Chronos OS: Phase 22 Report

Multi-Tasking GUI and Window Decoration

System Architect

January 21, 2026

Abstract

Phase 22 transformed the graphical interface from a single-context display into a multi-window desktop environment. We expanded the Shell architecture to manage a vector of window instances, implemented hit-testing logic to handle mouse focus events, and introduced "Window Decorations" (Title Bars and Borders) to visually distinguish overlapping applications.

Contents

1 Window Decorations (The Chrome)

To distinguish between multiple windows on a uniform background, we upgraded the `Window` struct to support "Chrome"—the non-client area.

- **Title Bar:** A 20px high strip at the top (Navy Blue), acting as a handle for dragging.
- **Borders:** A 2px Light Grey frame surrounding the content.
- **Client Area:** The inner region where text and applications are rendered.

This required modifying the coordinate system of the text renderer to apply a `(border_width, title_height)` offset to all character drawing operations.

2 Multi-Window Architecture

2.1 Shell Refactoring

The Shell originally owned a single `window` object. We refactored this into:

```
pub struct Shell {
    pub windows: Vec<compositor::Window>,
    pub active_idx: usize, // The window receiving Keyboard Input
}
```

Listing 1: Multi-Window State

The `term` command was implemented to instantiate new `Window` objects and push them onto this vector, dynamically expanding the desktop workspace.

3 Input Routing and Z-Ordering

3.1 Focus Logic

Handling mouse clicks in a multi-window environment requires determining which window is "on top." We implemented a reverse-iterator search:

1. Iterate through the window list from back (topmost) to front (bottommost).
2. Perform hit-testing (`win.contains(mouse_x, mouse_y)`).
3. The first match becomes the `active_idx`.

3.2 The Render Pass

To correctly visualize the stacking order (Z-Order), the main loop constructs a draw list every frame:

1. **Layer 0:** The Taskbar (Always bottom).
2. **Layer 1..N:** The windows from the Shell's vector.

By rendering the `windows` vector in order, newer windows draw over older ones. (Future optimization: The active window should be moved to the end of the vector to ensure it is drawn last/on top).

4 Conclusion

Chronos OS now functions as a true Window Manager. Users can spawn multiple terminals, organize them spatially via dragging, and switch context by clicking. This completes the core requirements for a Graphical Desktop Environment.