Charkoal – Developer Canvas for Visual Code Mapping

Overview

Charkoal is a Visual Studio Code extension that gives developers a native, infinite canvas inside the IDE to visually map, document, and reason about code. It's ideal for understanding complex architectures, onboarding teams, and designing new features by linking visual diagrams to real code.

For the **Auterity** and **RelayCore** projects, Charkoal enables architecture sketching and agent flow visualization that links your design directly to the codebase. With intelligent linking, symbol sketching, and modular canvases, Charkoal bridges visual workflows with backend orchestration.

Core Features

1. Infinite Canvas Workspace

- Visualize Auterity agent flows and RelayCore routing pipelines across a free-form canvas.
- Drag in Porter + Driver agents, model profiles, and plugin components as modular nodes.

2. One-Click Code Symbol Capture

- Capture Auterity orchestration modules or RelayCore steering rules directly from FastAPI or LangGraph code.
- Retains function name, file, and context for traceability.

3. Intelligent Linking

- Link nodes like "Token Router" to their Python definitions.
- Represent RelayCore "provider -> model -> action" relationships with visual arrows.
- Use `[[prompt-handler.py]]` to mark function call paths.

4. Nesting and Composition

- Create system-level canvases (e.g., Auterity Agent Framework), then zoom into sub-canvases (e.g., recursive prompt planner).
- RelayCore rule chains can nest into model profile subflows.

5. Git-Friendly JSON Format

- `.canvas.json` stored next to orchestration code or agent manifests.
- Easily reviewed alongside Pull Requests.

6. Developer Investigation Board

- Sketch new Porter agent roles.
- Debug cross-agent prompt handoffs.
- Refactor LangGraph chains with inline notes.

7. Language Compatibility

- Ideal for FastAPI, Python, TypeScript (Auterity stack), and CLI-layer routing in RelayCore.
- Handles modular file layouts well.

8. In-IDE Productivity

- Launch canvas, sketch selections, and update links all within VS Code.

Key Use Cases

Use Case	Description
System Architecture Mapping	Visualize backend services, plugins, and LangGraph flows in Auterity & RelayCore
Agent Workflow Modeling	Model Porter/Driver agent lifecycles and RelayCore rule evaluation logic
Reverse Engineering	Sketch token flow or prompt stacks from existing code
Feature Planning & Refactoring	Show "before vs. after" views of semantic router or UI orchestration
Developer Onboarding	Link documentation to actual agent canvases
Canvas-Powered PRs	Explain large refactors to Porter/Relay sub-agents or steering logic

---

Mapping to Auterity & RelayCore

Charkoal Concept	Auterity/RelayCore Equivalent
Canvas	Agent/steering flow or routing path visual
Node	Porter, Driver, Sub-agent, model selector, API adapter
Links	Semantic routing decisions, recursive logic branches
Nested Canvases	Agent internals, relay plugin stacks, LangGraph edge logic
Markdown Notes	Inline prompt templates, rule logic, or agent role commentary
JSON Canvas File	Saved representation of full agent or steering flow

Integration Strategy

1. **Design Before You Build**
2. Sketch Porter + Driver agents or RelayCore pipelines in Charkoal before implementing in code.
3. Visualize `task -> agent -> prompt -> model` flows.
4. **UI → Canvas Mapping**
5. Map Auterity's drag-and-drop blocks to Charkoal canvas nodes.
6. Use markdown links to highlight file paths or class names.
7. **Embed in DevOps**
8. Save `.canvas.json` next to agent definitions or model profile templates.
9. Add as artifacts to PRs or versioned config bundles.
10. **Version Controlled Documentation**
11. Tie canvas diagrams to refactoring branches.
12. Link visual diffs to Git commits across Auterity and RelayCore teams.

Canvas Plan: RelayCore + Auterity Integration Mapping

Phase 1: Visualize RelayCore Routing

- Sketch top-down relay pipeline: Incoming request → YAML rule engine → Provider/model selector → response.
- Include `token_count`, `content_match`, and `agent_profile` rules as nodes.

Phase 2: Visualize Porter + Driver Flows

- Build a modular canvas with `Porter -> Task Parser -> Driver -> Model Runner -> Logger`.
- Add inline notes about agent intent and prompt metadata.

Phase 3: Link Steering Rules to Agent Contexts

- Create connection maps from Auterity agents to their preferred RelayCore routing logic.
- Use nested canvases for recursive drivers or retry logic.

Phase 4: Shared Observability Canvas

- Design a joint observability and analytics view: Token usage → Model hit rates → Caching layer effectiveness.
- Nodes include: `Token Tracker`, `Cost Reporter`, `Cache Layer`, `Usage Dashboard`.

Getting Started

- Install "Charkoal – Visualize Your Codebase" from the VS Code marketplace.
- Run `Charkoal: New Canvas` from the command palette.
- Use `Charkoal: Sketch the Symbol/Selection` to add code, logic, or metadata.
- Save canvas as `.canvas.json` and commit to version control.

Recommended folders: - `/auterity/agents/canvases` - `/relaycore/steering/canvases` - `/shared/visuals/observability.canvas.json`

Summary

Charkoal enables precise, visual-first development for complex AI systems. For Auterity and RelayCore, it provides the connective tissue between your agentic logic and routing pipelines. Use it to plan, debug, onboard, and document—directly inside your IDE. By embedding canvas thinking into the dev cycle, both platforms can scale agent orchestration and model routing with confidence and clarity.