

Multi-Touch, Consumers and Developers

Charles Roberts

Media Arts & Technology Program
University of California, Santa Barbara
email: charlie@charlie-roberts.com

Abstract

Multi-Touch capable devices have just begun to hit their stride in the consumer marketplace. It is a relatively new haptic paradigm that provides for interface widgets that were not previously realizable. Current multi-touch devices vary greatly in their target audience, with some focusing on consumers and others focusing on expert users and developers. Development environments for multi-touch environments continue to be refined along with a standardized vocabulary of gestures.

Keyword: multi-touch, interface, gesture, development

Introduction

Touch sensitive control devices have been developed and explored for decades; as one example, *frustrated total internal reflection* (FTIR), one of the techniques used in current multi-touch research, traces its roots back to experiments from the 1960s [1]. It is only recently, however, that multi-touch technologies have become widely available to consumers and, possibly more impor-

tantly, developers have been able to embrace the technology as a new paradigm in interactivity. These technologies include consumer devices from the 1990s such as the *iGesturePad* by Fingerworks [2], to more recent consumer devices such as the Apple iPhone [3], to large scale devices found in the work of Jeffrey Han [4] and in the new Surface platform by Microsoft [5].

The advent of these devices poses questions about the benefits of multi-touch, the challenges of developing new models of interactivity which employ multi-touch and the challenges that developers face in order to include these new capabilities in their applications and prototypes.

This paper examines previous attempts to answer questions about these issues, and discusses the authors recent experience developing a touch based application for the Apple iPhone.

Why touch? Why multi-touch?

The immediate question facing developers creating applications or devices utilizing multi-touch is perhaps obvious but crucial: what advantages does employing

multi-touch gestural capabilities bring to my product?

For many applications touch interfaces are more intuitive than those driven by a mouse. Painting shapes is a simple example; most of us have finger painted at one time or another and are well accustomed to the idea of drawing lines with our fingers. Dragging items from one location to another is slightly less intuitive as this is not a action that we commonly perform using a single finger; but it does happen (think of sorting coins as an example). But what about scrolling through documents via finger swipes? What gestures should be employed for copy and paste? Should such gestures be universal across all applications and devices?

Two multi-touch by the company Jazz-Mutant [6] offer very different approaches to solving this problem. In their product *Dexter*, the interface is a 12x12" multi-touch sensitive screen that is used with digital audio workstations (DAW) such as Cubase, Logic and ProTools (Fig. 1). The interface is designed to be similar in many ways to the those of the applications it works with. This is ironic as much of the interface of many DAWs is designed to mimic hardware. We are thus left with hardware providing an interface to software that is designed to emulate hardware.

Fortunately, this is only the surface of *Dexter* (no pun intended). Since the graphics displayed on *Dexter's* touch

screen are all vector, they are also infinitely scaleable. By using pinching and expansion multi-touch gestures, users are able to zoom in on particular interface widgets. This is useful, for example, if you want to control a fader or knob with greater precision than would be allowed by the widgets normal size. Thus, even though in some ways it is designed to emulate a traditional hardware interface, it still provides features (there are others besides the aforementioned zooming) that would not be possible using hardware that was not augmented digitally.

In contrast to *Dexter*, the *Lemur* [7] by JazzMutant is a highly configurable multi-touch device. Instead of automatically configuring its interface to work with most major DAWs, the *Lemur* allows the user to create their own custom custom configurations using provided editor software. The *Lemur* ships with a library of standard interface widgets that users can arrange in any fashion they choose. It also contains its own physics engine, so that interface items (such as the multiball) can be controlled by gravity and laws of attraction in addition to gestures by the user. This allows users to "fling" items towards a general goal value rather than precisely placing them, and opens up novel methods of partially stochastic control when objects bounce off of container walls and gravitate towards each other.

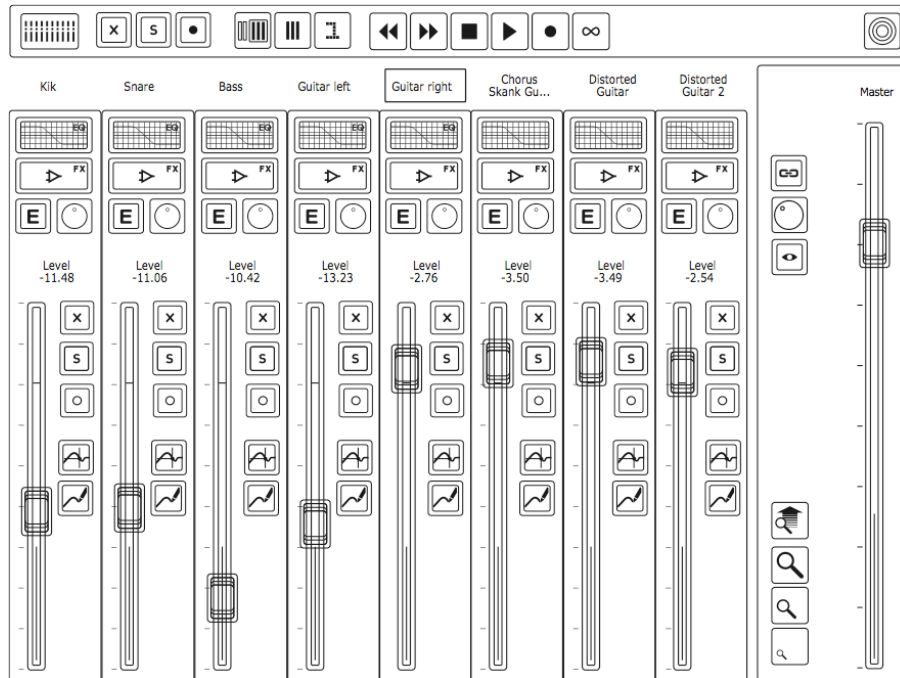


Fig 1. A vector rendering of the Jazz-Mutant Dexter interface. Note the similarity to traditional multi-channel hardware mixers.

The configurable nature of the Lemur is probably less objectionable to the electronic musician / visual artist than it would be to the average consumer, as people working in digital media are often used to customizing their interfaces to meet their particular needs. From the authors perspective, this customization is a critical part of creating any live performance setup.

There is a history of electronic music pushing advances in interface technology. From early analog synths featuring ribbon controllers to more recent devices such as the Continuum [8] and the Lemur, musicians and engineers have been exploring novel ways to allow performers to express themselves via touch. When we leave the niche of live performance, however, there is a very different set of concerns regarding multi-touch.

Mass Market Multi-Touch Concerns

The introduction of multi-touch to devices such as the iPod Touch and the iPhone has created the need for a standardized set of multi-touch gestures that can be used across applications, and, ideally, across devices. Apple took a giant step ahead in this arena when it acquired the company Fingerworks [2] in 2005. Fingerworks had already developed a number of products using multi-touch gestures, and had created a comprehensive gesture vocabulary to go with them (Fig. 2).

It could be argued that the vocabulary suggested by Fingerworks is overly ambitious, and the author posits that few users would be willing to learn all the gestures outlined in the Fingerworks documentation. Many of these gestures would be primarily used by power users.

But the many of the gestures that would be used more commonly, such as the one to scroll vertically or horizontally, have been adopted and simplified by Apple. Since Apple products do not currently employ the comprehensive set of gestures that the Fingerworks products possessed, the gestures that Apple products use can be simplified; the simplified gestures become more easily recognizable and stratified as there are fewer overall. Thus, scrolling moves from “Touch & slide four fingers up/down” as taken from the Fingerworks website [9] to the current two finger slide used in Apple products. There are currently ten different gestures used in the multi-touch trackpads shipped with the Apple MacBook Pro and the MacBook Air [10].

It would be excellent if the gestures used in Apple products were made into a standard that could be used across applications and devices. Unfortunately, Apple has filed for patents regarding their multi-touch gestures [11]. It is not certain at this time whether these patents will be granted, or, if they are granted, whether or not Apple will pursue violations. Apple might be attempting to acquire a so-called “defensive” patent; by obtaining the relevant patents themselves they ensure that other companies cannot obtain them and then sue Apple. It is highly likely that if Apple does patent multi-touch gestures that competing sets of gestures will come to the market; Synaptics, a maker of trackpads, is already working on their own set of gestures [11].

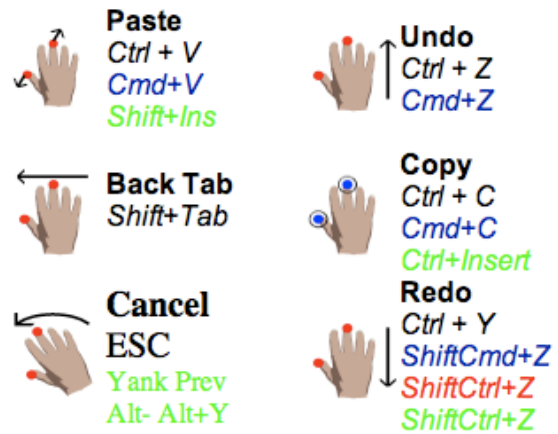


Fig 2. A small sample of the gestures used in the original Fingerworks products, as taken from their reference manual.

It is in the best interest of consumers and developers for a universal standard to be created.

The Next Step for Touch Devices?

In the author’s opinion, there is a great deal more appeal to interfaces based on tangible mechanical devices than on virtual widgets. By physically manipulating real, tangible, physical entities, many people feel that they are more connected to the interfaces they use. Current multi-touch screens share a lack of haptic feedback with traditional mouse based interfaces; the author would argue that overcoming this lack of haptic feedback would be an important next step in increasing the usability of touch screen interfaces.

There is an experiment using the vibration on the iPhone to attempt to provide such haptic feedback [12]. Everytime users click keys on the iPhone virtual key-

board, vibration is triggered in order to provide haptic feedback. In a poll on the internet [13], 28.4% of people responding to the question “Would you download a program to give your iPhone a haptic keyboard?” responded yes, while 26.7% responded no. Clearly, vibration alone is not enough to create the type of haptic feedback desired by users.

Nokia has gone one giant step further in creating haptic feedback with their announcement of the S60 internet tablet [14]. The S60 features a screen that’s actually able to move .1 mm in height to create the impression of keys moving up and down. In the words of one blogger: “In use, the touch feedback on the demo device was near on perfect. Each press of a key returned a clunky click and tactile snap on the touchscreen, which made typing feel incredibly responsive and very usable on the smooth screen surface. In fact it was hard to remember that you were using a touchscreen keyboard.” [14]

Development for Multi-Touch Devices

This article has already briefly discussed the process for “developing” interfaces with the Lemur; developers work entirely inside of a graphical development environment called the JazzEditor to create interfaces to their OSC enabled applications. A SDK for Microsoft Surface has yet to be announced; in July of 2007 the head of Microsoft’s Partner program stated that such a SDK could be avail-

able by April of 2008, potentially putting it just around the corner [15].

With the recent introduction of the iPhone SDK, we have the first concrete example of how developers might be expected to handle interactivity within a multi-touch enabled device. Much of the iPhone SDK is basically a pared down version of the standard Cocoa framework used in OS X applications. This pared down Cocoa framework is named UIKit. Any UIView object part of this framework that is created on screen can take advantage of multi-touch gestures; all iPhone interface widgets are at some level subclasses of the UIView class.

The simplest example of detecting a touch in a UIView follows on the next page.

```

- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event {
    UITouch *touch = touches.anyObject;
    int xCoordinate = touch.locationInView.x;
}

```

In above example, written in Objective-C, is triggered when a finger first touches the screen of the iPhone. One touch is extracted from a potential multi-touch set, and then the x coordinate of this touch is obtained from the `locationInView` struct.

It is not much more difficult to obtain the locations of multiple simultaneous touches on the screen:

```

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event {
    for (UITouch *touch in touches){
        if ((touch.info & UITouchSwipedRight)) {
            NSLog(@"swiped up");
        }
    }
}

```

In the above example, we enumerate all the different touch objects in the set *touches*. By examining the `info` property of each touch object, we can theoretically determine what type of gesture is occurring in addition to looking at the x and y coordinates of individual touches. Unfortunately, the author was unable to get gesture detection code to work, both in code that he wrote himself and in the sample code that Apple distributed through their developer website. It is assumed that this is a bug in the beta version of the iPhone SDK that will be fixed upon its final release.

No official documentation has been released for how Cocoa developers can take advantage of the new gestures introduced with the Macbook Air and also available in newer models of the Macbook Pro. Apple developers have already included use of the gestures in standard OS X apps like Safari and iPhoto. One developer did some hacking and figured out that the gestures will basically wind up being the following messages sent to the `NSResponder` class [16].

```

- (void)magnifyWithEvent:(id)fp8;
- (void)rotateWithEvent:(id)fp8;
- (void)swipeWithEvent:(id)fp8;
- (void)beginGestureWithEvent:(id)fp8;
- (void)endGestureWithEvent:(id)fp8;

```

There is also a new private method for `UIApplication _handleGestureEvent`: that receives every gesture passing through the application. Use of the above messages should make it relatively easy to enable these gestures in applications, and the blog entry referenced above provides a sample project that makes use of the new gestures and events.

Conclusion

The use of multi-touch gestural interfaces is on the rise. Their introduction and use in the iPhone and iPod touch has put millions of such devices onto the market and created substantial interest, as indicated by over 100,000 downloads of the iPhone SDK within four days of its initial release [17]. Continued advances in haptic technology such as the feedback provided by the Nokia N60 internet tablet and standardization of gestures used for interactivity will make multi-touch screens and interfaces increasingly common in everyday technologies. The development environment provided by Apple has already shown that gesture sensing can be abstracted to an extremely high level and easily integrated into applications; hopefully other device manufacturers will adopt a similar tact to make multi-touch interfaces a win for both consumers and developers.

References

1. Han, J. Y. 2005. Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. In *Proceedings of the 18th Annual ACM Symposium on User interface Software and Technology* (Seattle, WA, USA, October 23 - 26, 2005). UIST '05. ACM Press, New York, NY, 115-118.
2. Fingerworks. iGesturePad. <http://www.fingerworks.com> . Accessed 3.17.2008
3. Apple. 2008. iPhone. <http://www.apple.com/iphone/> . Accessed 3.17.2008
4. Han, Jeffrey. Multi-Touch Interaction Research. 2006. <http://cs.nyu.edu/~jhan/ftirtouch/> . Accessed 3.17.2008
5. Microsoft. Surface. 2007. <http://www.microsoft.com/surface/> . Accessed 3.17.2008
6. JazzMutant. Dexter. 2008 http://www.jazzmutant.com/dexter_introduction.php Accessed 3.17.2008
7. JazzMutant. Lemur. 2008. http://www.jazzmutant.com/lemur_overview.php Accessed 3.17.2008
8. Haken, L. 2005. Continuum Fingerboard. <http://www.cerlsoundgroup.org/Continuum/> . Accessed 3.17.2208
9. Fingerworks. Gesture/Hotkey Mappings. http://www.fingerworks.com/gesture_keymap.html . Accessed 3.19.2008

10. Apple. MacBook Air. 2008.
<http://www.apple.com/macbookair/features.html> . Accessed 3.19.2008
11. Gardinier, Brian. "Can Apple Patent the Pinch? Experts Say It's Possible". Wired.com
http://www.wired.com/gadgets/miscellaneous/news/2008/02/multitouch_patents . Accessed 3.20.2008
12. iPhone-Haptics.
<http://code.google.com/p/iphone-haptics/> . Accessed 3.20.2008
13. Gawker.com. "Would you download a program to give your iPhone a haptic keyboard?"
<http://polls.gawker.com/?key==MjN0QTM&voted=1> . Accessed 3.20.2008
14. Powell, Nigel. "Nokia perfects the clicky tactile touchscreen - iPhone gnashes teeth, swears revenge"
<http://www.redferret.net/?p=9533> Accessed 3.20.2008
15. PCRetailMag. "Microsoft to Showcase Surface to Partners"
<http://www.pcretailmag.com/news/28090/Microsoft-to-showcase-Surface-to-partners> . Accessed 3.20.2008
16. Harris, Elliot. "NSResponder Modifications: Swipe, Rotate and Magnify"
<http://cocoadex.com/2008/02/nsevent-modifications-swipe-ro.html> . Accessed 3.20.2008
17. Apple. "iPhone SDK Downloads Top 100,000"
http://www.apple.com/pr/library/2008/03/12ip_hone.html Accessed 3.20.2008