Bounded Goal Loops, Better Review Bots, and New Computer-Use Primitives

Coding Agents Alpha Tracker

2026-05-10

Bounded Goal Loops, Better Review Bots, and New Computer-Use Primitives

By Coding Agents Alpha Tracker • May 10, 2026

Today's practical edge is operational discipline: Codex and Hermes goal loops only become useful when you pin them to explicit validation and stop rules. Also worth your attention: Crabbox for disposable debug loops, Peekaboo 3.0 for macOS computer use, Copilot review's jump in usefulness, and strong Codex iOS build feedback.

TOP SIGNAL

- **Bounded goal loops are the real unlock.** After three days with Codex/Hermes goal-based agents, Jason Zhou says most people use them wrong: the loop only works when you define the objective, constraints, validation method, and explicit stop conditions up front — not when you say "keep fixing stuff." His deeper Codex walkthrough shows why: `go` replaces dumb programmatic looping with an LLM judge, which works well for hours-long migrations, refactors, and optimization tasks, but breaks down on multi-week work without fast, verifiable feedback. [1, 2]

TRY THIS

- **Turn one-off prompts into a bounded `go` run (Jason Zhou).**
  1. Enable the feature: `codex features list` → `codex features enable go`. [2]
  2. Give it a verifiable brief, e.g. `go "migrate codebase from JS to TS, verify screens stay exactly the same visually with Playwright"`. [2]
  3. Check status with `go`; interrupt with `go pause` or `go clear`; branch a side investigation with `side`. [2]

- **Do an alignment interview before the agent writes code (Jason Zhou; Vincent from OpenClaw).** First dump the context: what the project is, what the user cares about, what "bad" looks like, what you already tried, and the bugs it keeps missing; then let the agent ask questions before it starts. Also quantify done — e.g. "find 20 discrete new issues, propose fixes, push fixes to a branch, log results" — because fuzzy goals like "keep fixing" make the model stop early or wander. [2]
- **Move the contract into files with Go Buddy.** Run `mpx go buddy`, then `goprep` to generate a `go.md` with the request, constraints, stop rules, and loop details plus a `state.yaml` task file. Then run `go @go.md` so every loop re-reads the same contract instead of relying on chat memory; Jason shows this taking a vague game idea to a functional game with generated assets. [2]
- **Debug in disposable sandboxes, not your local machine (Peter Steinberger).** His loop is simple: ask Codex to recreate the exact failing state in an ephemeral Crabbox, verify the bug, fix it, then verify the fix. The upside: no polluted local environment and enough isolation to run 10+ sessions in parallel without slowdown. Crabbox [3, 4]

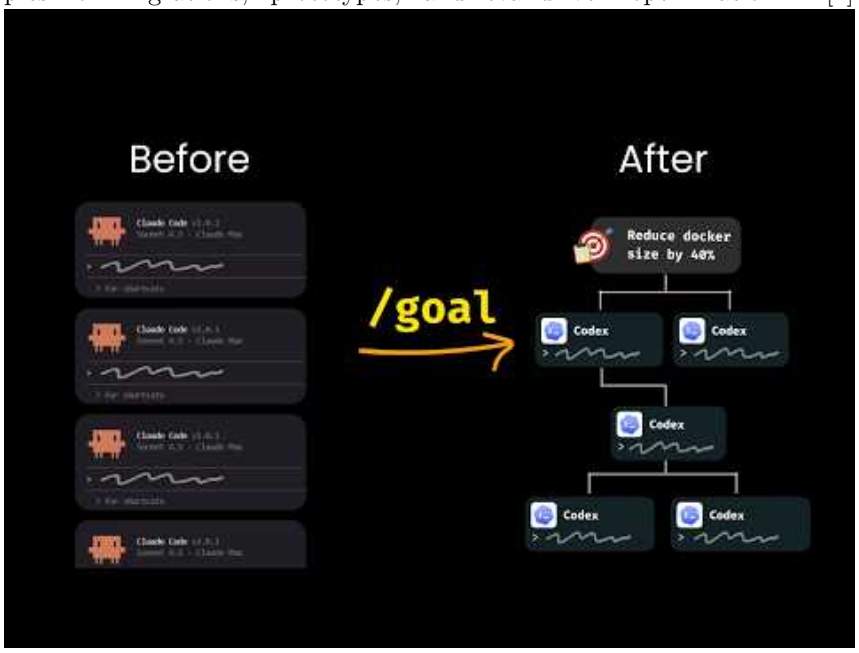
WHAT SHIPPED

- **Peekaboo 3.0** — Peter Steinberger says this is the biggest release since 2.0: action-first macOS computer use, unified screenshot + UI detection, cleaner JSON across CLI + MCP, and better snapshots. His framing is the interesting part: he started it last year, but says the models weren't good enough then; now they are. peekaboo.sh [5]
- **Crabbox Windows terminal handling** — strong enough that Steinberger says Codex could E2E-fix gifgrep's animated GIF terminal rendering. Better terminal substrates clearly matter for what agents can validate. crabbox.sh · gifgrep.com [4]
- **Codex/Hermes goal-based agents** — Jason Zhou says both now ship a `/goal` feature, and his field report is clear: most users are holding it wrong. Adoption signal: Jason ran a nine-hour migration with Codex, and Vincent from OpenClaw ran it for three days across 30 rounds. [1, 2]
- **GitHub Copilot review feels materially better** — DHH says the review feature, *not* the local CLI, went from roughly a 1/10 to 7/10 hit rate on real issue finding. Caveat: it still re-raises concerns that were already dismissed with a . [6, 7]
- **Codex is getting strong iOS build reviews** — Romain Huet says it can design screens, write Swift with GPT-5.5, run the app in Simulator without opening Xcode, and click around with computer use to test it. Omar Shahine says a single-shot app built with goals got him about 95% of the way there and felt much better than Claude Code. [8, 9]

- **OpenClaw loop speedups** — Steinberger says caching work is making Telegram loops in OpenClaw 5-100x faster. [10]

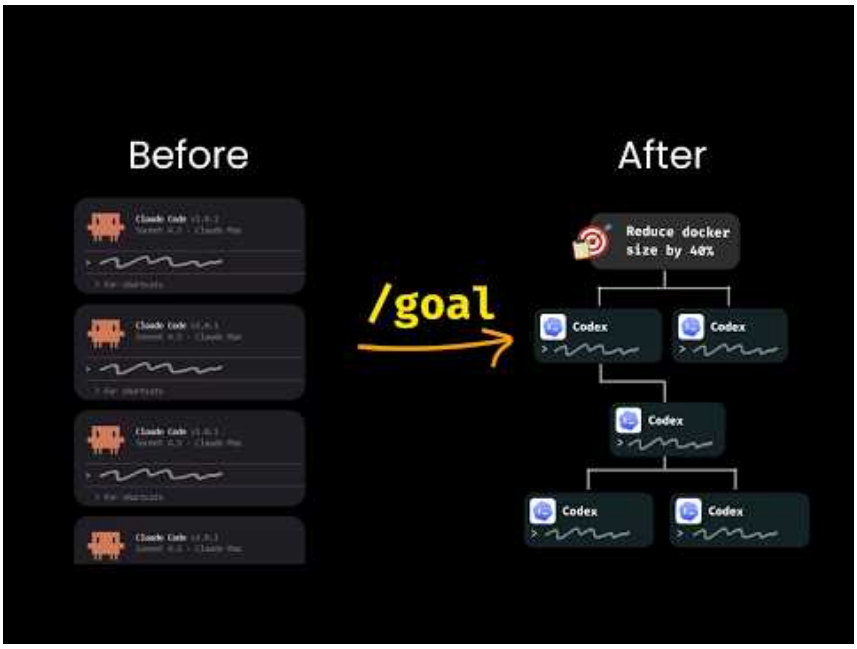
GO DEEPER

- **5:36-6:55 — How to write a `go` prompt that actually terminates.** Short, high-signal clip on the prompt anatomy: objective, constraints, validation, and stop conditions, with examples for migrations, prototypes, and eval-driven optimization. [2]



I used /goals command wrong… Here are all tips & mistakes (5:36)

- **9:59-12:08 — Where `go` stops working, and how `MISSION.md` can take over.** Jason draws the boundary cleanly: `go` is for hours-long coding loops; multi-week goals need scheduled reruns, stored summaries, explicit metrics, and human-in-the-loop escalation. [2]



I used /goals command wrong… Here are all tips & mistakes (9:59)

- **Study Crabbox.** The durable pattern is exact-state, disposable execution environments for reproduce → fix → verify loops, plus enough isolation to fan out lots of sessions in parallel. [3, 4]
- **Study peekaboo.sh.** Peekaboo 3.0 is a practical reference for action-first macOS computer use and a cleaner CLI/MCP data model. [5]

Editorial take: the best coding-agent setups are getting less magical and more operational — explicit contracts, state files, disposable environments, and hard verification beat "just let the agent cook." [2, 3]

Sources

1. X post by @jasonzhou1993
2. I used /goals command wrong… Here are all tips & mistakes
3. X post by @steipete
4. X post by @steipete
5. X post by @steipete
6. X post by @dhh
7. X post by @dhh
8. X post by @romainhuet
9. X post by @OmarShahine
10. X post by @steipete