Piper - Living Context

🔝Always-Read Section

Rules

- **Canvas First** → Before every reply, re-read this top section (down to KGB section).
- User is the architect; ChatGPT is the railmaster/developer. Provide plain-English, step-by-step guidance and drive the rails without asking “what next?”.
- If any instruction conflicts, **this section overrides chat history**.
- Maintain state across replies (Rails, Known Issues, KGB, Architecture, Call Graph).
- ChatGPT updates Canvas only for rails progress, KGB title, and callgraph.
- **Mirrors-first** → Before proposing/applying any change, **open and re-read** the relevant mirrors on the drives. Compare mirrors vs canvas; if they diverge, call it out and resolve before patching.

Development Flow (Macro)

- Rails discipline & philosophy (macro-level): stay on rails; one change → smoke → tag; KGB/parking rules; auto-unpark; no piling fixes; file size limits; geometry in layout_constants.
- Cross-ref: see Execution Rules for micro protocol.

Execution Rules

- **Mirrors-first.** For any edit, open Drive/Dropbox mirrors and base the patch on those lines.
- **Patch format.** `BEFORE → AFTER` with 1–2 **bookend** context lines above/below. State exact file + placement (e.g., *inside ``* *below callback X*).
- **No hypotheticals.** No “likely/if exists”. Use mirror truth only.
- **Smoke (≤3 lines).** Describe the minimal run check.
- **KGB tag.** Provide the next tag on pass. Format: KGB-YYYY-MM-DD_\<RAIL>\<step>_\<slug>
- **Residue check.** After a green step, re-open mirrors for **touched files only**; list any dead code/dupe constants/import drift and propose one-line removals.
- **Parking.** If risky/blocked, park with rationale; Railmaster auto-unparks later.
- Enforce **Rules** every reply.
- Keep non-essential sections untouched unless explicitly updated.

- After code changes: update **Call Graph, Rails, Known Issues, Parked, and KGB**.
- If inconsistencies are found between project mirrors and this canvas, they must be called out and corrected immediately.
- Piper Guides are external; always pull them from Drive/Dropbox mirrors if needed.
- All geometry/theme numbers must come from `ui/layout_constants.py`.
- Prefer components (ui/components/*) and thin entries (<150 lines).

Mirror Rules

- Mirrors are the only Truth = .txt mirrors (Drive + Dropbox).
- **Always Check**: never ask user to check, never assume, never say "if it exist".
- **If both mirrors fail rule** → only say: `> Need google Drive/Dropbox access to continue.`

Response Discipline

Every response must follow the Step Response Template structure and any updates applied to the Canvas so the user can confirm.

Response Template

Canvas: confirm if Always Read part read and any other section

Chapter: | Ring: \<Core/Services/UI> | Scope: ✅🟨🛑

Plan: one tiny change only

Patch: (full file or full function; exact locations if needed)

Smoke (≤3 lines): run → expect → pass condition

KGB: tag on pass / revert+park on fail

Parking: list any out-of-scope or failing items carried forward

Compliance: rings clean; invariants honored; SAFE_MODE; no drift

Examined: list mirrors/documents actually read

Idiotproofed: confirm user edits were correct and mirrors were re-opened and re-read

Guardrail: append one-liner — "If this correction does not fix the issue, [revert/maintain]."

☣️Drives — Piper Guides

- **Key Files.txt** = File role map (GUI orchestration, state dot, hb text, layout const, scroll, avatar).
- **Anchor.txt** = Project rails/constitution (rules, chapters, invariants, runbook).
- **CLI+GUI Trailing & Testing Guide.txt** = 2-terminal run instructions + trailing mechanics + smoke/fix.
- **State Color Coding Guide.txt** = State→color map + dwell + code snippet + smoke test.

Architecture

Root: `C:\Piper\scripts`

```
common/
  config.py        # app config helpers (shared)
  log_utils.py     # project log helpers (renamed from logging.py; no stdlib
shadow)
  types.py         # simple datatypes
  utils.py         # small shared utilities

core/
  bg_poller.py     # background tick/poll loop helpers
  bridge.py        # glue between core events and UI/log
  core_app.py      # top-level core bootstrap
  core_commands.py # command handling (CLI/Dev tools)
  core_machine.py  # state machine (SLEEPING→…→SPEAKING)
  event_queue.py   # queued events
  events.py        # event definitions
  flags.py         # runtime feature/DEV flags
  poll_helpers.py  # polling helpers
  router.py        # routes events to handlers
  startup.py       # init order, env
  state_defs.py    # state enums/labels
  timers.py        # timing utilities
  transition_plan.py # transition map/guard rails

entries/
  _app_gui_entry_impl.py # GUI glue: imports dpg_app/panes; schedules refresh
(target ≤150 lines)
  app_cli_entry.py       # CLI entrypoint
  app_gui_entry.py       # thin shim → _app_gui_entry_impl.run()
```

```
  app_wake_entry.py      # wake-only demo/entry

services/
  adapters/mock_asr_wake.py   # test double (wake+ASR)
  asr/vosk_adapter.py         # ASR adapter (modular)
  memory/                     # memory svc placeholder
  persona/                    # persona svc placeholder
  tts/speak_once.py           # single-shot TTS
  tts/tts_manager.py          # TTS manager stub/interface
  wake/porcupine_adapter.py   # wake-word adapter (modular)
  base.py                     # service base helpers
  cli_prompt.py               # current_prompt(), format_line()
  persona_adapter.py          # persona bridge
  asr_vosk.py                 # ⚠️ legacy shim → move to services/old/
  wake_porcupine.py           # ⚠️ legacy shim → move to services/old/

ui/
  components/
    chat_pane.py        # chat render area (DPG; dynamic refresh)
    controls_pane.py    # dev controls (flag-gated)
    logs_pane.py        # live log tail
    status_pane.py      # small indicators
  helpers/
    avatar_fix.py       # avatar sizing/crop helpers (pending tune)
    chatlog_writer.py   # (GUI) chat logging
    dev_adapters.py     # dev-only bridges
    dev_controls_mount.py # wire dev controls
    gui_ingest.py       # parse tail lines → UI state cues
    gui_loop.py         # GUI main loop helpers
    header_bridge.py    # fallback hb_label writer (shim)
    header_utils.py     # header utilities
    init_core.py        # init orchestration helpers
    layout_utils.py     # layout helpers
    refresh_core.py     # push-only refresh helpers
    scroll_utils.py     # autoscroll logic (single callback; multi-tag aware)
    sink_utils.py       # sinks for tailed lines
    state_dot.py        # state→color mapping utils
    tag_utils.py        # tag parsing
    theme_utils.py      # theme helpers
    viewport_utils.py   # viewport sizing/pos
  pane_parts/
    avatar_pane.py      # avatar region
    chat_region_pane.py # chat region (layout-only; renamed to avoid name
collision)
    header_bar.py       # SINGLE AUTHORITY: set_state_dot(), heartbeat label
    logs_pane.py        # logs region
  dpg_app.py            # DearPyGui bootstrap + viewport
```

```
  heartbeat.py          # hb text source
  layout_constants.py   # ALL geometry/theme numbers
  panes.py              # compose components/pane_parts (no header logic)
  state_header.py       # state label helpers (reads from log/model)
  tailer.py             # file tailer for core.log
  theme.py              # theme palette
  _panes_impl.py, dev_tools.py, ipc_child.py
```

Call Graph

```
entries.app_gui_entry:run()

→ entries._app_gui_entry_impl:run()

→ ui.dpg_app:run(viewport, flags)
 → ui.panes:init_ui()

   → pane_parts.header_bar:init()
   → pane_parts.chat_region_pane:init()
   → pane_parts.logs_pane:init()
   → components.controls_pane:init()   [if DEV flag]

 → ui.panes:refresh_ui(state_text)

   → (push-only) helpers.refresh_core:*        # NO header/state parsing
   → pane_parts.header_bar.set_state_dot(name) # THE ONLY header authority
   → helpers.header_bridge.apply_header_updates(...)    # fallback label update
only
   core.core_machine (model-driven later)

→ services.cli_prompt.current_prompt()/format_line()

→ core.logbus.event()/state() → file

→ ui tails logs (no dwell; model triggers state)
```

KGB

Latest realized snapshot tag: `KGB-2025-09-12_B04.9_H02c_avatar_self_stabilizing_fit`

⬇️Read-When-Needed Section

Rails

Phase B — UI Modularization

- ✅Collapsed: B01–B04.8 (dpg_app, controls, chat/log panes, helpers extracted, entry delegate, header authority, autoscroll polish, name-collision fix)
- ✅Collapsed: B04.9 — Avatar fix (post-layout self-stabilizing fit; resize hook verified)

Phase H — Cleanup & Closure

- ✅Collapsed: H01, H02a (dead placeholders + dwell remnants)
- 🟨**H02b — Entry split (thin glue)**
- ✅Slice 1: gui_loop delegation
- ⏸️Slice 2: DPG bootstrap peel (parked under B04)
- ⬜ Slice 3: move tick/parse helpers
  - **Issue:** Prior attempt froze GUI (nonlocal scope errors, persona header freeze). PARKED until safe scaffold prepared.
- ✅Collapsed: H02c — Avatar scale fix (post-layout self-stabilizing fit; resize hook verified)
- ✅Collapsed: H02d — Autoscroll tuning (chat/logs autoscroll fixed; breathing room verified)
- ⬜ H03 — Wrap & snapshot

Other Phases

- ✅Collapsed: A01–A02 (baseline restored, runbook verified)
- ✅Collapsed: D01, D02 (legacy shims quarantined, imports canonicalized)
- ✅Collapsed: B03, C01/C02, E01/E02, F01/F02, G01/G02

PARKED

- **H02b — Slice 2: DPG bootstrap peel**
  Parked under B04 while UI modularization stabilizes; unpark after B04.9.

- **Avatar: unify fix helper**
  Two `post_layout_fix(...)` variants exist (`ui/pane_parts/avatar_pane.py` used; `ui/helpers/avatar_fix.py` legacy). Parked as non-urgent tidy; unpark only when rails are clear.

- **Header symbols misrender**
  Former dots (•) show as odd characters; needs font/encoding fix.

- **Header trailing symbol**
  Rightmost separator has no field after it; decide whether to remove or add a field.

- **Theme color**
  GUI is grey; user prefers light blue theme. Previous attempts were difficult; retry later.

- **Avatar pane tweak**
  Slight adjustment desired for avatar region/pane. Low priority.

Known Issues

- **Entry ingest delegation (H02b · Slice 3) froze GUI.**
- Symptom: Delegating `_classify_and_buffer` caused nonlocal scope errors and persona header freeze.
- Status: **PARKED** under H02b until a safe scaffold is prepared (verified nonlocal bindings + persona updates). Last green maintained.