map_commands.rs and map_processor.rs — 3D Map Rendering & Coordinate Transforms

`map_processor.rs` + `map_commands.rs` — 3D Map Rendering & Coordinate Transforms

Converts 3D map files into top-down orthographic PNG images coloured by height (Z), and exposes coordinate transforms so the frontend can convert pixel clicks back to real-world metres. The pipeline runs on a blocking thread to avoid stalling the async runtime.

Supported formats

Extension	Library	Notes
`.obj`	`tobj`	OBJ Y-up convention remapped: world X = OBJ X, world Y = OBJ Z, height = OBJ Y
`.las`	`las`	X/Y/Z read directly from point records
`.laz`	`las`	Same as LAS (compressed)

Commands

`render_map(filename)` → `MapMeta` Renders a 3D map file to a top-down PNG. The source file must already exist in `<appDataDir>/maps/`. The output is written to the same directory as `<stem>_preview.png`. The image is sized to a 2048 px longest edge, aspect ratio preserved. Heavy work is offloaded via `spawn_blocking` so the async runtime is never blocked. Returns a `MapMeta` struct the frontend uses for pixel→world transforms.

`pixel_to_world(px, py, meta)` → `(f64, f64)` Converts a 2D pixel coordinate (origin bottom-left, X right, Y up) to real-world metres. Expects coordinates in the displayed image's frame — rotation, if any, must be accounted for by the frontend before calling this. The formula is simply `world = pixel × metres_per_pixel`, with the world origin anchored to the bottom-left corner of the image.

`MapMeta` fields

`MapMeta` is serialised and returned to the frontend after every `render_map` call.

Field	Type	Description
`img_width`	`u32`	PNG width in pixels (post-rotation)
`img_height`	`u32`	PNG height in pixels (post-rotation)
`world_x_min`	`f64`	Real-world X at the left edge (currently always `0.0`)
`world_y_min`	`f64`	Real-world Y at the bottom edge (currently always `0.0`)
`metres_per_pixel`	`f64`	Scale factor for pixel→world conversion
`format`	`String`	Source format detected (`"obj"`, `"las"`, `"laz"`)
`rotated`	`bool`	`true` if the image was rotated 90° to landscape

Rendering pipeline

`process_map()` runs the full pipeline in five stages:

1. **Load** — Parse the source file into a flat list of `Point3D` structs (`x`, `y`, `z` in world space).
2. **Bounding box** — Compute `x_min/max`, `y_min/max`, `z_min/max` over all points. World width and height must be non-zero or an error is returned.
3. **Rasterise** — Map each point to a pixel coordinate. Where multiple points land on the same pixel, keep the highest Z (i.e. the sky-facing surface wins). Pixel dimensions are derived from `img_size` (2048) with the aspect ratio preserved.
4. **Gap fill** — Run a two-pass nearest-neighbour distance transform (`fill_gaps`) to fill pixels that received no points. The forward pass sweeps top-left → bottom-right (checking left and top neighbours); the backward pass sweeps bottom-right → top-left (checking right and bottom neighbours). Each unfilled pixel inherits the Z of its closest filled neighbour, eliminating stripe artifacts.
5. **Colour + save** — Each pixel's Z is normalised to `[0.0, 1.0]` and passed through a five-stop colour ramp (`height_color`): deep blue → cyan → green → yellow → red. If all points share the same Z (flat terrain), mid-green is used. The image is rotated 90° if height exceeds width (to keep the longest edge horizontal), then saved as PNG.

Height colour ramp

Normalised Z	Colour
0.0	Deep blue
0.25	Cyan
0.5	Green
0.75	Yellow
1.0	Red

Error conditions

Condition	Error returned
Unsupported file extension	`"Unsupported format: .{ext}"`
File parsed but empty	`"File parsed but contained no points."`
All points collinear in X or Y	`"All points are collinear — cannot build a 2D map."`
Source file missing at invoke time	`"File not found: {path}"`
PNG write failure	`"Failed to save PNG: {e}"`

Revision #3
Created 2026-05-05 10:27:34 UTC by Candela Cimadevilla Gonzalez
Updated 2026-05-05 10:33:49 UTC by Candela Cimadevilla Gonzalez