

Overview

With the advent of dual and multi touch pointer devices users can now use gestures to interact with the operating system and applications. The major desktop operating systems are all catering for gesture interaction and promoting the use of gestures within applications with well defined gesture application interfaces.

UPDD can be configured to support single, dual and multi-touch devices. For each stylus in use the positional data is made available on the UPDD API as well as optionally passed into the OS and/or applications to cater for gesture utilization.

The injection of single, dual and multi-touch data into operating systems and their applications is dictated by the interface methods implemented within the OS environment. These are the interface methods utilised by the UPDD driver:

Operating system	Gesture Interface	Inking interface	Link
Mouse emulation	Passed to the OS on the mouse port interface as pen down, pen up and movement. This is the most basic touch interface on all operating systems.	N/A	n/a
Windows Vista	A virtual HID device is utilised to pass single stylus data to the OS. Vista caters for single touch gestures only. Touch applications interface with the OS via a touch API interface.	TBA	
Windows 7	A virtual HID device is utilised to pass all stylus data to the OS. Win 7 caters for single and multi touch gestures. Touch applications interface with the OS via a touch API interface.	Windows 7 users launch the Tablet Input Panel Written text is converted and inserted into application. Will work with UPDD Virtual HID device.	
Mac OS X	Passed as touch events to satisfy applications using the touch event interface.	Enabled if a tablet, virtual tablet or UPDD Gestures is installed. 'Pen' data passed as tablet events.	Lion
Linux	TBA	TBA	
CE	Since WE6 can be enabled in the CE image. Handled by Touch Screen (Stylus) GWES UPDD interface	Transcriber Handwriting Recognizer Application Handled by Touch Screen (Stylus) GWES UPDD interface.	Read

The interpretation of the gesture is dependant on the OS and application. There are many articles on the web to describe gesture utilisation by the OS desktop (windows manager) and key standard applications. However, a very useful gesture reference guide can be found [here](#).

Implementation

It is our intention to build full gesture support into the driver and/or supporting utilities but in some cases we currently utilise external UPDD API based apps to handle gesture functionality. The current implementation of gesture support for various operating system is as follows:

Windows desktop

Under Windows Vista and Windows 7 the driver has a user setting in the UPDD Console, called [Extended touch](#). If enabled all touches are fed to the OS via the virtual HID device to invoke the extended touch functionality (gestures etc) built into these operating systems. If disabled all single touches and the touch data from the 1st stylus (of a multi touch device) are passed to the OS via the mouse port interface (mouse emulation).

Windows CE

UPDD CE 4.1.10 handles touch via the CE standard GWES interface so [CE gesture support](#) can be utilised by any touch device using UPDD CE driver

Mac OS X

For this OS we use a standalone application. As of 7th July 2011 an updated version supports both Snow Leopard and Lion gestures and inking and is available [here](#). When using this version the start process automatically chooses the correct version to run.

With the latest version posted 21st Sept 2011, the two finger scroll, pinch/magnify, and rotate gestures now moves the cursor to the center of the gesture so that user interface elements will respond to it properly. In addition this contains a fix for a Inking drawing bug.

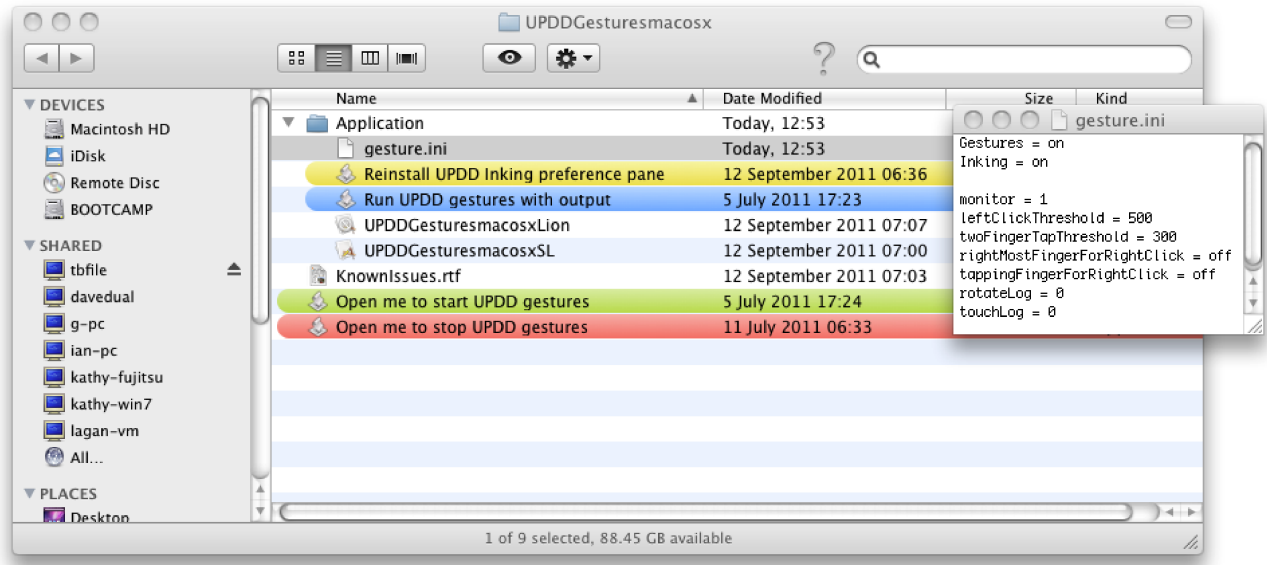
Installation

Given that this utility is for beta test purpose prior to full driver integration it expires on Jan 1, 2012, after which it will display the message "This version of UPDDGesturesmacosx has expired. Please obtain a new version of the application."

To utilise gestures and inking in the Mac environment you need to simply run the UPDDGesturesmacosx.app application on a system with UPDD 4.1.10 or above installed.

The easiest way to run the application is to expand the compressed file to create the container folder (this may happen automatically as part of the download depending on browser in use) and use the two control applications which start and

stop the application located in sub folder Application, as shown:



The application interacts with the core UPDD driver to receive all touches and calculate the gesture being performed and inject this into the OS as a touch event and also pass single touch events to the tablet interface. Use the Start and Stop control programs as required as the application does not utilise a dock item or menu bar. Utilising the gestures and inking functions are described below.

Gesture specifics

Gestures are performed on the touch screen exactly as they are on a track-pad. To utilize all available gestures you will need to use a multi-touch touch screen that supports up to 5 touches otherwise you will be restricted to the gestures that relate to the number of stylus supported on the touch screen.

Snow Leopard video taken on a dual touch touch screen (thus restricted to dual touch gestures). This video shows UPDD gesture functionality applied in Mac OS X Snow Leopard user interface and gesture enabled applications.

Lion video taken on a multi touch touch screen. This video shows UPDD gesture functionality applied in Mac OS X Lion user interface and gesture enabled applications.



A guide to Lion gestures in PDF / Printable format is available [here](#).

To view the gestures calculated by the gesture engine invoke the gesture engine via the 'Run UPDD gestures with output' control program. This will load a terminal windows and list the gesture being generated, as per the example below:

```
Terminal — UPDDGesturesmaco — 50x11
; exit;
Current gesture is: click and drag (drag gesture)
(gesture ended)
Current gesture is: magnify (pinch / zoom)
(gesture ended)
Current gesture is: magnify (pinch / zoom)
(gesture ended)
Current gesture is: click (tap)
Current gesture is: rotate
(gesture ended)
█
```

Gesture considerations:

The Two finger Tap invokes a right click which is generated by default under the left stylus. This behaviour can be changed to generate the right click under the right most stylus. A time threshold is also configurable to specify the time in which a two finger tap can occur.

The Press and Tap invokes a right click which is generated by default under the first stylus. This behaviour can be changed to generate the right click under the second stylus.

In OS X Snow Leopard, four finger swipes typically invoke one of the "Expose" features, or invoke the application switcher. Unfortunately there's no supported way to programmatically activate these features, so UPDDGesturesmacosx posts keystrokes that trigger them. Since the hot key for the "Expose" feature can be configured, UPDDGesturesmacosx reads in the Apple hot key preferences to determine which keystroke is the correct one to press. We believe this works quite successfully and in our test these features get activated consistently. We are keen to find out if it works consistently for our users – any feedback much appreciated!

Single Touch gestures

Mac OS gestures utilise 2 or more touches. However, in some circumstances, user may wish to map single touches to simulate flicks and swipes, especially when using single touch touch screens. The setting 'singleTouchMode = on' in the [setting file](#) invokes this mode of operation.

In this mode a "press" gesture can still be used for a left mouse button click and drag, but this function can be disabled by setting leftClickThreshold" to 0.

iOS simulator

The iOS simulator allows applications built for iOS (such as the iPhone, iPad) to be developed and tested on an iMac system. To test gestures in this environment you normally hold down the alt/apple key on the keyboard and use a mouse. For users wishing to test dual touch gestures with a dual/multi touch touch screen we have introduced an option in the gesture engine to run in 'iOS simulation mode'. The setting 'iosSimulatorMode = on' in the [setting file](#) invokes this mode of operation.

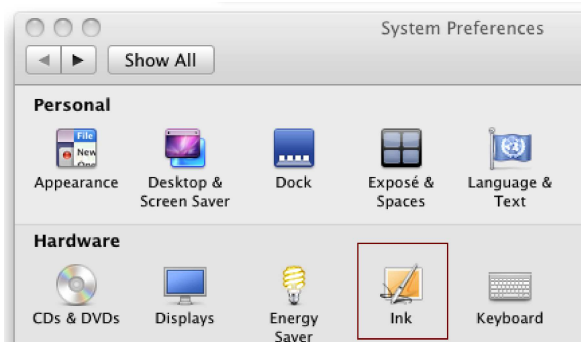
When running in iOS simulator mode please note the following:

1. **Important note:** There's a setting in Mac OS X that allows applications to use "accessibility" features for interacting with windows and other elements on the screen and it **must be enabled** for the gestures to work in an iOS Simulator mode. Here's how you turn it on:
 1. In the Apple menu, pick System Preferences
 2. In the System Preferences window, pick "Universal Access"
 3. Make sure the button labeled "Enable access for assistive devices" is checked
2. When starting a two finger gesture, it was necessary to send an event releasing the first finger before sending an event to press both fingers down. This didn't have any noticeable effect in our tests with iOS apps.
3. At the start of a two finger gesture there will be a little visual "blip". This is because the mouse is being repositioned so that the touches in the iOS Simulator match the touches on the touch-screen.
4. It is difficult to send the exact movement of both fingers into the simulator, so it's possible for the touches in the simulator to diverge slightly from the touches on the touch screen. However, performing individual pinch, rotate, and two finger drag gestures works as expected and this didn't have a noticeable effect in the tests we performed.

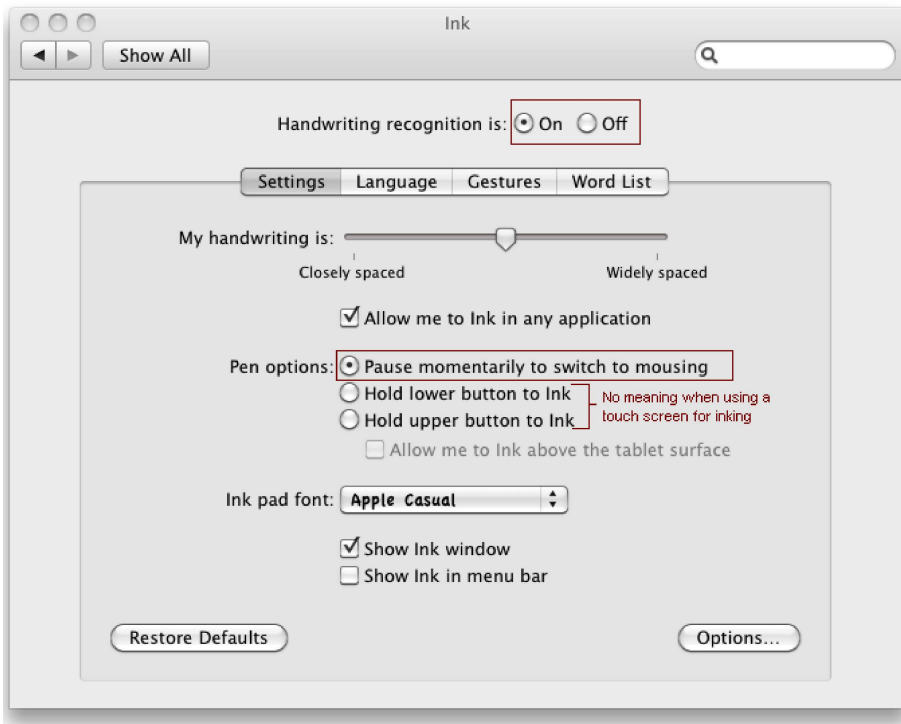
Inking specifics

Inking allows drawings and hand writing on tablet type devices to interact with applications. When a real or virtual tablet is seen by the OS the Inking function is enabled. With the latest version of the gesture application, available since 14/9/11, the inking function is also enabled and touch data is passed to the tablet interface. Real tablets pass more data than the X and Y co-ordinates, such as stylus angle, but when touch is being used this type of data has a fixed value.

After installing the software the Inking option is shown in the System preferences:



Launch the Ink settings panel to enable Hand recognition

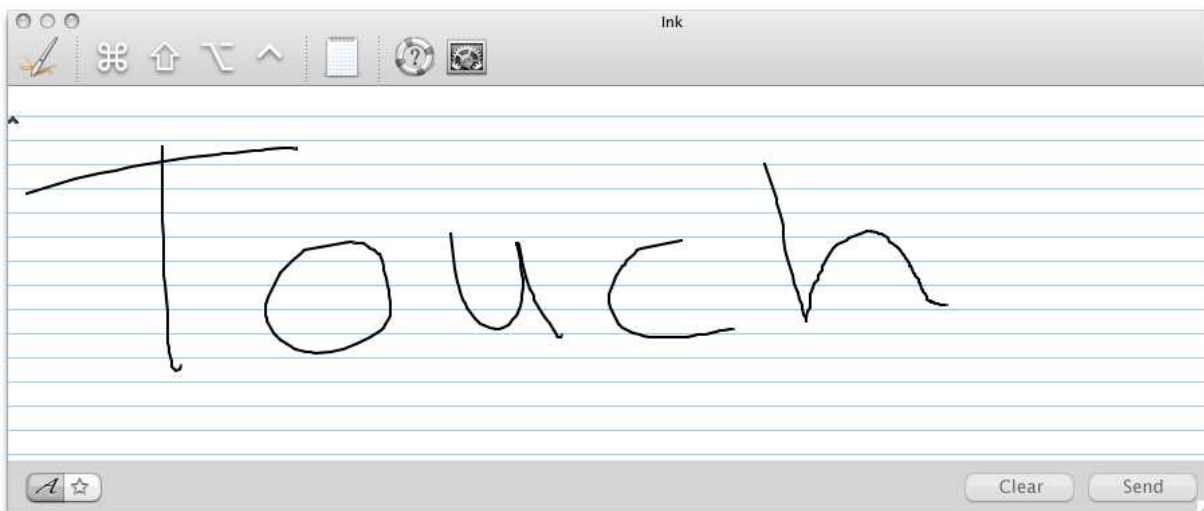


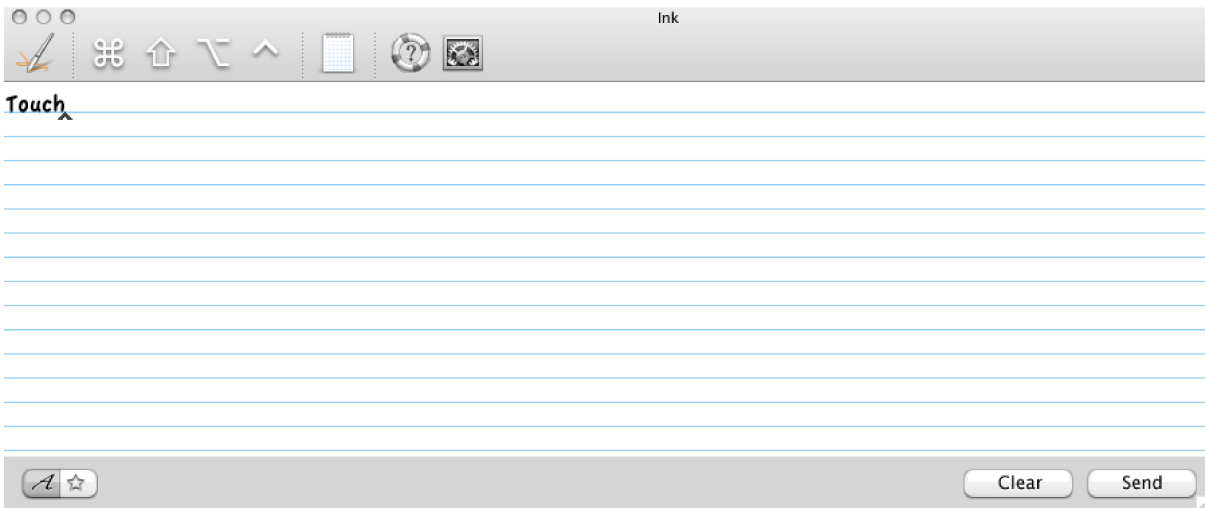
Once enabled the Ink floating windows will be displayed



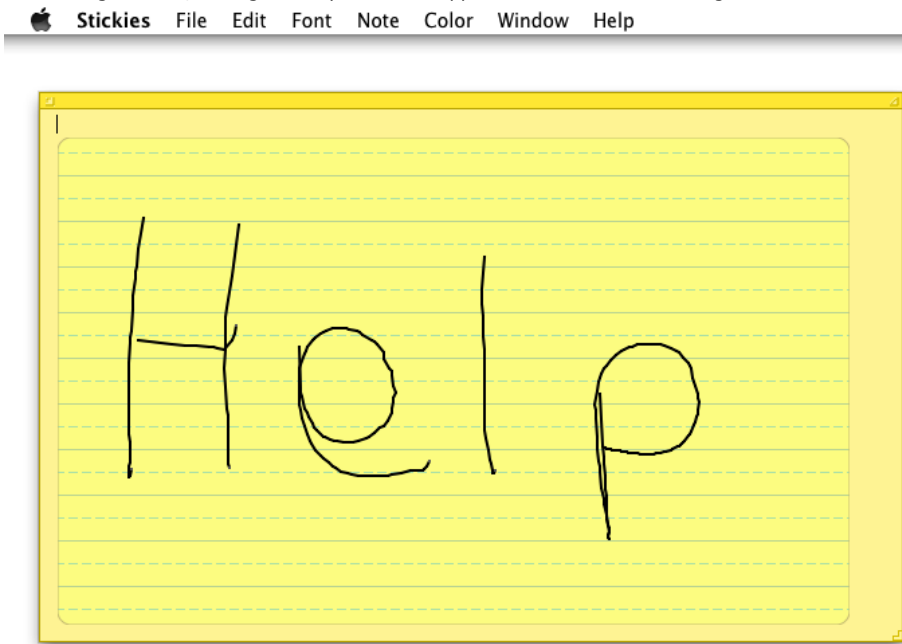
Activating handwriting recognition displays the Ink floating window. Whenever you want to enter text or drawing with your tablet, click the paper tablet icon in the Ink window. The A button in the lower left corner lets you enter text. The star button lets you draw. Clicking the Send button copies what ever you write or draw into the active application.

In the following example the touch screen has been used to write "Touch" on to the Inking paper and has been translated ready to be sent to the waiting application:

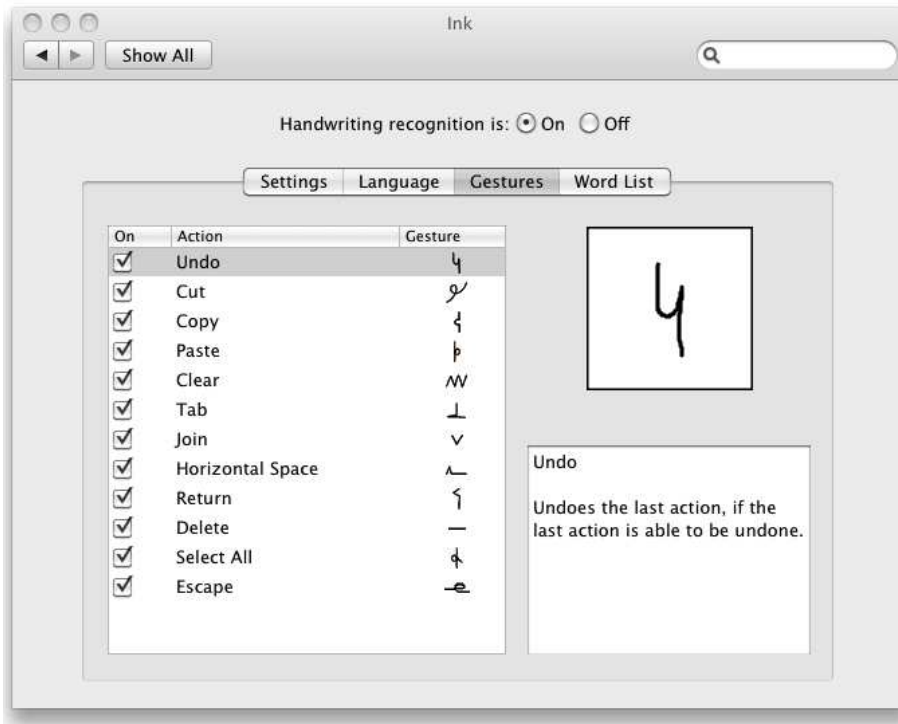




With Inking enabled, writing into any ink aware application will invoke an inking area in which to write, as in this example :



In addition to hand writing recognition and drawing, gestures can be used to perform various app functions, as listed below:



Limitations

Given that the UPDD inking function is implemented at a software level and does not create a virtual tablet device there may be some Inking applications that do not enable their inking capabilities due to the lack of a real tablet device on the system.

Further, given that there is no dedicated 'tablet stylus' in use the "hold lower button to ink" and "hold upper button to ink" settings have no meaning when inking with UPDD.

Gesture and inking settings

A number of settings are held in a settings file, gesture.ini. Edit this file as appropriate, the usage of which is as follows:

Setting	Meaning	Value
Gestures	Indicates if the gesture interface is to be utilised.	On or off
Inking	Indicates if the inking interface is to be utilised.	On or off
Monitor	Indicates the monitor in use for the touch screen. This is a temporary measure as this will be automatically determined when the gestures are fully integrated with UPDD.	Monitor number
Rotate	Indicates if the rotate gesture is to be processed. If using 2 finger touch gestures but the rotate gesture is never required this setting can be set to disable the rotate gesture. This is useful where 2 finger swipes may occasionally be processed as a rotate due to slight rotational tendencies of the finger movements. Defaults to On.	On or Off
ScrollSpeed	Indicates the scroll speed with the value being multiplied on to the normal scroll speed? Useful when working with large screen resolutions whereby scrolling is too fast. Defaults to 1.0.	n.n
leftClickWaitTime	By default the first touch is registered as a cursor move to the point of touch. If movement occurs a pen down is generated to produce a single touch drag. If the stylus is removed from the touch screen with no movement a mouse click (pen down /up) occurs at the point of release. If additional styluses are used whilst the first touch is down then the gesture engine performs gesture analysis to try and determine what gesture is being performed. In the case where single touches are to generate a pen down (followed by a pen up when the stylus is removed) then this setting indicates the stationary time threshold for the first touch to generate a pen down, defined in milliseconds. This is useful if you are using applications that need a pen down when single touch is being used without movement, such as a repeat key on a virtual keyboard. If other styluses make contact whilst the pen is down, a pen up is performed and then the gesture engine performs gesture analysis. Defaults to 500. The leftClickWaitTime value determines how long that length of time is in milliseconds and 0 removes the press gesture altogether. Note that when this is set to 0 with singleTouchMode on, it will no be longer possible to perform "left click and drags" with the touch screen, only left clicks.	Milliseconds Or 0
TwoFingerTapThreshold	Determines the amount of time in milliseconds in which a two finger tap can occur (since a two finger tap is by nature a fast action, and shouldn't be confused with a "press and tap").	Milliseconds
RightMostFingerForRightClick	Changes right click behavior to use the rightmost finger rather	On or off

TappingFingerForRightClick	than the first. By default this is disabled. When enabled (1), the second / tapping finger in a "press and tap" gesture will determine the location of the right click. Otherwise, the first finger's location is used.	On or off
iosSimulatorMode	Run in iOS simulator mode. Processes single and dual touches differently if they occur over an iOS Simulator window. Defaults to on.	On or off
iosSimulatorAccurateMultiTouch	When on, the two touches performed on the touch screen exactly correspond to the two touches received in the iOS Simulator. This is accomplished through some trickery, so if you experience any issues performing two finger gestures in the iOS Simulator turning this feature off may improve things. (When off, the two touches in the simulator follow the gesture movements but are repositioned around the center of the iOS Simulator window rather than directly under the point of touch.) Defaults to on.	On or off
iosSimulatorSingleTouchWaitTime	It's necessary to briefly withhold single touches from the iOS Simulator so that two finger gestures can be detected. This value is the amount of time in milliseconds a single touch is withheld before being sent to the simulator. Defaults to 150.	Milliseconds
iosSimulatorSingleTouchMovement	Single touches that are being withheld can also be sent to the simulator if they move an adequate number of pixels on the screen, as defined by this setting. Defaults to 10. Single touches on the iOS Simulator have a similar behavior to single touches elsewhere, in that UPDDGestures doesn't generate a pen down / mouse down event until either the touch has moved or if it's been held down for a certain amount of time. So the touch is "withheld" until one of those two conditions is met. iosSimulatorSingleTouchWaitTime setting is analogous to leftClickWaitTime. We introduced specific "iosSimulatorSingleTouchWaitTime" and "iosSimulatorSingleTouchMovement" settings because we noticed that the iOS Simulator needed its own settings for how to correctly respond to single touches, independent of the rest of UPDD Gestures.	Pixel value
SingleTouchMode	Puts UPDD Gestures in a mode where touches work similarly to an Apple iPad. When on, a "tap" gesture produces a left clicks, a "drag" gesture will scroll, and a "press" gesture will cause a "mouse down" depending on the value of leftClickWaitTime. Defaults to off.	On or off
hideCursorDuringTouches	If enabled, hides the cursor during touches. The cursor is shown again if a mouse or trackpad is used to move the mouse. Set to off by default so the cursor is shown.	On or off

Really important point: When using gestures in Mac OS X the gestures are processed by the application window under the mouse cursor. Dual and multi-touch gestures can be performed on any part of the touch screen but will be processed by the target area. So, for example, if you have a Preview window open and the cursor is in the viewing area then the area will respond to gestures. If the cursor is on the Preview dialog but not in the view area then gestures will be ignored.

Driver considerations

The gesture application turns off the UPDD mouse interface and receives all touch data. There are a number of UPDD utilities that re-enable the mouse interface when they terminate, such as calibration and test. Until we change these utilities to retain the current mouse port state they should only be used with the gesture application disabled.

A users reported that 'in Lion, moving the "mouse arrow" to the top of the screen may not reveal the Mac OS window bar necessary to get out of Full Screen mode necessitating a need of a proper mouse'. This may be because the cursor, being under the stylus, is stopping short of the top of the screen. You can force the cursor to the top by using Edge Acceleration settings in the UPDD Console, Properties dialog described [here](#). Look for the Advanced settings.

Linux

To follow.

Future gesture development

The standalone application utilised in Mac OS X calculates individual gestures from the incoming stylus data streams and as such can be considered a 'gesture engine'. It is our intention in a future release of UPDD build this gesture engine in the driver such that in all cases this gesture information is made available on UPDD's API so that there is a common interface across all platforms supported by the driver (all individual stylus information is also available).

Contact

For further information or technical assistance please email the technical support team at technical@touch-base.com.