Runeforge Canvas: Build Spec

what we're building

a click-driven thinking environment that renders reasoning as a graph, not a linear chat. user clicks operations, nodes spawn as children, context accumulates visually. built on textual (python TUI framework), piggybacking on claude code's oauth for zero-cost API calls.

why not just use claude chic

claude chic gets slow with extended conversations because it renders EVERY message as an expandable widget. scrolling recalculates layout for hundreds of widgets.

deeper problem: chic assumes engineering-style separability (parallel workers → merge). research contexts are ENTANGLED—insight from branch A changes what you'd ask in branch B. can't parallelize naively.

core architecture

data model

```
@dataclass
class CanvasNode:
    id: str
    type: NodeType  # ROOT, OPERATION, USER, LINK
    operation: Optional[str]  # "@antithesize", "@excavate", etc.
    content_compressed: str   # ≤100 chars, always available
    content_full: str         # loaded on expand
    parent_id: Optional[str]
    children_ids: list[str]
    links_to: list[str]       # cross-references (entanglement)
    context_snapshot: list[str]  # which nodes were in context when created

@dataclass
class Canvas:
    nodes: dict[str, CanvasNode]
    root_id: str
    active_path: list[str]  # ids from root to current focus
```

why this won’t get slow

component	complexity	why
minimap	O(n)	tiny boxes, no rich content
active path	O(depth)	only focused branch renders fully
siblings	O(1) per	compressed pills, not full widgets
expansion	lazy	content_full only on click

500 nodes, max depth 10 → only 10 widgets render with full content.

layout

```
┌─────────────────────────────────────────────────────────────┐
│  RUNEFORGE CANVAS                                           │
├─────────────────────────────────────────────────────────────┤
│                                                             │
│  ┌───────────────────────────────────────────────────────┐  │
│  │ GRAPH MINIMAP (30% height)                            │  │
│  │     [root]──[A]──[A.1]                                │  │
│  │              └──[B]──[B.1]───┐  cross-link            │  │
│  │                       [A.2]◄─┘                        │  │
│  │  click node to focus                                  │  │
│  └───────────────────────────────────────────────────────┘  │
│                                                             │
│  ┌───────────────────────────────────────────────────────┐  │
│  │ ACTIVE PATH (50% height, scrollable)                  │  │
│  │                                                       │  │
│  │ ┌─ root ────────────────────────────────────────────┐ │  │
│  │ │ Who are my first 1000 users?                      │ │  │
│  │ └───────────────────────────────────────────────────┘ │  │
│  │ ┌─ @excavate ───────────────────────────────────────┐ │  │
│  │ │ CRUXES: ⚡friction-aware users...                  │ │  │
│  │ └───────────────────────────────────────────────────┘ │  │
│  │ ┌─ @antithesize ────────────────────────────────────┐ │  │
│  │ │ rival thesis: most blame tool, not selves...      │ │  │
│  │ └───────────────────────────────────────────────────┘ │  │
│  └───────────────────────────────────────────────────────┘  │
│                                                             │
│  ┌───────────────────────────────────────────────────────┐  │
│  │ OPERATIONS (20% height)                               │  │
│  │ [@excavate] [@antithesize] [@stressify] [@simulate]   │  │
│  │ [@backchain] [@diverge] [@synthesize] ...             │  │
│  │ [+ link to...] [+ note] [⟲ regen]                    │    │
│  └──────────────────────────────────────────────────────┘    │
└──────────────────────────────────────────────────────────────┘
```

key UX flows

click operation → spawn child

```
async def on_operation_click(self, op: str) -> None:
    focus_id = self.canvas.active_path[-1]

    # 1. immediate UI feedback (<50ms)
    self.show_spinner_on_focused_node()

    # 2. gather context (parent chain + cross-links + relevant siblings)
    context = self.canvas.get_context_for_operation(focus_id)

    # 3. build prompt from runeforge skill definition + context
    prompt = self.build_prompt(op, context)

    # 4. async API call (5-30s, UI stays responsive)
    result = await self.call_claude(prompt)

    # 5. create child node
    new_node = CanvasNode(
        id=generate_id(),
        operation=op,
        content_compressed=result[:100] + "...",
        content_full=result,
        parent_id=focus_id,
        context_snapshot=[n.id for n in context],
    )
    self.canvas.add_node(new_node)

    # 6. auto-focus new node, refresh UI
    self.canvas.set_focus(new_node.id)
```

cross-linking (research entanglement)

user working on path B realizes path A's output is relevant:

1. focus on B.node
2. click [+ link to...]

3. select A.node from minimap
4. next operation on B.node includes A.node in context

this is explicit entanglement management—branches separate by default, linked when useful.

context gathering (differs from engineering)

```
def get_context_for_operation(self, node_id: str) -> list[CanvasNode]:
    context = []

    # 1. full parent chain (always)
    for pid in self.active_path:
        if pid == node_id: break
        context.append(self.nodes[pid])

    # 2. explicit cross-links
    for link_id in self.nodes[node_id].links_to:
        context.append(self.nodes[link_id])

    # 3. older siblings (entanglement—their existence informs this node)
    parent = self.nodes[self.nodes[node_id].parent_id]
    for sib_id in parent.children_ids:
        sib = self.nodes[sib_id]
        if sib.created_at < self.nodes[node_id].created_at:
            context.append(sib)  # compressed form

    return context
```

auth: piggyback on claude code

read oauth tokens from same location claude code uses:

- linux: `~/.claude/.credentials.json`
- macos: keychain

```
{
  "claudeAiOauth": {
    "accessToken": "sk-ant-oat01-...",
    "refreshToken": "sk-ant-ort01-...",
    "expiresAt": 1748658860401,
    "scopes": ["user:inference", "user:profile"]
  }
}
```

use `claude-agent-sdk` for API calls. marginal cost = 0 after login.

runeforge operations

load definitions from skill files. core operations to support first:

operation	what it does
@excavate	assumption archaeology, surface cruxes
@antithesize	generate standalone opposition
@stressify	probe for fragility
@simulate	unfold forward in time
@backchain	work backward through causal mechanisms
@diverge	expand search radius, branch options
@synthesize	compress conflicting positions

each operation has a full procedural definition in `/mnt/skills/user/runeforge/{op}/` that should be injected into the prompt.

dependencies

```
textual>=0.50.0
claude-agent-sdk  # or whatever chic uses
rich  # textual dependency, also useful for markdown rendering
```

build order

1. **data model** — Canvas, CanvasNode, serialization to disk
2. **minimal textual app** — three panels, static content
3. **click → focus change** — prove <50ms UI latency
4. **async operation stub** — click op → delay → spawn child (no real API yet)

5. **real API integration** — claude-agent-sdk with oauth
6. **operation prompts** — load runeforge skills, build prompts
7. **cross-linking** — UI for creating/visualizing links
8. **persistence** — save/load canvas to disk
9. **minimap graph layout** — probably the hardest visual problem

open questions

1. **minimap layout**: force-directed graph in ASCII is hard. might need to punt to simple tree view initially, add proper graph viz later.
2. **streaming**: should operation results stream token-by-token into the node, or appear all at once? streaming feels more responsive but complicates the content model.
3. **concurrent operations**: can user start op on node A, then while waiting, click node B and start another op? async model supports this, but UI needs to show multiple spinners.
4. **export**: how to flatten a graph branch back to linear text for sharing/publishing?

reference

- claude chic source: https://github.com/mrocklin/claudechic
- textual docs: https://textual.textualize.io/
- runeforge skills: `/mnt/skills/user/runeforge/`