“Input.Touches” version 1.0

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 Prefabs to your scene. The script components on the prefab will allow you to configure various parameter in regarding to the various event covered.

This library uses an event system. Upon any recognizable input/gesture, a corresponding event will be fired.

1. Components

Gesture.cs

The primary component which responsible for firing events.

General.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
- 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.

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 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

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

Configurable

shortTapTime - maximum time for a short tap to be valid
longTapTime - the time required for a long tap event to fired
maxLTapSpacing - the maximum cursor position change in pixel from the start point for a long tap to be valid.

doubleTapTime - maximum time for a double tap to be valid
maxDTapPosSpacing - maximum position difference in pixel between two tap for a double tap to be valid
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/maxChargeTme
maxChargeTime - maximum time possible for a charge event. The value of percent passed by the event at this point would be 1

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.

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

DualFingerDetector.cs

The component which detect common single touch event. These events **DOES NOT** support mouse input. They are:

- rotate: rotate two finger on screen
- pinch: pinch the screen with two fingers
- TwoFingersdrag: drag on the screen with 2 fingers

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

Configurable
shortTapTime - maximum time for a two fingers short tap to be valid
longTapTime - the time required for a two fingers long tap event to fired
maxLTapSpacing - the maximum cursor position change in pixel from the start point for a two fingers long tap to be valid
doubleTapTime - maximum time for a two fingers double tap to be valid
maxDTapPosSpacing - maximum position difference in pixel between two tap for a two fingers double tap to be valid
minChargeTime - minimum time required for a two fingers charge event to start trigger. The value of percent passed by the event at this point would be minChargeTime/maxChargeTme
maxChargeTime - maximum time possible for a two fingers charge event. The value of percent passed by the event at this point would be 1

2. Events

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

public static event Action<Vetor2> onShortTapE
- fired when a short tap is detected. The screen position where the event take place is passed

public static event Action<Vetor2> onDFShortTapE
- similar to onShortTapE, only this is for two fingers. The position pass is the centre between two fingers position.

public static event Action<Vetor2> onLongTapE
- fired when a long tap is detected. The screen position where the event take place is passed

public static event Action<Vetor2> onDFLongTapE
- similar to onLongTapE, only this is for two fingers. The position pass is the centre between two fingers position.

public static event Action<Vetor2> onDoubleTapE
- fired when a double tap is detected. The screen position where the event take place is passed

public static event Action<Vetor2> onDFDoubleTapE
- similar to onDoubleTapE, only this is for two fingers. The position pass is the centre between two fingers position.

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> onDFChargingE*
- similar to onChargingE, only this is for two fingers. 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> onDFChargeEndE*
- similar to onChargeEndE, only this is for two fingers. 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> onDraggingE **
- fired when a dragging event is detected. Relevant info of the event is passed. This event can be triggered by both left/right mouse button and single finger.

public static event Action<DragInfo> onDualFDraggingE **
- fired when a two fingers dragging event is detected. Relevant info of the event is passed.

public static event Action<Vector2> onDraggingEndE
- fired when a dragging event ended. Relevant info of the event is passed.

public static event Action<Vector2> onDualFDraggingEndE
- fired when a two fingers dragging event ended. Relevant info of the event is passed.

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

public static event Action<SwipeInfo> onSwipeE ***
- fired when a swipe is detected. Relevant info of the swipe is passed
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.

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

public static event Action<Vetor2> onTouchDown
- 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> onTouchUp
- 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> onTouch
- 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).

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 3 custom class used in this library.

DragInfo class

Passed with a drag event. Contain all the information for a drag event

public member
public int type: the input type which trigger the event
0 - finger touch
1 - left mouse button
2 - right mouse button
public Vector2 pos: the screen position of the cursor
public Vector2 delta: the moved direction of the event

ChargedInfo class

Passed with a charge event. Contain all the information for a 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 two fingers event, this is the position between the two fingers
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.

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.

Contract Note

Thanks for using this library. I hope you find it useful. I'll try my best to provide support regarding issue. I aim to provide unprecedented support within my best ability. Please visit http://songgamedev.blogspot.com/ to leave a comment or email me directly at k.songtan@gmail.com.

As you may aware this is version0.9 of this particular library. I definitely hope to expand and add more feature, example. So any idea and feedback are welcome.