Gestural Interfaces: A Step Backward In Usability

Donald A. Norman
Nielsen Norman Group and Northwestern University | norman@nngroup.com

Jakob Nielsen
Nielsen Norman Group | nielsen@nngroup.com

One step forward, two steps back.

Once again, the usability crisis is upon us. We suspect most of you thought it was over. After all, HCI certainly understands how to make things usable, so the emphasis has shifted to more engaging topics, such as new applications, new technological developments, and the challenges of social networks and ubiquitous connection and communication. Well, you were wrong.

In a recent column for *interactions*, Norman pointed out that in the rush to develop gestural (or "natural") interfaces, well-tested and understood standards of interaction design were being overthrown, ignored, and violated [1]. Yes, new technologies require new methods, but the refusal to follow well-established principles leads to usability disaster.

Recently, Raluca Budui and Hoa Loranger from the Nielsen Norman Group performed usability tests on Apple's iPad, reaching much the same conclusion. The new applications for gestural control in smart cell phones (notably the iPhone and Android devices) and the coming arrival of larger screen devices built upon gestural operating systems (starting with Apple's iPad) promise even more opportunities for well-intended developers to screw things up. Nielsen put it this way: "The first crop of iPad apps revived memories of Web designs from 1993, when Mosaic first introduced the image map that made it possible for any part of any picture to become a UI element. As a result, graphic designers went wild: Anything they could draw could be a UI, whether it made sense or not. It's the same with iPad apps: Anything you can show and touch can be a UI on this device. There are no standards and no expectations." [2]

Why are we having trouble? Several reasons:

- The lack of established guidelines for gestural control
- The misguided insistence by companies (e.g., Apple and Google) to ignore established conventions and establish ill-conceived new ones.
- The developer community's apparent ignorance of the long history and many findings of HCI research, which results in their feeling empowered to unleash untested and unproven creative efforts upon the unwitting public.

In response to Nielsen's article about the iPad usability studies, some critics claimed it is reasonable to experiment with radically new interaction techniques when given a new platform. We agree. But the place for such experimentation is in the lab. After all, most new ideas fail, and the more radically they depart from previous best practices, the more likely they are to fail. Sometimes a radical idea turns out to be a brilliant radical breakthrough. Those designs should indeed ship, but it's important to realize radical breakthroughs are extremely rare in any discipline. Most progress is made through small and sustained incremental steps. Bold explorations should remain inside the company and university research laboratories and not be inflicted on any customers until those recruited to participate in user research have validated the approach.

There are several fundamental principles of interaction design that are completely independent of technology:

- Visibility (also called perceived affordances or signifiers)
- Feedback
- Consistency (also known as standards)
- Non-destructive operations (hence the importance of undo)
- Discoverability: All operations can be discovered by systematic exploration of menus.
- Scalability: The operation should work on all screen sizes, small and large.

[1] Norman, D. A. "Natural User Interfaces Are Not Natural." *interactions* 17, 3, (2010); http://interactions.acm.org/content/?p=1355/

[2] Nielsen, J. "iPad Usability: First Findings from User Testing." Jakob Nielsen's Alertbox, 26 April 2010. http://www.useit.com/alertbox/ipad.html/

• Reliability: Operations should work. Period. And events should not happen randomly.

All of these principles are rapidly disappearing from designers' tool kits, aided, we must emphasize, by the weird design guidelines issued by Apple, Google, and Microsoft.

What are we talking about? Let us explain.

Visibility

Nonexistent signifiers. To delete an unread message in Mail on the iPhone, swipe right across the unopened mail and a dialog appears, allowing you to delete the item. Open the email and the same operation has no result. In the Calendar, the operation does not work. How is anyone to know, first, that this magical gesture exists, and second, in which settings it operates?

With the Android, pressing and holding on an unopened email brings up a menu that allows, among other items, deletion. Open the email and the same operation has no result. In the Google calendar, the same operation has no result. How is anyone to know, first, that this magical gesture exists, and second, in which settings it operates?

Whenever we discuss these examples with others, we invariably get two reactions. One is "Gee, I didn't know that." The other is, "Did you know that if you do this [followed by some exotic swipe, multifingered tap, or prolonged touch] that [the following] happens?" Usually it is then our turn to look surprised and say, "No, we didn't know that." This is no way to have people learn how to use a system.

Misleading signifiers. For Android phones, there are four permanent controls at the bottom of the screen: back, menu, home, and search. They are always visible, suggesting they are always operative. True for three out of the four—not for the menu button. This visible menu button implies there is a menu available, but no, many applications (and places within applications) don't have menus, and even those that do don't always have them everywhere. There is no way to tell without pushing the button and discovering that nothing happens. (Actually, it means multiple pushes because the lack of a response the first time may reflect the unreliability of the technology.)

Worse, when on the home screen, pushing the menu will occasionally bring up the on-screen keyboard. Usually a second push of the same key undoes the action done by the first, but sometimes, the second push brings up a menu that floats above the material being displayed (The keyboard does not always appear. Despite much experimentation, we are unable to come up with the rules that govern when this will or will not occur.)

Feedback

Both Apple and Google recommend multiple ways to return to a previous screen. Unfortunately, for any given implementation, the method seems to depend upon the designer's whim. Sometimes one can swipe the screen to the right or downward. Generally, one uses the back button. On the iPhone, if you are lucky, there is a labeled button. (If not, try swiping in all directions and pushing everything visible on the screen.) With the Android, the permanently visible back button provides one method, but sometimes the task is accomplished by sliding the screen to the right. The back button has a major flaw, however. Push the back button to go to the previous page, then again, and then again: Oops, suddenly you are out of the application, never having been warned that the next button-push exits instead of simply going back. (The same flaw exists on the Blackberry.) The back button moves the user through the "activity stack," which always includes the originating activity: home.

This programming decision should not be allowed to affect the user experience: Falling off the cliff of the application to the home screen is not good usability practice. (Note too that the stack on the Android does not include all the elements the user model would include: It explicitly leaves out views, windows, menus, and dialogs.) Yes, provide a back button—or perhaps call it a dismiss button—but make it follow the user's model of "going back," not the programmer's model that is incorporated into the activity stack of the operating system. Among other things, it should have a hard stop at the top level of the application. The forced exit from the application is wrong.

Consistency and Standards

Whatever happened to the distinction between radio buttons and checkboxes? Radio buttons meant selection of only one out of all the possibilities: Selecting one precluded the selection of others. Check boxes, however, allow one to select multiple alternatives. Now, with these new systems, check boxes can work any

When users think they did one thing but actually did something else, they lose their sense of controlling the system because they don't understand the connection between actions and results.

way the developer chooses, often to the distress of the poor person trying to use the system.

Some applications allow pinching to change image scale; others use plus and minus boxes. Some allow you to flip screens up, some down, some to the right, some to the left, and some not at all. Touching an image can enlarge it, hyperlink from it, flip it over, unlock it so it can be moved, or whatever the developer and his whim decided.

The different operating-system developers have provided detailed interface guidelines for their products. Unfortunately, the guidelines differ from one another, in part because different companies wish to protect their intellectual property by not allowing other companies to follow their methods. But whatever the reason, proprietary standards make life more difficult for everyone. For sure, they undermine the main way in which users learn from each other.

Discoverability

The true advantage of the Graphical User Interface (GUI) was that commands no longer had to be memorized. Instead, every possible action in the interface could be discovered through systematic exploration of the menus. Discoverability is another important principle that has now disappeared. Apple specifically recommends against the use of menus. The Android UI team takes the opposite position, even providing a dedicated menu key, but does not require that it always be active. Moreover, swipes and gestures cannot readily be incorporated in menus: So far, nobody has figured out how to inform the person using the app what the alternatives are.

Scalability

Home computers, whether laptop or desktop, always came with a wide variety of screen sizes. Now that computer operating systems are starting to support multitouch technology, this means gestures must work on large screens as well as small. There is a plethora of screen sizes for cell phones, including the emergence of an in-between form; we now have midsize screens. Eventually, screens will range from tiny to huge, conceivably wall-size (or at least, whiteboard-size). However, gestures that work well for small screens fail for large ones, and vice versa. Small little checkboxes and other targets that work well with mouse and stylus are inappropriate for fingers. Larger screens have their own problems with control sizes. Are the new controls to be used while held in the hand, laid flat upon a surface, or tilted at an angle? All varieties now exist.

Sensitive screens give many opportunities for accidental selection and the triggering of actions. This happens on small screens because the target items might be small and close together. This happens on large screens because the same hands necessary to hold and stabilize the device can accidentally touch the screen.

Reliability

Accidental activation is common in gestural interfaces, as users happen to touch something they didn't mean to touch. Conversely, users frequently intend to touch a control or issue a gestural command, but nothing happens because their touch or gesture was a little bit off. Traditional GUIs do have similar problems: for example, when the mouse is clicked one pixel outside the icon a user intended to activate. But at least the mouse pointer is visible on the screen so that the user can see it's slightly off.

Since gestures are invisible, users often don't know that they made such mistakes. Also, a basic foundation of usability is that errors are not the user's fault; they are the system's (or designer's) fault for making it too easy to commit the error. When users think they did one thing but actually did something else, they lose their sense of controlling the system because they don't understand the connection between actions and results. The user experience feels random and definitely not empowering.

Some reliability issues can be alleviated by following usability guidelines such as using larger objects and surrounding

them with generous click zones. Others are inherent in any new technology that will have its bugs—that much more reason to enhance user empowerment by designing according to the other interaction principles we have listed in this article.

Lack of undo. Undo! One of the most brilliant inventions of usable computer interfaces seems mostly to have been forgotten. It is very difficult to recover from accidental selections or checking of boxes. First, the result often takes you to a new location. Second, it may not even be obvious what action got you there. For example, if a finger accidentally scrapes an active region, triggering an action, there is almost no way to know why the resulting action took place because the trigger was unintentional.

Novel Interaction Methods

Gestural systems do require novel interaction methods. Indeed, this is one of their virtues. We can tilt and shake, rotate and touch, poke and probe. The results can be extremely effective while also conveying a sense of fun and pleasure. But these interaction styles are still in their infancy, so it is only natural to expect that a great deal of exploration and study still needs to be done.

Shaking has become a standard way of requesting another choice, a choice that seems to have been discovered accidentally, but that also feels natural. Note, however, that although it is easy and fun to shake a small cell phone, shaking a large pad is neither easy nor much fun. Scrolling through long lists can now be done by rapid swiping of the fingers, providing some visual excitement, but we still need to work out the display dynamics, allowing the items to gather speed, to keep going through a form of "momentum," yet to make it possible to see where one is in the list while it whizzes past, and to enable rapid stopping once the desired location seems near.

Although pinching and spreading seem like natural ways of zooming an object out and in, when the dynamics are badly set, the movements are difficult to control. Different applications today use different rules, which end up confusing people. Moreover, even if they could, not all places allow this: another source of confusion.

Rotation and tilting the device are also often used to change the display, although for some applications, such as reading, it has been found necessary to provide a lock to prevent the otherwise natural rotation of the displayed image that would prevent easy reading.

The Promise of Gestural Interfaces

The new interfaces can be a pleasure to use and a pleasure to see. They also offer the possibility of scaling back the sometimes heavy-handed visual language of traditional GUIs that were designed back when nobody had seen a scrollbar. In the early 1980s, usability demanded GUI elements that fairly screamed "click me."

Desktop GUIs are already less neon than Windows 3.0, and we can afford to dial back the visual prominence a bit more on tablets, which will further enhance their aesthetics. But dialed back doesn't mean invisible.

The new displays promise to revolutionize media: News and opinion pieces can be dynamic, with short video instead of still photographs and adjustable figures that can be manipulated instead of static diagrams. *Consumer Reports* could publish its rating tables with reader-controlled weights, so each viewer would have a tailored set of recommendations based upon standardized test results.

The new devices are also fun to use: Gestures add a welcome feeling of activity to the otherwise joyless ones of pointing and clicking.

But the lack of consistency and inability to discover operations, coupled with the ease of accidentally triggering actions from which there is no recovery, threatens the viability of these systems.

We urgently need to return to our basics, developing usability guidelines for these systems that are based upon solid principles of interaction design, not on the whims of the company-interface guidelines and arbitrary ideas of developers.

ABOUT THE AUTHORS Don Norman and Jakob Nielsen are co-founders of the Nielsen Norman group. Norman is a professor at Northwestern University, visiting professor at KAIST (South Korea), and author. His latest book is *Living with Complexity*. Norman can be found at jnd.org.

Nielsen founded the "discount usability engineering" movement for interface design and has invented several usability methods, including heuristic evaluation. He holds 79 U.S. patents, mainly on ways of making the Internet easier to use. Nielsen can be reached at useit.com.

DOI: 10.1145/1836216.1836228