Column Map

The authoritative specification of how the world is stored and edited. It replaces the voxel storage model (MAP_RULES §3-10's storage sections are superseded by this document). The *rules* that act on this model — materials, movement thresholds, determinism, rendering discipline — still live in MAP_RULES.

1. Purpose & scope

The world is a **rectangular grid of columns**. A column is a vertical stack of **layers**; each layer is a 1×1 slab of one material defined by four bottom-corner heights and four top-corner heights. This is FFT's model — one surface per tile, terrain type per tile — generalized to stacked surfaces (floors above floors, water as a typed surface layer, floating platforms) and optional slopes (non-flat corner heights).

Why columns over voxels: the game is surface-centric (units stand on surfaces), authoring matches how a designer thinks (paint a tile, stamp a building), arbitrary geometry needs no cell quantization, and it reproduces FFT maps exactly. Voxels' only edge — true 3D volume — is not needed for tactics maps and adds authoring friction.

2. The model

2.1 Columns

- The map has a size `(width, height)` in columns, indexed `(x, z)`. Maps may be any size.
- A column contains an ordered list of **layers**, bottom→top. An empty column is open air.
- `(planned)` — chunking/streaming for very large maps; not needed for authored battlefields.

2.2 Layers

A layer is a vertical slab occupying the column's 1×1 footprint:

field	meaning
`type`	material index into the catalog (append-only, unchanged from MAP_RULES)
`bottom`	4 corner heights of the layer's underside
`top`	4 corner heights of the layer's upperside

field	meaning
`integrity`	current hit points (only meaningful when the material is destructible)

Corner order (fixed): `[NW, NE, SE, SW]` = `(x+0, z+0), (x+1, z+0), (x+1, z+1), (x+0, z+1)`.

- A flat-topped layer has all four top corners equal (e.g. ground top at 1.0). A slope has differing top corners (e.g. `[0, 1, 1, 0]`).
- A layer's thickness at each corner is `top[i] - bottom[i]`; it must be ≥ 0 (zero-thickness layers are rejected).
- **Snapping**: authored heights snap to the `0.25 / 0.5 / 1.0` grid by default. Non-snapped floats are allowed (slopes); snapping is an authoring convention, not a storage constraint. Heights are stored as exact floats either way.

2.3 Invariants (validated, never repaired)

For every column:

1. Layers are ordered bottom→top: `layer[i].bottom ≥ layer[i-1].top` at every corner (equal = sitting directly on top).
2. Every layer has non-negative thickness at all four corners.
3. Corner heights stay within the map's vertical bounds.
4. `type` is a valid catalog index; `integrity` is in `1..max` for destructible materials, ignored otherwise.
5. Columns with zero layers are valid (pure air/void).

Validation **rejects** violations with an error; it never silently repairs.

3. Surfaces

3.1 The walkable surface

A column's **surface** is the top surface of its **topmost layer** (its four top corners). A column with no layers has no surface.

- Terrain = a layer whose top is the ground you stand on.
- Water = a layer of type `water`; its top surface is the water's surface (wading per rules).
- A roof you stand on = a layer; a room beneath = the gap between its bottom and the layer below.
- Air is never stored — it is the absence of layers between surfaces.

3.2 Surface for pathfinding

The pathfinder consumes surface heights exactly as it does today (MAP_RULES §8, COMBAT pathfinding pipeline):

- Surface height of a column = the topmost layer's top corners.
- When stepping between adjacent columns, the height differential uses the **shared-edge corners**: stepping east, compare the east edge (`NE, SE` on the source, `NW, SW` on the target); stepping south, compare `SE, SW` vs `NE, NW`. Rise = `max(0, mean(target_edge) - mean(source_edge))`.
- The rise feeds `MovementRules` unchanged (free_step 0.25, step_ceiling 1.0, climb cost) — so two adjacent columns need not meet exactly; walkability is decided by the rules, exactly the "do these blocks align?" question answered by rules rather than storage.

4. The layer algebra (edits)

All edits are **layer operations** applied to a rectangular region of columns, atomically, as one commit (COMBAT §10). Nothing mutates the map out-of-band.

op	effect
`insert_layer`	Add a layer to columns at its height; reorder to maintain ordering. Rejected if it would violate invariants.
`remove_layer`	Remove a layer (by surface index) from columns; then run collapse (§5.2).
`resize_layer`	Change a layer's bottom/top corners; re-validate against neighbors.
`retype_layer`	Change a layer's material.
`split_layer`	Split a layer into two at a horizontal plane (below/above) — used for craters and collapse.

Commit kinds (COMBAT §10.1): `set_layer`, `remove_layer`, `resize_layer`, `retype_layer` (each carrying a region + layer payload). The remaining kinds (`move_unit`, `use_action`, `apply_damage`, `apply_status`, `field_event`, `end_match`, `wait`) are unchanged.

5. Dynamic world (the living map)

5.1 Destruction

A material may be `destructible` with an `integrity_max` (TerrainRule, COMBAT §9.1). A destructible layer carries current `integrity`; damage reduces it (element weakness doubles damage). At zero, the layer is removed and collapse runs.

Destruction is **layer-level**: you destroy a surface (a bridge, a wall layer, a roof), not an arbitrary chunk of a cell. Craters in the middle of a wall = `split_layer` above and below the impact, then

remove the middle.

5.2 Collapse (gravity)

When a layer is removed (or the layer below it shrinks/removes), layers above **fall**: they translate down until their bottom rests on the new support's top, or the floor. Deterministic resolution: affected columns processed bottom-up, columns in fixed iteration order. A falling layer that would exit the map's vertical bounds is destroyed. Falling layers displace and damage units (COMBAT §9.4).

5.3 Field events

Scheduled events keyed to the tick counter operate on layers (COMBAT §9.3): lava rising = raise a lava layer's top or insert one; water freezing = `retype_layer` water→ice; fire spreading = `retype_layer` topmost layers per the flammability rule. All are commits.

6. Serialization

Byte-exact, schema-versioned, reject-on-mismatch (same discipline as the voxel codec it replaces):

```
header:  magic, schema_version (u32), catalog_version (u32), seed (u32),
         width (u32), height (u32), vertical_bounds (u32)
spawns:  count (u16), then per spawn { x (u16), z (u16), surface (i16, -1 = topmost) }
columns: per non-empty column { x (u16), z (u16), layer_count (u8) },
         then per layer { type (u8), integrity (u8), 8x f32 corners }
```

Empty columns are omitted; the file is the set of non-empty columns plus header. Packing is deterministic (fixed column iteration order). Load rejects bad versions, invalid types, or invariant violations.

7. Authoring

- **Snap grid** 0.25 by default (0.5 / 1.0 quick-snap options); floats for slopes.
- **Shape templates** — the voxel shape vocabulary survives as corner presets the editor stamps:

shape	bottom	top
FULL	`[0,0,0,0]`	`[1,1,1,1]`
HALF	`[0,0,0,0]`	`[0.5,0.5,0.5,0.5]`
SLAB	`[0.8,0.8,0.8,0.8]`	`[1,1,1,1]`

shape	bottom	top
SLOPE (rot 0)	`[0,0,0,0]`	`[0,1,1,0]` (rises along +x; rotations permute corners)
WEDGE (rot 0)	`[0,0,0,0]`	`[0,1,1,0]` with the `SW` corner collapsed onto `SE` (triangle)

- **Buildings** are layer-template sets stamped onto a footprint (floor layers, wall layers, roof layer with a defined bottom+top). Placing a 0.2 slab at any height = stamping one SLAB-shaped layer.
- **Undo/redo** is free: the commit log is the command history, and snapshots are compact.

8. Status

`[spec]` — this document is the contract. Implementation (codec, surface derivation, renderer, generators, commit kinds, tests) is `[planned]` and tracked in CONTEXT.md. Until it lands, the voxel storage remains the running implementation and is treated as legacy.

Revision log: created as the authoritative storage/geometry spec, superseding MAP_RULES §3-10's storage model. Decided with the voxel-vs-column evaluation: one format, layered columns, snapped-by-default heights, layer-level destruction.

Revision #1
Created 2026-08-17 18:14:48 UTC by Fifthdread
Updated 2026-08-17 18:14:48 UTC by Fifthdread