Icon Icon

Graphical User Interface

Aden Evens
Dartmouth College, United States of America

An abiding fantasy of the digital age imagines the computer so attuned to human desire that it responds almost instantly to the user's slightest gesture. This ideal computer would require no mediation to grasp and execute the user's instructions. The human complement to this machine would be the ideal user, one who has internalized the computer's capacities to such a degree that they feel natural; she would offer her commands in a language already assimilable to the digital algorithms of the computer. This fantasy of immediacy, identified in Jay Bolter' and Richard Grusin's *Remediation* as one of two opposed drives that characterize new media, locates its telos in the disappearance of the interface. It is tempting to regard the user interface as a crutch, an intermediary that sullies the otherwise pure relationship we could have to the digital domain "inside" the computer. To eliminate the interface would promote an immediate and intuitive relationship to the machine, saving much needless effort. "Intel research scientist Dean Pomerleau told *Computerworld* that users will soon tire of depending on a computer interface, and having to fish a device out of their pocket or bag to access it. He also predicted that users will tire of having to manipulate an interface with their fingers. Instead, they'll simply manipulate their various devices with their brains" (Gaudin, 2009).

I aim in this essay to debunk this fantasy as ill founded. The user interface is necessary not just to mitigate poor computer design or to hold us over until we

understand the machine well enough to operate on it more directly. Rather, the interface serves the essential role of negotiating an irreconcilable difference between two domains, the human and the digital. Human beings are not calculators, do not read or think or express themselves in a binary code and cannot practically do so. The user interface accommodates the differences between these two worlds but in so doing it necessarily misrepresents them to each other. Any choice of interface involves a particular set of limitations on the possibilities for expression in that interface, pointing users down certain paths while cutting off access to other possibilities. Similarly, every interface distorts the digital domain it is supposed to present, erecting phantasmic objects and coherent operations where there are in the machine only sequences of 0s and 1s. Examining the *Graphical User Interface* (GUI), now pervasive on personal computing devices, this essay shows the design choices that constitute the interface and articulates the consequences of those choices. The interface is that edge where the digital meets the human, but the two sides are inevitably altered in this juncture.

Introducing the computer for the rest of us, the GUI marked a decisive step in the history of the digital. During the '70s, the computer was an administrative and research tool, increasingly adopted by hobbyists, by the upwardly mobile, and in classrooms. In 1984 the Macintosh brought the GUI, under development for more than fifteen years, into the mainstream. No more lists of commands to memorize, no more arcane codes and rigid syntax. Much of what the computer has to offer to the user is illuminated right on the surface by the flickering pixels of the monitor, outlining objects on which to act and suggesting possible actions for those objects.

Windows, Icons, Menus, and a Pointing device: the GUI is sometimes called a WIMP interface. Using these elements, the file system and its standard operations (copy, delete, move, list, rename, etc.) are laid out in virtual overlapping planes, loosely governed by the metaphor of the monitor screen as a desktop. This metaphor, stretched to incredulity from the start, is further taxed

by the many applications hosted within the GUI, from three-dimensional projections in a gaming window to the hyperlinked space of a browser to the infinite, indexed cells of a spreadsheet. These are not easily recognizable as artifacts from atop someone's desk, but then we don't tend to decorate our cubicles with icons, either. The desktop metaphor was only ever a thin contrivance for directing the intuition, to give a hint of the familiar to an esoteric technology. Young people, too naive to be intimidated by computers and unburdened by the rigorous adult demand for metaphorical consistency, were possibly the perfect audience for the desktop and the GUI built on top of it. The GUI rendered the computer much easier to use without expertise but at the cost of some power and efficiency, a technology of a lower common denominator.

Of his own considerable contribution to the invention of the GUI, computer researcher and interface guru Alan Kay dedicates much to the young people who were his test subjects and "primary motivation" for much of the research he conducted in partnership with Adele Goldberg at Xerox PARC in the 1970s (1993: 79). Kay held that the computer offers extraordinary creative possibility, but he lamented that that possibility is out of reach for those lacking extensive technical expertise. Hoping to lower this barrier, Kay's team designed an interface simple enough for a child to operate, with a screen full of pictograms, symbols that resemble (at least somewhat) what they represent, and an intuitive pointing device for input that relies on an embodied sense of space acquired in early childhood. To generalize and popularize this interface, Kay helped to invent the programming language Smalltalk, including a visual environment wherein one writes a program by arranging icons within a window. Even children can use such a system to tap into the computer's potent possibility: Kay provides examples from toddlers through high school students (1986). Manifest in the immediacy of recognizable, iconic images, the computer reveals its inner logic to anyone with eyes to see and so becomes wholly available to a general audience.

Under the guidance of reputed visionary Steve Jobs, Apple Computer recognized

(and capitalized on) the underlying principle of the GUI: it puts a human face on computing. The award-winning box art for the original Macintosh featured primary colors and iconic line drawings, penned as to recall the crude and playful simplifications of a child's hand. A couple of system updates down the road, the Macintosh startup screen put the sign in the window (as it were), showcasing the "happy Mac," a drawing of an early Macintosh all-in-one computer with a smiley face on its screen. The happy Mac eventually insisted even more literally on the amalgam of computer and human, becoming a rectangular figure of a monitor broken by a squiggly vertical line to suggest eyes, nose, and mouth.

What would be more iconic than a smiley face? The happy Mac is the epitome of the icon, the iconic icon. The smiling face—eyes, nose, and mouth—resonates deep within our brains, such that even the simplified line-drawing of a smiley refers immediately to an elemental experience, a neural structure basic to human being. But the happy Mac is not a face, only ever an icon of one. Its simplified features do not so much resemble as signify the face, as underlined by the generic figural language of the smiley, seen on t-shirts and bumper stickers. The happy Mac enacts a strange self-referential logic: shortly after powering up under Mac OS 8.6, the Macintosh computer screen shows a small iconic image of the original Macintosh, on the tiny screen of which are the features of a generic smiley face. This cheery icon indicates that the startup sequence is proceeding correctly, that a disk with bootable software has been located and is being read in to memory. The on-screen representation of an iconic computer stages one of the central problems of the digital interface, namely, that the materials available to present on a computer are always already digital and hence iconic or generic. The interface can only present what is digital, offering to the user the computer "inside" the computer. The smiling face on the iconic computer screen pushes back against this tail-swallowing digital trap; after all, the point of including a smiling face is to insist, against all evidence, on the underlying humanity of the computer. (An early print advertisement for the original 1984 128k Mac shows the computer with the word "hello" on its screen, drawn in a cursive hand. This

is a subtler version of the same message: the human and the digital can indeed meet and can do so on human turf.) Even at the time, these pseudo-human gestures flickering on the grayscale screen of the Macintosh seemed awkward, calling attention to the clumsy mismatch of digital and human. With the distance of history, these miniature images—not just the small image of the computer but also the smiling face on its little screen—lose their humanity altogether, drawing attention not to any simulated emotion but only to the pathetic fallacy of the digital.

Iconic Mentality

From Xerox's corporate perspective in the '70s, Kay's researches into a personal computing interface may have seemed rather pointless: why would scientists, engineers, and information technologists want computers with cute little images on the screen? They charged Apple a laughably small fee to incorporate Kay's interface concepts into their new operating system. Though Kay was not the creator of the happy Mac image, it might still be thought of as his signature on the Macintosh interface, for it underlines the core principles of his conception of how the interface should work. Kay inherits his pedagogy from American psychologist Jerome Bruner, who loosely adapts from Jean Piaget a developmentalism, modifying the relatively strict developmental stages of Piaget to allow for lifelong learning and change. Bruner identifies three stages of mentality, the enactive, the iconic, and the symbolic. The standard paraphrase holds that the enactive mentality, dominant in infancy, is about acting bodily on the world, the iconic, onset in early childhood, is about images and resemblance, and the symbolic, blossoming in adolescence, is about abstract connections.[1]

The driving force behind the GUI is the mediate position in this triad of the iconic, which leads learners from an infantile enactive mentality to a sophisticated symbolic one. By providing simple and legible icons that visually suggest their meanings, the interface is supposed to encourage users to pass from an understanding of the computer through embodied action (the mouse) to an

understanding based on logic and symbolic reasoning (programming). Hence the rather trite title of Kay's well known video lecture, "Doing with Images makes Symbols" (1986), in which Kay outlines some of the history of the GUI, tracing its principles to Bruner and its technical foundations (mouse and windows) to Douglas Engelbart.

The GUI does not merely substitute suggestive images (icons) where formerly had been elliptical codes; it is not just a less intimidating means of accessing the same possibilities. Locating the icon between the material mouse and the symbolic binary, the GUI presents a different regime of possibility, a different structure of computing. How does the GUI convey the availability of the computer to the user? It encapsulates digital objects into icons, imbuing those objects with a visual and affective coherence.[2] Arraying digital objects in space, the GUI establishes relationships among those objects. It provides an enduring visual presence to the objects "in" the computer, altering a fundamental tenet of computing: one need no longer already know what the computer can do in order to work with it.

Crucially, not only digital states but also digital processes become objects punctuated in the interface, captured by an icon or a menu item. Prior to the icon, one would only encounter a file in the course of an operation on it.[3] Now it sits in plain sight on the desktop ready to be clicked, indicating its existence and availability. The computer's contents are laid bare by the GUI, such that it not only invites experimentation and use but also reveals its inner order to the eye, lowering its threat by voluntarily disclosing its innards.[4] The GUI renders some objects and some actions close at hand, promoting those objects and those actions, calling the user's attention to the set of possibilities surrounding those objects. Visuality rather than logic or memory comes to dominate the interaction with the computer, not only foregrounding certain choices, but changing the kind of cognition associated with computer use.

Before the introduction of the GUI, the interface took its cues primarily from the

computing hardware. The computer follows a linear process, accepting input, processing that input through a series of logical calculations, and then producing an output. Each tick of the clock yields the next state. The text-based, command line interface mirrors this linear process of computing. It spews out a stream of data, each character proceeding the previous one, each line following the last, and only the most recent information is current or active. The command line prompt represents the context of current operations, an edge between what has been established (above the prompt) and what is still to be calculated (below the prompt). The text already on the screen (or printed out on the printer) is inert, showing former states of the program that are not available except as a passive reference.

With a graphical interface, many items and many operations are simultaneously available, visible on the screen and ready to be clicked. Instead of entering codes according to an inflexible syntax, one need only point to a location on the monitor and click there. For Kay, this flat simplicity of the interface promoted creative possibilities: "The flitting-about nature of the iconic mentality suggested that having as many resources showing on the screen as possible would be a good way to encourage creativity and problem solving and prevent blockage" (Kay, 2001: 129).[5] Filling the monitor screen with icons encourages the recognition of unforeseen connections, the choice of otherwise unfamiliar operations. In part the novelty presented to the user derives from the flatness of the GUI: not only does the flat monitor surface allow only a limited representation of depth and a relative simplicity of structure, but this same interface admits the juxtaposition of heterogeneous digital objects and operations. The icon for a volume of data can sit on the desktop next to the icon of a text clipping containing a phone number. There is little in the interface to suggest the disparity of these two adjacent icons, and from the user's perspective they are indeed mostly equipotent. Each can be opened, copied, renamed, moved around, or removed from the desktop. The interface helps to hide the differences between these elements, and the consequent misrecognition may well spark some creative accidents. A visual interface encourages unexpected

connections by placing in the foreground icons that the user might otherwise never consider in a given context. A user will more likely try a different font when the possibility of switching fonts is presented right on the screen. An image filter represented as a button will be invoked more frequently and in more contexts than one buried in many layers of menus or only accessible by typing its name.

Given that the possibilities of action are laid out on the monitor, a graphical user interface—even a more moderate, less busy one—favors certain kinds of users, those who are *learning* to use the computer. With available options displayed on the screen, users can find what they are looking for without having to rely on memory or symbolic thinking. The GUI encourages experimentation and play (within the boundaries of its representations), while comforting users and reducing the pervasive anxiety and intellectual effort of having to call forth an expertise. This is Kay's explicit goal, reflecting his philosophical commitments and his youthful audience of testers, but what about users who already know how to use the computer and now want to get their work done? For many years after Apple commercialized the GUI, expert users disparaged the Macintosh as an unsophisticated toy. The tired debate pitting Microsoft Windows against the Apple Mac remains a perpetuation of dueling philosophies: the computer as a learning environment versus the computer as a productivity machine. (Thus it was never incidental nor merely a matter of marketing that Apple dominated in education while Microsoft ruled the business world.) It may well be of benefit to an expert user to be able to call forth any file without having to navigate through layers of folders, without having to remove her hands from the keyboard, without having to engage in visual cognition at all. The command line intimidates non-experts, but it provides a simple mechanism to invoke any process or any file with any modifying parameters often using a single line of coded text. Moreover, the command line interface is consistent and efficient; input and output share a common form (text), and just about every character conveys essential information. By contrast, icons carry very little information relative to the amount of space they occupy and the amount of time they persist

on screen. Unlike an icon, the textual interface doesn't tell the user (unbidden) what is available, but such information is at best superfluous to one who already knows what her machine can do. That Apple's interface (if not their operating system) ended up winning the desktop wars suggests much about how our culture has adopted the computer.

Modelessness

Designing an interface that functions in accord with the cognitive style of non-expert human beings presented a new set of challenges. Experts could be counted on to follow a set of procedures, not just a precise syntax but an order of operations corresponding to the computer's own linear ordering of tasks. The two-dimensional spread of icons, menus, and windows invited a less linear interaction, with diverse possible actions visually available all at once. The GUI was designed to encourage experimentation, clicking around just to see what happens. To accommodate this loosened ordering of operations, the GUI emphasized *modelessness*; a modal interface, that is, an interface with modes, necessitates multiple steps to complete a particular operation. Many operations require that the computer first be switched into the proper mode, and this typically involves exiting the current mode in which the computer is operating. "This is what *modeless* came to mean for me—the user could always get to the next thing desired without any backing out" (Kay, 2001: 129). In a modal interface, each application might institute its own mode, such that switching to another application means first terminating or suspending the current one.

Given this description, modality might seem entirely undesirable, an unnecessary limitation on user choices that is at best clumsy and inefficient. However, with limited processing and memory resources, modes actually allow greater computing efficiency, since programs can exercise a greater control over the context in which they are run. In text-entry mode, a program need only interpret each input as either text to be entered or an instruction to exit the mode; no other inputs are possible, which limits what the user can (immediately) do but

makes the programmer's job a lot more cut and dry.

The gain of a modeless interface is a perceived efficiency for the user: no longer does she have to execute commands just to get ready for the next task. The administrative labor of computer use is reduced, with a savings primarily in terms of time. A mode functions as a temporal context; at a given moment, the computer is prepared for one kind of task or for another, and switching modes takes some time. Instituting a kind of modelessness, the GUI trades space for time, laying out the options on a plane (or on a sheaf of overlaid planes).[6] Now instead of switching modes, the user clicks on an already visible icon or window and immediately begins working at her new task. She need not terminate her current operation before switching to another. In the GUI, (what were formerly) modalities are now represented by different icons or different planes or different windows, spatial arrays that reduce the amount of time spent switching modes. (Modes have been much reduced but have not entirely disappeared from the modern GUI; windows such as alert messages and administrative password authorizations require action before any other task can be resumed.)

Modelessness typifies progress in the history of the interface, as many interface innovations compress the temporal dimension of computing. Consider the history of the interface in terms of the shift from batch processing to interactive processing. For the first twenty years of commercial digital computing, computing resources were scarce and most computers were shared among many projects and many users. Rather than interacting with the machine directly, users would submit programs (for example, on punch cards) to a queue, to be run in their turn. Fed into the computer, the program would run for the most part without human interaction: input program, output result. A bunch of programs to be run were a *batch*, so this style of allocating scarce resources over time with little interaction during program execution was called *batch processing*. Batch processing takes for granted the perspective of the computer, and does not work to accommodate the exigencies of human being. That is, a batch process assumes that the computer's state can be known in advance, that it will not

generally get interrupted, that its task is intended to proceed from start to finish without extra-digital guidance. This structure understands the computer as task-oriented, designed to complete a series of tasks it has been assigned. It works on computer time, on clock time, as clock cycles are considered expensive or valuable and so must be used efficiently, getting the most computing done in the least amount of time and resources. A modal interface works well in this case, as the moments when the interface is available to the computer operator are relatively anomalous and separated from the actual calculations of the program.

The shift to a modeless interface corresponds to a need to make the computer more available to the user, more prepared to accommodate the contingent, arbitrary, or unpredictable nature of human needs and desires. *Personal* computing calls for modelessness, as a person's attention is, generally speaking, modeless, ready to switch focus with little preparation or provocation. Working within the fixed linearity of a computer chip does not come easily to most of us. The modeless interface offers a way of interweaving or enfolding human and computer temporalities. We might even claim that not just time but space gets enfolded, as the linear zero-to-one dimensional computer process gets arrayed on the flat screen to provide a convincing simulacrum of two-to-three dimensional availability.[7]

Modelessness was one of the watchwords of this transformative technological invention, but Kay introduced other substantive emphases in designing the interface. He gives special credit to the inventions of Douglas Engelbart. Engelbart's complementary developments of windows and mouse might stem from a common inspiration, the flat window of the monitor screen. The light pen, similar to today's ubiquitous stylus, was very useful for certain tasks but was not in general an ergonomic input device, especially when used with a keyboard. Not only did it obstruct visibility of the screen on which the user was "drawing", but it also required an awkward arrangement of the user's body, to write on a monitor positioned vertically at arm's length. Engelbart recognized, however, that in the digital domain even place could be dislocated and re-placed.

He virtualized the plane of the monitor screen: the mouse is the light pen moved to a more ergonomic plane, one that mirrors the monitor but is no longer coincident with it.

Kay effected a similar deterritorialization on Engelbart's other major contribution to the GUI, the window. Confronted with the problem of limited monitor space, Kay recognized that windows, already understood as independent virtual spaces, could be stacked in overlapping planes, with the front-most window remaining visible and hiding those behind it. The now conventional GUI element of stacked windows met with resistance when first suggested, as some Xerox PARC researchers believed that users would be too confused by obstructed or partially obstructed information. This objection may have seemed more pressing because visible borders were then only just being introduced to windowing systems, so the visual clarity of frames around windows could not be taken as given. In any case, what now seems commonplace was then a stroke of genius, the virtual third dimension in which windows are stacked. (Note that this is not the virtual third dimension of Albertian perspective; the planar area of a window does not shrink as it recedes into the virtual depth of the monitor screen. There is no vanishing point, hence no simulated visual or spatial depth. Rather the plane containing each window is infinitely thin, and they are all stacked within a single infinitely thin plane. They cover each other but they never stack up to a thickness in depth. The shadows and other edge effects that decorate windows in modern operating systems are thus unmotivated and even incorrect according to the visual logic of window depth.)

The Purloined Letter

From modality to modelessness, from corporate to personal, from expert to novice user, the GUI coalesces at the convergence of many strands of computing progress. Kay's dream was to bring computing to the masses but at the same time to transform the computer from a rapid calculating machine to a locus of creative expression. Though the industry presents this history as a continual and

ongoing progress, this analysis hints at the losses that accompany the gains, suggesting that any interface design involves both advantages and compromises. Windows and mouse answer to the most general conundrum of interface design: the human and digital domains are at odds, and the interface must negotiate the awkward or uncomfortable juncture of their meeting. In the temporal domain the digital follows a linear progress, each tick of the crystal clock measuring a single-step advance; past and future narrow to a slice of present, the current value of the output, the command to be executed. For human beings, the world is confined neither to a linear progress nor to a narrow present; our world comprises a context of attention, a simultaneous manifold availability of a whole perceptual field, whose meaning derives from its relations to past and future. (Kay calls this the "flitting-about nature of the iconic mentality.") The human world is wide. Modelessness expresses each world to the other, simulating in the digital the general availability of context by rapidly circulating among many options, one at a time, while the side facing the human retains its digital surveyability, constituted by individual elements that can be easily distinguished and understood as wholly separate.[8] Human consciousness is multiple, ambiguous, contingent, roaming, and associative, and the modeless GUI helps to embed this style of thinking into the determinate, narrow, linear process of the digital.

The awkward accommodation of human and digital domains that we call the interface is laid bare in another of Kay's remarkable innovations.[9] He describes his solution to the problem of modality in a text editor (word processor):

> The most difficult area to get to be modeless was a very tiny one, that of elementary text editing. How to get rid of 'insert' and 'replace' modes that had plagued a decade of editors? […] Over a weekend I built a sample paragraph editor whose main simplification was that it eliminated the distinction between insert, replace, and delete by allowing selections to extend *between* the characters. Thus, there could be a zero-width selection, and thus every operation could be a replace. 'Insert' meant replace the zero-

> width selection. 'Delete' meant replace the selection with a zero-width string of characters (Kay, 2001: 130).

Prior to this zero-width selection technique, text editing required that a user first put the computer into a mode suitable to her next operation, inserting, replacing, or deleting. Kay's trick is to make each of these operations digitally equivalent by programming the editor so that when the user has no text selected, the computer nevertheless "has" a selection consisting of the non-space, the boundary between characters where the cursor is currently positioned. Instead of a selection of a few contiguous characters or a single character, such as a space or a letter, the selection has a width of zero. As Kay explains, this allows for each of the three operations to work the same way: whatever is next typed replaces the selection (and if a backspace or delete is typed, this replaces the current selection with a zero-width selection).

The notion of a zero-width selection provides an archetype of virtualization. In order to connect the digital and human domains, the interface posits a strange non-entity, a purely digital or virtual object, a selection that consists of nothing. It is the introduction of this contrivance, this digital thing that corresponds to no actual or human thing, that makes possible the most natural or human interaction with the text editor. From the perspective of the digital, a selection of zero characters is just as valid as a selection of one or twenty-one characters. The same rules apply, the same operations can be conducted, the same algorithms, the same code can manipulate this virtual object.[10]

Due to Kay, the letter is now well and truly purloined. The zero-width selection comprises a letter that is not there, a letter whose content is only the pure possibility of being filled in by text that the user has yet to enter. The text editor attains modelessness by virtue of a digital object whose only reference is to itself, an absent letter, not missing from its place for it is an abstraction that defies the sticky demands of the material letter that must always be somewhere. (The digital is often compared to spoken or written language or to the alphabet, and though they all share a crucial quality of abstraction, the digital intensifies this

quality much further to divorce itself altogether from the particularity of the material.) Not (yet) filled in, this zero-degree selection could (but does not) have any of a number of predetermined values; its value is to not be any of those values but to be ready to become any of them. This noncommittal readiness is the key to modelessness.

The zero-width selection may seem like an odd contortion, a *kludge*, as programmers would say, but it mirrors the essence of the digital's underlying ontological modality. The digital depends fundamentally on a difference between 0 and 1 but a difference without content. In the binary code, the difference between 0 and 1 is purely formal, which is to say, modeless. Every bit is what value it is, 0 or 1, but the substance of its value is precisely that of not being its other. The bit, as digital operator, has an unusual ontological status: it is that it might have been its other. *Might have been* is in this case substantive, not an idle counterfactual. A 0 asserts nothing more than not-1 and so also asserts could-be-1. Whatever sense a 1-bit makes, it does so because it could be 0. Thus the zero-width selection, which also operates by virtue of what it might be or could be, typifies digitality, reproducing the fundamental dynamics of the binary at a higher level of interactivity: the ontological mode of the digital is precisely the might be, pure possibility rendered materially effective in the technology.

The might be, the defining character of the bit, casts the digital as a technology of material abstraction. That is, by reducing all difference to the difference between 0 and 1, the digital captures a manifold world including the processes that manipulate it. But only because 0 and 1 carry with them the subjunctive modality of the binary, the abstraction of the digital includes always its pure possibility. This is not merely a metaphysical claim about the digital, an undetectable fact operating behind the scenes; rather, the might be thoroughly infuses the culture of the digital. From its origins, the dominant mood of digital technology has been futural; the digital promises more to come, a world of endless possibility. (Recall the cursive *hello* on the Mac's screen in 1984. It was clearly inadequate, all too digital, but it conveyed a promise about the future of

the digital. The digitally rendered *hello* and the smiley face icon were the shape of things to come.)

Unspecified possibility, or possibility per se, derives from the central power of the digital; the digital's decisive technique is the power of abstraction. Rendering all difference as the formal difference between 0 and 1, the digital claims to capture every difference but only by neutering it, by draining difference of its dynamism, its generative or creative force. Difference in the material, in the actual, is not a static posit but an activity, an event. The digital journals the results of this activity but always and ever fails to grasp its essence in action, its production.

Interface as Icon

And this is the core conflict that the interface must navigate, to connect the crystalline abstractions of the digital code to the overflowing and scattered attentions of the human. Abstraction preserves a similarity of structure between the human and digital domains. But it draws off (*tract*, to draw + *abs-*, away) the material specificity of the human, leaving a skeleton of code without a fleshy materiality. Thus stands the problem of the icon, the chief representative of the graphical user interface. The icon epitomizes the interface, a flat surface with its visible side turned toward the user and its invisible side facing the binary. It stands *between* user and computer, or enactive and symbolic, while it stands *for* a file or a process and presents in a perceptible form some of the pure possibility that is the modality of that digital file or process.

From the user's perspective, the icon shows the status of something digital, evincing not so much the existence of a file or process but its availability. The icon asserts that something can be clicked. The notion of representation may be relevant here, but it only partially captures the relationship between icon and digital object. The icon is also a kind of handle by which the user gets a grip on the file or process it stands for. It declares that there is "in" the computer an

encoded object, a data structure or algorithm, but this *there is* is not passive or static. The stood-for data is significant because it might be clicked, because it might be altered, because it might be other.

The icon defines an area of geometric equipotential, such that clicking anywhere within it has the same meaning. This is its unity or coherence, its object-orientation. Presenting the general availability of a digital object, the icon occupies a continuous planar field, an enclosed shape whose geometric integrity corresponds to the purported coherence of the object it stands for. Frequently, a click anywhere within an icon has a uniform effect. But icons can also be internally differentiated, where a particular part is highlighted, or a click on one part of the icon produces a result different from a click elsewhere within the icon. In general, the meaning of an icon derives in part from these internal differentiations and from the possible visual differences that distinguish an icon from others like it or from the same icon in a different state. (An icon selected looks different from the same icon unselected. An icon representing an alias to a file looks marginally different from an icon representing the original file. An unavailable process or menu item appears dimmed or grayed out, a possibility repealed, deferred, or aborted.) The icon must be read against a field of its possibilities, what it is not.

The icon reports the results of alterations to the object, showing both current and possible states. As style guides frequently remark, the icon exists as a collection of similarities and differences. It is related by similarity to a group of other icons that it resembles, which tells the user that it represents a particular kind of document or is associated with a particular application or can perform some particular kind of function. Icons for mathematical operations might have a different visual style than do icons for alphabetic manipulations, for example. It is related by difference to every other icon, including even itself. That is, its differences not only distinguish it as the particular icon it is (for one application versus a different application, or a cut operation versus a copy operation) but also distinguish its current status.

Its confinement to a vertical plane coincident with the monitor screen gives the icon two sides. On one side the icon presents information and accepts instructions. On the other (virtual) side the icon invokes a digital process, to open a file, or change a font, or start a calculation. Thus, the flatness of the icon is no technological limitation, as though the monitor imposed a final constraint on it. The interface may be graphical or visual, but the objects to be manipulated in the computer and the manipulations themselves are digital, structural rather than spatial, abstract rather than material. Some data may lend itself to three-dimensional, spatial representation, but, as I've been arguing vis-à-vis the icon, the connection between data and visuality is usually strained at best. Our chosen compromise so far has been to deal with abstractions largely in two dimensions, or rather, in planes. The original abstract image was drawn in two dimensions or carved into the surface of a plane. (Perhaps a club wielded as a weapon is already the abstract image of a head or a hand, a sculptural, three-dimensional abstraction. But the club is an abstraction only after the fact, for it remains precisely concrete, operative at the level of materiality. Its structural similarity to the head is not accidental, since it serves as the basis of the club's effectiveness for bashing heads, but neither is its form determined principally by a process of abstraction.)

Thus popular fantasies about a digitized spatiality, a three-dimensional, digitally simulated world, are misguided. The flat icon is actually the easiest way to get to the computer. There may be a number of things that we do with computers that benefit from a representation in three dimensions, but not most things. Simulated three-dimensionality could become wildly popular as a medium on the computer, but the primary computer interface will remain stubbornly two-dimensional, icons, windows, documents, layers, desktops. Some scientists and engineers can work effectively in three dimensions, as can plenty of artists, educators and business people. Gamers have a field day in a simulated space. But as long as information is primarily abstract, textual or structural, its presentation will be best suited to two-dimensional planes. For Kay, two-

dimensional representation has the added advantage of a leveling effect, the flattening of hierarchy: "What seemed to be required were states in which there was a transition arrow to every other state […]. In other words, a much 'flatter' interface seemed called for […]" (Kay, 1993: 72).

This analysis of the icon can likely be generalized to describe the whole interface. The visual interface follows the logic of the icon throughout. Not just the small images that stand for files and folders, not just the pictograms that populate palettes and toolbars, but every element of the visual interface, from menu items to scroll bars to characters of text, all adhere to an iconic logic. Each object in the interface stands as representative of some non-presentable version of itself, a binary encoding whose visual presentation is only one face, one aspect. What one sees on the computer screen is in principle generic, iconic, representing a digital artifact that is both more and less than its material analog. It exceeds its material analog in that it includes possibility as part of its way of being, a set of possibilities for how it might otherwise be. But it falls short of its material analog in that it lacks the generative specificity of the material, the creative renewal that is the unique ontological mode of the material world.

Take as an example the computer game. Computer games frequently simulate three-dimensional spaces, and their on-screen presentations are dynamic and supple, which would seem to contrast with iconic logic. Nevertheless, just as an icon does, each item in a game appears in a certain way that represents its generic state and its available possibilities. The player reads the possibilities from the representation and adjusts her algorithm accordingly. Not just the images but the habits, the twitches, the tics, the slogans, the patterns. In the game persona of a hard-boiled but good-hearted rogue hero, you're crossing the street, on your way to eliminate an evil terrorist boss. Suddenly a siren sounds followed right away by a voiceover cop calling over the radio, "We've got him, boys. Close in." Then you hear shots fired, *ricochet*, maybe the music changes pace. The first time you experience this sequence, you're so busy figuring out what to do, which buttons to push, how to escape, that you do not really notice

what is going on. But after two or three nearly identical episodes, you come to recognize the signs, the elements of this icon unfolding in sequence. Soon you develop a strategy for evading the cordon of police (make for the rooftops?), thence employing it at need, with the siren triggering your response so rapidly that you have left the scene before the gunshots ring out. (Perhaps you still hear the shots if the icon is especially stubborn.)

Other interface elements fit even more comfortably into the logic of the icon. Menus and their items trigger digital processes, occupy clearly defined spaces, and indicate a *mouseover* event by lighting up. The desktop itself acts iconically, accepting clicks indifferently over its whole surface, and playing host to an image that is a kind of pure presentation with only an aesthetic significance. Text on screen functions as an archetypal icon, the iconic principle in action. Each character stands for itself, or, what is the same thing, its difference from the others. Clearly defined, each character nevertheless has a great variety of standard forms, conventionally separated into fonts. A given character is significant primarily by virtue of the ideal for which it stands, the immaterial form of the letter that takes on a particular material instantiation in a given font at a given size. In a digital text, everything is encoded, encode-able, nothing escapes; the digital letter too bears an oddly arbitrary relationship to its presentation, as even its material appearance no longer grounds its meaning. Instead it is the digital code "behind" the letter that undergirds its significance, and this is the hallmark of the icon, an essential relationship between an ultimately superficial appearance and an immaterial ideal of code "inside" the machine.

It would be folly to question the need for a user interface. Human beings are not calculators and do not think or act in the 0s and 1s that constitute the digital domain. But this essay has argued that the problem of the interface is more intractable than a matter of translation. It's not just about finding a way to make the interface more convenient or appealing or efficient. The gap between digital and human stems from an ontological incommensurability for which there is no

natural or correct resolution. Every interface involves its share of compromises. Every interface encourages certain choices over others and presents the digital to the human in a particular light. The history of the interface is the story of the ongoing negotiation between digital and human, and the two domains do not invariably approach each other. Individual elements of the interface represent this negotiation in progress, a dynamic compromise that avails the user of some aspects of the computer while hiding or burying other aspects. How to represent abstraction using colors and shapes; how to make available something of the pure possibility of the digital in an icon; how to offer an invitation and a provocation, a learning environment and a productive one; how to stimulate creativity in a context of prescribed choice; these are the challenges whose response is the developing design of the user interface.

Notes

[1] Piaget proposes human development in four stages: sensorimotor stage, from birth to age 2; preoperational stage, from ages 2 to 7; concrete operational stage, from ages 7 to 12; and formal operational stage, from age 12 onwards. Bruner borrows from Piaget throughout his career; his three stages of learning are not so chronologically anchored as Piaget's four stages and are discussed directly in *Toward a Theory of Instruction* (Harvard University Press, 1966).

[2] Though this essay does not address the question in detail, the status of what gets encapsulated in an icon is problematic. What is the *thing* that the icon represents?

[3] This claim requires qualification: in the normal operation of the computer, a user constantly encounters thousands of files that are part of the operating system. But she is rarely aware of any of those files, and explicit awareness of a file tends to occur only when that file is accessed deliberately.

[4] The GUI has been criticized for hiding the inner operations of the computer behind its simplified images, for falsely presenting the computer on human rather than digital terms. This critique has real merit, but the situation is more complex: the GUI reveals certain aspects of the underlying structure of the digital data while burying other aspects.

[5] "As many resources showing on the screen as possible" sounds like a user interface disaster, but it indicates the excited sense of revolutionary possibility that motivated Kay.

[6] Note that the flat planes of the computer interface not only fall short of three dimensions because they are flat, but also because, aside from the top plane, the order of the planes makes little difference. The z-dimension for each plane is maintained for visual consistency and legibility, but the layering does not distinguish in any significant way between the second plane from the front and the rearmost plane. Perhaps the desktop constitutes a counterexample, as it is always the rearmost plane and so also has a kind of priority, as when a command allows the user to sweep temporarily everything off of the desktop so as to access the icons that sit always on that desktop.

[7] A quick note of explanation: If a point is zero-dimensional and a line is one-dimensional, then the process of the CPU in a computer measures in between these two dimensionalities, as it can be thought of as the leading edge of a line being drawn, the current operation at the end of a chain of operations. The GUI is two-to-three–dimensional because it layers two-dimensional planes in a third dimension of depth. Critics sometimes call this a 2.5-dimensional or 2.5D visual perspective.

[8] That is, one way for the digital to accommodate the GUI to its relentless linearity is by successively polling each icon, each menu item to see if it is being activated. If this polling takes place sufficiently rapidly, the user has the impression that each interface element is always at the ready. Correspondingly, the interface is available to the user as a set of distinct possibilities, a list arranged in two dimensions of things that might be clicked.

[9] As is evident throughout Kay's writing and video presentations, he is genuinely modest and inordinately generous, deferring credit and acknowledging the contributions of his collaborators and of the many others who often inspired his ideas, labored to implement them, or developed similar ideas in parallel. I have elided from this quotation about zero-length selections Kay's acknowledgement that Larry Tesler had already developed this same idea. It is less important to identify the author of an idea than to celebrate its worth. For each of the inventions attributed in this present essay to Kay, he acknowledges multiple authorship, and a broad range of significant influences.

[10] Readers who tend to think more like a computer may notice one problem with this account: if nothing is selected (the zero-width selection) and the user presses

backspace, it would not achieve the intended result to replace the zero-width selection with another zero-width selection. In effect, nothing would happen. Instead, special code must be included for this very situation. One way of addressing the issue is to check whether this situation obtains (a *backspace* pressed when nothing is selected) and if so, to immediately and automatically select the character right before the zero-width selection, then to proceed with the replace operation. In other words, select the previous character then replace it with nothing. Such complications to accomplish something seemingly simple (a *backspace*) may be typical of solutions in computing, which gather together multiple cases into a common set of code but sometimes by leaving out certain exceptional cases.

Bibliography

Bolter, David Jay and Grusin, Richard. 1999. *Remediation: Understanding New Media.* Cambridge: MIT Press, Cambridge.

Gaudin, Sharon. 2009. "Intel: Chips in brains will control computers by 2020", *Computerworld*. (19 November).
http://www.computerworld.com/s/article/9141180/Intel_Chips_in_brains_will_control_computers_by_2020.

Kay, Alan. 2001. "User Interface: A Personal View", in Packer, Randall and Jordan, Ken (eds.), *Multimedia: From Wagner to Virtual Reality*. New York: WW Norton & Co. 121-131.

Kay, Alan. 1993. "The Early History of Smalltalk", *ACM SIGPLAN Notices*, 28.3 (March). 69-95.

Kay, Alan. 1986. "Doing with Images makes Symbols." [QuickTime Video from] *The History of the Personal Workstation*. (27 May).
http://video.google.com/videoplay?docid=-533537336174204822