Programmed Visions

Software and Memory

Wendy Hui Kyong Chun

The MIT Press
Cambridge, Massachusetts
London, England

For information about special quantity discounts, please email special_sales@mitpress.mit.edu

This book was set in Stone Sans and Stone Serif by Toppan Best-set Premedia Limited. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data
Chun, Wendy Hui Kyong, 1969–
Programmed visions : software and memory / Wendy Hui Kyong Chun.
p. cm. — (Software studies)
Includes bibliographical references and index.
ISBN 978-0-262-01542-4 (hardcover : alk. paper)
1. Computer software—Development—Social aspects. 2. Software architecture—Social aspects. 3. Computer software—Human factors. I. Title.
QA76.76.D47C565 2011
005.1—dc22

2010036044

10 9 8 7 6 5 4 3 2 1

Contents

Introduction: Software, a Supersensible Sensible Thing

Debates over new media resonate with the parable of the six blind men and the elephant. Each man seizes a portion of the animal and offers a different analogy: the elephant is like a wall, a spear, a snake, a tree, a palm, a rope. Refusing to back down from their positions since they are based on personal experience, the wise men engage in an unending dispute with each "in his own opinion / Exceeding stiff and strong / Though each was partly in the right, / And all were in the wrong!" The moral, according to John Godfrey Saxe's version of this tale, is: "So oft in theologic wars, / The disputants, I ween, / Rail on in utter ignorance / Of what each other mean, / And prate about an Elephant / Not one of them has seen!"[1] It is perhaps irreverent to compare a poem on the incomprehensibility of the divine to arguments over new media, but the invisibility, ubiquity, and alleged power of new media (and technology more generally) lend themselves to this analogy. It seems impossible to know the extent, content, and effects of new media. Who can touch the entire contents of the World Wide Web or know the real size of the Internet or of mobile networks? Who can read and examine all time-based online interactions? Who can expertly move from analyzing social networking sites to Japanese cell phone novels to hardware algorithms to databases? Is a global picture of new media possible?

In response to these difficulties, many within the field of new media studies have moved away from specific content and technologies toward what seems to be common to all new media objects and moments: software. All new media objects allegedly rely on—or, most strongly, can be reduced to—software, a visibly invisible or invisibly visible essence. Software seems to allow one to grasp the entire elephant because it is the invisible whole that generates the sensuous parts. Based on and yet exceeding our sense of touch—based on our ability to manipulate virtual objects we cannot entirely see—it is a magical source that promises to bring together the fractured field of new media studies and to encapsulate the difference this field makes. To know software has become a form of enlightenment—a Kantian release from self-incurred tutelage.

This notion of knowing software as a form of enlightenment—as a way to comprehend an invisible yet powerful whole—is not limited to the field of new media

studies. Based on metaphor, software has become a metaphor for the mind, for culture, for ideology, for biology, and for the economy. Cognitive science, as Paul Edwards has shown, initially comprehended the brain/mind in terms of hardware/software.[2] Molecular biology conceives of DNA as a series of genetic "programs." More broadly, culture itself has been posited as "software," in opposition to nature, which is "hardware."[3] Although technologies, such as clocks and steam engines, have historically been used metaphorically to conceptualize our bodies and culture, software is unique in its status as metaphor for metaphor itself. As a universal imitator/machine, it encapsulates a logic of general substitutability: a logic of ordering and creative, animating disordering. Joseph Weizenbaum has argued that computers have become metaphors for all "effective procedures," that is, for anything that can be solved in a prescribed number of steps, such as gene expression and clerical work.[4]

The clarity offered by software as metaphor—and the empowerment allegedly offered to us who know software—however, should make us pause, because software also engenders a sense of profound ignorance. Software is extremely difficult to comprehend. Who really knows what lurks behind our smiling interfaces, behind the objects we click and manipulate? Who completely understands what one's computer is actually doing at any given moment? Software as metaphor for metaphor troubles the usual functioning of metaphor, that is, the clarification of an unknown concept through a known one. For, if software illuminates an unknown, it does so through an unknowable (software). This paradox—this drive to understand what we don't know through what we do not entirely understand—this book argues, does not undermine, but rather grounds software's appeal. Its combination of what can be seen and not seen, can be known and not known—its separation of interface from algorithm, of software from hardware—makes it a powerful metaphor for everything we believe is invisible yet generates visible effects, from genetics to the invisible hand of the market, from ideology to culture.

Every use entails an act of faith, and this book tries to understand what makes this trust possible not in order to condemn and move "beyond" computer software and interfaces, but rather to understand how this combination of visibility and invisibility, of past experiences with future expectation, makes new media such a powerful thing for each and all. It also takes seriously new media's modes of repetition and transmission in order to understand how they open up gaps for a future beyond predictions based on the past. Computers—understood as software and hardware machines—this book argues, are mediums of power. This is not only because they create empowered users, but also and most importantly, because software's vapory materialization and its ghostly interfaces embody—conceptually, metaphorically, virtually—a way to navigate our increasingly complex world.

How Soft Is Software?

Software is, or should be, a notoriously difficult concept. Historically unforeseen, barely a thing, software's ghostly presence produces and defies apprehension, allowing us to grasp the world through its ungraspable mediation.

Computer scientist Manfred Broy describes software as "almost intangible, generally invisible, complex, vast and difficult to comprehend." Because software is "complex, error-prone and difficult to visualize," Broy argues, many of its "pioneers" have sought to make "software easier to visualize and understand, and to represent the phenomena encountered in software development in models that make the often implicit and intangible software engineering tasks explicit."[5] Software challenges our understanding not only because it works invisibly, but also because it is fundamentally ephemeral—it cannot be reduced to program data stored on a hard disk. Historian Michael Mahoney describes software as "elusively intangible. In essence, it is the behavior of the machines when running. It is what converts their architecture to action, and it is constructed with action in mind; the programmer aims to make something happen."[6] Consequently, software is notoriously difficult to study historically: most "archived" software programs can no longer be executed, and thus experienced, since the operating systems and machines, with which they merge when running, have disappeared. Although these systems can be emulated, what is experienced is a reconstruction.[7] Hence, not only does software's ephemerality make analysis difficult, so does the lack of clear boundaries between running programs and between running software and live hardware. Theorist Adrian MacKenzie aptly calls software a "neighbourhood of relations"; "in code and coding," he argues, "relations are assembled, dismantled, bundled and dispersed within and across contexts."[8] Software "pioneers" Herman H. Goldstine and John von Neumann, in their 1940s explication of programming, similarly described it as "the technique of providing a dynamic background to control the automatic evolution of a meaning."[9]

To be apprehended, software's dynamic porousness is often conceptually transformed into well-defined layers. Software's temporality, in other words, is converted in part to spatiality, process in time conceived in terms of a process in space. Historian Paul Ceruzzi likens software to an onion, "with many distinct layers of software over a hardware core."[10] Application on top of operating system, on top of device drivers, and so on all the way down to voltage charges in transistors. What, however, is the difference between an onion's layers and its core? Media archeologist Friedrich Kittler, taking this embedded and embedding logic to its limit, has infamously declared "there is no software," for everything, in the end, reduces to voltage differences. More precisely, he contends, "there would be no software if computer systems were not surrounded . . . by an environment of everyday languages. This environment . . . since

a famous and twofold Greek invention, consists of letters and coins, of books and bucks."[11] Less controversially, Mahoney has argued that software "is an artifact of computing in the business and government sectors during the '50s"; software, as Paul Ceruzzi and Wolfgang Hagen have shown, was not foreseen: the engineers building high-speed calculators in the mid-1940s did not plan or see the need for software.[12]

At first, software encompassed everything that was not hardware, such as services. The term *soft*, as this book elaborates, is gendered. Grace Murray Hopper claims that the term *software* was introduced to describe compilers, which she initially called "layettes" for computers; J. Chuan Chu, one of the hardware engineers for the ENIAC, the first working electronic digital computer, called software the "daughter" of Frankenstein (hardware being the son).[13] Software, as a service, was initially priced in terms of labor cost per instruction.[14] Herbert D. Benington remarks that attendees at the 1956 symposium on advanced programming methods for digital computers were horrified that his Lincoln Laboratory group, working on what would become the groundbreaking SAGE (Semi-Automatic Ground Environment) Air Defense System, could do no better than $50 per instruction. In that 1956 address Benington also stresses the growing importance of software: "our colleagues who build computers," he notes, "have come to realize that a computer is not useful until it has been programmed."[15] As this statement reveals, the word *program*, at that time, was predominantly a verb, not a noun.[16]

Legal battles over software copyrights and patents make clear the stakes of this transformation of software from a service, priced per instruction, to a thing. Not surprisingly, software initially was considered neither patentable nor copyrightable because of its functional, intangible, and "natural" status. The U.S. Supreme Court in 1972 first rejected engineers Gary Benson and Arthur Tabbot's claim to patent an algorithm for converting digital into binary digits. It decided, as legal scholar Pamela Samuelson argues, that "mathematical innovations should be treated like scientific truths and laws of nature, and scientific truths and laws of nature are unpatentable subject matter."[17] Software algorithms, in other words, were "natural" mental processes, not artificial things. As Samuelson and as legal scholar Margaret Jane Radin both note, key to the eventual patenting of software was its transformation from a set of instructions to a machine.[18] In 1981, the Supreme Court in *Diamond v. Diehr, 450 U.S. 175* (1981) upheld the patenting of an algorithmic-based process for curing rubber because the algorithm resulted in a tangible physical process: it cured rubber. By 1994, the U.S. Court of Appeals Federal Circuit held in *In re Alappat* (1994) that all software was inherently machinic, since it changed the material nature of a computer: "a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software."[19] A change in memory, it seems, a change in machine.

As a physical process, however, software would seem uncopyrightable.[20] Copyright seeks to protect creative expression; as Radin notes, patents and copyrights were supposed to be mutually exclusive: "Copyright is supposed to exclude works that are functional; patent is supposed to focus on functionality and exclude texts."[21] To address this contradiction, the U.S. Congress changed the law in 1975, so that expressions, as opposed to the actual processes or methods, adopted by the programmer became copyrightable.[22] The difference, however, between expression and methods has been difficult to determine, especially since the expression of software has not been limited to source code.

Further, copyright law insists on the tangibility of the copy, where a copy is a "fixation in a tangible medium of expression"—performances, in other words, were initially considered outside the purview of copyright.[23] Although information is often considered to be immaterial, the forces behind copyrighting (and taxing) software stress the fact that, regardless of information's ephemerality, information is always embodied; it always, as Matthew Kirschenbaum argues, leaves a trace.[24] Indeed, digital information has divorced tangibility from permanence, with "courts and commentators in the United States adopt[ing] the notion that the momentary arrangement of electrons in a computer memory, which we might have thought of as intangible information, amounts to a tangible physical object, a copy."[25] Since, as I have argued elsewhere, computer reading is a writing elsewhere, viewing the momentary arrangement of electrons in memory as a tangible copy technically makes all computer reading a copyright infringement. Indeed, this redefinition of copy as thing, as Radin notes, has had far-reaching consequences since "a great many activities that were not covered by copyright in the offline environment are being brought under copyright—that is, under control of an owner—in the online environment. . . . The physical analogy to browsing in a bookstore is obliterated by the more powerful assimilation of the activity involved in a physical object—the production of physical 'copies' by a computer."[26] This definition also muddies questions of responsibility: given that every networked computer regularly downloads all materials in a network and then erases those not directly addressed to it, should everyone whose computer has unwittingly downloaded child pornography or pirated media be prosecuted?

These changes, brought about by the "hardening" of software as textual or machinic thing through memory, point toward a profound change in our understanding of what is internal and external, subject and object. According to Radin, "the distinction between tangible objects and intangible information is a distinction upon which much of our modern understanding of the world was built, and hence, from which a great many legal categorizations derive," for this traditional distinction "owes much to the 'modernist' dichotomies of the Enlightenment—between subject and object, between autonomous persons and heteronomous things."[27] The notion of intellectual property, which seems to break this dichotomy, was initially a compromise, she

contends, between the Enlightenment notion that the intellect was internal and property external.[28] (It is not simply, though, that information was once inside a person and then externalized, but also that information was considered inseparable from a person. Symptomatically, the meaning of information has moved from "the action of informing . . . the formation or moulding of the mind or character, training, instruction, teaching" to "knowledge communicated concerning some particular fact, subject, or event."[29]) Crucially, Radin argues that the information age has compromised the compromise of intellectual property, since, by breaking down the distinction between tangibility and intangibility, it makes it possible to conceive information, whether internal or external, as always external to the self (hence the patentability of DNA). As I've argued elsewhere, the Internet and computers—which have offered enlightenment for all—have exploded enlightenment by literalizing it.

Software as thing has led to all "information" as thing. Software as thing reconceptualizes society, bodies, and memories in ways that both compromise and extend the subject, the user. Importantly, software as thing cannot be reduced to software as a commodity: software as "thing" is a return to older definitions of thing as a "gathering," as pertaining to anything related to "man."[30] Treating software as a thing means treating it, again, as a neighborhood, as an amalgamation. It also means thinking through its simultaneous ambiguity and specificity. Further, it means thinking beyond this legal history, this legal framework, toward the historical and theoretical stakes of the reemergence of things as relations. Indeed, this book argues that the remarkable process by which software was transformed from a service in time to a product, the hardening of relations into a thing, the externalization of information from the self, coincides with and embodies larger changes within what Michel Foucault has called *governmentality*. Software as thing is a response to and product of changing relations between subjects and objects, of challenges brought about by computing as a neoliberal governmental technology.

Soft Government

According to Foucault, governmentality and government broadly encompass acts and institutions that govern, or steer, conduct and thus cannot be reduced to the state. (Not coincidentally, the term *cybernetics* is derived from the Greek term "kybernete" for governing.) As Colin Gordon notes, government for Foucault is "the conduct of conduct," that is, "a form of activity aiming to shape, guide or affect the conduct of some person or persons." Governmentality could concern "the relation between self and self, private interpersonal relations involving some form of control or guidance, relations within social institutions and communities and, finally, relations concerned with the exercise of political sovereignty."[31] The move from the Enlightenment, with its dichotomy of subjects and objects, to our current compromised

situation corresponds to a transition from liberal to neoliberal governmentality (and, even further, to a neoconservative one).

Liberal governmentality, which emerged during the eighteenth century, is an "economic government": government that embraces both liberal political economy and the principle of noninterference. It is based on two principles: the principle of blind self-interest and the principle of freedom. According to its vision, actors, who cannot know the whole picture, blindly and freely follow their own self-interests so that "the invisible hand of the market" can magically incorporate their actions into a system that benefits all. This unknowability is fundamental, for it enables a transition from sovereign to liberal forms of governmentality. The liberal market undermines the power of the monarch by undermining his or her knowledge: no one can have a totalizing view. It also consumes freedom: it both produces freedom and seeks to control it.[32] Liberal governmentality also makes possible biopolitical power—a collection of institutions and actions focused on "taking care" of a population, rather than a territory, focused on masses rather than on sovereign subjects.

Historically, computers, human and mechanical, have been central to the management and creation of populations, political economy, and apparatuses of security.[33] Without them, there could be no statistical analysis of populations: from the processing of censuses to bioinformatics, from surveys that drive consumer desire to social security databases. Without them, there would be no government, no corporations, no schools, no global marketplace—or, at the very least, they would be difficult to operate. Tellingly, the beginnings of IBM as a corporation—the Herman Hollerith's *Tabulating Machine Company*—dovetails with the mechanical analysis of the U.S. census.[34] Before the adoption of these machines in 1890, the U.S. government had been struggling to analyze the data produced by the decennial census (the 1880 census taking seven years to process). Crucially, Hollerith's punch-card-based mechanical analysis was inspired by the "punch photograph" used by train conductors to verify passengers.[35] Similarly, the Jacquard Loom, a machine central to the industrial revolution, inspired (via Charles Babbage's "engines") the cards used by the MARK1, an early electromechanical machine. Scientific and industrial projects linked to governmentality also drove the need for data analysis: eugenics projects that demanded vast statistical analyses, nuclear weapons that depended on solving difficult partial differential equations.[36]

Importantly, though, computers in the period this book focuses on (post-World War II) coincide with the emergence of neoliberalism. As well as control of "masses," computers have been central to processes of individualization or personalization. Neoliberalism, according to David Harvey is "a theory of political economic practices that proposes that human well-being can best be advanced by liberating individual entrepreneurial freedoms and skills within an institutional framework characterized by strong private property rights, free markets, free trade."[37] Although neoliberals,

such as the Chicago School economist Milton Friedman, claim merely to be resuscitating classical liberal economic theory, Foucault argues that neoliberalism differs from liberalism in its stance that the market be "the principle, form, and model for a state."[38] Tying together individual economic and political freedom has been key to its success: competitive capitalism, Friedman writes, "is a system of economic freedom and a necessary condition for political freedom."[39] Harvey argues that neoliberalism has thrived by creating general "culture of consent"—even though it has harmed most people economically by fostering incredible income disparities. In particular, it has incorporated progressive 1960s discontent with government and, remarkably, dissociated this discontent from its critique of capitalism and corporations.

In a neoliberal society, the market has become an ethics—it has spread everywhere so that all human interactions, from motherhood to education, are discussed as economic "transactions" that can be assessed in individual cost–benefit terms. The market, as Margaret Thatcher argued, "change[s] the soul"[40] by becoming, Foucault argues, the "grid of intelligibility" for everything.[41] This transforms the *homo oeconomicus*—the individual who lies at the base of neoliberalism—from "the [liberal] man of exchange or man the consumer" to "the man of enterprise and consumption."[42] It rests on the "proposition that both parties to an economic transaction benefit from it, *provided the transaction is bi-laterally voluntary and informed*."[43] It focuses on discourses of empowerment in which the worker does not simply own his/her labor, but also possesses his/her own body as a form of "human capital."[44] Since everyone is in control of this form of capital—the body—neoliberalism relies on voluntary, individual actions.[45] Thus, this changed man who has imbibed the market ethic is thus "eminently governable, for *homo oeconomicus* is shaped through "rational" and empowering management techniques that make him "self-organized" and "self controlling."[46]

Relatedly, "user-friendly" computer interfaces have been key to empowering and creating "productive individuals." As Ben Shneiderman, whose work has been key to graphical user interfaces (GUIs) has argued, these interfaces succeed when they move their users from grudging acceptance to feelings of mastery and eagerness.[47] Moreover, this book argues, interfaces—as mediators between the visible and the invisible, as a means of navigation—have been key to creating "informed" individuals who can overcome the chaos of global capitalism by mapping their relation to the totality of the global capitalist system. (Conversely, the ability to track both individuals and totalities at the same time, through the data traces produced through our mappings.) The dream is: the resurgence of the *seemingly* sovereign individual—the subject driven to know, driven to map, to zoom in and out, to manipulate, and to act. The dream is: the more that an individual knows, the better decisions he or she can make. Goldman Sachs and other investment companies, for instance, invest millions of dollars on computer programs that can analyze data and execute trades milliseconds faster than their competition. This "informing" is thus intriguingly temporal. New media empowers individuals by informing them of the future, making new media the

future. "The future," as William Gibson famously and symptomatically quipped, "is already here. It's just not very evenly distributed."[48] This future—as something that can be bought and sold—is linked intimately to the past, to computers as capable of being the future because, based on past data, they shape and predict it.[49] Computers as future depend on computers as memory machines, on digital data as archives that are always there. This future depends on programmable visions that extrapolate the future—or, more precisely, a future—based on the past. As chapter 1 elaborates, computers, understood as software and hardware machines, have made possible a dream of programmability—a return to a world of Laplaceian determinism in which an all-knowing intelligence can comprehend the future by apprehending the past and present. They have done so through a conflation of words with things that both externalizes knowledge and creates a position from which a subject can try to "hack" the invisible hands and laws that drive the system.

This book, therefore, links computers to governmentality neither at the level of content nor in terms of the many governmental projects that have been enabled by computers, but rather at the level of their architecture and their logic.[50] Computers embody a certain logic of governing or steering through the increasingly complex world around us. By individuating us and also integrating us into a totality, their interfaces offer us a form of mapping, of storing files central to our seemingly sovereign—empowered—subjectivity. By interacting with these interfaces, we are also mapped: data-driven machine learning algorithms process our collective data traces in order to discover underlying patterns (this process reveals that our computers are now more profound programmers than their human counterparts). This logic of programmability, it also argues, is not limited to computer technology; it also stems from and bleeds elsewhere, in particular modern genetics, with its conceptual formations of codes and programs as central to inheritance. Crucially, though, this knowledge is also based on a profound ignorance or ambiguity: our computers execute in unforeseen ways, the future opens to the unexpected. Because of this, any programmed vision will always be inadequate, will always give way to another future. The rest of this book unpacks this temporality and the odd combination of visibility and invisibility these visions enable.

In part I, chapters 1 and 2 focus on how software is invisibly visible. Chapter 1 argues that software emerged as a thing—as an iterable textual program—through an axiomatic process of commercialization and commodification that has made code *logos*: a word conflated with and substituting for *action*. This formulation of instruction as source—source code as fetish—is crucial to understanding the power and thrill of programming, in particular the fantasy of the all-powerful programmer, a subject with magical powers to transform words into things. This separation of code from execution, however, itself a software effect, is also constantly undone, historically and theoretically. Thus, it concludes by analyzing how code as fetish can open up surprising detours and ends.

Chapter 2 analyzes how this invisibly visible (or visibly invisible) logic works at the level of the interface, at the level of "personal computing." It investigates the extent to which this paradoxical combination of rational causality and profound ignorance grounds the computer as an attractive model for the "natural" world. Looking both at the use of metaphor within the early history of human–computer interfaces and at the emergence of the computer as metaphor, it contends that real-time computer interfaces are a powerful response to, and not simply an enabler or consequence of, postmodernism and neoliberalism. Both conceptually and thematically, these interfaces offer a simpler, more reassuring analog of power, one in which the user takes the place of the sovereign "source," code becomes law, and mapping produces the subject.

Chapters 3 and 4 of part II examine the intertwining of computer technology and biology, specifically the emergence of memory and its importance to notions of programmability. Through this focus on the relation between biology and computing technology, part II explores how software, as an axiomatic, came to embody the logic of the "always already there." By exploring the ways in which biology and computer technology have been reduced to complementary strands of a double helix, chapters 3 and 4 embed computer technology within the larger epistemic field of programmability, a larger drive for "permanence" that conflates memory with storage and conflates the ephemeral with the enduring, or rather turns the ephemeral into the enduring (the enduring ephemeral) through a process of constant regeneration.

Chapter 3 argues that software was not foreseen, because the drive for software—for an independent program that conflates legislation with execution—did not arise solely from within the field of computation, but also from early Mendelian genetic and eugenics. Through a reading of Erwin Schrödinger's *What Is Life*, it contends that Mendelian genetics and software envision a return to a reductionist, mechanistic understanding of life, in which the human body becomes an archive. This chapter thus complicates the standard narrative within the history of science that the notion of a program was adapted by biologists from computer science, a narrative that rather remarkably treats software as through it always already existed. It also shows how computers, not just in terms of content but also of form, are deeply intertwined with questions of biopower.

The final chapter takes up this intertwining of biology and computer technology, specifically in terms of memory and transmission. Revising the running hypothesis of the first three chapters, chapter 4 shows how digital hardware, which grounds software, is itself axiomatic. Through the reading of early work on neural nets and of John von Neumann's work on automata, it reveals how logical hardware reduces events to words. Analyzing the importance of the analog to conceptualizing the digital, it argues that the digital emerged as a clean, precise logic through an analogy to an analogy. Crucially, it argues that computer memory, as a constantly regenerating and degenerating archive, does not simply erase human agency, but rather makes possible new dreams of human intervention and responsibility.

As this synopsis hopefully makes clear, understanding software as a thing does not mean denigrating software or dismissing it as an ideological construction that covers over the "truth" of hardware. It means engaging its odd materializations and visualizations closely and refusing to reduce software to codes and algorithms—readily readable objects—by grappling with its simultaneous ambiguity and specificity. As Bill Brown has influentially argued, things designate "the concrete yet ambiguous within the everyday," that is, the thing "functions to overcome the loss of other words or as a place holder for some future specifying operation. . . . It designates an amorphous characteristic or a frankly irresolvable enigma. . . . *Things* is a word that tends, especially at its most banal, to index a certain limit or liminality, to hover over the threshold between the nameable and unnameable, the figureable and unfigureable, the identifiable and unidentifiable."[51] Things thus "lie both at hand and somewhere outside the theoretical field, beyond a certain limit, as a recognizable yet illegible remainder or as the entifiable that is unspecifiable."[52] Because things simultaneously name the object and something else, they are both reducible to and irreducible to objects.[53] Whereas we "look *through* objects (to see what they disclose about history, society, nature, or culture—above all, what they disclose about *us*)," we "only catch a glimpse of things."[54] We encounter, but do not entirely comprehend, things. According to Brown:[55]

> A *thing* . . . can hardly function as a window. We begin to confront the thingness of objects when they stop working for us: when the drill breaks, when the car stalls, when the windows get filthy, when their flow within the circuits of production and distribution, consumption and exhibition, has been arrested, however momentarily. The story of objects asserting themselves as things, then, is the story of a changed relation to the human subject and thus the story of how the thing really names less an object than a particular subject-object relation.[56]

Crucially, this effort to rethink, and indeed theorize things, is intimately intertwined with media: Martin Heidegger begins "The Thing" by outlining the shrinking of time and space due to "instant information" (television being the peak of this abolition of every possibility of remoteness); Brown argues, "if the topic of things attained a new urgency in the closing decades of that [twentieth] century, this may have been a response to the digitization of our world—just as, perhaps, the urgency in the 1920s was a response to film."[57]

This book sees this renewed interest in things, things which always seem to be disappearing, not simply as an effect of new media on other "things," but rather as central to the temporality of new media itself. *New media, like the computer technology on which it relies, races simultaneously toward the future and the past, toward the bleeding edge of obsolescence*. Software as thing is inseparable from the externalization of memory, from the dream and nightmare of an all-encompassing archive that constantly regenerates and degenerates, that beckons us forward and disappears before our very eyes.