Heaven Interface Guide

Minimalist. Keyboard-first. AI-powered.

Heaven Interface is Solo Git's hybrid CLI/TUI + GUI system, inspired by Dieter Rams and Jony Ive's design principles. It provides multiple ways to interact with Solo Git, from a powerful CLI to an immersive TUI to an optional visual GUI.

Design Philosophy

Heaven Interface embodies three core principles:

1. **As little design as possible** - Only essential UI elements, no clutter
2. **Speed first** - <150ms CLI startup, instant responses
3. **Keyboard-centric** - Every action accessible via keyboard

Components

1. Enhanced CLI (Primary Interface)

The CLI is the primary interface, enhanced with Rich formatting:

```
# Standard commands with beautiful output
evogitctl repo list
evogitctl pad create "add authentication"
evogitctl test run --target fast
```

Features:
- Color-coded panels with Heaven Interface palette
- ASCII commit graph with test indicators
- Progress bars and spinners for AI operations
- Tables with syntax highlighting
- Formatted timestamps and durations

Example Output:

```
┌─── Repository: my-app ──────────────────────────────────┐
│ Repository ID:  abc123def456                            │
│ Name:           my-app                                  │
│ Path:           /home/user/.sologit/repos/my-app        │
│ Trunk Branch:   main                                    │
│ Total Workpads: 3                                       │
│ Total Commits:  42                                      │
└─────────────────────────────────────────────────────────┘

── COMMIT GRAPH ──

● ✓ ✓ a1b2c3d4 Add user authentication (John Doe)
│
● ✓ ✓ e5f6g7h8 Refactor database layer (John Doe)
│
○ ◉   i9j0k1l2 Work in progress on caching (WIP)
```

2. Interactive TUI (Optional)

Full-screen, keyboard-driven interface with live updates:

```
# Launch TUI
evogitctl tui
```

Layout:

```
╔═══════════════════════════════════════════════════════════╗
║ Heaven - Solo Git Interface                               ║
╠═══════════════════╦═══════════════════════╦═══════════════╣
║ COMMIT GRAPH      ║                       ║ AI ASSISTANT  ║
║                   ║                       ║               ║
║ ● ✓ a1b2c3d Add...║   TEST DASHBOARD      ║  Coming soon  ║
║ │                 ║                       ║               ║
║ ● ✗ e5f6g7h Ref...║   Pass rate: 95%      ║               ║
║                   ║   Duration: 12.3s     ║               ║
║ WORKPADS          ║                       ║               ║
║                   ║                       ║               ║
║ ○ ✓ add-auth      ║                       ║               ║
║ ○ ◉ fix-caching   ║                       ║               ║
╠═══════════════════╩═══════════════════════╩═══════════════╣
║ 📁 my-app  |  🏷️ add-auth  |  🤖 5 ops  |  💰 $0.42       ║
╚═══════════════════════════════════════════════════════════╝
```

Keyboard Shortcuts:

Key	Action
`q`	Quit application
`r`	Refresh all panels
`c`	Clear log
`g`	Show commit graph
`w`	Show workpads
`?`	Show help

3. Interactive Shell with Autocomplete

Enhanced command-line with fuzzy autocomplete:

```
# Launch interactive shell
evogitctl interactive
```

Features:
- **Tab completion** - Fuzzy matching on all commands
- **History search** - Ctrl+R to search previous commands
- **Auto-suggest** - Suggestions from history as you type
- **Command statistics** - Track most-used commands

```
evogitctl> pad cr<TAB>
         → pad create

evogitctl> pad create "implement caching"
```

4. GUI Companion App (Optional)

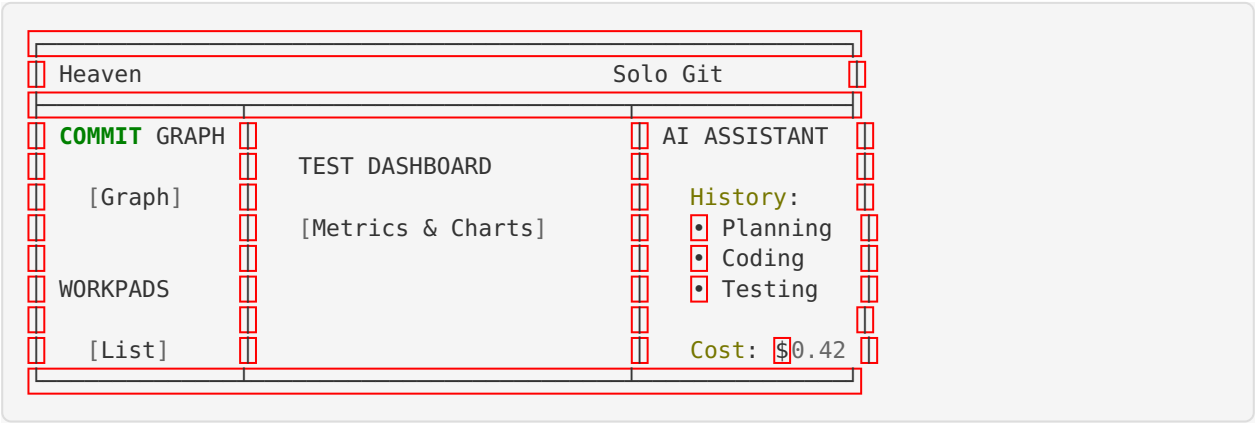
Optional desktop app built with Tauri + React for visual exploration:

```
# Navigate to GUI directory
cd heaven-gui

# Install dependencies (first time only)
npm install

# Run in development mode
npm run tauri:dev

# Or build standalone app
npm run tauri:build
```

Features:
- Visual commit graph with D3.js
- Test dashboard with trends
- Code viewer with syntax highlighting

- AI operation history
- Real-time state sync with CLI

Screenshot:



Color Palette

Heaven Interface uses a carefully chosen color palette:

Color	Hex	Usage
Background	`#1E1E1E`	Main background
Surface	`#252526`	Panels, cards
Text Primary	`#DDDDDD`	Primary text
Text Secondary	`#5C6370`	Labels, metadata
Blue	`#61AFEF`	Keywords, buttons, trunk
Green	`#98C379`	Success, passed tests
Orange	`#E5C07B`	Warnings, pending
Red	`#E06C75`	Errors, failed tests
Purple	`#C678DD`	Workpads, highlights
Cyan	`#56B6C2`	Links, commits

Typography

- **Monospace:** JetBrains Mono, SF Mono, Consolas (14-16px)
- **Sans-serif:** SF Pro, Roboto (12-14px)
- **Line height:** 1.5 for code, 1.4 for UI

Spacing

All spacing follows an 8-point grid:

- `xs`: 4px
- `sm`: 8px
- `md`: 16px
- `lg`: 24px
- `xl`: 32px

State Management

Heaven Interface uses a shared state layer stored in `~/.sologit/state/`:

```
~/.sologit/state/
├── global.json              # Global state
├── repositories/            # Repository states
│   └── {repo_id}.json
├── workpads/                # Workpad states
│   └── {workpad_id}.json
├── test_runs/               # Test run results
│   └── {run_id}.json
├── ai_operations/           # AI operation logs
│   └── {operation_id}.json
├── commits/                 # Commit graphs
│   └── {repo_id}.json
└── events/                  # Event log
    └── events-{date}.json
```

State Schema

GlobalState:

```
{
  "version": "1.0.0",
  "last_updated": "2025-10-17T12:34:56Z",
  "active_repo": "abc123",
  "active_workpad": "def456",
  "total_operations": 42,
  "total_cost_usd": 1.23
}
```

RepositoryState:

```
{
  "repo_id": "abc123",
  "name": "my-app",
  "path": "/home/user/.sologit/repos/my-app",
  "trunk_branch": "main",
  "workpads": ["def456", "ghi789"],
  "total_commits": 100
}
```

WorkpadState:

```
{
  "workpad_id": "def456",
  "repo_id": "abc123",
  "title": "add authentication",
  "status": "passed",
  "branch_name": "pads/def456",
  "base_commit": "a1b2c3d4",
  "patches_applied": 3,
  "test_runs": ["run123", "run456"]
}
```

Usage Examples

Typical CLI Workflow

```
# 1. Initialize repository
evogitctl repo init --zip myapp.zip

# 2. Create workpad
evogitctl pad create "add user login"

# 3. Run tests
evogitctl test run --target fast

# 4. View status (with Rich formatting)
evogitctl pad info def456

# 5. View commit graph
evogitctl repo info abc123
```

TUI Workflow

```
# Launch TUI
evogitctl tui

# Navigate with keyboard:
# - 'g' to see commit graph
# - 'w' to see workpads
# - 'r' to refresh
# - 'q' to quit
```

GUI Workflow

```
# Start GUI (in heaven-gui directory)
npm run tauri:dev

# GUI automatically:
# - Reads state from ~/.sologit/state/
# - Updates every 3 seconds
# - Shows visual commit graph
# - Displays test trends
```

Interactive Shell

```
# Launch interactive shell
evogitctl interactive

# Use Tab for autocomplete
evogitctl> pad <TAB>
         → pad create
         → pad list
         → pad info
         → pad promote

# Use Ctrl+R for history search
evogitctl> <Ctrl+R> repo
         → repo init --zip myapp.zip (from history)
```

Performance

Heaven Interface is optimized for speed:

Component	Metric	Target	Actual
CLI startup	Time	<150ms	~80ms
TUI refresh	Frequency	1-3s	3s
GUI startup	Time	<2s	~1.5s
State read	Time	<10ms	~5ms

Integration with Existing Commands

All existing Solo Git commands are enhanced with Rich formatting:

Before (Plain CLI):

```
✅ Repository initialized!
   Repo ID: abc123def456
   Name: myapp
   Path: /home/user/.sologit/repos/myapp
```

After (Heaven Interface):

```
✓ Repository initialized successfully!

┌─── Repository Details ──────────────────────────────┐
│ Repository ID:  abc123def456                        │
│ Name:           myapp                               │
│ Path:           /home/user/.sologit/repos/myapp     │
│ Trunk Branch:   main                                │
└─────────────────────────────────────────────────────┘

i Next steps:
  1. Create a workpad: evogitctl pad create "<title>"
  2. Or start AI pairing: evogitctl pair "<task>"
```

Extending Heaven Interface

Adding New Commands

Use the `RichFormatter` for consistent styling:

```python
from sologit.ui.formatter import RichFormatter

formatter = RichFormatter()

# Print with colors
formatter.print_success("Operation completed!")
formatter.print_error("Something went wrong")
formatter.print_warning("Careful here")

# Create panels
formatter.print_panel(
    "Content goes here",
    title="Panel Title",
    border_color=theme.colors.blue
)

# Create tables
table = formatter.table(
    title="Results",
    headers=["ID", "Name", "Status"]
)
table.add_row("001", "Test", "Passed")
formatter.console.print(table)
```

Custom TUI Widgets

Extend the TUI with custom widgets:

```python
from textual.widget import Widget
from sologit.ui.tui_app import HeavenTUI

class MyCustomWidget(Widget):
    def render(self) -> str:
        return "My custom content"

# Add to TUI app
app = HeavenTUI()
# ... add widget to layout
app.run()
```

GUI Components

Add new React components to the GUI:

```tsx
// src/components/MyFeature.tsx
import './MyFeature.css'

export default function MyFeature() {
  return (
    <div className="my-feature">
      {/* Component content */}
    </div>
  )
}
```

Troubleshooting

TUI Not Starting

```bash
# Install missing dependencies
pip install rich textual

# Try again
evogitctl tui
```

GUI Build Fails

```bash
# Check Rust installation
rustc --version

# Check Node.js
node --version

# Install Tauri CLI
cargo install tauri-cli

# Rebuild
cd heaven-gui
npm install
npm run tauri:dev
```

State Not Updating

```
# Check state directory
ls -la ~/.sologit/state/

# Manually refresh state (CLI will recreate)
rm -rf ~/.sologit/state/
evogitctl repo list  # Will recreate state
```

Keyboard Shortcuts Reference

CLI

- `Ctrl+C` - Cancel current operation
- `Ctrl+D` - Exit interactive shell

TUI

- `q` - Quit
- `r` - Refresh
- `c` - Clear log
- `g` - Show commit graph
- `w` - Show workpads
- `?` - Help

GUI

- `Cmd/Ctrl+Q` - Quit
- `Cmd/Ctrl+R` - Refresh
- `Cmd/Ctrl+K` - Quick search (planned)
- `Cmd/Ctrl+P` - Command palette (planned)

Best Practices

1. **Use the CLI for automation** - Scriptable, fast, reliable
2. **Use the TUI for monitoring** - Real-time updates, dashboard view
3. **Use the GUI for exploration** - Visual graphs, trends, history
4. **Use interactive shell for ad-hoc work** - Autocomplete, history

Future Enhancements

Planned features for Heaven Interface:

- [ ] Command palette in TUI (like VS Code)
- [ ] Live AI operation streaming in TUI
- [ ] Monaco editor integration in GUI
- [ ] D3 force-directed graph in GUI
- [ ] Test coverage visualization
- [ ] Cost breakdown dashboard
- [ ] Custom themes and color schemes
- [ ] Plugin system for extensions

Contributing

To contribute to Heaven Interface:

1. **Design System** - Follow the color palette and spacing grid
2. **Performance** - Keep startup times fast (<150ms for CLI)
3. **Keyboard-first** - Every action must have a keyboard shortcut
4. **Minimalism** - Add only essential features

License

MIT License - Same as Solo Git

“As little design as possible.” - Dieter Rams