Chronos OS: Phases 20 & 21 Report

Window Interaction, Collision Detection, and CMOS RTC

System Architect

January 21, 2026

Abstract

With the graphical compositor established, the next stage of development focused on transforming the passive display into an interactive environment. Phase 20 introduced the logic required to manipulate windows via mouse input, including hit-testing and stateful dragging. Phase 21 integrated the motherboard's Real-Time Clock (RTC) to provide persistent timekeeping within the desktop environment.

Contents

1 Phase 20: Interactive Windows

1.1 Input Subsystem Expansion

The initial mouse driver tracked only X/Y movement. To support interaction, we extended the driver to decode the **Button State**.

Protocol: The first byte of the PS/2 packet contains the button flags.

- **Bit 0:** Left Button (1 = Pressed, 0 = Released).
- **Bit 1:** Right Button.

We exposed this state via the `mouse::get_state()` function, allowing the main kernel loop to query for clicks.

1.2 Collision Detection (Hit Testing)

To determine which window the user is interacting with, we implemented a bounding-box check within the `Window` struct.

```
pub fn contains(&self, px: usize, py: usize) -> bool {
    px >= self.x && px < self.x + self.width &&
    py >= self.y && py < self.y + self.height
}
```

Listing 1: Hit Test Logic

1.3 The Dragging State Machine

A naive implementation of dragging sets the window's position directly to the mouse coordinates ($Window_x = Mouse_x$). This causes the window's top-left corner to "snap" to the cursor, creating a jarring user experience.

The Solution: We implemented an offset-based drag logic.

1. **On Click (Start):** Calculate and store the delta between the mouse and the window origin.
$$\Delta x = Mouse_x - Window_x$$
$$\Delta y = Mouse_y - Window_y$$
2. **On Drag (Hold):** Update the window position by subtracting the stored offset.
$$Window_{new\_x} = Mouse_{current\_x} - \Delta x$$

This ensures the window moves relative to the point where it was grabbed.

2 Phase 21: Real-Time Clock (RTC)

2.1 CMOS Architecture

To display the current time, Chronos interfaces with the legacy CMOS chip. Communication is handled via two ports:

- **0x70 (Address Port):** Selects the register index.
- **0x71 (Data Port):** Reads/Writes data.

2.2 Binary Coded Decimal (BCD) Decoding

The RTC stores time values in BCD format to save space. For example, 35 seconds is stored as hex `0x35` (binary `0011 0101`). To perform calculations or formatting, this must be converted to standard binary integers.

$$Binary = (Value \ \& \ 0x0F) + ((Value \ / \ 16) * 10) \tag{1}$$

2.3 GUI Integration

We updated the `time.rs` driver to poll the RTC registers (Seconds, Minutes, Hours) while checking the "Update in Progress" bit (Register A) to avoid reading mid-tick.

The formatted time string is passed to the Taskbar window logic in the main loop, providing a live digital clock in the bottom-right corner of the desktop environment.

3 Conclusion

Chronos OS has evolved from a static window manager to a dynamic desktop environment. Users can now intuitively organize their workspace via drag-and-drop mechanics, and the system provides persistent temporal context via the hardware clock.