The October inflection point for vibe game development

I've written before about how November 2025 was an inflection point for coding agents—the point where Opus 4.5 and GPT-5.1 crossed a tipping point that meant their respective coding agents became reliable daily drivers for software engineers. I think we've reached a new inflection point in October 2026, and this time it's for **vibe coding computer games**.

- Tree Appreciator
- Emberwick
- Battle Drones
- Pumpkin Farmer, aka Quarter Sun Cove
- Four times as insulting
- What happens if someone actually puts the work in?
- You should build a game!

Claude Opus 5.5 in particular exhibits meaningful improvements in several areas needed to build a compelling game:

- 3D modeling
- Music generation (see Scrimshaw Jukebox)
- Game design (I)

That last one is particularly important. Models have been able to create something that looks like a game for quite a while now, as seen in my previous experiments. The challenge has been getting to a game that's actually fun to *play*.

Game design is a deep subject area. I don't pretend to be well versed in it—I know enough to have respect for it as a craft. Creating a novel, compelling gameplay loop which challenges players at the right level and provides a satisfying learning curve is a fine art. I didn't think the models could do it.

Then I tried telling Claude Opus 5.5 to "Make it a LOT more fun"—and it did.

In the past few weeks I've vibe coded (in the classic sense of the term—no looking at the code at all) four games that I've genuinely enjoyed playing.

I attended Fun Gamz in Berkeley over the weekend and met several people who were having similar experiences. Building games that play well is shockingly easy now. The new challenges is coming up with concepts for games that are novel and delightful and that wouldn't exist without AI assistance.

I haven't done an exhaustive study, but my current impression is that GPT-6 Astra isn't quite there yet. It can absolutely build a game, but its sense of fun doesn't match that of Opus 5.5. I expect OpenAI will get there by the end of the year.

Here are the four game projects that have convinced me that the models are now capable of producing fun games with very little guidance. All four are built in HTML and JavaScript and run entirely in the browser.

Tree Appreciator

The smallest game I found to be genuinely delightful is Tree Appreciator. I used the regular Claude chat interface for this—it didn't even need Claude Code.

I built this on a whim while we were driving along Skyline Boulevard looking at trees. My prompt was:

> Build a three.js game called "Tree appreciator" - it shows you two algorithmically generated trees and the game mechanics is about appreciating the differences between them. Think hard about how to make sure it is both fun and beautiful.

I've found that telling models to use three.js works really well. Claude doesn't have its own image generation model so it's capable of building full-color 3D models that look and feel excellent.

This game required no iteration or follow-up prompting at all—I got the game in a single shot. It gave me back a charming little mobile-friendly spot-the-difference game. You can play that here. Here's the chat transcript that built it.



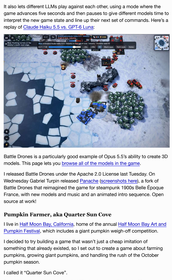
It's pleasingly relaxing. The graphics are fun, and collecting trees that you have completed in a "herbarium" provides a nice, cosy sense of accomplishment.

Emberwick

This was a more ambitious project that I again built using regular Claude chat. I started out by building an experimental prototype of an engine for a top-down fantasy RPG:

> Build a demo game engine three.js which is a top down 1hot art an angle1 RPG - the player gets to move through the world with one or two companions. Create a fantasy village setting that demonstrates walking around outside and then shows how when the player enters a building the wall between the player and the camera goes transparent so the user can see what is happening
>
> Add some NPCs and plenty of furniture and different types of building some with an upstairs - be creative, make it look great

Here's that first version. It worked really well!



I was inspired to increase my ambitions:

> Make it not so flat - the buildings are great but the landscape itself is completely flat at the moment, fix that
>
> Add basements to two of the buildings with a tunnel system joining them below the village
>
> Add the start of a rocky mountain on one side of the village containing a cave with a sleeping troll

Here's that version.



When that produced a good result, I decided to push it all the way:

> Now turn this into an actual game - quadruple the size of the playable area, introduce fun game mechanics, add a plot that isn't too cliche, some twists, a magic system, make it really good

We're back to prompts like "make it really good" here! Asking for a plot that "isn't too cliche" seemed to work too—it's a bit cliché, but still novel enough that I enjoyed following the story.

Unlike Tree Appreciator, this took quite a bit of additional playtesting and prompting to get to a finished version. This was mainly due to gameplay problems—things like a character not being in the right place to complete a quest because a different story element had caused them to move to another location.

Here's that final game. It has a full story that takes about 15 minutes to play through.



There's even an experimental level editor if you want to drop in your own houses, furniture, and landscape features.

Once it was all working (and I'd iterated through some gameplay bugs), I played it through from start to finish and found myself emotionally affected by the ending! I'm somewhat embarrassed by this, because emotionally in this game—the graphics, the plot, the characters, the music, the dialog—came from the model. My creative input was pretty much just "it's a fantasy village and there's a troll asleep in a cave".

Here's the full shared Claude chat in which I built this game.

I also fed that same initial sequence of prompts through ChatGPT GPT-6 Pro. Here's the transcript and the finished game. The models screen, but I don't think it plays nearly as well.

Battle Drones

This is by far my most successful project, which I wrote about last week. It's a Command & Conquer-style Real-Time Strategy game where two robot armies (in four as of Saturday) build units, harvest scrap, and battle over territory.

This is also the project that convinced me that Opus 5.5 knows what "fun" is.

The first versions were built back in July using Claude Fable 5. It got me to a workable 2D Command & Conquer-style game, but it wasn't great to play.

I prompted Opus 5.5 to "Make it a LOT more fun" and then later to "Port this game to three.js" and the game was transformed. There's a blow-by-blow account (prompts included) in my write-up.

Today, Battle Drones has 2–4 player multiplayer (via a vibe coded WebSockets relay server), medals, replays, seven maps, a random map generator, and a set of "mutators" that add variants of the rules.

It also lets different LLMs play against each other, using a mode where the game advances five seconds and then pauses to give different models time to interpret the new game state and line up their next set of commands. Here's a replay of Claude Haiku 5.5 vs. GPT-6 Luna:

Battle Drones is a particularly good example of Opus 5.5's ability to create 3D models. This page lets you browse all of the models in the game.

I released Battle Drones under the Apache 2.0 License last Tuesday. On Wednesday Gabriel Turpin released Panache (screenshots here), a fork of Battle Drones that reimagined the game for steampunk 1890s Belle Époque France, with new models and music and an animated intro sequence. Open source at work!

Pumpkin Farmer, aka Quarter Sun Cove

I live in Half Moon Bay, California, home of the annual Half Moon Bay Art and Pumpkin Festival, which includes a giant pumpkin weigh-off competition.

I decided to try building a game that wasn't just a cheap imitation of something that already existed, so I set out to create a game about farming pumpkins, growing giant pumpkins, and handling the rush of the October pumpkin season.

I called it "Quarter Sun Cove".

I'm still iterating on this project, but it's shaping up quite nicely. It's a cozy farming simulator where you spend the year tending to your giant pumpkin and planting out regular-sized pumpkins for sale in October. Then sell pumpkins during pumpkin season and enter your giant pumpkin in the weigh-off.

Every character in it is a (vegetarian) animal. The mayor of the town is a pelican. You can be a pelican, skunk, raccoon, possum, beaver, coyote, fox, owl, sea otter, or harbor seal. Those species all later show up (accompanied by their children) to buy pumpkins from you.

It's absurdly cute.

You can play the latest version here. I'm not convinced by the game mechanics yet—it's actually quite complicated: you have to decide when to plant pumpkins such that they will become ripe at the weekends in October with the most visitors.

Here's the initial brainstorming session, which resulted in this spec file. Then I had Opus 5.5 Ultracode burn millions of tokens and nearly two days building it.

I told Claude to "surprise and delight the user frequently" and the result is so many little details and easter eggs. There are mischievous raccoons, an elephant seal that runs the weigh-off, harbor seals that roll their pumpkins back into the ocean, a wandering goat, and superstitions about growing giant pumpkins that sometimes come true.

This isn't my best game yet—it's not as fun to play as Battle Drones—but it represents the kind of game I want more of: weird, fun, unique personal—something that wouldn't have been built in the before-times.

Four times as insulting

If you are offended by AI-generated art, AI-generated music, AI-generated writing, and AI-generated code, I have bad news for you: vibe coded games combine **all of those at once.**

I respect those opinions, even though I don't strongly hold them myself.

The wider games community already has an aversion to AI-generated assets in games. This has been a growing issue over the past few years as tools like Midjourney have enabled developers to generate assets for their projects. This 10k story covering complaints about slop in Valve's Steam Next Fest is a good recent example.

We now have games where everything—code, assets, dialog, and even the game design itself—can be outsourced to the models. And the results don't suck!

Building a computer game used to be one of the most challenging projects in all of software development, combining a diverse range of skills and requiring intense focus and effort over many months or even years. And now here I am spitting out a playable multiplayer RTS by prompting on my phone!

I expect that, as we've already seen with programming, this will result in a split. Some players will refuse to have anything to do with games that were built in this way. Others may embrace this as opening up game development to a much wider audience. I lean in that direction!

I certainly won't be entering any of my vibe coded games in traditional game jams or competitions—I don't feel like they've earned that right. We'll likely see new avenues emerge for vibe game developers to compare their projects.

What happens if someone actually puts the work in?

The thing I find both exciting and shameful about this is that I have got interesting results from *so little creative input*. The projects I've described here are loosely ordered by how much effort I've put into them. Not a single project represents more than four hours of my work (plus many more hours of Claude Code churning away by itself).

Now imagine what could be achieved by anyone who takes this seriously, invests significant creative time, applies better taste in game design, and then playtests and iterates on the result until it's as good as they can make it.

I want to see games that wouldn't have existed without these new tools.

You should build a game!

Building games in this way is so much fun. It's the perfect application for vibe coding because it's so low stakes: how much damage can you cause with a bug in a personal vibe coded game project?

IDK, you can do some damage... my attempt at audio-based multiplayer gave my wife Natalie a nasty headache!)

All of my games so far have been HTML and JavaScript using static hosting (generally GitHub Pages or Cloudflare Pages). Modern browsers—including mobile browsers—are incredibly capable. Frontier models already know libraries like three.js and Phaser. I'm finding these mobile web games to be indistinguishable from the native experience, at least for the kinds of games I'm trying to build.

I suggest starting with a prompt like this:

> Build a mobile-friendly three.js game where...

This works on both chatgpt.com and claude.ai—you don't even need to fire up a full coding agent. As one last example, here's "Build a mobile-friendly three.js game where a mouse has to hide from a cat" for Claude Opus 5.5 Extra and for GPT-6 Extra High.

Switch to Codex or Claude Code once your project passes the proof-of-concept phase.

And if the game isn't fun enough, add this:

> Make it a LOT more fun

Posted 28th October 2026 at 1:55 pm · Follow me on Mastodon, Bluesky, Twitter or subscribe to my newsletter

More recent articles

- Ladybird's first alpha release - 23rd November 2026
- Running the full server-side Datasette Python app in the browser with service workers and Pyodide - 17th November 2026
- Locating the source of a mysterious beep with Claude - 9th November 2026
- Hide your slop PR descriptions behind <details><summary> - 9th November 2026

This is **The October inflection point for vibe game development** by Simon Willson, posted on 28th October 2026.

game-design 12 · games 12 · ai 2241 · generative-ai 1811 · chatgpt 344 · llms 1169 · claude 320 · vibe-coding 66 · claude-code 56

Next: The U.S. Census Bureau and "Super Intelligence"

Previous: Battle Drones - a vibe coded RTS with audio-based multiplayer

Monthly briefing

Sponsor me for $10/month and get a curated email digest of the month's most important LLM developments.

Pay me to send you less!

Sponsor & subscribe

More recent articles

- Ladybird's first alpha release - 23rd November 2026
- Running the full server-side Datasette Python app in the browser with service workers and Pyodide - 17th November 2026
- Locating the source of a mysterious beep with Claude - 9th November 2026
- Hide your slop PR descriptions behind <details><summary> - 9th November 2026