Case Study: Reeldemo

Agentic music production from reference to editable Ableton sessions

Julian M. Kleber | Founder | reeldemo.github.io

julian.kleber@sail.black | Last updated: July 15, 2026

Outcome snapshot

- Built a **three-repo music stack**: agentic Studio (commercial), MIT ReelSynth wavetable engine, and Kaleidoscope ASCII promo engine.
- Shipped **ReelSynth v0.1.0** with standalone app and export CLI for macOS, Linux, and Windows.
- Designed an inspectable agent loop — compose, grade, handoff — with JSON Schema contracts and eval gates before anything reaches Ableton Live.
- Launched org landing with **Studio early-access waitlist**, live Kaleidoscope hero, and integrated Supabase signup pipeline.

CONTEXT

Indie producers want AI-assisted speed without surrendering authorship. Most tools either output frozen bounces or hide decisions in a black box. Ableton Live remains the edit surface, but moving from reference audio to structured, revisable session material is slow, error-prone, and hard to audit.

Reeldemo was created for producers who ship under pressure: keep creative control, inspect agent decisions, and hand off *editable* tracks, MIDI, and devices — not a finished master file.

PROBLEM DEFINITION

In typical AI-music workflows, teams and solo producers face:

- loss of edit authority when tools export only bounced audio,
- opaque generation with no contract or trace for what the agent decided,
- brittle reference-to-MIDI pipelines that break arrangement intent,
- separate tooling for sound design, session building, and promo visuals.

APPROACH

Reeldemo combines three standalone tools under one production mindset — **reference in, editable session out**:

- **Reeldemo Studio** — agentic compose, remix, and grade with Oracle workflows, Demucs stem separation, and Ableton handoff (OSC, Max for Live, Live 12 Extension inbox).
- **ReelSynth** — MIT open-source wavetable synthesizer; AI never renders audio directly — stems come from PyO3 offline rendering through a readable DSP engine.
- **Kaleidoscope ASCII** — Rust-fast mirror engine for landing heroes, social ads, and CLI/npm promo pipelines.

Oracle tasks (`copy`, `remix`, `create`, `grade`) run against JSON Schema contracts with automated judges before handoff.

The agent loop is explicit: compose → compose_view → evolve → layer_grades → handover_plan → handover.

SYSTEM OUTLINE

- **Studio Oracle layer**: copy/remix/create orchestration, instrument plans per layer (drums, bass, chords, melody, fx), rubric grading, trace logs.
- **Stem pipeline**: Demucs separation into remixable layers with arrangement steering.
- **Render engine**: ReelSynth Rust core — 256-frame wavetable banks, multi-osc voice, filter, mod matrix — exposed via standalone UI, export CLI, and PyO3 for headless Studio renders.
- **Ableton handoff**: OSC scene/clip automation; M4L clip writers and drum-rack loaders; Extension import of `session_handover.json` + WAV bundles; modes for audio, raw MIDI, MIDI+device JSON, and drum racks.
- **Licensing**: Supabase-backed 14-day trial and paid keys for commercial Studio features.
- **Promo engine**: Kaleidoscope Rust core + `@reeldemo/kaleidoscope-ascii` (napi-rs) powering the org landing hero and exportable HTML/ANSI assets.

IMPLEMENTATION STACK

- **Agent orchestration**: Python services, JSON Schema contracts, MCP/API export endpoints (commercial `reeldemo-ableton` repo).
- **DSP / instrument**: Rust ReelSynth engine; PyO3 bindings; cross-platform standalone and CLI (`reelsynth-app`, `reelsynth-export`).
- **DAW integration**: AbletonOSC, Max for Live handover tools, Live 12 Extension inbox import.
- **Visuals**: Rust Kaleidoscope core; npm CLI and browser demo on reeldemo.github.io.
- **Web / GTM**: Static org landing (Studio, ReelSynth, Kaleidoscope product pages); Supabase waitlist at `supabase.reeldemo.io`.
- **Licensing & auth**: Supabase Postgres for Studio keys and waitlist capture.
- **Brand**: Majico design tokens applied to Reeldemo org palette and landing surfaces.

OSS VS COMMERCIAL BOUNDARY

A deliberate license split keeps the sound engine reusable while protecting the agent layer:

- **MIT (`reelsynth`)**: DSP engine, `.reelwt`/`.reelpreset` formats, importers (Vital, Serum, WAV), export CLI, Python render API.
- **Commercial (`reeldemo-ableton`)**: agent compose, text-to-wavetable, session export, Ableton handoff orchestration, license-gated MCP/API.

RESULTS (JUL 2026)

- **ReelSynth v0.1.0 shipped** — standalone synthesizer app and export CLI for macOS (ARM + Intel), Linux, and Windows; plugin waitlist open.
- **Three-product org landing live** — bento stack (Studio / ReelSynth / Kaleidoscope), production-loop narrative, integrated waitlist forms.

- **Inspectable handoff design validated** — explicit compose contracts, per-layer synthesis roles, and grading gates documented in integration specs; AI does not render audio directly.
- **Promo pipeline unified** — Kaleidoscope powers homepage hero and CLI/npm export paths with shared Rust semantics.

Treat Studio conversion and user-count metrics as **waitlist-stage** until refreshed from production analytics.

WHAT THIS CASE STUDY DEMONSTRATES

- Agentic systems with **inspectable contracts** — not one-shot black-box generation.
- Clean OSS/commercial architecture for creative tooling (MIT engine + licensed orchestration).
- Full-stack indie delivery: Rust DSP, Python agents, DAW extensions, static marketing site, and npm CLI in one org.
- Domain authenticity — built by a producer who uses Ableton Live daily.

WHAT I WOULD IMPROVE NEXT

- Ship ReelSynth VST3/AU plugin (S7) to complete the DAW-native instrument story.
- Expand Studio early-access cohort with production telemetry on handoff success rates and edit retention in Live.
- Harden eval rubrics with more reference-track regression fixtures per genre.

LINKS

- Org landing: reeldemo.github.io
- Studio: reeldemo.github.io/studio
- ReelSynth (MIT): github.com/reeldemo/reelsynth
- Kaleidoscope: reeldemo.github.io/kaleidoscope
- GitHub org: github.com/reeldemo