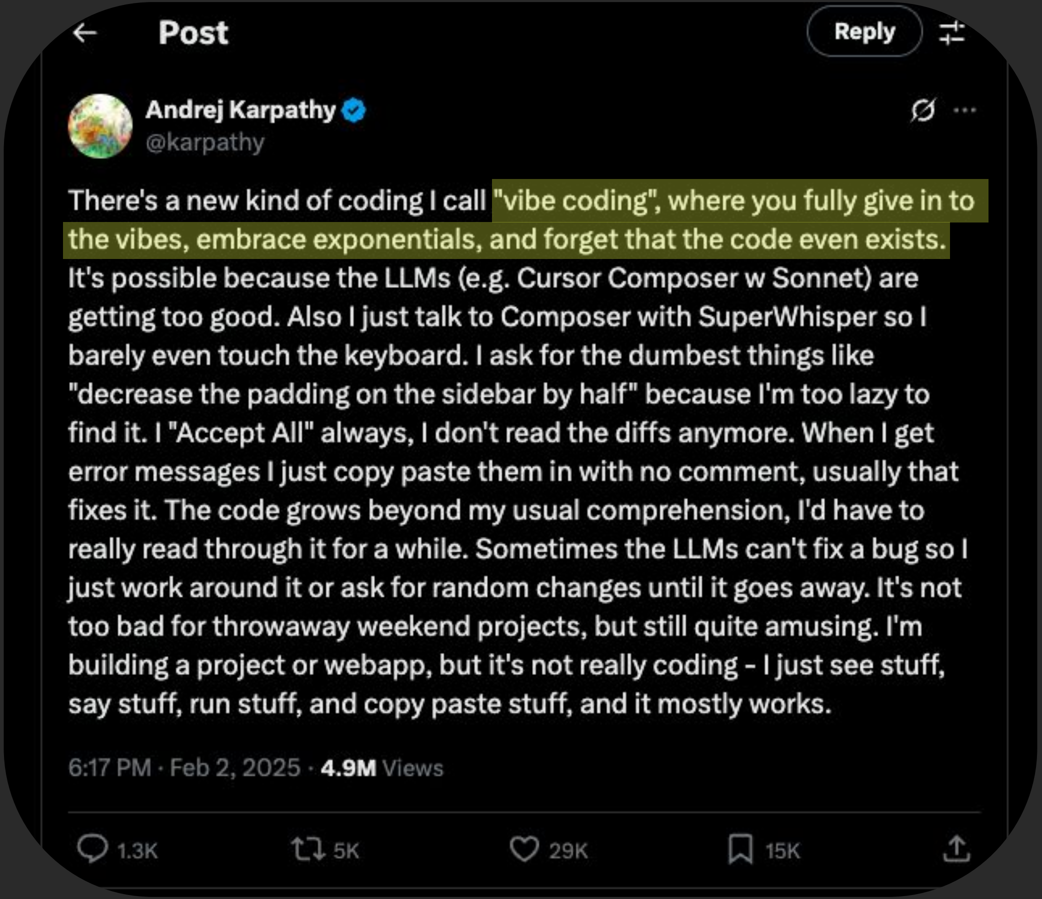


Post

Reply

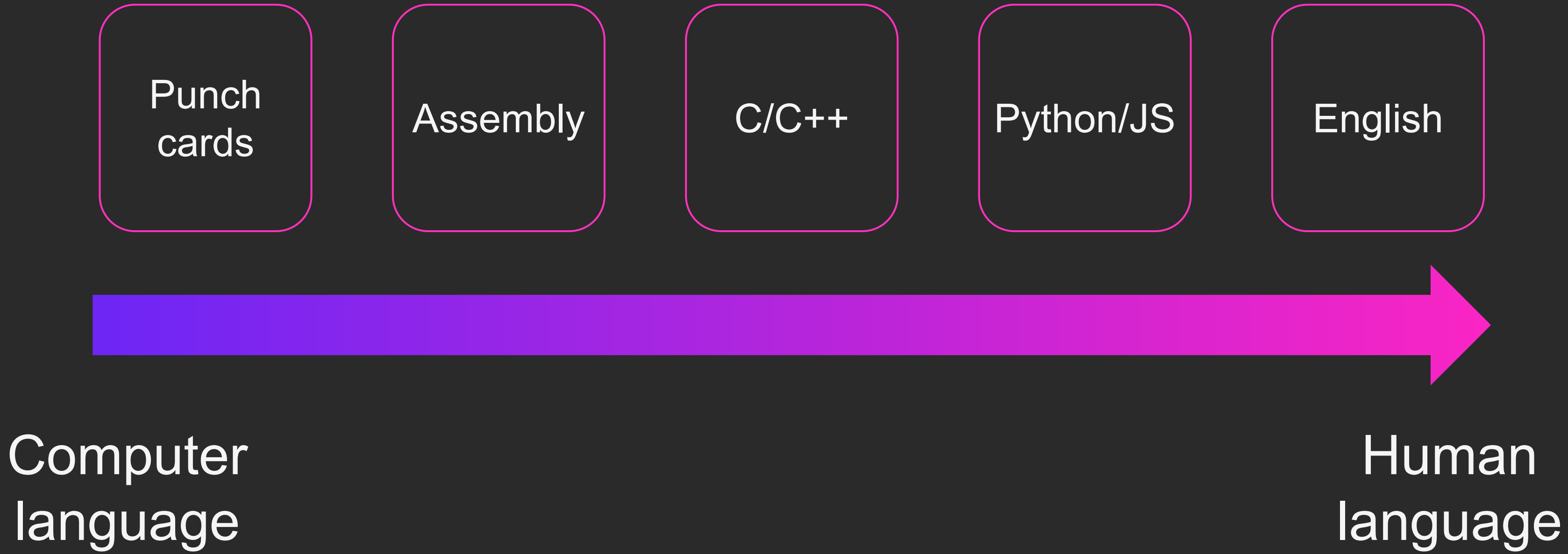
Andrej Karpathy
@karpathy

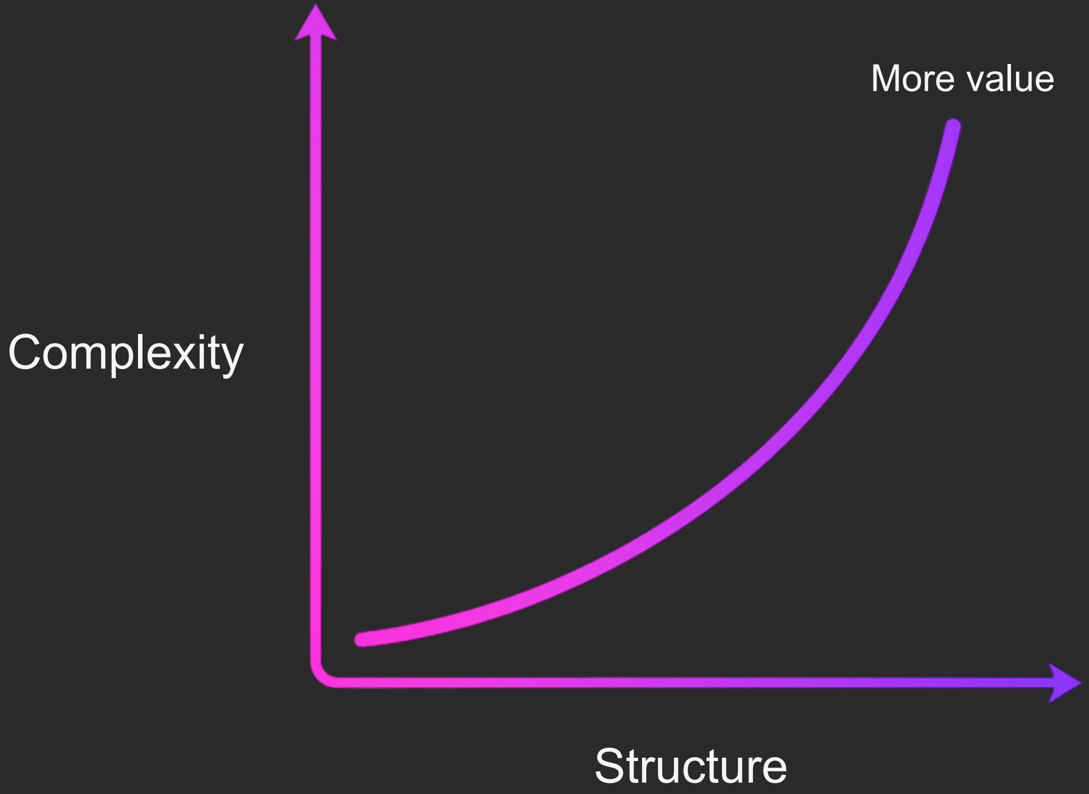
There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

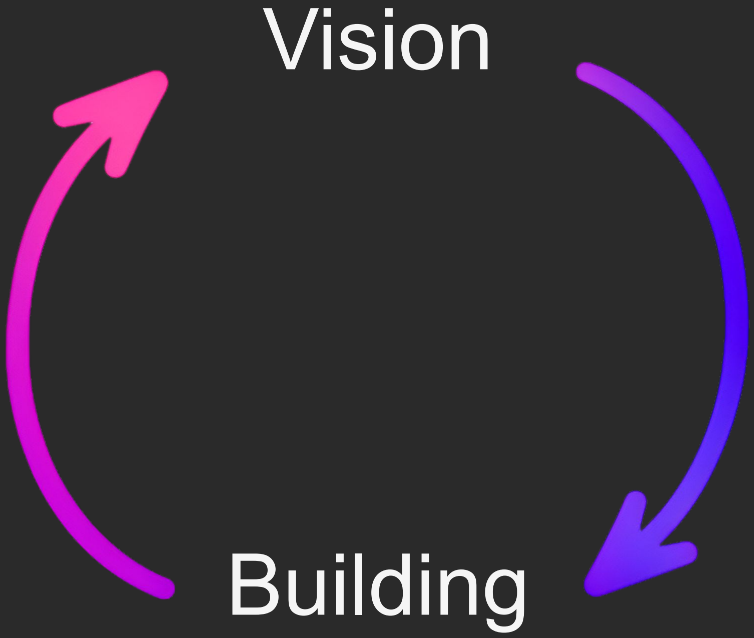
6:17 PM · Feb 2, 2025 · **4.9M** Views

1.3K 5K 29K 15K

Vibes


Punch cards
Assembly
C/C++
Python/JS
English
Computer language
Human language

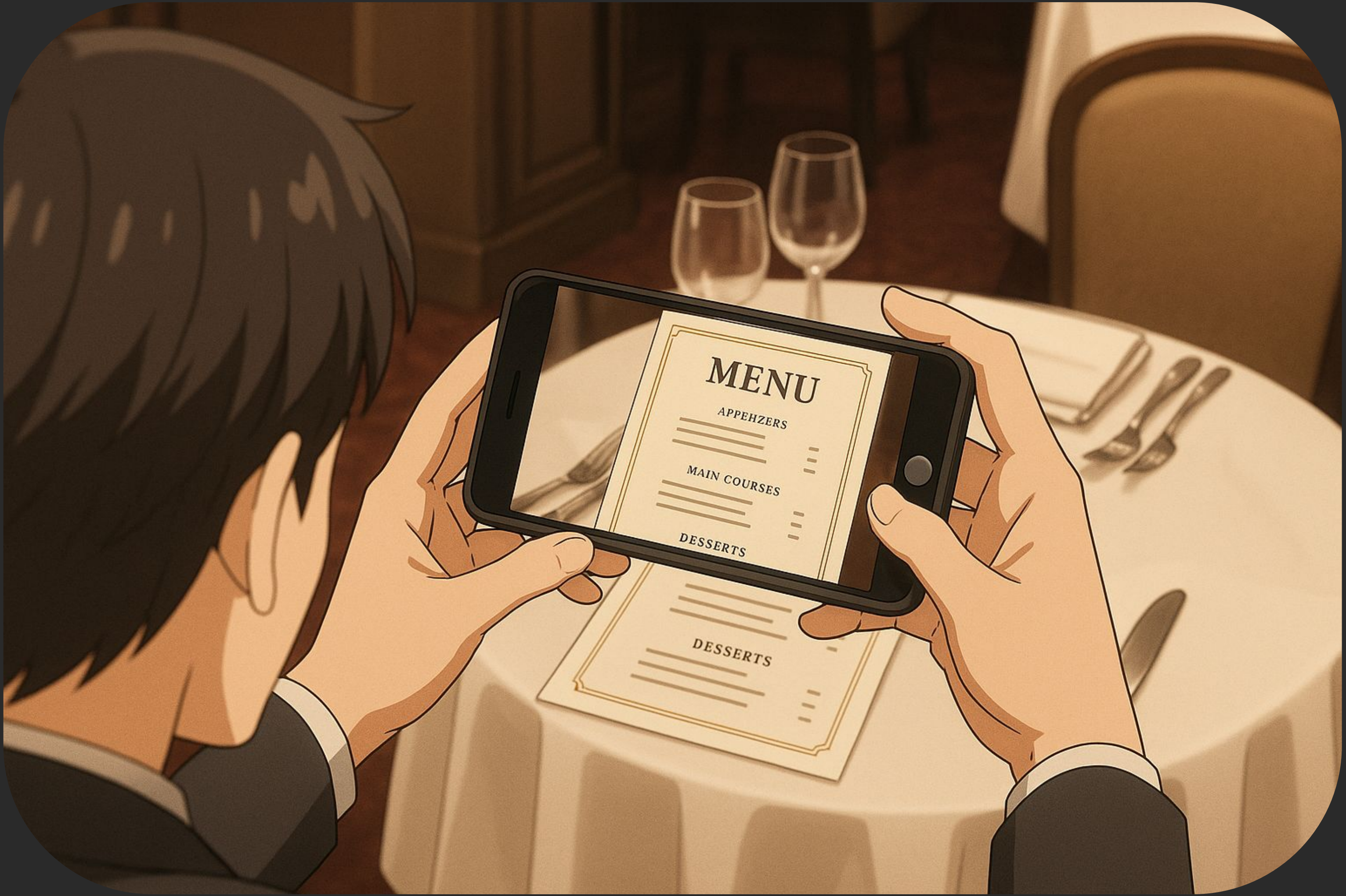

More value
Complexity
Structure


Vision
Building

Specification

Blueprint

Roadmap


MENU
APPEHZERS
MAIN COURSES
DESSERTS
DESSERTS

Fast & Smart

Context

Description

User

Job to be done

Research

APIs

Code libraries

Models

Ask me one question at a time so we can develop a thorough, step-by-step spec for an idea. Each question should build on my previous answers, and our end goal is to have a detailed specification I can hand off to a developer. Let’s do this iteratively and dig into every relevant detail.

Remember, only one question at a time.

Here’s the idea:

<IDEA>

Spec Demo

Now that we’ve wrapped up the brainstorming process, can you compile our findings into a comprehensive, **developer-ready specification?** Include all relevant requirements, architecture choices, data handling details, error handling strategies, and a **testing plan** so a developer can immediately begin implementation.

OpenAI o3

Gemini 2.5 Pro

Grok 3

Draft a detailed, step-by-step blueprint for building this project. Then, once you have a solid plan, **break it down into small, iterative chunks that build on each other.** Look at these chunks and then **go another round to break it into small steps.** Review the results and make sure that the steps are **small enough** to be implemented safely with strong testing, but **big enough** to move the project forward. Iterate until you feel that the steps are right sized for this project.

From here you should have the foundation to provide a series of **prompts for a code-generation LLM** that will implement each step in a **test-driven manner**. Prioritize best practices, incremental progress, and early testing, ensuring no big jumps in complexity at any stage. Make sure that each prompt builds on the previous prompts, and ends with wiring things together. There should be no hanging or orphaned code that isn't integrated into a previous step.

Make sure to separate each prompt section and use prompting best practices. **Use markdown.** Each prompt should be tagged as text using code tags. The goal is to output prompts, but context is important as well.

Make sure to **write out all the prompts**, shorten them if needed to ensure we have every prompt needed to build this entire project. Each **prompt should stand alone** and not reference other prompts

<SPEC>

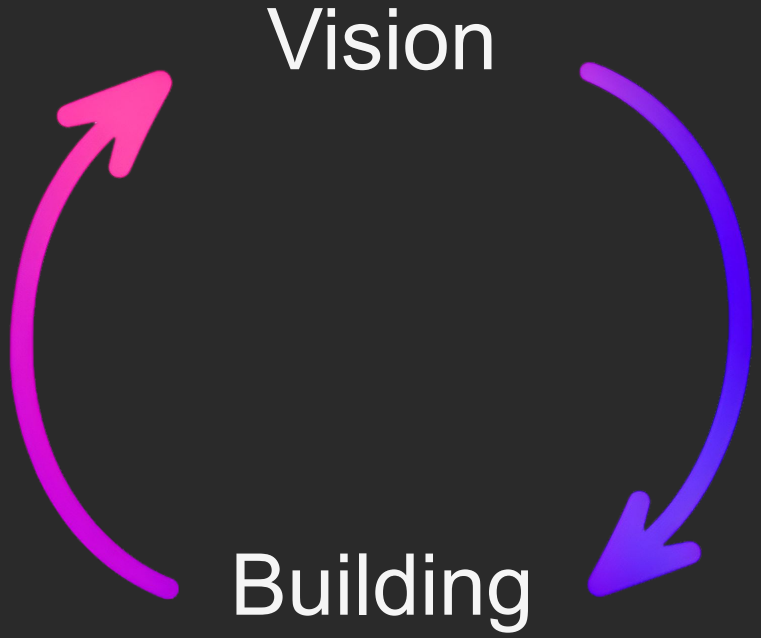
Blueprint Demo

Roadmap Demo

Specification

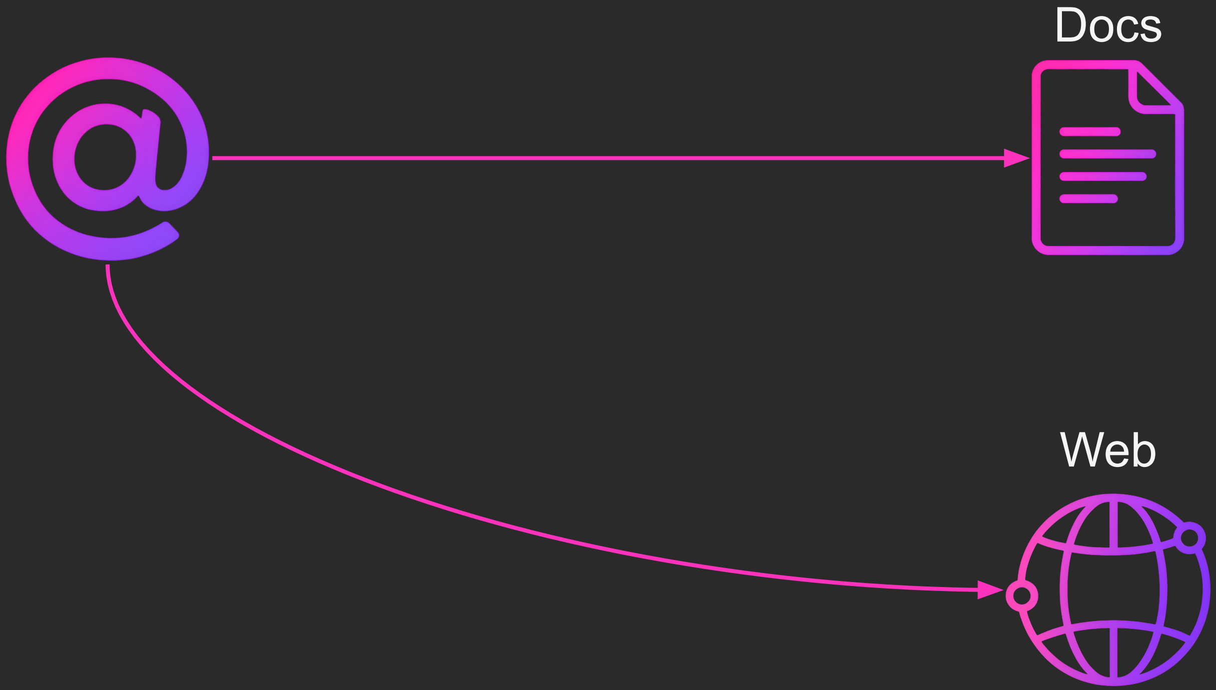
Blueprint

Roadmap



aider

Plan	Code
o3	GPT 4.1
Gemini 2.5 Pro	3.5 Sonnet
3.7 Sonnet (thinking)	3.7 Sonnet


Docs
Web