, as there is a child element inside the joystick that you have to scale up as well.

5. FAQs

The Joystick is not showing up when I drag it onto a Scene! What do I do?

If it is not showing up, that's because it wasn't parented to a Canvas. If you put it onto the Scene without parenting it to a Canvas, it won't work! You will also get the following warning on your Joystick component:


A friendly warning from the Joystick component.
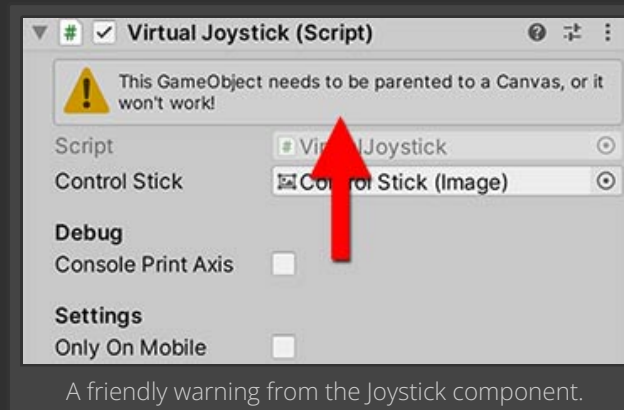
To fix this, make sure you always parent your Virtual Joystick GameObjects under your Canvas.

Now the Joystick is showing up, but it is not responding when I tap or drag it!

If your Joystick is not responding to your interaction, check to see if there are any warning or error messages on the component in the Inspector. If you see the following error message:

Follow the instructions above to rectify this.
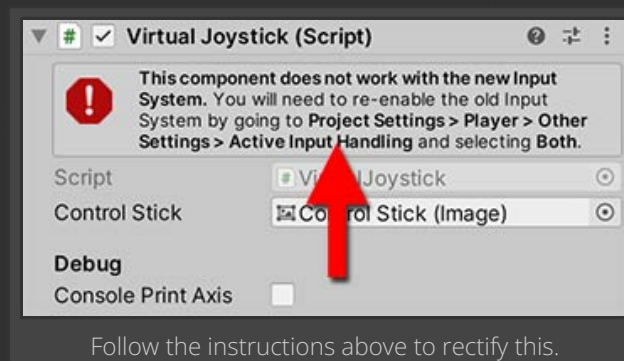
That means you have Unity's new Input system installed on your project, and you have configured it to disable the original Input system. To get the joystick working again, you will need to turn on the old Input system as well, by heading to **Edit > Project Settings**. Under the Window that pops up, go to the **Player** tab, open the **Other Settings** subwindow, and find the **Active Input Handling** dropdown. Then, set its value to **Both**.
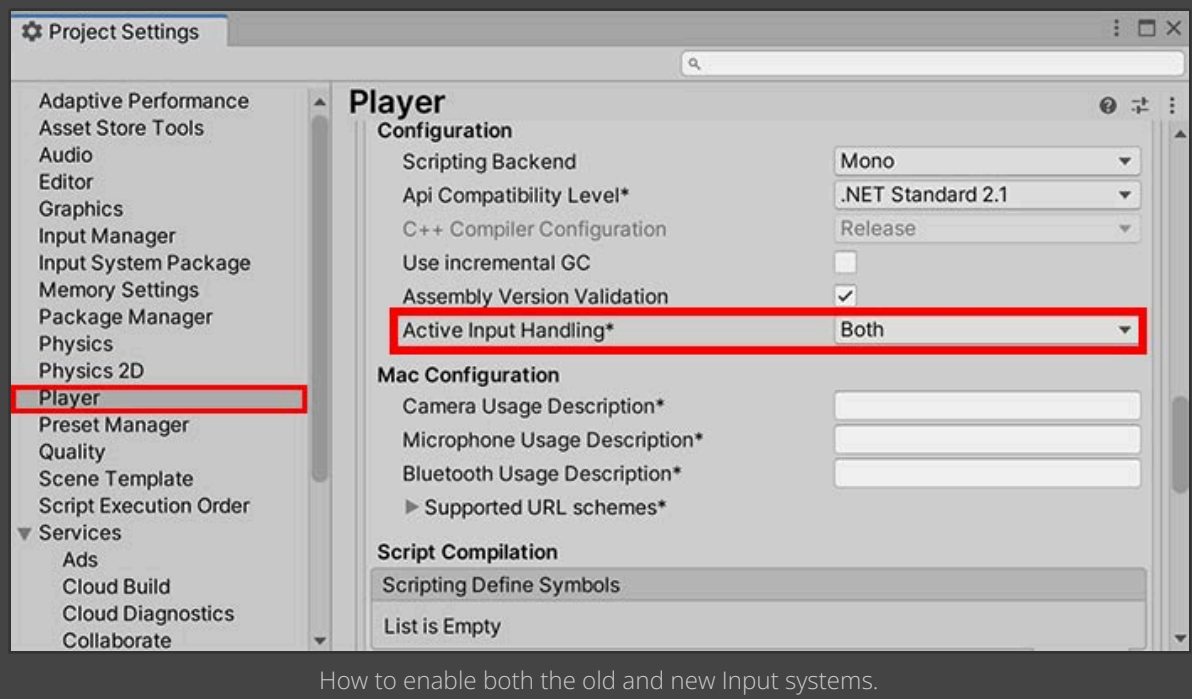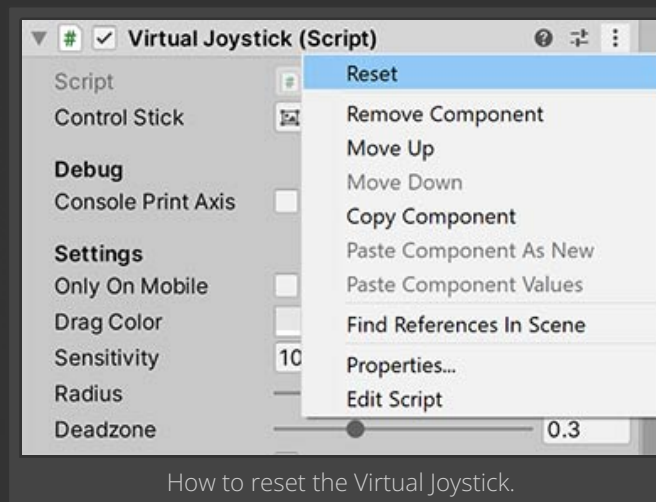
How to enable both the old and new Input systems.

We are currently working on **getting the Virtual Joystick to be compatible with the new Input system** as well.

The Joystick is still not responding to my input.

If you have made sure Unity's old Input system is enabled, but the joystick is still unresponsive, try resetting the Virtual Joystick component.

How to reset the Virtual Joystick.

You might have gotten some of the configurations wrong (e.g. if you set Sensitivity to 0, the joystick will also stop responding to player input).