TOUCH BASE documentation

UPDD – Gesture and Inking

Revision 1.42 – 12th Jan 2015
www.touch-base.com\documentation\General

Overview Windows Win CE Mac OS X Linux Future Development

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	Stylus data pass thro' a virtual touch device registered as having multi-touch capabilities.	Enabled if the distribution contains 'uinput' support.	
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 dependent 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 which supports gestures and inking in Snow Leopard 10.6, Lion 10.7, Mountain Lion 10.8, Mavericks 10.9 and Yosemite 10.10.

Development history can be viewed from the menu bar item and can also be viewed here.

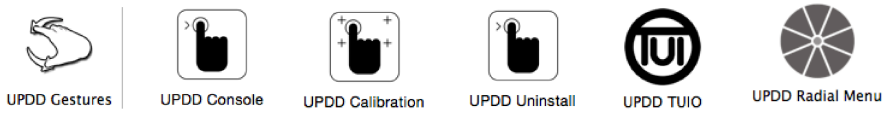
Installation and running the gesture software

To utilise gestures and inking in the Mac environment you need to simply download the gesture .zip file below that matches your release of the driver (Stable or Beta) and expand the compressed file to create the application file 'UPDD Gestures':

Version	UPDD 5.0.2	UPDD 5.1.xxx	Date	Changes since last build – full development history here
2.0.42	Stable	Stable	15th Sept 2014	Added optional visualisation for press gestures Can now open UPDD Gestures Finder icon to open the settings window Worked around crash when disabling the menu bar icon
2.0.50	Beta	Beta	12th Jan 2015	Two finger rotate settings now listed under the two finger Pinch and Spread section. Added - Associate action with sliding in from the left and right edges with a single fin Added - Cater for gesture dialog help files Fix - Performing Two Finger Swipes from the screen edge could simultaneously trigge to Two Finger Drag Change - UPDD Gestures crash guard will no longer restart UPDD Gestures; it will on regular UPDD mouse emulation; fixes issue where "UPDD Gestures is already running erroneously Added: Gesture action for toggling the keyboard viewer

It is highly recommended that this file is moved to the standard Mac OS X Utilities folder along with the other UPDD Mac

applications.



Simply click on the application to run gestures. When running, a gesture menu bar item, if enabled, will be shown in the menu bar .

If this is enabled but does not appear see the troubleshooting section below.

If the menu bar item is disabled but the gesture software is loaded then running the gesture software again will invoke the gesture settings dialog.

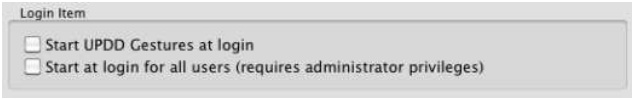
Important installation notes:

1. **Retain original name**
   If you have previously downloaded and expanded the gesture software and the original file still exists in your download folder then the new file will have a number appended to the name, e.g. UPDD Gestures 2.app. If this is the case please ensure that when moved to the Utilities folder the app is renamed back to UPDD Gestures.app.

2. **Requires production version of the driver**
   Gesture functionality will only work on a production version of the driver. If you run the software on an evaluation version of the driver the following message will be displayed: "The UPDD Gestures software requires a licensed version of the UPDD driver. The driver currently installed is an evaluation version. To purchase the full version of the driver, please contact sales@touch-base.com.

3. **Automatically installed with driver**
   Note that some UPDD driver installs will automatically install the gesture software and create a startup item to invoke the software at user logon. As of Sept 13 the installer will create a global start up item for all user accounts – not just the foremost user account.

4. **Invoking gesture software at startup**
   The easiest way to get UPDDGestures to start at boot is the following:
   - Enable via the Gesture GUI
     1. Invoke the Gesture GUI via the Gesture Menu item, Settings entry.
     2. Select the Other Settings dialog
     3. Check the 'Start UPDD Gestures at login' or "Start at login for all users" option:



     The "all users" option will create a gesture startup item for all current and future users.
     Note – this is the default setting for Gestures that are installed as part of the driver installation, see Note 3 above.

     or

   - Manually set up a start up item
     1. Open System Preferences (in the Apple menu)
     2. In the System Preferences window, select "Users & Groups"
     3. Select the "Login Items" section
     4. Add "Open me to start UPDD Gestures" to the login items.
     (You can drag its icon onto the list, or click the "+" button and add it using a file browser dialog.).

Gesture uninstall

There are 2 simple steps to manually uninstall the gesture software:

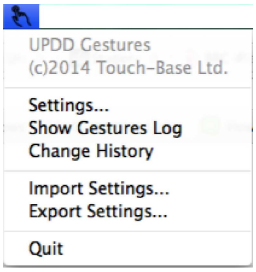
1. Remove the start up item if gestures have been configured to be invoked at start up:
   - Invoke the Gesture GUI via the Gesture Menu item, Settings entry.
   - Select the Other Settings dialog
   - Uncheck the 'Start UPDD Gestures at login' option

2. Delete the application
   - Locate and drag the application to the trash can.

Menu Bar item

Once invoked, and if enabled, a Menu Bar icon indicates that the gesture application is loaded and running and can be used to quit the gesture touch function.

Since version 2.0.14 the Menu Bar item can be optionally disabled and with some OEM versions of the gesture software this is the default state so no Menu Bar item is shown. The Menu Bar item can be enabled / disabled as required in the gesture settings dialog.



Menu Bar icon	Introduced
Load settings dialog	2.0.5
Show gesture log – displays calculated gesture in log file	
View change history	2.0.10
Import settings from a '.ini' file	2.0.34
Export settings to a '.ini' file	2.0.34
Quit Gestures	

Gesture Overview

The gesture software disables the driver's own posting of single touch data into the system and receives all touch from the touch device via the driver's API interface. If gestures software is quit then touch will revert back to single touch via the driver's own system interface. Should gestures crash a background process named UPDDGesturesCrashGuard will automatically restart the program and ensures that mouse emulation is re-enabled after Gestures terminates so that touches won't become unresponsive.

The Gesture software posts ***all*** received touches as OS X's native touch events and as TUIO touches if configured (and UPDD TUIO server is running), regardless of whether there is a gesture being performed, as the gestures are more or less separate from the touches themselves.

If gesture processing is not required (you do not require Gestures triggering any actions for the performed gestures on the touch screen), you can set the gestures to "No action", and applications will still be able to receive the individual touches.

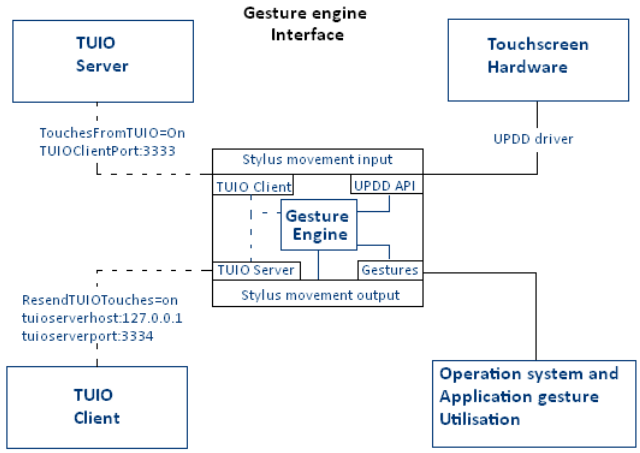
There is also a setting to request that touch data is posted into the system as tablet events, allowing a touch device to be used with Inking and other tablet-related features. It only affects mouse events produced by Gestures, though, since in OS X a tablet event is also a mouse event.

Tablet/Pen device inputs supported by the driver/gestures are always passed into the system as tablet events. Pen nib and co-ordinate information bypasses the gesture engine, i.e. is not processed for gesture consideration.

Application schematic

As described in the overview above, the gesture software interacts with the core UPDD driver or TUIO Server to receive all touches, calculates the gesture being performed, initiates the associated action and then injects the touches into the OS as native touch events and also optionally pass mouse actions as tablet events.

The schematic for the gesture interface is as follows:



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. The action associated with each gesture can be defined in the gesture settings dialog. 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 number of videos have been posted on the web from end uses such as this one here.

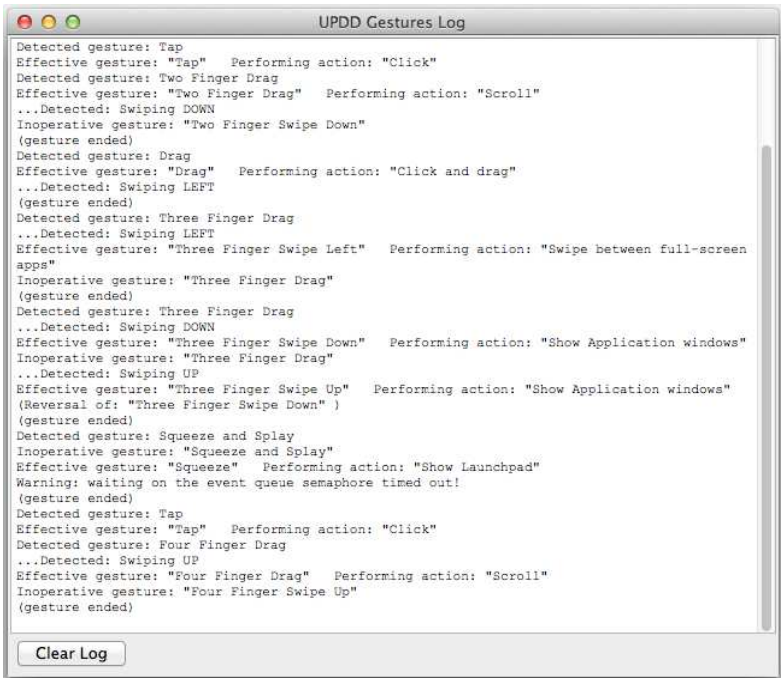
A guide to Lion gestures in PDF / Printable format is available here. Mountain Lion gestures are described here. Gesture videos can be seen here.

Saving and importing gesture configurations

Gesture settings can be saved to and imported from a file, [filename].ini. This allows for gesture configuration sets to be imported as required for specific desktop / application use or to share configurations.

Gesture log

To view the gestures being calculated by the gesture engine in real-time invoke the Show Gestures Log option in the menu bar:



For each "Detected gesture" there should be a corresponding "Effective gesture" indicating the gesture performed.

The example log above is from version 2.0.23 with additional logging information implemented to track down any reported occurrences where some "detected" gestures don't have an "effective" gesture. E.g. In the case of three finger drags and swipes, it should choose an "effective" gesture for one or the other every single time but we have had reported incidences of no effective gesture being selected. Should you experience this please send log output along with your support email.

A more detailed explanation follows of the log entries:

Detected gesture: *** ---> means the analysis detected a basic gesture, like three finger drag.

...Detected: *** ---> reveals new information from the analysis about the current gesture, e.g. the three finger drag is swiping left

Effective gesture: *** Performing action: *** ---> is which of the configurable gestures in the GUI is matched to the gesture from the analysis, and which action is being performed, e.g. it picks "three finger swipe left" instead of "three finger drag" and performs the appropriate action

Inoperative gesture: *** ---> is which of the configurable gestures could have been matched to the analysis but wasn't because of the user's settings, e.g. "three finger drag" because the effective gesture was "three finger swipe left"

(gesture ended) ---> means the analysis detected the end of a gesture

Gesture considerations

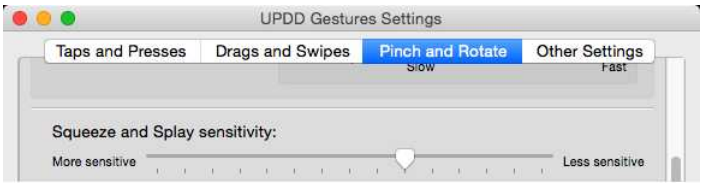
The Two finger Tap invokes a right click which is generated by default under the left stylus. This behaviour can be changed to generate the right click under the right most stylus. A time threshold is also configurable to specify the time in which a two finger tap can occur.

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!

Squeeze and Splay sensitivity

Some users reported difficulty in activating actions associated with 4 and 5 finger squeeze and splay gestures. The gesture log would show that the gesture had been recognised but that the gesture hadn't proceeded far enough along to be considered a full squeeze or splay hence the action was not invoked. To overcome this issue we have introduced in version 2.0.46 a sensitivity control that can be used to fine tune the decision process!



Squeeze and splay sensitivity determines how much your fingers must contract or expand during a squeeze/splay gesture to have it be detected as either a squeeze gesture or a splay gesture respectively. This comes into play after Gestures has detected that the current gesture is going to be a squeeze or splay gesture (as opposed to a different gesture like a four finger drag), but before there's been adequate movement to determine which of a squeeze or a splay it is. Due to it involving several fingers moving in different directions, squeeze/splay gestures are a little bit more difficult to detect than

the other gestures, hence requiring more movement.

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.

With the latest gesture software each gesture can now be configured on a gesture-by-gesture basis. A typical gesture configuration for single touch touch screens would be as follows:

Set Tap -> Click*

Set Press -> Click and drag*

Set Drag -> Scroll

* note for these gestures that action is the default.

Check the "Disable multitouch gestures" option in the "Other Settings" section.

Finally, when configuring Gestures to be used with single touch we recommend the gesture setting "milliseconds for Press to occur" should be between 500-1000 milliseconds as you may experience an immediate click event on finger press when trying to perform a drag gesture.

Observed gesture action delay

These issues were initially raised by users using Avid Pro Tools and Proppelerhead Reason when using small audio faders or adjusting some controls to adjust their values.

Tap

Gestures cannot be certain a single touch is actually a tap until the finger has been released. You may observe a slight delay when tapping on buttons that's not present when Gestures isn't running or when using a mouse.

Drag

By default, a single touch needs to move a minimum of 10 screen pixels in order for Gestures to detect it as a "drag" gesture so applications won't receive a mouse event until the 10 pixels have been traversed, creating the slight lag. With version 2.0.22 the pixel threshold is now configurable in the settings program, Drags and Swipes dialog and a smaller value reduces the lag or can eliminate it altogether.

Note: As of version 2.0.31 we have added a setting for gesture detection sensitivity to adjust how accurate or responsive of the gesture detection process.

Understanding Swipes and drags

As far as the gesture analysis function is concerned a swipe is fundamentally a drag gesture that's moving in one of the cardinal directions, which is to say swipes are a subset of drag gestures. The speed of the fingers or the time for which they contact the touch screen isn't considered, so a quick temp swipe on the screen and a slower, well defined movement should both be detected as a drag, and depending on the direction of movement also a swipe. It is for the reason the Gestures GUI has a setting for choosing between the two.

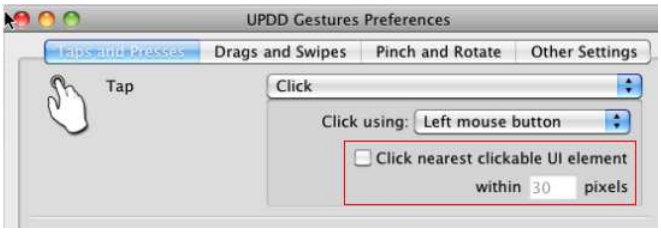
What's going on behind the scenes in Gestures is that when the analysis determines that the touches are performing some manner of drag gesture, it also reports the swipe direction, if one is detected. This way an app using the analysis can decide which specific gesture it wants to respond to.

We designed the analysis to be more general purpose, reporting all of the relevant information about a current gesture. There's separate logic in gestures to take the results of the analysis and decide which of the specific gestures in the GUI to respond to (like swipes vs. drags) based on the current gesture settings.

Note: As of version 2.0.32 we both fixed a bug where scrolling using slow moving gestures would produce jerky, undesirable motion and generally improved feel and smoothness of scrolling and scrolling momentum so that it is now more similar to using an Apple trackpad.

Clickable UI element feature

Starting with gesture release 2.0.27 we have introduced an option for each click action to locate the nearest clickable UI element within a defined pixel radius to the point of touch and click it.



This is currently disabled by default but nevertheless is a very useful feature and improves touch usability, especially when dealing with small controls not designed for touch interface.

Please note with this enabled the gesture software deliberately distorts tap and press touch locations that occur near a UI control element.

This option is very useful for selecting small UI controls difficult to accurately touch, such as the dialog close, minimise and maximise UI buttons, as shown, and similar desktop and application controls

Notes when using this feature:

1. This feature makes use of the Accessibility API and there are a few quirks when using this API interface which we have to code around. This feature has mainly be been tested in 10.9.

2. This feature will activate any GUI button within a certain radius, even under the dialog being touched, and this behaviour is by design. The click nearest feature will consider every UI element within a radius of the touch, regardless of what window it is in and what window contains the touch. This caters for the situation whereby a click is required on a button near the edge of a window, like for example the close or minimize buttons. If the nearest UI element is restricted to only the window underneath of the actual touch, then touching just slightly above the top edge won't result in any of those buttons getting clicked, and it's pretty easy to miss. We understand that not restricting the search to within the touched window can sometimes yield undesirable results. Further, GUI buttons can be enabled or disabled and Gestures restricts clicks to enabled buttons only.

3. In our tests this feature really improved the touch experience and makes the selection of small buttons and UI elements a lot easier. However, this is an 'unproven' feature and will probably have a few eccentricities and issues to iron out as we have had to code in some decision making into Gestures in order for it to intelligently deteremine what it should click, when, and where. We suspect there's probably going to be a few edge cases where the wrong item gets clicked, or something doesn't get clicked that should. We would really appreciate any constructive feedback you may have when using this feature that may help us improve functionality in future releases.

4. We believe there are a few improvements we could make already which may come in future releases. Specifically, this feature doesn't work as well when browsing the web in Safari, as Safari reports practically every element in the

browser window as being clickable, even when they aren't. We think we could improve its detection of 'real' clickable items to make it work properly in Safari, including finding text links.

5. We also briefly tested in Logic Pro X and found that it works partially as Logic Pro X supports the Accessibility API, but it reports odd information about some of its UI elements, similar to Safari. We think we could code round these issues and may do so in a future release.

6. Finally, this feature won't readily work in any applications that do not support Apple's accessibility API, such as Chrome, Firefox, and Qt applications build using a Qt version prior to 5.2. It'll only recognize a small subset of their UI elements, like the menu bar.

Pan and Zooming

Starting with version 2.0.44 we have added actions to pan and zoom. This has been tested in 10.6 thro' 10.10 and appears to work well.

To allow these actions to be included as default actions without replacing existing ones we have split the squeeze and splay gestures into two separate gestures depending on whether four or five fingers are used and set it up so that four finger squeeze and splay zooms the screen in and out and four finger drags pans the screen around. However, we do think it works best though when using pinch and spread gestures for zooming and one-finger drags for panning so we recommend configuring it that way for best results. Since version 2.0.47 we have you can now hold the control key and use pinch/expand and two finger drag gestures to zoom and pan the screen respectively

Correct zoom operation requires the OS X Zooming feature to be configured correctly in the Zoom section of the Accessibility system preferences (in OS X 10.8+) or the Universal Access system preferences (in OS X 10.6 and 10.7). The most important settings are "Use scroll gesture with modifier keys to zoom", "Zoom style", and "When zoomed, the screen image moves...". The last one is accessed by pressing the "More Options..." button.

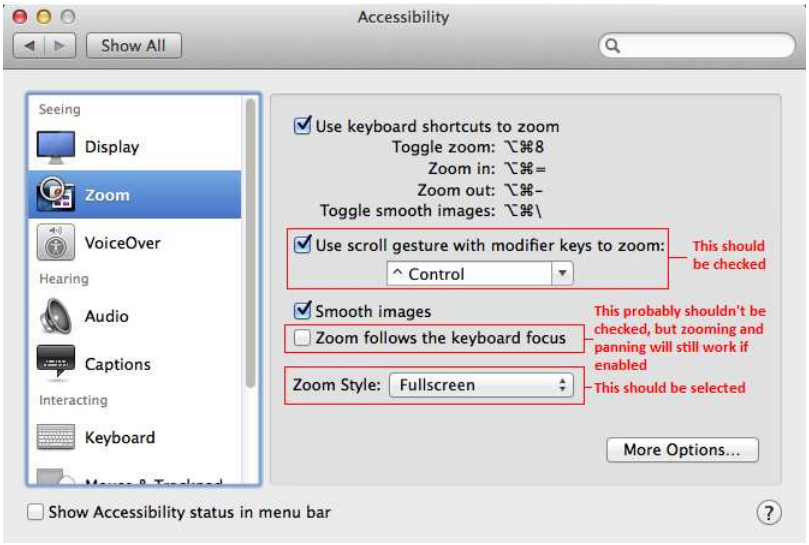
"Use scroll gesture with modifier keys to zoom" doesn't need to be enabled for zooming to work, but we did find that in one of our systems, we needed to turn it on before the zoom action would have any effect. This appears to be inconsistent but we recommend to have this option on.

Zoom style can be set to either "Fullscreen", where zooming will enlarge the image for the entire screen, or "Picture-in-picture", which does it just for a small section. The zoom action can be used with either setting, but panning can only be used when it's set to "Full screen".

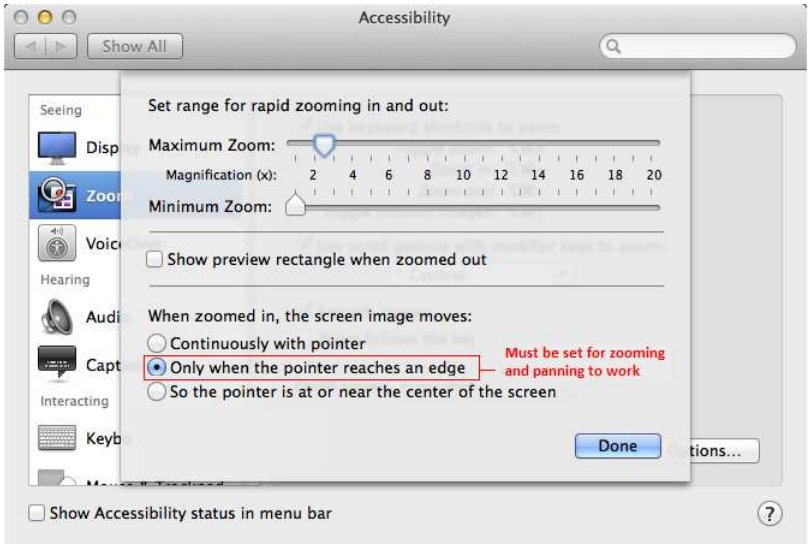
When zoom style is set to "Fullscreen", "When zoomed, the screen image moves..." becomes available by pressing the "More Options..." button. As we discussed in previous emails, Gestures only works when it's set to "Only when the pointer reaches an edge" and that continues to be the case for the zoom and panning actions.

There's also a setting called "zoom follows keyboard focus" which we think has the potential to interfere with using zooming and panning, but some users might prefer to keep it on. You need to experiment if you want this on.

These screenshots show the correct settings in OS X 10.9 (we think these apply for OS X 10.8 and 10.10 as well).



Regarding the above setting 'Zoom follows the keyboard focus' one customer reported that with this unchecked in Yosemite 10:10 the 'cursor will jump randomly on higher zoom levels'. Checking this setting 'fixed' the problem.



Gestures will catch if any of these are configured incorrectly, disable the zoom and panning actions in the settings window, and display this message: "Cannot perform screen zooming with current Accessibility zoom settings. Consult the UPDD Gestures documentation for instructions for configuring Accessibility correctly."

Due to the way screen panning works in OS X with multiple monitors, screen panning will work much better if the entire desktop (spanning all connected displays) is perfectly rectangular. It can be used with display configurations where two or more displays aren't set to the same resolution, or are arranged such that their sides aren't exactly aligned, but the panning motion will be a little bit jerkier, and in some rare situations the screen won't be able to be panned in a particular direction. The less rectangular the desktop is, the more potential there is for trouble. That said, it does still seem to work pretty consistently even when I set my display configuration to something very odd (e.g. the two displays are wildly offset from each other). And this doesn't apply in the case of a single display -- that works fine with no caveats.

Screen panning works by moving the cursor to the edge of the screen, causing it to be pushed around according to the user's touches. So when "hide cursor during touches" is turned off, the cursor will be visible jumping around to various positions on the screen periphery. A future release will make the cursor hidden when the screen is being panned.

Given this feature uses some hacky tricks in its implementation, especially for screen panning, and we expect there will be a few bugs to iron out, and we've already found a few. One is that it's possible to pan the screen (causing the cursor to jump around) when the screen is fully zoomed out even though it has no effect. The other is that when resuming use of the mouse after panning the screen, the cursor doesn't always stay where it was last placed by Gestures. These will be addressed in a future release.

Onscreen keyboard

Starting with gesture release 2.0.12 you can configure the system onscreen keyboard to be automatically invoked.

Using the Gesture settings program, Other Settings tab, you can configure the keyboard to be displayed when a text input field has focus or restrict the usage to secure text fields only:

Unfortunately this feature has some limitations:

- It does not work at system login – we may be able to overcome this limitation if required – please contact us.
- It only works with applications that support OS X's accessibility features. Two prominent examples of apps that don't support it are Chrome and Firefox. Fortunately it works great in Safari. Also, all Qt apps currently don't support OS X accessibility, though apparently Qt 5.1 addresses this issue.

In order for this feature to work in OS X 10.6, the "Show Keyboard & Character Viewers in menu bar" button must be checked in the Keyboard system preferences.

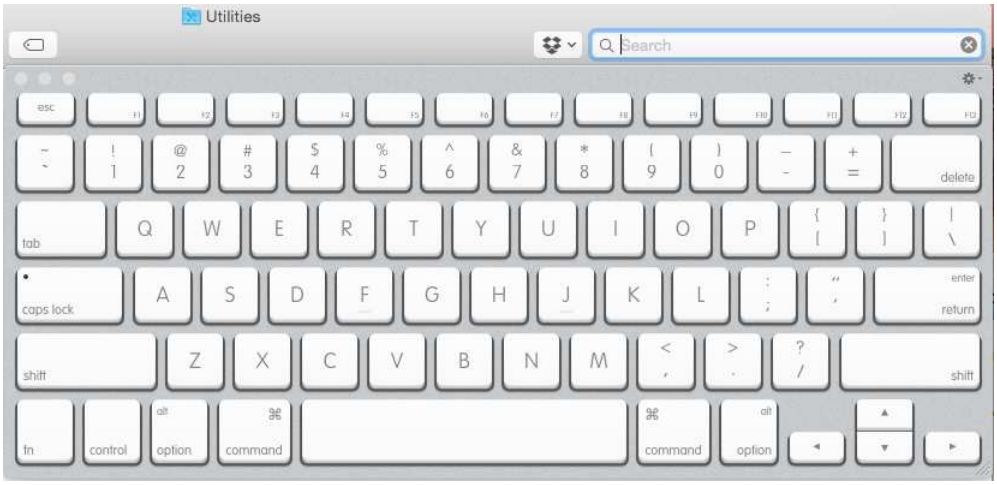
Example:

A user reported that is that the Mac OS X native pop-up keyboard doesn't latch control buttons i.e. Shift, Ctrl, Command and Alt keys. An on-line discussion suggested activating Sticky Keys in preferences/universal access/keyboard. If this function is required consider using the KeyUp keyboard below.

KeyUp by Irradiated Software

With gesture version 2.0.49 you can now invoke KeyUp from Irradiated Software.

The KeyUp software must be located in the Application folder.



Hide Mouse Cursor

The gesture software can be used to hide the mouse cursor. The feature uses an undocumented API (presumably a feature not encouraged by Apple) to hide the cursor so there are no guarantees it works 100%. This feature can be enabled in the gesture settings dialog, Other Settings tab, 'Hide Mouse Cursor during touches' checkbox. The cursor is enabled as soon as the cursor is moved by something other than the touch screen, including a mouse or trackpad.

We have found that in some circumstances the cursor is made visible, such as switching apps or touching into flash movies or moving over the system dock, so since gesture version 2.0.21 the cursor is periodically hidden if it is found to be visible. The periodic test to check the cursor is started when the "hide cursor during touches" feature is enabled, but it will only hide the cursor if it is supposed to be hidden, such that if it is being shown due to mouse/trackpad movement then the cursor will stay visible until the next time a touch begins.

Unfortunately, if an application constantly forces the cursor to be visible, such as if the cursor is over a flash movie, then the cursor will stay visible.

Trackpad mimicking

Some applications check for the device id of the device generating the gesture before processing the gesture. One such feature that can be enabled if a real track pad is connected is the Asian Trackpad handwriting feature. If a trackpad is connected and the gesture is performed on the trackpad only then will this feature work.

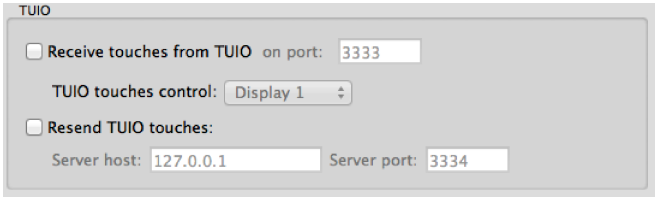
Since gesture version 2.0.21 the gesture software will utilise the same device id ***of a connected trackpad*** to cater for any features or applications that are testing the source of the gesture. If no trackpad is found then gestures uses its own 'dummy' device id. Gestures can be forced to use its own id, even if a trackpad is connected, if the setting gesturedefaultdeviceid=1 is defined in the UPDD Setting file.

Note: We observed that using OS X's "trackpad handwriting" feature with Gestures works much better under OS X 10.9 than under OS X 10.8. Under 10.8 we found trackpad handwriting with Gestures to be inconsistent and occasionally it won't detect when a finger has been lifted. Under 10.9.1 this did not appear to be the case.

TUIO Server interface

The gesture software can be configured to receive touch data from a TUIO server and generate gesture functions. This is useful on systems where touch co-ordinate and stylus data is being delivered on the TUIO interface, such as a touch screen device that may not be supported by the UPDD driver but does have Mac OS X software to create a TUIO server interface. This setup would normally be created to run TUIO client applications.

Using the Gesture settings program, Other Settings tab, you can setup the TUIO interface as shown below:



When used to work with a touch device not directly supported by our UPDD driver, the driver must still be installed on the system to satisfy the gesture software that it is working with our UPDD driver.

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 touch gestures with a dual/multi touch touch screen we have introduced an option in the gesture engine to run in 'iOS simulation mode'. This option can be enabled in the gesture settings dialog.

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 to 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.

Pen Support

Since version 2.0.24 the gesture extension has added support for Pen devices that present proximity, left click (via nib) Right click barrel button, pressure and eraser features. Utilising the driver only will support the Left (nib) and Right clicks but pressure and eraser support is currently built into the gesture extension software.

Tablet/Pen device inputs are always passed into the system as tablet events. Pen nib and co-ordinate information bypasses the gesture engine, i.e. is not processed for gesture consideration.

Currently there is no support for the pen upper side switch (as seen on Wacom pen type devices).

With release of 2.0.29 of gestures you can now define the min and max pressure range of the device in the UPDD driver setting file, tbupdd.ini. When these settings are not defined the gesture software calculates the maximum theoretical range based on the pressure definition of the device as configured in the driver. For Wacom / Surface Pro and Slate pens this is defined as 10 bits giving a theoretical range of 0 to 1023. However in data logs we have captured the actual range seems to be in the order of 900 to 100 (dropping immediately to 0 thereafter).

In UPDD driver version 5.1.x, available June 2014, you can use the TButils command line utility to define / update settings in the UPDD settings file.

Application considerations

QT – Cross platform development tool

Some multi-touch applications use a cross platform development tool called QT and use the QTouchEvents interface to receive system-level touches. Unfortunately the standard way Qt determines the screen location of the touches in Mac OS X is incompatible with UPDDGestures: Qt assumes that the touches are coming from a trackpad since all system touches are assumed to originate from a trackpad. In that case, Qt has the touch start at the mouse cursor location (which is not what is needed in a touch screen environment), and the touch's movement speed is calculated using the physical dimensions of the trackpad. However, it won't get any dimensions since no trackpad is present. Instead, it calculates that the trackpad has a width and height of 0, and consequently the touches won't move anywhere.

For touch-enabled Qt applications to work with UPDDGestures they must use the normalized position of the touches, not the screen position. Sadly, we suspect most Qt apps use a touch's screen position.

One such popular multi-touch enabled application is Snowflake from NUITEQ. The developers of Snowflake are working on changing the interface to utilize the normalized position but until this change is made you will need to use our UPDD TUIO bridge to utilize Snowflake's TUIO interface. This has been tested and works well but it does mean that the gesture software cannot be used at the same time as the TUIO interface as it causes a phantom touch in Snowflake.

Mouse Events

Programmers reported that when using Gestures, the initial touch does not trigger MouseDown code within their application. Further investigation showed that if gestures was to invoke a mouse down on touch then there's no way to "cancel" a mouse down if a single touch ends up being a gesture without causing a click at the point of touch. Clicking at the point of touch, when performing, say a scroll, could cause highly irritating and confusing situations to occur -- all sorts of things on the screen will end up being unintentionally clicked! This is why Gestures withholds any mouse events until it's determined what gesture is being performed and hence a mouse down event will not immediately be triggered.

It is our recommendation that a developer should use the system-wide touch events that Gestures generates (rather than mouse events) to cause a 'mouse down' event when a touch is occurring. These touch events are delivered as soon as a touch occurs, so there's no delay, and they won't trigger the button to be clicked at inappropriate times.

Multi-touch browser applications

We are aware that UPDD Gestures cannot be used to control touch-enabled web pages. The problem is that OS X doesn't pass its system-level touch events to web browsers as HTML5 / javascript touches. So while these pages will work with an iPad (and similar) it can't work in OS X, even when using Apple's magic trackpad. However, as of 20th Aug 2013 we have expanded on some browser extension work, initially written by Boris Smus, to implement multi-touch TUIO interface in Chrome and Firefox browsers. This is described in full here.

JavaFX support

Starting with JavaFX 2.2, users can interact with JavaFX applications using touches and gestures on touch-enabled devices. Touches and gestures can involve a single point or multiple points of contact. The type of event that is generated is determined by the touch or type of gesture that the user makes. This link describes working with events from Touch-enabled devices. We have compiled and tested the Gesture Events example program with our Mac OS X gesture implementation and found it all to work as expected therefore any JavaFX touch event driven applications should work as expected.

Logic Pro X

A Logic Pro user reported they were trying to make logic learn commands that come from the touch screen.

We were able to control Logic Pro X using Gestures by assigning different gestures to the "Keystroke" action, and then setting the keystroke either to an existing key command for Logic, or a new keystroke, and then teaching it to Logic using its "Key Commands" editor.

Inking specifics

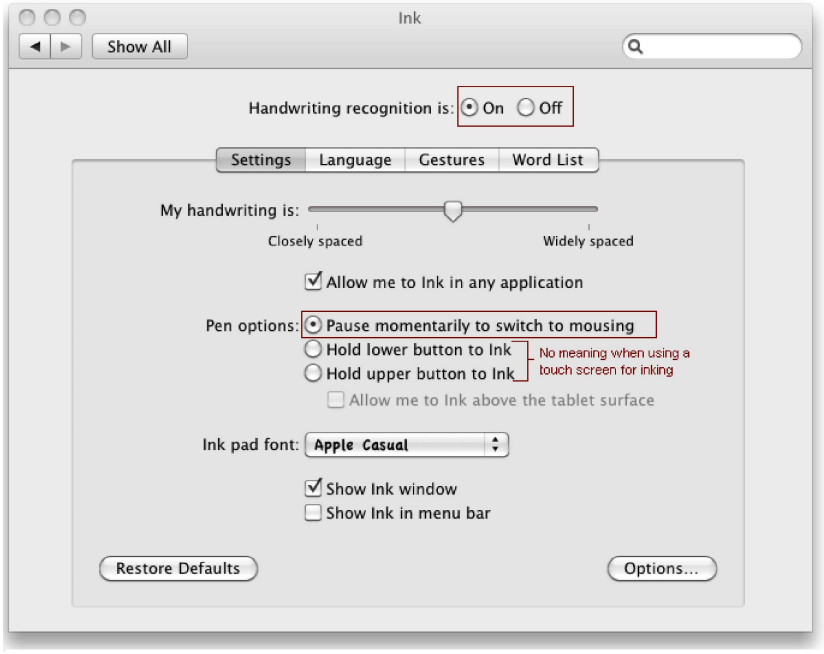
Inking allows drawings and hand writing on tablet type devices to interact with applications. When a real or virtual tablet is seen by the OS the Inking function is enabled. The Gesture software can also be configured to support the inking function whereby touch data is passed to the tablet interface. The tablet interface is enabled in the Gesture, Other Settings dialog, 'Touches simulate tablet input' option. If Inking is enabled, it can be used with Gestures through the "Click" and "Click and drag" actions.

After installing the software, and if Inking is available on the system, the Inking option is shown in the System preferences dialog.

Notes:

1. *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.*
2. *The "Ink" system preference pane usually only appears when a tablet is connected. However, if a tablet is not connected, and to make Inking configurable with Gestures, we make a copy of the Ink preference pane that appears only when the original one is hidden. In this situation the Ink will appear in the "Other" section of System Preferences rather than the "Hardware" section and the gesture log will show the entry "Inking enabled: installing UPDD Inking pref pane if it's not there"*

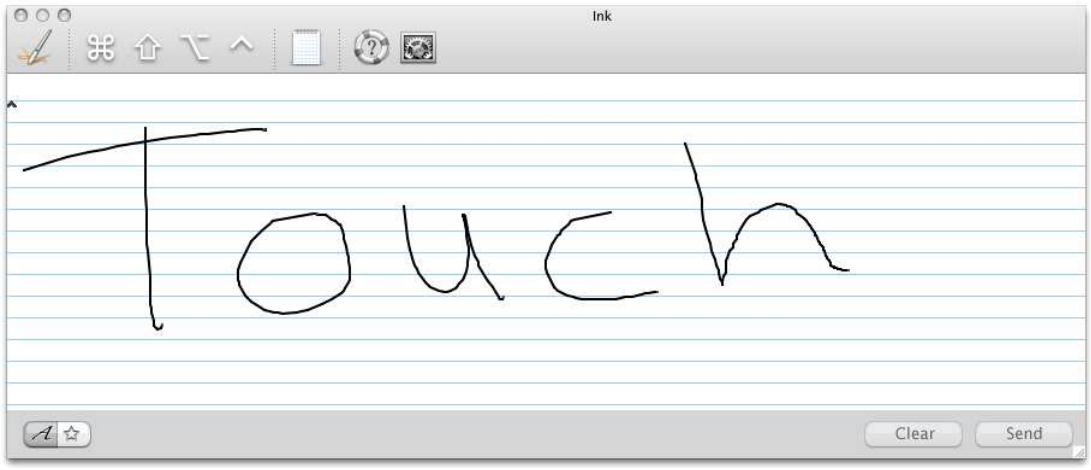
Launch the Ink settings panel to enable Hand recognition

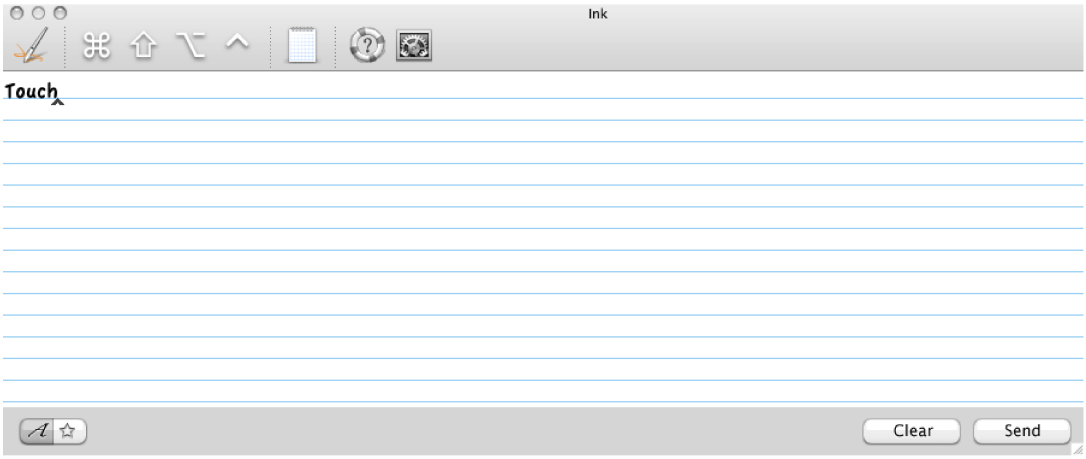


Once enabled the Ink floating windows will be displayed

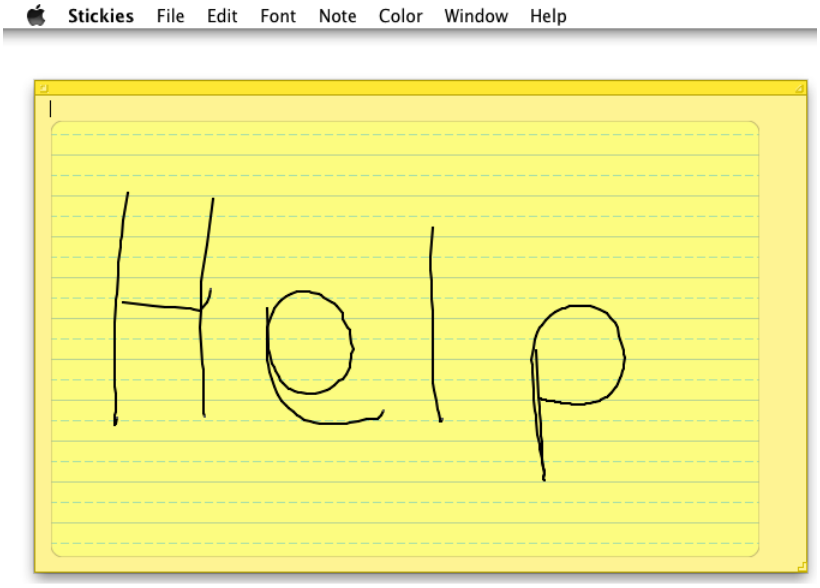
Activating handwriting recognition displays the Ink floating window. Whenever you want to enter text or drawing with your tablet, click the paper tablet icon in the Ink window. The A button in the lower left corner lets you enter text. The star button lets you draw. Clicking the Send button copies what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

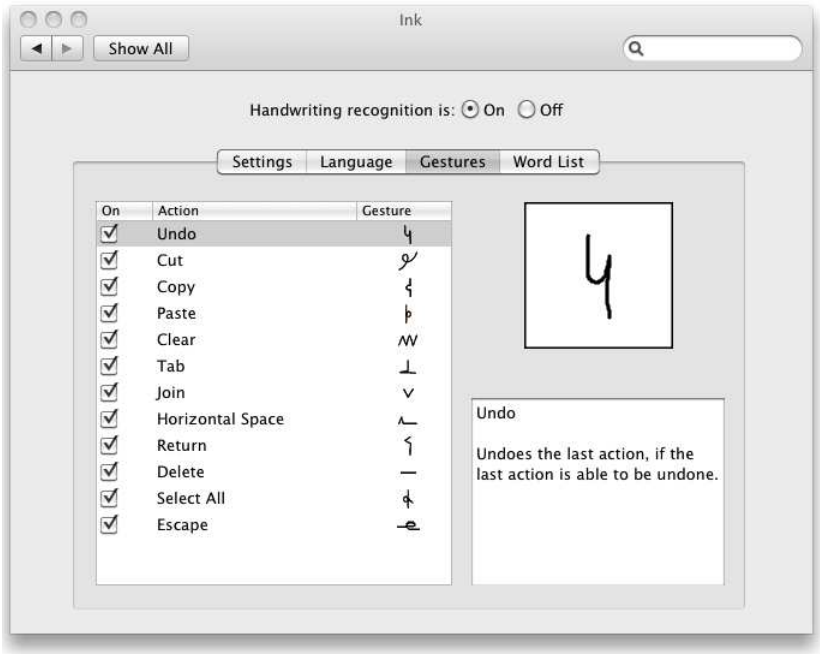




With Inking enabled, writing into any ink aware application will invoke an inking area in which to write, as in this example :



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.

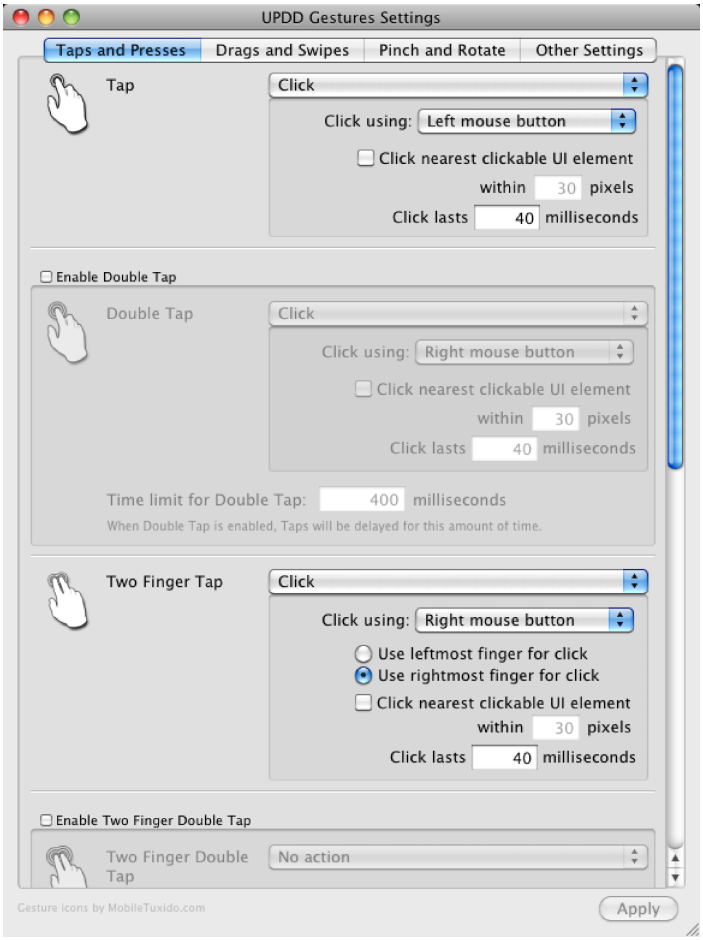
Further, given that there is no dedicated 'tablet stylus' in use the "hold lower button to ink" and "hold upper button to ink" settings have no meaning when inking with UPDD.

Gesture and inking settings

By default the gesture actions mimic those gestures associated with an Apple multi touch track for the host version of Mac OS X but can be defined as required. Gesture actions and other gesture settings are held either in the UPDD Settings file. Further, the latest version now utilises a graphics user interface for maintaining and updating the settings.

Gesture GUI and UPDD settings file

With version 2 of the gesture software, first release Dec 2012, the gesture settings are now stored in the UPDD settings file and a graphic interface is available for defining and maintaining the gesture settings. Settings in the UPDD settings file can also be updated with the UPDD command line interface

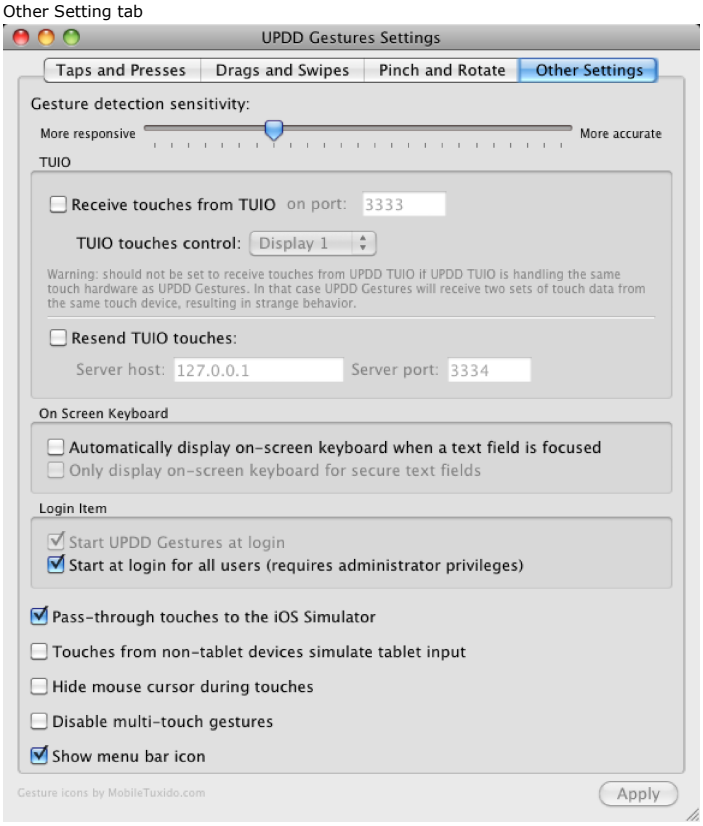


The gesture preference dialog is used to define and maintain the gesture settings and is invoked from the Gesture Menu item.or by running the gesture application whilst gestures are already loaded.

Gesture settings and actions

Indicates the action associated with each gesture appropriate to the actions that are available to the version of the OS. Some actions are restricted to that expected by the gesture whilst others can be defined to perform system and other actions.

Other Setting tab



Notes:

Gesture Detection sensitivity

Sets the sensitivity of the gesture recognition. Gesture sensitivity determines how much movement must occur in general before UPDD Gestures has decided on which gesture is being performed. In general, more movement yields more accurate detection but will decrease responsiveness. If gestures are being miscalculated try improving the accuracy. If recognition is too slow set for more responsiveness.

TUIO settings

Held under the 'Other Settings' tab these settings relate to the TUIO server (receive touch data from) and client (send touch data to) interface.

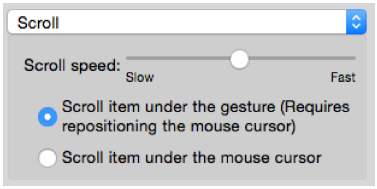
Specialsed settings

Hopefully these self-explanatory settings are use to

- Invoke the onscreen keyboard
- Start gestures at login
- pass touches to the iOS simulator
- Emulate Tablet input (inking)
- Disable mouse cursor during touches
- Disable multi-touch gestures
- Enable / disable Menu Bar icon

Notes:

1. ***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. Since gesture version 2.0.47 Zoom, rotate and scroll actions have an option for either performing the gesture on the item under the mouse cursor or under the position where the gesture occurs on the screen (the mouse cursor is automatically repositioned): e.g.

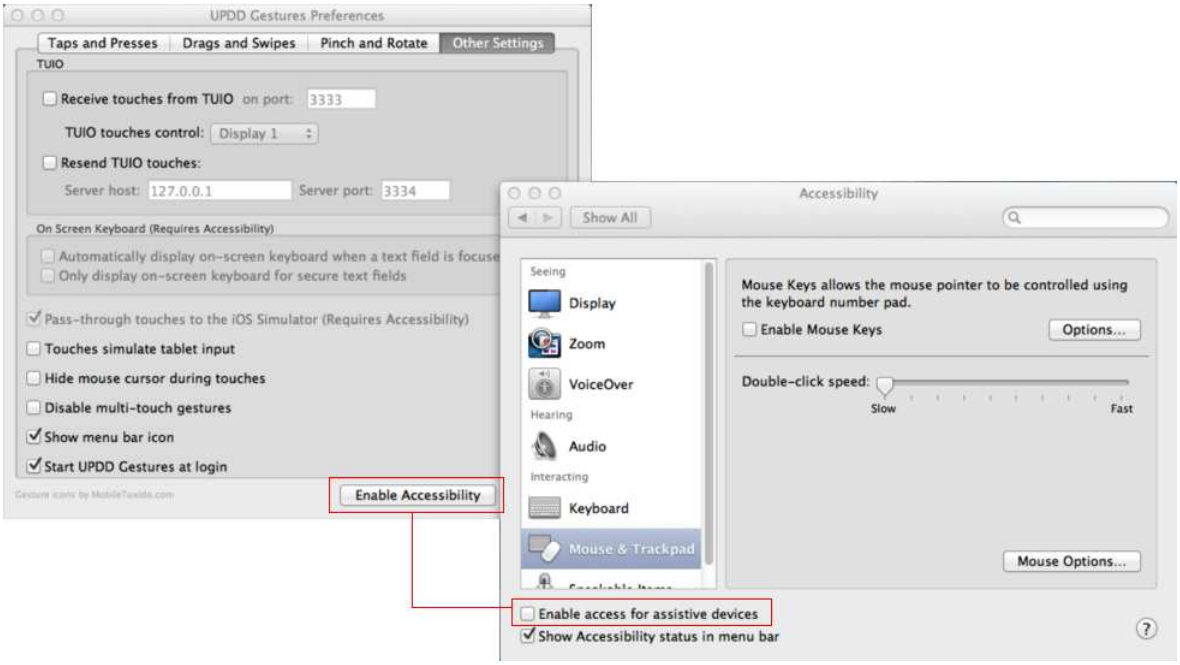


   We were asked if this setting could be made available for any gesture. Unfortunately the current design of the settings dialig does not readily cater for this so although the code is in place to facilitate this request the setting has to be set manually in the settings file.

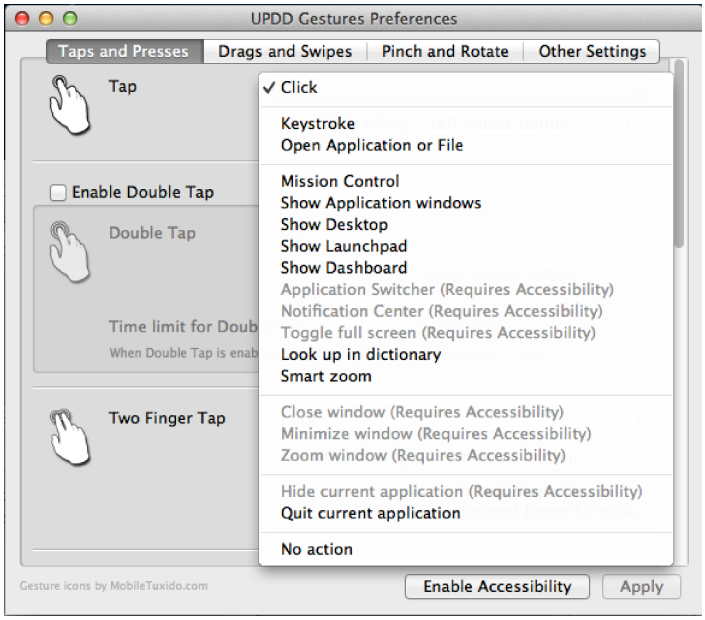
2. Several of the actions that can be invoked by a gesture, such as notification center, require that "Enable Access for Assistive Devices" is turned on in the "Universal Access" system preferences as described below.

   Access for assistive devices requirements

   A number of functions performed by the gesture software require that 'Access for Assistive Devices' be enabled in the system. If this is not enabled then the Gesture GUI will carry an Enable Accessibility option as shown below:



Whilst 'Access for assistive devices' is disabled the Gesture GUI will disable any functions that require it to be enabled:



3. Touches simulate tablet input

   When this setting is enabled:

   - Any mouse events created by Gestures through "Click" and "Click and drag" actions are also tablet events. This is because in OS X tablet events are actually special mouse events with extra tablet data included.
   - The "Ink" preference pane will be visible in System Preferences, if it was not already, allowing Inking to be configured. If Inking is enabled, it can be used with Gestures through the "Click" and "Click and drag" actions.

Driver setting considerations

Mouse interface

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. *Since release April 2013 the Gesture software now caters for this situation.*

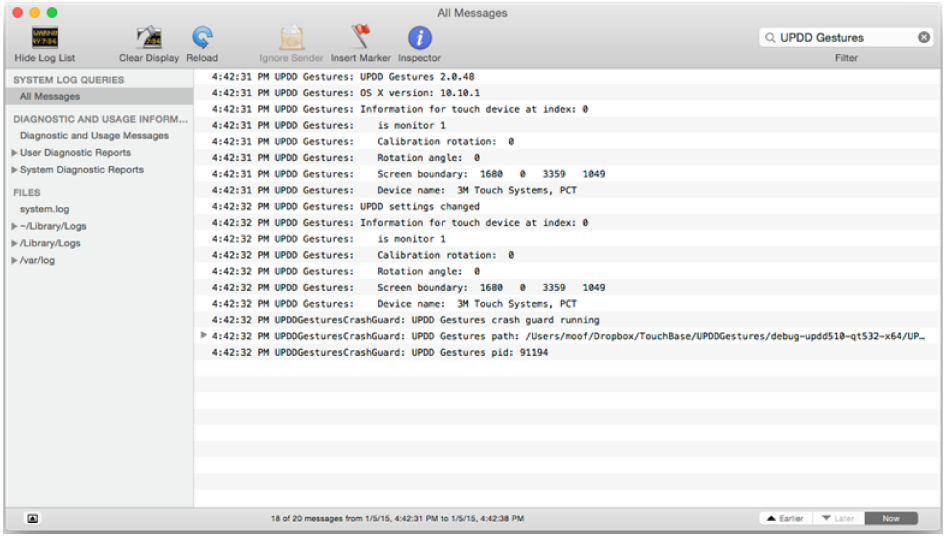
Lion Full Screen Mode

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.

Trouble shooting

Gestures does not load (no error issued) or loading incorrectly or does not function correctly

When the gesture software loads a gesture menu bar item will be shown (if enabled to be shown in the gesture settings). This is the indication that the program is running and hopefully working as expected. If the gesture software does not appear to be running or loading correctly then since version 2.0.48 you can load Console app from the Utilities folder and search for UPDD to view the startup log: (the log below shows a successful startup)

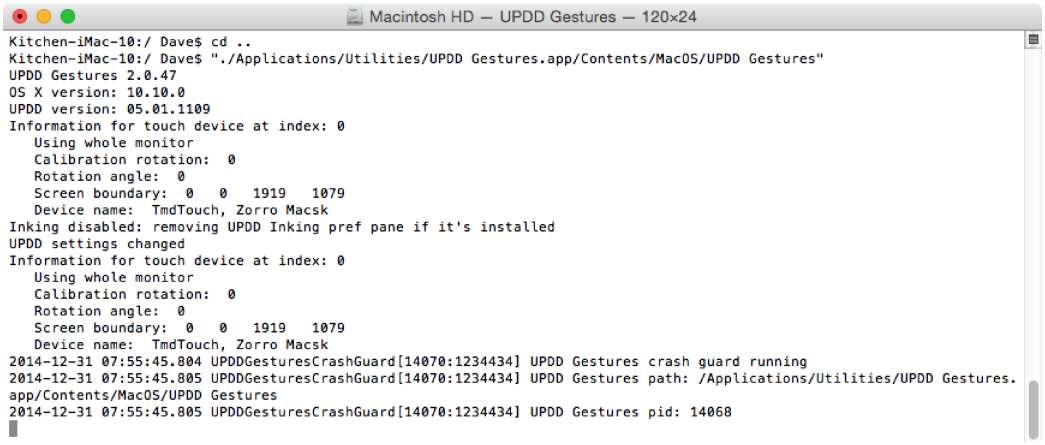


Alternatively you can run the gesture application from a terminal window and see if any error messages are issued at the time the program is invoked.

To run the gesture application execute the following command line:

/Applications/Utilities/UPDD\ Gestures.app/Contents/MacOS/UPDD\ Gestures

Here is example of gestures loading without error:



Please report any error message shown to technical@touch-base.com.

Incorrect gesture version

Some OEM manufacturers that ship our Mac driver and also make available the gesture software for download take time to update they download pages with the latest software which can cause issue. For example, running an old version of the gesture software (prior to version 2.0.13) in Mavericks will silently fail unless running it from a terminal Window in which case you will see the error message, typically "Enabling accessibility failed!"

Gestures stop working

If the gesture menu bar item is enabled but 'disappears' from the menu bar and/or gestures stop working it is possible that the gesture program has crashed. If this is the case there should be a crash log located in the following path:

~/Library/Logs/DiagnosticReports/UPDD Gestures_[timestamp]_[computerName].crash

Usually the Library folder is hidden, so you may need to do the following to open it:

1. In the Finder pick the "Go to folder..." menu item in the "Go" menu, or press Command K.
2. Type in the following:

~/Library/Logs/DiagnosticReports

Difficulty generating gesture

The performed gesture does not work as expected. In this instance please view the gesture log, as selected from the Menu Bar item, to see what gesture is being calculated by the gesture engine. If the correct gesture is shown and it is consistent with the gesture being performed then ensure your usage of the gesture is correct for the application or desktop function.

If the log shows that the gesture is inconsistent in its generation see if any setting in the driver is causing the issue, such as the lift off time needing to be increased so that short breaks in the touch are ignored (UPDD Console, Properties, Lift off time).

Safari bug

Due to a bug in Safari, two-finger-dragging to the right when there is no page to go forward to can cause unintended vertical scrolling.

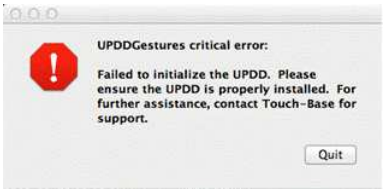
Critical error

The Gestures app receives touch data from the UPDD driver and if there is a failure with the UPDD interface a critical error dialog is issued.

This could in theory happen for a number of reasons but the most common reason is that the UPDD driver does not exist on the system and therefore the UPDD API is not enabled.

Since version 2.0.15, after this error is issued any startup

item defined for the UPDD Gesture app will be disabled. This can be re-enabled again in the Gesture setting dialog, Other Settings tab.



Linux

Creates a virtual touch device that is registered with the system as a multi-touch capable device thro' which all stylus touch data is passed.

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.