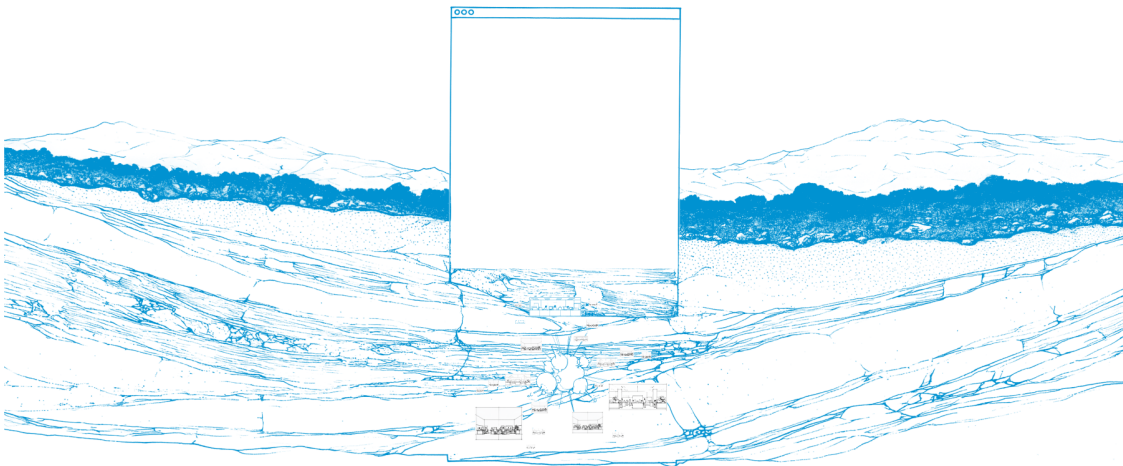


EVERY

How Codex Happened

Inside OpenAI's comeback in coding, from a failed launch to the app that took over ChatGPT

By Dan Shipper



I can remember exactly when I began to think that OpenAI would lose the mandate of heaven—the seemingly insurmountable lead they had built in AI coding since ChatGPT originally took off. I was in their Mission Bay office, in the summer of 2025, the night before the launch of GPT-5.

There was a certain buzz to the building. The overhead lights burned like stage lights, and people seemed to walk the warm rusts and browns of its halls just a little faster than usual. Even the gravity felt heavier—as if the sheer size of the following day's launch had drawn the rest of the world into its orbit.

This made what I had come to say a little awkward.^[F1]

Every, the company I run, had begun to build a reputation for testing new AI models on real work and explaining what they could—and couldn't—do. OpenAI had given us access to GPT-5 about a month before launch so we could help them understand the model's strengths and weaknesses and publish our Vibe Check—a detailed review—on launch day.

Ten of us had spent a week in a house in upstate New York for a team retreat, eating and working together at one long wooden table and gathering on a couch in front of a big

screen to test the long-awaited model. We were especially excited to try it for coding, where AI was advancing fastest. But while we liked it for everyday use, our reaction to it in Codex was muted.

Today, Codex is the centerpiece of OpenAI's comeback—the product that has swallowed ChatGPT from the inside and become the foundation for everything the company is building next. But that summer it was still a scrappy effort to build an AI software engineer. A version launched in the cloud a few months earlier hadn't found product-market fit, and the team was mid-pivot to a new version that ran on a developer's computer in their terminal and in their code editor.

In Codex, GPT-5 was fast, and it seemed smart. One of our most technical engineers, **Naveen Naidu**, had used it to track down a mysterious freeze in his voice dictation app Monologue that had stumped him for days. But it also seemed to stop too early, as if it were overly wary of stepping outside the bounds of what it was allowed to do. It felt designed to be used as a pair programmer.

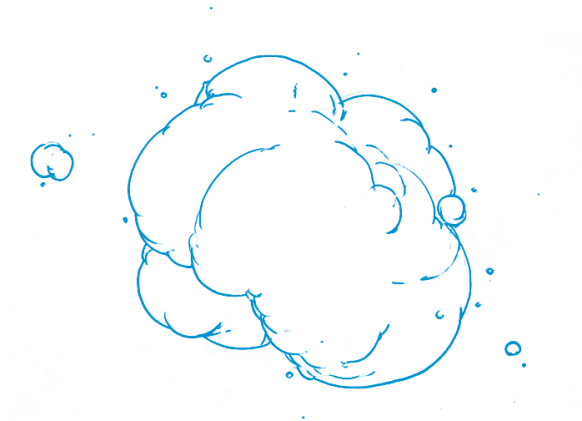
As a team, we were moving beyond step-by-step collaboration with AI models. Using Anthropic's terminal agent, Claude Code, a group of us led by **Kieran Klaassen** had started handing entire tasks to Opus 4 and walking away until a finished feature—or sometimes an entire app—came back. With minimal help from an engineer, staff writer **Katie Parrott** built and deployed her own AI content management system in a few days. It had some bugs, but building it would previously have been out of reach.

Next to Opus 4 inside Claude Code, GPT-5 inside Codex felt boxed in.

Sharing what we'd found in our testing had brought me to a conference room in OpenAI's office with **Lindsay McCallum** from the communications team and **Alexander Embiricos** from the Codex team. I sensed that McCallum, who has a keen radar for vibe shifts in the market, had set up the conversation so I could explain to Embiricos why I suddenly wouldn't shut up about this new way of working.

McCallum was typing with intense focus on her laptop across the conference table from me. Embiricos sat beside me, coding while he talked, which I took as a good sign. He was rushing to finish a GPT-5 release feature that would let the agent pause and ask a human for permission to access files or run commands outside its sandbox.

“With Claude Code, you just let it rip,” I told Embiricos. “GPT-5 feels a little timid—like it's held back by the sandbox and it seems afraid to use its own power.”



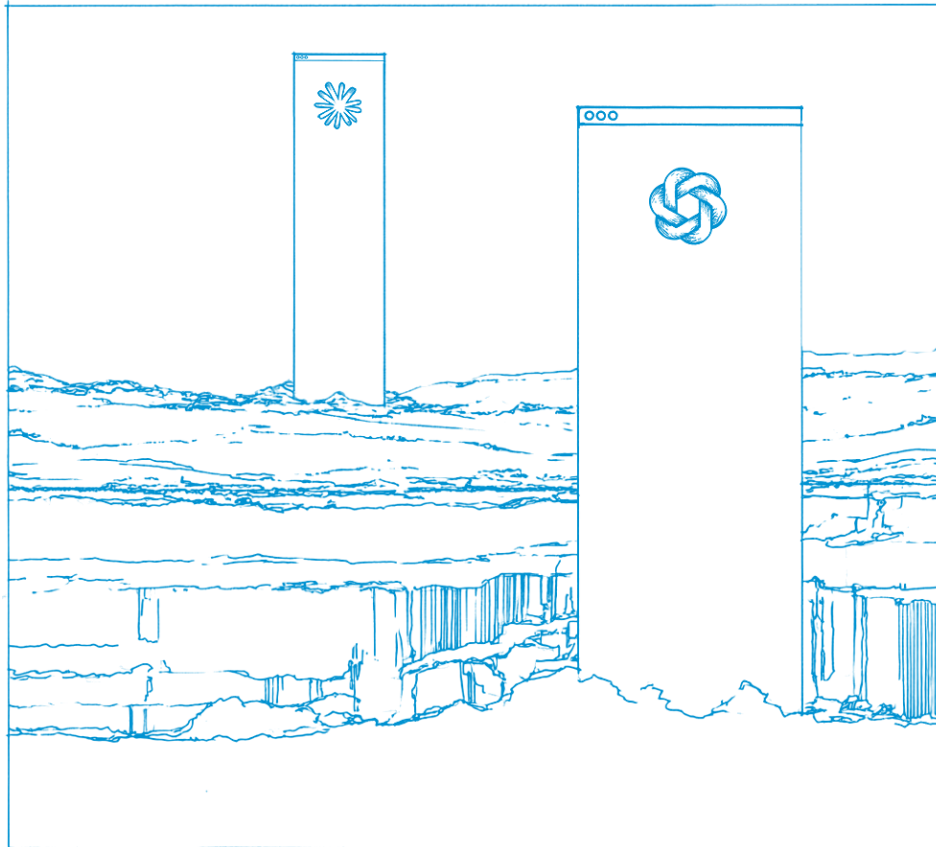
The tapping of McCallum’s keys slowed. “Tell Alexander about this new engineering workflow you have,” I remember her saying without looking up from her laptop.

Embiricos swiveled his chair to focus fully on me.

“We don’t even really look at the code anymore,” I began. But I blushed as I said it. I felt ridiculous explaining agentic delegation to Embiricos, of all people—building an autonomous coding agent was his job. I also wasn’t sure, at the time, that our engineering approach would work beyond our tiny team.

A Slack ping lit up Embiricos’s screen and broke the sudden quiet of the conference room. He muted his computer and pushed its lid three-quarters shut—just enough to keep his Codex agent thinking.

Leaving Mission Bay that night, I wondered if OpenAI was building the perfect tool for a dying workflow. Burned once by its failed cloud agent, the company looked like it was retreating to the code editor, where AI helped programmers work step by step. Meanwhile, Claude Code was taking off among people handing over entire tasks and walking away.



There were good reasons to stay in the editor. After GPT-5 launched, the Codex team shipped an extension that tucked Codex into the sidebar of code editors like Microsoft's Visual Studio Code (VS Code)—where programmers write, run, and debug code—and Cursor. Usage grew more than 10-fold by last October, and the editor extension quickly overtook the terminal to become Codex's most-used surface.[1]

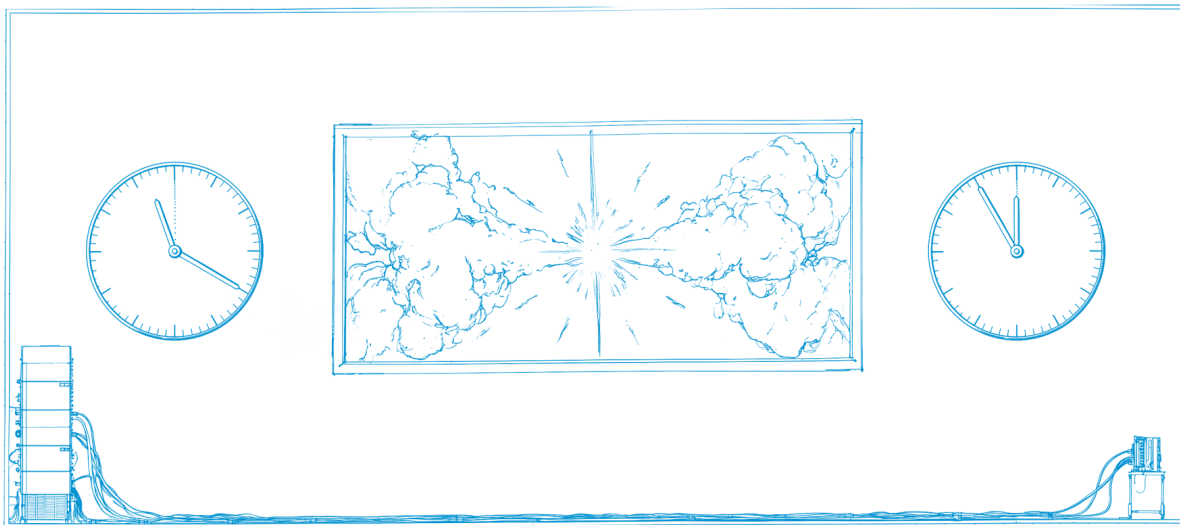
In most companies, that would have been the business to double down on. But Embiricos and a small group of engineers, designers, and researchers inside OpenAI had bigger ambitions.

In the months that followed, the Codex team trained and rapidly improved a dedicated line of coding models, separate from those powering ChatGPT, and built a desktop app for managing multiple agents at once. By June 2026, Codex had more than 5 million weekly active users and was spreading beyond programming into other kinds of knowledge work.

By July 2026, Codex had merged into the flagship ChatGPT desktop app. Its leader, **Thibault Sottiaux**, had been elevated to head all of OpenAI's core products and platforms, including ChatGPT, Codex, and the API. Today, Codex and ChatGPT for Work combined have 40 million weekly active users.

In less than a year, the Codex team went from a failed cloud launch to becoming the foundation for OpenAI's flagship product—and in doing so, they helped remake the company.

This is the story of how Codex happened.



Thinking...

Authored imagined transition; not an authentic model reasoning trace.

Thinking...

Something is off. The lead I had a year ago isn't showing up in the results...

It looks like the problem starts earlier than I assumed...

Going back to the beginning...

OpenAI's comeback begins with a simple line: "Thinking...."

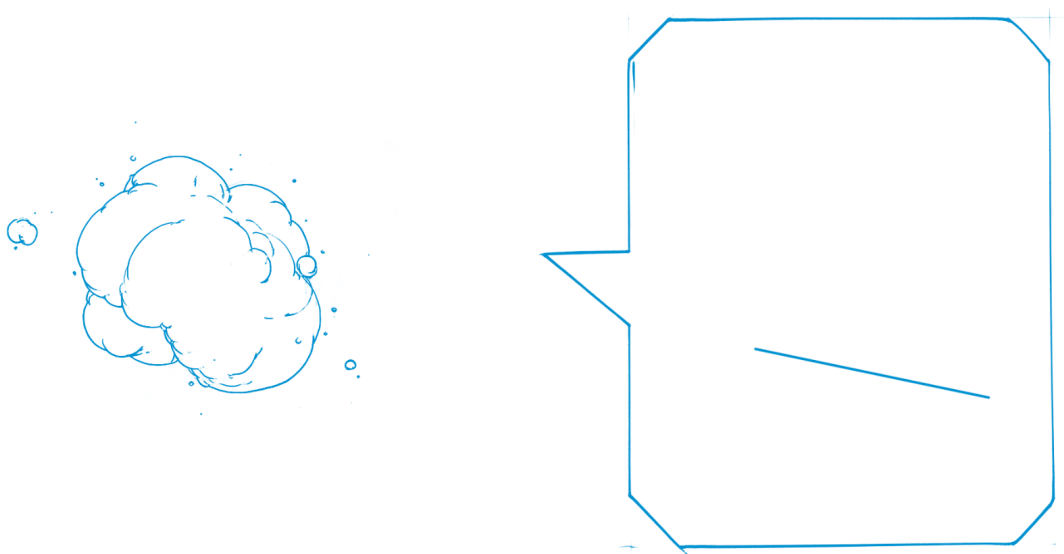
In 2026, any time you interact with an AI model on any kind of complex request, you'll see that word—*thinking*—shimmering in the place where your response will appear.

It might seem like a throwaway: a one-word replacement for the spinning beach ball on macOS or the circling blue ring Windows displays beside its cursor, each a sign that the computer is too busy to respond.

But "Thinking," in this context, denotes a new style of computing without which Codex would never have happened.

The first generation of large language models were external processors. They had no shimmering status label for their internal reasoning; instead, they did their thinking directly by responding to you. Whether you asked them to write an email or a line of code,

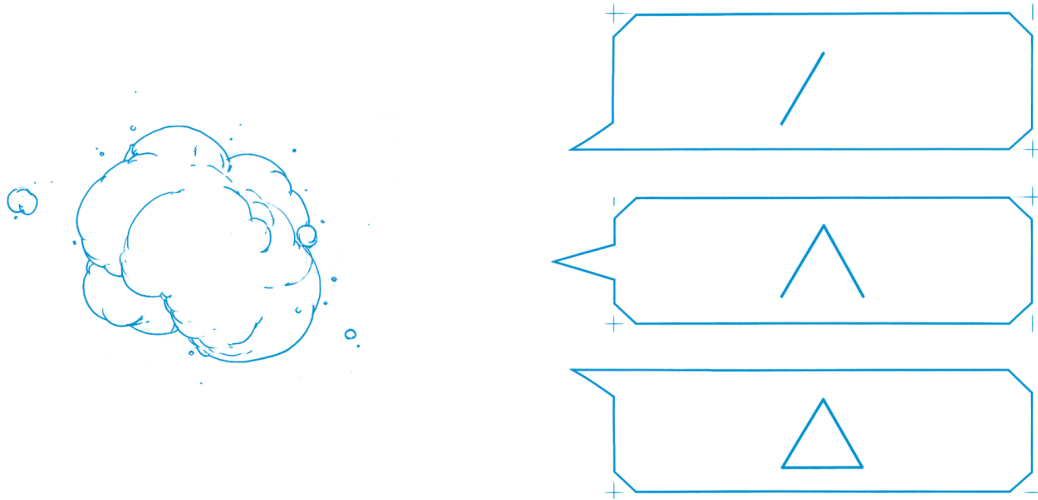
they had to work out their answer one word at a time, with no opportunity to plan ahead or revise afterward.



Three strokes produce an unclosed triangle in a public response.

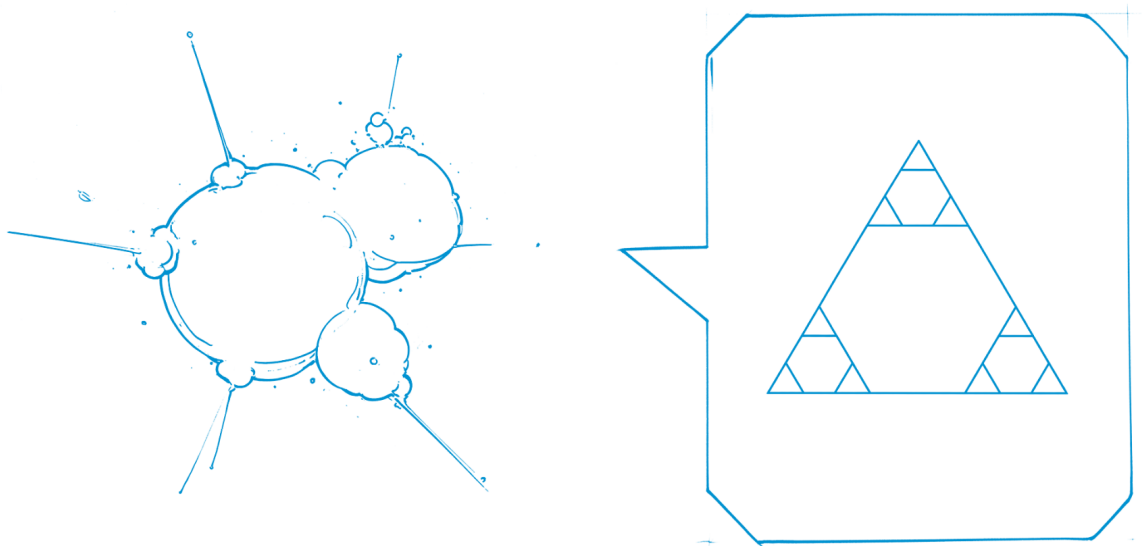
In that way, early chat models were like improv actors given a prompt by the audience as the basis for a scene invented on the spot. As these tools composed on the run, their results could be impressive—and sometimes unintentionally funny. (They would insist “strawberry” had two r’s, spell it out to prove it, and congratulate themselves at the end.)

Early users of these models noticed that if you asked the model to write out its reasoning step by step before answering, it would often arrive at a better answer. This technique, the same advice as a grade school math teacher would give a student, was called chain-of-thought prompting.



Three public steps draw one stroke, two strokes, then a correctly closed triangle.

Chain-of-thought prompting led to the second generation of AI models and the shimmering “Thinking” user interface. Rather than requiring the user to ask the model to work through a problem step by step, OpenAI trained that behavior into the model itself. Before responding, a reasoning model works in a scratch pad, called a reasoning trace, where it can attack a problem from different angles, backtrack out of a blind alley, and pick a path more likely to succeed. It responds only when it has settled on an answer.



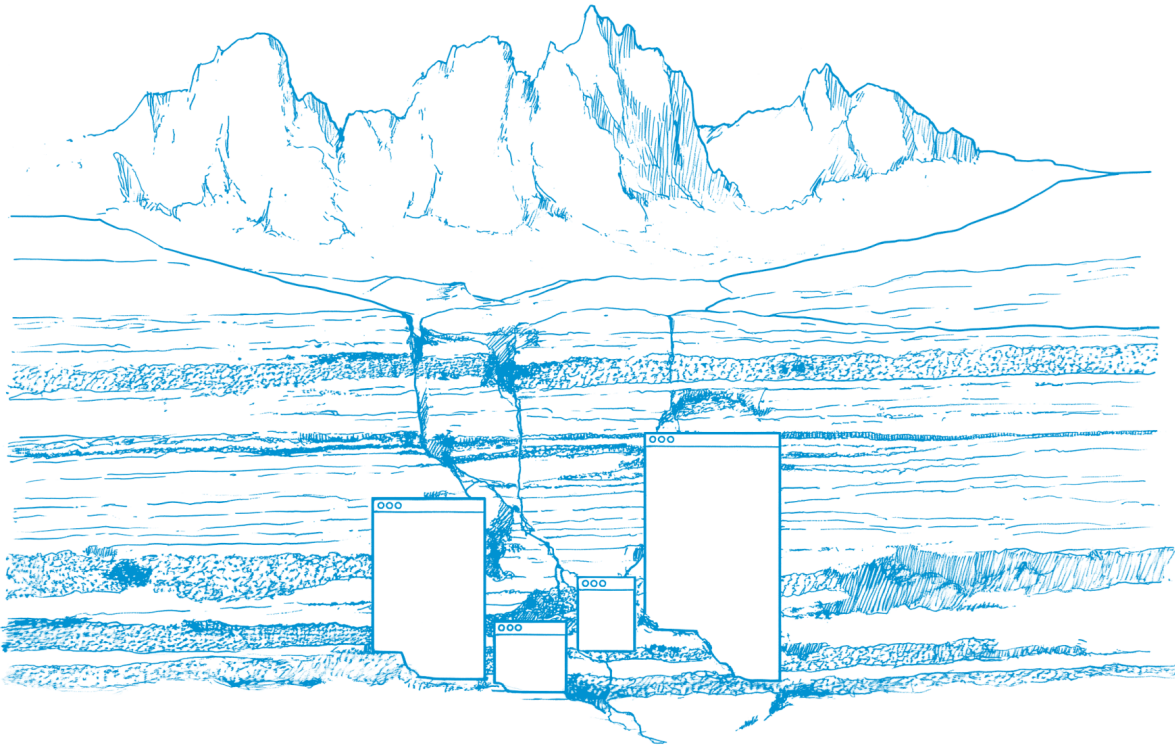
A model works on a fractal in a thought bubble, then presents the same fractal in a speech bubble.

OpenAI called its first model of this kind o1 and released a preview of it in September 2024. Until then, progress had depended mainly on training larger models for longer. With

o1, OpenAI found that the same model could improve its answers on a difficult problem simply by spending more time on it.

One kind of work stood to gain more from this than any other: programming. Programming is uniquely suited to reasoning models because it offers a closed loop: A model can edit code, run it, and see within seconds whether the change worked, allowing it to test and revise fixes on its own without needing a person to check.

Left to think for long enough, theoretically, a reasoning model could do any programming task by itself.



Many beginnings

Building an AI programmer has been a goal at OpenAI for so long that Codex has had many beginnings—starting with its name.[2]

The first Codex, released in 2021, was a version of GPT-3 fine-tuned on public code that powered the original GitHub Copilot. Three years later, in the weeks after o1's preview showed what reasoning models could do, a prompt from leadership set several groups across OpenAI chasing the dream again from different directions.

One of those beginnings came from Embiricos in late 2024.

“The first real, massive ‘feel-the-AGI’ moment for me was [when] I saw someone running a reasoning model that was connected to a terminal,”[3] he told me, referring to when an AI stops feeling like a tool and starts feeling like a machine that can think and act like a human. “I know [it’s] not exciting now,” but watching it read files, run commands, and work through errors on its own “was fucking cool.”

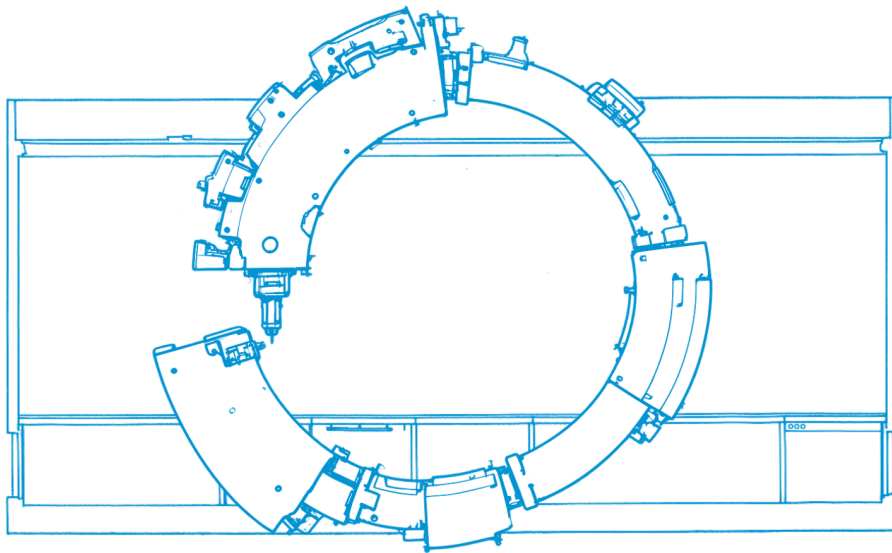
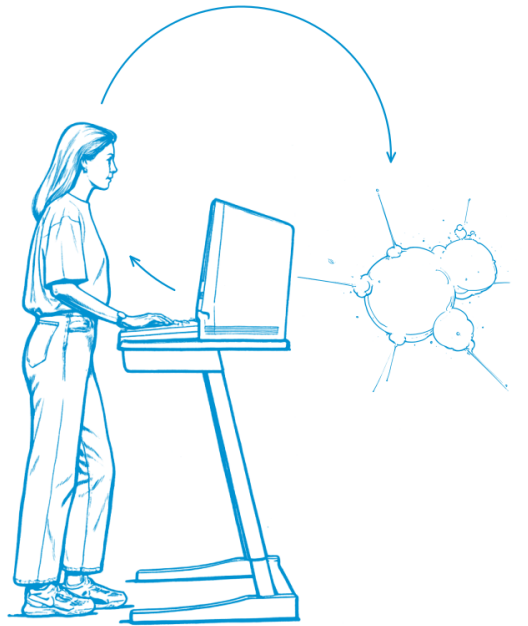
Embiricos, the product lead I’d met on the eve of GPT-5, had come to OpenAI earlier that year when it acquired Multi, a multiplayer screen-sharing tool for programmers he’d cofounded. At OpenAI he’d pursued a related idea: letting humans and agents work together inside the apps they used. He chased that vision first on the ChatGPT desktop app and then on Atlas, an AI-powered browser,[4] but the models at the time were too slow and clumsy at reading a user’s screen^[F2] to be useful.

In the terminal, though, the models didn’t have to read a screen—they could work directly in text and code. Seeing an early reasoning model in the terminal convinced Embiricos that they could power a real coding product. Coding wasn’t yet his full-time job, but he immediately started sketching one. For it to work, he believed, users would need to see what the model was doing every 30 seconds, or at most every minute.

Speed was the problem again. Most AI coding products at the time grew around VS Code. GitHub Copilot added AI autocomplete through an extension; Cursor and Windsurf built their own editors on VS Code’s open-source code, adding AI chat alongside the programmer’s work.

These tools depended on a tight feedback loop. Copilot completed the line you were typing; Cursor and Windsurf answered questions within seconds. None asked the model to work on its own for long, because the models couldn’t. OpenAI’s early reasoning models were the reverse: slow and weak when their thinking time was cut short.

Could he turn their flaw—the time it took to get a response—into an advantage?



Building the builders

Meanwhile, in another part of OpenAI, Sottiaux, who goes by Tibo, was trying to turn reasoning models back on themselves: using their powers to help researchers build the next generation of models.

“I got involved with coding agents around December 2024,”^[5] he told me. We were sitting in the OpenAI cafeteria. His glossy brown hair fell to one side, and he wore thin wire glasses that caught the afternoon light. If above the nose he looked like an architect, below he was all programmer: A beard of brown stubble paired with a dark brown hoodie printed with “Research and Deployment Co,” and a slice of pepperoni pizza in hand.

The goal, he told me between bites of pizza, “was to help us write infrastructure and do research faster.”^[6]

Sottiaux had spent much of his career building infrastructure for researchers, first at DeepMind and then at OpenAI. Research infrastructure—the shared code that AI researchers rely on to run, evaluate, reproduce, and deploy experiments—might seem like an odd origin story for the person who would eventually lead Codex. When you think of infrastructure, you might think of servers and undersea cables, the tech world’s equivalent of plumbing: necessary, but invisible until a pipe springs a leak.

In the AI world, though, infrastructure—and particularly research infrastructure—is where much of the action happens. If you can cut the time it takes to run an experiment, you can run more experiments and get a better model sooner.

“At what point did you split out the model line?” I asked. “How did you decide?” I wanted to fast-forward to the action. Having its own model line felt core to Codex’s magic, and I pictured a meeting inside OpenAI where Sottiaux, president and cofounder **Greg Brockman**, or CEO **Sam Altman** made the farsighted choice to separate Codex from the rest of ChatGPT—giving the team the freedom to ship quickly and upend the company from within.

“There was no decision to split off the model line^[F3],” Sottiaux said matter-of-factly. At the start, he said, “this was a research exploration and an optimization of our velocity.”

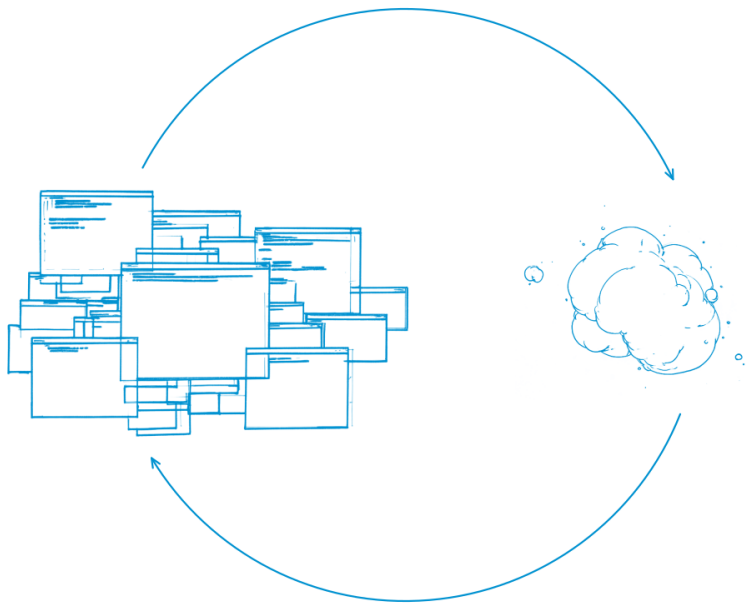
Rather than building a model for public consumption, Sottiaux’s group trained internal coding agents on OpenAI’s monorepo,^[7] the single repository where all of the organization’s code lives. The agents wrote the Python code OpenAI’s researchers needed to run and improve the systems that built new models. Those researchers used the agents every day, giving the team a tight feedback loop.

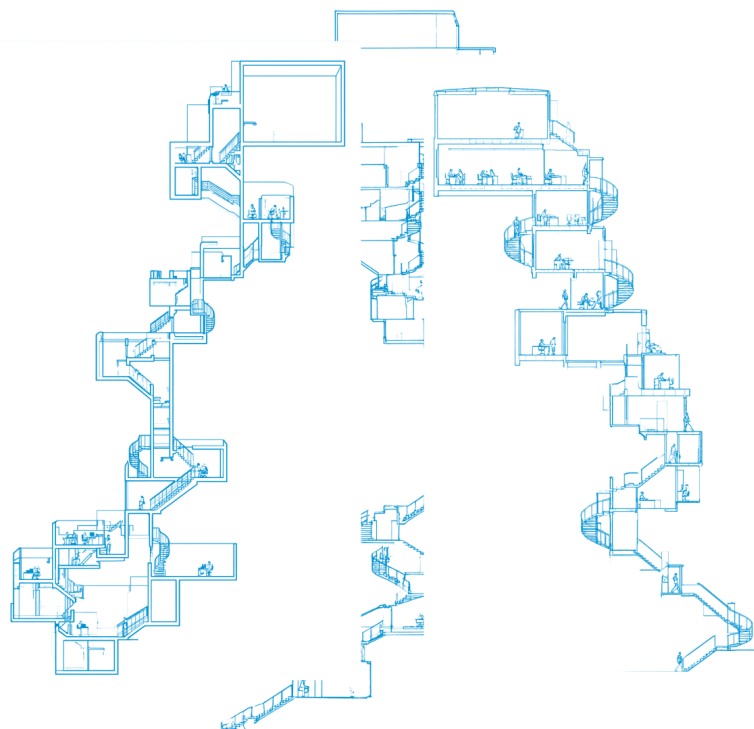
“We were seeing rapid gains on coding,” Sottiaux said. His team was training models frequently, using techniques that differed from—and for a while, were hard to reconcile with—how the larger organization trained models for the public.

Sottiaux, like everyone else at the AI labs, is tight-lipped about those training techniques. But it became clear they were working: His group had successfully turned the models back on the research process that created them, and researchers were using those models to make the next generation.[8]

Engineers call this bootstrapping. Make a model that's good at helping researchers run experiments, and the researchers can use it to build a more powerful one, which runs better experiments, which leads to a better model, and so on.

This effort supplied one half of what became Codex: agents that could improve the infrastructure for building new models. To become a product that the rest of the company could use—and that would eventually make its way beyond the confines of OpenAI—it needed the other half: Embiricos's thesis about how people should work with those agents.





Convergence

Authored imagined transition; not an authentic model reasoning trace.

Thinking...

Three approaches so far...

Two of them overlap...

I'll merge those and continue with what works...

After exploration comes convergence.

We can picture OpenAI from above in the last months of 2024 and early 2025. After coding became a top-level company goal,[9] several teams of individuals including Sottiaux and Embiricos had started working at oblique angles on the same question.

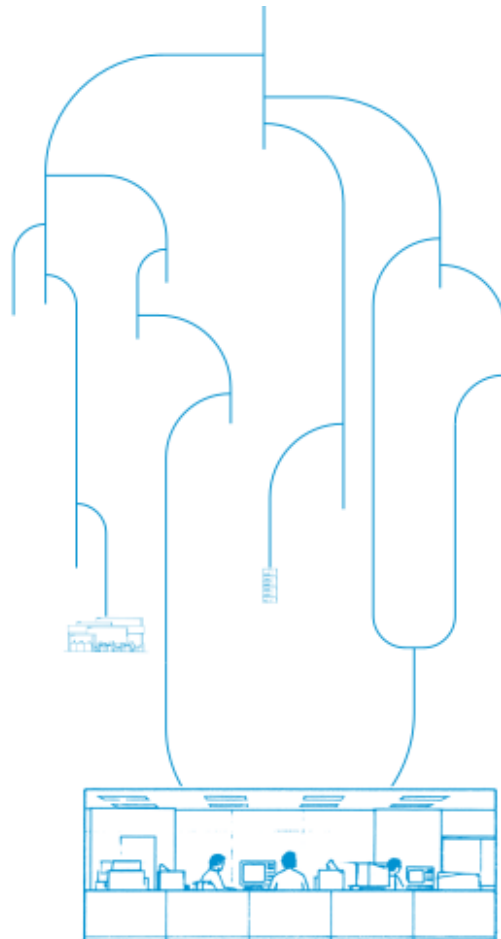
The open question, Brockman told me, was the right “form factor”:[10] whether the agent belonged in a terminal where developers execute commands, in an editor where they write code, or in the cloud working on its own computer. Some of these paths were blind alleys, but at the time nobody knew which.

“There’s some sense in which OpenAI is betting on a series of startups, but they’re all internal,” Altman said in an interview with **Ben Thompson** in 2025. This spirit of parallel inquiry inside of the company sometimes feels chaotic and wasteful^[F4] to a few insiders I spoke to. Who wants to work on the team that built the third-best agent product inside of OpenAI?^[F5] Having run a similar strategy at my own company, I know that there’s a thin line between creating space to explore and creating an utter mess.

But in watching OpenAI—and despite the internal grumbling I’ve heard—I have also come to see it as necessary when the frontier keeps moving. In AI, fast “sustaining” progress at the model level—to use the late Harvard professor **Clayton Christensen**’s term—makes new kinds of apps possible, constantly disrupting the ones that came before.

A model that can finish a line of code works well as an autocomplete tool; a model that can think for 30 seconds can answer questions in a chat sidebar; a model that can work for an hour wants to be something else entirely. Building a successful product around the previous model doesn’t mean you know what’s next. You need several teams trying different ideas, then bringing their work together once you see what’s useful.

By early 2025 OpenAI was ready to do just that. “Eventually, we thought we should combine forces with another effort, which was called Scientist Assistant,”^[11] Sottiaux told me—a merger that brought him together with Embiricos. Across the various internal coding efforts, Brockman said, “We ended up merging two of them. And then merging another two of them.”^[12] The combined team was a small multidisciplinary crew of technical product thinkers like Embiricos, infrastructure and engineering wizards like Sottiaux, and researchers like **Mia Glaese**.



Glaese came to OpenAI from DeepMind, where she worked on Gemini and model alignment. At OpenAI, she was responsible for the data used for model training and alignment,[13] meaning she specialized in getting and using data that would shape the specific behaviors of the models.

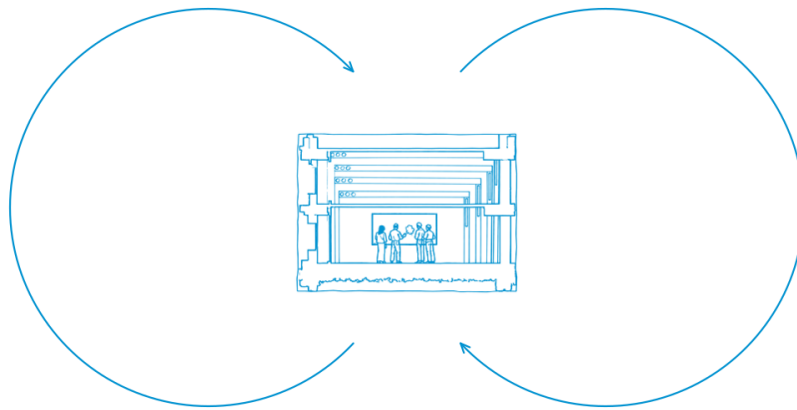
On one level, she is the ultimate unsentimental empiricist. In our interview, she was unapologetically quantitative: “Just trusting vibes is not that useful in research,” she said. Yet when she talks about the process of making a model, she is quick to laugh at how much of it is a leap into the unknown.

“Every time we train a model, we go on this magical adventure where we’re like, ‘How are we gonna do that?’” she said.

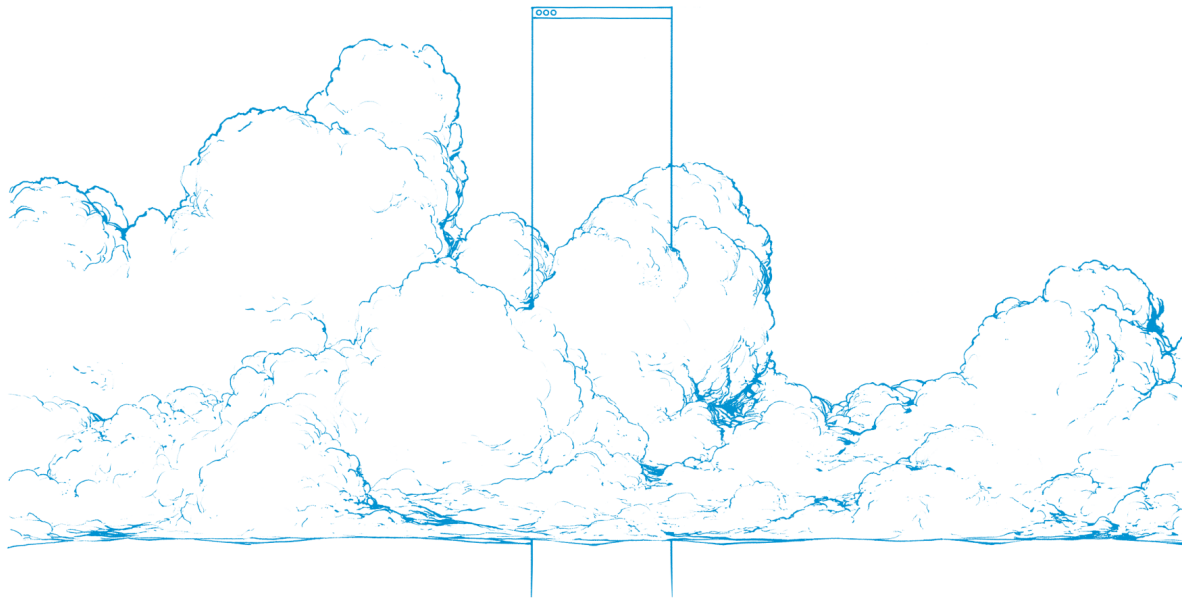
This sort of multidisciplinary team, with product people and researchers working side by side, is somewhat unusual in AI. AI has historically been a research-driven field with the best researchers fetching multimillion-dollar **LeBron James**–sized contracts for their services while everyone else is paid industry-standard engineering salaries. (AI labs have

long tried to counter the exalted superstar dynamic by giving most technical employees, whether they are researchers or not, the job title “Member of Technical Staff.”)

Working alongside the product team gave Glaese and the other researchers a chance to see how their models were being used—and hear what people wanted them to do next. Product people could ask, “Wouldn’t it be amazing if the model could do this thing?”^[F6] Then researchers could bring it to life. It was Sottiaux’s bootstrapping loop again, this time running between a model and the product built on it.



OpenAI finally had its Codex team—product, design, and research—all working together. Now they just had to decide what to build.



The cloud, and its limits

Authored imagined transition; not an authentic model reasoning trace.

Thinking...

I could try the terminal, the editor, or the cloud...

Which approach fits the models I have?

Our researchers work in the cloud. Could other developers work there, too?

At Anthropic, a promising approach had already begun to take off.

On February 24, 2025, the company released Claude 3.7 Sonnet and, with it, a research preview of Claude Code. Unlike Cursor and similar tools, Claude Code had no built-in code editor. It ran in the terminal and worked directly on the files on your own computer: You told it what you wanted, and it read the code, made the changes, and ran them.

Because it ran on your own computer, its work landed in the same files you were already working in, with the same tools you already had installed, and with the context of everything you'd done together so far. You could interrupt it mid-task. You could also bring lessons learned from one session into the next so that your system improved and compounded over time—something that Codex Cloud, with its ephemeral cloud computers, didn't allow for.

Developers took to Claude Code quickly. At Every, for example, **Edmar Ferreira** wrote in a March 2025 Vibe Check that “Claude Code is better than anything that I have used on the agent front.” The rest of the team had begun to agree.

By the spring of 2025, the merged Codex team—Embiricos, Sottiaux, Glaese, and a handful of others—set its sights on a different vision for the future of software engineering: Codex Cloud. They envisioned an agent in the cloud, with access to its own computer, designed for asynchronous work that would allow users to delegate coding tasks from their computer or their phone. They gave themselves roughly seven weeks to build and launch it.[14]

You could click Codex in ChatGPT’s sidebar and open a workspace where you could describe a task and hit “Code.” The assignment peeled off into a queue, clearing the prompt box for the next one. Codex copied your codebase into an isolated cloud machine, edited files, and ran tests and linters—tools that scan code for common mistakes. It returned a summary, a diff of the changes, and a button to open a pull request.

Codex Cloud launched on May 16, 2025, nearly three months before the night I met with McCallum and Embiricos in Mission Bay. It was “very successful internally,”[15] Sottiaux told me. Outside OpenAI, its launch drew interest but little sustained use.[16]

To get it working, developers had to recreate their laptop’s software, permissions, and settings in the cloud[17]—a process that could take hours, as you needed to copy over credentials and install critical software packages. Once set up, it could deliver on a well-specified task. But you couldn’t interrupt it to correct its course, and even a follow-up prompt started a fresh cloud computer.



“It’s a tool you’ll reach for if you’re a tech lead adding features or fixing bugs on an existing codebase. If you’re trying to vibe code a new one-person billion-dollar SaaS company, look elsewhere,” I wrote in our launch-day Vibe Check.

Inside OpenAI, standardized development environments had made setup automatic, and the researchers had clearly defined work to hand off. The team’s own working environment had hidden how much they were asking of everyone else.

The product wasn’t the only thing holding Codex Cloud back. “It was too early because the models weren’t smart enough,”^[18] Embiricos said.

“We were kind of approaching it with this more future vision^[F7],” Glaese told me. “We didn’t do this like, ‘Let’s go to the people where they are,’” which was “on their laptop.”

Why then did OpenAI skip the terminal form factor that began to work for Anthropic?

“We didn’t skip it,” Sottiaux told me in our cafeteria conversation, referring to an unreleased internal prototype. “We built a version of that and then discarded it after a month to go for the full cloud version^[F8].”

“Why?” I asked. I knew somewhere out the large floor-to-ceiling windows on all sides of me, if I really looked, I could probably find Anthropic’s office on the horizon. I could certainly walk there without much trouble. San Francisco is a small town.

“Because we were optimizing for what we thought we would benefit from the most if we were willing to invest a lot in the setup,”[19] he said.

Behind that answer was a belief I’d heard over and over from people inside of OpenAI. To them, the ideal form of an AI agent has always been one that works autonomously for you in the cloud, that doesn’t stop responding when you shut your laptop lid. Brockman had said the same thing in our interview: “If you project into the future, the right form factor is an AI in the cloud that has its own laptop. And so we actually built that.”[20]

That choice was also a truce with model limits. The team built the first Cloud product around models that could spend longer working on a task,[21] rather than respond to a programmer moment by moment.

OpenAI did, in fact, release a small, open-source Codex command-line interface^[F9] (CLI) reminiscent of Claude Code on April 16, exactly one month before Codex Cloud. It was one more of Codex’s many beginnings.

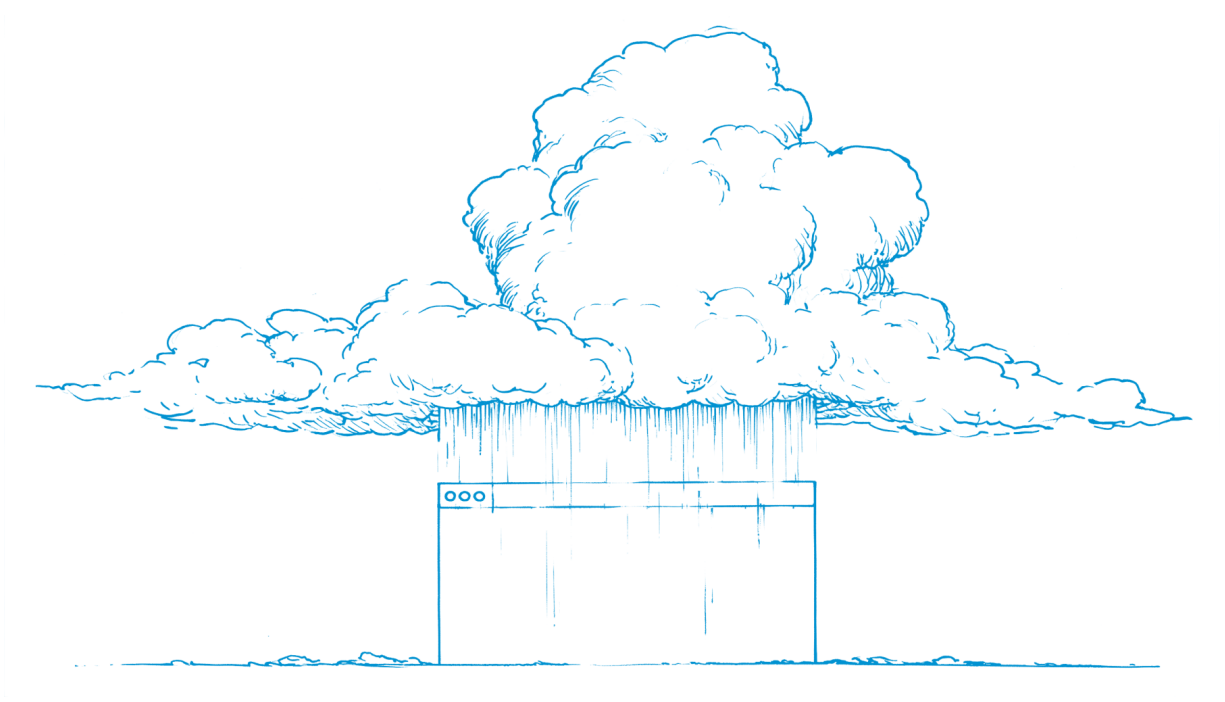
Embiricos described that public CLI as a separate side project built “heroically by, like, two people.”[22] But this CLI was not where OpenAI had placed most of its resources—or the specialized model it was training for Codex Cloud. “You couldn’t even use the Codex model in the CLI,” he said.

Running an agent on a user’s computer also raised safety questions. A coding agent could do more than suggest code: It could edit files and run commands on the user’s computer. “As a company, we have a very conservative safety stance,”[23] Embiricos said. “We actually had to do a lot of work to figure out how to get comfortable deploying that level of capability.” Early users instead had to run the CLI with a general-purpose model—o4-mini by default—that hadn’t been built to work as a coding agent. The CLI existed, but the attention of the team—and its best model—was on Cloud.

The problem was it didn’t work. The Cloud product failed to find product-market fit. The team’s engineering manager left for Cursor, the front-runner in AI coding with a valuation of \$9 billion. Around this time, OpenAI reportedly reached a \$3 billion agreement to acquire Windsurf, Cursor’s closest competitor. That deal would fall apart in July.

“The most challenging time was when we felt like we didn’t know what to do. Or we knew what to do, but it was really hard to see that [progress] actually happening,” Embiricos told me.

For the first time, the Codex team—and to an extent OpenAI’s larger coding ambitions—were lost in the wilderness.



A different path

Authored imagined transition; not an authentic model reasoning trace.

Thinking...

Hmm, this worked in our environment. Why isn't it working for users?...

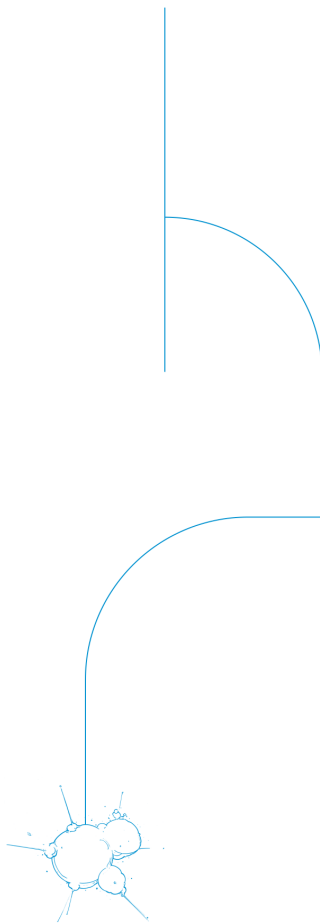
Maybe I'm trying to fix the wrong thing...

Going back to the local approach I set aside...

There's something here. But I'll need to rebuild it...

What makes a reasoning model powerful is its ability to backtrack. If it believes a bug lives in a particular part of the code and rewrites it, only to find the bug is still there, it does not typically continue marching confidently in the wrong direction. It can reconsider its assumptions, broaden its search for solutions, and decide on a new, more promising path based on what it's learned.

This ability to use mistakes to course-correct is what allows reasoning models to find answers that were not obvious at the start.



In the aftermath of the Cloud launch, in the late spring and summer of 2025, the Codex team was ready to try another path. At the time, the project occupied what designer **Ed Bayes**—who had recently joined the team—recalled as “the backwaters” of OpenAI. “If you wanted to be the sexy designer,” he told me, “you work[ed] on ChatGPT.”

Here, into this neglected outpost, Sottiaux ascended as the team’s engineering manager. “I took over the team, and then I changed the direction quite a bit,”[24] he told me. “We pivoted to local.”

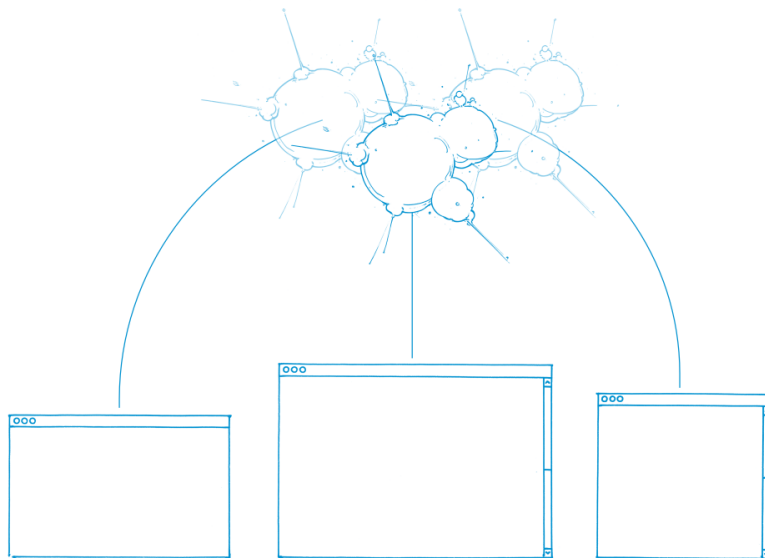
“Local” meant running the agent on the developer’s own computer, the way Claude Code did, instead of in the cloud. They started with the local Codex CLI that, as Embiricos mentioned, had been “heroically” built as a two-person side project. Rather than start adding features to it immediately, which would quickly generate momentum, Sottiaux took a different tack. He decided to rewrite the agent.[25]

“We were kind of slow to have really good CLI features,”[26] he told me, “because we were so busy developing the agent.”

The CLI was the interface users interacted with, but Sottiaux’s priority was the machinery underneath: the agent that read files, ran commands, and more. He wanted to improve it and separate it from the terminal interface so it could live anywhere—inside a code editor, a desktop or mobile app, the cloud, or ChatGPT itself.

The team rebuilt the agent in Rust,[27] a language designed for systems where speed and safety matter. The same machinery could then power both the terminal CLI and, later, the editor extension.

What they still lacked, though, was a model good enough to drive it.



While the Codex team was rebuilding its agent under Sottiaux’s leadership, **Michelle Pokrass** was working to make GPT-5 better at real-world programming.

Pokrass, who had led research on GPT-4.1, led post-training—refining an already trained model’s behavior so it could handle real programming work—with a focus on customers who had asked for better interactive coding.[28]

A coding model can be like a student with a perfect SAT score who has never held a job. OpenAI’s models had excelled at coding competitions, but those results hadn’t carried over to everyday engineering. “They never had seen a messy codebase or how a programmer might interrupt it halfway through^[F10],” Brockman told me.

To give the book-smart models real-world experience, Pokrass and her team sat with programmers and watched them work to see where the models broke down and turned those failures into repeatable training tasks.



That was how OpenAI arrived at the model—and the moment—that had brought me to that conference room in Mission Bay in August 2025.

Seen from inside the Codex team’s tumultuous summer, the caution I had felt in GPT-5 that night made sense. Burned by trying to leap straight to full autonomy with Codex Cloud and guided by customers asking for a better interactive collaborator, OpenAI had built a model that excelled at working *with* a human programmer step by step. It could understand how the pieces of a messy codebase fit together, explain its plan, investigate a bug, and pause to ask before stepping out of line.

Compared with Claude Code, GPT-5 in the Codex CLI still felt behind to us at Every. Kieran wrote in our launch-day Vibe Check, “GPT-5 is a Sonnet 3.5 killer, not a leap into the future.” But for the vast majority of software engineers at the time, a careful

collaborator was what they wanted. Delegating production code to autonomous agents was still commonly snorted at by programmers used to writing every line by hand.

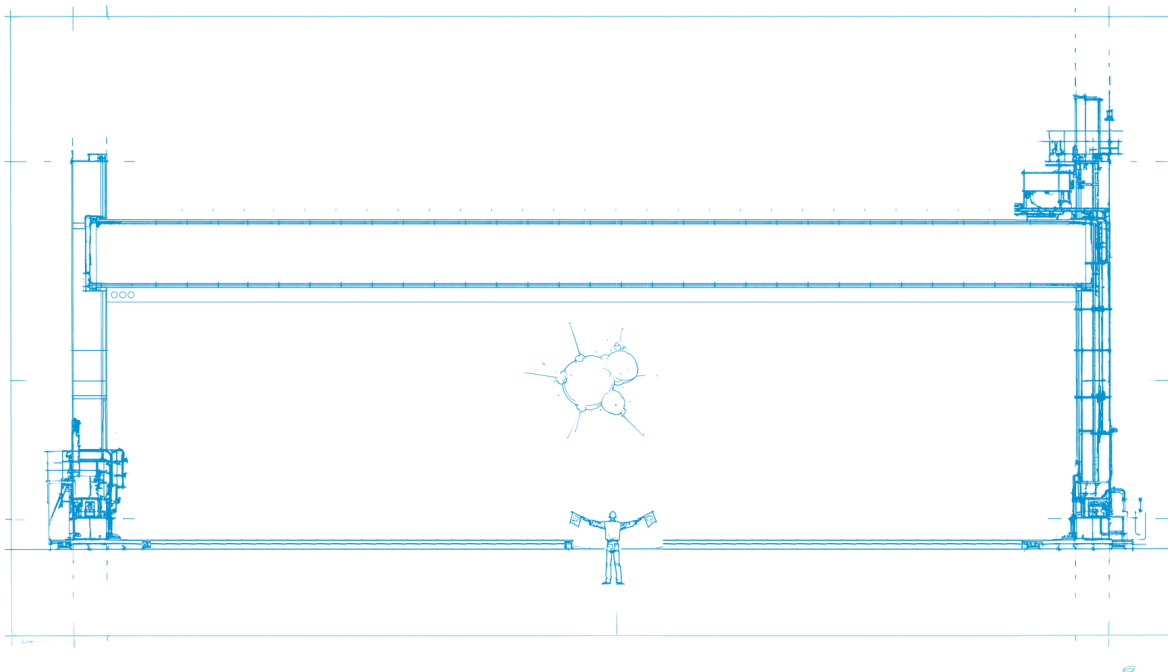
“I don’t let it drive my code,” **David Heinemeier Hansson**, the creator of Ruby on Rails and an influential voice in the engineering community, said around the time of the GPT-5 launch. “The joy of a programmer, of me as a programmer, is to type the code myself.” Besides, when the agent drove, he said, “I can literally feel competence draining out of my fingers.” (By January 2026 he would write that AI agents were “fully capable of producing production-grade contributions to real-life code bases,” and at Rails World in September he declared the end of writing code by hand at his company.)

When the Codex team followed up the GPT-5 launch in late August with an extension for VS Code and Cursor, those developers flocked to it. Their enthusiasm gave OpenAI every reason to keep improving the extension. Each new user and feature request pulled the team toward helping programmers work step by step and away from building agents that could take on whole tasks.

For Embiricos, though, the same launch meant the team could pursue their larger ambitions. After months in the wilderness, the rebuilt local agent finally had a model capable of driving it.

Once they shipped GPT-5 with the revamped CLI, he told me, “it created this self-fulfilling flywheel.”

He remembered thinking, “Finally we can fight now.”



The desktop bet

Authored imagined transition; not an authentic model reasoning trace.

Thinking...

The CLI and editor extension are finally taking off...

Customers love the sidebar, and Claude Code is the product to beat in the terminal...

Let's double down on the editor extension and catch up in the terminal...

[Interrupted by Tibo]

With the Codex CLI and VS Code extension exploding in popularity, the logical choice would be to continue to invest in both of them.

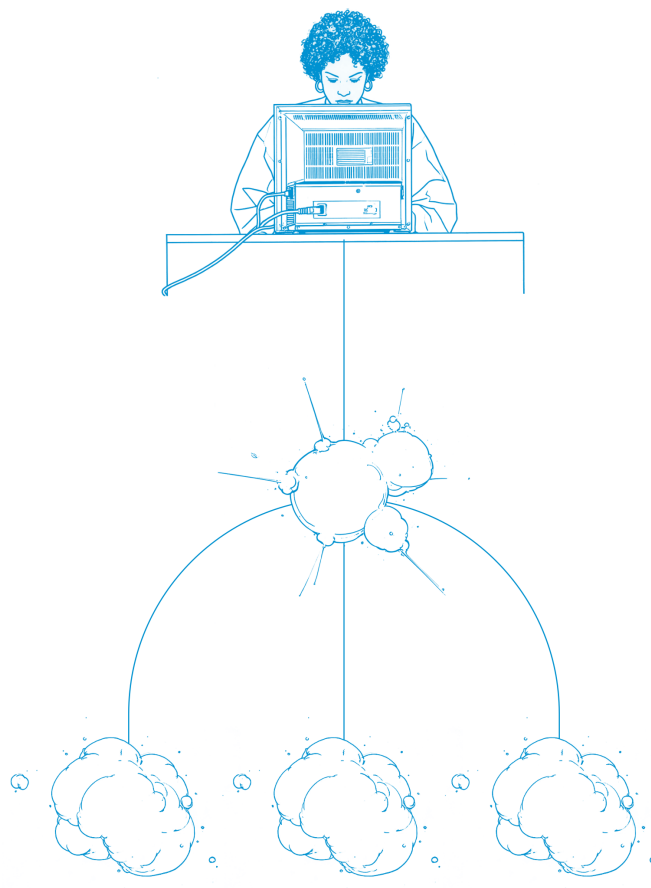
Instead, the Codex team did something unexpected.

“We decided we should continue maintaining the CLI, but the next biggest immediate investment should be a native desktop app^[F11],” Sottiaux told me.

Inside OpenAI, he could see the models completing longer tasks and drawing on information from outside the codebase, with voice and images on the way. People would

soon be running several agents at once and spending their whole day with them. “It has to be this very rich interface if you’re gonna spend all your time in it,” he said. Trying to juggle half a dozen agents communicating in text, images, voice, and video inside of a text-only window built in the 1970s seemed silly to him.

For Embiricos, a local desktop app also pointed toward the workflow that had eluded them with Codex Cloud, where users could not interrupt the agent mid-task. With an app on their own computer, he thought, people could start by working closely with the agent and then, like onboarding a teammate, learn which tasks they could leave to it.[29] The same interface could eventually take Codex beyond developers to people who would never open a terminal.



To help design and build it, Sottiaux brought in **Andrew Ambrosino**, a designer, programmer, and former startup founder.

Ambrosino reminds me of B.J. Novak’s character in *The Office*: dark hair, thick eyebrows, and a small, tight-lipped smile. In person and online, he gives the sense of perpetually

being inwardly amused^[F12]; whether or not anyone else joins him in the joke is a consideration but often appears to be somewhat secondary.

Anyone following him for insights into the future of artificial intelligence might scroll his feed and find a post with 617 likes consisting entirely of a single word: “ai.” Another post is, mysteriously, “tibo” (1,000-plus likes).

(If you’re wondering what he meant by “tibo,” I can report that Sottiaux seems to be consistently on his mind. When I asked him whose comedy he admires, he replied, “Tibo is my comedy idol.” I leave it to the reader to decide how much this clarifies.)

When Ambrosino joined the small group working on the app, they knew they wanted to make a desktop app, but not what it would look like.[30] All the team knew was that they wanted to make something they would use themselves.

Their first mandate, Ambrosino told me, was to make the app their main tool for everyday work.[31] “We knew we wanted a surface that could be a daily driver, but [one] that was more lightweight than an IDE,” he said.

As Claude Code took off, a handful of apps had begun experimenting with what a home for coding agents might look like outside the terminal. Conductor, a Mac app for running coding agents, had a sidebar of tasks, each with its own agent and chat where you could give instructions and follow the work. ChatGPT supplied another reference point: a list of conversations down the side and a single thread in the middle. The Codex app took “more inspiration from chat than it does VS Code,”[32] Ambrosino said of their design process. Anthropic, too, by that point, had a Claude desktop app.

The team worked quickly, answering the basic questions like how users would keep track of their work (each task had its own conversation with an agent), how the sidebar should be organized^[F13] (conversations grouped by project), and how the basic interface should be laid out (a sidebar of tasks beside the selected conversation).

Then came a harder question: When an agent got something wrong, could you fix the code yourself by hand inside of Codex? The team decided on a risky answer: No built-in code editor. If you wanted to make any changes to your code, you *had* to ask the agent.

“We specifically wanted it to be a different thing [from a code editor],” Ambrosino told me. Let developers type code in the app, and they’d expect everything else their editor did—undo, search, keyboard shortcuts—and thereby notice what Codex lacked. Meeting, let alone exceeding, these expectations would have required a massive effort from their team of four.

Codex couldn’t win by being a bad code editor. Coding with agents had to be so good that developers wouldn’t miss the editor.

The only problem was that they were still building a replacement for VS Code *with* VS Code.

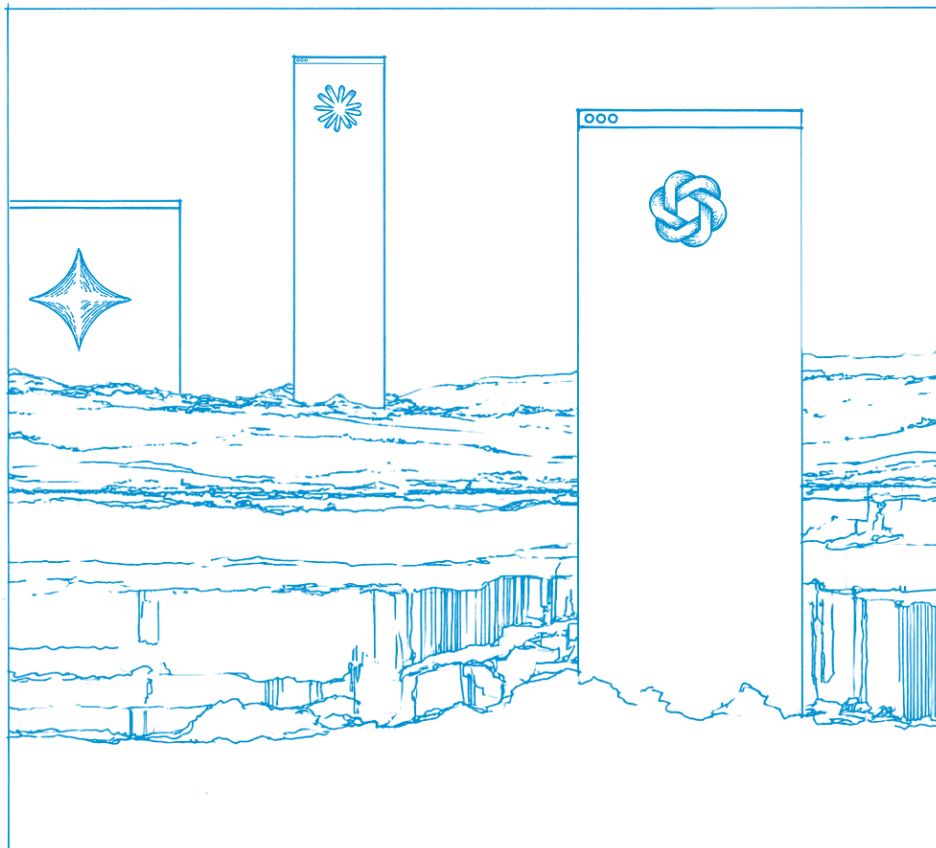
“Every day Alexander would come in and be like, ‘Let’s go around the table. Are you maining this app? Are you maining this app?’”[33] said Ambrosino.

At first the answers were all “no”—none of them was using Codex as their main coding tool. Without a code editor, even a small correction required a conversation with an agent.

Dominik Kundel, who works in developer experience, wrote in an early Slack message that “it feels cumbersome to have to prompt Codex” just to replace a sentence.

Even Ambrosino had doubts: “I think ‘no file editing’ is the launch scope I’m most skeptical about,” he wrote in November to the team.

The outside market made the decision to leave out a code editor look even riskier. In November of 2025, Google released Antigravity, another coding tool built on VS Code, combining AI agents with a full editor.



“We just kinda looked at each other,” Ambrosino recalled. “Did we make a huge mistake by not forking VS Code?”^[F14] We felt very conflicted on this, and I don’t think the team even agreed with each other.”

The models powering the app were a problem, too. OpenAI had launched its dedicated Codex model line in September, but by November it still wasn’t good enough for the desktop product they were building. Even a fully featured app would have been “unreleasable because of the quality of the models at that point,”[34] Ambrosino told me.

Over the next several weeks, that began to change. On November 19, OpenAI released GPT-5.1 Codex Max, which could code for hours at a time. GPT-5.2 Codex followed on December 18, bringing further improvements in making changes across large codebases.

Glaese worked directly with the rest of the Codex team to further shorten the feedback loop between developer feedback and model improvements. Crucially, they focused on making Codex work better for the people using it^[F15], rather than just raising the models' scores on popular benchmarks. And because the Codex model line was separate from the main ChatGPT model, they could move more quickly without worrying about how those changes would affect the 800 million people using ChatGPT.

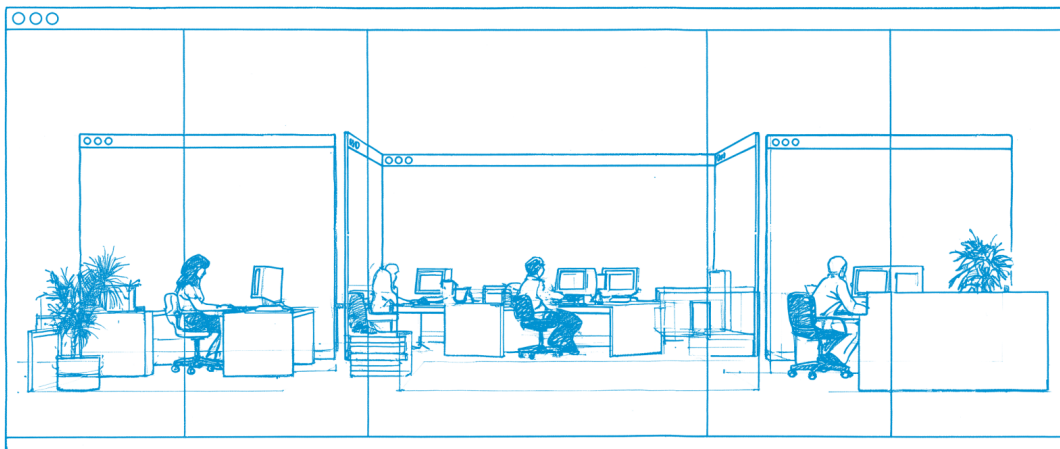
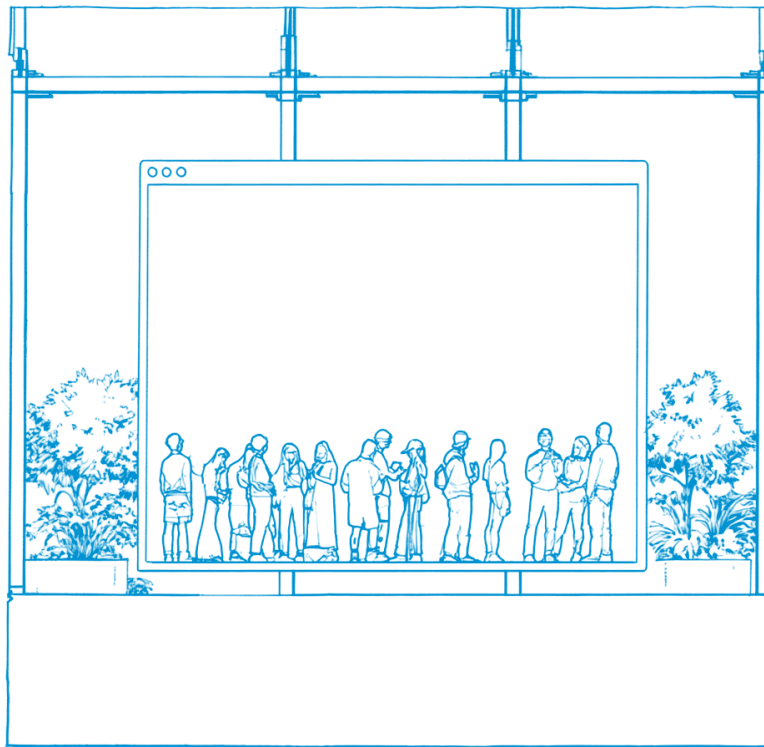
By December, more of the team was using the Codex app over their code editors. "I was originally craving a code editor," Kundel wrote in Slack. "Now most of my changes I just leave inline comments and it's rapid fast." Ambrosino found that he still had to steer the model, but he no longer needed to edit its code by hand.^[35] Glaese, too, became aware that the sentiment had changed internally. "5.2 was a turning point where I was like, 'Wow, I'm very excited about what this can do^[36] for me,'" she said.

Across the industry, the holidays became an inflection point, as developers spent Christmas break building entire apps end to end with Anthropic's Opus 4.5 and OpenAI's GPT-5.2 without touching a line of code.

By early January the whole team was using the Codex app. Now they needed to find out whether the rest of OpenAI would use it, too. If not, Ambrosino said, it was "back to the drawing board."^[37]

By February the Codex desktop app was gaining wide adoption on the engineering and research teams. OpenAI researcher **Noam Brown** said he had "fully switched" to it, and Brockman, a devoted user of the terminal and the Emacs editor, had, too. He said returning to his old workflow "felt like going back in time." Altman would describe it later as "the most loved internal product we've ever had."

The Codex team had built something that developers inside of OpenAI loved. But Codex Cloud had also been embraced inside OpenAI and flopped once it was launched. Would this time be different?



The Codex app

Authored imagined transition; not an authentic model reasoning trace.

Thinking...

A lot of people here are using it...

But I've mistaken that for success before...

This time it works where the user already works...

Time to test it outside...

The Codex macOS desktop app was released to the public on February 2, 2026. It looked more like a chat app than a code editor: A sidebar held ongoing and historical tasks, and clicking one showed what the agent had done and let you send it further instructions.

You could ask one agent to fix a bug, send another off to build a feature, and move between them to answer questions or review their changes. Each could work on its own copy of the code without overwriting the others' work. You could stay close and steer, or leave several assignments running and come back when they needed you.

At the time, none of this was especially groundbreaking. The Claude desktop app, for example, had allowed technical and nontechnical users to do the same for months. But the Claude desktop app had been built step by step, starting with Chat, then adding Code, and then adding Cowork—which made its interface feel confusing and tacked together. It was hard to figure out which tab to use for which task and what set of features or permissions would be available in each.

The Codex app bore none of these scars. When I used it for the first time—alongside the new Codex model released a few days later, GPT-5.3 Codex—it was fast, simple, and totally agent-native. There was a fluidity to the experience, a feeling you get when all of the details of a product are right. Because of the team's early insistence on excluding a code editor, they made sure that the Codex agent could do almost anything inside of the app that a user could do. That meant you didn't have to dig through settings to, for example, connect a new tool. Instead, you could just ask the agent.^[F16]

Partly, this was a latecomer advantage. Because the OpenAI team had started its desktop app after the Claude desktop app added tabs for both Claude Code and Claude Cowork, the Codex team could see a validated trajectory for their app.

GPT-5.3 Codex was also a turning point in OpenAI's model capabilities. According to Glaese, it was the first time they had started to branch out from programming-specific training for the Codex line. They still named it Codex, and she wasn't claiming it was better

at everything. But the team had begun asking what else a coding agent needed to know to be useful for all kinds of knowledge work^[F17].

The impact on my workflow was immediate: I switched from using Claude almost exclusively to using Codex for large portions of my work. On Proof, a document editor I was building, I watched Codex fix complex bugs autonomously without much intervention. Soon I was leaving it working overnight on difficult problems, and anytime I was faced with a particularly gnarly task, I went to Codex first.

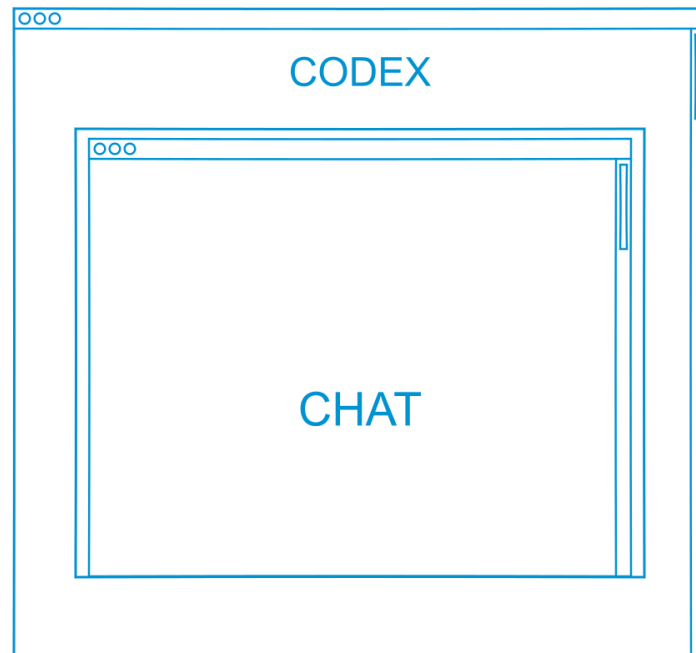
GPT-5.4's arrival in early March cemented Codex as my daily driver. It was still an expert coder, but now it explained itself clearly and better understood what I wanted. And because the Codex app had a clean and fast interface, I started to do all of my work inside of it.

By then many people at Every were using OpenAI's tools and models every day too—a reversal from just a few months earlier, when Claude was far and away the dominant choice. As we put it in our GPT-5.4 Vibe Check: OpenAI was back.

The same thing was happening far beyond Every. By March 19, OpenAI said Codex had passed 2 million weekly users, while overall usage had grown fivefold since the start of the year. In April, OpenAI added an in-app browser, so the agent could look things up and check its work on a live web page. It could also work across other Mac apps and show previews of documents, slides, and spreadsheets. Even someone who had never written code could ask for a presentation or a budget report and see the result without leaving the app.

By June, Codex had more than 5 million weekly users, and roughly one in five was a knowledge worker. On July 9, OpenAI made the Codex desktop app the new ChatGPT desktop app, with chat inside it.^[38] Under the hood, the new ChatGPT still tells your Mac its name is `com.openai.codex`.

Codex Cloud had shipped as a tab inside ChatGPT. Now ChatGPT runs on Codex.



Ed Bayes remembers walking out of the all-hands where the merge was announced, thinking about how far the team had come. What had started as "a few people in the corner" had now become "the future of ChatGPT^[F18]."

"This is like a piece of American corporate history," he told me.



The next curve

Ten months after the GPT-5 launch, I was back in a conference room with Alexander Embiricos.

“I’m just going full honesty here, right?” Embiricos asked.

“Please do,” I said.

It was early June, a month before the now red-hot Codex was to merge into ChatGPT, and we sat in OpenAI’s downtown Manhattan office. He wore a gray zip-up with the sleeves pushed up. His role had recently changed from product lead for the Codex agent to OpenAI’s head of enterprise product. Behind him was a whiteboard.

I had come to ask Embiricos how his team went on from Codex Cloud’s failure to building what would become the foundation of ChatGPT—and how they had avoided getting stuck in the pair-programming trap I’d worried about in Mission Bay. Most teams build on what’s working. The Codex team hadn’t. What had allowed them to keep moving?

Embiricos picked up a black marker and sketched Codex’s growth on the whiteboard. He started with a short upward curve for Codex Cloud. It was a nice spike, but it flattened out quickly.[39]

Then he drew a steeper, bigger curve for the CLI after GPT-5; another, higher still, for the editor extension. Last came 5.3 Codex and the desktop app, a curve that dwarfed the others.[40] The first mark now looked like a scratch near the bottom of the board.

The curves he drew on the whiteboard looked like a diagram from Christensen's *The Innovator's Dilemma*.

Christensen famously traced how steady advances in hard drives and processors drove the shift from mainframes to minicomputers to desktop PCs, laptops, and eventually phones—each leap in the underlying hardware enabling a simpler form factor that unlocked a larger, less technical market and overtook the last.

A pattern that took half a century inside of a succession of new companies in traditional computing had played out in miniature inside OpenAI in 10 months with a team of just a handful of people. Relentless leaps in the models kept outgrowing the interfaces built around them—forcing the Codex team to move from a code-editor sidebar, to a terminal, to a lightweight desktop app that stripped out the code editor entirely.

In Christensen's book, the team riding a winning curve almost never rides the next one; the success of the existing product suffocates whatever comes after it. The Codex team avoided that fate partly because OpenAI's executives had grown up in the startup world studying that exact trap. The company was structured to behave like a portfolio of parallel bets, and the Codex team was given space to iterate on its own product and model line outside of the strictures of ChatGPT's mammoth scale and user base.

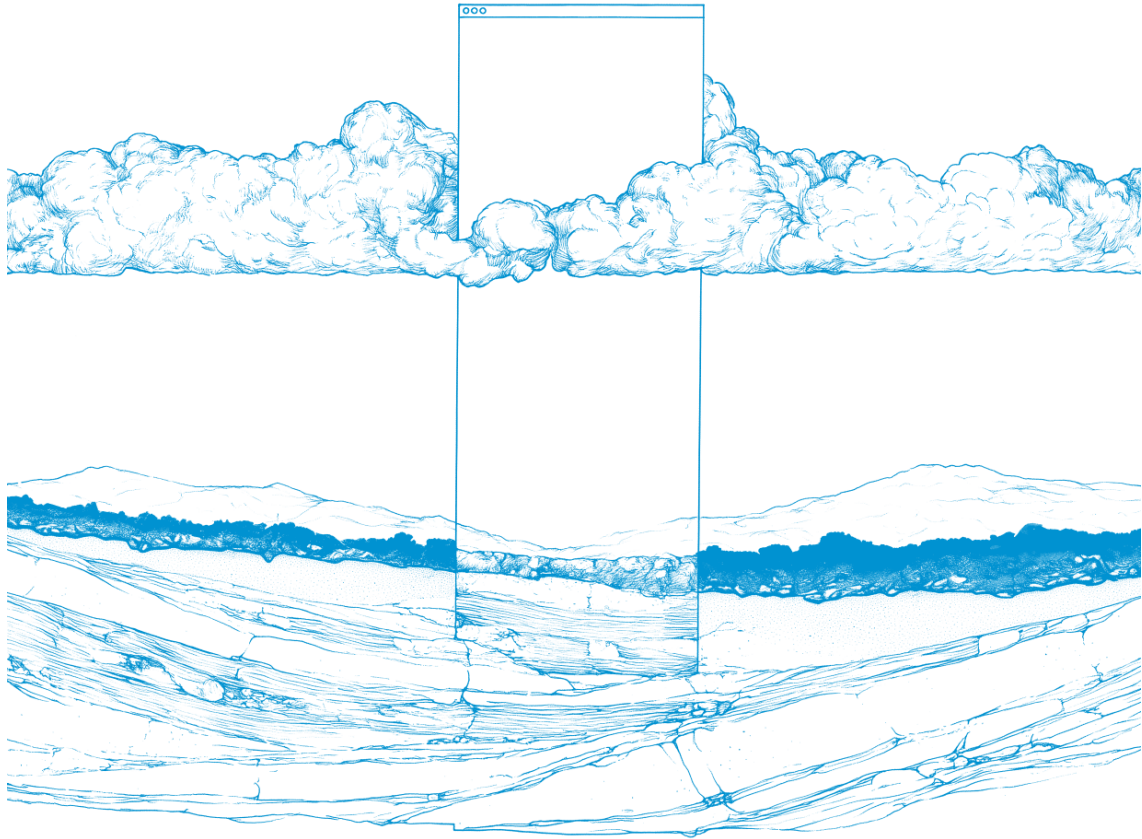
Using AI to build AI also changed the physics of product development. In a traditional company, starting a new product requires months of meetings to win budget and engineers—debates where the booming existing product almost always wins. At OpenAI, because the team was using Codex to build Codex, a handful of people could spin up a new surface in weeks.[41]

"It takes way less talking now to convince people,"[42] Sottiaux had told me.

Looking at the whiteboard, Embiricos had one more new curve to draw. Now that millions of people trusted the agent on their computers, the next step was to return to the vision that had failed the previous spring—an agent that kept working long after you walked away.

"We're gonna move to asynchronous," he said, "which means we're gonna have to rehoist to the cloud."

This time, I couldn't help but notice his laptop sitting on the conference table, its lid firmly closed.



Sources and notes

The linked interviews accompany this article online. Access follows the site's normal registration and membership rules. Numbered notes correspond to source markers in the article; F markers identify featured interview links.

- [1]** Codex's most-used surface [Alex interview](#).
- [2]** Codex has had many beginnings—starting with its name [Tibo interview](#).
- [3]** I saw someone running a reasoning model that was connected to a terminal [Alex interview](#).
- [4]** an AI-powered browser [Alex interview](#).
- [5]** I got involved with coding agents around December 2024 [Tibo interview](#).
- [6]** was to help us write infrastructure and do research faster [Tibo interview](#).
- [7]** trained internal coding agents on OpenAI's monorepo [Tibo interview](#).
- [8]** researchers were using those models to make the next generation [Tibo interview](#). Related context: Tibo describes the goal of accelerating research. This passage does not establish the full chain of one model producing a more powerful next model.
- [9]** coding became a top-level company goal [Greg interview](#).
- [10]** right “form factor” [Greg interview](#).
- [11]** Eventually, we thought we should combine forces with another effort, which was called Scientist Assistant [Tibo interview](#).
- [12]** We ended up merging two of them. And then merging another two of them. [Greg interview](#).
- [13]** she was responsible for the data used for model training and alignment [Mia interview](#).
- [14]** They gave themselves roughly seven weeks to build and launch it. [Tibo interview](#).
- [15]** very successful internally [Tibo interview](#).
- [16]** its launch drew interest but little sustained use. [Alex interview](#).
- [17]** developers had to recreate their laptop's software, permissions, and settings in the cloud [Tibo interview](#). Related context: Tibo describes high setup friction outside OpenAI. This passage does not establish how often setup failed.
- [18]** It was too early because the models weren't smart enough [Alex interview](#).
- [19]** Because we were optimizing for what we thought we would benefit from the most if we were willing to invest a lot in the setup [Tibo interview](#).
- [20]** If you project into the future, the right form factor is an AI in the cloud that has its own laptop. And so we actually built that. [Greg interview](#).
- [21]** The team built the first Cloud product around models that could spend longer working on a task [Alex interview](#).
- [22]** heroically by, like, two people [Alex interview](#).
- [23]** As a company, we have a very conservative safety stance [Alex interview](#).
- [24]** I took over the team, and then I changed the direction quite a bit [Tibo interview](#).
- [25]** He decided to rewrite the agent. [Tibo interview](#).

- [26] We were kind of slow to have really good CLI features [Tibo interview](#).
- [27] The team rebuilt the agent in Rust [Tibo interview](#).
- [28] with a focus on customers who had asked for better interactive coding [Alex interview](#).
- [29] like onboarding a teammate, learn which tasks they could leave to it [Alex interview](#).
- [30] not what it would look like. [Andrew interview](#).
- [31] make the app their main tool for everyday work [Andrew interview](#).
- [32] more inspiration from chat than it does VS Code [Andrew interview](#).
- [33] Are you maining this app? Are you maining this app? [Andrew interview](#).
- [34] unreleasable because of the quality of the models at that point [Andrew interview](#).
- [35] still had to steer the model, but he no longer needed to edit its code by hand. [Andrew interview](#).
- [36] Wow, I'm very excited about what this can do [Mia interview](#).
- [37] back to the drawing board [Andrew interview](#).
- [38] with chat inside it [Tibo interview](#).
- [39] He started with a short upward curve for Codex Cloud. It was a nice spike, but it flattened out quickly. [Alex interview](#).
- [40] Last came 5.3 Codex and the desktop app, a curve that dwarfed the others. [Alex interview](#).
- [41] a handful of people could spin up a new surface in weeks [Tibo interview](#). Transcript note: the capture says "free agents"; an annotation suggests "pre-agents." That wording needs checking. The four-to-five-month figure is Tibo's estimate, not a measured result.
- [42] It takes way less talking now to convince people [Tibo interview](#).
- [F1] This made what I had come to say a little awkward. [Alex interview](#). In this later interview, Dan revisits his criticism of the GPT-5 launch and Alexander gives a different recollection. This is not a recording of their original Mission Bay conversation.
- [F2] the models at the time were too slow and clumsy at reading a user's screen [Alex interview](#).
- [F3] There was no decision to split off the model line [Tibo interview](#).
- [F4] sometimes feels chaotic and wasteful [Andrew interview](#).
- [F5] Who wants to work on the team that built the third-best agent product inside of OpenAI? [Greg interview](#). Greg answers Dan's question about motivating people when a new project gets the attention. The article's rhetorical question opens that related conversation; it does not claim three products were formally ranked.
- [F6] Wouldn't it be amazing if the model could do this thing? [Mia interview](#).
- [F7] We were kind of approaching it with this more future vision [Mia interview](#).
- [F8] We built a version of that and then discarded it after a month to go for the full cloud version [Tibo interview](#).
- [F9] open-source Codex command-line interface [Tibo interview](#).
- [F10] They never had seen a messy codebase or how a programmer might interrupt it halfway through [Greg interview](#).

[F11] We decided we should continue maintaining the CLI, but the next biggest immediate investment should be a native desktop app [Tibo interview](#).

[F12] perpetually being inwardly amused [Andrew interview](#). This exchange illustrates Andrew's humor. It is not the source of the nearby social posts or the comedy-idol quotation. The Pets usage figure is spoken by the unidentified OpenAI participant.

[F13] how the sidebar should be organized [Ed interview](#). Ed's doubling estimate is his design rationale, not a measured productivity result.

[F14] Did we make a huge mistake by not forking VS Code? [Andrew interview](#).

[F15] they focused on making Codex work better for the people using it [Mia interview](#).

[F16] Instead, you could just ask the agent. [Andrew interview](#). Katie describes asking Codex to configure her working environment. This exchange illustrates that workflow; it does not establish that the agent can control every part of the app or prove why leaving out an editor enabled it.

[F17] the team had begun asking what else a coding agent needed to know to be useful for all kinds of knowledge work [Mia interview](#).

[F18] the future of ChatGPT [Ed interview](#).

Illustration notes

Authored editorial illustration of a model inside an application environment; not a literal model architecture or a correctness guarantee.

Three illustrated approaches across model generations, not one conversation or authenticated model reasoning.

Source-derived conceptual feedback loops, not automatic self-training or a literal organization chart. Original authored frame inventory is retained.

A conceptual illustration of parallel exploration and convergence, not a literal map of teams, people, dates, or organizational mergers. Figma room placeholders use existing anonymous office artwork. Historical native frames remain in the inventory but are no longer displayed.

Read-only Figma storyboard 522:30470; exact supplied image layers. Constant registered computer scale, uniformly growing/converging clouds, fully occluded binary handoff. The retained sequence is historical native inventory only, not displayed. Mobile marker 476:27171 precedes the exact setup paragraph. This diagram illustrates environment recreation, not the later future loop architecture.

A conceptual editorial illustration, not a recording of private model reasoning.

A conceptual editorial illustration of shared agent capabilities across interfaces, not three separately trained models being combined.

A conceptual illustration of working in real codebases; anonymous source figure, not a literal environment or a measured capability claim. Canonical source prose remains unchanged.

A conceptual editorial illustration of task delegation; the stationary parent and three different subagent drawings do not depict literal architecture or recorded private reasoning.

Editorial metaphor; pillar sizes do not represent measured revenue or market share.

User-supplied closing window/cloud/geological illustration. Complete image at native aspect ratio below the reporting footer; no added animation.

Stylized illustration; the figures are not portraits or a literal record of the room.

Conceptual desktop app host reversal, not all ChatGPT infrastructure or a company merger. Six unanimated authored poses become a continuous pen-window sequence at Mike's request; no source prose, dates or citations changed.

Conceptual research/product feedback before and after shared work. Five anonymous source room drawings consolidate into shared work. This is not a literal organization chart or automatic research.

[Read the article and explore its reporting online](#)