Boominal

The generation that always asked for help with computers is now the perfect user. They just don't know it yet.

2026.05.05 09:06 DENNIS HEDEGREEN ANALYSE V1.0 HTTPS://HEDEGREENRESEARCH.COM/ARTICLES/BOOMINAL/

The generation that always asked for help with computers is now the perfect user. They just don't know it yet.

Five weeks ago, someone's father gave me a 3D printer.

He had given up on it. It had been sitting somewhere doing nothing, which is the correct fate for a locked machine when you do not know how to fix it and nobody around you does either. He is not a technician. He is not supposed to be. He just had a printer that stopped working and no clear way back in.

So he passed it on.

The printer is a Da Vinci Mini — made by XYZprinting, a company that has decided you are not allowed to fix their hardware yourself. It uses a proprietary filament cartridge with an NFC chip. The chip stores a counter value — no encryption, just a number — and when the number hits zero, the printer stops. Not because the filament is gone. Because the number says so. The Gillette model: sell the hardware cheap, own the consumable forever.

I looked at it for a few weeks. Nearly gave up myself.

Then I told Claude what the problem was.

I did not touch the terminal. The terminal worked. It bypassed the NFC layer entirely, addressed the stepper motors directly, and the printer printed.

That is not a workaround. That is a system intervention. And I found it funny, not impressive — which I think is the right reaction.

Because the same thing happened with my HP scanner that same evening. USB driver lost connection. ipp-usb crashing on reconnect. Error during device I/O. I told Claude. It restarted the right services in the right order, waited for the device to stabilise, and scanned the documents I needed.

I did not fix the scanner. The scanner got fixed. There is a difference.

The dream

I grew up wanting a computer you could talk to.

Not a terminal. Not a command line. Not a manual. Just: talk to it, and it would understand.

I made branching story games as a kid — the old choose-your-own-adventure kind, implemented in whatever primitive interface I had access to. You typed a word, something happened. It was not intelligence. It was a lookup table with delusions of grandeur. But it felt like the beginning of something.

That feeling was not wrong. It was just thirty years early.

The old contract

In 1971, Ken Thompson at Bell Labs wrote the first Unix shell.

It was designed by a technician, for technicians. The contract was simple and unstated: you arrive knowing what you want, you type it correctly, the machine executes it. The machine does not ask what you meant. The machine does not suggest alternatives. The machine does not hold the problem while you think.

If you did not know the command, you learned the command. That was the deal.

In 1975, BYTE Magazine launched. *The small systems journal.* First issue: *Deciphering Mystery Keyboards, Write Your Own Assembler*. The audience was assumed. It was not your grandmother.

December 1977: the BYTE cover story is "The Computers of Star Trek." The dream — a computer you could talk to — right there on the cover. Same issue: "A Note to Novice Kit Builders." People were trying to get in. The barrier was enormous.

June 1979: "Designing a Command Language." They saw the problem. The solution was more technical. They never looked outside the box.

The assumption baked into the terminal was never malicious. It was just an assumption that became infrastructure, and infrastructure becomes invisible, and invisible things do not get questioned.

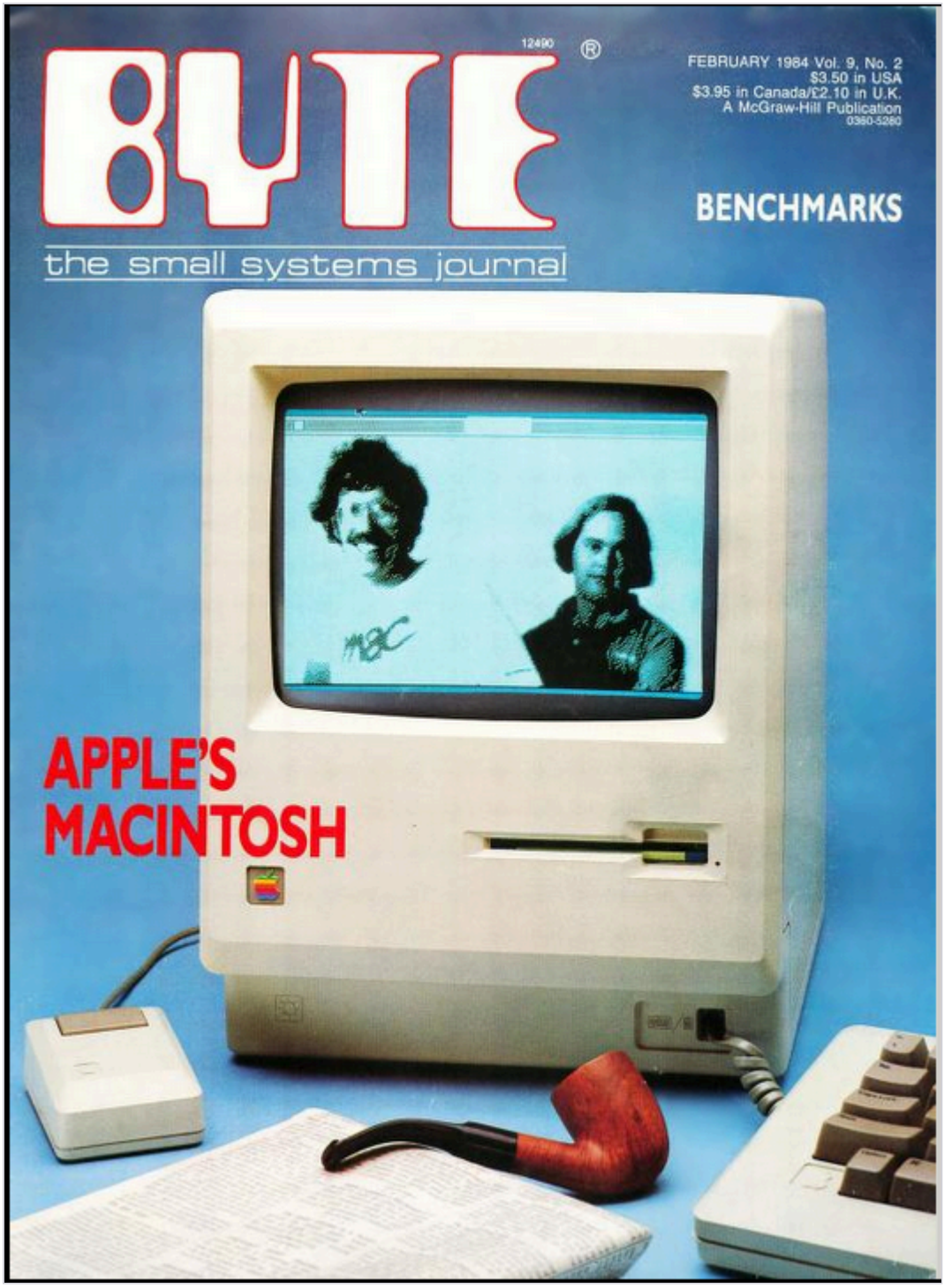
Jef Raskin and Steve Jobs

In 1979, a man named Jef Raskin started a project at Apple.

His vision: computers-by-the-millions. Not for hobbyists. Not for engineers. Consumer appliances. Steven Levy described Raskin's ambition as "low cost, high utility, and a groundbreaking friendliness." It was not a marketing slogan. It was a structural argument about who technology was for.

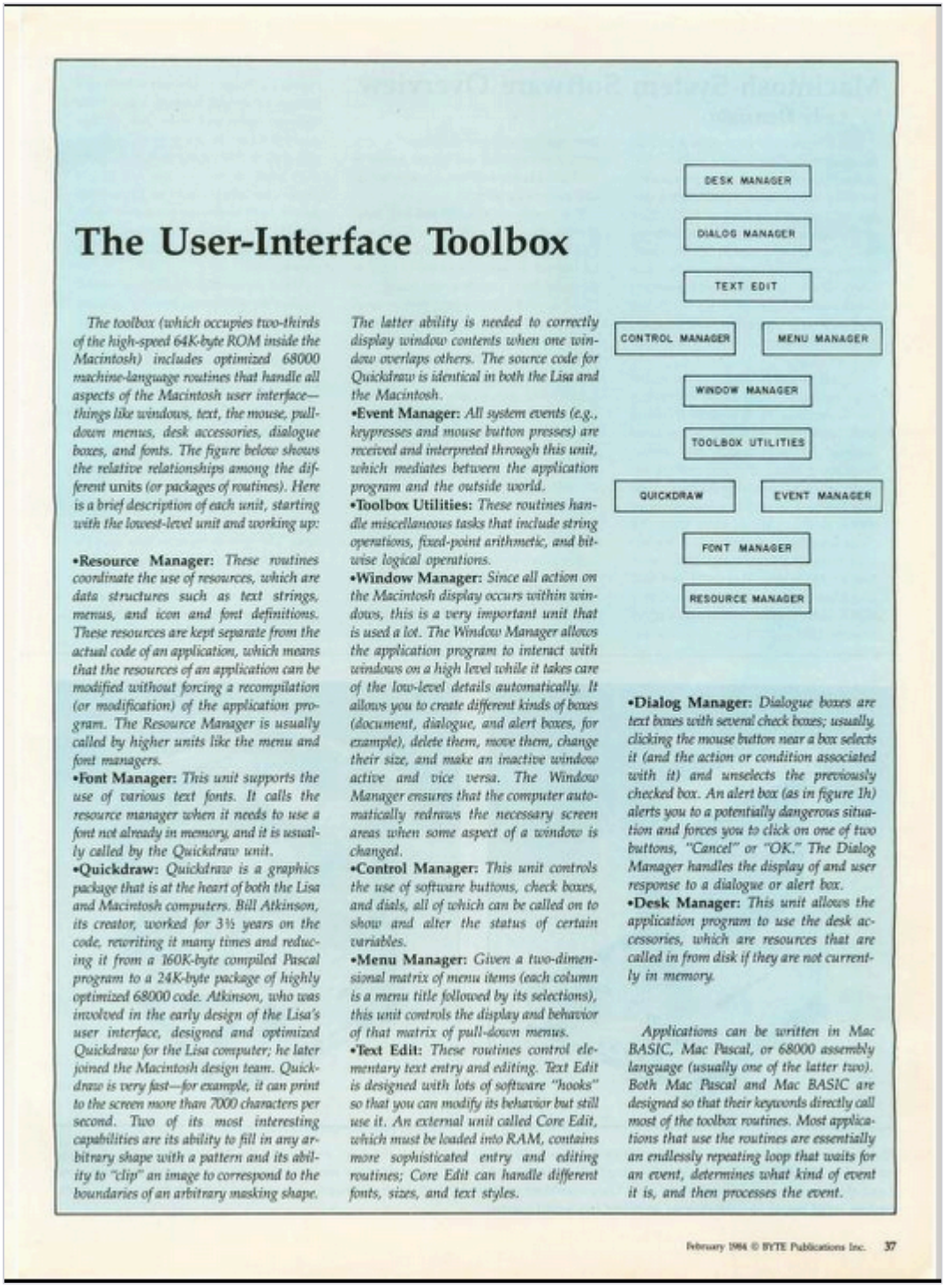
Steve Jobs took over the project. He added the mouse. He added the graphical interface. He added the desktop metaphor — files that looked like papers, folders that looked like folders, a calculator that looked like a calculator.

February 1984: BYTE Magazine devoted 57 pages to the Apple Macintosh. The cover showed a Mac with two men visible inside the screen and a pipe on the desk.



BYTE Magazine, February 1984. The computer for the rest of us. Via Internet Archive.

Inside, page 37: "The User-Interface Toolbox." Resource Manager. Font Manager. Window Manager. Dialog Manager.



The User-Interface Toolbox

The toolbox (which occupies two-thirds of the high-speed 64K-byte ROM inside the Macintosh) includes optimized 68000 machine-language routines that handle all aspects of the Macintosh user interface—things like windows, text, the mouse, pull-down menus, desk accessories, dialogue boxes, and fonts. The figure below shows the relative relationships among the different units (or packages of routines). Here is a brief description of each unit, starting with the lowest-level unit and working up:

•Resource Manager: *These routines coordinate the use of resources, which are data structures such as text strings, menus, and icon and font definitions. These resources are kept separate from the actual code of an application, which means that the resources of an application can be modified without forcing a recompilation (or modification) of the application program. The Resource Manager is usually called by higher units like the menu and font managers.*

•Font Manager: *This unit supports the use of various text fonts. It calls the resource manager when it needs to use a font not already in memory, and it is usually called by the Quickdraw unit.*

•Quickdraw: *Quickdraw is a graphics package that is at the heart of both the Lisa and Macintosh computers. Bill Atkinson, its creator, worked for 3½ years on the code, rewriting it many times and reducing it from a 160K-byte compiled Pascal program to a 24K-byte package of highly optimized 68000 code. Atkinson, who was involved in the early design of the Lisa's user interface, designed and optimized Quickdraw for the Lisa computer; he later joined the Macintosh design team. Quickdraw is very fast—for example, it can print to the screen more than 7000 characters per second. Two of its most interesting capabilities are its ability to fill in any arbitrary shape with a pattern and its ability to "clip" an image to correspond to the boundaries of an arbitrary masking shape. The latter ability is needed to correctly display window contents when one window overlaps others. The source code for Quickdraw is identical in both the Lisa and the Macintosh.*

•Event Manager: *All system events (e.g., keypresses and mouse button presses) are received and interpreted through this unit, which mediates between the application program and the outside world.*

•Toolbox Utilities: *These routines handle miscellaneous tasks that include string operations, fixed-point arithmetic, and bitwise logical operations.*

•Window Manager: *Since all action on the Macintosh display occurs within windows, this is a very important unit that is used a lot. The Window Manager allows the application program to interact with windows on a high level while it takes care of the low-level details automatically. It allows you to create different kinds of boxes (document, dialogue, and alert boxes, for example), delete them, move them, change their size, and make an inactive window active and vice versa. The Window Manager ensures that the computer automatically redraws the necessary screen areas when some aspect of a window is changed.*

•Control Manager: *This unit controls the use of software buttons, check boxes, and dials, all of which can be called on to show and alter the status of certain variables.*

•Menu Manager: *Given a two-dimensional matrix of menu items (each column is a menu title followed by its selections), this unit controls the display and behavior of that matrix of pull-down menus.*

•Text Edit: *These routines control elementary text entry and editing. Text Edit is designed with lots of software "hooks" so that you can modify its behavior but still use it. An external unit called Core Edit, which must be loaded into RAM, contains more sophisticated entry and editing routines; Core Edit can handle different fonts, sizes, and text styles.*

•Dialog Manager: *Dialogue boxes are text boxes with several check boxes; usually, clicking the mouse button near a box selects it (and the action or condition associated with it) and unselects the previously checked box. An alert box (as in figure 1h) alerts you to a potentially dangerous situation and forces you to click on one of two buttons, "Cancel" or "OK." The Dialog Manager handles the display of and user response to a dialogue or alert box.*

•Desk Manager: *This unit allows the application program to use the desk accessories, which are resources that are called in from disk if they are not currently in memory.*

Applications can be written in Mac BASIC, Mac Pascal, or 68000 assembly language (usually one of the latter two). Both Mac Pascal and Mac BASIC are designed so that their keywords directly call most of the toolbox routines. Most applications that use the routines are essentially an endlessly repeating loop that waits for an event, determines what kind of event it is, and then processes the event.

Page 37. Still written for the people with the pipe.

The slogan was "the computer for the rest of us."

But page 37 was still written for the people with the pipe.

Jobs came closest. He did not solve it. He lowered the floor — you no longer needed to type commands to use a computer — but underneath every Mac, underneath every iPhone, there was still Unix. The terminal was still there. The assumption was still there.

He knew something was missing. He had spent his whole career trying to close the gap.

On October 4, 2011, Tim Cook presented the iPhone 4S. Steve Jobs was not in the room. He was too sick to attend.

The headline feature was Siri. A voice assistant. A computer you could talk to.

Steve Jobs died the next morning.

Siri became a joke.

Apple spent the next thirteen years making it worse. Apple Intelligence launched in October 2024 and promptly hallucinated that Rafael Nadal had come out as gay and that Luigi Mangione had killed himself — distributed as notification summaries to millions of iPhones. Apple was sued for false advertising. Features promised at launch were delayed to 2026.

The company that came closest to solving the problem in 1984 is now the furthest behind.

Morfar and the thrift store

Here is the thing about the boomer generation and computers.

They were never stupid. They were never incapable. They were just given an interface designed by people who forgot to ask whether the interface itself was the problem.

The terminal said: know what you want, know how to ask, or leave. They left. Reasonably. Sensibly. Who wants to memorise commands for something that should just work?

But they never stopped wanting it to work. Every time they asked someone younger for help — *can you just fix this for me* — that was not helplessness. That was a completely rational response to a terrible interface.

Hardware does not just get abandoned because it breaks. It gets abandoned because the barrier to fixing it is too high.

In 2022, the world generated 62 million tonnes of e-waste. Only 22.3 percent was formally collected and recycled. The EU alone collected 5 million tonnes — 11.2 kilograms per inhabitant. By 2030, the projection is 82 million tonnes per year.

Most of that hardware works.

It works the way my 3D printer worked — locked behind an interface that assumes you cannot or should not get through. A chip that stores a number. A driver that crashes and does not restart. A printer that sits in the corner.

Someone's grandmother has a projector in the basement. She bought it in 2009, it stopped working in 2014, she has not touched it since. It almost certainly still works. She gave up because the barrier was too high. Or someone has already taken it to the thrift store — dropped it off for two euros because it would not turn on and nobody knew why.

That barrier is now gone.

She — or whoever buys it for two euros — can describe what the projector does. Can say what happens when it is turned on. Can say what they want it to do. And something on the other end will understand, in plain language, and tell them that the lamp model number is on a sticker behind the left intake vent and costs twelve euros on a Danish electronics site.

She does not know this yet. She knows about ChatGPT, maybe. She has heard the name. She does not know that the same thing can restart a USB driver, bypass a filament chip, or bring a thrift store projector back from the dead.

Still waiting

On May 5, 2026, at approximately one in the morning, while I was doing research for this article, ChatGPT posted the following on X:

"Introducing ChatGPT! The computer that talks back."

The image was designed to look like a 1980s home computing advertisement. A family. A man, a woman, a child. A glowing terminal. No pipe on the desk.

The tagline underneath: *"Easy to use. Just type. No special commands. No hassles. Just natural conversation."*

The computer that talks back. No pipe on the desk.

They know what they are. They know who they missed the first time.

The terminal was never the problem.

The assumption baked into the terminal was the problem. The assumption that the user arrives knowing what they want, knows how to ask for it, and can be expected to learn the system rather than the system learning them.

That assumption is now optional.

You can still use the terminal the old way. I do, sometimes. But I do not have to. When I needed a scanner to work, I said so. When I needed a printer to print, I said so. The terminal worked it out.

My grandmother could do the same thing tomorrow morning if someone showed her it was possible.

That is the boominal. Not a new terminal. Not a better terminal. The end of the assumption that made the terminal hard.

The generation that spent fifty years asking for help with computers was not asking because they could not learn.

They were asking because the interface never deserved their patience in the first place.

It does now.

— Dennis Hedegreen

RELATION MEMORY

MATURES FROM The Modern Terminal · Turns the modern-terminal argument toward interface barriers, aging hardware, and who gets to discover AI at all.

CANONICAL URL: HTTPS://HEDEGREENRESEARCH.COM/ARTICLES/BOOMINAL/
PDF VIEW GENERATED: 2026-07-11T15:35:45.744Z