PlotPunk

An easy but controllable tool for video generation

Runway API Hackathon — Calluna Labs · May 2026

The Problem We Started With

Generative video has reached the point where a single prompt can produce something genuinely impressive. The catch is what happens after that first clip. The moment a creator wants a second shot of the same character, a third with a different camera angle, a fourth in a new location at the same time of day — the tools fall apart. Identity drifts. Wardrobe changes between cuts. Lighting resets. The entire creative act collapses into "regenerate and pray."

The workaround today is to stitch together a dozen separate services — a generator for stills, another for video, a TTS service, a lipsync tool, a non-linear editor, a prompt-engineering scratchpad, a colour grading step — and to spend most of your time managing handoffs between them rather than directing your story. That isn't filmmaking. It's plumbing.

We wanted to know if a single environment could absorb all of that plumbing without absorbing the creative authorship along with it.

The Idea

PlotPunk is built on a deliberate distinction: the **technical** complexity belongs to the system, the **creative** decisions belong to the user. Modern one-shot video tools have a tendency to collapse both into the same prompt box — and in doing so they take the most interesting part of filmmaking, the directing, away from the human and hand it to the model.

We took the opposite stance. The user brings the characters, the story, the visual language, the pacing. The system handles prompt construction, asset linking, model selection, retry logic, and cross-shot context propagation. Bulk operations — multi-aspect exports, social-media one-liner dailies, alternate cuts — are supported as expansions of a user-authored idea, never as substitutes for one. A creator who has never written a prompt in their life should be able to realise their vision; a creator who knows exactly what they want should not be forced to surrender control to get it produced.

The mental model we ship is **the user is the Studio Boss**. AI agents play production-crew roles — a Showrunner, a Casting Director, a Production Designer, a Drehbuchautor (Screenwriter), a DOP, a Sound Designer, an Editor — each with their own persistent chat history and domain-expert system prompt. They *propose* work. The user *approves* it, edits it, or rejects it with a reason. The reason flows back into the next iteration so the persona learns what the boss does not want, not just what they do.

Three approval gates bound cost and protect the creative process from runaway inference spend: **G1** after the script, **G2** after the keyframes, **G3** after final assembly. Nothing expensive runs until the previous step is signed off.

How We Got Here

We did not begin with this architecture. The journey to a working pipeline shaped both what we built and what we learned was not yet possible.

The concept came first. We began with GPT-4.1, asking it to take the user's raw idea and produce a *Production Concept*: a cast with characters, a visual style described in text, the pacing, the target audience, the background information that would later anchor every prompt downstream. The Production Concept became the persistent context that every later persona could read from. We later split the pre-production phase into discrete Showrunner, Casting Director, and Production Designer steps so that each creative decision had its own conversation and its own approval point.

Keyframes followed. With the Production Concept locked, we asked another GPT-4.1 instance to write image prompts for the *start frame of each shot* — what we call a keyframe. This is where consistency would either succeed or collapse. We tested several Runway image backends and settled on `gemini_2.5_flash` (colloquially "Nano Banana Pro") as our primary, with `gen4_image` as a safety fallback. Fed with the previous shot's keyframe plus a camera-only textual instruction, this chain reliably retained character identity, attire, and environment from one shot to the next. The user reviews every keyframe at G2 — an entire episode's frames laid out in a grid — and approves, rejects, or requests specific re-renders before any expensive video generation runs.

Then came the hardest problem: dialogue. This is where the journey was the most instructive, because we tried three approaches before finding one that worked.

The first was straight text-to-speech via Runway-proxied ElevenLabs (`eleven_multilingual_v2`, 49 voice presets). It produces natural, intelligible spoken audio from user-written lines, and it integrated cleanly. But TTS alone gives you sound — not a speaking character. We needed both.

The second was Runway's `gwm1_avatars` model. The promise was native lipsync: feed a still image and an audio track, get back video where the character's mouth and head animate to match the speech. We built the full integration end-to-end. The result disappointed us on two counts. The visual quality was limited — minimal head motion, flat expressiveness, a rendering style that fell below the cinematic bar a storyboard-driven product needs to clear. But the deal-breaker was framing: the model imposes a fixed virtual camera at head-and-shoulders distance and that framing cannot be overridden. A wide shot of a speaking character, an over-the-shoulder, a tracking close-up — none of these are expressible. For a tool whose entire point is shot-by-shot camera direction, this was incompatible at the foundation. We kept the integration in the codebase behind a feature flag (`ENABLE_AVATAR_INTEGRATION`, currently off) so it can be revived the day a more flexible avatar model ships, but it is not on the path to the demo.

The third approach, which became our actual answer, was Runway's `seedance2`. Given a visual prompt — character description, scene, camera intent — it synthesises both motion and speech in a single inference call. The result is what cinematic dialogue ought to look like: an actor in a real shot, mouthing intelligible words, with full per-shot framing freedom. The trade is real: the specific spoken words are determined by the model and cannot be prescribed by the user. For shots that need scripted dialogue verbatim, we render `seedance2` silent and layer ElevenLabs audio underneath via a deterministic Hetzner FFmpeg post-production service. The composite path preserves both authored dialogue and full camera control, at the cost of native lipsync.

What the System Looks Like Today

The user moves through eight steps in a single web application: **Concept → Casting → Style → Script → Shots → Keyframes → Audio → Videos → Final Assembly**, with the three approval gates dropped in at the right cost-control moments.

Under the hood, **Inngest** orchestrates every long-running inference call as a durable, retry-aware step. Each Runway call uses a *kickoff + poll* pattern so no single network round-trip exceeds a few seconds — long generations run reliably without hitting edge-function timeouts. **Coolify** deploys the whole stack to a Hetzner VPS. A small dedicated FFmpeg service handles post-production: per-shot concat, dialogue and ambience mixing, libass subtitle burn-in, and parallel multi-aspect export (16:9 plus 9:16 plus 1:1 from a single generation pass). The generative layer is consolidated behind a single Runway API key for image, video, avatar, TTS, and ambience — the only second provider in the system is OpenAI for the LLM layer.

The asset library — characters, voices, styles, reference images — is account-scoped, so a creator who has built a recurring cast in one project can pull them straight into the next. Bindings into projects are time-scoped and snapshotted, so a generated asset records the exact inputs it was made from and is never retroactively invalidated by an upstream edit. This is what lets a creator iterate confidently over weeks without their earlier work silently shifting under them.

What We Learned, and What's Missing

The work surfaced a concrete and tangible limit in today's API-accessible generative video. No model we tested simultaneously supports all three of:

1. **Arbitrary user-provided speech audio** — the exact words and performance the writer authored
2. **High visual fidelity** — cinematic enough to live next to a storyboard
3. **Unconstrained camera framing** — wide, OTS, tracking, insert, the full DOP vocabulary

TTS gives you (1) but no moving image. Avatar models give you (1) and a degraded (2), but lock framing — you lose (3). `seedance2` gives you a strong (2) and (3) but takes (1) away from the user. The composite path of `seedance2` plus FFmpeg audio mixing recovers (1) at the cost of native lipsync. This is the gap. It is, in our view, the most useful single feature Runway could add to the API in the next year: a video model that accepts an audio waveform as input, a reference image, and a free-form camera instruction. Such a model would close the last hole in a fully directable narrative-video pipeline.

PlotPunk is what a generative video tool looks like when you take seriously the idea that filmmaking is *directing*, not prompt-writing. The technical complexity is gone. The creative work is exactly where it belongs — with the human in the chair. The Studio Boss approves the work. The AI crew renders it. That distinction is the whole point.