PhysicsLab — Code See V5.3 Roadmap & Agent Prompts

Visual monitoring of modularity, extensibility, and runtime activity
(Developer/Contributor tool)

Document date: 2025-12-30

1) Vision for Code See

- Code See is not only an architecture diagram. It is a live, navigable view of the whole app: packs/blocks/modules, services, jobs, runs, files/artifacts, logs, and error/failure states.
- It must help contributors understand the app quickly: where extension points live, what depends on what, and what is happening right now (activity trails across edges).
- It must remain safe: it should not mutate state just by observing, and it must have a Safe Viewer mode for crash inspection and snapshot-first browsing.
- Everything should be drillable: root → pack/topic → block → subcomponent → artifacts (runs, exports, snapshots, logs) with clear breadcrumbs.

2) Current baseline (already implemented in V5.0–V5.2)

- Graph canvas interactions: pan/zoom/selection, drag-to-organize, subgraph navigation, layout persistence per workspace.
- Atlas collectors + snapshot IO + snapshot diff.
- Badges/icon sheet (color + mono) with tooltips, inspector, and severity borders.
- Lenses + quick filters + filter clarity status.
- Detachable Code See window + geometry persistence.
- Resolution module v1 (UI scale + density) and Code See scaling.
- Expected-vs-Actual (EVA) checks + mismatches (from validation/export/import), persisted in recent-check buffer.
- Safe Viewer mode + crash capture + crash view.
- Runtime bus bridge + activity persistence, plus thread-safe GUI ingestion; travel pulses and activity tint decay (v2).
- Toolbar declutter into menus/dialogs (in progress).

3) Agent prompt guardrails

- ⚠️ Codex is running as a full-access agent (not restricted Agent mode). Treat the Touchlist/Forbidden list as strict boundaries.

- Prefer incremental slices: implement one milestone, self-verify, then stop. Do not “bundle” unrelated refactors.
- Do not commit runtime junk. Tighten .gitignore as needed, but do not delete user data unless explicitly instructed.

4) V5.3 milestone overview

Milestone	Theme	Primary outcomes	Risk	Suggested reasoning
V5.3a	Deep hierarchy & extensibility map	Subgraphs inside subgraphs (packs→blocks→subcomponents) + plugins/extensions/dependencies lens	Medium	High
V5.3b	Runtime coverage expansion	Lab runs/jobs/storage/install/uninstall/log pointers → live spans/pulses + persistent activity	High	High
V5.3c	Topology evolution	Graph grows/shrinks with inventory changes; diff overlays; safe reconciliation	Medium	High
V5.3d	Pulse trails v3	Multi-pulse trails, fade-on-arrival, per-topic styling, adjustable intensity/duration; detached window parity	Medium	Medium
V5.3e	UX declutter &	Compact toolbar	Low	Medium

	controls	+ "View/Live/Filters" popups, presets, and keyboard shortcuts; reduce panel clutter		
V5.3f	Verification harness (Crash Instigator)	Dev-only test module that simulates failure/activity to validate Code See end-to-end	Low	Medium

5) V5.3 agent prompts (copy-paste ready)

Each prompt follows the standard format: Model + reasoning effort at top, then Goal → Touchlist → Forbidden list, plus verification and git checkpoint commands.

V5.3a — Agent Prompt

Model: GPT-5.2-Codex | Thinking level: High

Justification: Multi-file feature work (hierarchy model + collectors + subgraph navigation rules) with correctness requirements.

Goal

Make Atlas support true deep drill-down: packs → blocks → subcomponents → artifacts, and add a first-class Extensibility/Dependencies lens so contributors can understand where plugins/extensions live and what depends on what.

Touchlist

- app_ui/codesee/collectors/** (extend collectors or add a new collector for plugins/extensions/dependencies)
- app_ui/codesee/graph_model.py (node/edge types, metadata, subgraph keys)
- app_ui/codesee/atlas_builder.py (build deep hierarchy graphs; stable graph_ids)
- app_ui/codesee/screen.py (subgraph navigation rules; breadcrumb titles; inspector fields)
- docs/codesee/** (update semantics + lens documentation)

Forbidden list

- Do NOT refactor unrelated parts of the app, rename large folders, or change public APIs outside the Touchlist.
- Do NOT delete or rewrite user runtime data (data/workspaces/**, data/roaming/**) unless explicitly instructed.
- Do NOT introduce new third-party dependencies without explicit approval.
- Do NOT change Management Core job behavior or storage policies.
- Do NOT change content schemas/loader behavior beyond adding read-only inspection hooks needed for the graph.

Deliverables

- Atlas shows deep hierarchy with subgraphs inside subgraphs (Pack → Block → Subcomponent).
- New lens: “Extensibility/Dependencies” with clear edge labels (e.g., provides/consumes/depends-on).
- Inspector shows extension points: which registry/pack/manifest declared the item, and any dependency list.
- All graph IDs are Windows-safe and stable across sessions (layout persistence preserved).

Self-verify

python -m compileall app_ui content_system core_center component_runtime runtime_bus ui_system schemas diagnostics

python -m app_ui.main # open Code See (Explorer), Source=Atlas, drill down 3+ levels

Manual tests

- Explorer: Code See → Source=Atlas → open Pack → open Block → open Subcomponent (3+ breadcrumb levels).
- Switch Lens to Extensibility/Dependencies → verify edges change, and inspector shows dependency lists.
- Close/reopen Code See → confirm layout persists at each graph depth.

Git checkpoint (run these 3 commands)

git add -A

git commit -m "feat(V5.3a) codesee deep hierarchy subgraphs and extensibility lens"

git push

V5.3b — Agent Prompt

Model: GPT-5.2-Codex | Thinking level: High

Justification: Infrastructure-level runtime coverage expansion touching multiple producers (jobs/runs/labs) and live UI ingestion.

Goal

Expand runtime coverage so Code See visibly reflects real app activity (runs, jobs, installs/uninstalls, cleanup, exports), with live multi-pulse trails across relevant edges and persistent activity state even if Code See is opened later.

Touchlist

- app_ui/codesee/runtime/** (hub/events/hooks/spans/pulse engine; persistence)
- app_ui/codesee/bus_bridge.py (add topic mappings; ensure GUI-thread ingestion)
- app_ui/main.py and/or existing UI screens (publish minimal read-only activity events/spans)
- core_center/bus_endpoints.py (ONLY if needed: expose read-only activity summary endpoints)
- docs/codesee/** (document new runtime signals + meaning)

Forbidden list

- Do NOT refactor unrelated parts of the app, rename large folders, or change public APIs outside the Touchlist.
- Do NOT delete or rewrite user runtime data (data/workspaces/**, data/roaming/**) unless explicitly instructed.
- Do NOT introduce new third-party dependencies without explicit approval.
- Do NOT add high-volume telemetry or logging by default; keep instrumentation minimal and bounded.
- Do NOT change existing job terminal-state behavior/timeouts.

Deliverables

- Live pulses visible for common real actions: job queue/complete, cleanup, module install/uninstall, export/import, run creation/prune.
- Multi-dot trail (not a single dot) with configurable duration/intensity; fade on arrival at target node.
- Persistent activity state per workspace (recent events/spans survive closing Code See and reopening).
- Detached Code See window shows the same live activity as embedded view.

Self-verify

python -m compileall app_ui content_system core_center component_runtime runtime_bus ui_system schemas diagnostics

python -m app_ui.main # trigger System Health cleanup/job and verify Code See pulses

python -c "from app_ui.codesee.runtime.hub import hub; print('hub ok', hub())"

Manual tests

- Trigger a cleanup job (System Health) → see pulses travel and activity tint decay on involved nodes.
- Install/uninstall a pack/module (if available) → see graph reflect activity and changes.
- Open Code See in Window → verify same live pulses appear.

Git checkpoint (run these 3 commands)

git add -A

git commit -m "feat(V5.3b) codesee runtime coverage expansion with persistent multi-pulse trails"

git push

V5.3c — Agent Prompt

Model: GPT-5.2-Codex | Thinking level: High

Justification: Requires careful reconciliation logic (inventory changes + snapshots + layout) across multiple sources.

Goal

Make the graph topology evolve with the app: nodes/edges appear or disappear as inventory changes (installed/uninstalled, enabled/disabled), and snapshots/diff make these changes explicit (Added/Removed/Changed), without breaking layout persistence.

Touchlist

- app_ui/codesee/collectors/** (inventory refresh hooks + stable IDs)
- app_ui/codesee/snapshot_io.py + diff.py (diff overlays for topology change; per-node change flags)
- app_ui/codesee/screen.py + canvas/scene.py (render “Added/Removed/Changed” overlays, filter chips, status row counts)

Forbidden list

- Do NOT refactor unrelated parts of the app, rename large folders, or change public APIs outside the Touchlist.

- Do NOT delete or rewrite user runtime data (data/workspaces/**, data/roaming/**) unless explicitly instructed.
- Do NOT introduce new third-party dependencies without explicit approval.

Deliverables

- When a pack/block/module is installed/uninstalled or enabled/disabled, Atlas refresh shows nodes added/removed.
- Diff Mode highlights Added/Removed/Changed nodes (and provides quick filters).
- Layout persistence remains stable: removed nodes don't corrupt saved layout; re-appearing nodes reuse their prior position when possible.

Self-verify

python -m compileall app_ui content_system core_center component_runtime runtime_bus ui_system schemas diagnostics

python -m app_ui.main # do an install/uninstall action and compare snapshots

Manual tests

- Capture snapshot → change inventory (install/uninstall or enable/disable) → capture snapshot → Diff Mode shows Added/Removed.
- Toggle filters to show only Added/Removed and confirm counts match status row.

Git checkpoint (run these 3 commands)

git add -A

git commit -m "feat(V5.3c) codesee topology evolution and snapshot diff overlays"

git push

V5.3d — Agent Prompt

Model: GPT-5.2-Codex | Thinking level: Medium

Justification: Mostly UI/animation tuning and settings; architecture already exists. Keep scope tight to avoid churn.

Goal

Polish pulse trails (v3): make pulses clearly visible, multi-dot, with arrival fade and configurable settings. Ensure no thread/timer warnings, and keep parity between embedded and detached windows.

Touchlist

- app_ui/codesee/canvas/scene.py
- app_ui/codesee/canvas/items.py
- app_ui/codesee/view_config.py (pulse settings persistence)
- app_ui/codesee/screen.py (Pulse Settings dialog UX)

Forbidden list

- Do NOT refactor unrelated parts of the app, rename large folders, or change public APIs outside the Touchlist.
- Do NOT delete or rewrite user runtime data (data/workspaces/**, data/roaming/**) unless explicitly instructed.
- Do NOT introduce new third-party dependencies without explicit approval.

Deliverables

- Pulse visibility improved: trails are obvious, not “blink and disappear”.
- Fade on arrival at target node; optional “linger” setting for stuck/active states.
- Settings: intensity, travel duration, trail length, decay, per-topic enable/disable.
- No QTimer thread warnings; no crashes when switching lens/source or closing windows mid-pulse.

Self-verify

python -m compileall app_ui content_system core_center component_runtime runtime_bus ui_system schemas diagnostics

python -m app_ui.main # enable Live, trigger a job, tune pulse settings

Manual tests

- Enable Live → trigger cleanup/job → confirm multi-dot pulse trails travel across edges and fade at target.
- Open detached Code See → confirm same pulse behavior.
- Rapidly switch lens/source while pulses are active → confirm no warnings/crashes.

Git checkpoint (run these 3 commands)

git add -A

git commit -m "fix(V5.3d) codesee pulse trail visibility and settings polish"

git push

V5.3e — Agent Prompt

Model: GPT-5.2-Codex | Thinking level: Medium

Justification: UI declutter pass; should be mostly screen.py layout changes with minimal risk.

Goal

Reduce toolbar clutter without losing power: move advanced controls into dropdown menus/popups, add presets, and keep only the core controls visible by default.

Touchlist

- app_ui/codesee/screen.py (toolbar/menu restructuring)
- app_ui/codesee/view_config.py (persist presets)

Forbidden list

- Do NOT refactor unrelated parts of the app, rename large folders, or change public APIs outside the Touchlist.
- Do NOT delete or rewrite user runtime data (data/workspaces/**, data/roaming/**) unless explicitly instructed.
- Do NOT introduce new third-party dependencies without explicit approval.

Deliverables

- Top row stays compact: Lens, Source, Snapshots, Live toggle, Open in Window, Icon Style, Refresh.
- Advanced: Filters/Layers/Badges/Diff/Pulse settings moved into menus or a small popover panel.
- Presets: save/load a small set of view presets per workspace (optional).

Self-verify

python -m compileall app_ui content_system core_center component_runtime runtime_bus ui_system schemas diagnostics

python -m app_ui.main # verify controls are accessible and not duplicated

Manual tests

- Find and toggle every previous feature (filters, layers, diff, pulse settings) via menus/popovers.
- Confirm no accidental regressions in snapshot capture/load and subgraph navigation.

Git checkpoint (run these 3 commands)

git add -A

git commit -m "chore(V5.3e) codesee toolbar declutter via menus and presets"

git push

V5.3f — Agent Prompt

Model: GPT-5.2-Codex | Thinking level: Medium

Justification: Adds a small, temporary verification harness; keep it isolated and removable.

Goal

Add a temporary “Crash Instigator” / activity harness to validate Code See end-to-end: emit spans, trigger mismatches, simulate crash records, and generate inventory changes in a controlled way. Must be easy to remove or gate behind a flag.

Touchlist

- app_ui/codesee/** (add a small harness module and a hidden UI entry or CLI flag)
- docs/codesee/** (how to use the harness; how to remove it later)
- .gitignore (ensure harness output is ignored if it writes runtime artifacts)

Forbidden list

- Do NOT refactor unrelated parts of the app, rename large folders, or change public APIs outside the Touchlist.
- Do NOT delete or rewrite user runtime data (data/workspaces/**, data/roaming/**) unless explicitly instructed.
- Do NOT introduce new third-party dependencies without explicit approval.
- Do NOT leave destructive actions enabled by default. Gate behind an explicit env var or CLI flag.

Deliverables

- Harness can: emit synthetic pulses/spans across a known path; create an EVA mismatch; write a fake crash record for Safe Viewer.
- Code See clearly shows the activity (embedded + detached), and snapshots/diff capture the changes.
- One switch to disable/remove (e.g., --codesee-harness or PHYSICSLAB_CODESEE_HARNESS=1).

Self-verify

python -m compileall app_ui content_system core_center component_runtime runtime_bus ui_system schemas diagnostics

python -m app_ui.main # enable harness flag and run the instigator actions

Manual tests

- Enable harness → click “Emit test activity” → confirm visible pulse trails + activity tint + inspector records.
- Trigger harness EVA mismatch → Only mismatches filter shows it; diff captures it.
- Trigger harness crash → Safe Viewer shows crash indicator and Build info.

Git checkpoint (run these 3 commands)

git add -A

git commit -m "test(V5.3f) add gated codesee crash-instigator verification harness"

git push