The Collapse of Effort and the Birth of Abundance

By Austen Tucker · 2026

I always have ideas. They simmer around like popcorn: some are obviously good, some are probably terrible, and some look delicious right up until I calculate what they would take to make.

For most of my life, I have had a very useful internal censor, and it has saved me from scatterbrain projects more times than I can count. An idea had to get good enough to rise above a certain floor before I would actually make it. Sometimes I would sit with an idea for months or years, write a little ditty in a notebook, wait, combine it with another idea. Eventually, if the spark came together, maybe it became a story. Maybe it became a project.

That filtering process has been part of my creative life for as long as I can remember. The internal math was ruthless, but it was usually right: three weekends. Needs an illustrator. Needs a backend. Needs networking. Needs a hundred pieces of art. Cute, but not worth it.

This week, the math stopped mathing. And that is fucking with me.

A silly idea for a gift

The first domino was a silly idea for a gift. I have a neighborhood chat full of people I love. They are my neighbors, but they are also dear friends of mine, and I wanted to make something that showed my appreciation for them.

So I decided to make a game. The first version was just baseball. My request was basically: make NES baseball with furry creatures.

I had been playing with Astra and seeing people build these little 3D worlds, and I wanted to do that. I do not particularly like baseball games, but I love basic baseball games. You pitch. You bat. It feels less like a modern sports simulator and more like one of those old pitch-and-bat pinball machines from the 1960s. That is my vibe.

Furball baseball ground throw in playtest lab

At first, I assumed the character models were going to look bad, so I decided to make that intentional: Nintendo 64 characters. That changed everything.

I already had a character in my harness named Moxie, who is kind of the mascot of my website. She is everywhere, and I have her inference saved so I can generate her consistently whenever I want. I wondered what would happen if I took that inference and asked Astra to turn Moxie into a Nintendo 64 character. So I did it, and it worked really well.

Moxie holding a bowling ball in the playable 3D scene")

And because it only took me a few minutes, my brain immediately went to the dangerous place: what if I did this at scale? That question split into two more. How do I get my friends to upload their own fursonas into the game? And if I am already doing baseball, what else lives in this world? From there, one thing led to another, then another, then another.

Rogers Bark Municipal Club

There is a Rogers Park chat full of furries. Naturally, we call it the Rogers Bark Chat.

Once I had baseball working, I started thinking about games like Pikmin, games where you return to a consistent world. I wanted the place itself to feel coherent. Local. Personal. So I thought: what if we had a municipal club, but it was just our fursonas? Now we had a framing story: a neighborhood baseball field, a golf course, a municipal recreation complex populated by my friends.

I based the golf course on a real Chicago course near Montrose Harbor, partly because I am lazy and partly because I love the joke of taking a municipal sports club far too seriously.

Rogers Bark golf prototype with ball in flight

And somewhere in the middle of this, I realized something important: I am not trying to sell this game. I might put it on itch.io someday for pocket change. Fine. But this is not a career strategy. It is a gift.

And the only reason this gift is possible is that the cost of making something has gotten so low. Not zero, but low enough, and that distinction matters.

A few years ago, “make a baseball game for my friends” was an idea I could carry around for years. This week, I sat down for an afternoon and had a working prototype. If it sucked, I had lost an afternoon. Big deal. If it was good, I could put more time into it. That is a completely different risk profile for an idea.

The filter inverted

For most of my creative life, the question was whether an idea deserved to get made. This week, that filter inverted. Now it is almost always worth making the first version. The question is: How polished an experience do I want to give this? That is the switch.

I have lots of things in my harness that work and that I do not care to share. I have one that checks my email. It does not need to be amazing because I am not selling it or giving it away; it can be janky. If I am making something for myself, “works” may be enough. If I am making something for my friends, maybe I want charm. If I am making something public, maybe I want polish.

The floor of creation has collapsed so far that I no longer need to decide what deserves existence. I need to decide what quality of existence it deserves. That is a very different problem.

And this was the first week where I really felt it. I had an unusual amount of free inference from DevDay, basically unlimited ChatGPT inference, and I am unemployed right now. For once I had both the compute and the time. You better fucking believe I used every single drop of it.

For a few days, I got to live inside an artificial economy of abundance, and I stopped rationing. That may have been the most important part.

The old accountant goes offline

The internal censor used to be my best friend. It protected me from becoming the person with forty unfinished projects and nothing to show for any of them. It forced me to collate ideas, remix them, and wait for enough sparks to become a fire.

But when the cost of the first prototype collapses, that censor stops being purely useful, because now I can test the idea today instead of preserving it for someday. I can build the version that teaches me whether the thing has a pulse, and if it does not, I throw it away.

The cost of being wrong has collapsed. That is maybe the part that matters most.

For the municipal sports project, even playtesting became something I could hand off. Instead of thinking, “Oh man, I have to playtest this and tune it,” I can send an overnight agent into the game and say:

Play this all night. Come back with feedback. Make changes. Try again. Keep iterating until it feels good. And that works.

This does not mean judgment disappears. It means judgment becomes the scarce resource: taste, intent, knowing whether the thing feels right, knowing when the weird detail is the whole point, knowing which compromises are charming and which are just broken.

Implementation used to dominate the budget. Now it is increasingly something I orchestrate.

I do not really play Minecraft anymore

My friends and I run a Minecraft server every winter. It is one of my favorite rituals because it is a chance to hang out with my closest friends and family while hibernating, which is possibly the best thing.

The joke is that I do not really play Minecraft. I am on the Minecraft server, and I am definitely doing things in Minecraft, but I am not really playing Minecraft anymore. I have probably spent around two thousand hours in this game across years and servers, and at some point you start seeing the seams.

You stop seeing a cool area and start seeing the biome; you stop seeing a dungeon and start seeing the spawn rules. You know where the items come from and what the systems are doing.

So instead of playing the game, I use the game as a canvas. My goal is always to make something that breaks the game in some way, or plays it in a weird way.

Last year, that meant ComputerCraft. ComputerCraft is a Minecraft mod that gives you programmable computers running Lua inside Minecraft. Yes, it is completely unhinged. It is a lot of fun.

Last year, using the frontier models and an IDE, I built a handful of games. Slot machines. Blackjack. A little casino.

Original ASCII horse race

Original blackjack deal hit and stand

This year, I pulled the old code back out. And I looked at it with completely different eyes.

The work became portable

While I had been building the Rogers Bark Municipal Club, I had made a structural decision that turned out to matter a lot. I kept the game rules separate from the rendering engine. Baseball rules were rules. Golf rules were rules. The simulation did not need to know what screen it lived on, partly because I wanted an agent to be able to tune the game overnight without looking at the visuals.

Then I realized: it is just code. If blackjack exists as a clean ruleset in TypeScript, there is no reason it has to stay there. The LLM does not really care about rewriting the same idea in another language.

So I handed it the TypeScript that defined the games I had been making for the municipal project and said: Turn these into ComputerCraft games. And use Pine3D.

Pine3D is a fancy, completely unhinged 3D engine running inside a computer inside Minecraft. Please hold your applause.

Pine3D Derby overview and follow camera

I was not looking forward to rewriting blackjack in Pine3D. It would have been a slog: I would write things, the model would get something wrong, I would fix it, something else would break, and I would keep chasing bugs through code that I did not actually care about.

But now I know enough to write the test harness first, define the contracts, and hand the implementation off. Suddenly the middleman work disappears, and what remains for me is idea, intent, and taste. It was electric.

Pine Arcade carousel

Pine Jack winning hit and stand

Pine Jack split and double

Generative AI in Minecraft design, three hours or less, guaranteed

Then I looked at my old turtle code. Last year, I tried to program Minecraft turtles to do my mining for me. The idea was that robots would do all of my work. It did not quite get there. This year is going to be different. Robots are going to do all of my work.

As I was working on the Pine arcade games, I had this thought: I wish I could design in Minecraft the way I design in my AI harness. Doing that directly inside a twenty-year-old game sounded miserable. So I stopped trying to solve it inside the game. Instead, I built an external Electron app.

Turtle Blueprints 3D structure with selected blocks and editing wheel

The app can generate Minecraft structures and export them into a format that Builder's Gadgets 2 can paste into the world. I already knew which mod I wanted. I knew the format existed because the author had open-sourced it.

The documentation was not especially complete, and a few years ago that might have killed the afternoon. Now I do not care. The LLM can read the source, figure out the format, and write the conversion layer. I describe what I want; the app produces the blueprint.

Turtle Blueprints natural language generation dialog

Minecraft pastes the building. Generative AI in Minecraft design in three hours or less, guaranteed. It is funny. It is also not a joke.

The space between idea and execution has completely collapsed. And once you taste that, it changes you.

I only want to orchestrate now

I know there are people who love process: sitting down and doing the thing with their own hands, loving the code, the brush stroke, the act of craft for its own sake. That is beautiful.

I am not that kind of person. I am a builder. I want to make cool things and then move on to the next cool thing to build.

I am still building. I am just not always in the code anymore. I am directing the builder.

The conversation directing this essay and its project evidence

Once I have a test harness, a clear interface, a set of acceptance criteria, and a way to inspect the result, I do not get extra spiritual nourishment from typing the implementation myself. I want to say: Go do that.

Then I want to look at what came back. I want to decide whether it is good. I want to know what the next thing should be.

The job shifts upward. Architecture. Taste. Intent. Judgment. Orchestration. That is not less creative to me. It is closer to the part of creation I always wanted.

And then there is the Tak board

This is not only happening in software. This week, I have also been iterating on a folding Tak board, a real physical object with pins, hinges, friction fits, storage, clasp geometry, filament, and the extremely annoying fact that physics remains stubbornly unionized.

Physical folding Tak board and piece storage

I can still be wrong. A print can fail, a hinge can bind, and a 200-gram print can still become a very educational piece of plastic.

But the cost of thinking again is radically lower. I can make a test coupon, discover that I am wrong, explain why, revise the geometry, and print the next answer. Some mistakes that once would have cost me days of CAD work now cost me one small test print and another pass.

The atoms are still expensive. The thinking is not. And that means orchestration is starting to escape the screen.

The backlog starts to empty

During this same stretch, I also wrote about ten thousand words. Three essays. A roughly seven-thousand-word short story for a novel of stories I am working on. An essay about trans equality. That does not feel separate from the rest of this. It feels like I am emptying my to-do list.

As a creator, you end up with this enormous backlog of things you cannot give time to: ideas you love, ideas you might love, little systems you wish existed, stories with one good scene and nowhere to go yet, applications you wish someone would make, tools that are annoying but not annoying enough to justify building. Historically, a lot of that material stayed compressed into a note in a notebook, a half-page outline, a folder called “someday.”

Now I can generate a version. Not necessarily the final version, but enough of one to find out what the idea actually is. If I have an idea for an application, I can try it and see if it works. If I am working with an old program and it has an API, I am no longer tied down to your app. I can write my own shit.

That changes the psychology of having ideas. You do not have to keep all of them suspended in amber. You can let them out.

The weird economics of abundance

This is where I get nervous for a lot of software businesses. I keep hearing people talk about the “SaaSacre,” and I understand why. If your product is fundamentally a thin interface over something I can reproduce for myself, that is a dangerous place to be.

There are still services where scale, trust, regulation, physical infrastructure, network effects, or transaction systems matter enormously. But a lot of software has historically been protected by the sheer inconvenience of making an alternative. That inconvenience is disappearing.

The interface is no longer the constraint it once was. Neither is documentation, and increasingly neither is language choice. A missing integration may no longer mean “I guess I cannot do that.” It may mean, “fine, I will build the adapter.”

And for personal software, weird community software, tiny tools, gifts, experiments, and local systems, the economics become almost absurd. A game for twelve friends can be worth making. So can a personalized character, a weird little municipal golf course, or a Minecraft casino with a horse-racing screen hanging over a bar.

Derby practice betting station

Pine3D Derby finish

Shared practice house cashier and Pine Jack

A cinematic camera for the dice in Shut the Box can be worth making. Historically, this is the exact moment responsible adults stop. This week, I did not stop.

And some of the things that would once have been dismissed as scope creep are exactly the things that make software delightful. Texture, personalization, and weirdness used to be expensive. Now some of that cost is collapsing too.

What becomes scarce when effort does not?

The interesting question is not whether this lets me make more software. Obviously it does. The interesting question is what happens to creative judgment when feasibility stops being the primary filter.

I used to ask, “Can I afford to make this?” Now that question barely survives contact with reality.

The better questions are harder: How finished does this need to be? How much care does this idea deserve? Who is it for? What does it need to feel like? What should this thing be?

That is a harder set of questions. But I think it is also a more human set of questions. Because if implementation becomes abundant, then maybe we can justify making things for reasons that used to be economically ridiculous.

Because it helps twelve people feel included. Because it makes the neighborhood chat laugh. Because my friends will recognize themselves in it, or someone who could not participate before gets a door. Because winter is long and playing Minecraft with your family is nice. Because a silly gift is still a gift. Because delight counts. Maybe that is what abundance buys us: not just more productivity, but more permission.

The other side

I do not think we are at the easy part of this transition. The path is long, the road is hard, and there are enormous questions about labor, ownership, trust, culture, power, and what happens when the economics of creative work change this quickly.

I do not know what happens when millions of people discover that the distance between “I wish this existed” and “I made a version” can collapse from months to hours, or what happens when the backlog starts to empty.

But on the other side of this journey is a world of cognitive abundance the likes of which this world has never seen. I do not know what that does to us. I am excited to find out.