Interface, Access, Loss

Sean Dockray

Digital file sharing signaled the conclusion of that brief period in human history during which certain forms of culture were mass-produced and sold as commodity objects (records, books, etc.) to consumers.

Property has, in one sense, been undone.

On a massive scale, people have used *their* computers and *their* Internet connections to share digitized versions of *their* objects with each other, quickly producing a different, common form of ownership. The crisis that this provoked is well known. What is less recognized—because it is still very much in process—is the subsequent undoing of property, of both the individual and common kind. What follows is a story of "the cloud," the post-dot-com bubble techno super-entity, which sucks up property, labor, and free time.

OBJECT, INTERFACE

Amidst the development of "gas-works, telegraphy, photography, steam navigation, and railways," Karl Marx described how the progressive mechanization and automation of industry resulted in the irreversible expansion of an ultimately redundant "industrial reserve army."[1] It is difficult not to read his theory—and these technologies of connection and communication—against the background of our present moment, in which the rise of the Internet has been accompanied by the deindustrialization of cities, increased migrant and mobile labor, and jobs made obsolete by computation.

There are obvious examples of the impact of computation on the workplace: at factories and distribution centers, robots engineered with computer-vision can replace handfuls of workers with a saving of millions of dollars per robot over the life of the system. And there are less apparent examples as well, in which algorithms determine when and where to hire people and for how long, according to fluctuating conditions.

Both of these examples have parallels within computer programming, namely "reuse" and "garbage collection." Code reuse refers to the practice of writing software in such a way that the code can be used again later in another program to perform the same task. It is considered wasteful to give the same time, attention, and energy to the function, as the development environment is not an assembly line. Such repetition gives way therefore to copy-and-

pasting (or merely "calling"). When a program is in the midst of being executed, the computer's memory fills with data, some of which is obsolete (and no longer needed for the computer to run efficiently). If left alone, the memory would become clogged and the program would crash. It is the role of the garbage collector to "free up" memory, deleting what is no longer in use.

In Object-Oriented Programming (OOP), a programmer designs the software that he or she is writing around "objects," where each object is conceptually divided into "public" and "private" parts. The public parts are accessible to other objects, but the private ones are hidden to the world outside the boundaries of that object. This is one instance of a "black box"—a thing that can be known through its inputs and outputs, even in total ignorance of its internal mechanisms. What difference does it make if the code is written in one way or another if it behaves the same? As the philosopher William James argues, "If no practical difference whatever can be traced, then the alternatives mean practically the same thing, and all dispute is idle."[2]

By merely having a public interface an object is already a kind of social entity. It makes no sense for an object to provide access to its outside if there are no other potential objects with which to interact. So, to understand the object-oriented program, we must scale up—not by increasing the size or complexity of the object, but instead by increasing the number and types of objects such that their relations become denser. The result is an intricate machine with an on and an off state, rather than a beginning and an end. Its parts are interchangeable, provided that they reliably produce the same behavior—the same inputs and outputs. Furthermore, this machine can be modified: objects can be added and removed, changing but not destroying the machine; and it might be, using Gerald Raunig's appropriate term, "concatenated" with other machines.[3]

Inevitably, this paradigm for describing the relationship between software objects spreads outward, subsuming more of the universe outside of the immediate code. External programs, powerful computers, banking institutions, people, and satellites have all been "encapsulated" and "abstracted" into objects with inputs and outputs. Is this a conceptual reduction of the richness and complexity of reality? Yes, but only partially. It is also a real description of how people, institutions, software, and things are being brought into relationship with

one another according to the demands of networked computation (not to mention the often contradictory demands of business, government, or collective desire); and the expanding field of objects encompasses exactly those entities integrated into such a network.

Consider a simple example of decentralized file sharing: its diagram might represent an object-oriented piece of software, but here each object is a person-computer, shown in potential relation to every other person-computer. Files might be sent or received at any point in this machine, which seems particularly oriented toward circulation and movement. Much remains private, but a collection of files from every person is made public and opened up to the network. Taken as a whole, the entire collection of all files, which on the one hand exceeds the storage capacity of any one person's technical hardware, is on the other hand entirely available to every person-computer. If the files were books, then this collective collection would be a public library.

In order for a system like this to work, for the inputs and the outputs to actually engage with one another to produce action or transmit data, there needs to be something in place to enable meaningful couplings. Before there is any interaction or any relationship, there must be some common ground in place that allows heterogeneous objects to "talk to each other" (to use a phrase from the business-casual language of the Californian ideology). The term used for such a common ground—especially on the Internet—is "platform," or that which enables and anticipates future action without directly producing it. A platform provides tools and resources to the objects that run "on top" of the platform so that those objects do not need to have their own tools and resources. In this sense, the platform offers itself as a way for to externalize (and reuse) labor. Communication between objects is one of the most significant actions that a platform can provide, but it requires that the objects conform some amount of their inputs and outputs to the specifications dictated by the platform.

But haven't we only introduced another coupling, this time between the object and the platform, rather than describing how that coupling works in the first place? To work toward a description, we need to look at that meeting point between things, otherwise known as the "interface." In the terms of OOP, the interface is an abstraction

that defines what kinds of interactions are possible with an object. It maps out the public face of the object in a way that is legible and accessible to other objects. Similarly, computer interfaces like screens and keyboards are designed to meet with human interfaces like fingers and eyes, allowing for a specific form of interaction between person and machine. Any coupling between objects passes through some interface and every interface obscures as much as it reveals: it establishes the boundary between what is public and what is private, what is visible and what is not. The dominant aesthetic values of user interface design actually privilege such concealment as "good design," appealing to principles of simplicity, cleanliness, and clarity.

CLOUD, ACCESS

One practical outcome of this has been that there can be tectonic shifts behind the interface—where entire systems are restructured or revolutionized—without any interruption (so long as the interface itself remains essentially unchanged). In pragmatism's terms, a successful interface keeps any difference (in the back end) from making a difference (in the front end). To use books again as an example: after consumers became accustomed to the initial discomfort of purchasing a product online instead of from a shop, they saw an act such as "buying a book" to be something that could be interchangeably accomplished either by a traditional bookstore or the online "marketplace" equivalent. In each case, one gives money and receives a book. But behind that interface—most likely Amazon—the online bookseller has positioned itself through low prices and a wide selection as the most visible platform for buying books, and uses that position to push retailers and publishers to, at best, the bare minimum of profitability.

In addition to collecting data about its users (what they look at, what they buy) to personalize product recommendations, Amazon has also made an effort to be a platform for the technical and logistical parts of *other retailers*. Ultimately collecting data from them as well, Amazon realizes a competitive advantage from having a comprehensive, up-to-the-minute perspective on market trends and inventories. This volume of data is so vast and valuable that warehouses packed with computers are constructed to store it, protect it, and make it readily available to algorithms. Data centers such as these organize how commodities circulate (they run business applications, store data about retail, manage fulfillment) but also increasingly hold the commodity itself—for example, the book. Sales of digital books started the millennium very slowly but by 2010 had overtaken hardcover sales.

Amazon's store of digital books (or Apple's or Google's, for that matter) is a distorted reflection of the collection circulating within the file-sharing network, displaced from personal computers to corporate data centers. Here are two regimes of digital property: the swarm and the cloud. For *swarms* (a reference to swarm downloading where a single file can be downloaded in parallel from multiple sources), property is held in common between peers—property is positioned out of reach; but on *the cloud*, the same file might be accessible through an interface that has absorbed legal and business requirements. It is only half of the story, however, to associate the cloud with mammoth data centers; the other half is to be found in our hands and laps. Thin computing, including tablets and e-readers, iPads, Kindles, and mobile phones, has coevolved with data centers, offering powerful, lightweight computing precisely because so much processing and storage has been externalized.

In this technical configuration of the cloud, the thin computer and the fat data center meet through an interface, inevitably clean and simple, that manages access to the remote resources. Typically a person needs to agree to certain "terms of service," have a unique, measurable account, and provide payment information; in return, access is granted. This access is not ownership in the conventional sense of a book, or even the digital sense of a file, but rather a license that gives the person a "non-exclusive right to keep a permanent copy... solely for your personal and non-commercial use," contradicting the First Sale Doctrine, which gives the "owner" the right to sell, lease, or rent their copy to anyone they choose at any price they choose. The doctrine, established within America's legal system in 1908, separated the rights of reproduction from distribution as a way to "exhaust" the copyright holder's control over the commodities that people purchased, legitimizing institutions like used bookstores and public libraries. Computer software famously attempted to bypass the First Sale Doctrine with its "shrink-wrap" licenses that restricted the rights of the buyer once he or she broke through the plastic

packaging to open the product. This practice has only evolved and become ubiquitous over the last three decades as software began being distributed digitally through networks rather than as physical objects in stores. Such contradictions are symptoms of the shift in property regimes, or what Jeremy Rifkin called "the age of access." He writes: "Property continues to exist but is far less likely to be exchanged in markets. Instead, suppliers hold on to property in the new economy and lease, rent, or charge an admission fee, subscription, or membership dues for its short-term use."[4]

Thinking again of books, Rifkin provides the image of a paid library emerging as the synthesis of the public library and the marketplace for commodity exchange. Considering how, on the one side, traditional public libraries are having their collections de-accessioned, hours of operation cut, and are in some cases being closed down entirely, and on the other side, the traditional publishing industry finds its stores, books, and profits dematerialized, the image is perhaps appropriate. In photographs inside data centers, server racks strike an eerie resemblance to library stacks, while e-readers are consciously designed to look and feel something like a book. Whether it is in recognition of the centuries of design knowledge accrued in the form of the book, or simply to make the interface as consistent as possible while everything else changes behind the scenes, the e-reader's evocation of the book is undeniable. Yet, when one peers down into the screen of the device, one sees both the book *and* the library.

Like a Facebook account, which must uniquely correspond to a real person, the e-reader is an individualizing device. It is the object that establishes trusted access with books stored in the cloud and ensures that each and every person purchases their own rights to read each book. The only sharing that is allowed is sharing *the device itself*, which is the thing that a person actually does own. But even then, such an act must be reported back to the cloud: the hardware needs to be de-registered and then reregistered with credit card and authentication details about the new owner.

This is no library—or, it is only a library in the most impoverished sense of the word. It is a new enclosure, and it is a familiar story: things in the world (from letters to photographs to albums to books) are digitized (as e-mails, JPEGs, MP3s, and PDFs) and subsequently migrate to a remote location or service (Gmail, Facebook, iTunes, the Kindle store). The middle phase is the biggest disruption: that is, when the interface does the poorest job concealing the material transformations taking place, when the work involved in creating those transformations is most apparent, often because the person themselves is deeply involved in the process (of ripping vinyl, for instance). In the third phase, the user interface becomes easier, "frictionless," and what appears to be just another application or folder on one's computer is an engorged, property-and-energy-hungry warehouse a thousand miles away.

CAPTURE, LOSS

The enclosure of intellectual property is easy enough to imagine in warehouses of remote, secure hard drives. But the cloud internalizes processing as well as storage, capturing the new forms of co-operation and collaboration characterizing the new economy and its immaterial labor. Social relations are transmuted into database relations on the "social web," which absorbs self-organization as well. In this sense, the cloud's impact on the production of publications is just as strong as on their consumption, in the traditional sense.

Storage, applications, and services offered in the cloud are marketed for consumption by authors and publishers alike. Document editing, project management, and accounting are peeled slowly away from the office staff and personal computers into the data centers; interfaces are established into various publication channels from print-on-demand to digital book platforms. In the fully realized vision of cloud publishing, the entire technical and logistical apparatus is externalized, leaving only human laborers and their thin devices remaining. Little separates the author-object from the editor-object from the reader-object. All of them maintain their position in the network by paying for lightweight computers and their updates, cloud services, and broadband Internet connections.

On the production side of the book, the promise of the cloud is a recovery of the profits "lost" to file sharing, as all the exchange is disciplined, standardized, and measured. Consumers are finally promised the access to the history of human knowledge (that they had already improvised by themselves), but now, without the omnipresent threat of legal prosecution. One has the sneaking suspicion that such a compromise is as hollow as the promises to a desperate

city of jobs that will be created in a new constructed data center, and that pitting "food on the table" against "access to knowledge" is both a distraction from and a legitimatization of the forms of power emerging in the cloud. It is a distraction because it is by policing access to knowledge that the middleman platform can extract value from publication, both on the writing and reading sides of the book; and it is a legitimation because the platform poses itself as the only entity that can resolve the contradiction between the two sides.

When the platform recedes behind the interface, these two sides comprise the most visible antagonism: they are in a tug-of-war with each other, yet neither the "producers" nor the "consumers" of publications are becoming wealthier or working less to survive. If we turn the picture sideways, however, a new contradiction emerges between the indebted, living labor of authors, editors, translators, and readers on one side, and on the other, data centers, semiconductors, mobile technology, expropriated software, power companies, and intellectual property.

The talk in the data-center industry of the "industrialization" of the cloud refers to the scientific approach to improving design, efficiency, and performance. But the term also recalls the basic narrative of the Industrial Revolution: the movement from home-based manufacturing by hand to large-scale production in factories. As desktop computers pass into obsolescence, we shift from a networked but small-scale relationship to computation (think of "home publishing") to a reorganized form of production that puts the accumulated energy of millions to work through these cloud companies and their modernized data centers.

What kind of buildings *are* these blank superstructures? Factories for the twenty-first century? An engineer named Ken Patchett described the Facebook data center in a television interview: "This is a factory. It's just a different kind of factory than you might be used to."[5] Those factories that we're "used to" continue to exist (at Foxconn, for instance), producing the infrastructure under recognizably exploitative conditions, for this is "different kind of factory," a factory extending far beyond the walls of the data center. But the idea of the factory is only part of the picture—this building is also a mine and the dispersed workforce devotes most of its waking hours to mining-in-reverse, packing it full of data under the expectation

that someone soon will figure out how to pull out something valuable. Both metaphors rely on the image of a mass of workers (dispersed as it may be), and leave a darker and more difficult possibility: the data center is like the hydroelectric plant, damming up property, sociality, creativity, and knowledge, while engineers and financiers look for the algorithms to release the accumulated cultural and social resources on demand, as profit.

This returns us to the interface, the site of the struggles over management and control of access to property and infrastructure. Previously, these struggles were situated within the computer-object and the implied freedom provided by its computation, storage, and possibilities for connection with others. Now, however, the eviscerated device is more interface than object, and it is exactly here at the interface that the new technological enclosures have taken form (for example, see Apple's iOS products, Google's search box, and Amazon's "marketplace"). Control over the interface is guaranteed by control over the entire techno-business stack: the distributed hardware devices, centralized data centers, and the software that mediates the space between. Every major technology corporation must now operate on all levels to protect against any loss.

There is a centripetal force to the cloud and this essay has been written in its irresistible pull. In spite of the sheer mass of capital that is organized to produce this gravity and the seeming insurmountability of it all, there is no chance that the system will absolutely manage and control the noise within it. Riots break out on the factory floor; algorithmic trading wreaks havoc on the stock market in an instant; data centers go offline; 100 million Facebook accounts are discovered to be fake; the list will continue to grow. These cracks in the interface don't point to any possible future, or any desirable one, but they do draw attention to openings that might circumvent the logic of access. What happens from there is another question.

Notes

1 Karl Marx, *Capital: A Critique of Political Economy*, vol. 1, trans. Ben Fowkes (New York: Penguin Books, 1990), 573.
2 William James, *Pragmatism* (Cambridge, MA: Harvard University Press, 1975), 28.
3 See Gerald Raunig, *A Thousand Machines* (Los Angeles: Semiotext(e), 2010).
4 Jeremy Rifkin, *The Age of Access: The New Culture of Hypercapitalism, Where all of Life is a Paid-For Experience* (New York: Jeremy P. Tarcher/Putnam, 2000), 4.
5 Quoted in Frank Mungeam, "Facebook Building Facility in Prineville," KGW.com, February 11, 2011, www.kgw.com/news/business/Facebook-building-facility-in-Prineville-115964989.html.

Imaginary Property

Florian Schneider