Virtual JOYSTICK

user guide

Last updated: 23 March 2024

Foreword

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

Table of Contents

1. Version Changelogs

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.

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.

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

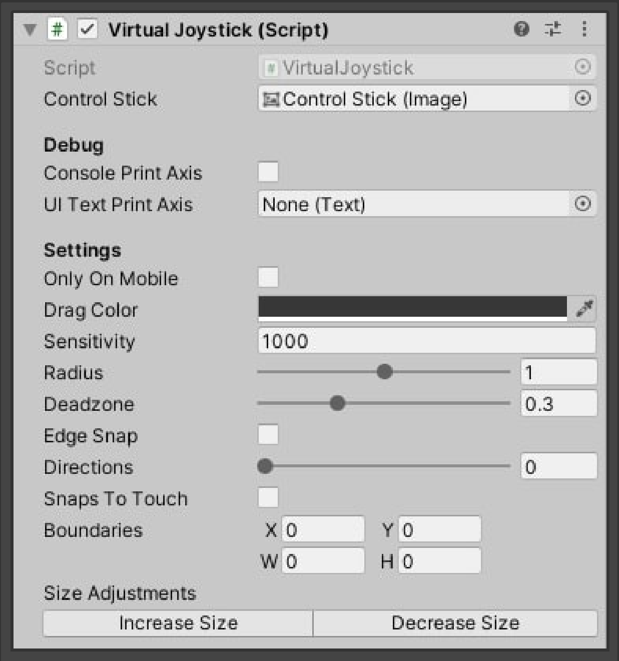
```
VirtualJoystick.GetAxis("Horizontal", 1); // Reads the horizontal input of the 2nd joystick
VirtualJoystick.GetAxis("Vertical", 2); // Reads the vertical input of the 3rd joystick on
```

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

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

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.



Virtual Joystick (Script)

Script	VirtualJoystick
Control Stick	Control Stick (Image)
Debug	
Console Print Axis	☐
UI Text Print Axis	None (Text)
Settings	
Only On Mobile	☐
Drag Color	■
Sensitivity	1000
Radius	1
Deadzone	0.3
Edge Snap	☐
Directions	0
Snaps To Touch	☐
Boundaries	X 0 Y 0 W 0 H 0
Size Adjustments	

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 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.

- **Radius:** You can set the radius property to set how far away from the joystick base the control stick can move away from.
- **Sensitivity:** This is the limit to how fast the control stick of the joystick can move to the position where your finger is.
- **Deadzones:** You can set Deadzones on the joystick either using a radius or a value. This axis will return as 0 while the control stick is either in the radius or is returning a value less than or equal to the Deadzone value.
- **Scaling:** You can increae or decrease the size of the joystick, control stick, radius, and deadzone radius by pressing the increase and deacrease button respectively.
- **Boundaries:** If Snap To Touch is toggled, you can set the W and H properties under **Boundaries** to define the area where the joystick will snap to the player's touch.

5. FAQ

Coming soon!