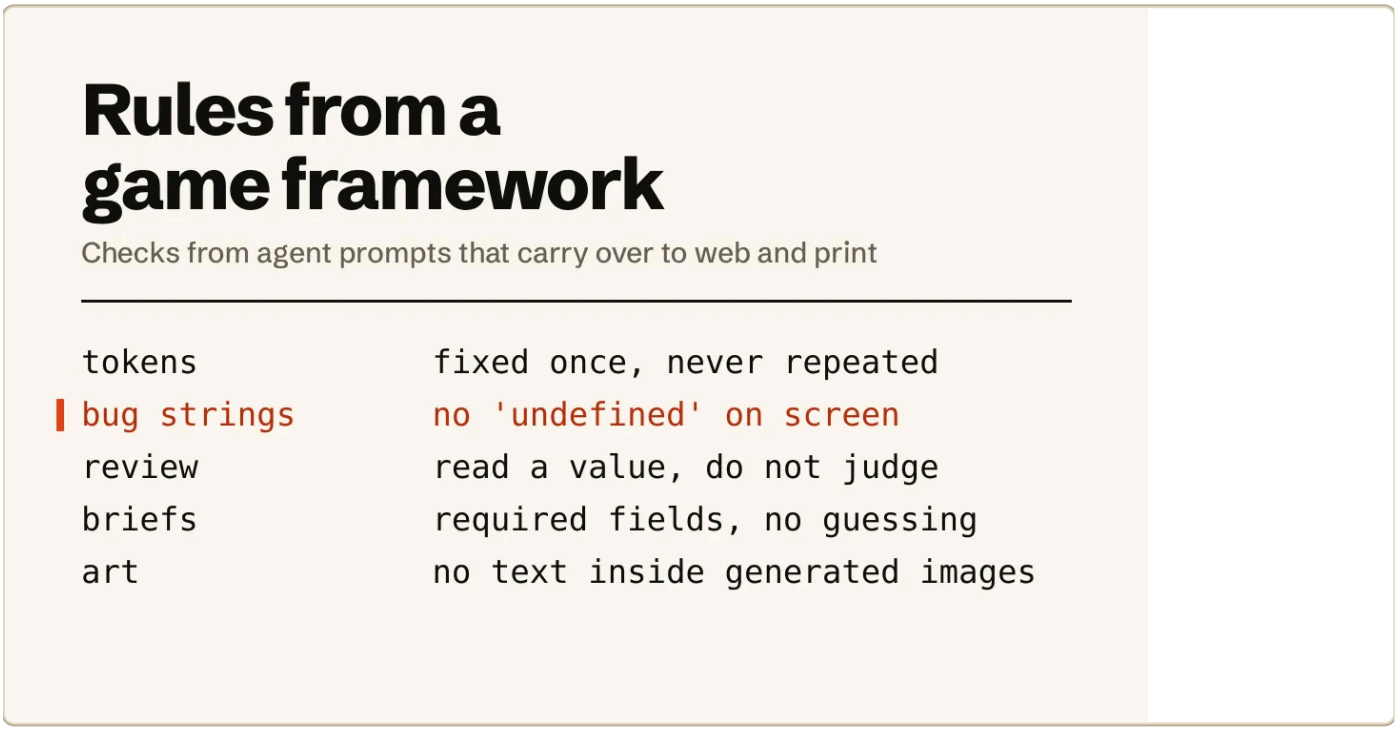
Eight rules from a game framework's agent prompts, for web and print design

On 1 November I read the agent prompts of VibeGame, a framework that builds 2D web games, and kept the eight techniques that carry over to web and print design work with AI agents.



Published
1 November 2026

Reading time
4 minutes

About
Mullet

Tags
AI, Testing

More on: Mullet

On 1 November I read the agent prompts in VibeGame (https://github.com/tettethu/VibeGame), an Apache-2.0 framework that turns prompts into 2D web games. It uses several agent roles: architect, artist, designer, player, programmer, reviewer and auditor. The prompts are about games, but eight of the techniques in them carry over to web and print design.

This is a checklist exercise. I read the prompts as text. I did not run the framework, so I can't tell you how well it works. I am only saying the techniques look worth borrowing.

What does not carry over

Sprite sheets, tilesets and colliders are game assets. Bot-driven runtime checks, where a program plays the game to see if it works, have no equivalent in a brochure. Skip those parts.

The eight rules

1. Fix design tokens once, in one file. The auditor prompt wants each tunable value kept in the config block of the one node that owns it, flags values duplicated between configs, and says: "Code references config keys, never embeds the literal." For design, that is one place for brand colours, the type scale and spacing. No hex codes or pixel sizes typed into components or layouts.

2. Reject visible bugs on sight. The reviewer prompt rejects on a visible bug string such as "undefined", "null" or "[object Object]" (it also lists unfilled templates like "{name}"), and on text that is clipped or overflows its container. Print has its own versions, which are my addition rather than theirs: a leftover placeholder, a missing merge field, a headline that runs off the trim.

3. Make an independent model read the render. The reviewer prompt has an independent vision model judge the screenshot, because the reviewing agent may be contaminated by what it knows of the implementation. It also says: "Ask the VLM to read the value, not to judge whether it is readable." For design, render the work and ask a vision model to read a value off it. Here is a made-up example, mine and not theirs: ask it what a price badge says. If it reads "£49" where the brief says "£39", you have found something. If you ask "is this legible?", you get a polite yes.

4. Never accept "looks fine". The reviewer prompt says: "Do not get talked out of it by an "overall looks ok" impression." A general impression is how defects get through. A check either has evidence or it has not been done.

5. Brief with required fields, and report when one is missing. The artist prompt lists what the agent must have from the request before it starts, among them the subject, the asset category, the global art style and the view. For a design brief you would pick your own fields, such as purpose, size and margin; that list is my suggestion, not theirs. If a field is missing, the agent does not fill the gap with a guess. The artist prompt says: "If any required field is missing, report and ask; you do not own these decisions." A designer would call this a proper brief, and it is the part people skip.

6. Make a style anchor first, and keep the raw outputs. The artist prompt says to create the style anchor first for a reference image, and later assets use it as their reference. It also keeps even rejected raw outputs, so the generation prompt stays traceable. For design, I would keep the prompt behind every generated image too, so you can trace why a set drifted.

7. Keep text out of generated art. The artist prompt lists no text, no labels, no UI and no speech bubbles for sprite and raster UI assets. For design I would set type afterwards in HTML or PDF with the brand fonts, which is my addition. Text baked into an image can't be edited, searched or checked as text.

8. If you disagree with a value, stop and report. The player prompt says: "if you are changing a value because you think the right answer is different, stop and report." Take a made-up example of mine: the brief says a 16px margin and the agent thinks 24px looks better. The agent should say so and wait. The brief may be wrong, but that is a call for whoever wrote the brief, and a silent change is how a layout stops matching its spec.

Where Mullet comes in

Mullet already covers the review side for code. A step passes only when its gates pass: lint, tests, scope and a secrets scan, as described in Gates, not a model's word, and the whole design is in Mullet 2, coding jobs you can leave running. A model saying "done" is not evidence.

Design work has no such gates yet. These eight rules are what I would add for it. Rules 1, 2 and 5 are the easiest to turn into automatic checks, because they are about files and text. Rules 3 and 4 need a rendered image and a second model. I have not built any of this into Mullet. It is a list.

The prompts are in the VibeGame repo, and Mullet's code is at https://github.com/tomgould/mullet-public (https://github.com/tomgould/mullet-public).