Part I — The Game

This is the high-level reference for **Operation Tactics** (codename `operation_tactics`, real name TBD) — a Final Fantasy Tactics–style tactical RPG built in Godot.

Think of this as the game's bible: it describes the whole game and how its systems and pipelines work, in plain language. It is the starting point for anyone (human or agent) working on the project.

How this relates to the other docs:

Doc	Purpose
This bible	The big picture: vision, every system, every pipeline, the baselines. Read this first.
`MAP_RULES.md`	The authoritative, precise spec of the world's data model and rules. This bible summarizes it; MAP_RULES wins on any conflict.
`CONTEXT.md`	The operational handbook: current state, engineering standards, architecture quick-reference, shipping. Agents read it before every task.
`AGENTS.md`	Identity card: one-paragraph purpose and the hard rules.

The bible is mirrored to the wiki at `wiki.fifthdread.com/books/operation-tactics` by `tools/sync_wiki.py`. The repo version is the source of truth.

Vision & pillars

Operation Tactics is a **grid-based, turn-based tactical RPG**: you command a party of units on a battlefield that has real height, and every move and action is a decision.

The design pillars:

1. **Decisions, not reflexes.** Every turn is a puzzle — where to stand, which way to face, who to hit. Terrain, height, and hazard all matter.
2. **The map is honest.** What you see is what the data says. If a tile is lava, the rules say it is lava; nothing hides it. Players can trust the battlefield.
3. **Grow through doing.** Units learn and improve through what they actually do in combat (classic job-point style), not just through leveling up.
4. **One world, one set of rules.** The player, the enemies, and the AI all play by the same rules. The map stores facts; everyone's rules derive from the same facts.
5. **Deterministic by construction.** The simulation is a pure function of the shared facts and the rules. A match cannot de-sync by design — which is what makes multiplayer a

natural consequence rather than an add-on.

Core loop

A battle plays out in a rhythm:

1. **Read the field.** The camera sits above the battlefield; you see elevation, hazards, and where everyone stands.
2. **Pick a unit.** Each unit gets a turn in a rotating order.
3. **Move.** The game shows a **movement bubble** — every tile the unit can reach within its move budget, considering terrain cost and height. You pick a tile.
4. **Act.** Attack, cast, use an item, or wait. Range and line-of-sight are computed from where the unit ended up.
5. **End turn.** Effects tick (hazards, statuses, day/night), the next unit goes, and the field is read again.

Beyond battles: a story-driven campaign, branching characters, and between-battle progression (shops, loadouts, jobs). Those systems are planned; the battle mechanics below are the growing core.

Player experience

- **Camera:** an isometric-style tactical camera. Hold to pan, wheel to zoom, rotate 45° at a time. F1 toggles between two views of the same battlefield.
- **Two views of the same truth: DEBUG** shows the map data literally — every block at its exact size, shape, and position, flat colors. **RENDER** shows the game look — the **exact same geometry** (slopes stay slopes, holes stay holes) with seeded per-cell tint, rocks and foliage, lighting, day/night. Both are derived from the same data; RENDER only adds cosmetics, never geometry.
- **Feedback without lies:** the top grid, hover highlighting, and selection all read from the data, so what you point at is exactly what exists.

Revision #10
Created 2026-08-15 18:17:00 UTC by Fifthdread
Updated 2026-08-15 18:59:59 UTC by Fifthdread