

The Old Limitations No Longer Apply — Part 1: The Price of Admission

By Austen Tucker · 2026

A Singularity Log Special · Part 1 of 4

In this series

1. The Price of Admission — this essay
2. The Rentier's Last Stand
3. What Happens When We Can Make Our Own?
4. What's Left to Build?

Simple pencil outlines of a desk with a central computer, a carved bird and game pieces at left, and a guitar at right; behind it, people work in a workshop and garden.

A Raspberry Pi 500 sits in my entertainment center. It's a computer built into a keyboard, and for the past few days I've been turning it into RetroMachine: a homebrew system that plays games and manages media the way I want. With AI assistance, the project races ahead. I think of another feature, ask for it, and get back to building.

The Keeper's Hall, with large catalog and recommendation choices above persistent TV controls.

RetroMachine's Hall provides entry points for browsing, asking the Keeper, and managing the TV queue, all from a user's phone.

Tonight, I wanted to tell ChatGPT to turn on the television and play a YouTube playlist. I'd been exposing the machine's capabilities through MCP, a protocol that lets AI assistants interact with external systems. This seemed like the next step. I asked. ChatGPT looked for the connection, found it wasn't available in the current session, and explained what needed fixing. The television stayed off.

RetroMachine's TV queue shows a Turn on TV button, ten queued songs, and a Ready to play status.

And I sat there marveling, because it hadn't occurred to me that I was asking for something extraordinary. I'm in my living room, telling an artificial intelligence to operate a computer I've built into my entertainment center so the computer can operate my television. The system isn't finished, but its components are

accessible and within my ability to assemble. What once sounded like science fiction has become an annoying configuration problem.

In the past, I would have ponied up for an Apple TV. Now, I can write my own in my pajamas in an afternoon.

For the first time, I looked at the unfinished arrangement and expected to make it work. I didn't dismiss the idea because I lacked the time, money, or knowledge to attempt it.

The Economics of Giving Up

For most of my life, making a computer do something new demanded a mountain of knowledge. The machine could do the work. Getting an idea through the operating system, programming languages, libraries, and protocols was the expensive part.

Programming has always been a Tower of Babel. Each layer of logic, logistics, and protocols had its own vocabulary and peculiar assumptions about how the world ought to work. Learning a programming language got you through the first door. Then you had to configure your environment, manage dependencies, and persuade tools built by different people to cooperate.

If you learned to program the way I did, your first couple of years involved variations on "Hello, World!" and fights with things unrelated to what you wanted to make. Another concept, another library, another technique: you clawed your way toward something interesting. By then, you'd learned that building software meant solving problems you hadn't intended to have.

This was the price of admission. We accepted it because there wasn't another way. Navigating the machinery became synonymous with creating software, and we forgot it stood between us and the things we wanted to build.

That price of admission is collapsing. The machinery still exists, and understanding it helps. But you can begin with what you want to accomplish and learn the technical details as you build. For the first time, the idea can come before the apprenticeship.

AI is becoming a translator between human intention and those layers of technical machinery. Microsoft Research's work on conversational programming explores this shift: using natural-language instructions to develop software while checking whether the machine understood the task. You still have to test the result. But you can start by saying what you want.

I've written code for a living for decades. I can learn new technologies when I need to. None of that kept me from accumulating a lifetime's worth of ideas I never built.

Time stopped me.

Suppose I wanted an application to organize a collection of files. Writing it could eat several evenings of development, testing, and debugging, then demand maintenance. Or I could search for twenty minutes, find a program that came close, tolerate its peculiarities, and get on with my life.

Making do was the rational decision. My imagined application suited me better, but the improvement didn't justify the work. Millions of people made the same bargain with thousands of small inconveniences.

We tend to think of limitations as properties of the world. We can't fly unaided because our bodies aren't built for it. We can't travel faster than light because physics has opinions about that sort of thing. But many of our everyday limits come from the time, effort, knowledge, and money we can spend. They're economic.

For decades, achievable ideas cost more than the people who had them could afford. We worked around those limits until we stopped recognizing them as limits. They became the texture of life.

Someone else's software organized our photographs, managed our calendars, and controlled our entertainment. We built our lives around their choices because building our own cost too much.

The gap between what we wanted and what we could make shaped an industry. Companies charged to cross it for us. Now we can begin crossing it ourselves.

As more people discover what they can build, those companies will evolve or die. And if it means the subscription-based model of software development goes away?

Good riddance.

This is a four-part series on software abundance: how AI reduces the cost of building what we want, who loses their moat, and what becomes possible when an idea doesn't need a business model. Next: The Rentier's Last Stand.