Combat

Combat

This document defines the battle system of **Operation Tactics**: turn order, movement, actions, damage, elements, statuses, and a battlefield that changes as the fight goes on. It is the authoritative spec — `GAME_BIBLE.md` summarizes it, and nothing here is superseded except by a newer version of this file.

What is assumed (FFT baseline). Turn order, the per-turn flow, option/tile selection, pathfinding, and units standing on tiles all behave like Final Fantasy Tactics. We state those behaviors here so no one (human or agent) re-invents or "fixes" them.

What is ours. The map is a *living* battlefield (mutation during combat), an element/status system, deterministic CT scheduling, and a fully deterministic simulation pipeline. These have no FFT equivalent and are specified precisely.

---

1. Assumed baseline (FFT-shaped, stated once)

1. One unit per tile; a unit stands on its tile's surface (see MAP_RULES §7).
2. Movement is bounded by a **move budget** and shown as a **movement bubble** (the reachability set — see the pathfinding pipeline).
3. A unit's turn is: choose a destination (or hold) → face a direction → choose an action → pick a target → confirm.
4. The action menu is **Attack / Ability / Item / Wait**. "Wait" ends the turn without acting.
5. Occupied tiles are impassable — you cannot move through or stand on a tile another unit stands on.
6. Targeting considers both horizontal range and vertical height difference.
7. Anything not specified below is expected to follow FFT behavior unless it contradicts these rules.

2. Our divergences (defined here, no FFT equivalent)

Feature	Rule
Turn order	Deterministic CT scheduling (§3), not a fixed round order

Feature	Rule
The map	Mutable mid-combat; every change is a commit (§9, §10)
Elements	Eight-element system with per-unit affinities and terrain elements (§7)
Statuses	A status catalog with tick-based durations (§8)
Simulation	Fully deterministic: integer math, seeded RNG, commit-ordered (§10)

3. Turn order & the tick clock

The match has one global clock: an integer **tick counter** (`tick`, starting at 0, +1 per simulation tick). Everything scheduled in a battle — turns and world events alike — is keyed to this counter.

- Every tick, every unit's **Charge Time (CT)** increases by its **SPD**: `ct += spd`.
- A unit is **eligible** when `ct >= 100`.
- The **actor** is the eligible unit with the highest CT. Ties are broken by ascending `unit_id` (stable ids assigned at match setup) — deterministic, never random.
- After a unit acts, `ct -= 100`; the **overflow stays** (a unit at CT 137 acts and sits at 37). This is why fast units act more often.
- **Movement does not consume CT.** CT only schedules *when* a unit's turn comes up; the turn itself is free.
- Tick cadence is a presentation concern — the *logic* only cares that ticks are ordered.

`[planned]` — `src/combat/battle.gd` owns the tick counter, CT queue, and actor selection.

4. A unit's turn

When a unit is the actor:

1. **Move** — pick a destination tile inside the movement bubble (or hold position). The path is computed by the pathfinder; entering hazardous tiles applies their effects as the unit walks (§5, §9).
2. **Face** — pick one of the four directions. Facing determines which targets are in an action's arc and enables the back-attack bonus.
3. **Act** — choose an action from the menu (Attack / Ability / Item / Wait).
4. **Target** — if the action targets something, pick a tile or unit that satisfies range + line-of-sight + height reach (§6).
5. **Confirm** — the turn is committed as one ordered set of decisions and resolved (§10).

Waiting is "Act = Wait": the unit ends its turn immediately.

5. Movement & occupancy

- **Bubble:** the set of tiles reachable within the unit's Move budget, computed by the pathfinder's reachability pass. Terrain cost, climb, and unit mobility all apply (see the rules layer).
- **Blocking:** a tile occupied by another unit (friend or foe) is impassable for both movement *through* and *onto*. Ranged actions may still target the occupant.
- **Height:** a unit stands at its tile's surface height (voxel geometry; see MAP_RULES §8). Reach checks use the height difference between attacker and target.
- **Falling:** if the tile under a unit is destroyed or removed (§9), the unit **falls** to the highest remaining surface at or below its position. Fall damage is documented in the tunable combat module (baseline: `max(0, levels_fallen - 1)` × `fall_damage_per_level`). A unit with no surface below is removed from the battle.

`[planned]` — falling resolution; pathfinding and bubble logic are `[built]`.

6. Actions & targeting

An **action** is a data record with the following fields (defined once, referenced by the menu, AI, and damage math):

Field	Meaning
`kind`	attack / ability / item
`power`	base strength fed to the damage formula
`element`	one of the eight elements (or none)
`range`	max horizontal reach in tiles (manhattan distance)
`height_reach`	max absolute height difference (in levels)
`area`	target tile, radius-N blast, or line
`target_type`	enemy / ally / self / tile
`mp_cost`	MP spent (abilities)
`status`	status applied on hit (and its ticks)
`map_effect`	e.g. break block, create block, ignite (§9)

Reach: a target is in range when the manhattan tile distance ≤ `range` **and** the height difference ≤ `height_reach`. Area effects apply to every unit standing in the affected tiles.

Line of sight: single-target actions require an unblocked line between attacker and target. A line is unblocked if no solid block whose rule carries `BLOCKS_LOS` intersects it. Abilities may declare they ignore LOS. `[planned]` — voxel raycast.

Resolving an attack (§7) and **resolving a map effect** (§9) both happen during the action's apply step; neither writes anything except through commits.

7. Stats, damage & elements

7.1 Streamlined stats

Stat	Meaning
`hp` / `max_hp`	hit points
`mp` / `max_mp`	mana for abilities
`atk` / `def`	physical offense / defense
`mag` / `mdf`	magical offense / defense
`spd`	CT per tick (§3)
`move_budget`	movement points per turn (existing)
`jump_height`	max climbable rise (existing; jump ability later)

7.2 Damage baselines

Physical: `damage = max(1, (atk + action.power) - def) × modifiers` Magical: `damage = max(1, (mag + action.power) - mdf) × modifiers`

Modifiers (multiplicative, applied in order, each `[0, 2]`): element affinity (§7.5), height advantage (+`height_bonus_per_level` per level *above* the target), back attack (×`back_multiplier` when the target faces away), variance (`DeterministicRng` ±`variance_pct`). Final damage rounds to an integer.

Hit chance: `hit = accuracy - evasion` (from attacker action/stat minus defender stat), clamped to `[min_hit, max_hit]`; the roll uses `DeterministicRng.chance(hit / 100)`.

Every constant above lives in one tunable combat module (`src/combat/damage.gd`, planned) — the same discipline as `MovementRules`. `[planned]`.

7.3 Determinism

Combat randomness comes **only** from `DeterministicRng`, seeded from the match's simulation seed and fed by the commit sequence. The engine RNG is never used in a simulation path. Damage, hits, variance, and any future rolls are therefore identical on every client.

7.4 The eight elements

`fire, ice, lightning, water, wind, earth, holy, dark` (+ `none`).

There is **no fixed element-vs-element matrix.** An element matters through two mechanisms only:

1. **Affinities** — per unit, per element, a multiplier (below).
2. **Terrain element** — standing on an elemental tile grants resistance to that element (§7.6).

7.5 Affinities

Each unit has an affinity per element: a damage multiplier.

Multiplier	Meaning
`0`	immune
`< 0`	absorbs (heals for that much)
`0.5`	resists
`1`	normal
`2`	weak

An action's `element` is looked up against the target's affinities and the result multiplies damage (floor at 0 for immunity; negative values heal).

7.6 Terrain element & standing resistance

A `TerrainRule` may declare an element (e.g. lava is fire). While a unit stands on an elemental tile, its affinity for that element is treated as `min(affinity, 0.5)` — standing on your element protects you, and the same terrain element in the rules layer drives hazard damage (§9).

7.7 Element × status interactions

Condition	Effect
`wet` + fire element	+50% fire damage taken
`wet` + ice element	applies `freeze` (immobilize)
fire element on flammable terrain	ignites (field event, §9.4)
lightning on `wet`	+50% lightning damage taken

The table is small and explicit on purpose; additions require a documented rule first.

8. Statuses

A **status** is a named effect on a unit with a duration in **ticks** (the global clock), counted from application and decremented every tick. Durations are deterministic and never "until cured"

without a stated expiry.

Mobility statuses (built) — `slow`, `stop`, `immobilize`, `float`, `swim`, plus `wet`, `cold`, `hot` (see `UnitProfile`). `slow`/`stop`/`immobilize` already modify the movement budget; `float`/`swim` grant traits.

Combat statuses (planned) — `poison` (damage per tick), `blind` (accuracy penalty), `silence` (no abilities), `charm` (targets allies), `protect`/`shell` (halve physical/magical damage), `regen` (heal per tick), `freeze` (immobilize), `haste`.

Sources — terrain (hazard reports from the pathfinder), actions, field events. **Application** and **expiry** are commits (§10).

9. The living map

The battlefield is mutable, and **every mutation is a commit** (§10.2). Clients never mutate the map directly — they apply commits in order. This is what keeps a destructible battlefield in sync in multiplayer.

9.1 Destruction

A `TerrainRule` may be `destructible` with an `integrity` (hits to break). Damage against a block reduces integrity; an attack whose element matches the block's **weakness** (documented per material) deals double integrity damage. At zero, the block is removed (`remove_block` commit) and its support rules apply (§9.5). `[planned]` — `TerrainRule` fields.

9.2 Placement

Abilities may create blocks (walls, ice pillars, bridges). The voxel payload (position, type, shape, rotation) is a `set_block` commit. Placement follows the map's placement rules (MAP_RULES §6) — no overwriting solid space without a documented rule. `[planned]`.

9.3 Scheduled field events

The match can carry a schedule: `{tick, effect}` pairs. When the tick counter reaches one, it fires as a `field_event` commit. Examples: lava rising (raising a surface by adding blocks), fire spreading to flammable neighbors, water freezing to ice, a bridge collapsing at a plot beat. The schedule is part of the match setup and is deterministic. `[planned]`.

9.4 Collapse chains

When a block loses its support (the voxel below is removed or empty), gravity applies. Collapse is resolved as a deterministic flood from the removal point: supports are evaluated bottom-up, blocks that lose support fall, and a falling block either lands on a surface (becoming `set_block`) or continues down. Falling blocks and displaced units deal/apply documented impact effects. The

order is fixed by the resolution algorithm — never by UI or timing. `[planned]`.

9.5 Hazards & pathfinding

Hazards (hazard reports) already flow from the rules layer. Any map commit **invalidates the surface graph cache**, so the next movement bubble and the next hazard report reflect the new battlefield. `[built]` — hazard reporting; `[planned]` — commit-driven invalidation wiring.

10. The combat pipeline

Every battle step is a pure function of the commit log:

```
input (player/AI) ──► validate ──► Commit (seq, kind, payload)
   │                                   │
   └──────────── broadcast to all clients ◄┘
                                       │
        each client simulates, in order:
        move (path + hazard effects) → resolve action (range/LOS/affinity/status)
        → apply map mutations → tick statuses → fire due field events
        → CT tick → next actor
```

- **Order is the contract.** Commits are sequence-numbered and append-only (`CommitLog`); a gap or reorder is a de-sync by definition. The host arbitrates order; clients only ever apply.
- **Randomness** comes from `DeterministicRng` only.
- **It runs headless.** The whole pipeline is unit-testable without rendering (the pathfinder already is).

10.1 The commit vocabulary

kind	payload (sketch)
`move_unit`	`{unit_id, path, facing}`
`wait`	`{unit_id}`
`use_action`	`{unit_id, action_id, target_tile, target_unit}`
`set_block`	`{pos, type, shape, rot}`
`remove_block`	`{pos}`
`apply_damage`	`{unit_id, amount, element, source}`
`apply_status`	`{unit_id, status, ticks}`
`field_event`	`{tick, effect}`
`end_match`	`{outcome}`

Payloads are validated by the handler for their kind; the `Commit` class carries them as data.

11. AI

Enemy AI consumes the **same reachability and cost model** as the player (see the bible): it computes its movement bubble, scores candidate tiles (threat, hazard, height, cover), picks the best, then chooses an action whose targets are in range and LOS. The AI has no special rules — only its own scoring.

12. Framework layout

Module	Status
`src/rules/elements.gd` — the eight elements + affinity helpers	`[built]`
`src/rules/unit_profile.gd` — streamlined stats + affinities	`[built]`
`src/net/commit.gd` / `commit_log.gd` — commit + vocabulary	`[built]`
`src/rules/terrain_rule.gd` — element / destructible / integrity / flammable	`[planned]`
`src/combat/battle.gd` — tick clock, CT queue, actor selection	`[planned]`
`src/combat/turn.gd` — turn phases	`[planned]`
`src/combat/action.gd` — action data	`[planned]`
`src/combat/damage.gd` — formulas (single tunable module)	`[planned]`
`src/combat/target.gd` — reach/LOS resolution (uses pathfinder reachability)	`[planned]`
`src/combat/field_effect.gd` — scheduled events	`[planned]`
`src/combat/collapse.gd` — gravity resolution	`[planned]`

13. Agent rules

1. **Combat simulation is deterministic.** Integer CT/ticks, `DeterministicRng` only, no engine RNG in a sim path.
2. **All map mutations are commits.** Nothing mutates the map out-of-band, in combat or out.
3. **Occupancy is a hard rule.** One unit per tile; exceptions require a documented rule.
4. **Formulas live in one module.** Damage/hit/status constants are not scattered once `damage.gd` lands.

5. **Docs change first.** Update this file before changing combat code.

Status: `[designed]` *— the spec is documented; elements and unit stats are built; the turn controller, damage math, targeting, field events, and collapse are* `[planned]`*.*