Usuals: Scaling AI Video Production Beyond One-Off Prompting

Problem

Usuals' AI video production ran at 150–200+ generations per project, across stills, animation, and live-action composites, using multiple models. At that volume, writing a fresh one-off prompt for every shot didn't scale: output drifted between shots, the same prompt-writing effort got repeated hundreds of times, and knowledge of what worked stayed undocumented instead of carrying over between projects.

Research and systems built

Rather than solving each shot individually, the research direction was to build a small set of standing systems that could be reused across every project:

System docs per category	Separate references for cinematic, animation, and IRL + animation composite work, so continuity rules and prompting logic are documented once per category instead of reinvented per project
A prompt-optimization step	A method for separating what a shot needs to say from how it's phrased for a given model, so creative intent survives the translation into a technical prompt
A cinematography parameter library	Mappings between creative intent (mood, grade, lighting) and its known visual and emotional result, built from testing rather than guesswork
Character consistency guidelines	A defined minimum set of reference inputs required to keep a character's identity stable across many separate generations
A personal reference bank	A tool for collecting and annotating visual references on a canvas, so a project's inspiration stays organized and explainable rather than scattered across bookmarks

Each system came out of running real production and tracking where shots broke or drifted, not from a single pass of upfront planning.

How each problem was solved

Consistency drift

Failure mode: The same style description, reused across a project, still produced visibly different results by shot 30–40, especially in animation, where the model has no memory of earlier shots in the sequence.

Investigation: Tested whether this was model randomness or a prompt problem by regenerating the exact same shot twice with an unchanged prompt. Results were close enough to rule out pure randomness. The actual cause was that the style description itself got reworded slightly between shots, and even small wording changes shifted the model's interpretation.

Fix: Stopped re-describing style per shot. Locked the exact wording once per project and reused it verbatim. When a shot drifted anyway, the fix was to regenerate from the same locked reference rather than write a new description to "correct" it.

Why it holds: Removing the model's room to reinterpret removes the main source of drift.

Duplicated effort

Failure mode: Writing a careful prompt per shot took several minutes each. At 150–200+ shots a project, that was multiple full days spent on prompt-writing before a single generation ran.

Investigation: Broke down what actually changes shot to shot versus what stays identical within a category. Most of a prompt's structure repeated nearly word for word; only a handful of details actually varied per shot.

Fix: Built a fixed skeleton per category with defined variable slots, so writing a new shot's prompt became filling in what actually changes, not composing from scratch.

Why it holds: Time per shot dropped because the parts that don't change stopped being rewritten.

Model fragmentation

Failure mode: A lesson learned prompting one model didn't transfer to a video model or a compositing step, and the same category of mistake got repeated across tools because each was treated as a separate problem.

Investigation: Mapped what each model actually responded to across categories, rather than assuming prompting logic was universal.

Fix: Built structurally parallel system docs per category: different exact content, same five-part shape (lock, prompt skeleton, continuity rules, insights log, resource storage).

Why it holds: A consistent shape across categories means picking up a new model is adapting a known structure, not starting from zero.

How the fixes were validated

Each fix went through the same loop: form a specific hypothesis about the cause, isolate it by changing one variable at a time across a small batch of test shots, review the result against the previous approach, and only keep the change if it held up across more than one project.

Outcome

Generations stopped depending on a single person's recall of what phrasing worked that day. New projects could start from documented references instead of a blank page, and output held up consistently shot to shot rather than needing constant manual correction. These systems remain living references at Usuals, updated whenever a new failure mode surfaces. The team keeps building on them from project to project, rather than treating them as a finished deliverable.

Utkarsh Bramhankar · Usuals AI