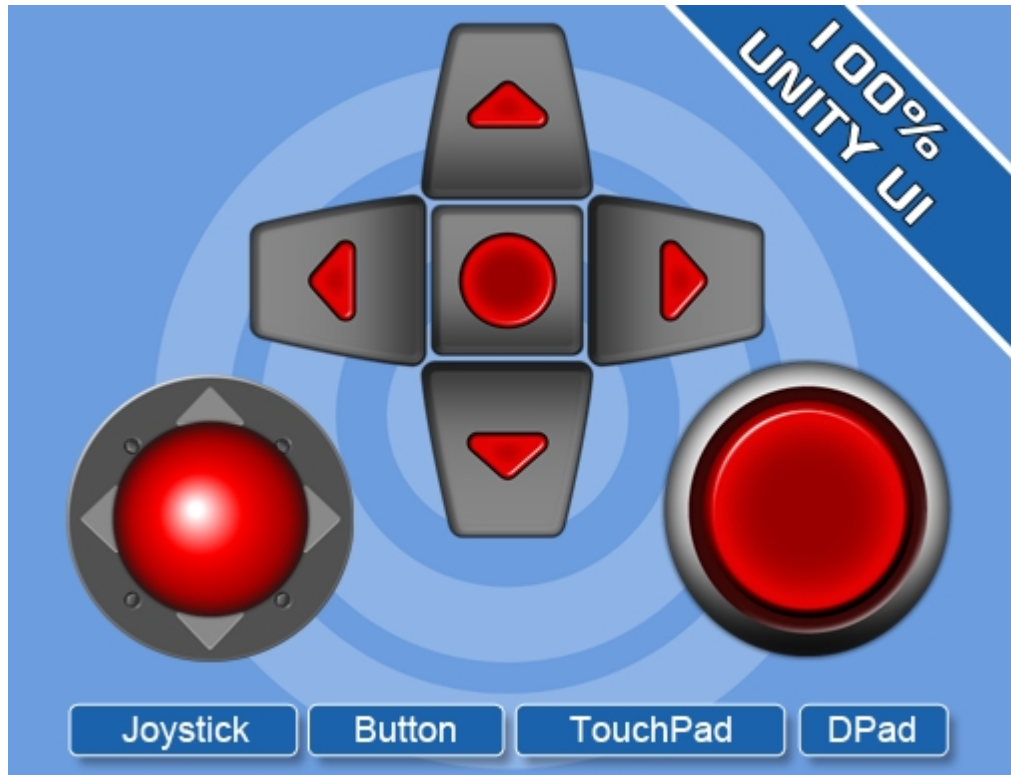


# Easy Touch Controls



## User Documentation

## Table des matières

---

Introduction .....	3
Joystick .....	5
Joystick Inspector .....	7
Position & Size .....	8
Dynamic Joystick .....	9
Static Joystick .....	10
Area .....	11
Axes properties .....	13
Sprites .....	15
Events .....	16
DPad .....	17
D-Pad Inspector .....	19
Position & Size .....	20
Axes properties .....	21
Sprites .....	23
Events .....	24
Touchpad .....	25
Touchpad Inspector .....	26
Position & Size .....	27
Axes properties .....	28
Sprites .....	30
Events .....	31
Button .....	32
Button Inspector .....	32
Position & Size .....	34
Behaviour .....	35
Sprites .....	36
Events .....	37
Direct Action Inspector .....	38
Inertia .....	40
Automatic stabilization .....	41
Clamp rotation .....	42
Events : Joystick-DPad-TouchPad .....	43
Add Events .....	45
Move Events .....	46
Touch Events .....	47
Down Events .....	48
Press Events .....	49

## Introduction

---

# Welcome

Easy Touch Controls is a set of virtual controllers exploiting the new UI & the new event system.

- **Joystick**
- **Dynamic joystick**
- **DPad**
- **DPad over the time**
- **TouchPad**
- **Button**
- **Button over the time**

### WYSIWYG

ETC is fully WYSIWYG, you can animate your player without writing a single line of code relative the complexity of what you want

### Power & Flexibility

Integrate controllers following 3 methods

- **Direct** : to animate your objects without code. Drag and drop your object and choose the axis and action (Translate, Rotate, AddForceetc ...).
- **Event** : The new event system allows you to call existing functions simply by Drag & drop your object, the inspector is responsible for exposing the existing methods.
- **Input manager** : To easily migrate your existing script. In 95% of cases, you just have to replace Input by ETCInput on your script

These three modes can be combined together, to create a amazing gaming experience.

### Unprecedented options

Depending on the nature of the controllers you have access to options like

- Dead Zone
- Speed
- Inertia
- Auto stabilization
- Clamp rotation
- On / Off axis
- Push over the time
- etc...

### Unity 4.6 Powered

Using the new UI and the new event system allows a quick and easy use controllers, with low drawcall.

### Videos

- [Migration example](#)
- [Direct action example](#)



## Joystick

---

### Joystick Overview

ETC manages two types of joystick:

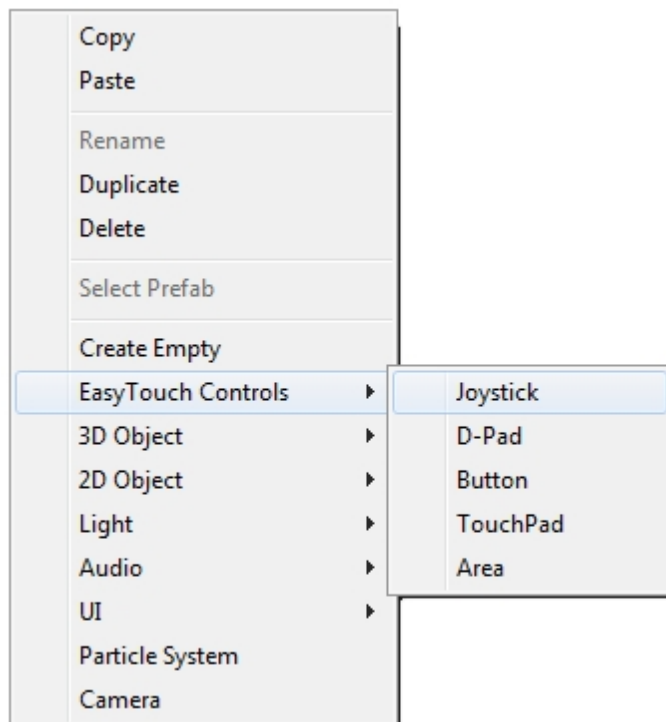
- **Static:** The joystick will be displayed at the position you have parameterized.
- **Dynamic:** The joystick will be displayed on the position of the touch. You can force the display to a given area.

A joystick is composed of 2 images, one for the background and one for the button. The joystick diameter depending on the background image (width or height according to a parameter).

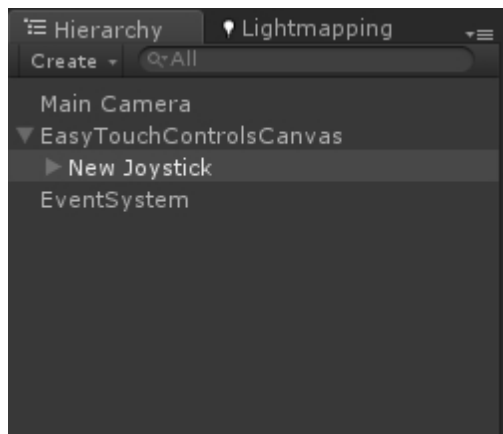
The joystick manages its axes with a value range from -1 to 1. But you can use the joystick axes in On / Off

### Creating Joystick

Right click in the hierarchy window => EasyTouch Controls => Joystick

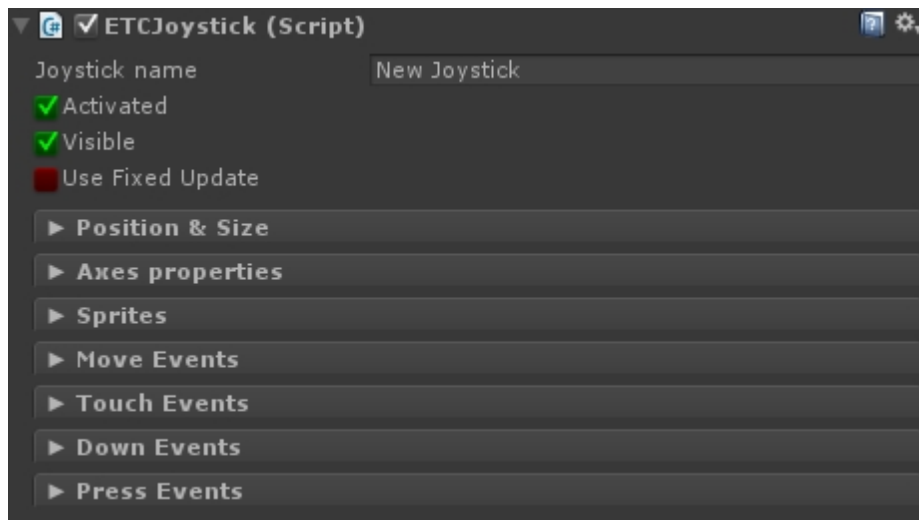


ETC will automatically create a canvas and an EventSystem gameobject. They will be set for optimum operation of ETC.



## Joystick Inspector

# Joystick Inspector



### Activated

Active or not the joystick, it is visible when disabled

### Visible

Displays or not the joystick

### Use Fixed Update

Enable this option if you use the physical.

### Position & Size

Sets the type and positioning joystick

### Axes properties

Settings axes

### Sprites

Sets the joystick images

### Move Events

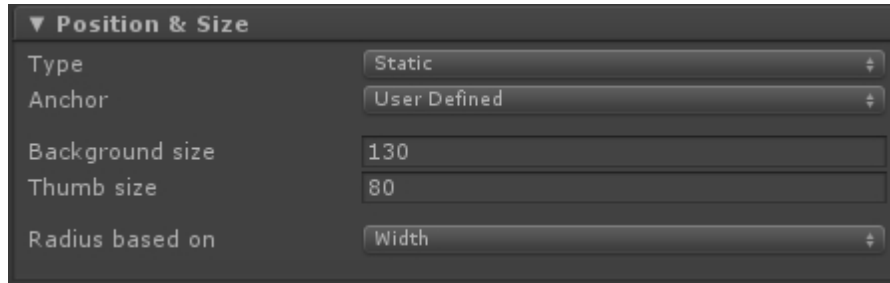
### Touch Events

### Down Events

### Press Events

## Position & Size

# Position & Size



### Type

Defines the type of joystick ([Static](#) or [Dynamic](#)).

### Background size

Size of the background image.

### Thumb size

Size of the thumb image.

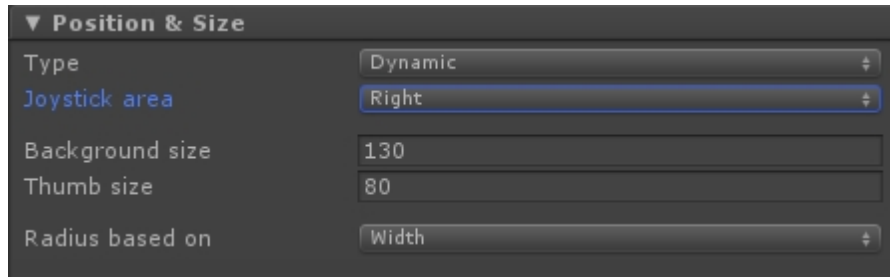
### Radius based on

Sets the dimension to be used as diameter for the joystick. Used if you don't use a square picture.



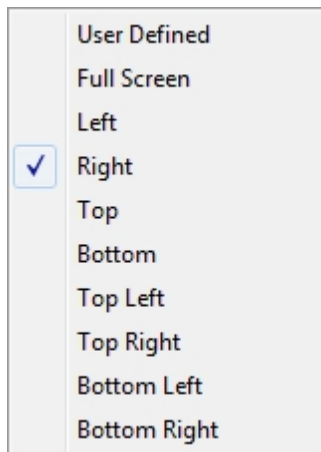
## Dynamic Joystick

# Dynamic Joystick



## Joystick area

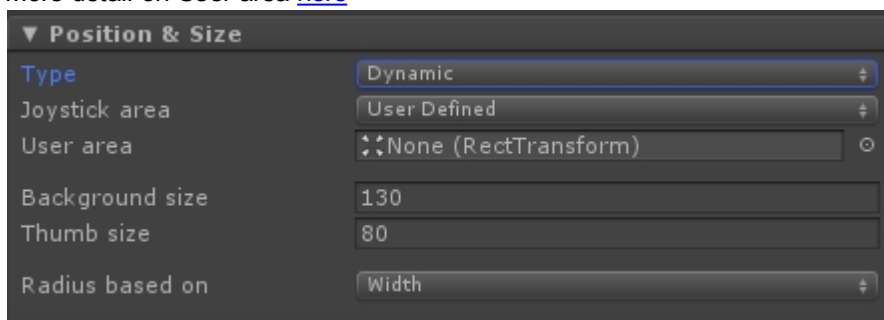
Defines the area will be allowed to display the joystick



## User defined

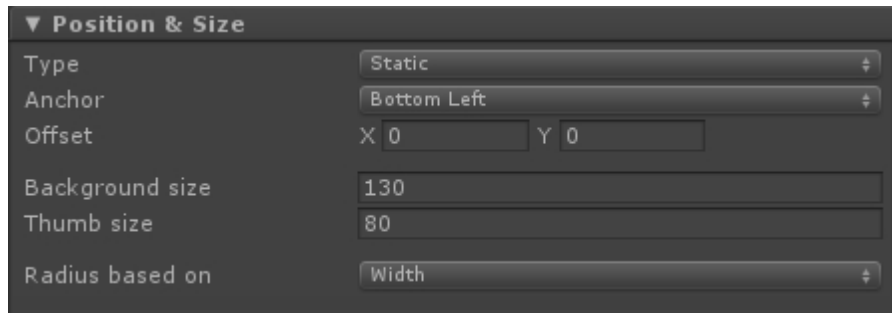
If you choose this option you must define an area manually.

More detail on User area [here](#)



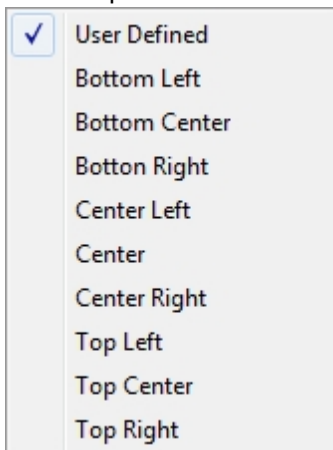
## Static Joystick

## Static Joystick



### Anchor

Sets the position and modified the joystick anchors.

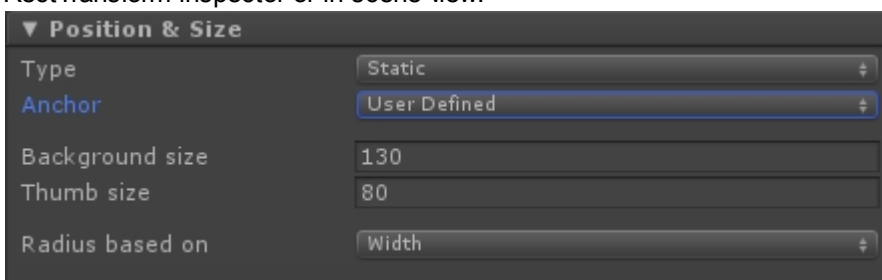


### Offset

Defines the offset to be applied with respect to the selected anchor.

### User Defined

By choosing this option, you can position the joystick manually and define yourself anchoring with the RectTransform inspector or in scene view.



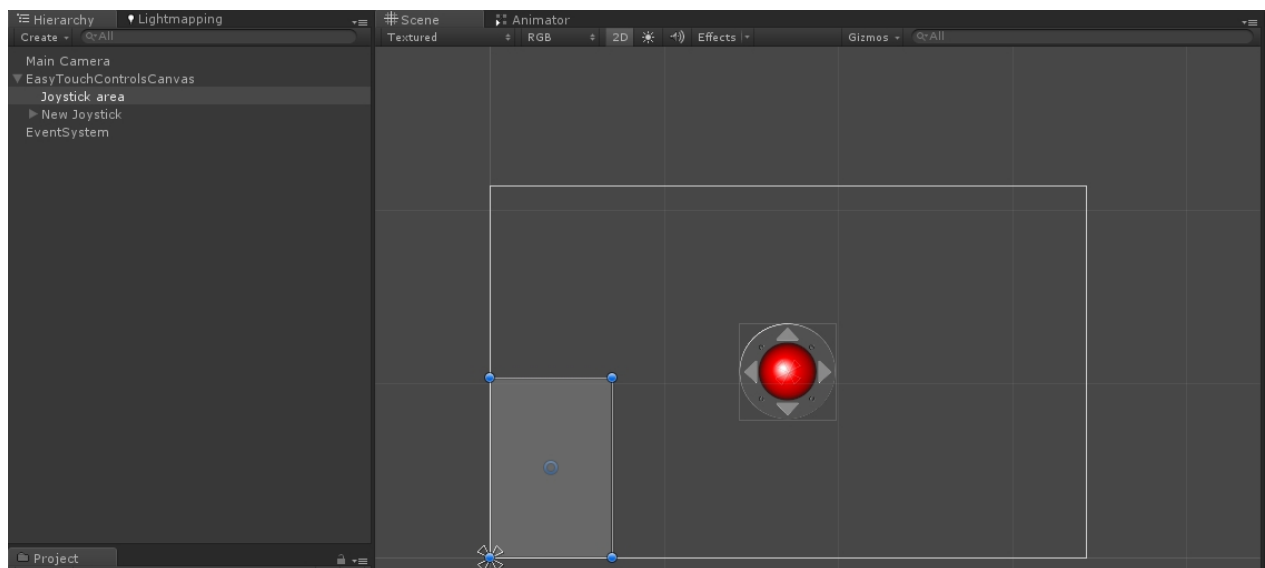
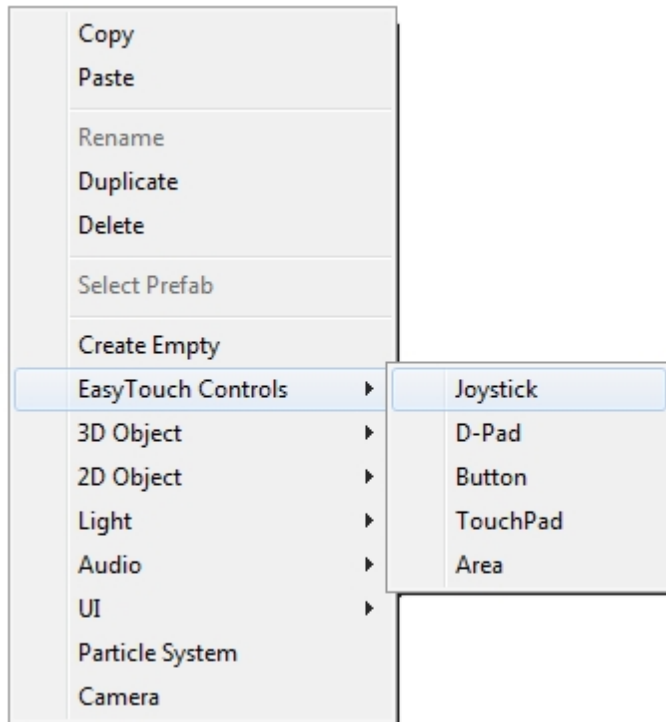
## Area

# Area

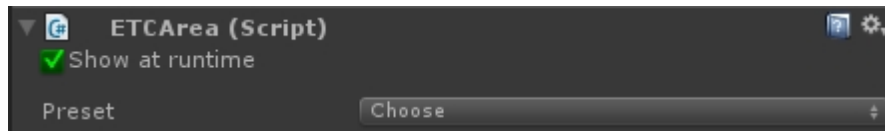
The area used to define a specific area of the screen to display a dynamic joystick

## Creating Area

Right click in the hierarchy window => EasyTouch Controls => Area



## Area Inspector



### **Show at runtime**

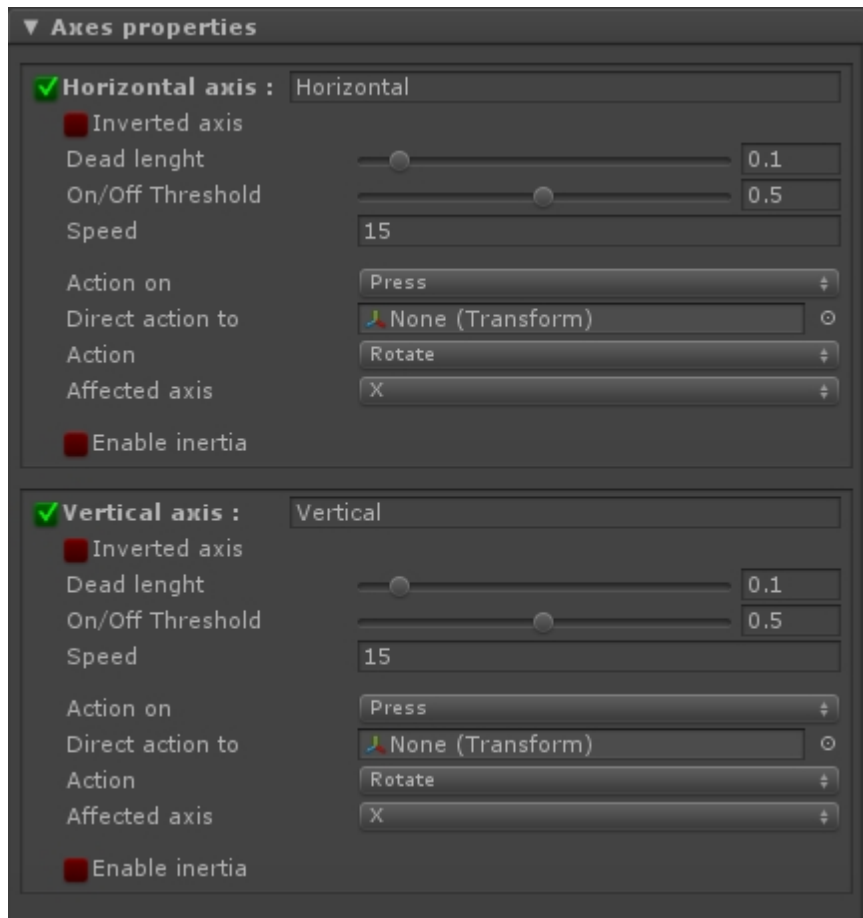
Show or not the area during the runtime

### **Preset**

Preset to set the position and anchor, if you do not want to position it yourself via the RectTransform

## Axes properties

# Axes properties



## Horizontal axis & Vertical axis

Enables or disables the axis, followed by his name for the input manager. The name of an axis must be unique for a scene

## Inverted axis

Reverse the axis

## Dead length

This value corresponds to a dead zone in relative value (0..1), where the axis will not be considered in motion

## On/Off Threshold

This value is used to determine the threshold when the axis will be considered down for the first time. Use this value if you want to manage your axis mode On / Off, or base direct action on Down axis.

## Speed

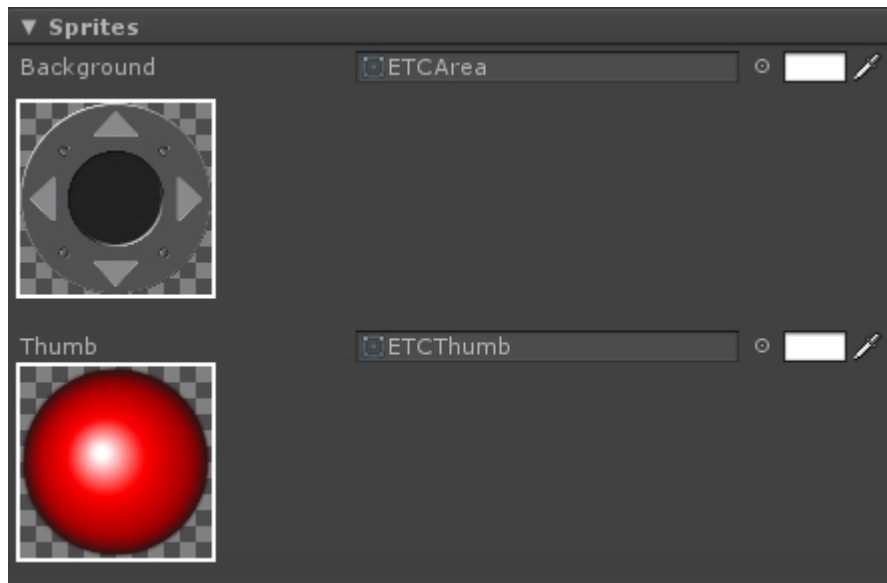
This value is used by direct mode to operate the action, and in the calculation of value returned by `ETCInput.GetAxisSpeed`. (Look at `ETCInput_API.PDF`)

## Direct Action Bloc

[More detail here](#)

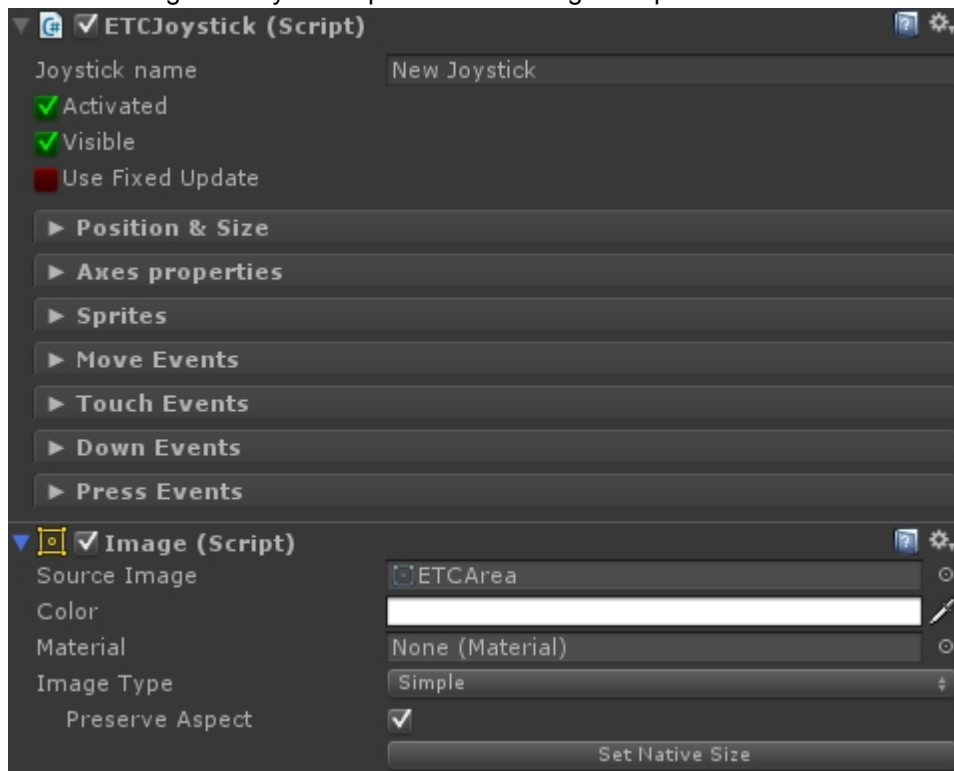
## Sprites

# Sprites



Sets the picture and color of the different parts of the joystick.

You can also go directly to the parts of each image component

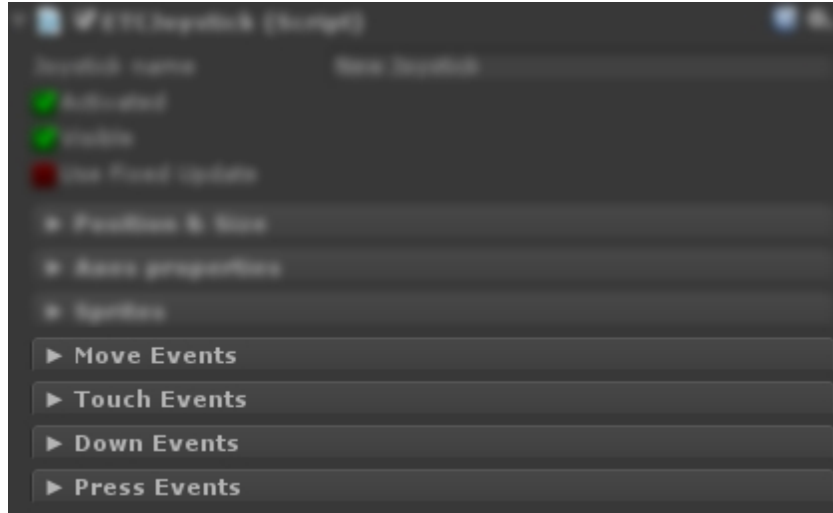


## Events

# Events

This part gives you access to all available events. Of course you can use only those that interests you.ETC uses the new event system that lets you easily use to call existing functions on scripts.

The joystick, DPad & TouchPad have the same event, they will be easy to replace a control by another one.



[More detail on all events here](#)



## DPad

---

# DPad Overview

ETC manages two types of DPad:

- **Classical:** The values return by the D-PAD are On/Off 0 or 1.
- **Over the time:** The values return by the D-PAD are relative to a step value over the time.

DPad is a square area divided into 9 parts 3X3 , ETC manages 2 or 4 axes on DPad

- **2 Axes**

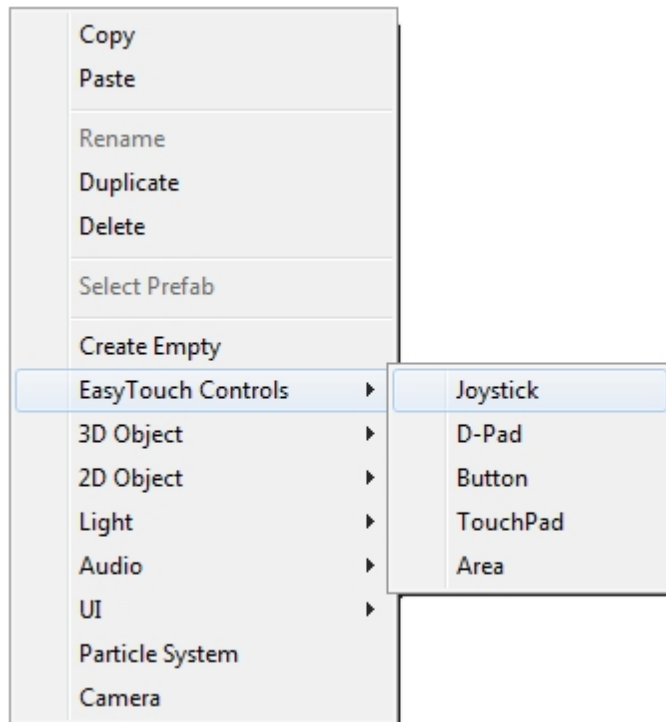
	Axi s +Y	
Axi s -X		Axi s +X
	Axi s -Y	

- **4 Axes**

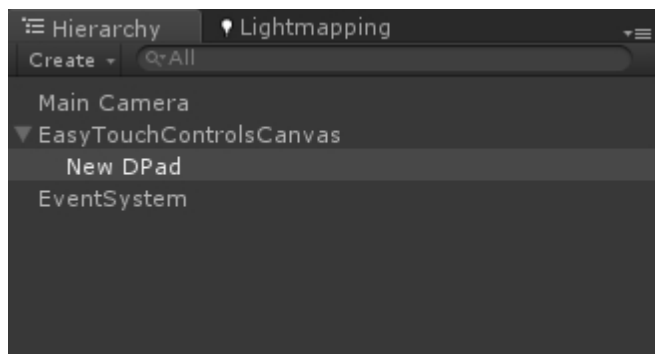
Axe s +Y- X	Axi s +Y	Axe s +Y +X
Axi s -X		Axi s +X
Axe s - Y-X	Axi s -Y	Axe s -Y +X

## Creating DPad

Right click in the hierarchy window => EasyTouch Controls => D-Pad



ETC will automatically create a canvas and an EventSystem gameobject. They will be set for optimum operation of ETC.



## D-Pad Inspector

# DPad Inspector



### Activated

Active or not the DPad, it is visible when disabled

### Visible

Displays or not the DPad

### Use Fixed Update

Enable this option if you use the physical.

### Position & Size

Sets DPad position

### Axes properties

Settings axes

### Sprites

Sets the DPad images

### Move Events

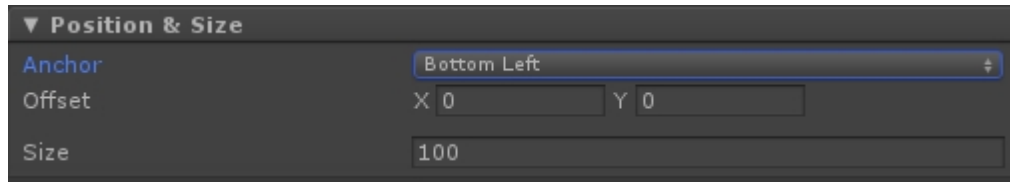
### Touch Events

### Down Events

### Press Events

## Position & Size

# Position & Size



## Anchor

Sets the position and modified the DPad anchors.

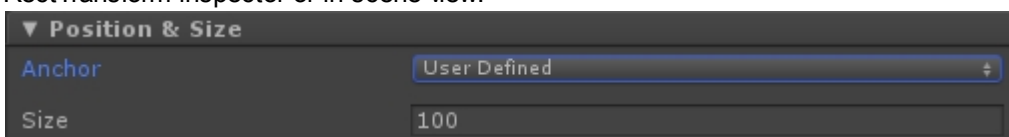


## Offset

Defines the offset to be applied with respect to the selected anchor.

## User Defined

By choosing this option, you can position the DPad manually and define yourself anchoring with the RectTransform inspector or in scene view.

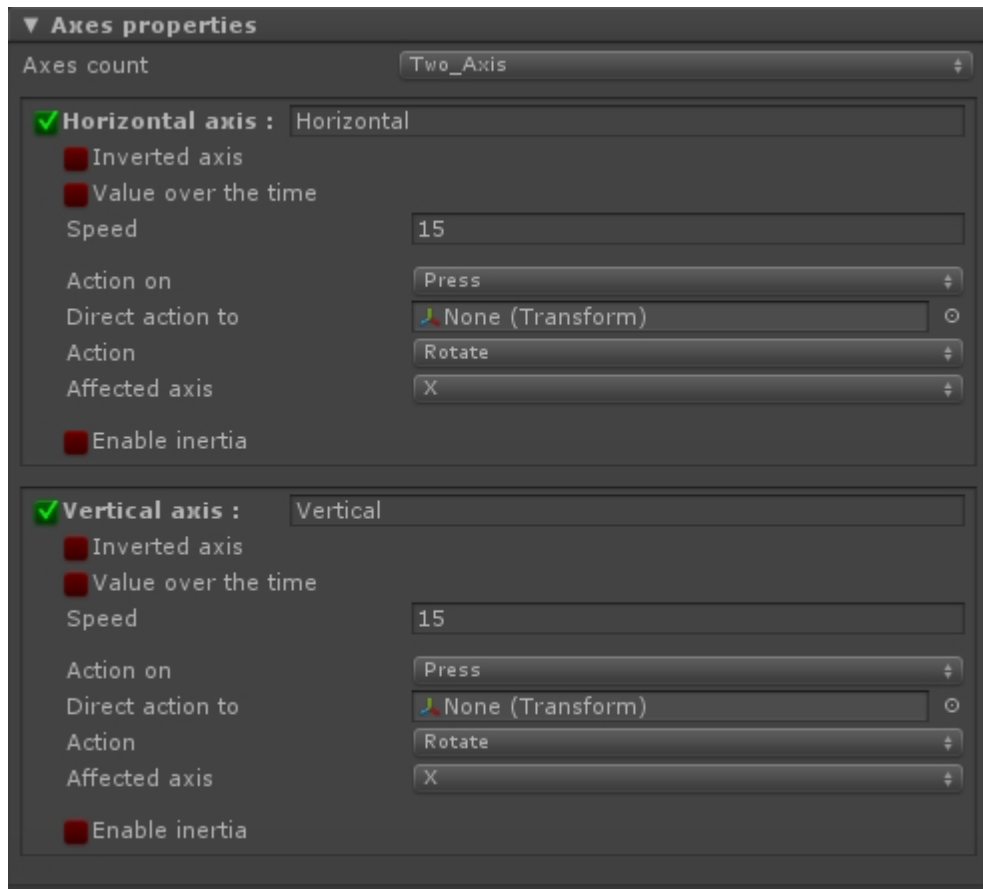


## Size

DPad size, you can uses the scene view to setup the size

## Axes properties

# Axes properties



### Axes count

Set the number of axes manage by the DPad

### Horizontal axis & Vertical axis

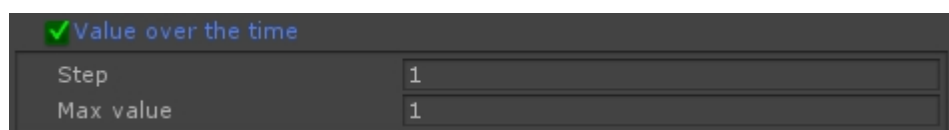
Enables or disables the axis, followed by his name for the input manager. The name of an axis must be unique for a scene

### Inverted axis

Reverse the axis

### Value over the time

To switch axis mode in over the time



#### Step

The increment.

#### Max value

The absolute value that could reach the axis

## **Speed**

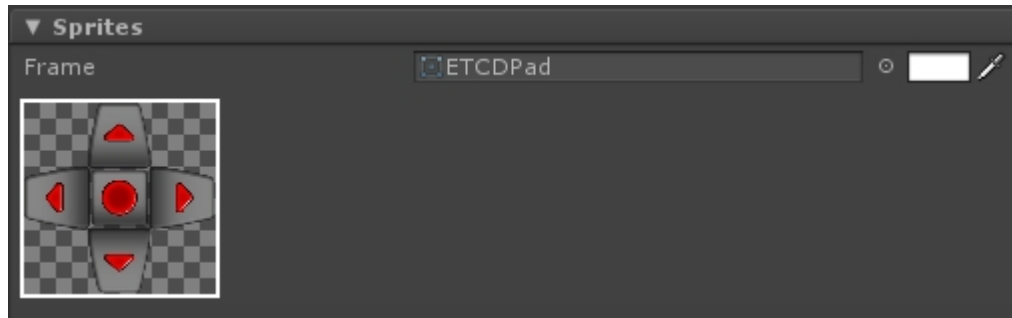
This value is used by direct mode to operate the action, and in the calculation of value returned by ETCInput.GetAxisSpeed. (Look at ETCInput\_API.PDF)

## **Direct Action Bloc**

[More detail here](#)

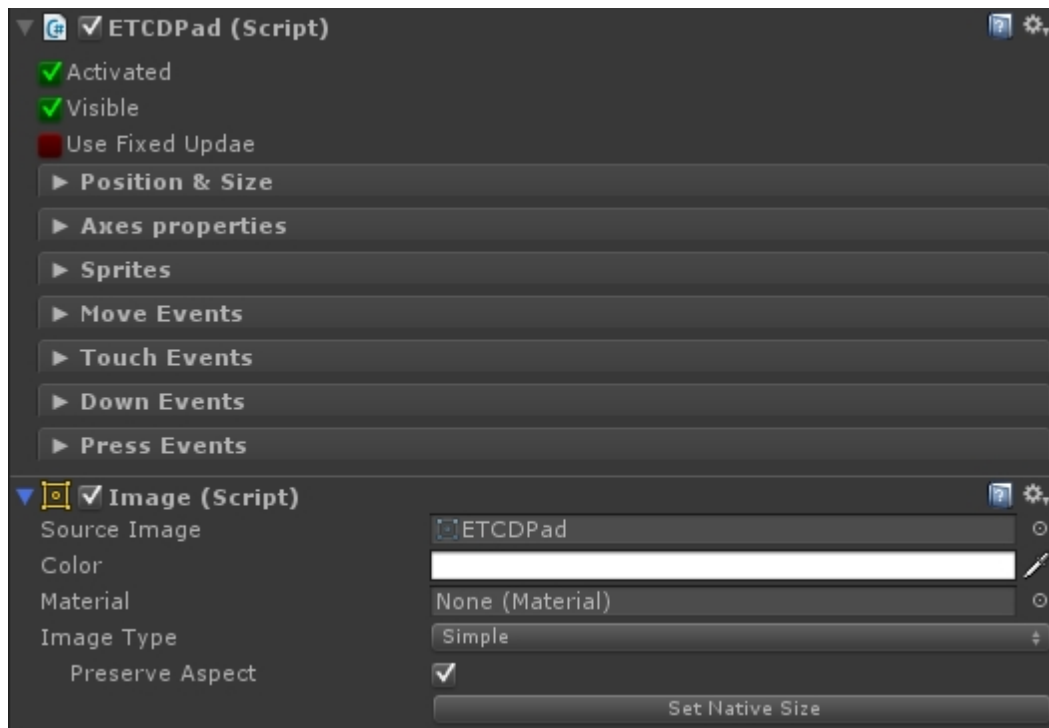
## Sprites

# Sprites



Sets the picture and color of the DPad.

You can also go directly to the parts of each image component

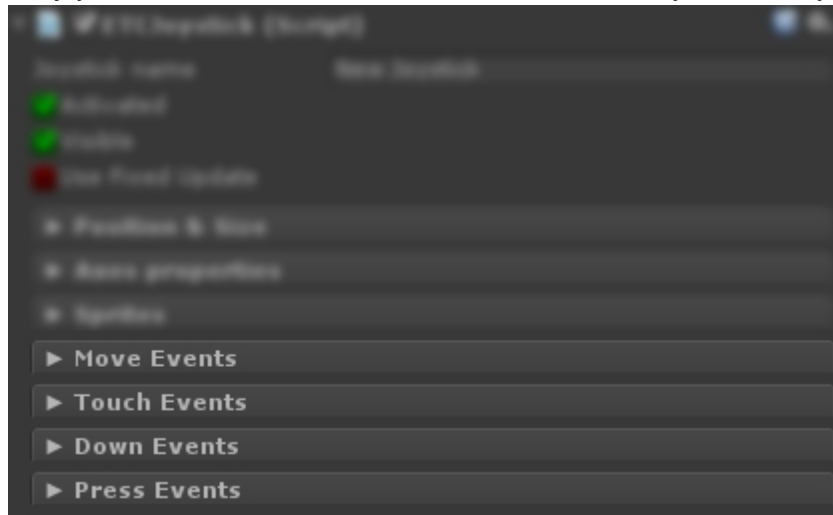


## Events

# Events

This part gives you access to all available events. Of course you can use only those that interests you.ETC uses the new event system that lets you easily use to call existing functions on scripts.

The joystick, DPad & TouchPad have the same event, they will be easy to replace a control by another one.



[More detail on all events here](#)



## Touchpad

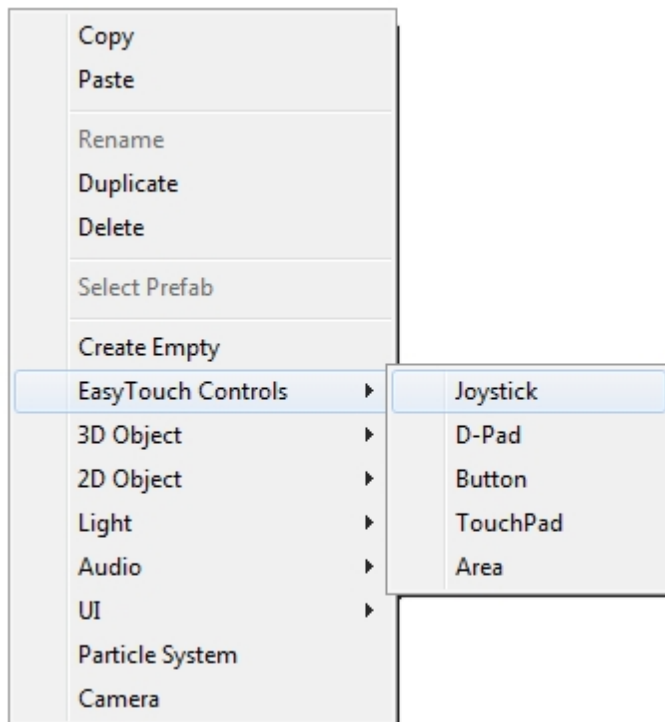
---

### Touchpad overview

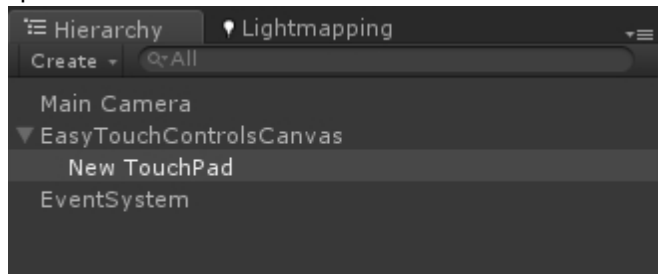
A touch is a simple area to detect user touch. Return values correspond to the delta position.

### Creating Touchpad

Right click in the hierarchy window => EasyTouch Controls => TouchPad



ETC will automatically create a canvas and an EventSystem gameobject. They will be set for optimum operation of ETC.



## Touchpad Inspector

# Touchpad Inspector



### Activated

Active or not the touchpad, it is visible when disabled

### Visible

Displays or not the Touchpad

### Use Fixed Update

Enable this option if you use the physical.

### Position & Size

Sets touchpad position

### Axes properties

Settings axes

### Sprites

Sets the touchpad images

### Move Events

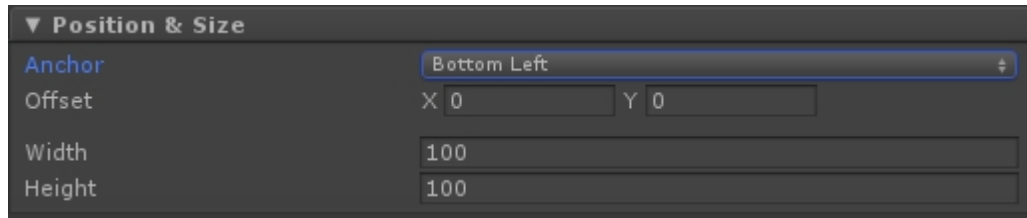
### Touch Events

### Down Events

### Press Events

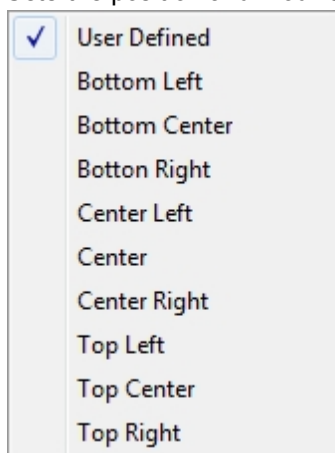
## Position & Size

# Position & Size



## Anchor

Sets the position and modified the DPad anchors.

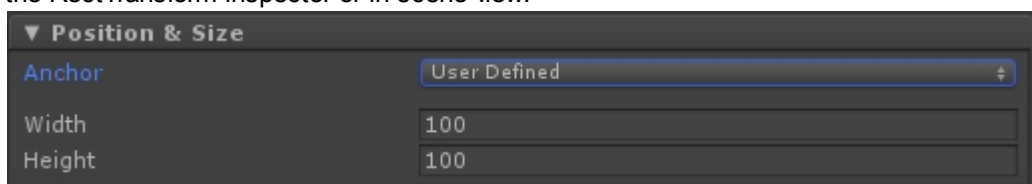


## Offset

Defines the offset to be applied with respect to the selected anchor.

## User Defined

By choosing this option, you can position the Touchpad manually and define yourself anchoring with the RectTransform inspector or in scene view.



## Width

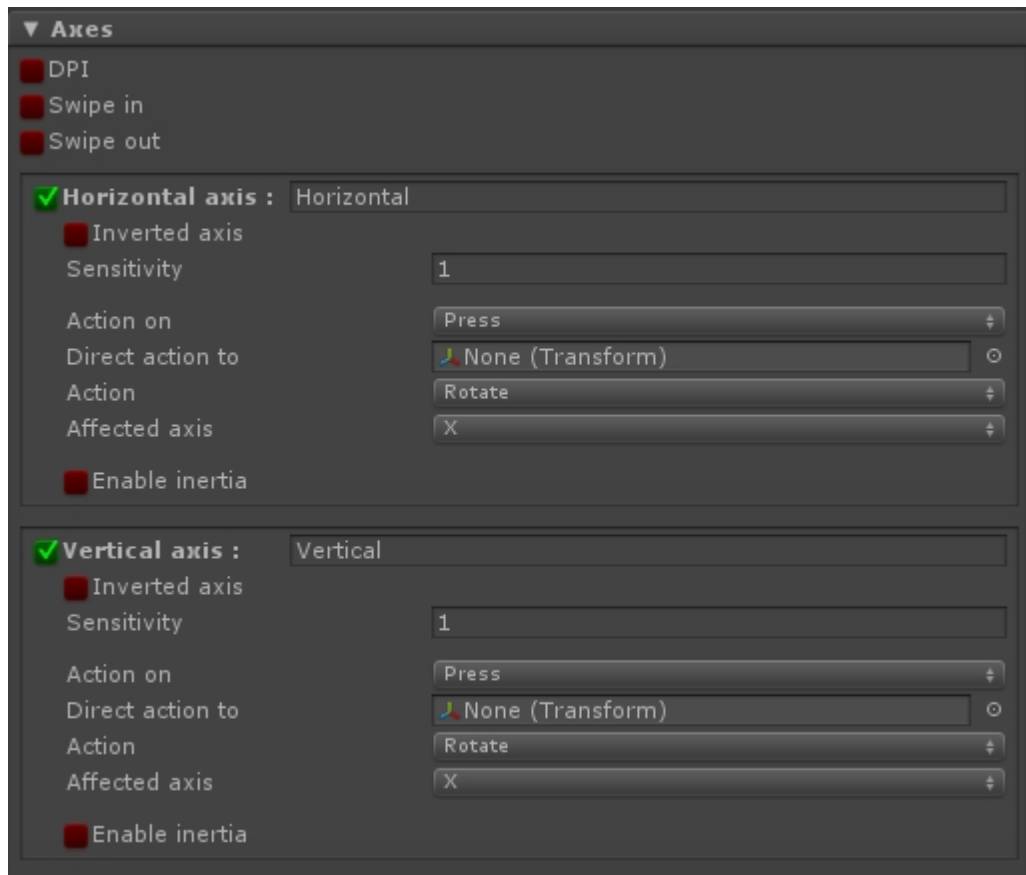
Touchpad width, you can uses the scene view to setup the size

## Height

Touchpad height, you can uses the scene view to setup the size

## Axes properties

# Axes properties



### DPI

It will take into account the screen resolution, so that the result is the same regardless of the size of the screen.

### Swipe In

Swipe in allows you to use the touchpad even if the start touch occurs outside the control and slid over him

### Swipe Out

Swipe Out allows you to use the touchpad even if the current touch position isn't over him but the touch start occurs over him

### Horizontal axis & Vertical axis

Enables or disables the axis, followed by his name for the input manager. The name of an axis must be unique for a scene

### Inverted axis

Reverse the axis

### Sensitivity

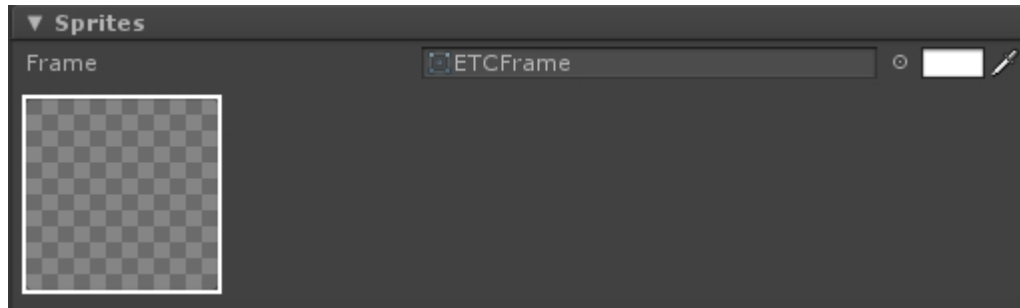
The touchpad sensitivity.

## **Direct Action Bloc**

[More detail here](#)

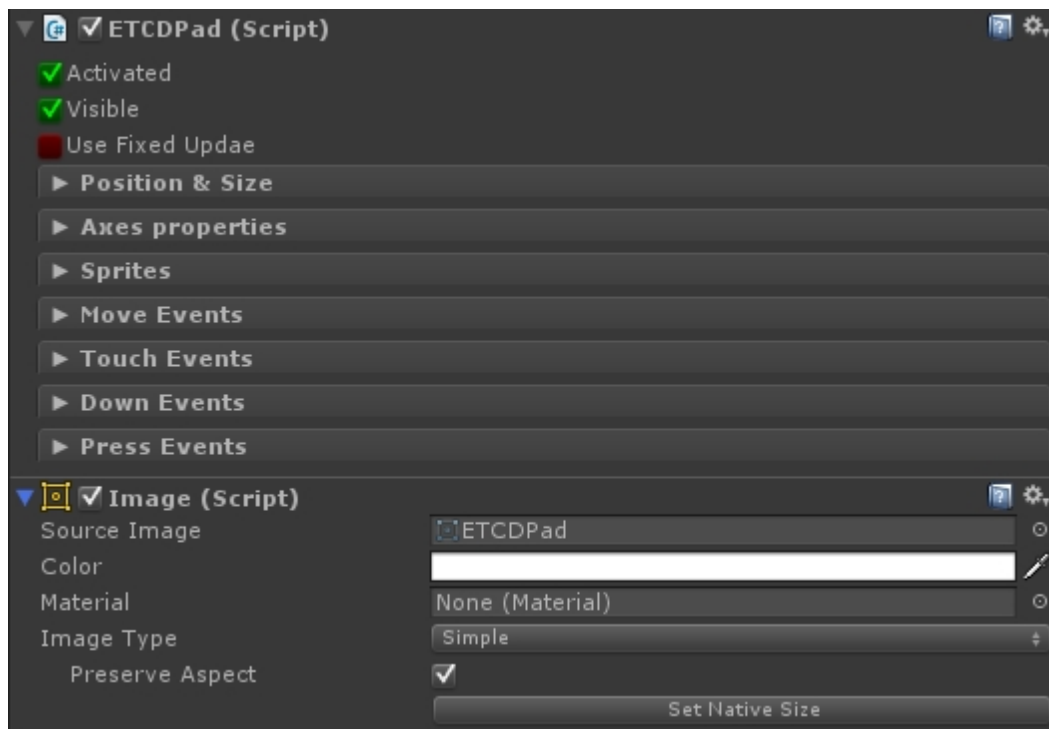
## Sprites

# Sprite



Sets the picture and color of the touchpad

You can also go directly to the parts of each image component

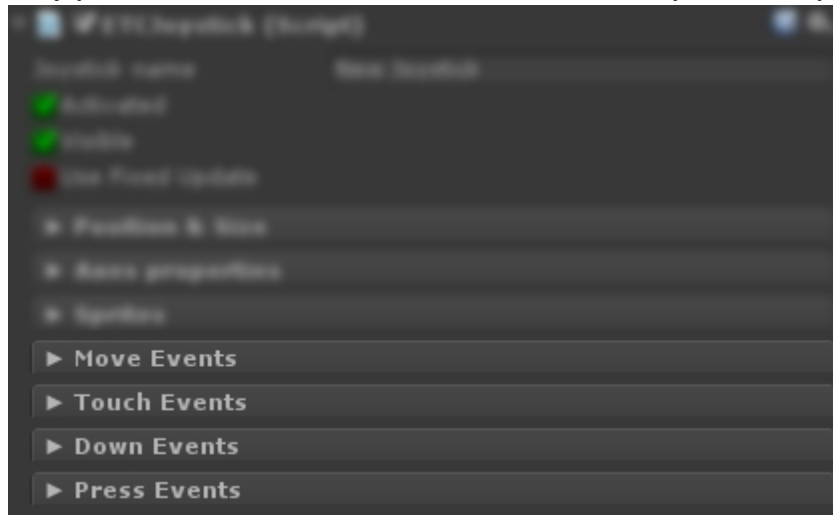


## Events

# Events

This part gives you access to all available events. Of course you can use only those that interests you.ETC uses the new event system that lets you easily use to call existing functions on scripts.

The joystick, DPad & TouchPad have the same event, they will be easy to replace a control by another one.



[More detail on all events here](#)

## Button

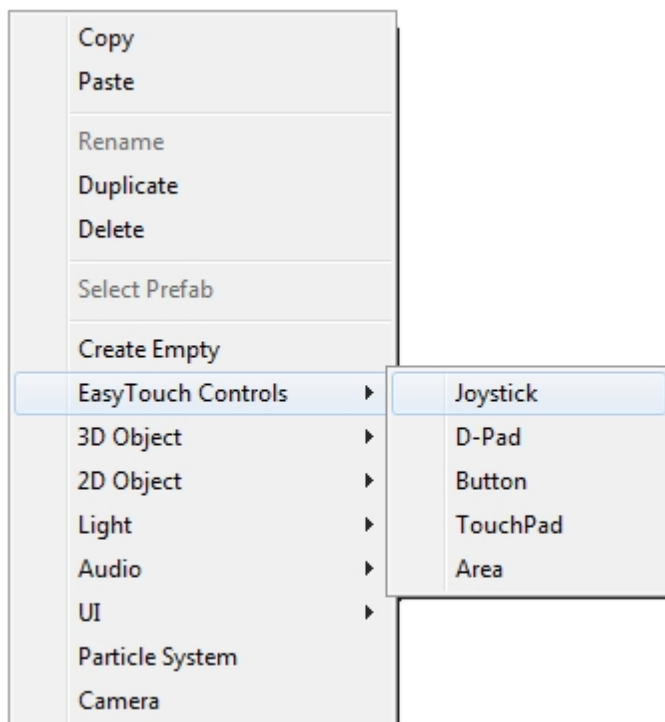
# Button Overview

ETC manages two types of button:

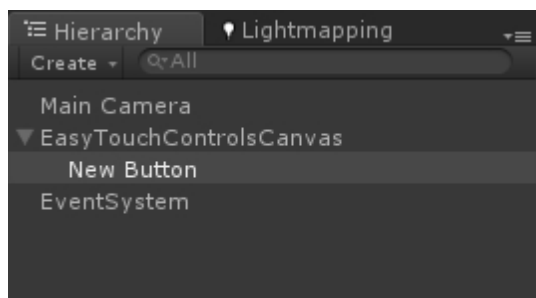
- **Classical:** The values return by the button are On/Off 0 or 1.
- **Over the time:** The values return by the button are relative to a step value over the time.

# Creating Button

Right click in the hierarchy window => EasyTouch Controls => button



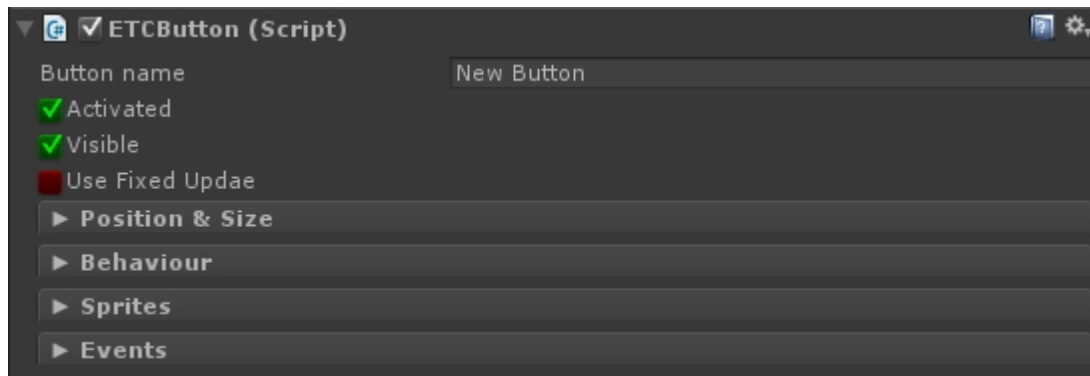
ETC will automatically create a canvas and an EventSystem gameobject. They will be set for optimum operation of ETC.



## Button Inspector



# Button Inspector



## Activated

Active or not the touchpad, it is visible when disabled

## Visible

Displays or not the Touchpad

## Use Fixed Update

Enable this option if you use the physical.

## Position & Size

Sets button position

## Behaviour

Settings button

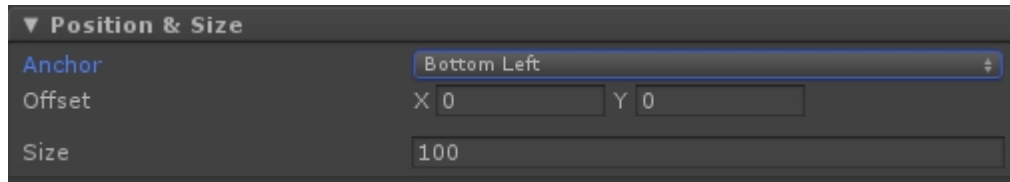
## Sprites

Sets the button images

## Events

## Position & Size

# Position & Size



## Anchor

Sets the position and modified the button anchors.

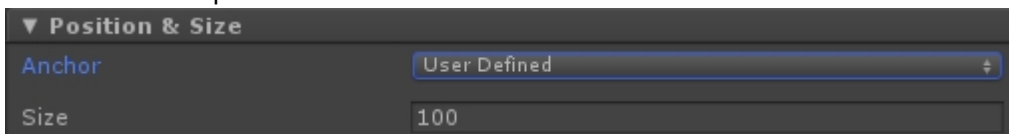


## Offset

Defines the offset to be applied with respect to the selected anchor.

## User Defined

By choosing this option, you can position the button manually and define yourself anchoring with the RectTransform inspector or in scene view.

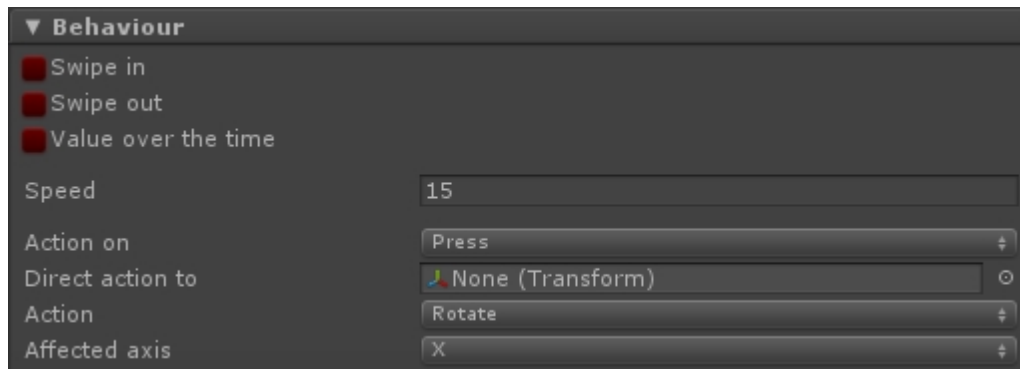


## Size

Button size, you can uses the scene view to setup the size

## Behaviour

## Behaviour

**Swipe In**

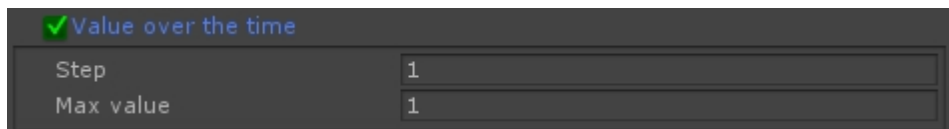
Swipe in allows you to use the button even if the start touch occurs outside the control and slid over him

**Swipe Out**

Swipe Out allows you to use the button even if the current touch position isn't over him but the touch start occurs over him

**Value over the time**

To switch button mode in over the time

**Step**

The increment.

**Max value**

The absolute value that could reach the axis

**Speed**

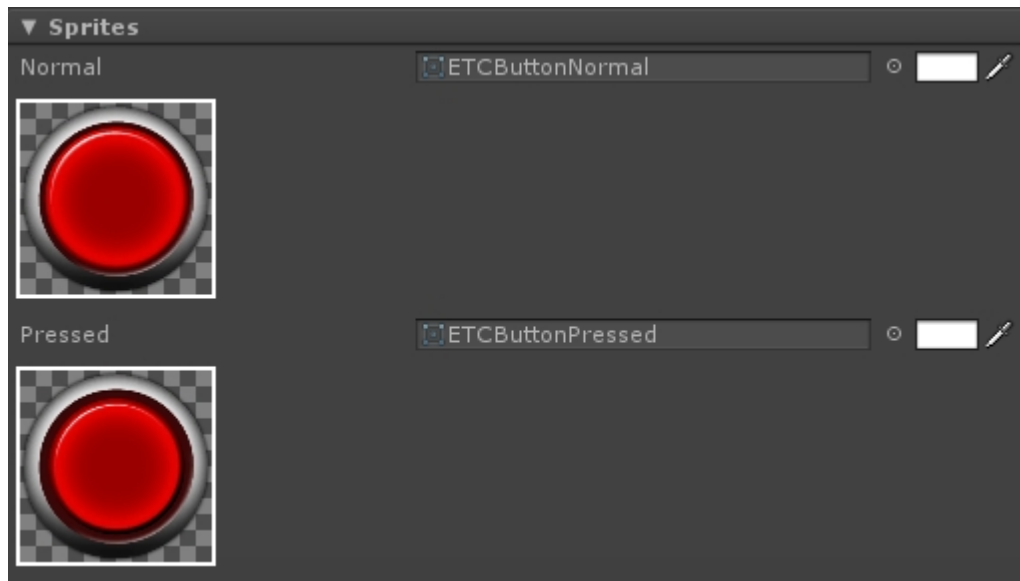
This value is used by direct mode to operate the action, and in the calculation of value returned by ETCInput.GetButtonValue. (Look at ETCInput\_API.PDF)

**Direct Action Bloc**

[More detail here](#)

## Sprites

# Sprites



Sets the picture and color of the different state of the button.

## Events

# Events



### **On Down()**

Called when a user press down the button for the first time

### **On Pressed(Vector2)**

Called while a user press the button.

### **On Pressed Value(Single)**

Called while a user press the button.

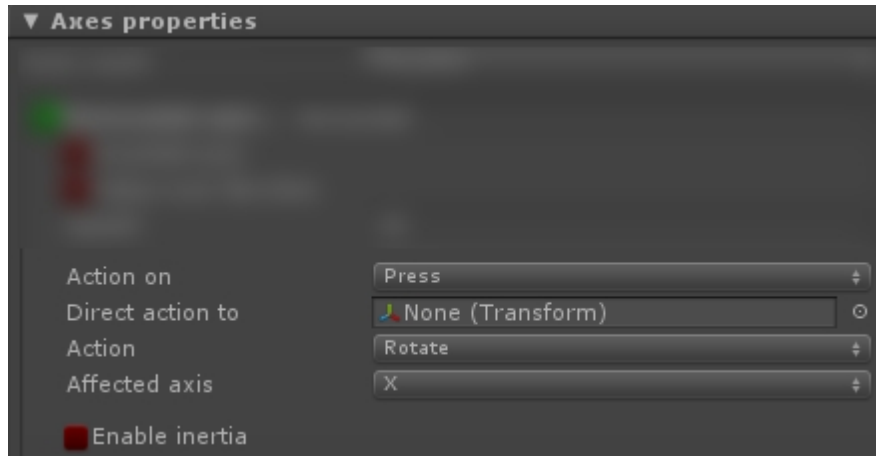
### **On Up()**

Called when a user release the button

## Direct Action Inspector

# Direct Action Inspector

These properties are similar to the axes of the joystick, DPad & TouchPad. They help define the direct action that will be applied



### Action on

To determine when the action is to take place (Press by default).



**Down** = Only when the axis is used for the first time (Reset when the axis back to 0)

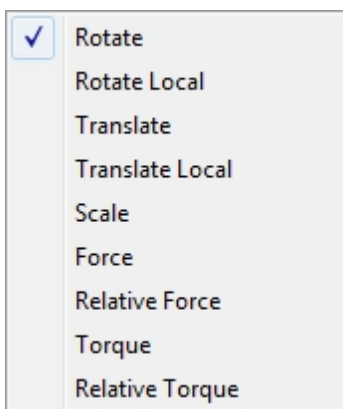
**Press** = while the axis is in motion

### Direct action to

The transform of the object that is acted.

### Action

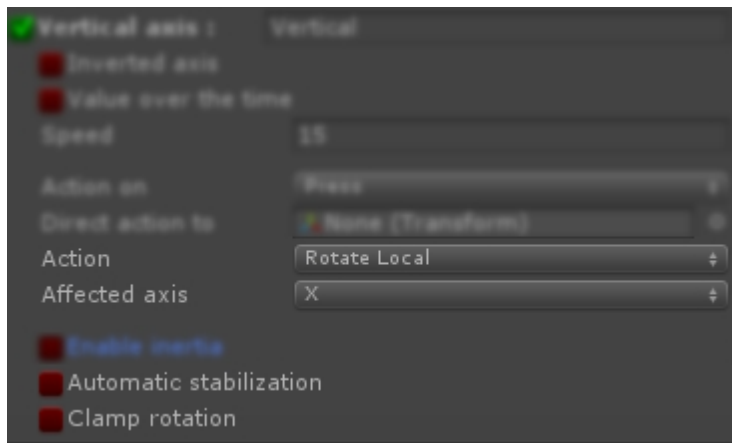
The action to be applied to the object



The action **Rotate Local** brings new following options

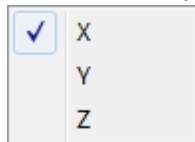
\* [Automatic stabilization](#)

\* [Clamp rotation](#)



### Affected axis

The axis will support action.



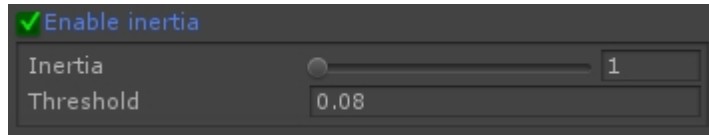
### Enable inertia

Enables / disables inertia : more detail [here](#)

## Inertia

# Inertia

Adds inertia to the axis. The axis will achieve its value relative to its position gradually



### Inertia

The higher the value is, the greater the inertia effect is important.

### Threshold

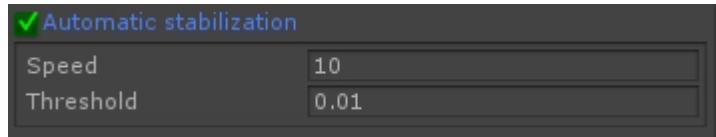
This value allows to determine the threshold below which the axis will be reset to the 0 position when inertia is enabled.



## Automatic stabilization

# Automatic stabilization

It allows the object to return its original rotation when you released the axis.



A screenshot of a settings panel for 'Automatic stabilization'. The panel has a dark background. At the top, there is a green checkmark followed by the text 'Automatic stabilization'. Below this, there are two input fields. The first is labeled 'Speed' and contains the value '10'. The second is labeled 'Threshold' and contains the value '0.01'.

Parameter	Value
Speed	10
Threshold	0.01

### Speed

Set the auto-stabilization speed

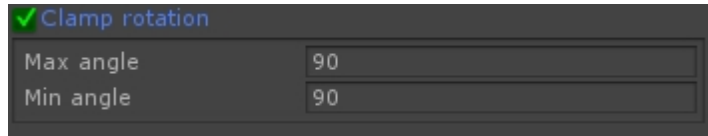
### Threshold

This value allows to determine the threshold below which the axis will be reset to the 0 position

## Clamp rotation

# Clamp rotation

Clamp the rotation relative to min & max angle.



✓ Clamp rotation	
Max angle	90
Min angle	90

### Horizontal Axis

min = limit angle to the left

max = limit angle to the right

### Vertical Axis

min = limit angle up

max = limit angle down

## Events : Joystick-DPad-TouchPad

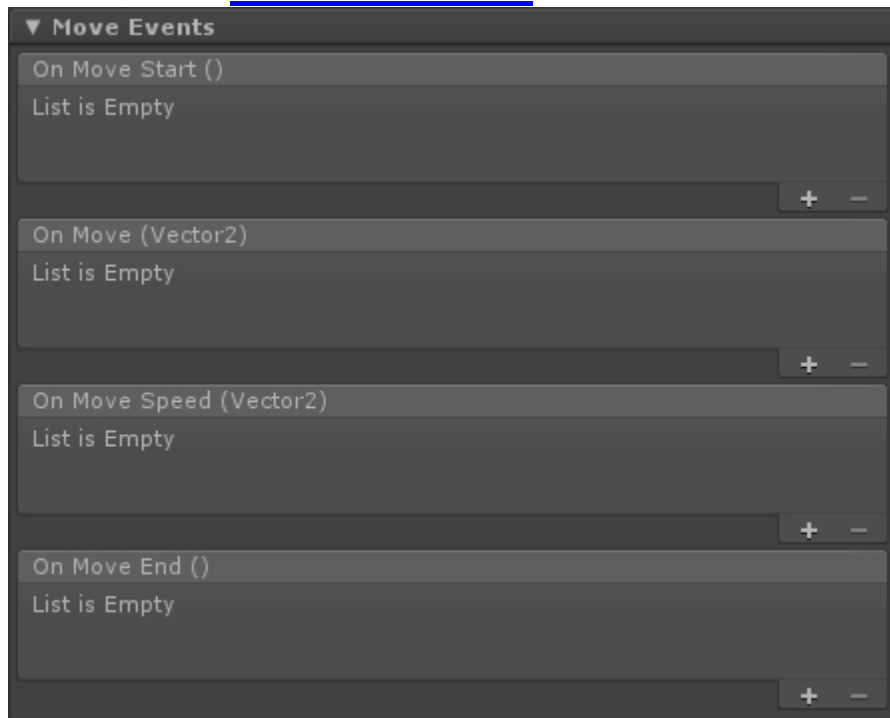
---

# Events

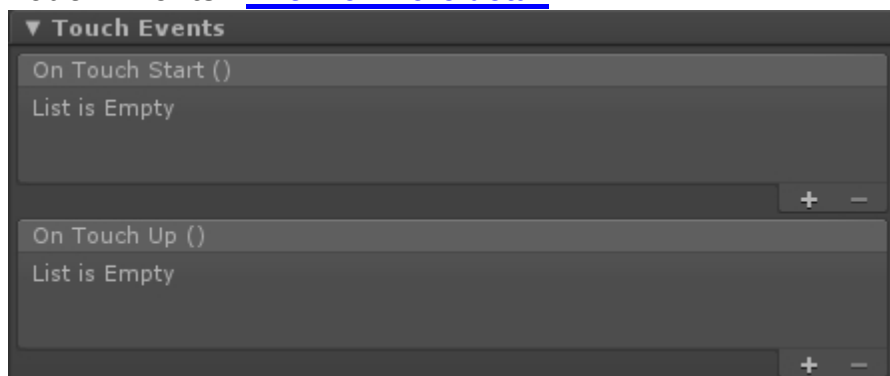
The joystick DPad & TouchPad have the same event, they will be easy to replace a control by another.

[How to add event, click here](#)

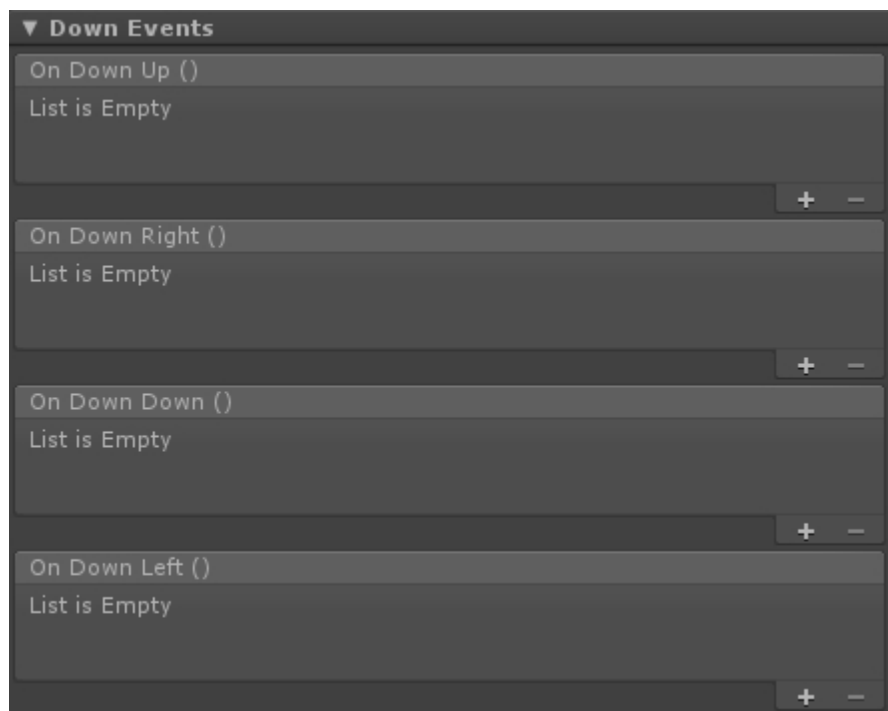
**Move Events :** [Click for more detail](#)



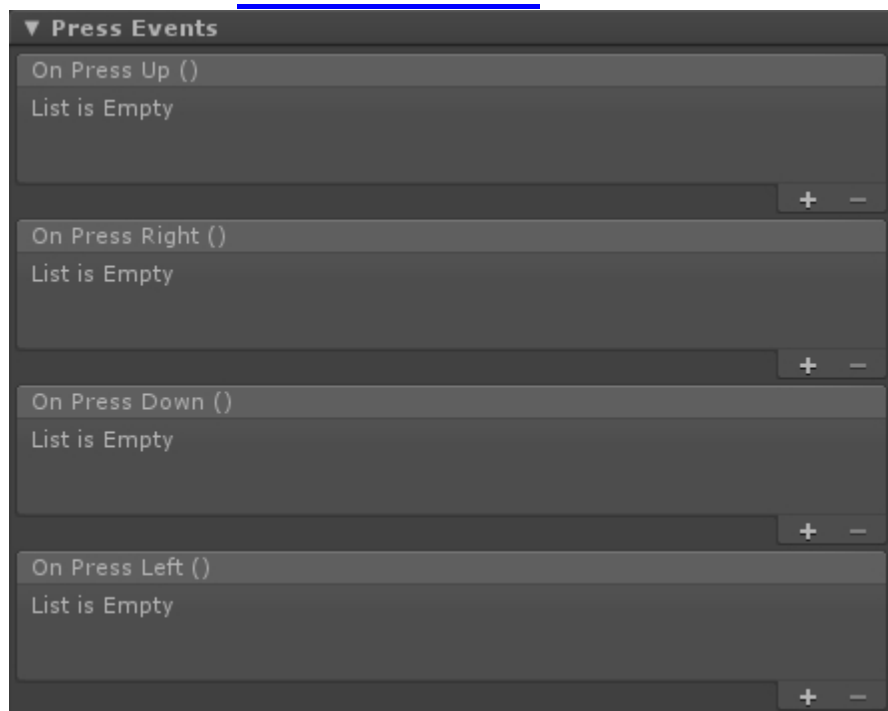
**Touch Events :** [Click for more detail](#)



**Down Events :** [Click for more detail](#)

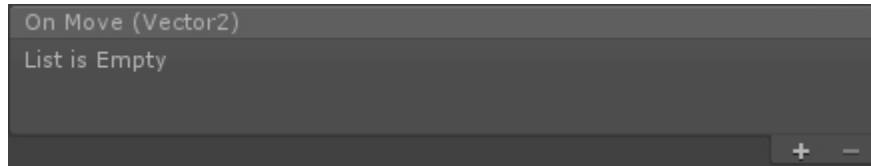


**Press Events :** [Click for more detail](#)

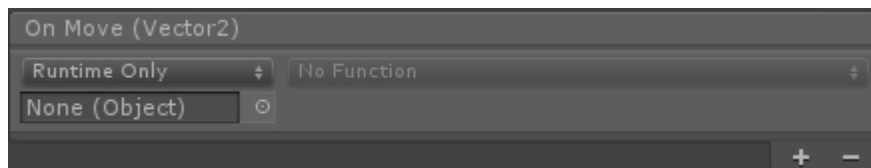


## Add Events

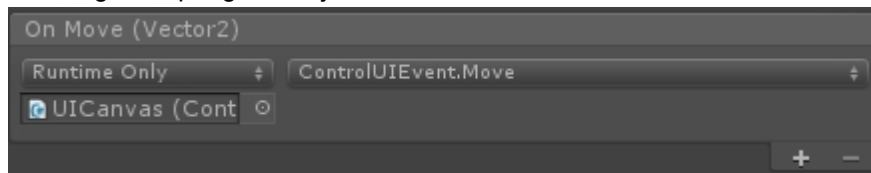
### Add events



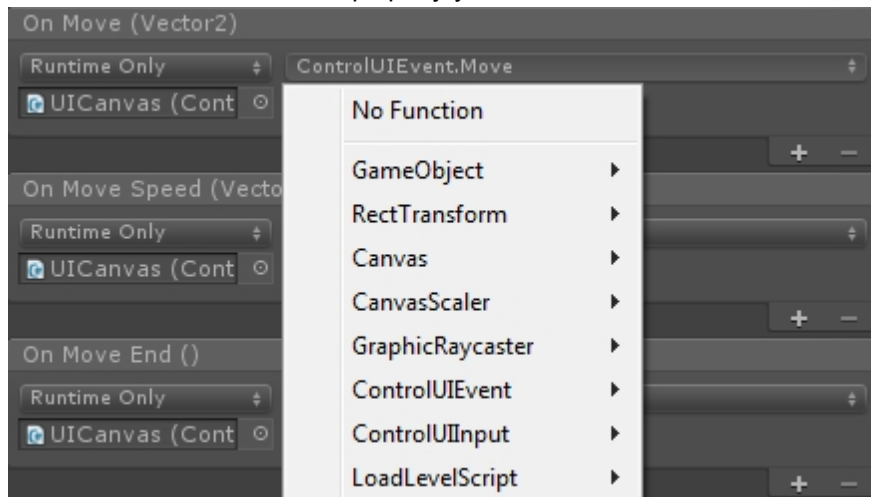
- Click on +
- A new line is created



- Drag & drop a game object



- Select the function or the property you want to call

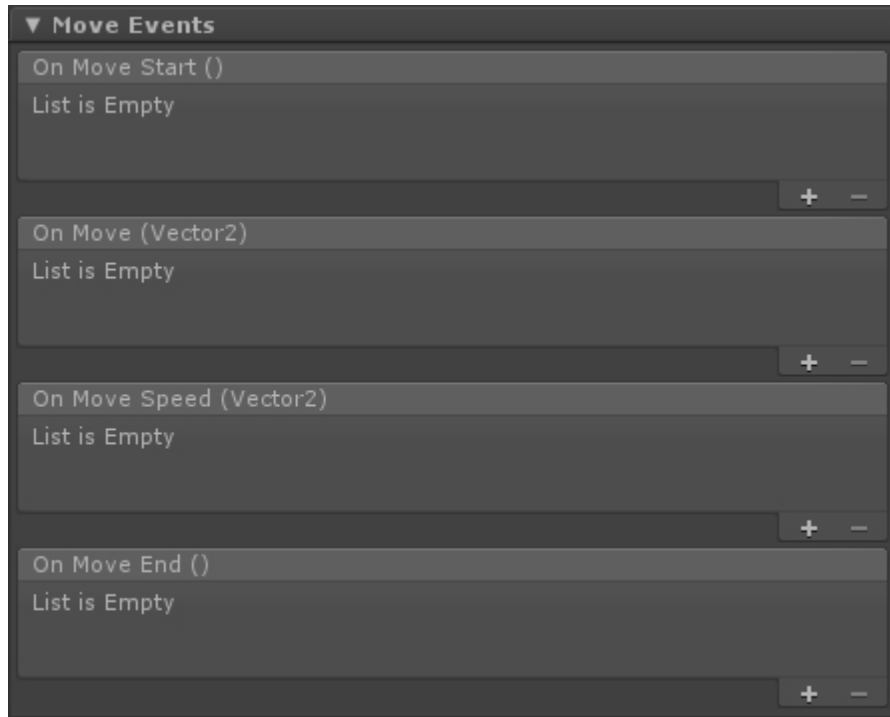


- You can add as many objects as you like

•

## Move Events

# Move events



### On Move start()

Called when the user moves the axis for the first time (reset when back to 0).

### On Move(Vector2)

Called while a user move the axis or don't back to 0. The axis value (-1..1) is passed to the responding function as a Vector2 type.

### On Move Speed(Vector2)

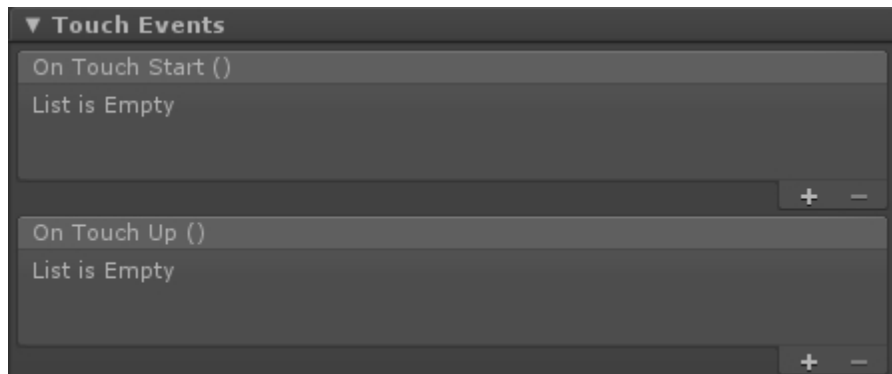
Called while a user move the axis or don't back to 0. The axis value (axisValue(-1..1) \* axis sensitivity \* Time.DeltaTime) is passed to the responding function as a Vector2 type.

### On Move End()

Called when axis back to 0 or when the user releases the joystick.

## Touch Events

# Touch Events



### On Touch Start

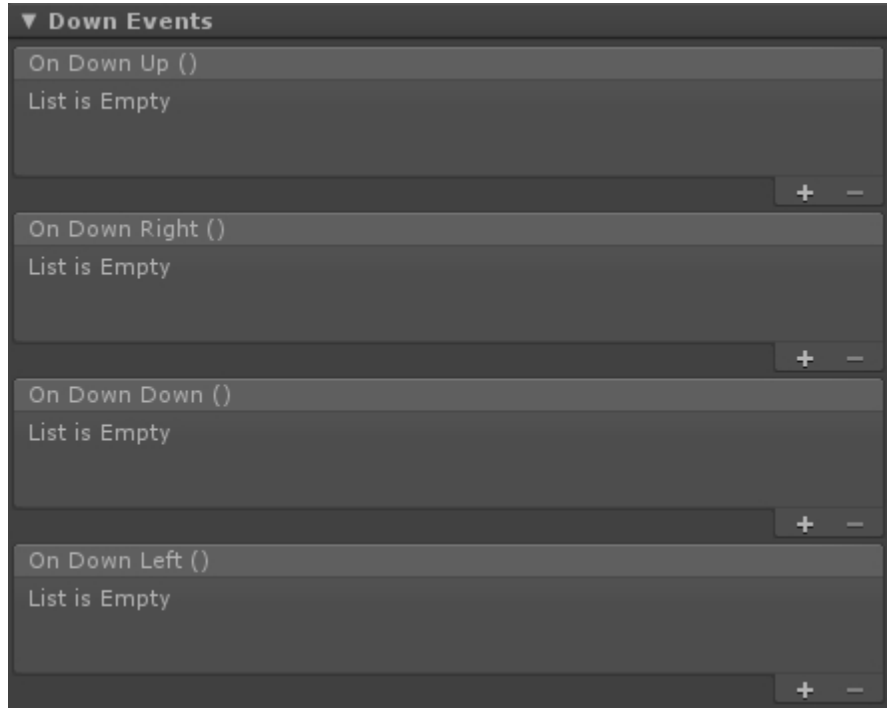
Called when the user touch the joystick for the first time.

### On Touch Up

Called when the user releases the joystick.

## Down Events

# Down Events



### On Down Up

Called when a user move up the axis the first time (Reset when back to 0 or axis value< threshold for a joystick)

### On Down Right

Called when a user move right the axis the first time (Reset when back to 0 or axis value< threshold for a joystick)

### On Down Down

Called when a user move down the axis for the first time (Reset when back to 0 or axis value< threshold for a joystick)

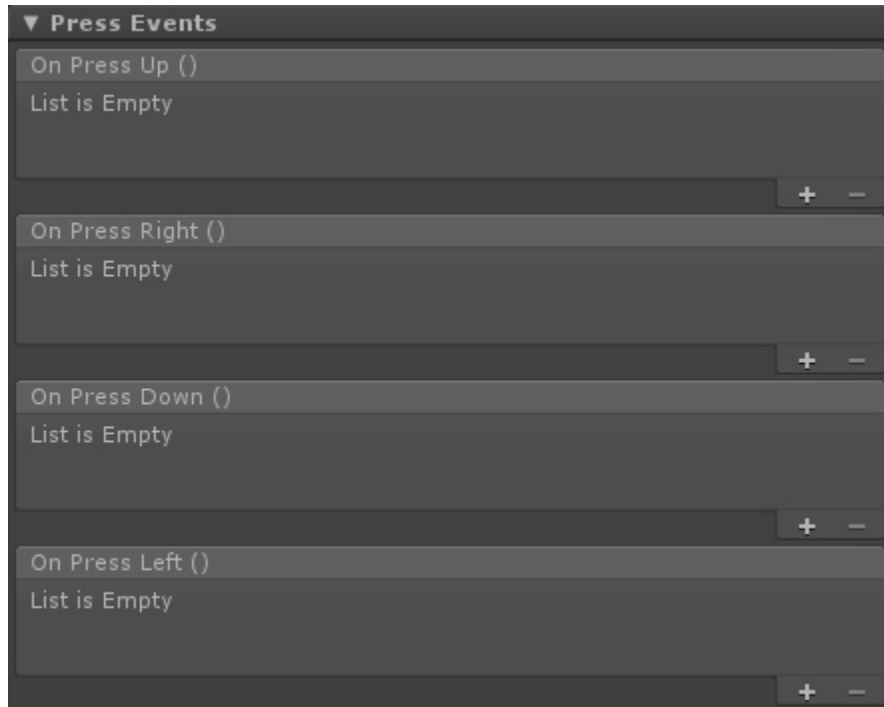
### On Down Left

Called when a user move left the axis the first time (Reset when back to 0 or axis value< threshold for a joystick)



## Press Events

# Press Events



### On Press Up

Called while a user move up the axis.

### On Press Right

Called while a user move right the axis.

### On Press Down

Called while a user move down the axis.

### On Press Left

Called while a user move left the axis.

