17. "Lemonade Stand," a text game created by Bob Jamison in 1973 for use on time-shared teletype terminals, ported to the Apple II computer in 1979. A copy of "Lemonade Stand" was included with every Apple system sold throughout most of the 1980s.

18. Carver Mead, *Analog VLSI and Neural Systems*.

Button

Søren Pold

Buttons are everywhere in software interfaces, they "initiate an immediate action" and are an essential part of the controls in the modern graphical user interface (GUI). An intensive design effort has gone into the sculpting of buttons, they have become sonified, texturized, sculpted, and various kinds are developed with distinct functionality and signification: push buttons, metal buttons, bevel buttons, round buttons, help buttons, and radio buttons.[1] They appeared from the moment of the earliest graphical user interfaces such as in Xerox's SmallTalk and the Xerox Star computer from the 1970s and early 1980s.[2] Buttons are a cornerstone in contemporary software interfaces. But why and what do they signify, and why are buttons so important and seductive?

Buttons signify a potential for interaction. When the mouse was invented by Douglas Engelbart's team in the 1960s, it was used to click on text and hypertext links. These gradually changed into buttons when the GUI became established. Already ASCII interfaces like DOS shells and the notorious Norton Commander (figure 1) had button-like text boxes to click on when the mouse became a standard interface with PCs. The GUI introduced icons and its buttons gradually became reactive, inverting the black and white colors when they were clicked. Later, in the 1990s, they became increasingly three-dimensional in style as the available screen resolution increased. The interface designer Susan Kare, who had earlier worked on the Macintosh, worked for Microsoft in the late 1980s on what was to become Windows 3.0 (1990), where she replaced "black rectangles with images that looked like three-dimensional 'pressable' buttons."[3] By the mid-1990s 3–D buttons were a fully fledged standard in, for example, Windows 95 (1995) and Mac OS 8.0 (1997).

A button indicates a functional control; something well defined and predictable will happen as a result of the user pressing it. The fact that it is often rendered in 3–D simulates a physical, mechanical cause-and-effect relationship

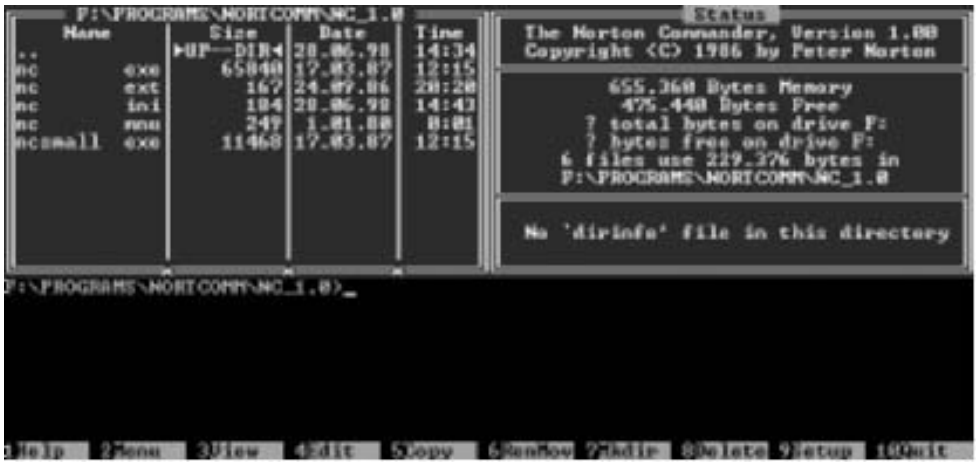

Figure 1 *Norton Commander* (1986)

which is often emphasized by the system event sound of a mechanical button being pressed. This is a simulation of how we know buttons from old machinery and electronics, where the buttons are in fact the mechanical interface, which might switch a relay through a mechanical lever, followed by an audible click and noise from the machinery and electronics. Since the connection is mechanical and not symbolic, such buttons are trustworthy, and one can feel them working tactilely. They do not change functionality; they always precipitate the same action. There is an analog connection between pressing the button and, by the force of one's finger transmitted through a lever, changing the state of the apparatus—as in old tape recorders, where one actually pushed the tape head into place with the button. The computer interface does away with the analog mechanical functionality, but the function of buttons here is to signify the same stable denotation, even though its material basis is gone. That is, interface buttons disguise the symbolic arbitrariness of the digital mediation as something solid and mechanical in order to make it appear as if the functionality were hardwired: they aim to bring the old solid analog machine into the interface. In this sense buttons are a part of a remediation[4] of the machine in the computer interface, a way of dressing it up as something well known and well understood, but there is more to it than this. It points directly to our limited understanding of the computer as a machine and as a medium and how it functions in culture and society.

One pioneer of computer graphics, computer art, and semiotics, Frieder Nake, has described the computer as an instrumental medium that we use in-

strumentally as a tool while communicating with it as a medium, thus it is both machine and mediation simultaneously.[5] Following Nake's concept of the instrumental medium, the computer is a new kind of media-machine that mediates the instrumental or functional and functionalizes the representational medium. That is, function becomes mediated and the mediated representation becomes functional. This chimerical quality, though difficult to grasp from both a functional perspective (e.g., engineering) and from a media perspective (e.g., postmodern media studies and aesthetic theory) has become a standard mode of expression in software interfaces, with the button as a central element of expression.

When pushing a button in an interface—that is, by movement of the mouse, directing the representation of one's hand onto the representation of a button in the interface and activating a script by clicking or double-clicking—we somehow know we are in fact manipulating several layers of symbolic representation and, as such, interacting with a complex mediation of a functional expression, engaging with what Steven Johnson characterizes as the "strange paradoxical quality" of direct manipulation.[6] But we nevertheless see and interpret it as something that triggers a function—and for good reason, since it is designed to perform in this way. It is a software simulation of a function, and this simulation aims to hide its mediated character and acts as if the function were natural or mechanical in a straight cause-and-effect relation. Yet it is anything but this: it is conventional, coded, arbitrary, and representational, and as such also related to the cultural.

Just think about how many codes and values—from programming, commerce, and ideology—are mobilized when you click "buy," pay with your credit card, and download a tune in a proprietary file format with technically and juridically imposed restrictions on how you can use, play, and copy it. The cultural, conventional, and representational elements are disguised or "black-boxed" as pure technical functionality; you do not even realize the consequences of the copy protection technology, the money transfer via your credit card company, or the way the music is produced, commercialized, and regulated by the recording company, the outlet, and the artist. The functional spell is only broken when the software crashes, or when the software becomes reflexive: either through artistic means as in net- and software art, in order to surprise, criticize, or inform; or through juridical necessities such as when submitting to licenses, etc. The installation screens where, before installing the software, one has to accept a lot of restrictions and modes of conduct by

pressing a button are perhaps some of the most perverse examples of using buttons in software. The long intricate message and the easy accept button seem contradictory, and even though you are asked in capitals to read the agreement carefully before using the software, it only seems symptomatically to point to the contradiction. For example when installing Apple's iTunes player, it states that by clicking the button you accept a 4000-word contract stating that you are only licensing the software, that you may only use it to reproduce material which is not in violation of copyright, that you will not use iTunes to develop nuclear missiles, chemical or biological weapons(!), and, among other things, that you will be solely responsible for any damages to your computer or data.

This example highlights how buttons force decisions into binary choices. There is no way of answering that one partially agrees, has not realized the consequences of accepting, or does not care, even though these would probably be franker answers from most users. Buttons are verbs that rule out tenses other than present tense, and rule out modal auxiliary, subjunctive, and other more sophisticated ways in which our language expresses activity. Buttons also designate you as a masterful subject in full control of the situation, which obviously is problematic in many cases, such as the one above, where one cannot oversee, predict, or even understand the consequences of clicking "I accept," or in other examples where the buttons effectively hide the scripts enacted by pressing it, such as in the "buy" example.

But as manufacturers of technological consumer goods from cars and hi-fi equipment to computer hardware and software know, buttons have seductive aesthetic qualities and should provide a satisfying response to the desire to push them. They should evoke confidence by returning a smooth response, not plastickey or cheap, even though it might have nothing to do with functionality. Buttons are tempting—just watch kids in technical museums. Their magnetism may reflect a desire for control or for the capacity to have an effect, and this is combined with a tactile desire that is emphasized by the adding of simulated textures (e.g., metal, shadows, lighting, grooves, 3-D, etc., shown in figure 2), as in the *Mac OS 7.5.3 CD-Player*. That buttons still are important for the success of a product is demonstrated by the iPod's *Apple ClickWheel*, which is the tactical icon for the extremely successful iPod.

In fact the *ClickWheel* points out how software buttons have increasingly become hardware. The *ClickWheel* is a button on the iPod hardware designed to control specific functions of the software, thus materializing the software into the hardware. Other and older examples of software buttons migrating

Figure 2 CD-player from *Mac OS System 7.5.3* (1996)

back to hardware are the mouse itself, buttons on a computer for controlling sound volume or various functions of the operating system (home, end, search) or the function buttons (F1–F12) on the computer keyboard. These kind of soft-hardware buttons are often seen when the universal computer is customized for special use, such as in mobile phones, iPods, game consoles, etc., and they seem to be flourishing currently as seductive branding on fashionable electronic gadgets. A special case is touch screens, where one interacts with the interface by touching the screen and tapping its buttons. Here the interface becomes directly touchable though it is only an illusion which does not exactly feel right—instead of actually touching the interface it feels as if one's finger becomes a mouse. Still, even if next generation touch screen producers feel tempted to produce screens that could automatically sculpt 3-D buttons with a tactile feel to them, it would not solve the paradox of the button as an expression of the interface's mediation of the functional and instrumentation of the representational, as pointed out previously. Software buttons incarnate this paradox. As exemplified by the function buttons, software buttons turned into hardware are often reconfigurable, programmable, and, as such, they reverse the logic of mechanical buttons from giving the interface a hardwired functional trustworthiness to softening the buttons on the box. This both leads to frustration (as when your keyboard layout is accidentally changed) and an at least momentary frisson (e.g., playing computer games or handling SMS's).

Powerful buttons have an unmistakably "trigger happy" feel to them. They make the world feel controllable, accessible, and conquerable, providing "Information at your fingertips" as the slogan goes, or, more broadly, the reduction of society, culture, knowledge, its complexity, countless mediations, and transformations to a "double-click" information society,[7] where everything becomes packaged in manageable and functional scripts activated by buttons

offering easy rewards. From this perspective, the interface button becomes an emblem of our strong desire to handle the increasingly complex issues of our societies by efficient technical means—what one may call the "buttonization" of culture, in which our reality becomes clickable.

In Adrian Ward's artistic software, *Signwave Auto-Illustrator,*[8] there is a big, tempting button in the preferences palette with the caption "Don't push this button," which paradoxically pinpoints and heightens the desire to push it. One could say that by its apparent denial of functional purpose the button self-consciously tempts our desire for the functional experience of tactical control and mastery—a strong ingredient in the aesthetics of the functional interface, even when denied.[9]

Notes

1. Apple Computer, Apple Human Interface Guidelines, Cupertino, CA, Apple Computer, Inc. Retrieved March 20, 2006 from http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/.

2. See Steven Johnson, *Interface Culture: How New Technology Transforms the Way We Create and Communicate*; N. Lineback, "GUI Gallery"; J. Petersen and J. H. Hansen, "MacLab Danmark"; M. Wichary, "GUIdebook, Graphical User Interface Gallery"; M. Tuck, "The Real History of the GUI"; J. Reimer, "A History of the GUI."

3. Susan Kare, "Design Biography." http://www.kare.com/design_bio.html/.

4. J. David Bolter and R. Grusin, *Remediation: Understanding New Media*.

5. Frieder Nake, *Der Computer als Automat, Werkzeug und Medium und unser Verhältnis zu ihm.*

6. Johnson, *Interface Culture.*

7. Bruno Latour, E. Hermant, et al. *Paris Ville Invisible.*

8. Adrian Ward, *Signwave Auto-Illustrator.*

9. Søren Pold, "Interface Realisms: The Interface as Aesthetic Form," in *Postmodern Culture,* vol. 15 no. 2, January 2005.