From the Philosophy of the Open to the Ideology of the User-Friendly

> In the Old Testament there was the first apple, the forbidden fruit of the Tree of Knowledge, which with one taste sent Adam, Eve, and all mankind into the great current of History. The second apple was Isaac Newton's, the symbol of our entry into the age of modern science. The Apple Computer's symbol was not chosen purely at random: it represents the third apple, the one that widens the paths of knowledge leading toward the future.
>
> —Jean-Louis Gassée, *The Third Apple*

Digging to Denaturalize

The second cut into the ground of our technological past in this study of reading/writing interfaces is into the era of the GUI-based personal computer that was preceded by Douglas Engelbart, Alan Kay, and Seymour Papert's experiments with computing and interface design from the mid-1960s to the mid-1970s. This era began with expandable homebrew kits and irrevocably transformed into so-called user-friendly, closed workstations with the release of the Apple Macintosh in late January 1984.[1] Whereas chapter 1 delves into the computing industry's present push to take us more deeply into the era of the interface-free, this chapter uncovers an earlier rupture in the history of computing that partly laid the groundwork for the interface-free.

I look more specifically into the idea that the interface is equal parts user and machine, so that the extent to which the interface is designed to mask its underlying machine-based processes for the sake of the user is the extent to which these same users are disempowered, as they are unable to understand—let alone actively create—using the computer. This chapter

concerns itself with a decade in which we can track the shift from a user-friendly computer as a tool that through a graphical user interface (GUI) encouraged understanding, tinkering, and creativity to a user-friendly computer that used a GUI to create an efficient workstation for productivity and task management, as well as the effect of this shift, particularly on digital literary production. Further, the turn from computer systems based on the command-line interface to those based on "direct-manipulation" interfaces that were iconic or graphical was driven by rhetoric that insisted the GUI, particularly that pioneered by the Apple Macintosh design team, was not just different from the command-line interface but *naturally* better, easier, friendlier. As I outline, the Macintosh was, as Jean-Louis Gassée (who headed up its development after Steve Jobs's departure in 1985) writes without any hint of irony, "the *third apple,*" after the first apple in the Old Testament and the second apple that was Isaac Newton's, "the one that widens the paths of knowledge leading toward the future."[2] It's worth noting that despite Gassée's hyperbole, which I use to demonstrate the ideological fervor of those working for Apple in the 1980s, his vision for Macintosh was quite different from Jobs's in that Gassée helped shepherd into the market three models of the Macintosh—the Mac Plus, the Mac II, and the Mac SE—that were all expandable, unlike the first-generation Macintosh, which prevented users from opening up the computer by giving them a small electrical shock if they did not adhere to the warnings. (I should point out, however, that the device was not deliberately booby-trapped so much as the Macintosh's power supply required very careful handling, a fact that made it all the more convenient to warn users away from opening it up at all.) While these later models of the Macintosh included expansion slots, which returned Apple philosophically to the era of Steve Wozniak's Apple II—whose eight expansion slots permitted a whole range of display controllers, memory boards, hard disks, etc.—it seems clear that the return of Jobs to Apple in 1997 meant, and continues to mean, a return to keeping the inner

workings of Apple computers and computing devices firmly closed off to users.

Despite studies released since 1985 that clearly demonstrate GUIs are not necessarily better than command-line interfaces in terms of how easy they are to learn and to use, Apple—particularly, under Jobs's leadership—created such a convincing aura of inevitable superiority around the Macintosh GUI that to this day the same "user-friendly" philosophy, paired with the no longer noticed closed architecture, fuels consumers' religious zeal for Apple products.[3] I have been an avid consumer of Apple products since I owned my first Macintosh PowerBook in 1995. As I write in chapter 1, however, what concerns me is that the user-friendly now takes the shape of keeping users steadfastly unaware and uninformed about how their computers, their reading/writing interfaces, work, let alone how they shape and determine their access to knowledge and their ability to produce knowledge. As Wendy Chun points out, the user-friendly system is one in which users are given the ability to "map, to zoom in and out, to manipulate, and to act," but the result is only a "*seemingly* sovereign individual" who is mostly a devoted consumer of ready-made software and ready-made information to which whose framing and underlying (filtering) mechanisms she or he is not privy.[4]

Thus, the content of this argument is about reversals, and its methodology is defined by tracing the messy, nonlinear rupture I describe in chapter 1—that the shift to the ideology of the user-friendly via the GUI is expressed in contemporary multitouch, gestural, and ubiquitous computing devices, such as the iPad and the iPhone, whose interfaces are touted as utterly invisible and whose inner workings are therefore de facto inaccessible. In this earlier chapter I also outline how this full realization of frictionless, interface-free computing, at least partly born out of the mid-1980s, is in turn critiqued by works of activist digital media poetics.[5]

Using a media archaeology–inspired methodology to understand the historical moment at hand, we can see that activist

media poetics played out quite differently in the 1980s than it did in the 1960s' era of the typewriter, as the 1980s was an era newly oriented toward the efficient completion of tasks over and beyond a creative use or misuse of the computer. Arguably, one reason for the heightened engagement in hacking type(writing) in the mid-1960s to the mid-1970s was that the typewriter had become so ubiquitous in homes and offices that it had also become invisible to its users. The point at which a technology saturates a culture is the point at which writers and artists, whose craft is utterly informed by a sensitivity to their tools, begin to break apart that same technology to once again draw attention to the way in which it offers certain limits and possibilities to thought and expression. There are examples of digital poems that inherit this emphasis on making, doing, and hacking, but once again, the vast majority of these works did not appear until both the personal computer and the user-friendly computer whose GUI was designed to keep the user passively consuming technology rather than actively producing it became practically ubiquitous. As I discuss in the following section, activist media poetics in the early to mid-1980s mostly took the form of experimentation with digital tools that at the time were new to writers—an experimentation that at least under the terms set by McKenzie Wark's *Hacker Manifesto,* certainly *could* be framed as hacking. Wark writes that "hackers create the possibility of new things entering the world" and that "the slogan of the hacker class is not the workers of the world united, but the workings of the world untied."[6] As I discuss later in the chapter, works by bpNichol, Geof Huth, and Paul Zelevansky did not make the command-line interface visible so much as they openly played with and tentatively tested the parameters of the personal computer as a still-new writing technology. This kind of open experimentation almost entirely disappeared for several years once Apple Macintosh's design innovations, as well as their marketing, made open computer architecture and the command-line interface obsolete and GUIs pervasive.

Open, Extensible, Flexible: NLS, Logo, Smalltalk

This chapter tackles the notion of the digital interface as a meshing of, even a friction between, human and machine. The degree to which a GUI masks the digital machine for the sake of a more human-like experience is the degree to which users no longer have access to (understanding) both the mechanisms and the flow of information underlying the machine. Likewise, what early human–computer interaction (HCI) designers and researchers struggled with was that the degree to which the interface *unmasks* the digital machine and provides more direct access to the underlying mechanisms is the degree to which it may become more difficult for nonexperts to learn how to use computers.

The interface is only superficially, as Steven Johnson concisely puts it in his canonical *Interface Culture* (1997), "software that shapes the interaction between user and computer . . . a kind of translator" that makes possible the representation of the computer to the user—in the GUI system, through metaphors.[7] As in any translation, there is never a perfect equivalent of the one in the language of the other, and so here, I make clear that we need to think about the nature of computer-to-human translation that takes place via the interface. We can think of the interface as a complex philosophical entity whose translation mechanism is not so much related to natural-language translation as it is to a threshold, along the lines of Matthew Fuller's definition of an interface as containing elements of "the underlying structure of [both] the program and the user."[8] In this way, we can look back and see the philosophy of computing embodied by the early experiments and writing of Douglas Engelbart, Seymour Papert, Alan Kay, and (even) Steve Wozniak as weighted toward a precise midpoint between computer/program and user, a balance that then irrevocably shifted to the user by 1984 with the release of the Apple Macintosh and its icon-based GUI. By contrast, Engelbart, Papert, Kay, and Wozniak show us that a user-friendly computer and graphical

interface need not close access to the computer/program for the sake of the user. It can be designed instead to empower users to access and then understand the hardware and the software basics of computing and, ultimately, to create their own tools and applications.

It is, then, not necessarily that the GUI per se is responsible for the creation of Chun's "*seemingly* sovereign individual" but rather that a particular philosophy of computing and design underlying a model of the GUI has become the standard for nearly all interface design. The earliest example of a GUI-like interface whose philosophy was fundamentally different from that of the Macintosh was Douglas Engelbart's oN-Line System (NLS), which he began work on in 1962 and famously demonstrated in 1968 at the Fall Joint Computer Conference in San Francisco. While his "interactive, multi-console computer-display system" with keyboard, screen, mouse, and something he called a chord handset (which allowed the user to issue commands to the computer by pressing different combinations of the five keys) is commonly cited as the originator of the GUI, Engelbart wasn't interested in creating a user-friendly machine so much as he was invested in "augmenting human intellect."[9] As he first put it in 1962, this augmentation meant "increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems."[10] The NLS was not about providing users with ready-made software and tools from which they chose or consumed but rather about *bootstrapping,* or "the creation of tools for expert computer users," and providing the means for users to create better tools, or tools better suited to their individual needs.[11] In his document editing program, this emphasis on tool building and customization came out of an augmented intellect in Engelbart's provision of view control, which allowed users to determine how much text they saw on the screen, as well as the form of that view—for example, line truncation and content filtering—and of chains of views, which allowed the user to link related files.[12] The NLS's use of view

control and chains of view provided a far more direct method of manipulating information—despite the added graphical layer between computer and user—than did the dominant method at that time of punch cards, which very often made the user not a user at all, certainly not one who interacted with the computer but handed over a numerical problem for the computer to solve.[13]

Underlining the fact that the history of computing is resolutely structured by stops, starts, and ruptures rather than by a series of linear firsts, in the year before Engelbart gave his "mother of all demos" Seymour Papert and Wally Feurzeig began work on a learning-oriented programming language called Logo that was explicitly for children but implicitly for learners of all ages. Throughout the 1970s Papert and his team at MIT conducted research with children in nearby schools as they tried to create a version of Logo that was defined by "modularity, extensibility, interactivity, and flexibility."[14] As I discuss briefly in the next section in relation to literary experiments on the Apple II, it was the most popular home computer throughout the late 1970s and until the mid-1980s, and given its open architecture, in 1977 Logo licensed a public version for Apple II computers, as well as for the less popular Texas Instruments TI 99/4. In 1980 Papert published the influential *Mindstorms: Children, Computers, and Powerful Ideas,* in which he makes claims about the power of computers that are startling for a contemporary readership steeped in an utterly different notion of what accessible or user-friendly computing might mean. Describing his vision of "computer-aided instruction" in which "the child programs the computer" rather than one in which the child adapts to the computer or, even, is taught by the computer, Papert asserts that children thereby "embark on an exploration about how they themselves think. . . . Thinking about thinking turns the child into an epistemologist, an experience not even shared by most adults."[15] Two years later, in a February 1982 issue of *Byte* magazine, Logo was advertised as a general-purpose tool for thinking, with a degree of intellectuality rare

for any advertisement: "Logo has often been described as a language for children. It is so, but in the same sense that English is a language for children, a sense that does not preclude its being ALSO a language for poets, scientists, and philosophers."[16] Moreover, for Papert, thinking about thinking by way of programming happens largely when the user encounters bugs in the system and has to then identify where the bug is to remove it: "One does not expect anything to work at the first try. One does not judge by standards like 'right—you get a good grade' and 'wrong—you get a bad grade.' Rather one asks the question: 'How can I fix it?' and to fix it one has first to understand what happened in its own terms."[17] Learning through doing, tinkering, experimentation, and trial and error is, then, how one comes to have a genuine computer literacy.

The year after Papert and Feurzeig began work on Logo and the same year as Engelbart's NLS demo, Alan Kay commenced work on the never-realized Dynabook, which was produced as an "interim Dynabook" in 1972 in the form of the GUI-based Xerox Alto, which ran the Smalltalk language. Kay thereby introduced the notion of "*personal* dynamic media" for "children of all ages" that "could have the power to handle virtually all of its owner's information-related needs."[18] Kay, then, along with Engelbart and Papert—all working at the same time, independently yet often influencing each other—very clearly understood the need for computing to move from the specialized environment of the research lab into people's homes by way of a philosophy of the user-friendly oriented toward the flexible production (rather than the rigid consumption) of knowledge.[19] It was a realization eventually shared by the broader computing community, for by 1976 *Byte* magazine was publishing editorials such as "Homebrewery vs the Software Priesthood," which declared, "The movement towards personalized and individualized computing is an important threat to the aura of mystery that has surrounded the computer for its entire history" (see Figure 12).[20] Moreover:

> The movement of computers into people's homes makes it important for us personal systems users to focus our efforts toward having computers do what we want them to do rather than what someone else has blessed for us. . . . When computers move into peoples' homes, it would be most unfortunate if they were merely black boxes whose internal workings remained the exclusive province of the priests. . . . Now it is not necessary that everybody be a programmer, but the potential should be there.[21]

It was the potential for programming or, simply, for novice and expert use via an open, extensible, and flexible architecture that Engelbart, Papert, and Kay sought to build into their models of the personal computer to ensure that home computers did *not* become "merely black boxes whose internal workings remained the exclusive province of the priests." As Kay later exhorted his readers in 1977, "Imagine having your own *self-contained knowledge manipulator* in a portable package the size and shape of an ordinary notebook."[22] Designed to have a keyboard, an NLS-inspired chord keyboard, a mouse, a display, and windows, the Dynabook would have allowed users to realize Engelbart's dream of a computing device that gave them the ability to create their own ways to view and manipulate information. Rather than the overdetermined post-Macintosh GUI computer, which has been designed to preempt each user's every possible need through the creation of an overabundance of ready-made tools and whose underlying workings are now utterly black-boxed such that, as homebrewers protested in the mid-1970s, "those who wish to do something different will have to put in considerable effort," Kay wanted a machine that was "designed in a way that any owner could mold and channel its power to his own needs . . . a metamedium, whose content would be a wide range of already-existing and not-yet-invented media."[23] More, Kay understood from reading Marshall McLuhan's *Understanding Media* that the design of

"...these are, and of right ought to be, Free and Independent. . . ."

— John Hancock, et al

Personal computing people stand to be largely independent of the priesthood because they are strikingly sophisticated and because they freely share their ideas. A very good example of both these traits can be found in the nearly spontaneous generation of Tiny BASIC through the medium of the People's Computer Company and Dr Dobb's *Journal of Computer Calisthenics and Orthodontia*. One issue published some rough design notes for a machine independent Tiny BASIC, but that was only the beginning. The next few issues published refinements on the design and later ones included an implementation in an interpretive language and then both octal and annotated source programs realizing the interpreter and the entire system in 3 K of 8080 code. To top it off, the whole project was done by far-flung individuals in less than a year.

While Tiny BASIC is a very striking example of what amateurs can do when they work together, we cannot afford to ignore its extreme dependence on good fortune to bring it to pass. Your own copy of BYTE magazine is another example; it is the result of one man's frustration at making his own computer work and his desire to let others profit by his experience. We've been very lucky to have a few people with high ideals to point the way for us, but we would be ill advised to depend on having these fortunate circumstances continue. The time is ripe for the community of personal computing enthusiasts to start thinking seriously about supplying its own steam to back up the energies put out by a few people with strong motivations to help launch the personal computing movement. It's launched now, and we have to provide the impetus and direction to make sure it develops in a way beneficial to the community at large.

A good example of a means to distribute software which divides the effort fairly and in a way nobody seems to mind is the software exchange of the Homebrew Computer Club in the San Francisco Bay Area. At each meeting (every two weeks) there is a table covered with paper tapes of programs contributed by all and sundry. Anybody is welcome to take any tape at all, subject only to the proviso that each copy taken from one meeting be replaced by at least one

FIGURE 12. *An image from the editorial "Homebrewery vs the Software Priesthood" that appeared in* Byte *magazine in October 1976.*

this new metamedium was no small matter, for the use of a medium changes an individual's and a culture's thought patterns.[24] Clearly, Kay wanted thought patterns to move toward a literacy that involved reading and writing in the new medium instead of the unthinking consumption of ready-made tools, for he wrote, "The ability to 'read' a medium means you can *access* materials and tools created by others. The ability to 'write' in a medium means you can *generate* materials and tools for others. You must have both to be literate."[25]

While Kay envisioned that the GUI-like interface of the Dynabook would play a crucial role in realizing this metamedium, the Smalltalk software driving this interface was equally necessary. Its goal was, as the principal designer, architect, and implementer Daniel Ingalls wrote in a 1981 special issue of *Byte* dedicated to Smalltalk, "to provide computer support for the creative spirit in everyone."[26] While 1971 was the year Alan Kay's Learning Research Group at Xerox PARC developed a working version of Smalltalk—also introducing for the first time, via Smalltalk-71, the term *object oriented programming* (OOP), a paradigm now supported by nearly all modern programming languages—1980 was the year Kay's group released Smalltalk-80 to the public, a version that was then featured in the aforementioned issue of *Byte*. Although examining how the workings of Smalltalk and OOP manifested their overarching philosophy is important, my interest is in tracking this philosophy as part of a broader trend in computing from the 1970s until the mid-1980s—one that is reflected largely in the discourse around GUIs and the user-friendly. Those who worked on Smalltalk saw it as a fundamental break from the philosophy of the closed, elitist, decidedly undemocratic "software priesthood." Not surprisingly, Kay and his collaborators began working intensely with children after the creation of Smalltalk-71. Influenced by developmental psychologist Jean Piaget, as well as Kay's own observation of Papert and his colleagues' use of Logo in 1968, Smalltalk relied heavily on graphics and animation through one particular incarnation of the

GUI: the Windows, Icons, Menus, and Pointers (WIMP) interface. Kay writes that in the course of observing Papert using Logo in schools, he realized that these children were "doing real programming":

> This encounter finally hit me with what the destiny of personal computing *really* was going to be. Not a personal dynamic *vehicle,* as in Engelbart's metaphor opposed to the IBM "railroads", but something much more profound: a personal dynamic *medium*. With a vehicle one could wait until high school and give "drivers ed", but if it was a medium, it had to extend into the world of childhood.[27]

As long as the emphasis in computing was on learning—especially through making and doing—the target demographic was going to be children, and as long as children could use the system, then so too could any adult, provided they understood the underlying structure, the how and the why, of the programming language. As Kay astutely remarks, "*We make not just to have, but to know.* But the having can happen without most of the knowing taking place."[28] As he goes on to point out, designing the Smalltalk user interface shifted the purpose of interface design from "access to functionality" to an "environment in which users learn by doing."[29]

Smalltalk designers did not completely reject the notion of ready-made software so much as they sought to provide users with a set of software building blocks that they could combine and/or edit to create their own customized systems. As Trygve Reenskaug, a visiting Norwegian computer scientist with the Smalltalk group at Xerox PARC in the late 1970s, put it:

> The new user of a Smalltalk system is likely to begin by using its ready-made application systems for writing and illustrating documents, for designing aircraft wings, for doing homework, for searching through old court decisions,

> for composing music, or whatever. After a while, he may become curious as to how his system works. He should then be able to "open up" the application object on the screen to see its component parts and to find out how they work together.[30]

With an emphasis on learning and building through an open architecture, Adele Goldberg—codeveloper of Smalltalk along with Alan Kay and author of most of the Smalltalk documentation—describes, in the 1981 special issue of *Byte,* the Smalltalk programming environment as one that sets out to defy the conventional software-development environment (see Figure 13).

In Figure 13, the Taj Mahal in the left-hand Figure 1 "represents a complete programming environment, which includes the tools for developing programs as well as the language in which the programs are written. The users must walk whatever bridge the programmer builds."[31] By contrast, the right-hand Figure 2 represents a Taj Mahal in which the "software priest" is transformed into one who merely provides the initial shape of the environment, which programmers can then modify by building "application kits" or "subsets of the system whose parts can be used by a nonprogrammer to build a customized version of the application."[32] The user or nonprogrammer is, then, an active builder in a dialogue with the programmer instead of a passive consumer of a predetermined and, perhaps, overdetermined environment.

At roughly the same time as Kay began work on Smalltalk in the early 1970s, he was involved with the team of designers working on the NLS-inspired Xerox Alto, which was developed in 1973 as, again, an "interim Dynabook" that had a three-button mouse and a GUI working in conjunction with the desktop metaphor and that ran Smalltalk. While only several thousand noncommercially available Altos were manufactured, its GUI and its network capabilities, as team members Chuck Thacker and Butler Lampson believe, made it quite likely the

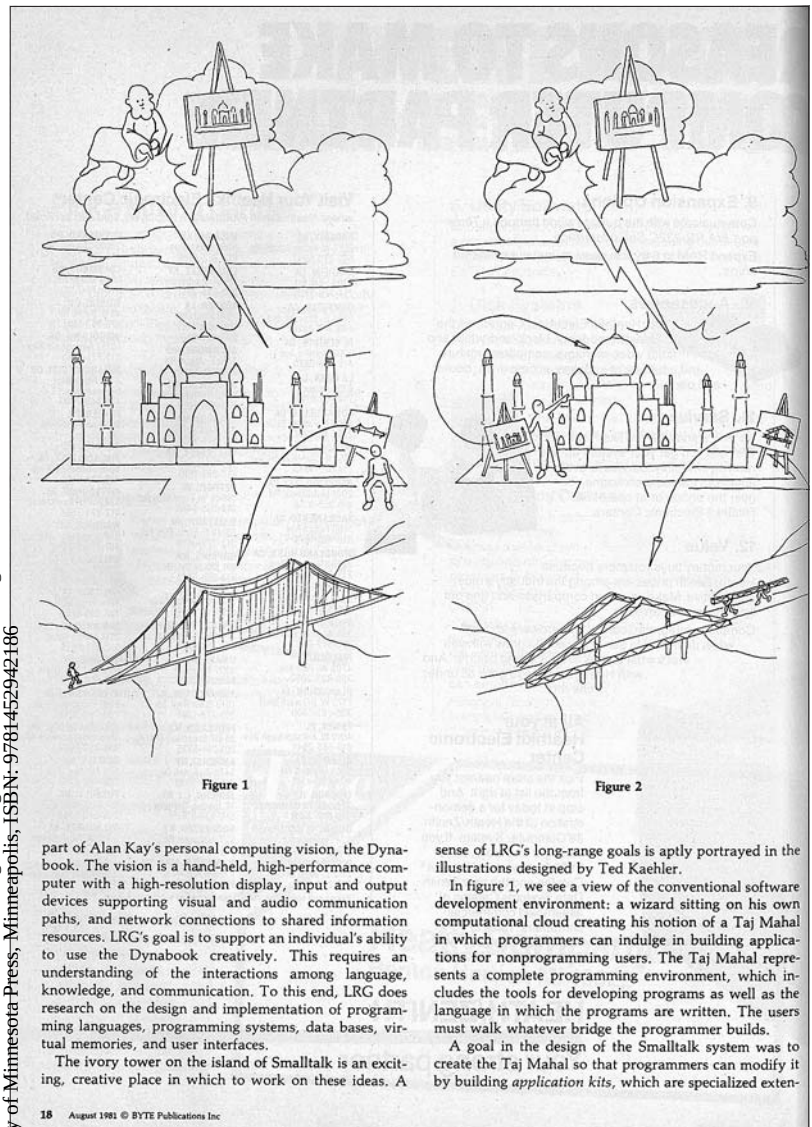

Figure 1

Figure 2

part of Alan Kay's personal computing vision, the Dynabook. The vision is a hand-held, high-performance computer with a high-resolution display, input and output devices supporting visual and audio communication paths, and network connections to shared information resources. LRG's goal is to support an individual's ability to use the Dynabook creatively. This requires an understanding of the interactions among language, knowledge, and communication. To this end, LRG does research on the design and implementation of programming languages, programming systems, data bases, virtual memories, and user interfaces.

The ivory tower on the island of Smalltalk is an exciting, creative place in which to work on these ideas. A sense of LRG's long-range goals is aptly portrayed in the illustrations designed by Ted Kaehler.

In figure 1, we see a view of the conventional software development environment: a wizard sitting on his own computational cloud creating his notion of a Taj Mahal in which programmers can indulge in building applications for nonprogramming users. The Taj Mahal represents a complete programming environment, which includes the tools for developing programs as well as the language in which the programs are written. The users must walk whatever bridge the programmer builds.

A goal in the design of the Smalltalk system was to create the Taj Mahal so that programmers can modify it by building *application kits*, which are specialized exten-

FIGURE 13. *Image by Adele Goldberg in a special issue of* Byte *magazine from 1981 on Smalltalk in which she contrasts the conventional philosophy of software driven by "wizards" (Figure 1) and that provided by Smalltalk for the benefit of the programmer/user (Figure 2).*

first computer explicitly called a "personal computer." By 1981 Xerox had designed and produced a commercially available version of the Alto called the 8010 Star Information System, which was sold along with Smalltalk-based software. As Jeff Johnson et al. point out, the most important connection between Smalltalk and the Xerox Star lay in the fact that Smalltalk could clearly illustrate the compelling appeal of a graphical display that the user accessed via mouse, overlapping windows, and icons (see Figure 14).[33]

The significance of the Star for this chapter is, however, partly the indisputable impact it had—or rather, Smalltalk had—on the GUI design of first the Apple Lisa and then the Macintosh. Its significance is also in the way it was labeled clearly as a workstation for "business professionals who handle information" rather than as a metamedium or as a tool for creating or for thinking about thinking that could be encompassed by the term *workstation,* as we can see in Douglas Engelbart's definition of it as a "portal into a person's 'Augmented Knowledge Workshop'—the place in which he finds the data and tools with which he does his knowledge work."[34] But the Star's interface, which was the first commercially available computer born out of work by Engelbart, Papert, and Kay that attempted to satisfy both novice and expert users in providing an open, extensible, flexible environment and that also happened to be graphical, was conflicted at its core. While in some ways the Star was philosophically very much in line with the open thinking of Engelbart, Papert, and Kay, in other ways its philosophy as much as its GUI directly paved the way to the closed architecture and consumption-based design of the Macintosh.

Take, for example, the overall design principles of the Star, which were aimed at making the system seem "familiar and friendly." Designers David Canfield Smith, Charles Irby, Ralph Kimball, and Bill Verplank avowed, in a 1982 special issue of *Byte,* to avoid the characteristics listed on the right while adhering to a schema that exemplified the characteristics listed on the left:

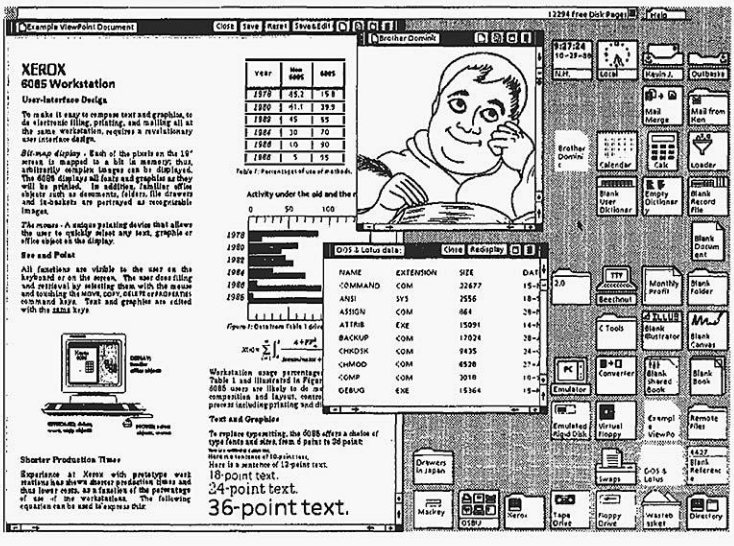

Figure 1. ViewPoint screen image. Star's bitmapped display, once unique in the marketplace, is now much more common. Such a display permits WYSIWYG editing, display of proportionally spaced fonts, integrated text and graphics, and graphical user interfaces.

windows and then cut and pasted together. For example, a MacDraw drawing put into a Microsoft Word or Aldus Pagemaker document can no longer be edited; rather, the original must be re-edited with MacDraw and then substituted for the old drawing in the document.

Not even Star is fully integrated in the sense used here. For example, though the original structured graphics editor, the new one (see "History of Star development" below), and the table and formula editors all operate inside text files, spreadsheets and freehand drawings are currently edited in separate application windows and transferred into documents, where they are no longer fully editable.

User-interface level. Star's user interface is its most outstanding feature. In this section we discuss important aspects of the interface in detail.

Desktop metaphor. Star, unlike all conventional systems and many window- and mouse-based ones, uses an analogy with real offices to make the system easy to learn. This analogy is called "the Desktop metaphor." To quote from an early article about Star:

> Every user's initial view of Star is the Desktop, which resembles the top of an office desk, together with surrounding furniture and equipment. It represents a working environment, where current projects and accessible resources reside. On the screen are displayed pictures of familiar office objects, such as documents, folders, file drawers, in-baskets, and out-baskets. These objects are displayed as small pictures, or icons.
>
> The Desktop is the principal Star technique for realizing the physical office metaphor. The icons on it are visible, concrete embodiments of the corresponding physical objects. Star users are encouraged to think of the objects on the Desktop in physical terms. You can move the icons around to arrange your Desktop as you wish. (Messy Desktops are certainly possible, just as in real life.) You can leave documents on your Desktop indefinitely, just as on a real desk, or you can file them away.[1]

Having windows and a mouse does not make a system an embodiment of the Desktop metaphor. In a Desktop metaphor system, users deal mainly with data files, oblivious to the existence of programs. They do not "invoke a text editor," they "open a document." The system knows the type of each file and notifies the relevant application program when one is opened.

Most systems, including windowed ones, use a Tools metaphor, in which users deal mainly with applications as tools. Users start one or more application programs (such as a word processor or spreadsheet), then specify one or more data files to edit with each. Such systems do not ex-

FIGURE 14. *Screenshot of the Xerox Star desktop that appeared in Jeff Johnson et al.'s "The Xerox Star: A Retrospective" in a 1989 issue of* Computer.

Easy	*Hard*
concrete	abstract
visible	invisible
copying	creating
choosing	filling
recognizing	generating
editing	programming
interactive	batch[35]

While there's little doubt that ease of use was central to Engelbart, Papert, and Kay—often brought about through interactivity and making computer operations and commands visible—the avoidance of "creating," "generating," and "programming" could not be further from their vision of the future of computing. This divided loyalty to two different notions of the user-friendly was more specifically exemplified by the Star's system of commands. Rather than typing out a command from memory via the command-line interface or, even, selecting a command from a menu, commands on the Star took the form of icons that functioned, as the designers describe it, as both noun and verb. The noun was whatever object on the screen the user wished to manipulate, whether file or document or application, and the verb was the type of action or manipulation the user wished to perform. Selection took place by the user hovering the cursor over the object, clicking the mouse button to select the action, and then hitting the Next key on the keyboard to select content from the next field in the document. Other commands that appeared on the keyboard included Find, Open, and Close. Curiously, the designers' explanation of Star's commands ends with the declaration, "Since Star's generic commands embody fundamental underlying concepts, they are widely applicable. . . . Few commands are required. This simplicity is desirable in itself, but it has another subtle advantage: *it makes it easy for users to form a model of the system. What people understand, they can use.*"[36] In other words, at the same time the

Star precluded creating, generating, and programming through its highly restrictive set of commands in the name of simplicity (restrictions that most certainly excluded certain creative possibilities), it also wanted to promote users' understanding of the system as a whole—although again, this particular incarnation of the GUI represented the beginning of a shift toward only a superficial understanding of the system. Without a fully open, flexible, and extensible architecture, the home computer became less a tool for learning and creativity and more a tool for simply "handling information."

Writing as Tinkering: The Apple II and bpNichol, Geof Huth, and Paul Zelevansky

We can clearly see this shift from the philosophy of the open to the ideology of the user-friendly work machine not only in the structure of Steve Wozniak's Apple II versus Steve Jobs's Apple Macintosh but also in the utterly different marketing strategies for these two machines. Wozniak's Apple II used a command-line interface instead of a GUI and was aligned philosophically with homebrewery in that its eight expansion slots allowed users to add on a range of devices, including display controllers, memory boards, and hard disks, which meant its open architecture was explicitly for tinkering and, thus, creativity. Writing for *Byte* in May 1977, the month before the public release of the Apple II, Wozniak declared:

> I designed the Apple-II to come with a set of standard peripherals, in order to fit my concept of a personal computer. In addition to the video display, color graphics and high resolution graphics, this design includes a keyboard interface, audio cassette interface, four analog game paddle inputs . . . three switch inputs, four 1 bit annunciator outputs, and even an audio output to a speaker. Also part of the Apple-II design is an 8 slot motherboard for IO.[37]

In the months leading up to its release, the Apple II was advertised as not only a task-management machine but also a means for imagination and invention:

> You can use your Apple to analyze the stock market, manage your personal finances, control your home environment, and to invent an unlimited number of sound and action video games. That's just the beginning. . . . *You don't want to be limited by the availability of pre-programmed cartridges. You'll want a computer, like Apple, that you can also program yourself.* . . . The more you learn about computers, the more your imagination will demand. So you'll want a computer that can grow with you as your skill and experience with computers grows. Apple's the one.[38]

Eight months later, in November 1977, Apple even issued a contest for "the most original use of an Apple since Adam," with creative use in near diametrical opposition to Gassée's later framing of the Macintosh as a computer that was, in and of itself and regardless of use, the most original "apple" since Newton.

Not surprisingly, then, the Apple II was by the early 1980s the first home computer that appealed to writers looking to experiment with this new medium of expression—writers who were keen to take up John Cage's injunction from 1966 to use a computer not as a labor-saving device but rather as one that increased work for the writer mostly insofar as the computer's graphical, algorithmic, and interactive capabilities encouraged experiments with form.[39] It also made sense that writers chose the Apple II over other available home computers of the time. Even though sales of the Apple II were initially slow in comparison with those of the much less expensive Commodore PET and TRS-80 Model 1, by 1981, once Apple had added their floppy-drive accessory and then both started a highly effective ad campaign (claiming that it was the "best-selling personal computer") and created the first-ever spreadsheet application,

VisiCalc, the Apple II was the best-selling personal computer.[40] Just two years later, in 1983, Apple released arguably its best-selling computer, the IIe, which crucially for writers not only allowed uppercase and lowercase letters but also had an eighty-column display, in contrast with the first-generation Apple II, which was uppercase only and had a forty-column display.

Canadian experimental writer bpNichol not only promptly purchased an Apple IIe the year it was released but also began work on one of the first published works of digital literature, *First Screening,* a series of twelve kinetic poems written in the Apple BASIC programming language (see Figure 15).[41] Given his typewriter-based experiments with highly visual, permutational, DIY-oriented, and processual concrete poems, which I discuss in chapter 3, coupled with his McLuhan-inspired understanding of writing tools as extensions of the writer, it is not surprising that Nichol's writing experiments extended to the computer, exploited the possibilities of a screen-based medium, and so resulted in the creation of these twelve kinetic, cinematic poems. In fact, as Nichol acknowledges in the accompanying printed matter from 1984, he was surprised that in the process of composing *First Screening* in BASIC,

> concerns that had been present for me in the mid-60s, issues of composition and content i was confronting while working with my early concrete poems, suddenly found a new focus. In fact, i was finally in a position to create those filmic effects that i hadn't had the patience or skill to animate at that time. . . . Computers & computer languages also open up new ways of expressing old contents, of revivifying them. One is in a position to make it new.[42]

Because the poems in *First Screening* move soundlessly across a black computer screen, the work is new in how it positions itself halfway between film and sound/concrete poetry and self-consciously (mis)uses the filmic medium to create one of the first kinetic digital poems.

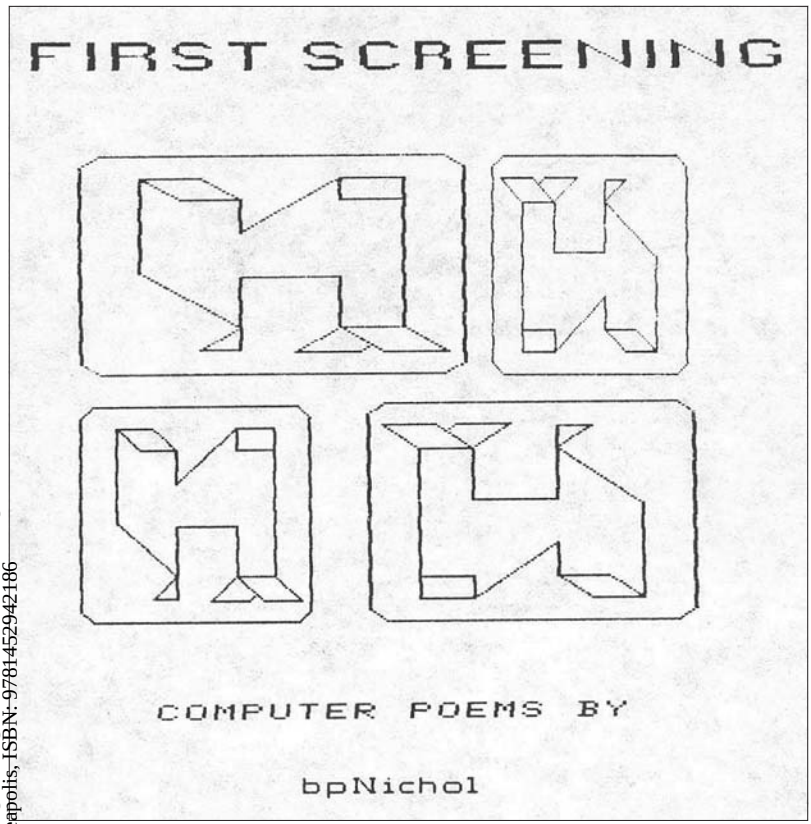

FIGURE 15. *A concrete poem by bpNichol presumably made on and printed from his Apple IIe. The poem appears on an insert that came with the 5.25-inch floppy for his 1983–84* First Screening. *Reprinted by permission of Eleanor Nichol on behalf of the bpNichol Estate.*

In *First Screening* it appears as though Nichol—writing at the very beginning of the era of the personal computer—understands the ease with which the digital computer has an entirely different effect on the body than that of a reading/writing machine such as the typewriter. For example, midway through the screening, the reader/viewer is introduced to "ANY OF YOUR LIP: a silent sound poem for Sean O'Huigin." The title of this piece alone gestures to the absent presence of the body.

Once the poem begins, we see/read the kinetic permutations that move between "MOUTH" and "mouth," "myth" and "MOUTH," "math" and "MOUTH," "mate" and "MOUTH," "maze" and "MOUTH," and "amaze" and "MOUTH" and then the alternation between "ing," "amaze," and "MOUTH," which closes with the repeated flashing of "ing" and, finally, "MOUTH." That said, while the poem is perhaps silent because of the limits of Nichol's own programming know-how (not to mention the limited sound capabilities of the Apple IIe itself), it is noticeable how this paradoxically silent sound poem draws attention to its silence at the same time it enacts and perhaps even encourages readerly interactivity. Especially with the repeated flashing of "ing" at the end of the poem, a verb ending that signals generalized or uncompleted action, "ANY OF YOUR LIP" invites readers to sound out or to "mouth" the words as they try to make sense of the connections between the words while they flash across the screen.

The poems in *First Screening* are not interactive in the sense to which we are accustomed, and the underlying code of the poems shows an iteration of interactivity that does not depend on clicking links. Looking at the BASIC code gives a clear sense of the permutational nature of the kinetic poems, as Nichol carefully moves each letter up and down the vertical axis through the VTAB command and across the screen with HTAB. For example, the following are the first four lines of code for the poem "SAT DOWN TO WRITE YOU THIS LETTER":

```
640 VTAB 12: HTAB 5: PRINT "AT DOWN TO WRITE YOU
    THIS POEM S"
645 VTAB 12: HTAB 5: PRINT "T DOWN TO WRITE YOU THIS
    POEM SA"
650 VTAB 12: HTAB 5: PRINT "DOWN TO WRITE YOU THIS
    POEM SAT"
655 VTAB 12: HTAB 5: PRINT "DOWN TO WRITE YOU THIS
    POEM SAT"
```

The fourth line of code then prints "DOWN TO WRITE YOU THIS POEM SAT" on-screen (see Figure 16).

More, as Jim Andrews astutely discovered in the process of putting *First Screening* online, the twelfth poem does not appear on-screen, as it is instead nested in the last eight lines of the code—a poem that is also one of the first works of codework, or literary writing that is code but not necessarily executable.[43] A reader would discover this piece only if she or he understood the underlying workings of the poem, rather than simply taking in its on-screen effects, and noticed that on line 116 was a REM (or remark, a way of leaving explanatory comments in the code) that states, "FOR FURTHER RE-MARKS LIST 3900,4000." Ideally, this statement would prompt the curious reader to type, "LIST 3900,4000," and view the following further "RE-MARKS":

```
3900 REM ARK
3905 REM BOAT
3910 REM AIN
```

FIGURE 16. *Screenshot of an emulated version of what appears on-screen as a result of line 650 of the Apple BASIC code of bpNichol's* First Screening.

```
3915 REM RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN
     RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN
     RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN
     RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN RAIN
     RAIN RAIN RAIN RAIN RAIN
3920 REM BOAT
3925 REM ARK
3930 REM BOW
3935 REM ARC
4000 END
```

Nichol begins his permutational concrete poem by breaking apart "REMARK" to form "REM" and "ARK," which is followed by "REM BOAT" to make sure we understand this *ark* is not only biblical—rather than the French-derived bow, sometimes spelled *arc*—but also a reference to Noah's ark, not the Ark of the Covenant. Continuing his permutational punning, "RE-MARK" is then turned into "REM AIN," the remains of which produce forty appearances of the word "RAIN." After leaving "ARK," "BOW" appears as an "ARC" across the sky that is, this time, a symbol of the promise God made with Noah to never again flood the earth. This work is, again, not an example of activist media poetics in the sense that becomes more prevalent once the model of the closed computer with an invisible GUI is ubiquitous but rather, given the homebrew-inspired open architecture of the Apple II, of writing as DIY tinkering.

First Screening was influential enough among experimental writers of the time that a few years later, in 1987, Geof Huth produced "Endemic Battle Collage"—what he called "aural and kinetic poems"—for the Apple IIe, in the tradition of Nichol's earlier kinetic/permutational poems (see Figure 17).[44] Huth clearly identified with Nichol's tinkering with the limits and possibilities of writing media. He writes, "My work is based on thinking about what new tools to use and what new possibilities to achieve. bpNichol, the poet I most identify with, seemed to me a poet who understood this and practiced this himself." A

fundamental difference between "Endemic Battle Collage" and *First Screening* is, however, that the former demonstrates a significantly more sophisticated understanding of Apple BASIC in that it incorporates color and sound. The letters twirl, spin, and rotate in more complex patterns, and Huth plays with a system of highlighting words on-screen as a way to play with white text on a black background and black text on a white background. It is also worth noting that by 1987 the Apple Macintosh had been available for three years, yet Huth still chose to experiment with the Apple IIe. Other than the fact, I would argue, that the IIe was a more appropriate machine for literary experimentation, it was also significantly less expensive and so more appealing to writers and artists, for in 1987 one could purchase a IIe for $1,400, whereas a Macintosh retailed for about $2,500.

Given this connection between those who sought to extend their formal and medium-specific writing experiments to the Apple II and its predecessors, it is no coincidence that those working with artists books (a genre concerned with playing with material dimensions and with conventions of the book as a technology) also looked to the Apple II as a means to create a new form that was a hybrid nestled between the computer and the book. In 1986 Paul Zelevansky published the second volume of his by now rare artist book trilogy *The Case for the Burial of Ancestors*. This second volume, *Book Two: Genealogy,* is supposedly the third edition (which is a fiction, since there was only one edition) of a fictional translation of an equally fictional ancient text, which is itself a translation of an oral account of the Hegemonians from the twelfth and thirteenth centuries that is "attributed to a score of mystics, religionists and scholars, none of whom has ever stepped forward."[45] The text focuses particularly on the stories of four priests, each of whom is identified throughout the book with a different typeface, which Zelevansky claims makes it possible "to build a reading of the text around a typographical sequence."[46] Also included in *Book Two* is a sheet of sixteen stamps—each a miniature, layered collage of letters and found objects. As Zelevansky puts it in "Preface to

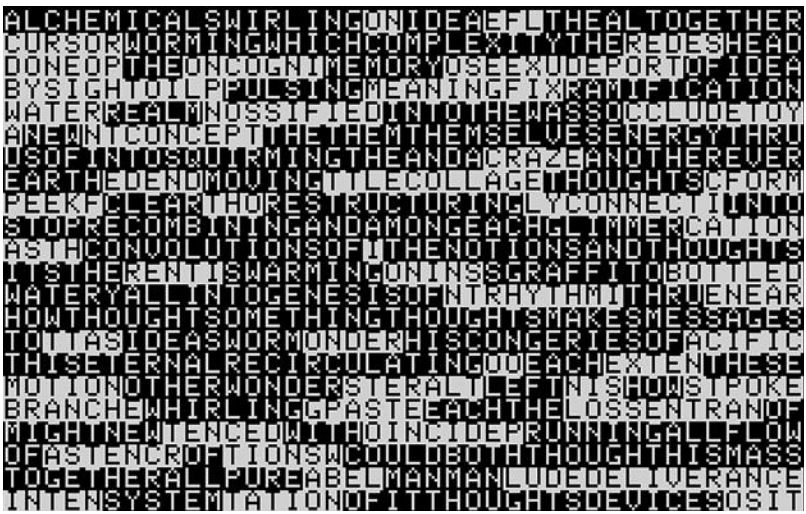

FIGURE 17. *Screenshot from an emulated version of a later sequence in Geof Huth's 1987 Apple BASIC poem "Endemic Battle Collage."*

the Third Edition," "Each stamp has a particular part to play in the narrative. It is left to the Reader to attach them, where indicated, in the spaces provided throughout the text."[47] Finally, enclosed in an envelope on the inside of the back cover, the book comes with "SWALLOWS," a 5.25-inch floppy disk containing a video game that forms the first of the book's three parts. Programmed in Forth-79 for the Apple IIe or II+, the original "SWALLOWS" was supposedly created in 1985 and integrated into the first part of the print version of *Book Two* through a short text and image version.[48] Further, not only are the separate parts of *Book Two* tightly intertwined with each other, but so too are the first and second books of the trilogy, for the imagery and marginalia in the book itself are, we are led to believe, all drawn either from *Book One* or from the "SWALLOWS" disk:

1. Graphic symbols (black on white) based on those found in the second edition of GENEALOGY and in Book I. of THE CASE FOR THE BURIAL OF ANCESTORS.

2. Facsimiles of computer imagery (white on black) drawn from SWALLOWS.
3. Computer printout (black on white) sent ("dumped") from SWALLOWS to a dot-matrix printer.[49]

Not surprisingly, despite Zelevansky's claims that certain facsimiles and computer printouts in the "SWALLOWS" section are taken from the game, the book version includes images and text that do not in fact appear in "SWALLOWS," although it's possible they did exist in the original 1985 version of the game, though any notion of an original in the trilogy should be considered with a degree of skepticism. Appropriately enough, the opening text, "I. HOW IT BEGAN"—the first of eleven parts in the book, which of course correspond only occasionally to the seven parts in the game—is preceded by an image that *does not* appear in the game and begins with an excerpt that *does* appear in the opening scenes of the game:

> SITTING ABOVE THE ACTION, PULLING STRINGS,
> IT WAS THE PUPPETEER'S GAME TO PLAY—
> AND THE PUPPETEER LIKED TO PLAY.
>
> THE KNOWN WORLD WAS IN PLACE.
> THE KNOWN PEOPLE WERE IN MOTION.
> WHAT ELSE WAS NEW?
> THE PUPPETEER DEMANDED SOMETHING MORE.[50]

Thus, in both the book and the game, nearly all of the subjects—whether priest, swallow, or puppeteer—could also be stand-ins or even allegories either for each other or for us as readers/players. At a minimum this passage seems to frame "SWALLOWS" as a game about us playing the game as we sit "ABOVE THE ACTION, PULLING THE STRINGS" by navigating our way through, choosing menu options, and progressing from one level to another. And the "SOMETHING MORE" alluded to at the end of the passage? It is at least partly the unlocatable nature of the text, of

whose meaning, structure, origin, authorship, and even boundaries we can never be certain. As Zelevansky writes in the following lines, which appear only in the book and not in the game:

> AFTER ALL . . .
> WHO WAS THE RULER OF THE TENT?
> WHO WAS THE OWNER OF THE WORK?
> WHO WAS MINDING THE STORE?
> WHO WAS KEEPING THE SCORE?
> SUDDENLY, LIGHTNING STRUCK THE DINNER TABLE,
> THE DIRECTOR BROUGHT FORTH SPECIAL EFFECTS
> AND THE GAME OF SWALLOWS WAS BORN.
> IT WAS ELECTRONIC MYTHOLOGY FOR AN AUDIENCE
> OF ONE—IN HONOR OF THE PUPPETEER.[51]

At this point, most of the text in the "SWALLOWS" section of the book recaps or sometimes replicates certain parts from the story, on the disk, of four swallows (which, again, could stand in for the four priests or could stand in for our experience reading/interacting with the trilogy as a whole) who lose their way home because of "SOME SMOKE." Once the smoke clears, the swallows discover they cannot "FIND THE ORIGINAL" home, and so they go about trying to rebuild another home. Eventually, the swallows find they can no longer avoid a looming existential question: "WHO ARE THEY ANYWAY?" The answer that appears in both the book and the game is as follows:

> IT HAD TO BE FACED, THE SWALLOWS WERE
> EXPENDABLE. THE MONITOR ATE THEM WITH
> GREAT REGULARITY.
> WAS THIS FAIR? DID THE PUPPETEER CARE?
> DOES A MACHINE KNOW IT'S ACTING LIKE A
> MACHINE?[52]

The existential question looming over the swallows now looms over us as players/readers, as well as the machine. If we are the

puppeteer, do we in fact care about what happens on-screen? Is it possible that the machine knows, let alone cares, what happens in the machine and on-screen? And further, Zelevansky urges us "TO CONSIDER THIS":

> THOUSANDS OF ELECTRONIC SIGNALS SETTING
> OFF THOUSANDS OF FLASHING LIGHTS, PRODUCED
> THE VISIBLE EFFECTS OF THIS GAME.
> AS WITH THE SWALLOWS, EACH LIGHT FOLLOWS
> A PATH THROUGH THE GAME; EACH LIGHT HAS
> ITS BEGINNING AND ITS END, ITS HOME AND A
> MULTITUDE OF POSSIBLE DESTINATIONS.[53]

This passage is just one of many examples of how the game self-reflexively talks about itself both as a fictional construct and as nonfiction, insofar as the game is mediated and structured specifically by the computer. Further, once the reader/player interacts with the game "SWALLOWS," some of these "POSSIBLE DESTINATIONS" include nine choices offered via "CAMEL MENUS" that appear in four of the seven parts of the game: "F TO FLY," "B TO BUILD," "W FOR CAMEL WISDOM," "D FOR DIVINE INTERVENTION," "Q FOR OLD QUESTIONS," "ESC TO RETURN TO THE BEGINNING," "& FOR THE NEXT CHAPTER," "P FOR PAST CHAPTER," and "R FOR RANDOM FLIGHT." These choices are not only decidedly unconventional for a video game but also foreshadowed in the game's text such that the narrative seems to be aware of itself rather than the reader/player being the sole owner of an awareness that is usually structurally reinforced in a game because of the separation that exists between the story and the game controls.

Last, if you thought "SWALLOWS" couldn't remix itself any more or recede any more from the present moment as a result of obsolescence, thanks to Matthew Kirschenbaum and the expertise and resources at the Maryland Institute for Technology in the Humanities, in 2012 Zelevansky resurrected "SWALLOWS" by first creating a disk image and then an emulation

of the original. From there, Zelevansky was able to free himself from his slowly ailing Apple computer and go on to create "SWALLOWS 2.0"—a movie we can watch and download. "SWALLOWS 2.0" is, in his words, "a conversation between an Apple IIe, and a Macbook Pro" and is yet another self-conscious, self-referential remix of the "original" version that makes it clear we are watching an emulated, thoroughly mediated version that includes additional audio, video, and even fake sequences from the Apple IIe that masquerade as pieces from the so-called original "SWALLOWS." It is as if "SWALLOWS 2.0" acts out the story of the swallows from the text of the book, who, again, find they cannot "FIND THEIR ORIGINAL" home and need to rebuild another, for without using the 5.25-inch floppy itself along with an Apple IIe or II+, there is no original "SWALLOWS." It's remix all the way down.

In short, then, "SWALLOWS," or even *Book Two* as a whole—more so than *First Screening* and "Endemic Battle Collage"—is a very early literary instance of a work that self-consciously uses its own text, distributed across different media, to comment on these media and on the nature of our interactions with the text as it is mediated by these particular reading/writing technologies—whether book, video game, or stamp. It also thereby works against the grain of each medium to accomplish this level of metacommentary, leading the way quite clearly to later works of activist media poetics that seek to make visible once again the underlying workings of the computer and the digital interface lying at the threshold of the user and the computer.

Closed, Transparent, Task Oriented: The Apple Macintosh

Where are the works of digital literature created for the Apple Macintosh, the successor to the Apple II line of computers? I would say that if they do exist (most likely, a number of early Storyspace works were created on the Macintosh), then they do

so in spite of the Macintosh, a computer clearly designed for consumers, not creators. As seen in the advertisements in Figures 18 and 19, it was marketed as a democratizing machine when in fact it was democratizing only insofar as it marked a profound shift in personal computing away from the sort of inside-out know-how one needed to create on an Apple II to the kind of perfunctory know-how one needed to navigate the surface of the Macintosh—one that amounted to the kind of knowledge needed to click this or that button. The Macintosh was democratic only in the manner any kitchen appliance was democratic.

Along with the way in which terms such as *transparency, customization,* and *user-friendly* were used, altered, and eventually turned inside out en route to the release of the Macintosh, Apple's redefinition of the overall philosophy of personal computing exemplified just one of many reversals that abounded in the ten-year period from the mid-1970s to the mid-1980s. In relation to the crucial change that took place in the mid-1980s from open, flexible, and extensible computing systems for creativity to ones that were closed, transparent, and task oriented, the way in which the Apple Macintosh was framed at the time of its release in January 1984 represented a near-complete purging of the philosophy promoted by Engelbart, Kay, and Papert. This purging of the recent past took place under the guise of Apple's version of the user-friendly, which among other things, pitted itself against the supposedly "cryptic," "arcane" "phosphorescent heap" that was the command-line interface, as well as, it was implied, any earlier incarnation of the GUI.[54]

It is important to note, however, that although the Macintosh philosophy purged much of what had come before it, it did in fact emerge from the momentum gathering in other parts of the computing industry, which in 1982 and 1983 were particularly concerned with defining standards for the computer interface. Up to this point, personal computers were remarkably different from each other. Commodore 64 computers, for example, came with both a Commodore key that gave the user access to an alternate character set and four programmable function

keys that with the Shift key could each be programmed for two different functions. By contrast, Apple II computers came with two programmable function keys, and Apple III, IIc, and IIe computers came with open-Apple and closed-Apple keys that provided the user with applications shortcuts such as cut-and-paste and copy, in the same way that the contemporary Command key functions.

No doubt in response to the difficulties this variability posed to expanding the customer base for personal computers, *Byte* magazine ran a two-part series in October and November 1982 dedicated to the issue of industry standards by way of an introduction to a proposed uniform interface called the Human Applications Standard Computer Interface (HASCI). Asserting the importance of turning the computer into a "consumer product," author Chris Rutkowski declared that every computer ought to have a "standard, easy-to-use format" that "approaches one of *transparency*. The user is able to apply intellect directly to the task; the tool itself seems to disappear."[55] Of course a computer that is easy to use is entirely desirable. At this point, however, ease of use is framed in terms of the disappearance of the tool being used in the name of "transparency"—which then means users can efficiently accomplish their tasks with the help of a glossy surface that shields them from the depths of the computer, instead of the earlier notion of transparency, which refers to a user's ability to open up the hood of the computer in order to directly understand its inner workings.[56] In some ways, then, Rutkowski's proposed HASCI marked "the beginning of an era of consumer-oriented computers," with the emphasis no longer on learning or creativity but rather on, again, a computer that appealed to the widest possible swath of consumers, who wanted to use ready-made hardware and software mostly for the accomplishment of tasks and who most certainly did not want to tinker with expansion slots or programming.[57]

Throughout the following year, *Byte* continued to publish special issues on "easy software" and standards, as well as articles and editorials on the philosophy of the user interface, on

how "windowing is the most natural way to express task concurrency," on the role of metaphor in "man–computer systems," and on various other GUIs or menu-based interfaces that never caught on, such as VisiOn and the Starburst User Interface. Thus, no doubt in a bid to finally produce a computer that realized these ideas and to appeal to consumers who were "drivers, not repairmen," Apple unveiled the Lisa in June 1983 for nearly $10,000 as a cheaper and more user-friendly version of the Xerox Alto/Star, which sold for $16,000 in 1981.[58] At least partly inspired by Larry Tesler's Xerox PARC 1979 demo of the Star to Steve Jobs, the Lisa—designed by Tesler himself, who moved to Apple a year later in 1980—used a one-button mouse, overlapping windows, pop-up menus, a clipboard, and a trash can. As Tesler was adamant to point out in the 1985 article "Legacy of the Lisa," it was "the first product to let you drag [icons] with the mouse, open them by double-clicking, and watch them zoom into overlapping windows."[59] The Lisa moved that much closer to the realization of the dream of transparency with, for example, its mode of double-clicking that attempted to naturalize the Star's text-based commands by no longer making the user actively choose "OPEN" and "CLOSE" and instead having them develop the quick, physical action of double-clicking that bypassed the intellect through physical habit. More, its staggering 2048K worth of software and three expansion slots firmly moved it in the direction of a ready-made, closed consumer product and definitively away from the Apple II, which when it was first released in 1977 came with 16K of code and, again, eight expansion slots.

Expansion slots symbolized the direction that computing was to take from the moment the Lisa was released to the release of the Macintosh in January 1984 to the present day. Jeff Raskin, who originally began the Macintosh project in 1979, and Steve Jobs both believed that hardware expandability was one of the primary obstacles in the way of personal computing's broader consumer appeal.[60] In short, expansion slots made standardization impossible (partly because software writers

needed consistent underlying hardware to produce widely functioning products), whereas what Raskin and Jobs both sought was a system that was an "identical, easy-to-use, low-cost appliance computer." At this point, customization was no longer in the service of building, creating, or learning. It was, instead, for using the computer as one would any home appliance, and ideally this customization would be possible only through software that the user dropped into the computer via disk, just as one would a piece of bread into a toaster. Predictably, the original plan for the Macintosh had it tightly sealed so that the user was only free to use the peripherals on the outside of the machine. Although team member Burrell Smith managed to convince Jobs to allow him to add slots so that users could expand the machine's RAM, according to Steven Levy, Macintosh owners were still "sternly informed that only authorized dealers should attempt to open the case. Those flouting this ban were threatened with a potentially lethal electric shock."[61]

That Apple could successfully gloss over the aggressively closed architecture of the Macintosh while marketing it as a democratic computer "for the people" marked just one more remarkable reversal from this period in the history of computing. As is clear in the advertisement in Figure 18, which came out in *Newsweek* during the 1984 election cycle, the Macintosh computer was routinely touted as embodying the principle of democracy. While it was certainly more affordable than the Lisa (in that it sold for the substantially lower price of $2,495), its closed architecture and lack of flexibility could still easily allow one to claim it represented a decidedly *undemocratic* turn in personal computing.

Thus, 1984 became the year that Apple's philosophy of the computer as appliance, encased in an aesthetically pleasing exterior, flowered into an ideology. We can partly see how their ideology of the user-friendly came to fruition through their marketing campaign, which included a series of magazine ads, along with one of the most well-known TV commercials of the late twentieth century. In the case of the commercial, Apple

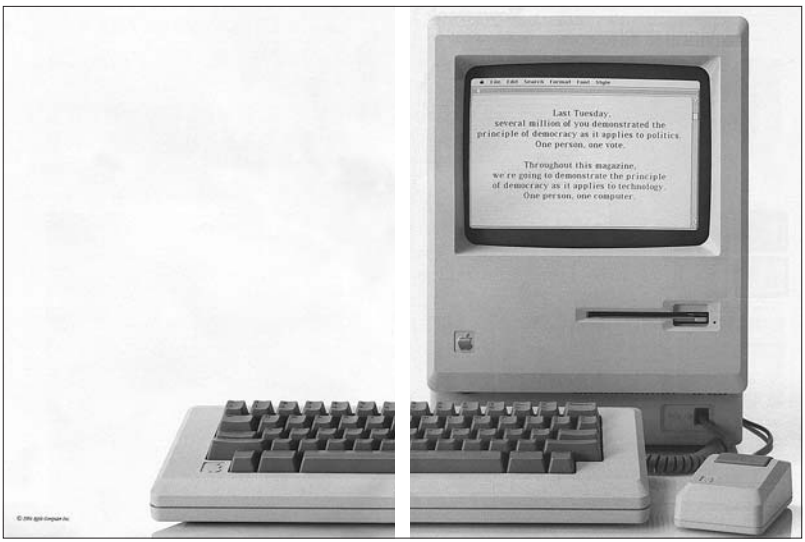

FIGURE 18. *Two-page advertisement for the Apple Macintosh from the November/December 1984 issue of* Newsweek.

took full advantage of the powerful resonance still carried by George Orwell's dystopian post–World War II novel *1984* by reassuring us in the final lines of the commercial, which aired on January 22, 1984, "On January 24th Apple Computer will introduce Macintosh. And you'll see why 1984 won't be like '1984.'"[62] Apple positioned Macintosh, then, as a tool for and of democracy while also pitting the Apple philosophy against a (nonexistent) other (perhaps communist, perhaps IBM or Big Blue) who was attempting to oppress us with an ideology of bland sameness. Apple's ideology "saved" us, then, from a vague and fictional, but no less threatening, Orwellian, nightmarish ideology. As lines of robot-like people, all dressed in identical grey, shapeless clothing, march in the opening scene of the commercial, a narrator of this pre-Macintosh nightmare appears on a screen before them in something that appears to be a propaganda film (see Figure 19).

We hear, spoken fervently, "Today, we celebrate the first

FIGURE 19. *Screenshot of the commercial advertising the release of the Apple Macintosh (directed by Ridley Scott), which aired on January 22, 1984, during the third quarter of Super Bowl XVIII.*

glorious anniversary of the Information Purification Directives." And as Apple's hammer thrower then enters the scene, wearing bright-red shorts and pursued by soldiers, the narrator of the propaganda film continues:

> We have created for the first time in all history a garden of pure ideology, where each worker may bloom, secure from the pests of any contradictory true thoughts. Our Unification of Thoughts is more powerful a weapon than any fleet or army on earth. We are one people, with one will, one resolve, one cause. Our enemies shall talk themselves to death, and we will bury them with their own confusion.[63]

Just before the hammer is thrown at the film screen, causing a bright explosion that stuns the grey-clad viewers, the narra-

tor finally declares, "We shall prevail!" But who exactly is the hammer-thrower-as-underdog fighting against? Who shall prevail—Apple or Big Brother? Who is warring against whom in this scenario and why? In the end all that mattered was that at this moment, just two days before the official release of the Macintosh, Apple had created a powerful narrative of its unquestionable, even natural superiority over other models of computing, a narrative that continues well into the twenty-first century. It was an ideology that of course masked itself as such and that was born out of the creation of and then opposition to a fictional, oppressive ideology from which we users/consumers needed to be saved.[64] In this context the fervor with which Macintosh team members believed in the rightness and goodness of their project is somewhat less surprising. They were quoted in *Esquire* earnestly declaring, "Very few of us were even thirty years old. . . . We all felt as though we had missed the civil rights movement. We had missed Vietnam. What we had was the Macintosh."[65]

We can see how this transformation from philosophy to ideology took place partly through their design bible from 1988, *Apple Human Interface Guidelines: The Apple Desktop Interface,* in which we learn, first, of the importance of an interface that is utterly consistent and familiar and that provides a believable environment via visual metaphors, such as the trash can icon or images of file folders, so that "people can perform their many tasks." We are told, "People are not trying to *use computers*—they're trying to get their jobs done."[66] Of course, use, not the accomplishment of tasks, is what makes creativity and learning on a computer possible. Second, we learn of the importance of an interface that makes commands visible for the user—and "visible" is yet another reversal, for here it means not the ability to see and understand the underlying processes but rather that "the screen displays a representation of the 'world' that the computer creates for the user. On this screen is played out the full range of human–computer interactions."[67] In fact, what we see on-screen is not "the full range" of possible

human–computer interactions but rather a predetermined set of interactions, designed to appear as though it is a full range of interactions, from which the user must choose. If the interface is indeed a threshold between user and computer, then what the Macintosh interface offered was an entirely simulated environment for the user with no access at all to the machine on the other side.[68]

Again, though, the "believable environment" offered by the Macintosh was so appealing, so seductive that it was nearly impossible to see its clear limitations. Even nonfiction accounts of the Macintosh by non-Apple employees could not help but endorse it in as breathless terms as those used by the Macintosh team members themselves. Steven Levy's *Insanely Great,* from 1994, is a document remarkable for an endorsement of this new model of personal computing as wholesale as that of any Macintosh advertisement or guidebook. Recalling his experience seeing a demonstration of a Macintosh in 1983, he writes:

> Until that moment, when one said a computer screen "lit up," some literary license was required. . . . But we were so accustomed to it that we hardly even thought to conceive otherwise. We simply hadn't seen the light. I saw it that day. . . . By the end of the demonstration, I began to understand that these were things a computer *should* do. There was a better way.[69]

The Macintosh was not simply one of several alternatives—it represented the unquestionably right way for computing. Even when he wrote the book in 1993, Levy still declared that each time he turned on his Macintosh, he was reminded "of the first light I saw in Cupertino, 1983": "It is exhilarating, like the first glimpse of green grass when entering a baseball stadium. I have essentially accessed another world, the place where my information lives. It is a world that one enters without thinking of it . . . an ephemeral territory perched on the lip of math and firmament."[70] But it is precisely the legacy of the unthinking,

invisible nature of the so-called user-friendly Macintosh environment that has precluded using computers for creativity and learning and that continues in contemporary multitouch, gestural, and ubiquitous computing devices such as the iPad and the iPhone, whose interfaces are touted as utterly invisible and, therefore, whose inner workings are de facto as inaccessible as they are invisible. That said, once roughly fifteen years had passed since the release of the Macintosh, critiques of this model of frictionless, closed computing began to surface in activist digital media poetics.

This page intentionally left blank