“Input.Touches” version 1.1

for Unity3D by K.SongTan

Input.Touches are a library aims to provide simple and easy to use scripting solution for various touch input. To use the library, simple drag the prefab Gesture (placed in 'InputPrefab' folder) to your scene. The script components on the prefab will allow you to configure various parameter in regarding to the various event covered. Alternatively you could attached the script Gesture.cs to any empty object in your scene. The dependencies on the script will take care of the rest.

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

1. Components

Gesture.cs

The primary component which responsible for firing events.

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 touch on screen
- touch up - when a finger left the screen
- while touch - when a finger is touching the screen
- mouse down - when the mouse is first click
- 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 event above are needed.

DragDetector.cs

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

- finger/mouse drag - drag on screen, by touch and mouse click. Multiple instance 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 event above are needed.

Configurable

float **minDragDistance**: minimum distance in term of pixel for the cursor has to travel before a drag event is start being registered.

bool **enableMultiDrag**: check to enable multiple drag instance to be fired simultaneously. Please note that this is only supported for single-finger drag.

TapDetector.cs

The component which detect common single touch event. 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 mode

this component can be disabled if none of the event above are needed.

Configurable
***float* shortTapTime** - maximum time for a short tap to be valid
***float* longTapTime** - the time required for a long tap event to fired

***float* multiTapInterval – the maximum time window for a second tap to take place for a multiTap event to registered. Otherwise the tap will be registered as new but not a continuous event.**
***float* multiTapPosSpacing – the maximum spacing in pixel allowed for a consecutive tap to be registered as a multiTap.**
***int* maxMultiTapCount – the maximum multiTap count allowed. Set 1 to disable multiTap, 2 to enabled double-tap, 3 to enable double-tap and triple-tap.**

***_ChargeMode* chargeMode**: The charge mode for charge event. There are 4 modes which are listed as below:

> ***Once***: The charge end event will only triggered once and will triggered as soon as the charge reach full amount.
> ***Clamp***: The charge end event will only triggered once and will only triggered when the touch/mouse has been released
> ***Loop***: The charge end event will triggered as soon as the charge reach full amount. The charge will reset and restart immediately.
> ***PingPong*:** The charge end event will only triggered once the touch/mouse has been released. The charging process will be switching from forward and reverse each time the charge amount reach 0% or 100%.

***float* minChargeTime**: minimum time required for a charge event to start trigger. The value of percent passed by the event at this point would be minChargeTime/maxChargeTime.
***float* maxChargeTime**: maximum time possible for a charge event. The value of percent passed by the event at this point would be 1.

***float* maxFingerGroupDist**: 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 finger allowed. 1.5 inches for 2 finger, 2 inches for 3 fingers and so on.

SwipeDetector.cs

The component which detect swipe event. These events support **BOTH** touch input and mouse click. this component can be disabled if none of the swipe event is not needed in the scene. Multiple instance of swipe event can be triggered simultaneously.

Configurable

float **maxSwipeDuration** - maximum duration for a swipe
float **minSpeed** - minimum relative speed required for a swipe
float **minDistance** - minimum distance in pixel required from the start to the end of the swipe
flaot **maxDirectionChange** - maximum change of direction allowed during the swipe

DualFingerDetector.cs

The component which detect common special 2 fingers touch gesture. These events **DOES NOT** support mouse input. They are:

- rotate: rotate two finger on screen
- pinch: pinch the screen with two fingers

this component can be disabled if none of the event above are needed.

2. Events

All events are fired from Gesture.cs. They are listed as follow:

public static event Action<MultiTap> onMultiTapE*
- fired when single or series of short taps is detected. The information of the multiple taps is passed. This include the position, the tap count, and the input type.

public static event Action<MultiTap> onLongTapE*
- fired when a long tap is detected. The screen position where the event take place is passed.

public static event Action<MultiTap> onMFMultiTapE*
- fired when single or series of short taps is detected. The information of the multiple taps is passed. This include the position, the tap count, and the input type.

public static event Action<MultiTap> onMFLongTapE*
- fired when a long tap is detected. The screen position where the event take place is passed.

** see MultiTap (pg.6) for more information*

==========

public static event Action<ChargedInfo> onChargeStartE**
- fired when a holding tap is first detected. The screen position where the event take place and the amount charged is passed.

public static event Action<ChargedInfo> onChargingE**
- fired when a holding tap is detected. The screen position where the event take place and the amount charged is passed.

public static event Action<ChargedInfo> onChargeEndE**
- fired when a charging tap is released. The screen position where the event take place and the amount charged is passed.

public static event Action<ChargedInfo> onMFChargingStartE**
- fired when a holding tap is first detected. The screen position where the event take place and the amount charged is passed.

public static event Action<ChargedInfo> onMFChargingE**
- fired when a holding tap is detected. The screen position where the event take place and the amount charged is passed.

public static event Action<ChargedInfo> onMFChargeEndE**
- fired when a charging tap is released. The screen position where the event take place and the amount charged is passed.

*** see ChargedInfo (pg.6) for more information*

==========

public static event Action<DragInfo> onDraggingStartE***
- fired when a dragging event is first detected. Relevant info of the event is passed. This event is obsolete, use onMFDraggingStartE instead when possible.

public static event Action<DragInfo> onDraggingE***
- fired when a dragging event is being executed. Relevant info of the event is passed. This event is obsolete, use onMFDraggingE instead when possible.

public static event Action<DragInfo> onDraggingEndE***
- fired when a dragging event ended. Relevant info of the event is passed. This event is obsolete, use onMFDraggingEndE instead when possible.

public static event Action<DragInfo> onMFDraggingStartE***
- fired when a dragging event is first detected. Relevant info of the event is passed. This event is obsolete, use onMFDraggingStartE instead when possible.

public static event Action<DragInfo> onMFDraggingE***
- fired when a dragging event is being executed. Relevant info of the event is passed. This event is obsolete, use onMFDraggingE instead when possible.

public static event Action<DragInfo> onMFDraggingEndE***
- fired when a dragging event ended. Relevant info of the event is passed. This event is obsolete, use onMFDraggingEndE instead when possible.

*** *see DragInfo (pg.6) for more information*

==========

public static event Action<SwipeInfo> onSwipeStartE****
- fired when a swipe is first detected. Relevant info of the swipe is passed. Please note The swipe event passed in this instance are not subject to all the specified parameter in SwipeDetector.cs and thus maybe not be a valid event.

public static event Action<SwipeInfo> onSwipingE****
- fired when a swipe event is being executed. Relevant info of the swipe is passed. Please note The swipe event passed in this instance are not subject to all the specified parameter in SwipeDetector.cs and thus maybe not be a valid event.

public static event Action<SwipeInfo> onSwipeEndE****
- fired when a swipe event has finished. Relevant info of the swipe is passed. Please note The swipe event passed in this instance are not subject to all the specified parameter in SwipeDetector.cs and thus maybe not be a valid event.

public static event Action<SwipeInfo> onSwipeE****
- fired when a swipe is detected. Relevant info of the swipe is passed.
**** *see SwipeInfo (pg.6) for more information*

==========

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.
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.

==========

public static event Action<Vetor2> 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 are more than 1 touch on screen.

public static event Action<Vetor2> 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 are more than 1 touch on screen.

public static event Action<Vetor2> 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 are more than 1 touch on screen.

public static event Action<Vetor2> onMouse1DownE
- fired when a left mouse click is first initiated. The screen position where mouse is passed. This is equivalent of Input.GetMouseButtonDown(0).

public static event Action<Vetor2> onMouse1UpE
- fired when the left mouse pressed is released. The screen position where mouse is passed. This is equivalent of Input.GetMouseButtonUp(0).

public static event Action<Vetor2> onMouse1E
- fired while the left mouse button is pressed. The screen position where mouse is passed. This is equivalent of Input.GetMouseButton(0).

public static event Action<Vetor2> onMouse2DownE
- fired when a right mouse click is first initiated. The screen position where mouse is passed. This is equivalent of Input.GetMouseButtonDown(1).

public static event Action<Vetor2> onMouse2UpE
- fired when the right mouse pressed is released. The screen position where mouse is passed. This is equivalent of Input.GetMouseButtonUp(1).

public static event Action<Vetor2> onMouse2E
- fired while the right mouse button is pressed. The screen position where mouse is passed. This is equivalent of Input.GetMouseButton(1).

Following event are obsolete in current version but are still supported, please use the suggested alternative when possible

public static event Action<Vetor2> onShortTapE
- fired when a short tap is detected. The screen position where the event take place is passed. This event is obsolete, use onMultiTapE instead when possible.

public static event Action<Vetor2> onDoubleTapE
- fired when a double tap is detected. The screen position where the event take place is passed. This event is obsolete, use onMultiTapE instead when possible.

public static event Action<Vetor2> onDFShortTapE
- similar to onShortTapE, only this is for two fingers. The position pass is the center between two fingers position. This event is obsolete, use onMFMultiTapE instead when possible.

public static event Action<Vetor2> onDFDoubleTapE
- similar to onDoubleTapE, only this is for two fingers. The position pass is the center between two fingers position. This event is obsolete, use onMFMultiTapE instead when possible.

public static event Action<Vetor2> onDFLongTapE
- similar to onLongTapE, only this is for two fingers. The position pass is the center between two fingers position. This event is obsolete, use onMFLongTapE instead when possible.

public static event Action<ChargedInfo> onDFChargingE**
- similar to onChargingE, only this is for two fingers. The screen position where the event take place and the amount charged is passed. This event is obsolete, use onMFChargingE instead when possible.

public static event Action<ChargedInfo> onDFChargeEndE**
- similar to onChargeEndE, only this is for two fingers. The screen position where the event take place and the amount charged is passed. This event is obsolete, use onMFChargeEndE instead when possible.

*** see ChargedInfo (pg.6) for more information*

public static event Action<DragInfo> onDualFDraggingE***
- fired when a dragging event is being executed. Relevant info of the event is passed. This event is obsolete, use onMFDraggingE instead when possible.

public static event Action<Vector2> onDualFDraggingEndE***
- fired when a dragging event ended. Relevant info of the event is passed. This event is obsolete, use onMFDraggingEndE instead when possible.

**** see DragInfo (pg.6) for more information*

Little note on using event

To listen to any particular event, you have to "subscribe" to it. Also you will need to "unsubscribe" upon disable a particular sciprt. To do these simply add the following line to your script:

for C#,

```
void OnEnable(){
    Gesture.EventName += YourCustomFunction;
}

void OnDisable(){
    Gesture.EventName -= YourCustomFunction;
}

void  YourCustomFunction(Type parameter){
    //Your custom code;
}
```

for js,

```
function OnEnable(){
    Gesture.EventName += YourCustomFunction;
}

function OnDisable(){
    Gesture.EventName -= YourCustomFunction;
}

function  YourCustomFunction(Type parameter){
    //Your custom code;
}
```

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

You can find more than adequate examples in each example scene. The corresponding script are place in folder named "Scripts/C#".

3. Public class and class member

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

Tap class

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

public member

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

ChargedInfo class

Passed with a charge event. Contain all the information for a charge event. The class cover single and multi-fingers charge event

public member

public float percent: the percent of the charge. takes value from 0.0 - 1.0
public Vector2 pos: the screen position of the cursor for click/single finger event. For multi-fingers event, this is the averaged position between the all triggering fingers
public int fingerCount: The number of fingers triggering this event.
public Vector2[] positions: All the positions of all the fingers triggering this event.
public bool isMouse: boolean flag indicate if the input type is a mouse input.
public int index: unique index indicate which touch/mouse input trigger the event. This is so an event by a particular touch can be keep tracked when there are multiple event on screen. For indexes of multi-finger event, use indexes.
Public int[] indexes: All the unique index of the touch triggering the event.

Follwing member has obsolete and no longer in use

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

DragInfo class

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

public member

public Vector2 pos: the screen position of the cursor. This is the averaged position for all the fingers is the event is triggered by multiple fingers.
public Vector2 delta: the moved direction of the event.
public int fingerCount: the number of finger triggering this event.
public bool isMouse: boolean flag indicate if the input type is a mouse input.
public int index: unique index indicate which touch/mouse input trigger the event. This is so an event by a particular touch can be keep tracked when there are multiple event on screen. This is not applicable for multi-finger event.

SwipeInfo class

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

public member

public Vector2 startPoint: the screen position of the cursor when the swipe event start
public Vector2 endPoint: the screen position of the cursor when the swipe event end
public Vector2 direction: the direction vector of the swipe
public float angle: the angle of the swipe on screen. Start from +ve x-axis in counter-clockwise direction
public flot duration: the duration taken for the swipe
public float speed: the relative speed of the swipe.
public bool isMouse: boolean flag indicate if the input type is a mouse input.
public int index: unique index indicate which touch/mouse input trigger the event. This is so an event by a particular touch can be keep tracked when there are multiple event on screen.

Contract Note

Thanks for using this library. I hope you find it useful. I'll try my best to provide support regarding issue you might have with the library. 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 Change – v1.1

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

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.