map.svelte + navigation_plan.svelte + interest_locations.svelte — Map & Navigation Components

map.svelte

The core map display component. Loads a map file from `<appDataDir>/maps/`, renders it in a letterboxed `<img>` element, and overlays interactive pins on top. Accepts a `mode` prop that controls what happens when the operator clicks the map.

Mode	Click behaviour	Pins shown
`navigation`	Adds a `PinnedCoord` to the `pinnedCoords` store	Unassigned pins + start/waypoint/end markers
`science`	Adds an `InterestLocation` to `scienceLocations`	Science location pins
`probing`	Adds an `InterestLocation` to `probingLocations`	Probing location pins

Map loading — On mount, if `displayedMap` store already holds a filename it is opened immediately; otherwise the component calls `list_task_files("maps")` and shows a selection modal. If only one map file exists it is auto-selected. The reload button (⟳) clears all state and returns to the selection modal.

3D formats (`.obj`, `.las`, `.laz`, `.e57`) trigger a `render_map` invoke before display, showing a spinner while the backend works. Plain image formats are loaded directly via `convertFileSrc`. After rendering, the `_preview.png` path is used for all subsequent display.

Coordinate geometry — `getRenderedRect()` computes the actual rendered rectangle of the image inside its element (accounting for letterboxing / `object-fit` behaviour). `eventToImgPixel()` converts a raw `MouseEvent` into a pixel coordinate within the PNG, with Y flipped so the origin is bottom-left. `worldToCSSPos()` is the inverse — takes a world-space `(x, y)` in metres and returns a `left/top` percentage suitable for absolute positioning an overlay element on top of the image. Both functions short-circuit to `null` when `mapMeta` is unavailable.

Mouse interaction — `onMouseMove` calls `eventToImgPixel` and then invokes `pixel_to_world` to display a live coordinate overlay in the bottom-left corner (pixel position + world metres + "Click to pin" hint). The overlay disappears when the cursor leaves the image.

GPS marker — Listens for `gps-update` Tauri events on mount. The payload's `longitude`/`latitude` fields are reused directly as map-space X/Y metres. When a valid GPS position is in the store, a directional arrow marker (▶) is rendered at the corresponding map position, rotated by `heading` via a CSS custom property `--heading`.

Pinned coordinate list — In `navigation` mode, a sidebar panel lists all `pinnedCoords` with copy-to-clipboard (□□) and remove (×) buttons. Hovering a row highlights the corresponding pin on the map, and vice versa, via `hoveredPinId`.

navigation_plan.svelte

A sidebar panel for building a navigation route from map-pinned coordinates. Displays a structured plan of start point → ordered waypoints → end point, and surfaces any `pinnedCoords` that haven't yet been assigned a role.

Route structure — The plan always shows three sections in order: a Start card, a draggable Waypoints list, and an End card. Unset slots show "Not set". Hovering any card cross-highlights its corresponding pin on the map via the `hoveredNavId` store (shared with `map.svelte`).

Waypoint reordering — The waypoint list uses `svelte-dnd-action` (`dndzone`) for drag-and-drop reordering. Both `consider` and `finalize` events write the new order directly to the `waypoints` store.

Promoting pinned coords — Each `PinnedCoord` from the map appears in a "Pinned from map" section at the bottom of the list. Three buttons let the operator promote a pin to Start, add it as a new Waypoint, or set it as End. In all cases the pin is removed from `pinnedCoords` after promotion.

Destructive actions — Removing a waypoint and clearing the entire plan both trigger a native `confirm()` dialog before proceeding.

Map import — The "+ Add Map File" button opens a file picker (via `@tauri-apps/plugin-dialog`) filtered to `.json`, `.geojson`, `.txt`, `.jpeg`, `.obj`, `.las`, `.laz`, `.e57`, then calls `invoke("import_map_file")` to copy the chosen file into the app's maps directory. The "▶ Plan Route" button is present but not yet wired to a backend command.

interest_locations.svelte

A generic, reusable sidebar list for named locations of interest. Used by both the Science and Probing task panels, which pass in their respective stores (`scienceLocations` / `probingLocations` and the matching hovered-id store) as props.

Props: `locations` — a `Writable<InterestLocation[]>` store; `hoveredId` — a `Writable<string | null>` store shared with `map.svelte` for cross-highlighting.

Each location row shows its auto-generated name and its `(x, y)` coordinates in metres. Hovering a row sets `hoveredId`, which causes the corresponding pin on the map to highlight. Two actions are available per row:

- **Rename (✏)** — Switches the name label to an inline `<input>`. The edit is committed on blur or Enter; if the input is left empty the original name is kept. Only one location can be in edit mode at a time (`editingId` state).
- **Remove (×)** — Removes the location from the store immediately with no confirmation.

When the list is empty a hint instructs the operator to click the map to add a location.

Revision #1
Created 2026-05-06 09:18:17 UTC by Candela Cimadevilla Gonzalez
Updated 2026-05-06 09:22:23 UTC by Candela Cimadevilla Gonzalez