

from *In the Beginning was the Command Line*¹

by Neal Stephenson

About twenty years ago Jobs and Wozniak, the founders of Apple, came up with the very strange idea of selling information processing machines for use in the home. The business took off, and its founders made a lot of money and received the credit they deserved for being daring visionaries. But around the same time, Bill Gates and Paul Allen came up with an idea even stranger and more fantastical: selling computer operating systems. This was much weirder than the idea of Jobs and Wozniak. A computer at least had some sort of physical reality to it. It came in a box, you could open it up and plug it in and watch lights blink. An operating system had no tangible incarnation at all. It arrived on a disk, of course, but the disk was, in effect, nothing more than the box that the OS came in. The product itself was a very long string of ones and zeroes that, when properly installed and coddled, gave you the ability to manipulate other very long strings of ones and zeroes. Even those few who actually understood what a computer operating system was were apt to think of it as a fantastically arcane engineering prodigy, like a breeder reactor or a U-2 spy plane, and not something that could ever be (in the parlance of high-tech) “productized.”

Yet now the company that Gates and Allen founded is selling operating systems like Gillette sells razor blades. New releases of operating systems are launched as if they were Hollywood blockbusters, with celebrity endorsements, talk show appearances, and world tours. The market for them is vast enough that people worry about whether it has been monopolized by one company. Even the least technically-minded people in our society now have at least a hazy idea of what operating systems do; what is more, they have strong opinions about their relative merits. It is commonly understood, even by technically unsophisticated computer users, that if you have a piece of software that works on your Macintosh, and you move it over onto a Windows machine, it will not run. That this would, in fact, be a laughable and idiotic mistake, like nailing horseshoes to the tires of a Buick.

A person who went into a coma before Microsoft was founded, and woke up now, could pick up this morning’s New York Times and understand everything in it—almost:

Item: the richest man in the world made his fortune from-what? Railways? Shipping? Oil? No, operating systems. Item: the Department of Justice is tackling Microsoft’s supposed OS monopoly with legal tools that were invented to restrain the power of Nineteenth-Century robber barons. Item: a woman friend of mine recently told me that she’d broken off a (hitherto) stimulating exchange of e-mail with a young man. At first he had seemed like such an intelligent and interesting guy, she said, but then “he started going all PC-versus-Mac on me.”

What the hell is going on here? And does the operating system business have a future, or only a past? Here is my view, which is entirely subjective; but since I have spent a fair amount of time not only using, but programming, Macintoshes, Windows machines, Linux boxes and the BeOS, perhaps it is not so ill-informed as to be completely worthless. This is a subjective essay, more review than research paper, and so it might seem unfair or biased compared to the technical reviews you can find in PC magazines. But ever since the Mac came out, our operating systems have been based on metaphors, and anything with metaphors in it is fair game as far as I’m concerned.

MGBs, TANKS, AND BATMOBILES

Around the time that Jobs, Wozniak, Gates, and Allen were dreaming up these unlikely schemes, I was a teenager living in Ames, Iowa. One of my friends’ dads had an old MGB sports car rusting away in his garage.

1. Reprinted from <http://www.cryptonomicon.com/beginning.html>

Sometimes he would actually manage to get it running and then he would take us for a spin around the block, with a memorable look of wild youthful exhilaration on his face; to his worried passengers, he was a madman, stalling and backfiring around Ames, Iowa and eating the dust of rusty Gremlins and Pintos, but in his own mind he was Dustin Hoffman tooling across the Bay Bridge with the wind in his hair.

In retrospect, this was telling me two things about people's relationship to technology. One was that romance and image go a long way towards shaping their opinions. If you doubt it (and if you have a lot of spare time on your hands) just ask anyone who owns a Macintosh and who, on those grounds, imagines him- or herself to be a member of an oppressed minority group.

The other, somewhat subtler point, was that interface is very important. Sure, the MGB was a lousy car in almost every way that counted: balky, unreliable, underpowered. But it was fun to drive. It was responsive. Every pebble on the road was felt in the bones, every nuance in the pavement transmitted instantly to the driver's hands. He could listen to the engine and tell what was wrong with it. The steering responded immediately to commands from his hands. To us passengers it was a pointless exercise in going nowhere—about as interesting as peering over someone's shoulder while he punches numbers into a spreadsheet. But to the driver it was an experience. For a short time he was extending his body and his senses into a larger realm, and doing things that he couldn't do unassisted.

The analogy between cars and operating systems is not half bad, and so let me run with it for a moment, as a way of giving an executive summary of our situation today.

Imagine a crossroads where four competing auto dealerships are situated. One of them (Microsoft) is much, much bigger than the others. It started out years ago selling three-speed bicycles (MS-DOS); these were not perfect, but they worked, and when they broke you could easily fix them.

There was a competing bicycle dealership next door (Apple) that one day began selling motorized vehicles—expensive but attractively styled cars with their innards hermetically sealed, so that how they worked was something of a mystery.

The big dealership responded by rushing a moped upgrade kit (the original Windows) onto the market. This was a Rube Goldberg contraption that, when bolted onto a three-speed bicycle, enabled it to keep up, just barely, with Apple-cars. The users had to wear goggles and were always picking bugs out of their teeth while Apple owners sped along in hermetically sealed comfort, sneering out the windows. But the Micro-mopeds were cheap, and easy to fix compared with the Apple-cars, and their market share waxed.

Eventually the big dealership came out with a full-fledged car: a colossal station wagon (Windows 95). It had all the aesthetic appeal of a Soviet worker housing block, it leaked oil and blew gaskets, and it was an enormous success. A little later, they also came out with a hulking off-road vehicle intended for industrial users (Windows NT) which was no more beautiful than the station wagon, and only a little more reliable.

Since then there has been a lot of noise and shouting, but little has changed. The smaller dealership continues to sell sleek Euro-styled sedans and to spend a lot of money on advertising campaigns. They have had GOING OUT OF BUSINESS! signs taped up in their windows for so long that they have gotten all yellow and curly. The big one keeps making bigger and bigger station wagons and ORVs.

On the other side of the road are two competitors that have come along more recently.

One of them (Be, Inc.) is selling fully operational Batmobiles (the BeOS). They are more beautiful and stylish even than the Euro-sedans, better designed, more technologically advanced, and at least as reliable as anything else on the market—and yet cheaper than the others.

With one exception, that is: Linux, which is right next door, and which is not a business at all. It's a bunch of RVs, yurts, tepees, and geodesic domes set up in a field and organized by consensus. The people who live there are making tanks. These are not old-fashioned, cast-iron Soviet tanks; these are more like the M1 tanks of

the U.S. Army, made of space-age materials and jammed with sophisticated technology from one end to the other. But they are better than Army tanks. They've been modified in such a way that they never, ever break down, are light and maneuverable enough to use on ordinary streets, and use no more fuel than a subcompact car. These tanks are being cranked out, on the spot, at a terrific pace, and a vast number of them are lined up along the edge of the road with keys in the ignition. Anyone who wants can simply climb into one and drive it away for free.

Customers come to this crossroads in throngs, day and night. Ninety percent of them go straight to the biggest dealership and buy station wagons or off-road vehicles. They do not even look at the other dealerships.

Of the remaining ten percent, most go and buy a sleek Euro-sedan, pausing only to turn up their noses at the philistines going to buy the station wagons and ORVs. If they even notice the people on the opposite side of the road, selling the cheaper, technically superior vehicles, these customers deride them cranks and half-wits.

The Batmobile outlet sells a few vehicles to the occasional car nut who wants a second vehicle to go with his station wagon, but seems to accept, at least for now, that it's a fringe player.

The group giving away the free tanks only stays alive because it is staffed by volunteers, who are lined up at the edge of the street with bullhorns, trying to draw customers' attention to this incredible situation. A typical conversation goes something like this:

Hacker with bullhorn: "Save your money! Accept one of our free tanks! It is invulnerable, and can drive across rocks and swamps at ninety miles an hour while getting a hundred miles to the gallon!"

Prospective station wagon buyer: "I know what you say is true...but...er...I don't know how to maintain a tank!"

Bullhorn: "You don't know how to maintain a station wagon either!"

Buyer: "But this dealership has mechanics on staff. If something goes wrong with my station wagon, I can take a day off work, bring it here, and pay them to work on it while I sit in the waiting room for hours, listening to elevator music."

Bullhorn: "But if you accept one of our free tanks we will send volunteers to your house to fix it for free while you sleep!"

Buyer: "Stay away from my house, you freak!"

Bullhorn: "But..."

Buyer: "Can't you see that everyone is buying station wagons?"

BIT-FLINGER

The connection between cars, and ways of interacting with computers, wouldn't have occurred to me at the time I was being taken for rides in that MGB. I had signed up to take a computer programming class at Ames High School. After a few introductory lectures, we students were granted admission into a tiny room containing a teletype, a telephone, and an old-fashioned modem consisting of a metal box with a pair of rubber cups on the top (note: many readers, making their way through that last sentence, probably felt an initial pang of dread that this essay was about to turn into a tedious, codgerly reminiscence about how tough we had it back in the old days; rest assured that I am actually positioning my pieces on the chessboard, as it were, in preparation to make a point about truly hip and up-to-the minute topics like Open Source Software). The teletype was exactly the same sort of machine that had been used, for decades, to send and receive telegrams. It was basically a loud typewriter that could only produce UPPERCASE LETTERS. Mounted to one side of it was a smaller machine with a long reel of paper tape on it, and a clear plastic hopper underneath.

In order to connect this device (which was not a computer at all) to the Iowa State University mainframe across town, you would pick up the phone, dial the computer's number, listen for strange noises, and then slam the handset down into the rubber cups. If your aim was true, one would wrap its neoprene lips around the ear-piece and the other around the mouthpiece, consummating a kind of informational *soixante-neuf*. The teletype would shudder as it was possessed by the spirit of the distant mainframe, and begin to hammer out cryptic messages.

Since computer time was a scarce resource, we used a sort of batch processing technique. Before dialing the phone, we would turn on the tape puncher (a subsidiary machine bolted to the side of the teletype) and type in our programs. Each time we depressed a key, the teletype would bash out a letter on the paper in front of us, so we could read what we'd typed; but at the same time it would convert the letter into a set of eight binary digits, or bits, and punch a corresponding pattern of holes across the width of a paper tape. The tiny disks of paper knocked out of the tape would flutter down into the clear plastic hopper, which would slowly fill up what can only be described as actual bits. On the last day of the school year, the smartest kid in the class (not me) jumped out from behind his desk and flung several quarts of these bits over the head of our teacher, like confetti, as a sort of semi-affectionate practical joke. The image of this man sitting there, gripped in the opening stages of an atavistic fight-or-flight reaction, with millions of bits (megabytes) sifting down out of his hair and into his nostrils and mouth, his face gradually turning purple as he built up to an explosion, is the single most memorable scene from my formal education.

Anyway, it will have been obvious that my interaction with the computer was of an extremely formal nature, being sharply divided up into different phases, viz.: (1) sitting at home with paper and pencil, miles and miles from any computer, I would think very, very hard about what I wanted the computer to do, and translate my intentions into a computer language—a series of alphanumeric symbols on a page. (2) I would carry this across a sort of informational cordon sanitaire (three miles of snowdrifts) to school and type those letters into a machine—not a computer—which would convert the symbols into binary numbers and record them visibly on a tape. (3) Then, through the rubber-cup modem, I would cause those numbers to be sent to the university mainframe, which would (4) do arithmetic on them and send different numbers back to the teletype. (5) The teletype would convert these numbers back into letters and hammer them out on a page and (6) I, watching, would construe the letters as meaningful symbols.

The division of responsibilities implied by all of this is admirably clean: computers do arithmetic on bits of information. Humans construe the bits as meaningful symbols. But this distinction is now being blurred, or at least complicated, by the advent of modern operating systems that use, and frequently abuse, the power of metaphor to make computers accessible to a larger audience. Along the way—possibly because of those metaphors, which make an operating system a sort of work of art—people start to get emotional, and grow attached to pieces of software in the way that my friend's dad did to his MGB.

People who have only interacted with computers through graphical user interfaces like the MacOS or Windows—which is to say, almost everyone who has ever used a computer—may have been startled, or at least bemused, to hear about the telegraph machine that I used to communicate with a computer in 1973. But there was, and is, a good reason for using this particular kind of technology. Human beings have various ways of communicating to each other, such as music, art, dance, and facial expressions, but some of these are more amenable than others to being expressed as strings of symbols. Written language is the easiest of all, because, of course, it consists of strings of symbols to begin with. If the symbols happen to belong to a phonetic alphabet (as opposed to, say, ideograms), converting them into bits is a trivial procedure, and one that was nailed, technologically, in the early nineteenth century, with the introduction of Morse code and other forms of telegraphy.

We had a human/computer interface a hundred years before we had computers. When computers came into being around the time of the Second World War, humans, quite naturally, communicated with them by simply grafting them on to the already-existing technologies for translating letters into bits and vice versa: teletypes and punch card machines.

These embodied two fundamentally different approaches to computing. When you were using cards, you'd punch a whole stack of them and run them through the reader all at once, which was called batch processing. You could also do batch processing with a teletype, as I have already described, by using the paper tape reader, and we were certainly encouraged to use this approach when I was in high school. But—though efforts were made to keep us unaware of this—the teletype could do something that the card reader could not. On the teletype, once the modem link was established, you could just type in a line and hit the return key. The teletype would send that line to the computer, which might or might not respond with some lines of its own, which the teletype would hammer out—producing, over time, a transcript of your exchange with the machine. This way of doing it did not even have a name at the time, but when, much later, an alternative became available, it was retroactively dubbed the Command Line Interface.

When I moved on to college, I did my computing in large, stifling rooms where scores of students would sit in front of slightly updated versions of the same machines and write computer programs: these used dot-matrix printing mechanisms, but were (from the computer's point of view) identical to the old teletypes. By that point, computers were better at time-sharing—that is, mainframes were still mainframes, but they were better at communicating with a large number of terminals at once. Consequently, it was no longer necessary to use batch processing. Card readers were shoved out into hallways and boiler rooms, and batch processing became a nerds-only kind of thing, and consequently took on a certain eldritch flavor among those of us who even knew it existed. We were all off the Batch, and on the Command Line, interface now—my very first shift in operating system paradigms, if only I'd known it.

A huge stack of accordion-fold paper sat on the floor underneath each one of these glorified teletypes, and miles of paper shuddered through their platens. Almost all of this paper was thrown away or recycled without ever having been touched by ink—an ecological atrocity so glaring that those machines were soon replaced by video terminals—so-called “glass teletypes”—which were quieter and didn't waste paper. Again, though, from the computer's point of view these were indistinguishable from World War II-era teletype machines. In effect we still used Victorian technology to communicate with computers until about 1984, when the Macintosh was introduced with its Graphical User Interface. Even after that, the Command Line continued to exist as an underlying stratum—a sort of brainstem reflex—of many modern computer systems all through the heyday of Graphical User Interfaces, or GUIs as I will call them from now on.

GUIs

Now the first job that any coder needs to do when writing a new piece of software is to figure out how to take the information that is being worked with (in a graphics program, an image; in a spreadsheet, a grid of numbers) and turn it into a linear string of bytes. These strings of bytes are commonly called files or (somewhat more hiply) streams. They are to telegrams what modern humans are to Cro-Magnon man, which is to say the same thing under a different name. All that you see on your computer screen—your Tomb Raider, your digitized voice mail messages, faxes, and word processing documents written in thirty-seven different typefaces—is still, from the computer's point of view, just like telegrams, except much longer, and demanding of more arithmetic.

The quickest way to get a taste of this is to fire up your web browser, visit a site, and then select the View/Document Source menu item. You will get a bunch of computer code that looks something like this:

```
<HTML>
<HEAD>
<TITLE> C R Y P T O N O M I C O N</TITLE>
</HEAD>
<BODY BGCOLOR="#000000" LINK="#996600" ALINK="#FFFFFF" VLINK="#663300">
<MAP NAME="navtext">
<AREA SHAPE=RECT HREF="praise.html" COORDS="0,37,84,55">
<AREA SHAPE=RECT HREF="author.html" COORDS="0,59,137,75">
<AREA SHAPE=RECT HREF="text.html" COORDS="0,81,101,96">
```

```

<AREA SHAPE=RECT HREF="tour.html" COORDS="0,100,121,117">
<AREA SHAPE=RECT HREF="order.html" COORDS="0,122,143,138">
<AREA SHAPE=RECT HREF="beginning.html" COORDS="0,140,213,157">
</MAP>
<CENTER>
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" WIDTH="520">
<TR>
  <TD VALIGN=TOP ROWSPAN="5">
<IMG SRC="images/spacer.gif" WIDTH="30" HEIGHT="1" BORDER="0">
</TD>
  <TD VALIGN=TOP COLSPAN="2">
<IMG SRC="images/main_banner.gif" ALT="Cryptonomincon by Neal Stephenson" WIDTH="479" HEIGHT="122"
BORDER="0">
</TD>
</TR>

```

This crud is called HTML (HyperText Markup Language) and it is basically a very simple programming language instructing your web browser how to draw a page on a screen. Anyone can learn HTML and many people do. The important thing is that no matter what splendid multimedia web pages they might represent, HTML files are just telegrams.

When Ronald Reagan was a radio announcer, he used to call baseball games by reading the terse descriptions that trickled in over the telegraph wire and were printed out on a paper tape. He would sit there, all by himself in a padded room with a microphone, and the paper tape would eke out of the machine and crawl over the palm of his hand printed with cryptic abbreviations. If the count went to three and two, Reagan would describe the scene as he saw it in his mind's eye: "The brawny left-hander steps out of the batter's box to wipe the sweat from his brow. The umpire steps forward to sweep the dirt from home plate." and so on. When the cryptogram on the paper tape announced a base hit, he would whack the edge of the table with a pencil, creating a little sound effect, and describe the arc of the ball as if he could actually see it. His listeners, many of whom presumably thought that Reagan was actually at the ballpark watching the game, would reconstruct the scene in their minds according to his descriptions.

This is exactly how the World Wide Web works: the HTML files are the pithy description on the paper tape, and your Web browser is Ronald Reagan. The same is true of Graphical User Interfaces in general.

So an OS is a stack of metaphors and abstractions that stands between you and the telegrams, and embodying various tricks the programmer used to convert the information you're working with—be it images, e-mail messages, movies, or word processing documents—into the necklaces of bytes that are the only things computers know how to work with. When we used actual telegraph equipment (teletypes) or their higher-tech substitutes ("glass teletypes," or the MS-DOS command line) to work with our computers, we were very close to the bottom of that stack. When we use most modern operating systems, though, our interaction with the machine is heavily mediated. Everything we do is interpreted and translated time and again as it works its way down through all of the metaphors and abstractions.

The Macintosh OS was a revolution in both the good and bad senses of that word. Obviously it was true that command line interfaces were not for everyone, and that it would be a good thing to make computers more accessible to a less technical audience—if not for altruistic reasons, then because those sorts of people constituted an incomparably vaster market. It was clear the the Mac's engineers saw a whole new country stretching out before them; you could almost hear them muttering, "Wow! We don't have to be bound by files as linear streams of bytes anymore, *vive la revolution*, let's see how far we can take this!" No command line interface was available on the Macintosh; you talked to it with the mouse, or not at all. This was a statement of sorts, a credential of revolutionary purity. It seemed that the designers of the Mac intended to sweep Command Line Interfaces into the dustbin of history.

My own personal love affair with the Macintosh began in the spring of 1984 in a computer store in Cedar Rapids, Iowa, when a friend of mine—coincidentally, the son of the MGB owner—showed me a Macintosh running MacPaint, the revolutionary drawing program. It ended in July of 1995 when I tried to save a big important file on my Macintosh Powerbook and instead instead of doing so, it annihilated the data so thoroughly that two different disk crash utility programs were unable to find any trace that it had ever existed. During the intervening ten years, I had a passion for the MacOS that seemed righteous and reasonable at the time but in retrospect strikes me as being exactly the same sort of goofy infatuation that my friend's dad had with his car.

The introduction of the Mac triggered a sort of holy war in the computer world. Were GUIs a brilliant design innovation that made computers more human-centered and therefore accessible to the masses, leading us toward an unprecedented revolution in human society, or an insulting bit of audiovisual gimcrackery dreamed up by flaky Bay Area hacker types that stripped computers of their power and flexibility and turned the noble and serious work of computing into a childish video game?

This debate actually seems more interesting to me today than it did in the mid-1980s. But people more or less stopped debating it when Microsoft endorsed the idea of GUIs by coming out with the first Windows. At this point, command-line partisans were relegated to the status of silly old grouches, and a new conflict was touched off, between users of MacOS and users of Windows.

There was plenty to argue about. The first Macintoshes looked different from other PCs even when they were turned off: they consisted of one box containing both CPU (the part of the computer that does arithmetic on bits) and monitor screen. This was billed, at the time, as a philosophical statement of sorts: Apple wanted to make the personal computer into an appliance, like a toaster. But it also reflected the purely technical demands of running a graphical user interface. In a GUI machine, the chips that draw things on the screen have to be integrated with the computer's central processing unit, or CPU, to a far greater extent than is the case with command-line interfaces, which until recently didn't even know that they weren't just talking to teletypes.

This distinction was of a technical and abstract nature, but it became clearer when the machine crashed (it is commonly the case with technologies that you can get the best insight about how they work by watching them fail). When everything went to hell and the CPU began spewing out random bits, the result, on a CLI machine, was lines and lines of perfectly formed but random characters on the screen—known to cognoscenti as “going Cyrillic.” But to the MacOS, the screen was not a teletype, but a place to put graphics; the image on the screen was a bitmap, a literal rendering of the contents of a particular portion of the computer's memory. When the computer crashed and wrote gibberish into the bitmap, the result was something that looked vaguely like static on a broken television set—a “snow crash.”

And even after the introduction of Windows, the underlying differences endured; when a Windows machine got into trouble, the old command-line interface would fall down over the GUI like an asbestos fire curtain sealing off the proscenium of a burning opera. When a Macintosh got into trouble it presented you with a cartoon of a bomb, which was funny the first time you saw it.

And these were by no means superficial differences. The reversion of Windows to a CLI when it was in distress proved to Mac partisans that Windows was nothing more than a cheap facade, like a garish afghan flung over a rotted-out sofa. They were disturbed and annoyed by the sense that lurking underneath Windows' ostensibly user-friendly interface was—literally—a subtext.

For their part, Windows fans might have made the sour observation that all computers, even Macintoshes, were built on that same subtext, and that the refusal of Mac owners to admit that fact to themselves seemed to signal a willingness, almost an eagerness, to be duped.

Anyway, a Macintosh had to switch individual bits in the memory chips on the video card, and it had to do it very fast, and in arbitrarily complicated patterns. Nowadays this is cheap and easy, but in the technological regime that prevailed in the early 1980s, the only realistic way to do it was to build the motherboard (which

contained the CPU) and the video system (which contained the memory that was mapped onto the screen) as a tightly integrated whole—hence the single, hermetically sealed case that made the Macintosh so distinctive.

When Windows came out, it was conspicuous for its ugliness, and its current successors, Windows 95 and Windows NT, are not things that people would pay money to look at either. Microsoft's complete disregard for aesthetics gave all of us Mac-lovers plenty of opportunities to look down our noses at them. That Windows looked an awful lot like a direct ripoff of MacOS gave us a burning sense of moral outrage to go with it. Among people who really knew and appreciated computers (hackers, in Steven Levy's non-pejorative sense of that word) and in a few other niches such as professional musicians, graphic artists and schoolteachers, the Macintosh, for a while, was simply the computer. It was seen as not only a superb piece of engineering, but an embodiment of certain ideals about the use of technology to benefit mankind, while Windows was seen as a pathetically clumsy imitation and a sinister world domination plot rolled into one. So very early, a pattern had been established that endures to this day: people dislike Microsoft, which is okay; but they dislike it for reasons that are poorly considered, and in the end, self-defeating.

[. . .]

Contemporary culture is a two-tiered system, like the Morlocks and the Eloi in H.G. Wells's *The Time Machine*, except that it's been turned upside down. In *The Time Machine* the Eloi were an effete upper class, supported by lots of subterranean Morlocks who kept the technological wheels turning. But in our world it's the other way round. The Morlocks are in the minority, and they are running the show, because they understand how everything works. The much more numerous Eloi learn everything they know from being steeped from birth in electronic media directed and controlled by book-reading Morlocks. So many ignorant people could be dangerous if they got pointed in the wrong direction, and so we've evolved a popular culture that is (a) almost unbelievably infectious and (b) neuters every person who gets infected by it, by rendering them unwilling to make judgments and incapable of taking stands.

Morlocks, who have the energy and intelligence to comprehend details, go out and master complex subjects and produce Disney-like Sensorial Interfaces so that Eloi can get the gist without having to strain their minds or endure boredom. Those Morlocks will go to India and tediously explore a hundred ruins, then come home and built sanitary bug-free versions: highlight films, as it were. This costs a lot, because Morlocks insist on good coffee and first-class airline tickets, but that's no problem because Eloi like to be dazzled and will gladly pay for it all.

Now I realize that most of this probably sounds snide and bitter to the point of absurdity: your basic snotty intellectual throwing a tantrum about those unlettered philistines. As if I were a self-styled Moses, coming down from the mountain all alone, carrying the stone tablets bearing the Ten Commandments carved in immutable stone—the original command-line interface—and blowing his stack at the weak, unenlightened Hebrews worshipping images. Not only that, but it sounds like I'm pumping some sort of conspiracy theory.

But that is not where I'm going with this. The situation I describe, here, could be bad, but doesn't have to be bad and isn't necessarily bad now:

It simply is the case that we are way too busy, nowadays, to comprehend everything in detail. And it's better to comprehend it dimly, through an interface, than not at all. Better for ten million Eloi to go on the Kilimanjaro Safari at Disney World than for a thousand cardiovascular surgeons and mutual fund managers to go on "real" ones in Kenya. The boundary between these two classes is more porous than I've made it sound. I'm always running into regular dudes—construction workers, auto mechanics, taxi drivers, galoots in general—who were largely aliterate until something made it necessary for them to become readers and start actually thinking about things. Perhaps they had to come to grips with alcoholism, perhaps they got sent to jail, or came down with a disease, or suffered a crisis in religious faith, or simply got bored. Such people can get up to speed on particular subjects quite rapidly. Sometimes their lack of a broad education makes them over-apt to go off on intellectual wild goose chases, but, hey, at least a wild goose chase gives you some exercise. The spectre of a

polity controlled by the fads and whims of voters who actually believe that there are significant differences between Bud Lite and Miller Lite, and who think that professional wrestling is for real, is naturally alarming to people who don't. But then countries controlled via the command-line interface, as it were, by double-domed intellectuals, be they religious or secular, are generally miserable places to live. Sophisticated people deride Disneyesque entertainments as pat and saccharine, but, hey, if the result of that is to instill basically warm and sympathetic reflexes, at a preverbal level, into hundreds of millions of unlettered media-steepers, then how bad can it be? We killed a lobster in our kitchen last night and my daughter cried for an hour. The Japanese, who used to be just about the fiercest people on earth, have become infatuated with cuddly adorable cartoon characters. My own family—the people I know best—is divided about evenly between people who will probably read this essay and people who almost certainly won't, and I can't say for sure that one group is necessarily warmer, happier, or better-adjusted than the other.

MORLOCKS AND ELOI AT THE KEYBOARD

Back in the days of the command-line interface, users were all Morlocks who had to convert their thoughts into alphanumeric symbols and type them in, a grindingly tedious process that stripped away all ambiguity, laid bare all hidden assumptions, and cruelly punished laziness and imprecision. Then the interface-makers went to work on their GUIs, and introduced a new semiotic layer between people and machines. People who use such systems have abdicated the responsibility, and surrendered the power, of sending bits directly to the chip that's doing the arithmetic, and handed that responsibility and power over to the OS. This is tempting because giving clear instructions, to anyone or anything, is difficult. We cannot do it without thinking, and depending on the complexity of the situation, we may have to think hard about abstract things, and consider any number of ramifications, in order to do a good job of it. For most of us, this is hard work. We want things to be easier. How badly we want it can be measured by the size of Bill Gates's fortune.

The OS has (therefore) become a sort of intellectual labor-saving device that tries to translate humans' vaguely expressed intentions into bits. In effect we are asking our computers to shoulder responsibilities that have always been considered the province of human beings—we want them to understand our desires, to anticipate our needs, to foresee consequences, to make connections, to handle routine chores without being asked, to remind us of what we ought to be reminded of while filtering out noise.

At the upper (which is to say, closer to the user) levels, this is done through a set of conventions—menus, buttons, and so on. These work in the sense that analogies work: they help Eloi understand abstract or unfamiliar concepts by likening them to something known. But the loftier word “metaphor” is used.

The overarching concept of the MacOS was the “desktop metaphor” and it subsumed any number of lesser (and frequently conflicting, or at least mixed) metaphors. Under a GUI, a file (frequently called “document”) is metaphrased as a window on the screen (which is called a “desktop”). The window is almost always too small to contain the document and so you “move around,” or, more pretentiously, “navigate” in the document by “clicking and dragging” the “thumb” on the “scroll bar.” When you “type” (using a keyboard) or “draw” (using a “mouse”) into the “window” or use pull-down “menus” and “dialog boxes” to manipulate its contents, the results of your labors get stored (at least in theory) in a “file,” and later you can pull the same information back up into another “window.” When you don't want it anymore, you “drag” it into the “trash.”

There is massively promiscuous metaphor-mixing going on here, and I could deconstruct it 'til the cows come home, but I won't. Consider only one word: “document.” When we document something in the real world, we make fixed, permanent, immutable records of it. But computer documents are volatile, ephemeral constellations of data. Sometimes (as when you've just opened or saved them) the document as portrayed in the window is identical to what is stored, under the same name, in a file on the disk, but other times (as when you have made changes without saving them) it is completely different. In any case, every time you hit “Save” you annihilate the previous version of the “document” and replace it with whatever happens to be in the window at

the moment. So even the word “save” is being used in a sense that is grotesquely misleading—“destroy one version, save another” would be more accurate.

Anyone who uses a word processor for very long inevitably has the experience of putting hours of work into a long document and then losing it because the computer crashes or the power goes out. Until the moment that it disappears from the screen, the document seems every bit as solid and real as if it had been typed out in ink on paper. But in the next moment, without warning, it is completely and irretrievably gone, as if it had never existed. The user is left with a feeling of disorientation (to say nothing of annoyance) stemming from a kind of metaphor shear—you realize that you’ve been living and thinking inside of a metaphor that is essentially bogus.

So GUIs use metaphors to make computing easier, but they are bad metaphors. Learning to use them is essentially a word game, a process of learning new definitions of words like “window” and “document” and “save” that are different from, and in many cases almost diametrically opposed to, the old. Somewhat improbably, this has worked very well, at least from a commercial standpoint, which is to say that Apple/Microsoft have made a lot of money off of it. All of the other modern operating systems have learned that in order to be accepted by users they must conceal their underlying gutwork beneath the same sort of spackle. This has some advantages: if you know how to use one GUI operating system, you can probably work out how to use any other in a few minutes. Everything works a little differently, like European plumbing—but with some fiddling around, you can type a memo or surf the web.

Most people who shop for OSes (if they bother to shop at all) are comparing not the underlying functions but the superficial look and feel. The average buyer of an OS is not really paying for, and is not especially interested in, the low-level code that allocates memory or writes bytes onto the disk. What we’re really buying is a system of metaphors. And—much more important—what we’re buying into is the underlying assumption that metaphors are a good way to deal with the world.

Recently a lot of new hardware has become available that gives computers numerous interesting ways of affecting the real world: making paper spew out of printers, causing words to appear on screens thousands of miles away, shooting beams of radiation through cancer patients, creating realistic moving pictures of the Titanic. Windows is now used as an OS for cash registers and bank tellers’ terminals. My satellite TV system uses a sort of GUI to change channels and show program guides. Modern cellular telephones have a crude GUI built into a tiny LCD screen. Even Legos now have a GUI: you can buy a Lego set called Mindstorms that enables you to build little Lego robots and program them through a GUI on your computer.

So we are now asking the GUI to do a lot more than serve as a glorified typewriter. Now we want to become a generalized tool for dealing with reality. This has become a bonanza for companies that make a living out of bringing new technology to the mass market.

Obviously you cannot sell a complicated technological system to people without some sort of interface that enables them to use it. The internal combustion engine was a technological marvel in its day, but useless as a consumer good until a clutch, transmission, steering wheel and throttle were connected to it. That odd collection of gizmos, which survives to this day in every car on the road, made up what we would today call a user interface. But if cars had been invented after Macintoshes, carmakers would not have bothered to gin up all of these arcane devices. We would have a computer screen instead of a dashboard, and a mouse (or at best a joystick) instead of a steering wheel, and we’d shift gears by pulling down a menu:

PARK — REVERSE — NEUTRAL — 3 2 1 — Help...

A few lines of computer code can thus be made to substitute for any imaginable mechanical interface. The problem is that in many cases the substitute is a poor one. Driving a car through a GUI would be a miserable experience. Even if the GUI were perfectly bug-free, it would be incredibly dangerous, because menus and buttons simply can’t be as responsive as direct mechanical controls. My friend’s dad, the gentleman who was restor-

ing the MGB, never would have bothered with it if it had been equipped with a GUI. It wouldn't have been any fun.

The steering wheel and gearshift lever were invented during an era when the most complicated technology in most homes was a butter churn. Those early carmakers were simply lucky, in that they could dream up whatever interface was best suited to the task of driving an automobile, and people would learn it. Likewise with the dial telephone and the AM radio. By the time of the Second World War, most people knew several interfaces: they could not only churn butter but also drive a car, dial a telephone, turn on a radio, summon flame from a cigarette lighter, and change a light bulb.

But now every little thing—wristwatches, VCRs, stoves—is jammed with features, and every feature is useless without an interface. If you are like me, and like most other consumers, you have never used ninety percent of the available features on your microwave oven, VCR, or cellphone. You don't even know that these features exist. The small benefit they might bring you is outweighed by the sheer hassle of having to learn about them. This has got to be a big problem for makers of consumer goods, because they can't compete without offering features.

It's no longer acceptable for engineers to invent a wholly novel user interface for every new product, as they did in the case of the automobile, partly because it's too expensive and partly because ordinary people can only learn so much. If the VCR had been invented a hundred years ago, it would have come with a thumbwheel to adjust the tracking and a gearshift to change between forward and reverse and a big cast-iron handle to load or to eject the cassettes. It would have had a big analog clock on the front of it, and you would have set the time by moving the hands around on the dial. But because the VCR was invented when it was—during a sort of awkward transitional period between the era of mechanical interfaces and GUIs—it just had a bunch of push-buttons on the front, and in order to set the time you had to push the buttons in just the right way. This must have seemed reasonable enough to the engineers responsible for it, but to many users it was simply impossible. Thus the famous blinking 12:00 that appears on so many VCRs. Computer people call this “the blinking twelve problem”. When they talk about it, though, they usually aren't talking about VCRs.

Modern VCRs usually have some kind of on-screen programming, which means that you can set the time and control other features through a sort of primitive GUI. GUIs have virtual pushbuttons too, of course, but they also have other types of virtual controls, like radio buttons, checkboxes, text entry boxes, dials, and scroll-bars. Interfaces made out of these components seem to be a lot easier, for many people, than pushing those little buttons on the front of the machine, and so the blinking 12:00 itself is slowly disappearing from America's living rooms. The blinking twelve problem has moved on to plague other technologies.

So the GUI has gone beyond being an interface to personal computers, and become a sort of meta-interface that is pressed into service for every new piece of consumer technology. It is rarely an ideal fit, but having an ideal, or even a good interface is no longer the priority; the important thing now is having some kind of interface that customers will actually use, so that manufacturers can claim, with a straight face, that they are offering new features.

We want GUIs largely because they are convenient and because they are easy— or at least the GUI makes it seem that way. Of course, nothing is really easy and simple, and putting a nice interface on top of it does not change that fact. A car controlled through a GUI would be easier to drive than one controlled through pedals and steering wheel, but it would be incredibly dangerous.

By using GUIs all the time we have insensibly bought into a premise that few people would have accepted if it were presented to them bluntly: namely, that hard things can be made easy, and complicated things simple, by putting the right interface on them. In order to understand how bizarre this is, imagine that book reviews were written according to the same values system that we apply to user interfaces: “The writing in this book is marvelously simple-minded and glib; the author glosses over complicated subjects and employs facile generalizations in almost every sentence. Readers rarely have to think, and are spared all of the difficulty and tedium

typically involved in reading old-fashioned books.” As long as we stick to simple operations like setting the clocks on our VCRs, this is not so bad. But as we try to do more ambitious things with our technologies, we inevitably run into the problem of:

METAPHOR SHEAR

I began using Microsoft Word as soon as the first version was released around 1985. After some initial hassles I found it to be a better tool than MacWrite, which was its only competition at the time. I wrote a lot of stuff in early versions of Word, storing it all on floppies, and transferred the contents of all my floppies to my first hard drive, which I acquired around 1987. As new versions of Word came out I faithfully upgraded, reasoning that as a writer it made sense for me to spend a certain amount of money on tools.

Sometime in the mid-1980’s I attempted to open one of my old, circa-1985 Word documents using the version of Word then current: 6.0 It didn’t work. Word 6.0 did not recognize a document created by an earlier version of itself. By opening it as a text file, I was able to recover the sequences of letters that made up the text of the document. My words were still there. But the formatting had been run through a log chipper—the words I’d written were interrupted by spates of empty rectangular boxes and gibberish.

Now, in the context of a business (the chief market for Word) this sort of thing is only an annoyance—one of the routine hassles that go along with using computers. It’s easy to buy little file converter programs that will take care of this problem. But if you are a writer whose career is words, whose professional identity is a corpus of written documents, this kind of thing is extremely disquieting. There are very few fixed assumptions in my line of work, but one of them is that once you have written a word, it is written, and cannot be unwritten. The ink stains the paper, the chisel cuts the stone, the stylus marks the clay, and something has irrevocably happened (my brother-in-law is a theologian who reads 3250-year-old cuneiform tablets—he can recognize the handwriting of particular scribes, and identify them by name). But word-processing software—particularly the sort that employs special, complex file formats—has the eldritch power to unwrite things. A small change in file formats, or a few twiddled bits, and months’ or years’ literary output can cease to exist.

Now this was technically a fault in the application (Word 6.0 for the Macintosh) not the operating system (MacOS 7 point something) and so the initial target of my annoyance was the people who were responsible for Word. But. On the other hand, I could have chosen the “save as text” option in Word and saved all of my documents as simple telegrams, and this problem would not have arisen. Instead I had allowed myself to be seduced by all of those flashy formatting options that hadn’t even existed until GUIs had come along to make them practicable. I had gotten into the habit of using them to make my documents look pretty (perhaps prettier than they deserved to look; all of the old documents on those floppies turned out to be more or less crap). Now I was paying the price for that self-indulgence. Technology had moved on and found ways to make my documents look even prettier, and the consequence of it was that all old ugly documents had ceased to exist.

It was—if you’ll pardon me for a moment’s strange little fantasy—as if I’d gone to stay at some resort, some exquisitely designed and art-directed hotel, placing myself in the hands of past masters of the Sensorial Interface, and had sat down in my room and written a story in ballpoint pen on a yellow legal pad, and when I returned from dinner, discovered that the maid had taken my work away and left behind in its place a quill pen and a stack of fine parchment—explaining that the room looked ever so much finer this way, and it was all part of a routine upgrade. But written on these sheets of paper, in flawless penmanship, were long sequences of words chosen at random from the dictionary. Appalling, sure, but I couldn’t really lodge a complaint with the management, because by staying at this resort I had given my consent to it. I had surrendered my Morlock credentials and become an Eloi.

[. . .]