EXHIBIT G

‘853 Infringement Contentions

Motorola’s infringing products (“Accused Devices”) include mobile devices, such as smartphones, associated software, and components thereof. The Accused Devices include Motorola’s Android based phones which include, but are not limited to, the Motorola Droid X, Droid 2, Droid 2 Global, Cliq 2, Defy, Bravo, Droid Pro, Droid 2 R2-D2, Droid X 2, Charm, Droid, Flipside, Flipout, Atrix, Droid Bionic, Xoom, Devour A555, Backflip, Cliq/Dext, Cliq XT/Quench, Citrus, Spice, i1 and other Motorola Android based phones incorporating hardware and/or software that is substantially similar. The figures and illustrations in the infringement chart below display exemplary devices.

U.S. Patent No. 6,897,853 (‘853 Patent)	Accused Devices
7. A method for classifying a user's input to a computer comprising the steps of:	**Each Accused Device classifies a user's input to a computer**. Each Accused Device is a handheld computer. For example, Figure 7-1 shows the Motorola device branded as Droid 2 (hereinafter, "Droid 2").  Figure 7-1. *See also* "Droid 2 by Motorola,"

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/Motorola-DROID-2-US-EN (visited March. 18, 2011). Each Accused Device includes a graphical user interface that includes a touch screen display. (*See id.*) (describing the Droid 2's touch-sensitive interface)
receiving a user's input;	**Each Accused Device receives a user's input.** The touch screen display receives input in the form of physical contact from a user's finger. (Ex. *Id.*) (describing the Droid 2's touch-sensitive interface).
and first determining whether the input is a stroke based on a first move threshold;	**Each Accused Device determines whether the input is a stroke based on a first move threshold.** The Android operating system on Each Accused Device provides a *GestureDetector* class for identifying "gestures." (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.html.) *GestureDetector* detects fling (i.e., "stroke") gestures. When a user releases his or her finger from the touch screen, the function *onTouchEvent()* determines whether the user's finger swept across the screen based on a move threshold. Next, the nested class *GestureDetector.OnGestureListener* calls a function based on the gesture that the user performed. (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.OnGestureListener.html.) By way of example, this nested class includes *onFling(),* which corresponds to a fling gesture. (*See id.*) With reference to an example, Figure 7-2 shows that Droid 2 includes a home screen that displays widgets, including a widget showing the date.

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 7-2. When a user performs a fling gesture in the leftwards direction, the screen advances to expose another screen panel that can optionally contain additional program icons.

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 7-3. Thus, in this example, the Droid 2 determines whether a user performed a fling in order to determine whether to advance the screen to expose another screen panel.
if the input is not a stroke, then second determining whether the input is a tap based on a time threshold;	**<u>If the input is not a stroke, Each Accused Device determines whether the input is a tap based on a time threshold</u>**. *GestureDetector* detects tap gestures. In a "tap" gesture, the user's finger is held against the touch screen for less than a threshold amount of time. The threshold is identified as a constant in the *GestureDetector* class. *(See android/frameworks/base/core/android/view/GestureDetector.java*). The nested class *GestureDetector.OnGestureListener* calls a function based on the gesture that the user performed. (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.OnGestureListener.html.) By way of example, this nested class includes *onSingleTapUp()*, which corresponds to a "tap" gesture. (*See id.*) With reference to the example described above, when a user performs a tap gesture on the date widget, the widget opens and

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	a calendar view for that day appears.  Figure 7-4. Thus, in this example Droid 2 determines whether the input was a tap in determining whether to open the widget.
if the input is neither a stroke nor a tap, then third determining whether the stroke is a hold or a hold and drag.	**If the input is neither a stroke nor a tap, Each Accused Device determines whether the stroke is a hold or a hold-and-drag**. *GestureDetector* determines whether the user performed a long press (i.e., "hold") based on a time threshold. In a "tap" gesture, the user's finger is held against the touch screen for more than a threshold amount of time. The threshold is identified as a constant in the *GestureDetector* class. *(See android/frameworks/base/core/android/view/GestureDetector.java*). Next, the nested class *GestureDetector.OnGestureListener* calls a function based on the gesture that the user performed. (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.OnGestureListener.html.) By way of example, this nested class includes *onLongPress()*, which corresponds to a "hold" gesture. (*See id.*)

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	By way of example, when a user performs a long press on the date widget and then releases his or her finger, the screen switches to a mode that allows the user to resize the widget, as shown in Figure 7-5.  Figure 7-5. Each Accused Device also determines whether the user, after pressing on the screen for more than a threshold amount of time, moves his or her finger to perform a "hold and drag." If the gesture is a hold and drag, the phones determine that the user wants to move the widget. It makes this determination based on whether the user moves the widget more than a threshold distance from its original location. By way of example, in Figure 7-6, the user has pressed the icon but has not moved the widget more than a threshold distance from its original location and the widget will not be moved to a new location.

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 7-6. In Figure 7-7, the user has moved the widget more than a threshold distance from its original location.  Figure 7-7. Thus, this is classified as a hold and drag and the icon is placed in a new location as shown in Figure 7-8.

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 7-8. Similarly this functionality can also be seen in relation to icons placed on the home screen. When a user performs a long press on an icon for a program application, the screen switches to an alternate mode that allows the user to move the icon to a new location. This is depicted in Figure 7-9 below, in which a user performed a long press on the "Browser" icon. In this mode, the selected icon becomes enlarged.

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 7-9 Each Accused Device then determines whether the gesture that the user has performed is a "hold" or a "hold and drag." If the gesture is a hold and drag, Each Accused Device determines that the user wants to move the icon. It makes this determination based on whether the user moves the icon more than a threshold distance from its original location. In Figure 7-10, the user has not moved the icon more than a threshold distance from its original location.

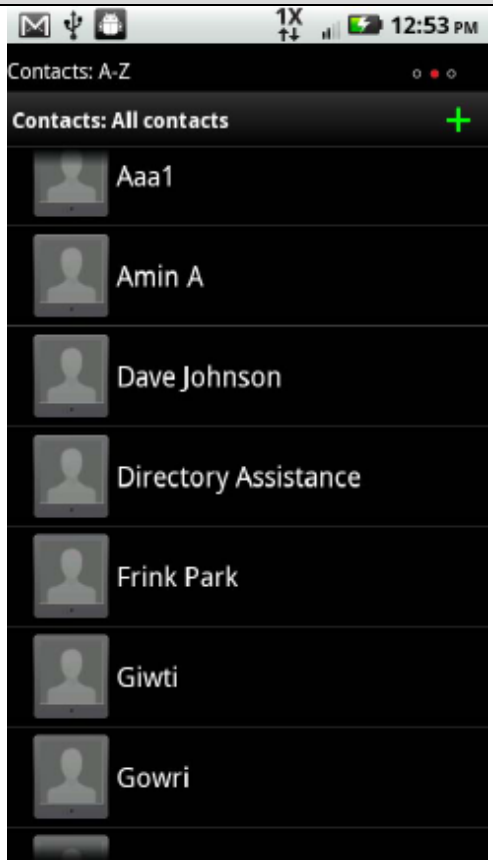
U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 7-10. Thus, this motion is not classified as a hold and drag, and the icon returns to its original location as shown in Figure 7-11.  Figure 7-11.

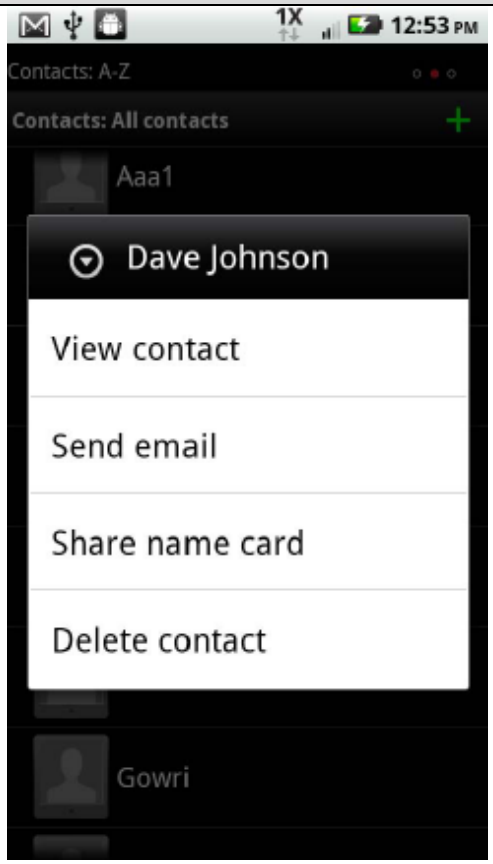
U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	In Figure 7-12, the user does move the icon more than the threshold distance.  Figure 7-12. Thus, this motion is classified as a hold and drag and the icon is placed in its new location as shown in Figure 7-13.

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 7-13. Thus, Each Accused Device determines whether the user has performed a hold on the icon, or a hold and drag on the icon.
8. The method of classifying a user's input according to claim 7, wherein, if said input satisfies said first move threshold, the input is classified as a stroke.	**Each Accused Device classifies a user's input as a stroke if the input satisfies a first move threshold**. As explained above, the Android Operating System on each Accused Device provides a *GestureDetector* class for identifying "gestures." (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.html.) *GestureDetector* detects fling (i.e., "stroke") gestures. When a user releases his or her finger from the touch screen, the function *onTouchEvent()* determines whether the user's finger swept across the screen based on a move threshold. If this condition is met the Droid classifies the move as a "fling" or stroke.
9. The method of classifying a user's input according to claim 7, wherein, if said input does not satisfy said first move threshold and said input does not satisfy said	**Each Accused Device classifies a user's input as a tap if the input does not satisfy a first move threshold and time threshold.** As explained above, *GestureDetector* detects tap gestures. In a

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
time threshold, the input is classified as a tap.	"tap" gesture, the user's finger is held against the touch screen for less than a threshold amount of time. The threshold is identified as a constant in the *GestureDetector* class. *(See android/frameworks/base/core/android/view/GestureDetector.java*). The nested class *GestureDetector.OnGestureListener* calls a function based on the gesture that the user performed. (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.OnGestureListener.html.) By way of example, this nested class includes *onSingleTapUp(),* which corresponds to a "tap" gesture. (*See id.*) In this way the accused phones classify a user's input as a "tap."
10. The method of classifying a user's input according to claim 7, wherein, if said input does not satisfy said first time threshold and said input does not satisfy said second move threshold, said input is classified as a hold.	**Each Accused Device classifies a user's input as a hold if the input does not satisfy a first time threshold and second move threshold.** As explained above, *GestureDetector* determines whether the user performed a long press (i.e., "hold") based on a time threshold. In a "hold" gesture, the user's finger is held against the touch screen for more than a threshold amount of time. The threshold is identified as a constant in the *GestureDetector* class. *(See android/frameworks/base/core/android/view/GestureDetector.java*). Next, the nested class *GestureDetector.OnGestureListener* calls a function based on the gesture that the user performed. (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.OnGestureListener.html.) By way of example, this nested class includes *onLongPress(),* which corresponds to a "hold" gesture. (*See id.*) In this way the accused phones classify the gesture as a LongPress or "hold."
11. The method of classifying a user's input according to claim 10, further comprising the step of: simulating a right mouse click responsive to said input being classified as a hold.	**Each Accused Device classifies a user's input as a hold if the input does not satisfy a first time threshold and second move threshold.** As explained above, *GestureDetector* determines whether the user performed a long press (i.e., "hold") based on a time threshold. In a "hold" gesture, the user's finger is held against the touch screen for more than a threshold amount of time. The

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	threshold is identified as a constant in the *GestureDetector* class. *(See android/frameworks/base/core/android/view/GestureDetector.java*). Next, the nested class *GestureDetector.OnGestureListener* calls a function based on the gesture that the user performed. (*See* http://developer.android.com/intl/zh-TW/reference/android/view/GestureDetector.OnGestureListener.html.) By way of example, this nested class includes *onLongPress()*, which corresponds to a "hold" gesture. (*See id.*) When a LongPress is detected on a list item the Android system simulates a right mouse click. For instance, "[t]he Android system provides two types of menus you can use to provide functionality or navigation." (Menu Design Guidelines.) One of the menus that the accused devices generate is a context menu: "This is a floating list of menu items that may appear when you perform a long-press on a View (such as a list item)." Ex. B , "Creating Menus Documentation," http://developer.android.com/guide/topics/ui/menus.html (visited March. 18, 2011). An exemplary list of items is shown in Figure 11-1 below:

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 Figure 11-1. Figure 11-2 shows an exemplary context menu that is generated when the user performs a LongPress or "hold" on one of the list items in figure 11-1.

U.S. Patent No. 6,897,853 ('853 Patent)	Accused Devices
	 **Figure 11-2** "The Android context menu is similar, in concept, to the menu revealed with a 'right-click' on a PC. When a view is registered to a context menu, performing a 'long-press' (press and hold for about two seconds) on the object will reveal a floating menu that provides functions relating to that item." *Id.* Thus, each Accused Device simulates a right mouse click responsive to a user input being classified as a hold.