ARTICLE | BY FOX

How We Built Super Chicken With AI in About a Month

By Fox, founder of Shrunken Ship

A man in a chicken suit. A big black mustache. A pig on a motorcycle.

Somewhere in there, we decided we were making a video game.

About a month later, we had our first full playable game. We brought it to our own website, Shrunken Ship, and kept improving it as people played.

That game is **Super Chicken**, a comedy browser adventure that grew into a campaign of 20 playable levels. I made it with help from ChatGPT, combining my ideas, direction and playtesting with AI-assisted coding and iteration.

The month was the first playable milestone. The lessons below also include the polish that followed - because releasing a game is an excellent way to discover all the things your chicken can still break.

The chicken existed before the code

Super Chicken started with my little brother.

In our family comedy and music show, he performed in a chicken costume with a black mustache. It was ridiculous. It made people laugh. Years later, that character still felt like someone who deserved his own adventure.

Our family also made movies. One was called The Mystery of the Shrunken Ship, which eventually inspired the studio name.

So the game began with something personal: a character, a sense of humor, and the desire to turn a family memory into something other people could play.

That turned out to be useful direction. Super Chicken had to remain recognizably a man in a chicken suit. The mustache mattered. The giant chicken shoes mattered. Being slightly absurd mattered.

The first lesson: give your game something specific to be. A clear character helps you make decisions about art, movement, jokes and even combat. “Make it awesome” gives you considerably less to work with.

What we actually built

Super Chicken runs in a web browser. Its game code uses JavaScript, with Canvas drawing the action on screen. In plain English, the browser runs the rules, draws the characters and scenery, and responds to your controls.

The campaign grew into a collection of ridiculous adventures, including platforming, boss fights, chute sequences and a ski-village brawl. It supports keyboard and mobile touch controls.

That variety was fun. It also created work. A control that feels right in a platforming level can behave differently when the next level is a brawler.

There was also a whole experience around the action: starting the game, choosing levels, pausing, retrying, controlling sound and continuing through the campaign. The website added its own presentation and optional account features.

A full game is a surprisingly large collection of little things that all need to agree about what happens when someone presses a button.

How I worked with AI

I brought the character, the jokes, the level ideas and the decisions about what felt right. ChatGPT helped turn directions into code, investigate problems and make revisions.

The important part was the loop:

1. Describe a specific experience.

2. Build a playable version.

3. Play it and observe what actually happens.

4. Explain the difference between the intention and the result.

5. Make a focused change.

6. Check the fix and the things it might have broken.

7. Save a working version.

My most useful feedback became increasingly concrete.

A request to improve mobile controls became a request to catch quick taps, support movement and jumping together, and stop shallow sideways swipes from triggering the wrong move.

A request to make a power-up better became a request for bigger attacks and enemies flying farther when hit.

That is creative direction you can implement and test.

AI helped us move quickly, but I still had to play the game, choose what to keep and say when something felt wrong. You cannot outsource the question “Is this actually fun?” to a successful build message.

When a cool effect made combat feel broken

One of the useful lessons came from the ski-village combat.

Hit pauses can make attacks feel heavy: the action freezes briefly when a blow lands. But our earlier version stopped the simulation during impacts, and fresh inputs could get lost before they were processed.

In one reproduced test route, those pauses added up to almost nine seconds of stopped action.

That is a lot of time to spend wondering whether your chicken has suffered an administrative delay.

The fix kept movement, jump inputs, enemy behavior and attack recovery updating through impacts. We also smoothed the camera across district boundaries and reduced overlapping impact-sound work.

The lesson was specific: **an effect meant to make combat feel better can interfere with the controls that make combat work.** When something feels glitchy, investigate the timing and input handling underneath the animation.

The improvements that came from people playing

Some of our best lessons arrived during later polish. These were improvements to a playable game, rather than proof that every launch detail had been perfect.

That distinction matters if you are starting your own project. Finishing a first version gives you something real to learn from. It does not grant your game immunity from thumbs.

Mobile controls had to forgive real thumbs

Players reported that early mobile controls needed to feel more responsive and accurate.

A thumb drifts. It slides between buttons. It presses jump while the other thumb is still moving the character. It sometimes taps a move a fraction of a second before that move becomes available.

We adjusted the directional stick so small sideways variations were less likely to become accidental diagonal actions. Buttons tolerated a little thumb drift. Deliberate slides between buttons transferred the input.

For Chute There It Is, we added a short roll-input buffer: a tap just before the move was ready could wait briefly and trigger once when it became available.

Think of it as the game saying, “Yes, I heard you.”

It still needed limits. Holding the button should not generate endless unintended rolls, and a queued move should not survive a restart or pause and surprise the player later.

The lesson: responsiveness includes understanding what the player meant, while keeping the rules predictable. Your thumb did not read the test plan.

Porky needed to communicate his attack pattern

A friend struggled with Porky, our motorcycle-riding pig boss.

The fix focused on making the fight easier to read. His tired pose became clearer: hunched forward, hooves braced, visibly panting. His stomp gained a stronger warning pose and a clearer landing effect. His vulnerable recovery window became easier to recognize and use.

We also added a safe introduction that explained the sequence before the fight started: jump the bike, avoid the ground waves, attack while Porky is winded.

The guide waited for the player to begin. It did not explain the rules while the boss was already attempting vehicular poultry slaughter.

We preserved his health, damage, attack order and wave speed in that readability pass.

The lesson: difficulty depends on what the player can understand. A boss can be challenging while clearly showing when to dodge and when to attack.

Super mode needed to feel super

In Cluck Knuckle Village, a friend reported that turning golden did not feel like much of a transformation.

The power-up already added damage. The problem was that the experience did not communicate enough power.

If your superhero transformation feels like selecting a yellow shirt, you have some work to do.

We widened powered attack reach, increased launch force and lift, strengthened impact feedback, and let launched skiers cause more satisfying collision chains.

The launch-force increase was 65 percent. That is a game-development measurement I am delighted exists.

The bug pass also caught actual logic problems: powered attacks could help refill their own power meter, and repeated activation could interfere with the transformation's timing. Those needed fixes too.

The lesson: check both the rules and the sensation. A number can be correct while the attack still feels disappointing.

One stretch needed fewer obstacles

The final fifth of Chute There It Is had too much debris.

We reduced debris in that final section by 20 percent. We preserved the earlier stretch.

That made the change easy to describe, compare and playtest. We were addressing a particular section where the pressure had become excessive.

The lesson: tune the part that is causing the problem. Specific feedback lets you make a specific improvement without rebuilding a level that people already enjoy.

Fix bugs without deleting the personality

In Death Fryer, pressing down makes Super Chicken crouch into a little ball.

I love that stupid little ball.

During the bug pass, I specifically wanted that movement animation preserved. The controls and logic could improve while the character kept doing the thing that made me laugh.

This is a useful distinction when giving AI feedback. Identify the approved parts as clearly as the broken parts:

- Keep the crouching pose.

- Keep the character design.

- Fix missed inputs and unintended behavior.

- Check that the fix still leaves the funny moment intact.

Your game’s personality often lives in details that would look strange on a serious design document.

Sometimes the correct creative decision is to protect the stupid little ball.

A screenshot cannot prove that a game works

A screenshot tells you what one moment looks like.

A playable game must survive a person holding the wrong button, pressing two things together, pausing halfway through an attack, losing, retrying and changing their mind.

Our checks grew to include complete simulated campaigns, repeated inputs, keyboard and simulated touch routes, and pause-and-restart behavior. Later, we also checked basic flows in the hosted browser game.

We learned to distinguish these forms of evidence. A simulated touch test can catch input-logic problems. A person holding a real phone can tell you whether the controls are comfortable. Both are useful; they answer different questions.

The same applies to performance. We reduced repeated drawing and interface-update work in some levels, but a lower workload measurement does not prove smooth performance on every device.

The lesson: name what you tested accurately, and keep playing. “The automated test passed” is a useful sentence. It still leaves room for a friend to say, “Why does this pig keep killing me?”

Save the game while you are making the game

We kept source history, working checkpoints, playable copies, assets and development notes.

That sounds less exciting than launching skiers across a village. It is also what makes ambitious revisions manageable.

A useful checkpoint tells you what worked, what changed, what remains uncertain and how to recover the previous playable version.

It also gives a future AI session enough context to continue. Otherwise, you can spend the next session explaining your entire chicken universe again.

The lesson: preserve decisions as well as files. “This animation is approved” and “this level still needs a real-phone playtest” are valuable pieces of project information.

Getting it onto our own website

The destination was a game people could open through Shrunken Ship:

Play Super Chicken (https://shrunkenship.com/games/super-chicken).

That introduced another layer of work: hosting the game, connecting the custom domain, checking that assets loaded, and verifying the player's route from the website into the action.

Guest play remained available. Optional accounts and saved progress were additional website features.

We also needed to distinguish the hosted experience from a standalone playable backup. A saved copy of the game could preserve the campaign and assets while leaving out website accounts and cloud saves.

The lesson: test the link you actually share. A working development copy and a working public launch are separate milestones.

What I would tell someone making their first game

Our campaign grew to 20 levels. Your first milestone can be much smaller.

Start with one character, one enjoyable action, one obstacle, a win condition, a lose condition and a retry button. Put that complete loop in someone else's hands.

Watch where they hesitate. Ask what they expected to happen. Notice whether they want another attempt.

Then make the feedback specific enough to act on. "The roll ignores my quick tap near the end of its cooldown" gives you a better starting point than "the controls suck," even if both were said with equal passion.

Build outward from something that already works. Keep a recoverable version before substantial changes. Add complexity when the existing game earns it.

And give yourself permission to make something personal and ridiculous. Our hero began as my brother in a chicken costume. That turned out to be a better starting point for us than trying to sound like an enormous game studio.

The first month got us to a playable game. The revisions taught us how to become better at making one.

Try Super Chicken on Shrunken Ship (https://shrunkenship.com/games/super-chicken), and tell me which moment made you laugh - or which button made you swear. Both are useful feedback.

If you are making your first game too, share what you are building. I want to see more people turn the weird idea in their head into something the rest of us can play.

Questions about making a first game with AI

Can you make a playable game with ChatGPT

Super Chicken is one example of using ChatGPT to help implement and improve a playable game. Human direction, testing, debugging, revision and publishing were all part of the process.

How long did Super Chicken take to make

The first full playable build took about a month. Improvements continued afterward. This article includes both the build story and lessons from later polish.

What engine does Super Chicken use

Super Chicken uses custom JavaScript game code and Canvas rendering in a web browser. Its website provides the surrounding experience, including optional accounts and progress features.

What was the most useful development habit

Turning playtest feedback into a specific, reproducible problem, making a focused fix, and checking both the changed behavior and the parts we wanted to preserve.

Where can I play Super Chicken

Super Chicken is available through Shrunken Ship (https://shrunkenship.com/games/super-chicken). It has keyboard and mobile touch controls.