Input.Touches
API Reference for Unity3D

Version: 1.1.2
Author: K.Song tan
LastUpdated: 21th July 2012
Forum: http://goo.gl/7EYur
AssetStore: http://goo.gl/3R8Fb

Input.Touches is a library that aims to provide simple and easy to use scripting solutions for various touch input. To use the library, simply drag the prefab Gesture (located in "InputTouches/InputPrefab/Resources" folder) to your scene. The script components on the prefab will allow you to configure various parameters in regards to the various event you wish to intercept. Alternatively you can attach the script Gesture.cs to any empty object in your scene. The dependencies of the script will be automatically added.

This library uses an event system. Upon any recognizable input/gesture, a corresponding event will be fired. You can 'listen' to these events in order to know when a particular input gesture had occurred and execute the corresponding code.

Table of Content

COMPONENTS

Gesture (Gesture.cs)

The primary component which is responsible for firing events.

BasicDetector (BasicDetector.cs)

The component which detect basic input event. These events support **BOTH** touch input and mouse click. They are:

- Touch down - when a finger first touches screen.
- Touch up - when a finger has left the screen.
- While touch - when a finger is touching the screen.
- Mouse down - when the mouse is first clicked.
- Mouse up - when a mouse click is released.
- While mouse - when a mouse button is being clicked.

This component can be disabled if none of the events above are needed.

DragDetector (DragDetector.cs)

The component which detects general and basic events. These events support both touch input and mouse click. They are:

- Finger/mouse drag - drag on screen, by touch and mouse click. Multiple instances of this event can be triggered simultaneously.
- Multi-finger drag – drag on screen, by multiple fingers.

This component can be disabled if none of the events above are needed.

CONFIGURABLE

Property	Type	Description
Min Drag Distance	**float**	Minimum distance in terms of pixel for the cursor to travel before a drag event is being fired.
Enable Multi Drag	**bool**	Check to enable multiple drag instances to be fired simultaneously. Please note that this is only supported for single-finger drag.

TapDetector (TapDetector.cs)

The component which detects common single touch events. These events support both touch input and mouse click. They are:

- Short tap - finger/mouse press and release within a short time frame.
- Long tap - finger/mouse press over a long time period.
- Double tap - double tap/click within a time frame.
- Charge - Charging up a value while holding down a touch/click. Note that there are multiple charge modes.

This component can be disabled if none of the events above are needed.

CONFIGURABLE

Property	Type	Description
Short Tap Time	**float**	The maximum time for a short tap to be valid.
Long Tap Time	**float**	The time required for a long tap event to be fired.
Multi-Tap Interval	**float**	The maximum time window for a second tap to take place for a multi-Tap event to be fired. Otherwise the tap will be registered as a new event and not a continuous event.
Multi-Tap Pos Spacing	**float**	The maximum spacing in pixels allowed for a consecutive tap to be registered as a multi-Tap.
Max Multi-Tap Count	**int**	The maximum multi-Tap count allowed. Set to 1 to disable multi-Tap, 2 to enable double-tap and 3 to enable double-tap and triple-tap and so on.
Charge Mode	**_ChargeMode**	The charge mode for charge event. There are 4 modes which are listed as below: • **Once**: The charge end event will only trigger once and will trigger as soon as the charge reaches full amount. • **Clamp**: The charge end event will only trigger once and will trigger when the touch/mouse has been released. • **Loop**: The charge end event will trigger as soon as the charge reaches full amount. The charge will reset and restart immediately. • **PingPong**: The charge end event will only trigger once the touch/mouse has been released. The charging process will be switching back and forth between 0% and 100%.

Min Charge Time	**float**	Minimum time required for a charge event to start trigger. The value of percent passed by the event at this point will be minChargeTime/maxChargeTime.
Max Charge Time	**float**	Maximum time possible for a charge event. The value of percent passed by the event at this point will be 1.
Max Finger Group Dist	**float**	The maximum distance between each finger for a multi-finger tap event. This is very much device dependent. Naturally this should be set so the pixel on screen covered the size of the number of fingers allowed. 1.5 inches for 2 fingers, 2 inches for 3 fingers and so on.

SwipeDetector (SwipeDetector.cs)

The component which detects swipe events. These events support both touch input and mouse click. This component can be disabled if none of the swipe events are needed in the scene. Multiple instances of swipe event can be triggered simultaneously.

CONFIGURABLE

Property	Type	Description
Max Swipe Duration	**float**	Maximum duration in section for a swipe.
Min Speed	**float**	Minimum relative speed required for a swipe. This is calculated using (pixel-travelled)/(time-swiped)
Min Distance	**float**	Minimum distance in pixels required from the beginning to the end of the swipe.
Max Direction Change	**float**	Maximum change of direction allowed during the swipe. This is the angle difference measured from the initial swipe direction.

DualFingerDetector (DualFingerDetector.cs)

The component which detects common two finger touch gestures. These events do not support mouse input. They are:

- Rotate(twist) - rotate two fingers on screen.
- Pinch - pinch the screen with two fingers.

This component can be disabled if none of the events above are needed.

CONFIGURABLE

Property	Type	Description
Rotation Smooth Frame Count	**int**	Number of previous frame value to be taken into account for smoothing. Only applicable if smoothing is turn on.
Rotation Smoothing	**_SmoothMethod**	The smoothing method to be used for rotation. This is to smooth any spike and give a more consistent value for rotation input. In most case, this is used to compensate the inconsistent nature of the input. There are 3 methods which are listed as below: • **None**: No smoothing will be done at all. • **Average**: Just use the average value across as many frame as specified in • Rotation Smooth Frame Count. • **Weighted-Average**: Like Average but weighted, the value which takes place in more recent frame will be given a higher priority and thus will carry more weight in calculated value.

EVENTS

TAP EVENTS

Method Signatures & Descriptions

Gesture.onMultiTapE

```
public static event Action<Tap> onMultiTapE
```

Fired when a single or a series of short taps are detected. The information of the taps is passed. This includes the position, the tap count, and the input type.

Gesture.onLongTapE

```
public static event Action<Tap> onLongTapE
```

Fired when a long tap is detected. The screen position where the event takes place is passed.

Gesture.onMFMultiTapE

```
public static event Action<Tap> onMFMultiTapE
```

Fired when a single or a series of short taps are detected. The information of the taps is passed. This includes the position, the tap count, and the input type.

Gesture.onMFLongTapE

```
public static event Action<Tap> onMFLongTapE
```

Fired when a long tap is detected. The screen position where the event takes place is passed.

Note: See class Tap for more information

CHARGE EVENTS

Method Signatures & Descriptions

Gesture.onChargeStartE

```
public static event Action<ChargedInfo> onChargeStartE
```

Fired when a holding tap is first detected. The screen position where the event takes place and the amount charged is passed.

Gesture.onChargingE

```
public static event Action<ChargedInfo> onChargingE
```

Fired when a holding tap is detected. The screen position where the event takes place and the amount charged is passed.

Gesture.onChargeEndE

```
public static event Action<ChargedInfo> onChargeEndE
```

Fired when a charging tap is released. The screen position where the event takes place and the amount charged is passed.

Gesture.onMFChargingStartE

```
public static event Action<ChargedInfo> onMFChargingStartE
```

Fired when a holding tap is first detected. The screen position where the event takes place and the amount charged is passed.

Gesture.onMFChargingE

```
public static event Action<ChargedInfo> onMFChargingE
```

Fired when a holding tap is detected. The screen position where the event takes place and the amount charged is passed.

Gesture.onMFChargeEndE

```
public static event Action<ChargedInfo> onMFChargeEndE
```

Fired when a charging tap is released. The screen position where the event takes place and the amount charged is passed.

Note: See class ChargedInfo for more information

DRAG EVENTS

Method Signatures & Descriptions

Gesture.onDraggingStartE

```
public static event Action<DragInfo> onDraggingStartE
```

Fired when a single-finger/mouse dragging event is first detected. Relevant info of the event is passed.

Gesture.onDraggingE

```
public static event Action<DragInfo> onDraggingE
```

Fired when a single-finger/mouse dragging event is being executed. Relevant info of the event is passed.

Gesture.onDraggingEndE

```
public static event Action<DragInfo> onDraggingEndE
```

Fired when a single-finger/mouse dragging event ended. Relevant info of the event is passed.

Gesture.onMFDraggingStartE

```
public static event Action<DragInfo> onMFDraggingStartE
```

Fired when a multi-finger dragging event is first detected. Relevant info of the event is passed.

Gesture.onMFDraggingE

```
public static event Action<DragInfo> onMFDraggingE
```

Fired when a multi-finger dragging event is being executed. Relevant info of the event is passed.

Gesture.onMFDraggeEndE

```
public static event Action<DragInfo> onMFDraggingEndE
```

Fired when a multi-finger dragging event ended. Relevant info of the event is passed.

Note: See class DragInfo for more information

SWIPE EVENTS

Method Signatures & Descriptions

Gesture.onSwipeStartE

```
public static event Action<SwipeInfo> onSwipeStartE
```

Fired when a swipe is first detected. Relevant info of the swipe is passed. Please note that the swipe events passed in this instance are not subject to all the specified parameters in SwipeDetector.cs and thus may not be a valid event.

Gesture.onSwipingE

```
public static event Action<SwipeInfo> onSwipingE
```

Fired when a swipe event is being executed. Relevant info of the swipe is passed. Please note that the swipe events passed in this instance are not subject to all the specified parameters in SwipeDetector.cs and thus may not be a valid event.

Gesture.onSwipeEndE

```
public static event Action<SwipeInfo> onSwipeEndE
```

Fired when a swipe event has finished. Relevant info of the swipe is passed. Please note that the swipe events passed in this instance are not subject to all the specified parameters in SwipeDetector.cs and thus may not be a valid event.

Gesture.onSwipeE

```
public static event Action<SwipeInfo> onSwipeE
```

Fired when a swipe is detected. Relevant info of the swipe is passed.

Note: See class SwipeInfo for more information

PINCH/ROTATE(TWIST) EVENTS

Method Signatures & Descriptions

Gesture.onPinchE

```
public static event Action<float> onPinchE
```

Fired when a pinch event is detected. The magnitude of the pinch is passed. The value passed is +ve if the pinch is inward pinch, -ve if outward pinch.

Gesture.onRotateE

```
public static event Action<float> onRotateE
```

Fired when a 2 fingers rotate gesture is detected. The magnitude of the rotation is passed. When rotating direction is clockwise, the value is -ve. Otherwise it's +ve.

TOUCH/CLICK EVENTS

Method Signatures & Descriptions

Gesture.onTouchDownE

```
public static event Action<Vector2> onTouchDownE
```

Fired when a touch first initiated. The screen position where the touch occurs is passed. Multiple instances of this event can be fired simultaneously if there is more than 1 touch on screen.

Gesture.onTouchUpE

```
public static event Action<Vector2> onTouchUpE
```

Fired when a touch is released. The screen position where the touch occurs is passed. Multiple instances of this event can be fired simultaneously if there is more than 1 touch on screen.

Gesture.onTouchE

```
public static event Action<Vector2> onTouchE
```

Fired while a touch is detected. The screen position where the touch occurs is passed. Multiple instances of this event can be fired simultaneously if there is more than 1 touch on screen.

Gesture.onMouse1 DownE

```
public static event Action<Vector2> onMouse1DownE
```

Fired when a left mouse click is first initiated. The position of the mouse is passed. This is equivalent of Input.GetMouseButtonDown(0).

Gesture.onMouse1UpE

```
public static event Action<Vector2> onMouse1UpE
```

Fired when the left mouse pressed is released. The position of the mouse is passed. This is equivalent of Input.GetMouseButtonUp(0).

Gesture.onMouse1E

```
public static event Action<Vector2> onMouse1E
```

Fired while the left mouse button is pressed. The position of the mouse passed. This is equivalent of Input.GetMouseButton(0).

Gesture.onMouse2DownE

```
public static event Action<Vector2> onMouse2DownE
```

Fired when a right mouse click is first initiated. The position of the mouse is passed. This is

equivalent of Input.GetMouseButtonDown(1).

Gesture.onMouse2UpE

```
public static event Action<Vector2> onMouse2UpE
```

Fired when the right mouse button is released. The position of the mouse is passed. This is equivalent of Input.GetMouseButtonUp(1).

Gesture.onMouse2E

```
public static event Action<Vector2> onMouse2E
```

Fired while the right mouse button is pressed. The position of the mouse is passed. This is equivalent of Input.GetMouseButton(1).

OBSOLETE EVENTS

Method Signatures & Descriptions

Gesture.onShortTapE

```
public static event Action<Vector2> onShortTapE
```

Fired when a short tap is detected. The screen position where the event takes place is passed. This event is obsolete, use onMultiTapE instead when possible.

Gesture.onDoubleTapE

```
public static event Action<Vector2> onDoubleTapE
```

Fired when a double tap is detected. The screen position where the event takes place is passed. This event is obsolete, use onMultiTapE instead when possible.

Gesture.onDFShortTapE

```
public static event Action<Vector2> onDFShortTapE
```

Similar to onShortTapE, only this is for two fingers. The position passed is the center between the position of the two fingers. This event is obsolete, use onMFMultiTapE instead when possible.

Gesture.onDFDoubleTapE

```
public static event Action<Vector2> onDFDoubleTapE
```

Similar to onDoubleTapE, only this is for two fingers. The position passed is the center between the position of the two fingers. This event is obsolete, use onMFMultiTapE instead when possible.

Gesture.onDFLongTapE

```
public static event Action<Vector2> onDFLongTapE
```

Similar to onLongTapE, only this is for two fingers. The position passed is the center between the position of the two fingers. This event is obsolete, use onMFLongTapE instead when possible.

Gesture.onDFChargingE

```
public static event Action<ChargedInfo> onDFChargingE
```

Similar to onChargingE, only this is for two fingers. The screen position where the event takes place and the amount charged is passed. This event is obsolete, use onMFChargingE instead when possible. See ChargedInfo for more information.

Gesture.onDFChargeEndE

```
public static event Action<ChargedInfo> onDFChargeEndE
```

Similar to onChargeEndE, only this is for two fingers. The screen position where the event takes place and the amount charged is passed. This event is obsolete, use onMFChargeEndE instead when possible. See ChargedInfo for more information.

Gesture.onDualFDraggingE

```
public static event Action<DragInfo> onDualFDraggingE
```

Fired when a two-fingers dragging event is being executed. Relevant info of the event is passed. This event is obsolete, use onMFDraggingE instead when possible. See DragInfo for more information.

Gesture.onDualFDraggingEndE

```
public static event Action<Vector2> onDualFDraggingEndE
```

Fired when two-fingers dragging event ended. Relevant info of the event is passed. This event is obsolete, use onMFDraggingEndE instead when possible. See DragInfo for more information.

NOTES ON USING EVENT

To listen to any particular event, you have to “subscribe” to it. Also you will need to “unsubscribe” when you no longer wish to receive the events. To do this simply add the following line to your script:

for C#,

```
void OnEnable(){
        //subscribe to an event
        Gesture.EventName += YourCustomFunction;
}

void OnDisable(){
        //unsubscribe to an event
        Gesture.EventName -= YourCustomFunction;
}

//function call when event is fired
void  YourCustomFunction(Type parameter){
        //Your custom code;
}
```

for js,

```
function OnEnable(){
        //subscribe to an event
        Gesture.EventName += YourCustomFunction;
}

function OnDisable(){
        //unsubscribe to an event
        Gesture.EventName -= YourCustomFunction;
}

//function call when event is fired
function  YourCustomFunction(Type parameter){
        //Your custom code;
}
```

You can write your own custom code in the custom functi on, just make sure you have the passing parameter type correctly declared. The parameter types are stated in the event listing above.

You can find more than adequate examples in each example scene. The corresponding script are placed in a folder named “Scripts/C#”.

PUBLIC CLASSES

To fully utilize the event, you will need to know the members of each gesture instance. Apart from the usual Unity Vector2 type, there are 4 custom classes used in this library. These classes contain information about each event type and are passed along with each instance of an event.

class Tap

Passed with a (single or multi) tap event. Contains all the information for a tap event. The class covers shortTap, longTap, multi-tap events and the multi-fingers counterpart.

PUBLIC MEMBER

Property	Type	Description
pos	**Vector2**	The screen position of the cursor for click/single finger event. For multi-fingers events, this is the averaged position between all the triggering fingers.
count	**int**	The number of taps for this particular event. 1 for single-tap, 2 for double-tap and so on.
fingerCount	**int**	The number of fingers triggering this event.
positions	**Vector2[]**	All the positions of all the fingers triggering this event.
isMouse	**bool**	Boolean flag to indicate if the input type is a mouse input.
index	**int**	Unique index to indicate which touch/mouse input trigger the event. This is needed so an event by a particular touch can be tracked when there are multiple events on screen. This is only applicable if the value fingerCount is 1. For indexes of multi-finger event, use indexes.
indexes	**int[]**	All the unique indexes of the touch triggering the event.

class ChargedInfo

Passed with a charge event. Contains all the information for a charge event. The class covers single and multi-fingers charge events.

PUBLIC MEMBER

Property	Type	Description
percent	**float**	The screen position of the cursor for click/single finger event. For multi-fingers events, this is the averaged position between all the triggering fingers.
pos	**Vector2**	The number of taps for this particular event. 1 for single-tap, 2 for double-tap and so on.
fingerCount	**int**	The number of fingers triggering this event.
positions	**Vector2[]**	All the positions of all the fingers triggering this event.
isMouse	**bool**	Boolean flag to indicate if the input type is a mouse input.
index	**int**	Unique index to indicate which touch/mouse input trigger the event. This is needed so an event by a particular touch can be tracked when there are multiple events on screen. This is only applicable if the value fingerCount is 1. For indexes of multi-finger event, use indexes.
indexes	**int[]**	All the unique indexes of the touch triggering the event.

Obsolete Member (following members are obsolete and are no longer in use)

pos1	**Vector2**	The screen position of the first finger. this is only valid if the event fired is two finger event.
pos2	**Vector2**	The screen position of the second finger. this is only valid if the event fired is two finger event.

class DragInfo

Passed with a drag event. Contain all the information for a drag event. This class covers drag event triggered by multiple fingers.

PUBLIC MEMBER

Property	Type	Description
pos	**Vector2**	The screen position of the cursor. This is the averaged position for all the fingers in the event, in case it was triggered by multiple fingers.
delta	**Vector2**	The moved direction of the event.
fingerCount	**int**	The number of fingers triggering this event.
isMouse	**bool**	Boolean flag to indicate if the input type is a mouse input.
index	**int**	Unique index to indicate which touch/mouse input triggered the event. This is needed so an event by a particular touch can be tracked when there are multiple events on screen. This is not applicable for multi-finger events.

class SwipeInfo

Passed with a swipe event. Contains all the information about the swipe event.

PUBLIC MEMBER

Property	Type	Description
pos	**Vector2**	The screen position of the cursor when the swipe event started.
count	**Vector2**	The screen position of the cursor when the swipe event ended.
direction	**Vector2**	The direction vector of the swipe.
angle	**float**	The angle of the swipe on the screen. Start from positive x-axis in counter-clockwise direction.
duration	**float**	The duration of the swipe in second.
speed	**float**	The relative speed of the swipe. Calculated by (number of pixel travelled)/(duration of the swipe).

isMouse	**bool**	Boolean flag to indicate if the input type is a mouse input.
index	**int**	Unique index to indicate which touch/mouse input triggered the event. This is needed so an event by a particular touch can be tracked when there are multiple event on screen.

CONTACT INFORMATION

Thanks for using this library. I hope you find it useful. I'll try my best to provide support regarding issues you might have. I aim to provide unprecedented support within my best ability. Please visit my development blog http://songgamedev.blogspot.com/ or the official support thread on unity forum to leave a comment or email me directly at k.songtan@gmail.com.

VERSION HISTORY

Version Change – v1.1

General Change:

- swipe event can now be triggered consecutively without the need to lift the finger.
- all dual-finger(DF) event has been replaced by multi-finger(MF) event. Multi-finger event
- support events triggered by 2 or more fingers.
- added support for concurrent event, following event now can be triggered in multiple instances with different fingers simultaneously:
    - swipe
    - single finger drag
    - single finger shortTap/longTap/charge/doubleTap
    - Multi-fingers shortTap/longTap/charge/doubleTap

Bug Fixes:

- fix bug where 2 finger twist (rotate) is not register correctly
- fix bug where some swipe are not tracked

Existing event changes:

- onLongTapE: passing parameter changed from Vector2 to Tap
- onDraggingEndE: passing parameter changed from Vector 2 to DragInfo
- onDualFDraggingEndE: passing parameter changed from Vector 2 to DragInfo

New Event:

- onMultiTapE
- onChargeStartE
- onDraggingStartE
- onMFMultiTapE
- onMFChargeStartE
- onMFChargingE
- onMFChargeEndE
- onMFDraggingStartE
- onMFDraggingE
- onMFDraggingEnd

New Class – Tap

Please note that this is a major, major update. Most of the library has been reworked. Thus not all change are listed here. Minor changes may have been left out.

Version Change – v1.1.1

General Change:

- added smoothing configuration to 2 fingers rotate (twist)
- added two new mini-game demos, turret and flick-shoot

Bug Fixes:

- fix bug where onSwipeStartE, onSwipingE and onSwipeEndE events wouldnt fire
- fix bug but with 2 fingers rotate (twist)
- fix error description on documentation, specifically for chargeEvent

Version Change – v1.1.2

General Change:

- documentation overhaul. Thanks to DannyB and LanceM

Bug Fixes:

- releasing finger on mobile device no longer triggered a spike in onDraggingE and onDraggingEnd
- pre-made gesture prefab is no longer missing BasicDetector.cs