

# EasyTouch 4.X



## User & API documentation

## Index

---

What's new .....	3
What is EasyTouch .....	4
Quick Start .....	5
Quick Start with auto-selection .....	7
EasyTouchTrigger .....	9
How to use ETT .....	10
EasyTouchTrigger Inspector .....	12
EasyTouch Inspector .....	13
GUI Compatibility .....	14
Automatic selection .....	15
General gesture properties .....	16
Two fingers gesture properties .....	17
Second finger simulation .....	18
Script Reference .....	19
Classes .....	20
EasyTouch .....	20
Events .....	22
Gesture Class .....	24
fingerIndex .....	26
touchCount .....	27
startPosition .....	28
position .....	29
deltaPosition .....	30
actionTime .....	31
deltaTime .....	32
swipe .....	33
swipeLength .....	34
swipeVector .....	35
deltaPinch .....	36
twistAngle .....	37
twoFingerDistance .....	38
pickedObject .....	39
pickedCamera .....	40
isGuiCamera .....	41
isOverGui .....	42
pickedUIElement .....	43
GetTouchToWorldPoint .....	44
GetSwipeOrDragAngle .....	45
NormalizedPosition .....	46
GetCurrentPickedObject .....	47
IsOverUIElement .....	48
IsOverRectTransform .....	49
GetCurrentFirstPickedUIElement .....	50

## What's new in EasyTouch 4

- Unity UI integration.
- [EasyTouchTrigger](#) component, that allow you to call existing functions with out a line of code relative to an EasyTouch event.
- New two finger gesture algorithm.
- The simulation of the second finger can be activated or not
- 2 methods for two fingers auto selection Finger & Average
- The double tap is now configurable.
- Gesture priority between tap & slips.
- 4 new build in swipe direction : UpLeft, UpRight, DownLeft , DownRight
- Auto update picked object option
- tmpArray of touches is now cached to prevent garbage collector.
- New inspector style

## features no longer supported

- EasyJoystick & EasyButton are no longer supported, they are replaced by Easy Touch Controls
- Reserved area are no longer supported, the new Unity UI is more flexible
- Message sending via the internal management system in Unity for javascript user has been replaced by the component [EasyTouchTrigger](#).

## What is EasyTouch

EasyTouch allow you to quickly and easily develop actions based on a touchscreen or a mouse gesture. All major gestures are recognized such as touch, tap, double tap, twist, pinch.. and much more.

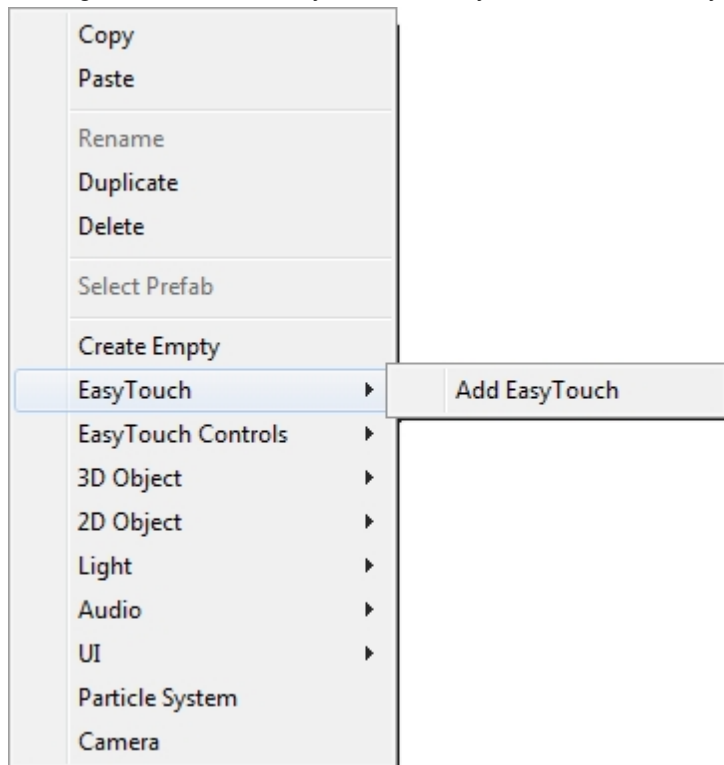
EasyTouch allows automatic detection of all gameobjects with collider on layers that you can set. To allow easy handling of the various elements that make up your scene

EasyTouch is written in C# , it notifies you of an action on the touch screen or with mouse by firing events. The events are sent by delegate system for C #, or you can use EasyTouchTrigger component if you are not familiar with events.

Each event passes a parameter class, with all the informations about the current action (position, angle, picked object ...)

## Quick start

- Import EasyTouch package
- Right click on hierarchy view => EasyTouch => Add EasyTouch



- Create a new C# script MyFirstTouch
- Add these methods

```
// Subscribe to events
void OnEnable(){
    EasyTouch.On_TouchStart += On_TouchStart;
}

// Unsubscribe
void OnDisable(){
    EasyTouch.On_TouchStart -= On_TouchStart;
}

// Unsubscribe
void OnDestroy(){
    EasyTouch.On_TouchStart -= On_TouchStart;
}

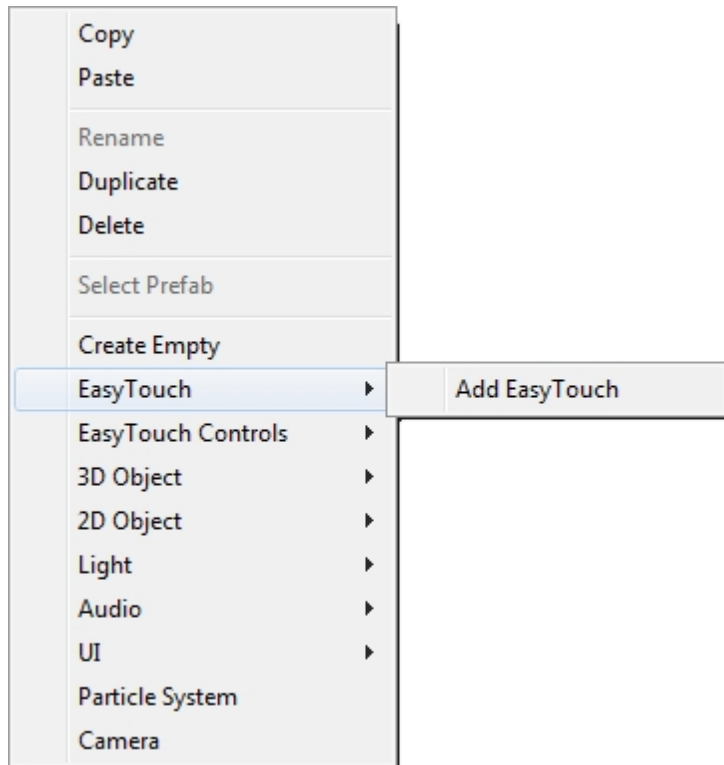
// Touch start event
public void On_TouchStart(Gesture gesture){
    Debug.Log( "Touch in " + gesture.position);
}
```

- Create an empty gameObject, and name it Receiver.

- Add MyFirstTouch script to the gameObject Receiver.
- Run it in editor, and click on the screen

## Quick start with auto-selection

- Import EasyTouch package
- Right click on hierarchy view => EasyTouch => Add EasyTouch



- Create a new C# script
- Add these methods

```
// Subscribe to events
void OnEnable(){
    EasyTouch.On_TouchStart += On_TouchStart;
}

// Unsubscribe
void OnDisable(){
    EasyTouch.On_TouchStart -= On_TouchStart;
}

// Unsubscribe
void OnDestroy(){
    EasyTouch.On_TouchStart -= On_TouchStart;
}

// At the touch beginning
public void On_TouchStart(Gesture gesture){

    // check that we have touched the sphere
```

```
if (gesture.pickedObject == gameObject)
    gameObject.renderer.material.color = new Color( Random.Range(0.0f, 1.0f),
Random.Range(0.0f, 1.0f), Random.Range(0.0f, 1.0f));
}
```

- Create a sphere and assign it a simple diffuse material
- Add script on the sphere
- Run it in editor, and click on the screen



## EasyTouchTrigger

EasyTouchTrigger is a new component that allow you to easily & quickly use EasyTouch. It can be used instead or in addition to the conventional method by C# event.

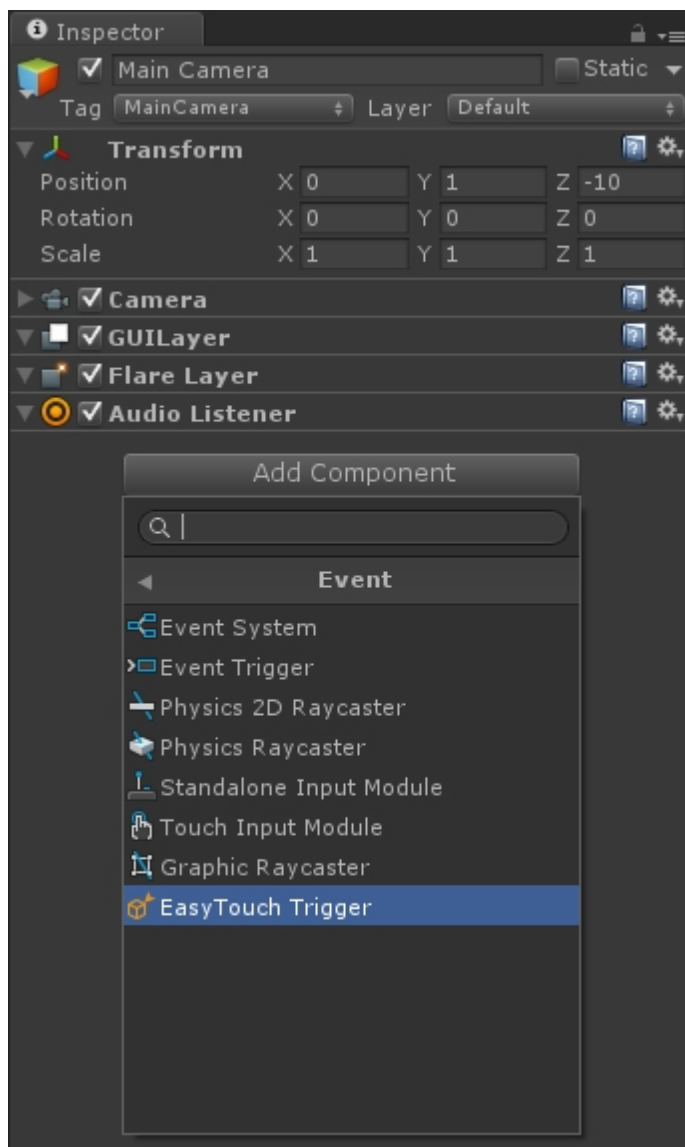
This component allows you to call functions on GameObjects according EasyTouch events without writing a line of code.

### How does it work?

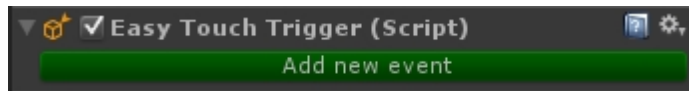
ETT allows you to define the object which will receive a gesture and the object on which you want to execute a function. You still have to specify the function name, and if necessary the parameter to send.

## How to use ETT ([Look at video on Youtube](#))

- Select the gameobject to interact with the touch screen, or one that will serve as a scheduler. You can add as many as you like EasyTouchTrigger.
- Add ETT component with Add Component button => Event => EasyTouch Trigger

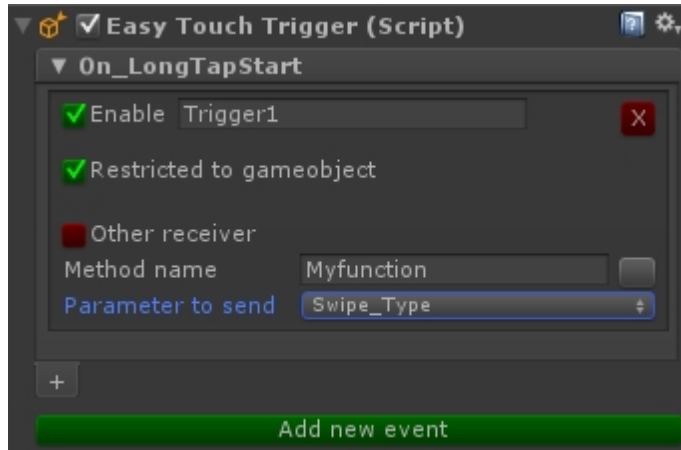


- The component is now active on your gameobject



- You can now add EasyTouch event by clicking on Add new event. A context menu will display all available events

## EasyTouchTrigger Inspector



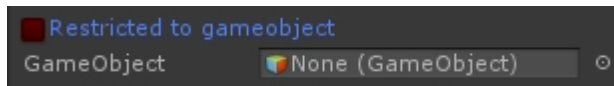
### Enable + Trigger Name

Enables or disable this trigger

### Restricted to gameobject

**Enable** : The beginning of the gesture must be on the current gameobject to trigger the action.

**Disable**



#### GameObject

Sets the selectable gameobject which will receive the start of the gesture to trigger the action.

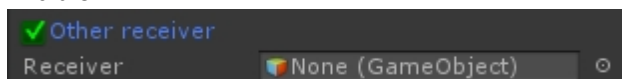
**Null** : The action will be triggered in all cases

### Other receiver

**Disable**

The message will be sent to the gameobject current.

**Enable**



Sets the gameobject which will receive the message

### Method name

The public function to call

### Parameter to send

The property of the gesture class to send

## EasyTouch Inspector



### Enable EasyTouch

Enables or disables EasyTouch. No events will be fired if EasyTouch is disabled

### Enable Unity Remote

Enabled this option if you want to test multi-finger on Unity Remote. (Don't forget to remove this flag before your final build)

### [GUI Compatibility](#)

Allow you to setup the behaviour of EasyTouch with NGUI & Unity UI

### [Automatic selection](#)

Allow you to setup the auto-selection.

### [General gesture properties](#)

Allow you to setup simple gesture

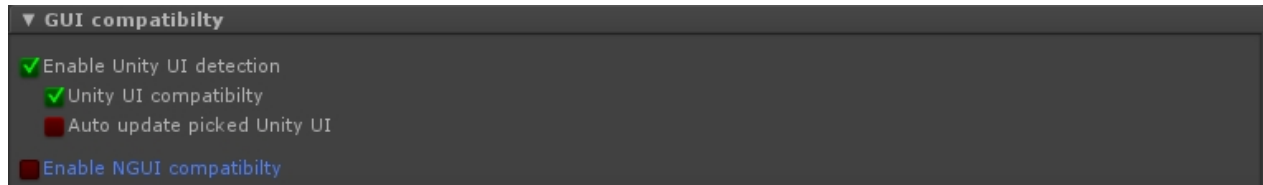
### [Two fingers gesture properties](#)

Allow you to setup two finger gesture

### Second finger simulation

Allow you to setup the second finger simulation

## GUI Compatibility



### Enable Unity UI detection

Enables or disables detection of selectable UI Element

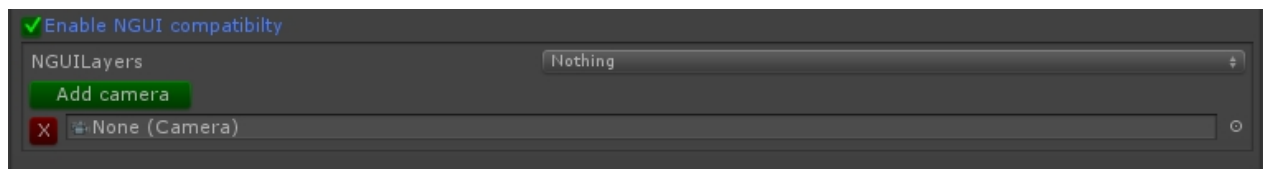
#### Unity UI compatibility

**Enable** : EasyTouch doesn't fire gesture events if touch is over a UIElement.

**Disable** : EasyTouch fires gesture events, and set [isOverUIElement](#) to true for [gesture](#) class

#### Auto update picked Unity UI

Enable : EasyTouch update UI data at each frame



### Enable NGUI Compatibility

Enable detection of NGUI object relative to NGUILayers & Camera. EasyTouch doesn't fire gesture event if touch is over NGUI Object.

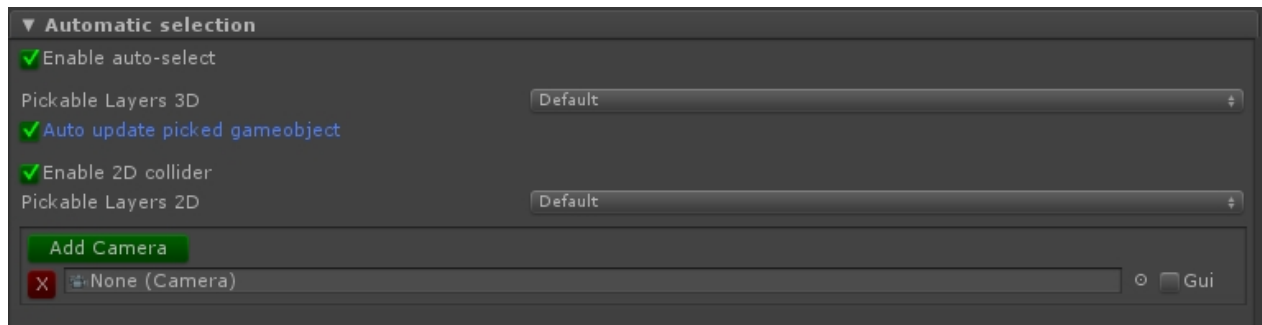
#### NGUI Layers

Layers mask for NGui object

#### Camera

All cameras for NGui detection

## Automatic selection



### Enable auto-select

enables 3D object detection under the touch relative to the setting.

### pickable layers 3D

Layer mask for selectable objects.

### Auto update picked gameobject

EasyTouch update picked object at each frame

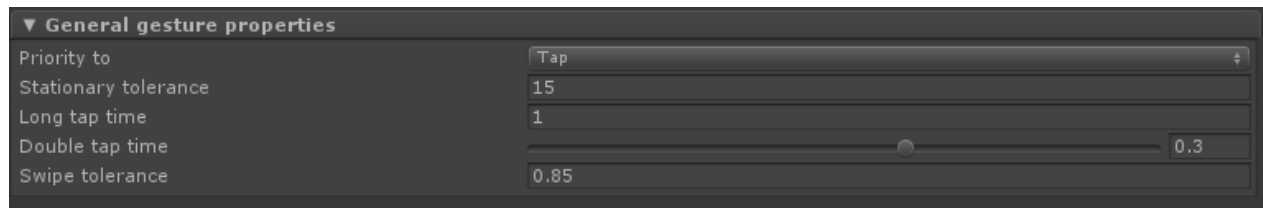
### Enable 2D collider

Enable 2Dcollider detection

### pickable 2D layers

Layer mask for selectable 2D objects.

## General gesture properties



### Priority to

You can choose between giving priority to tap gesture or slips gesture.

**Tap** : Stationary tolerance takes into account to detect a move from the current touch.

**Slips** : Stationary tolerance doesn't take into account to detect move. The motion is detect by Unity API (more reactive)

### Stationary tolerance

It's a diameter in pixel in which the key is considered stationary for Tap priority

### Long tap time

The time required to detect a long tap

### Double tap time

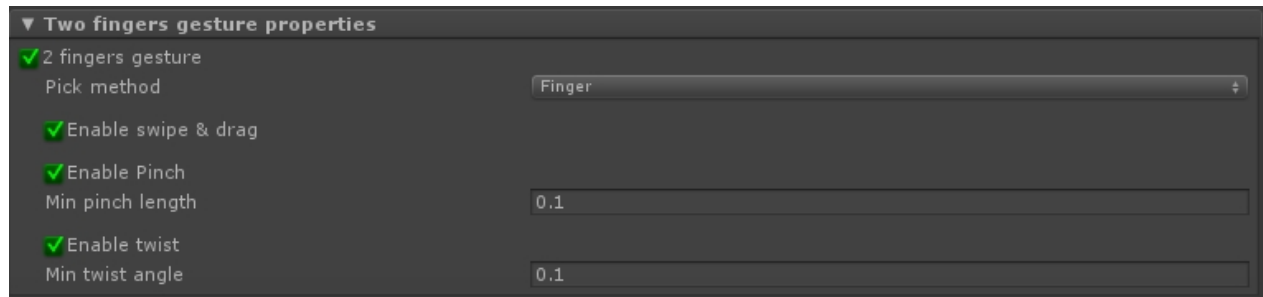
The maximum time to detect a double tap

### Swipe tolerance

The detection threshold between different build in swipe direction



## Two fingers gesture properties



### 2 Fingers gesture

Enables 2 fingers gesture

### Pick method

Can choose between 2 methods for auto-selection

**Finger** : the fingers must be over the object.

**Average** : It takes into account the average position to detect objet

### Enable swipe & drag

Enable swipe or drag gesture

### Enable Pinch

Enable pinch gesture

### Min pinch length

The minimum threshold distance in pixel for detecting a pinch

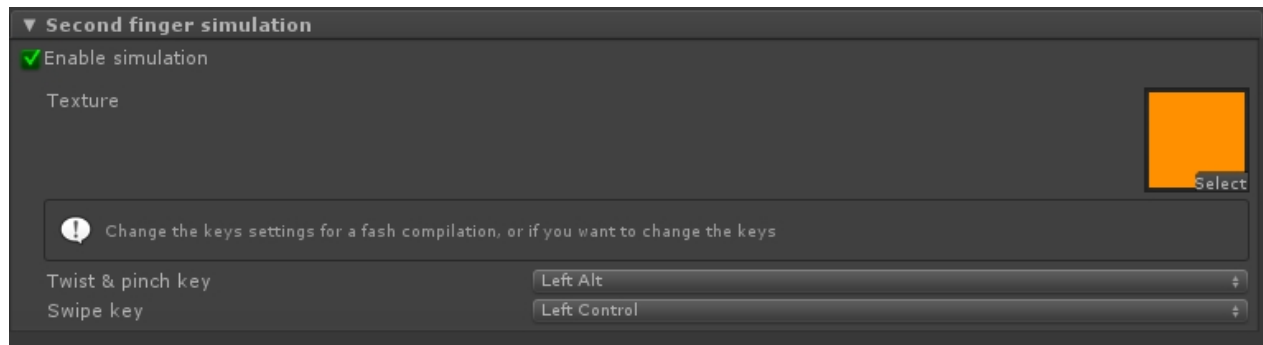
### Enable twist

Enable twist gesture

### Min twist angle

The minimum threshold angle for detecting a twist

## Second finger simulation



### Enable simulation

Enables second finger simulation

### Texture

The picture to display

### Twist & pinch key / Swipe key

Key to activate the simulation

## **Welcome to the EasyTouch Scripting Reference!**

This section of the documentation contains details of the scripting API that EasyTouch provides. To use this information, you should be familiar with the basic theory and practice of scripting in Unity.

# EasyTouch

## Enumerations

- GesturePriority** : Gesture priority **Tap, Slips**
- SwipeDirection** : Build in swipe direction **None, Left, Right, Up, Down, UpLeft, UpRight, DownLeft, DownRight, Other**
- TwoFingerPickMethod** : 2 fingers pick method **Finger, Average**

## Events

[Click here for all events](#)

## Static functions

- SetEnabled** : Enables or disables EasyTouch.
- GetEnabled** : Return true is EasyTouch is enabled.
- SetUICompatibility** : Enables or disables Unity UI compatibility.
- GetUIComptability** : Return true is Unity UI compatibility is enabled.
- SetAutoUpdateUI** : Enables or disables auto-update picked UI element.
- GetAutoUpdateUI** : Return true if auto-update UI element is enabled.
- SetEnableAutoSelect** : Enables or disables auto selection.
- GetEnableAutoSelect** : Return true if auto-selection is enabled
- SetAutoUpdatePickedObject** : Enables or disable auto-update picked object.
- GetAutoUpdatePickedObject** : Return true if auto-update picked object is enabled.
- Set3DPickableLayer** : Set layer mask for selectable object.
- Get3DPickableLayer** : Return the layer mask for selectable object.
- AddCamera** : Add a camera for selectable object.
- RemoveCamera** : Remove a camera.
- GetCamera** : Return a camera.
- SetEnable2DCollider** : Enables or disables 2D collider detection.
- GetEnable2DCollider** : Return true if 2D collider detection is enabled.
- Set2DPickableLayer** : Set layer mask for 2d selectable object.
- Get2DPickableLayer** : Get layer mask for 2D selectable object.

<b>SetGesturePriority</b>	: Set the gesture priority.
<b>GetGesturePriority</b>	: Return the gesture priority.
<b>SetStationaryTolerance</b>	: Set the stationary tolerance.
<b>GetStationaryTolerance</b>	: Get the stationary tolerance
<b>SetLongTapTime</b>	: Set the long tap time.
<b>GetlongTapTime</b>	: Get the long tap time.
<b>SetDoubleTapTime</b>	: Set the double tap time.
<b>GetDoubleTapTime</b>	: Get the double tap time.
<b>SetSwipeTolerance</b>	: Set the swipe tolerance
<b>GetSwipeTolerance</b>	: Get the swipe tolerance
<b>SetEnable2FingersGesture</b>	: Enables or disables 2 fingers gestures.
<b>GetEnable2FingersGesture</b>	: Return true if 2 fingers gesture is enabled.
<b>SetTwoFingerPickMethod</b>	: Set the 2 fingers selectable method.
<b>GetTwoFingerPickMethod</b>	: Return the 2 fingers selectable method.
<b>SetEnablePinch</b>	: Enables or disables pinch gesture.
<b>GetEnablePinch</b>	: Return true if pinch gesture is enabled.
<b>SetMinPinchLength</b>	: Set the min pinch detection length.
<b>GetMinPinchLength</b>	: Return the min pinch detection lenght.
<b>SetEnableTwist</b>	: Enables or disables twist gesture.
<b>GetEnableTwist</b>	: Return true if twist gesture is enabled
<b>SetMinTwistAngle</b>	: Set the min twist angle detection
<b>GetMinTwistAngle</b>	: Get the min twist angle
<b>GetSecondeFingerSimulation</b>	: Enables or disables second finger simulation.
<b>SetSecondFingerSimulation</b>	: Return true if second finger simulation is enabled.

## Events

Each event is firing with a parameter of type [Gesture](#).

### Single finger events

<b>On_Cancel</b>	: Occurs when The system cancelled tracking for the touch.
<b>On_TouchStart</b>	: Occurs when a finger touched the screen.
<b>On_TouchDown</b>	: Occurs while as the touch is active.
<b>On_TouchUp</b>	: Occurs when a finger was lifted from the screen.
<b>On_SimpleTap</b>	: Occurs when a finger was lifted from the screen
<b>On_DoubleTap</b>	: Occurs when the number of taps is equal to 2 in giving time.
<b>On_LongTapStart</b>	: Occurs when a finger is touching the screen, but hasn't moved since the time required for the detection of a long tap.
<b>On_LongTap</b>	: Occurs while as the touch is active after a LongTapStart.
<b>On_LongTapEnd</b>	: Occurs when a finger was lifted from the screen after a long tap.
<b>On_DragStart</b>	: Occurs when a drag start. A drag is a swipe on a selectable object
<b>On_Drag</b>	: Occurs while as the drag is active.
<b>On_DragEnd</b>	: Occurs when a finger that raise the drag event , is lifted from the screen.
<b>On_SwipeStart</b>	: Occurs when swipe start.
<b>On_Swipe</b>	: Occurs while as the swipe is active.
<b>On_SwipeEnd</b>	: Occurs when a finger that raise the swipe event , is lifted from the screen.

### Two finger events

<b>On_TouchStart2Fingers</b>	: Like On_TouchStart but for a 2 fingers gesture.
<b>On_TouchDown2Fingers</b>	: Like On_TouchDown but for a 2 fingers gesture.
<b>On_TouchUp2Fingers</b>	: Like On_TouchUp but for a 2 fingers gesture.
<b>On_SimpleTap2Fingers</b>	: Like On_SimpleTap but for a 2 fingers gesture.
<b>On_DoubleTap2Fingers</b>	: Like On_DoubleTap but for a 2 fingers gesture.
<b>On_LongTapStart2Fingers</b>	: Like On_LongTapStart but for a 2 fingers gesture.
<b>On_LongTap2Fingers</b>	: Like On_LongTap but for a 2 fingers gesture.
<b>On_LongTapEnd2Fingers</b>	: Like On_LongTapEnd but for a 2 fingers gesture.

<b>On_Twist</b>	: Occurs while a twist gesture.
<b>On_TwistEnd</b>	: Occurs at twist gesture end.
<b>On_Pinch</b>	: Occurs while as the pinch gesture is active.
<b>On_PinchIn</b>	: Occurs while as the pinch in gesture is active.
<b>On_PinchOut</b>	: Occurs while as the pinch out gesture is active.
<b>On_PinchEnd</b> screen.	: Occurs when the 2 fingers that raise the pinch event , are lifted from the
<b>On_DragStart2Fingers</b>	: Like On_DragStart but for a 2 fingers gesture.
<b>On_Drag2Fingers</b>	: Like On_Drag but for a 2 fingers gesture.
<b>On_DragEnd2Fingers</b>	: Like On_DragEnd but for a 2 fingers gesture.
<b>On_SwipeStart2Fingers</b>	: Like On_SwipeStart but for a 2 fingers gesture.
<b>On_Swipe2Fingers</b>	: Like On_Swipe but for a 2 fingers gesture.
<b>On_SwipeEnd2Fingers</b>	: Like On_SwipeEnd but for a 2 fingers gesture.

## Unity UI

<b>On_OverUIElement</b> is On	: Occurs when the touch is over a UI element when Unity Ui compatibility
<b>On_UIElementUp</b> compatibility is On	: Occurs when the touch is lifted from UI element when Unity Ui

## Gesture

Each [event](#) is firing with a parameter of type Gesture. It contains all information about the current gesture.

### Variables

<a href="#">fingerIndex</a>	: The index of the finger that fired the event, or -1 for a two fingers gesture.
<a href="#">touchCount</a>	: The number of active touches.
<a href="#">startPosition</a>	: The start position of the current gesture that fired the event.
<a href="#">position</a>	: The current position of the current gesture that fired the event.
<a href="#">deltaPosition</a>	: The delta position since last call.
<a href="#">actionTime</a>	: The elapsed time in second since the begin of gesture.
<a href="#">deltaTime</a>	: The delta time since last call.
<a href="#">swipe</a>	: The swipe or drag direction.
<a href="#">swipeLength</a>	: The swipe length in pixel.
<a href="#">swipeVector</a>	: The swipe vector direction.
<a href="#">deltaPinch</a>	: The delta pinch length in pixel since last call.
<a href="#">twistAngle</a>	: The delta angle of the twist since the last call.
<a href="#">twoFingerDistance</a>	: The distance in pixel between two finger for a two finger gesture.
<a href="#">pickedObject</a>	: The current picked gameObject.
<a href="#">pickedCamera</a>	: The picked camera.
<a href="#">isGuiCamera</a>	: Is the picked camera is flag as GUI
<a href="#">isOverGui</a>	: Is the current touch is over an Unity UI.
<a href="#">pickedUIElement</a>	: The current picked Unity UI.

### Functions

<a href="#">GetTouchToWorldPoint</a>	: Transforms touch position from screen space into world space.
<a href="#">GetSwipeOrDragAngle</a>	: Gets the swipe or drag angle.
<a href="#">NormalizedPosition</a>	: Gets the normalized position.
<a href="#">GetCurrentPickedObject</a>	: Gets the real picked object under the touch
<a href="#">IsOverUIElement</a>	: Is the current touch is over an Unity UI.



[IsOverRectTransform](#) : Is the current touch is over a specific RectTransform.

GetCurrentFirstPickedUIElement: Gets the real picked Unity UI under the touch.

## **Gesture.fingerIndex**

public int **fingerIndex**

### **Description**

The index of the finger that fired the event, or -1 for a two fingers gesture. Useful for managing the multi-touches.

## **Gesture.touchCount**

public int **touchCount**

### **Description**

The number of active touches. Useful for managing the multi-touches.

## **Gesture.startPosition**

public Vector2 **startPosition**

### **Description**

The start position of the current gesture that fired the event, or the average position between the 2 touches for a two fingers gesture.

## **Gesture.position**

public Vector2 **position**

### **Description**

The current position of the current gesture that fired the event, or the average position between the 2 touches for a two fingers gesture.

## **Gesture.deltaPosition**

public Vector2 **deltaPosition**

### **Description**

The delta position since last call. EasyTouch computes its own delta to prevent difference between all mobile platform.

## **Gesture.actionTime**

public float **actionTime**

### **Description**

The elapsed time in second since the begin of gesture.

## **Gesture.deltaTime**

public float **deltaTime**

### **Description**

The delta time since last call.



## **Gesture.swipe**

public EasyTouch.SwipeDirection **swipe**

### **Description**

The swipe or drag direction. This value is calculated with the [deltaPositon](#) for current gesture, and between [startPosition](#) and current [position](#) for gesture end.

## **Gesture.swipeLength**

public float **swipeLength**

### **Description**

The swipe length in pixel. This value is computed for end gesture between [startPosition](#) and [position](#).

## **Gesture.swipeVector**

public Vector2 **swipeVector**

### **Description**

The swipe vector direction.

## **Gesture.deltaPinch**

public float **deltaPinch**

### **Description**

The delta pinch length in pixel since last call.

This value is positive for On\_PinchIn & On\_PinchOut events

This value is positive for pinch out gesture, and negative for pinch in gesture for On\_Pinch event.

## **Gesture.twistAngle**

public float **twistAngle**

### **Description**

The delta angle of the twist since the last call.

Negative value for clockwise.

Positive value for anit-clockwise.

## **Gesture.twoFingerDistance**

public float **twoFingerDistance**

### **Description**

The distance in pixel between two finger for a two finger gesture.

## **Gesture.pickedObject**

public GameObject **pickedObject**

### **Description**

The current picked gameObject detected at the touch start, or the real object under the touch position if auto update is enabled.

## **Gesture.pickedCamera**

public Camera **pickedCamera**

### **Description**

The picked camera for find the picked object.



## **Gesture.isGuiCamera**

public bool **isGuiCamera**

### **Description**

Is the picked camera is flag as GUI

## **Gesture.isOverGUI**

public bool **isOverGui**

### **Description**

The current touch is over an Unity UI element detected at the touch start, or update in real time is auto-update is enabled.

This value is correct only when the Ui Compatibility Mode is disabled, otherwise it will always be equal to false.

## **Gesture.pickedUIElement**

public GameObject **pickedUIElement**

### **Description**

The current picked Unity UI element detected at the touch start, or update in real time is auto-update is enabled.

This value is correct only when the Ui Compatibility Mode is disabled, otherwise it will always be equal to Null.

## [Gesture](#).GetTouchToWorldPoint

```
public Vector3 GetTouchToWorldPoint(float z)
```

### Description

Transforms the current touch position from screen space into world space.  
The z position is in world units from the camera.

```
public Vector3 GetTouchToWorldPoint(Vector3 position3D)
```

### Description

Transforms the current touch position from screen space into world space.  
position3D is a the destination world position, from that will be computed the length from the camera.

## **Gesture.GetSwipeOrDragAngle**

public float **GetSwipeOrDragAngle**()

### **Description**

Return the swipe or drag angle.

## **Gesture.NormalizedPosition**

public Vector2 **NormalizedPosition**()

### **Description**

Return the normalized position relative to screen.

## **Gesture.GetCurrentPickedObject**

```
public GameObject GetCurrentPickedObject()
```

### **Description**

Gets the real picked object. You don't need to use the functions, if auto-update is enabled.

## **Gesture.IsOverUIElement**

```
public bool IsOverUIElement()
```

### **Description**

Return true if current touch is over an Unity UI element



## **Gesture.IsOverRectTransform**

```
public bool IsOverRectTransform(RectTransform tr, Camera camera=null)
```

### **Description**

Return true if current touch is a recttransform.

## **Gesture.GetCurrentFirstPickedUIElement**

public GameObject **GetCurrentFirstPickedUIElement**()

### **Description**

Gets the real picked Unity UI Element. You don't need to use the functions, if auto-update is enabled.

