Dungeon-Mind

Game design docs for Chris's roguelike: AI-native monsters (Ammann-style action profiles), tick-based PER combat, Pathfinder-inspired rules. Started 2026-09-27.

- Overview
- Combat System
  - Tick Engine and PER Cycle
  - Resolution, Initiative, and Reactions
  - Cancellation and Readability
  - Movement and Terrain
- Monster AI
  - Action Profiles (Ammann Framework)
  - Knowledge State and Decision Models
- Rules and Licensing
  - ORC and Pathfinder Licensing
- Planning
  - Open Questions
  - Roadmap and Next Steps
- Representation and Feedback Doctrine

Overview

Overview

Overview



Dungeon-Mind teaser key art. "No monster fights alone."

Overview

dungeon-mind is Chris's roguelike (ASCII dungeon crawling, procedural generation, LLM-powered content). The current direction (Sept 2026) is a redesign of the combat and monster ecosystems around two big ideas:

1. **Monsters that know what they're doing** - inspired by Keith Ammann's *The Monsters Know What They're Doing*: per-monster-type action profiles with ranked trigger lists.
2. **Tick-based combat (PER cycle)** - every action is prep / exec / recover on a fine-grained discrete tick clock.

Both inspirations are starting points, not targets. The design is going its own direction and may end up too far from either Pathfinder or Ammann to need direct reference beyond inspiration.

Pillars

- **AI-native monsters**: action profiles authored ahead of time as data; profiles evaluated in code with a decision table, not per-action LLM calls.
- **Readable intent**: prep phases are signals in the world; only intelligent/knowledgeable creatures read them. The player and monsters both play the knowledge game.
- **Player-paced, deterministic**: the world advances only when the player commits; seeded, replayable resolutions throughout.
- **Variety without chaos**: every run differs (procedural generation plus seeded distribution sampling on monster choices), but monster behavior stays in-character per profile.
- **Diegetic and numberless**: representations come from the game's own fiction (no cards, no imported tabletop artifacts), the player-facing UI is numberless in the target state (dev builds may overlay numbers behind a flag), and harm is a fixed body plus a scaling avoidance buffer. See Representation and Feedback Doctrine.

Status

- An MVP exists (d20 combat, LLM-driven monsters, ASCII, procedural generation) from earlier in 2026.
- The old repo does not reflect this design direction; treat it as deprecated for now.
- Next step: tick-engine spike (see Roadmap).

Combat System

Tick engine, PER cycle, resolution, reactions, cancellation, movement

Tick Engine and PER Cycle

Tick Engine and PER Cycle

The combat clock is a **global discrete tick**, not real time. Player-paced: the world advances only when the player commits an action or explicitly waits. "Whenever it's time for the user to enter something, everything stops."

Granularity

- Ticks are deliberately tiny: roughly **10 ticks per foot** on even terrain. Integer tick durations everywhere.
- Fine granularity minimizes same-tick ties between actors (though ties still happen - see Resolution).

The PER Cycle

Every action - attacks, spells, movement, cancels, reactions - is modeled identically:

Phase	Meaning
Prep	Wind-up: gathering reagents, mental focus, shifting grip. The action has not started. This is the **readable, interruptible window** - only intelligent/experienced observers can read prep as an intention.
Exec	The action is *active*: casting, swinging. You can't do anything else. May include zone windows (e.g. a whirlwind that slices anyone entering range). The action can be mitigated here, but it cannot be cancelled.
Recover	The action is *done*; the actor is winding down - regaining footing, recovering balance. Nothing can affect the completed action at this point, but enemies can **take advantage of the window** while the actor is not fully recovered.

- **Zero phases allowed**: a fighter feat can zero prep/recover on light attacks to enable combos. Readability and anti-spam live in the prep window, not recover.
- **One action state machine**: every entity in the sim runs the same Action state machine - PREP, then EXEC, then RECOVER. Attacks, spells, movement, cancels, and reactions are all instances of one Action type; they differ only in parameters (phase durations, exec windows, completion effects), never in code path. A cancel or a reaction is itself an action with its own PER cycle, so the same rules compose.

Movement is an action

Movement uses the same PER model; terrain is paid at the borders (see Movement and Terrain).

Resolution, Initiative, and Reactions

Resolution, Initiative, and Reactions

Tick resolution

Each tick runs one loop over every actor, in initiative order:

1. **Actor action processing.** Does the actor have an action in progress?
   - **No** - select the next action (monsters: from the profile's ranked triggers; player: input - the game stops until they commit) and process its first tick. A zero-prep action flows straight through to the next phase.
   - **Yes** - process the current tick of the PER cycle.
2. **Reaction collection.** After the actor's processing, loop the other actors: did this tick trigger a reaction? If yes, add it to the queue - an actor may collect multiple reactions in the same tick. The reactor-side gate applies at collection: an actor in its own exec adds nothing (it is committed, not observing).
3. **Reaction pass.** Loop the reaction queue in initiative order:
   - An actor with several queued reactions evaluates them and uses the best one for the current circumstances; the rest are discarded. Monsters rank by profile; the player is offered the choice.
   - If the actor executes the reaction: pause or cancel their current action (if any), per that action's interruptibility, then process the first tick of the reaction's own PER cycle. A short reaction may complete its exec within this tick and spend subsequent ticks in recover.

Movement is not a special phase - it is an action like any other and runs through the same actor loop. Within a movement action, "leaving" reaction triggers queue before "entering" triggers; initiative breaks ties within each kind.

Deterministic, seeded, logged. This is what makes fights replayable for debugging and tuning.

Initiative

- Rolled at combat start (Perception or Stealth style, as in Pathfinder), **fixed for the whole fight**.
- Used as **priority for same-tick, world-affecting resolutions only** (actor order, reactions, movement events) - not for internal phase transitions.
- Tiebreakers: dex, then level, then PCs-over-monsters, then a **coin flip pre-rolled once per combat** (so identical actors stay consistent all fight).
- **Initiative as a speed stat**: a small random jitter is added to PER cycles; initiative reduces prep and recover time (never exec - exec durations stay consistent so active windows remain readable). High initiative therefore produces more actions over a fight. The jitter is seeded per combat so replays reproduce.

Reactions

Reactions are short actions taken out of sequence, triggered by observing another actor's phase.

Can I react at all? - gated by your own phase:

My phase	React?	Cost
Prep	Yes	Pauses or cancels my prep, per the action's interruptibility
Exec	No	Committed - this is the point
Recover	Yes	The reaction's own PER handles the cost

What can my reaction do? - gated by the observed target's phase:

Observed phase	What a reaction can do
Prep	Can **cancel** the target's action (interrupt the windup). Knowledge-gated: only actors who can read prep as prep can target it.
Exec	Can **mitigate** the action (reduce or disrupt its effect) - or do something entirely unrelated.
Recover	Cannot affect the completed action at all - but the target is vulnerable, so this is an **exploit window**: act while they wind down.

Reactions never *need* to counter the triggering action. They trigger on phase events, not as restricted counters: a magehunter reacts to a mage's casting by throwing a knife; a mage under attack reacts by casting shield for extra AC.

Reactor cost - the reaction's own PER. The tax lives in the reaction's own cycle, never the parent action's. Reactions should be built with short or zero prep and short exec; the **recover is the throttle** - pad it to tax use. This composes with the exploit window: reaction spam leaves the actor in a long recover, where enemies can take advantage.

Interruptibility (pause vs cancel). When an actor reacts, what happens to their in-progress action is an action property: actions that demand deep concentration are cancellable on reaction; lighter actions pause and resume once the reaction fully completes (including its recover). Set per action, like any other parameter.

Baiting is a core player skill: a monster that has already spent its reaction window has no answer to a maneuver.

Cancellation and Readability

Cancellation and Readability

Cancel actions

Cancelling is itself an action with its own PER cycle - not a special-case mechanic. There will be multiple actions that implement cancels:

- **Boundary cancel** - the deliberate cancel at the prep-to-exec edge (below).
- **Deep-prep cancel** - the mid-prep bail action (below).
- **Cancel-on-reaction** - reactions that explicitly cancel the triggering actor's parent action as their effect.
- **Cancel-as-part-of-another-action** - feats or abilities that cancel the parent action as a side effect of doing something else.

Each cancel implementation is its own action with its own prep/exec/recover parameters and its own readability profile. Whether a cancel from a reaction or another action actually lands is subject to the target action's interruptibility (see Resolution). The one action state machine applies to cancels too.

Cancellation policy

Two windows, both in force:

- **Boundary cancel (primary)**: cancels allowed at the prep-to-exec boundary, never between exec and recover. Cheap but never free - even a minimal PER cycle, because a free cancel means commitment means nothing.
- **Deep-prep cancel (fallback)**: cancelling mid-prep is allowed but costs scale with invested prep: `cancel_time = min_cancel + invested_prep * factor` A minimum cancel time prevents instant-cancels whenever an interrupt appears.

Readability - the public/private state layer

Every actor's phase state has two layers:

- **Public**: what the world observes (and the UI shows).
- **Private**: what is actually happening in the sim.

Feats and abilities modify the broadcast, not the sim:

- Fake a cancel (broadcast "recovering", actually still prepping).
- Pretend to continue an action while actually cancelling.
- Read others' cancels more accurately (requires Lore/Int investment).

Cancelling must telegraph (the flinch): hands relax, grip shifts. Else cancel-spam has no counterplay. An experienced enemy reads the flinch and punishes it; a wolf does not.

This is also where the game's knowledge system lives: what you and the monsters can *read* about each other's intentions is a stat, not a given.

Movement and Terrain

Movement and Terrain

Movement is a PER action like any other. Terrain is paid **at the borders**:

Phase	Source of cost
Prep	Departure difficulty of the tile you're on (getting off the log is hard)
Exec	The transfer itself - physical or mental traits
Recover	Arrival difficulty of the destination tile (landing on the log is hard)

Physical traits for exec: dex for balance, strength and encumbrance for load. Mental traits can make the same transfer harder too - fighting through an illusion, or opposing a strong will to continue.

Worked examples (log crossing)

1. **Grassy bank -> log**: minimal prep (solid footing), traits-driven exec (simple step), significant recover (getting stable on the log).
2. **Log -> log**: harder prep (concentrating on footing before shifting weight), same exec (physical or mental traits), same recover (log is still hard to settle on).
3. **Log -> bank**: hard prep (leaving the log), same exec, easy recover.

Rules that fall out of this:

- A multi-tile crossing (long bridge) pays at **every edge** - each tile-to-tile transition is its own PER cycle.
- Difficult terrain can add to all three phases (departure and arrival both hard).
- **Reactions during movement follow the normal rules.** Each movement action is a single tile-to-tile step (not a multi-tile path), so there is no special "mid-path" case - the reactor-side phase table applies exactly as with any other action.

Danger tiles and 3D

Proposed direction: **tiles are 3D, so there are no "open" tiles.** Stepping into a space that is open in the current level means you **fall** until you hit a solid tile, then you're on that level. With a fly or levitate effect (equipment or spell), you stay at your current height unless you take an action to ascend or descend.

Why this stays an open question: ASCII display. Possible techniques: shading on terrain tiles to indicate space above, blank tile to indicate space below.

Open decisions

- **Tile size**: leaning toward 1 tile = 1 cubic foot. A typical human is then about two wide and one deep (footprint), which roughly matches the traditional roguelike glyph (taller than it is wide). Bigger creatures occupy multiple tiles - an ancient dragon should not be the same single character as a human.
- Confirm the per-tile movement baseline at even terrain and the tick basis (see Open Questions).

Monster AI

Ammann action profiles, knowledge state, decision models

Action Profiles (Ammann Framework)

Action Profiles (Ammann Framework)

Each monster type gets an **action profile**: a combat archetype (brawler, sniper, support, skirmisher, ...) plus a **ranked list of actions with trigger conditions**.

Profile structure

- Archetype from inherent physical and mental stats.
- Ranked list: e.g. "first round do X, second round do Y, if HP drops below X then Z". In tick terms: "while caster in prep, if interrupt available and reachable in 3 ticks, use it".
- Trigger axes: timing/windowing, health thresholds, knowledge, **opponent phase state** (the readable-intent system gives profiles a new axis no turn-based game has).

Evaluation model

- Profiles are **data, evaluated in code** - a decision table over game state. Deterministic, instant, free, testable.
- The decision model (Jev/Kev class) is optional garnish for non-computable judgments - see Knowledge State and Decision Models.
- The LLM is for narration only (what the goblin snarls), never for deciding what it does.

Difficulty curve

- **Profile complexity is the difficulty knob, not stat inflation.** Floor 1 monsters get 2-3 pattern profiles; deep floors get 6-8 plus reactions and profiles that read the party.
- **First monsters are characterful**: goblins (cowardly, pack-savvy, learnable) are the iconic early enemy. No boring rats/flies.
- **Knowledge-gated competence**: monsters act optimally against *what they know*, not omnisciently. They observe, spend actions to Recall, and get sharper mid-fight.

Source material

PF2e stat blocks are almost directly convertible into profiles (combat roles, special actions, tactics are built into the monster design). Machine-readable statblock JSON may be available (FoundryVTT pf2e repo - verify license before use) or hand-authored in the game's own format.

Knowledge State and Decision Models

Knowledge State and Decision Models

The judgment layer

Most trigger conditions are computable in code (round/tick counts, HP fractions, allies alive, distance). A few are genuinely judgment-shaped:

- "Is this party vulnerable / outmatched?"
- "Is the player kiting?" (kiting = attacking from range while retreating to stay out of reach - a behavior pattern over time, not a single state; a hardcoded detector misfires on feints, terrain, and repositioning)
- "Is that prep a feint or a commitment?" (especially once fake-cancel feats exist - pure state can't answer this; it's about habits and context)

Typed decision models (Jev/Kev class)

The model supplies judgment; your code owns policy. Ask typed questions over a compact state snapshot; get back probabilities/distributions, not text.

- **noul** booleans for non-computable conditions (one per condition - they must not compete for probability mass).

- **choice** for posture selection; **sample from the distribution**, don't take argmax - seeded for reproducible variety. This is the "every run differs" lever.
- Cost is fractions of a cent per call (tiny state envelopes); latency is a network round trip per monster decision.
- **Provider-swappable**: hosted Jev for dev, local Kev (0.8B fits Terra's P2200; 4B fits Saturn's M2 Pro via MLX) at play time, pure-code fallback for offline. Small local models are option-order sensitive - pin option order per question pack.
- Valid output never means correct judgment - the model's answer is a suggestion the profile policy resolves, not the final word.

What stays in code

- The Ammann profile and its trigger table: always evaluated in code.
- Thresholds, weights, branches - reviewable and tunable without re-prompting.
- Outcome determination and any side effects.

Rules and Licensing

ORC, Pathfinder SRD, scope decisions

ORC and Pathfinder Licensing

Rules and Licensing (ORC / Pathfinder)

The short answer

Building a Pathfinder-2e-inspired roguelike is legal. The PF2e Remaster rules are under the **ORC License** (Open RPG Creative License) - perpetual, irrevocable, royalty-free, and explicitly covering **video games**. Game mechanics aren't copyrightable anyway; ORC grants use of the actual SRD expression (rules text, statblocks, action text).

What ORC covers

- Rules, mechanics, statblocks, monster statistics, character options, conditions, actions, spells.

What it does NOT cover

- The **"Pathfinder" name and logo** (trademark - separate negotiation with Paizo, the Owlcat model).
- **Golarion setting, lore, proper nouns, art, music.**

Obligations if using SRD content

1. ORC Notice (standard copy-paste legal block)
2. Attribution notice (credit Paizo and upstream licensors)
3. Reserved Material designation (declare what's yours)
4. No endorsement claims; 60-day cure for breaches; no royalties, forever.

Chosen path

Path A: Pathfinder-inspired, own world. Own statblock format, own monster names, own setting (the game is redesigning its ecosystem anyway). If we later want specific SRD statblocks, bring them in under ORC with the notice. No Pathfinder branding in the title.

~~Path B: actual Pathfinder brand~~ - contacting licensing@paizo.com for a commercial deal is not needed for this project.

Archives of Nethys status (July 2026)

- Paizo's commercial license with AoN **ended July 24, 2026**, but a looser arrangement keeps AoN as an **official rules archive** (no royalties reporting; art moved to Community Use Policy coverage; CUP art expanding).
- Rules content is ORC-licensed regardless of host - the game plan is unaffected.

Scope

- **A couple of classes to start, not the whole shebang** - and port tactical verbs, not classes: Fighter (exec-window zone control), Rogue (feints), Wizard (prep-heavy spells with duration clocks), Cleric (delayed effects). That exercises every phase of the tick model.

Planning

Open questions and next steps

Planning

Open Questions

Open Questions

Open Questions

Open Questions

Things deliberately left undecided as of 2026-09-27 (design sessions 1-3):

1. **Damage during exec**: no auto-abort by default - large spells stay viable. Design direction: **significant damage triggers a concentration check**; the threshold and which stats the check uses are tuning.
2. **Reaction PER values**: exact phase costs for reaction actions - baseline guideline is short/zero prep, short exec, padded recover as the tax (the 0/1/N shape).
3. **Initiative as speed stat**: how much prep/recover jitter should initiative shave (and does it compound with zero-phase feats)? Also worth asking whether the speed effect is **necessary or even valuable** - initiative already provides tick ordering and reaction priority; the jitter may be complexity without payoff.
4. **Encumbrance model**: weight lands in the exec stat-side for movement (alongside dex) but the concrete numbers are open.
5. **Danger tiles and 3D**: proposed approach is 3D tiles - there are no "open" tiles; stepping into an open space at your level means falling until you hit the bottom, then you're on that level; fly/levitate keeps you at height unless you take an action to rise or sink. Open because ASCII display is hard: shading on tiles could indicate space above; blank tiles could indicate space below.
6. **UI/information design**: the *stance* is settled by Representation and Feedback Doctrine (diegetic representation, numberless player-facing UI in the target state, numbers behind a dev flag during development), so this question is now narrower than it was. Open residue: how ASCII conveys phase state, and whether the *player's* Lore/Int gates what the UI reveals (identification-tradition style). E.g. a first-ever encounter with a baby dragon - does it show as "d" (dragon) or as "L" (big lizard, because you've never seen one)?

7. **Exact tick basis and tile size**: confirm the ~10 ticks/foot granularity and the base tile (leaning to 1 cubic foot; human footprint ~2 wide x 1 deep, matching the taller-than-wide roguelike glyph). Per-tile movement baseline at even terrain.
8. **Decision-model backend choice**: hosted vs local vs fallback, when the judgment layer actually ships.
9. **Reaction timing**: lean is a **uniform rule - all reactions resolve on the tick after their trigger** (no same-tick vs next-tick split). Consequence: single-tick execs are effectively uninterruptible by reactions; multi-tick execs have interruption windows (see 17).
10. **Reactive builds**: acting almost entirely on reactions (a duelist stance that punishes anything acting nearby; reacting during your own reaction's recover to skip the tax). The exploit window makes this risky by design - a feat/stance space to explore.
11. **Interruptibility details**: which actions pause vs cancel when the actor reacts (concentration-dependent `on_react` property); the taxonomy needs building out per action.
12. **Terrain on all prep/recover?**: should terrain influence every action's prep and recover, not just movement? Casting on a mountain ledge in a thunderstorm should be harder than on steady ground. If yes, terrain applies a general prep/recover modifier, with movement-specific rules layered on top.
13. **Core resolution mechanic**: direction - **drop the d20**; it's all probabilities on a computer. Use a **uniform roll on 0-1**. Displaying it as 1-100 percentages (roll*100+1) is confined to dev tooling and tuning logs: the player-facing UI carries no numbers per Representation and Feedback Doctrine. Keep the **degrees of success** mechanic from Pathfinder - it prevents nat-20/nat-1 -style oversized effects for events that should happen 5% of the time, and gives leveling a solid mechanism (the experienced archer is reliably better). Note: a uniform 0-1 roll is mathematically identical to a d20 (both flat) - the real questions are which thresholds define success/crit bands (PF2e uses +10/-10 over the DC) and whether to keep the distribution flat or move to a bell curve (2d10-style) to tame the tails further. The open residue: how close to Pathfinder's threshold structure vs making it our own.
14. **Tick clock scope**: direction (initial) - while in the dungeon proper the clock **always runs**; any actor within XX tiles of activity is part of the PER loop. Open: the XX radius, rest, non-combat behavior, patrols and wandering (phased exploration).
15. **Player decision points**: direction - player *reactions* are automated via a **player-set decision policy** - the first real "let the AI handle something" option for the player. It must be **logged** so the player can review and tweak their policy. Open: the policy UI, default policies, and this doubling as the testbed for autopilot mode.
16. **Group tactics**: largely resolved - **no separate squad AI**. Even in a group, every monster decides independently; team tactics are built into each monster's decision policy as teammate-conditional rules: fighters "if a teammate is within x tiles, move to flank" or "if a teammate caster is present, stand between them and the enemy"; archers "move behind the meat shield / behind cover before firing". The specific rules are tuning.
17. **Interrupted or death mid-action**: broader than death - what happens to actions interrupted mid-tick? With the uniform next-tick reaction rule (see 9), **single-tick execs are effectively uninterruptible; multi-tick execs can be interrupted** (many spells, the whirlwind). Interruption behavior is specified **per action; default: nothing happens**

. Example: a two-tick fireball (tick 1 summon, tick 2 shoot) interrupted on tick 2 explodes in the caster's hands. Death is one cause of interruption, not the only one. Open.

18. **Senses and attraction**: strong direction - alerting monsters is a **full sensory model, gated by the monster's nature**: noise, smell, distant lights in darkness, the sudden *absence* of light, a patrol **not showing up when it should** (which triggers very different responses from lazy goblins vs trained mercenaries). Loud environments mask sounds - near an active forge or rushing waterfall, nothing triggers. Intelligence and Perception are the key drivers. Specs open.

Planning

Roadmap and Next Steps

Roadmap and Next Steps

Spike 1 - tick engine core (first build)

The smallest thing that proves the system:

- Two actors (player fighter vs goblin), one action each with full PER cycles.
- The whirlwind exec window (zone damage while active).
- A spell prep + interrupt case.
- The resolution doctrine (actor loop in initiative order, reaction queue, uniform next-tick reactions).
- **Seeded replay test** - same seed, identical fight.
- Plus the goblin action-profile schema (archetype + ranked triggers as data).

After the spike

- Terrain-at-borders movement costs and the 3D/falling model.
- Reactions and the cancellation policy (boundary + deep-prep + telegraph).
- Profile data + code evaluator (the decision table) and a few monster types.
- The resolution mechanic (0-1 roll + degrees of success) wired into the sim.

Later

- Decision-model question pack for intent-reading judgment (hosted/local/fallback).
- A couple of playable classes (the tactical verbs: zone control, feints, prep-heavy spells, delayed effects).

- Knowledge system polish: what monsters and players can read about each other.
- Sensory model (noise, smell, light, patrol rhythm) + attacker attraction (Open Questions 18).

Far future

- **Autopilot/intent mode**: the user defines the player's intent and the app controls the PC. The player-set decision policy for reactions (Open Questions 15) is the first step toward this.

Future planning (not yet detailed)

Areas identified but still need their own planning sessions - placeholders to revisit:

- **Treasure, loot, and inventory**: what drops, how inventory works (weight/slots - the encumbrance question folds in here).
- **Equipment and gear**: weapons and armor as PER parameter sets (attack speeds, reach, zone windows, lingering effects).
- **Leveling and progression**: XP, level-up mechanics, how the degrees-of-success math scales with experience.
- **Feat selection**: how players earn and choose feats (zero-phase combos, cancel feats, reactive builds).
- **Framework choice**: build on an existing roguelike framework/library or start from scratch? Also relevant to the tick engine.
- **Character creation**: race/class/stat generation, within the "two classes to start" scope.
- **Dungeon generation and the AI vision**: the original goals (AI-driven dungeon variety, NPC conversations, descriptive text) still need their own planning.
- **Non-combat simulation**: rest, world clocks, patrols (Open Questions 14), senses and attraction (Open Questions 18).

Representation and Feedback Doctrine

Representation and Feedback Doctrine

Purpose

This page fixes the game's presentation stance: what game objects are allowed to look like, what the interface is allowed to tell the player, and how harm is modeled and communicated. It is a governing principle, not a tuning sheet. Tick mechanics live in the Combat System chapter; this governs how they are *shown*.

Three rules, then their known costs.

Rule 1: representation must be diegetic

A game object is represented by something that exists in the game's fiction. A sword is a sword. A spell is cast, not played. A character does not hold a hand of cards.

The test that separates a legitimate abstraction from an import: **abstractions that evolved inside the medium are fine; objects borrowed from another medium are a tell.** A health bar, a minimap, a pause menu and a level number are all non-diegetic, and all of them grew up inside computer games. A card is different in kind. It is an artifact of tabletop, and importing it means the fiction has been subordinated to a convention from a different medium. The signal the player receives is "board game in a video game suit."

The card test: strip the cards off a design and present the same multiset of options in any other form, keeping the draw rules identical. If the game plays the same, the cards were skin and should be replaced with the diegetic object.

Consequence: **no literal card objects.** Where a design reaches for cards, use the diegetic object instead (an ability, a weapon, a stance, a readied slot) and represent its state in the world.

Hidden information is not a defence for cards. The *bands solved diegetic hidden information decades ago: items carry a flavor appearance and stay unidentified until handled, used, or read with an identify spell; monster knowledge accumulates through fighting. Same information asymmetry, no cardboard.

Reference points where the same mechanic is expressed as an object in the world: Into the Breach (abilities always available, constrained by position and turn order), FTL (weapon charge times shown on the ship), Civ production queues, Darkest Dungeon (the constraint is position and stress, not a hand).

Rule 2: numberless is the target state, numbers are a dev aid

End state: the shipped player-facing UI carries no numeric readouts for health, damage or state. Feedback is textual and qualitative.

During development: numbers are permitted in the display behind a dev flag, for engineering and tuning. The flag is a debug affordance, not a crutch: numberless is the default and the flag is opt-in, never the reverse.

The reason for the end state: a number on screen substitutes for a mental model. The player reads the number instead of learning the system. Removing it forces the design to teach through description.

Build the text layer first, overlay the numbers on top

The ordering matters more than the flag. If the text vocabulary is deferred until "final product", it gets retrofitted under schedule pressure and then quietly dropped, and the game will have been tuned for a readout it no longer has.

So: **the text channel is primary from the first playable build and is exercised in every session. Numbers are an overlay drawn on top of a system that already speaks.** Every state change must emit its qualitative line before the numeric flag can be considered finished, and a build with the flag off must never be the first time anyone reads the wording.

The legibility test runs with the flag OFF

The two playtest questions at the bottom of this page only measure the text layer if the numbers are hidden. Run them with the dev flag off even mid-development, then flip the flag on afterwards to check the player's answer against the real numbers. Numbers on for engineering, numbers off for legibility testing.

Numbers stay a calibration channel

Even at the end state, numbers do real work off-screen: they let a designer verify that the model matches the game's actual state. That channel must be *replaced*, not merely deleted, in the player-facing UI. **Every qualitative phrase must sit on a consistent, learnable, ordinal scale, globally.** "A little", "a lot" and "much" must mean the same magnitudes everywhere, and the same phrasing must mean the same thing whether the source was a sword or a fireball.

Text that is decorative rather than ordinal fails this. "The goblin looks annoyed" conveys nothing the player can calibrate against, so nothing is learned. If feedback cannot be ranked, it is flavour, not information.

Precedents: Angband's textual feedback (monotonic phrasing the player learns), Brogue (deliberately playable with no numeric readouts, bars and description only), Dead Space (health on the character's spine, ammo on the weapon) for the diegetic HUD case.

Rule 3: fixed body, scaling avoidance

Health does not scale with level.

- Bodily capacity is essentially fixed, varying slightly between individuals and over a life, not by experience.
- What experience buys is **a buffer of avoidance**: honed reflex, knowing when to duck, tucking and rolling, rolling with the punch, deflecting with a blade.
- The buffer depletes through a fight, accumulating nicks and scratches in proportion to the character's experience.

- When the buffer is exhausted, real damage lands, and real damage has **real effects**: on attack, on movement, on breathing.

Progression therefore shows up as *you get harder to hit*, not *you have a bigger pool*. Survivability still grows. It grows as competence.

Precedents: Into the Odd (hit protection depletes first, then damage goes to attributes), Blades in the Dark (harm levels that reduce effect), Sekiro (posture and vitality as separate tracks).

Failure mode: damage must degrade verbs, not subtract quantity

If the buffer is a hit point pool with a new name, the design has rebuilt HP and bolted text onto it. The model only pays off when real damage changes **capability**:

- broken rib: cannot draw a bow, breathing is audible, stealth suffers
- cut tendon in the leg: slower movement, cannot sprint
- blood in the eye: attacks miss more

These are categorical state changes. A wound system is a set of disabled or degraded verbs. A hit point pool is a subtraction.

Test: if resting restores a character completely, it is HP regardless of what it is called. Wounds have to change how the player plays.

Accepted cost: variance

Fixed body plus an avoidance buffer means an experienced character takes fewer hits, but each real hit is a larger fraction of their life. The distribution widens: most fights are comfortable, then one unlucky exchange nearly kills a veteran.

This is deliberate. It buys tension and costs computability. In a numbers game a careful player can prove they are safe; here they cannot compute survival at all. That is the trade, and it defines the audience. The min-max players lost here are lost on purpose, not by accident.

Feedback as rising action

A depletion buffer lets combat carry a narrative arc instead of a subtraction: fresh, working, winded, guard failing, exposed, wounded. Each stage is a beat with its own vocabulary. This is the main creative payoff of Rule 3, and it is only available if the feedback is ordinal (Rule 2).

Production cost: text at volume. Every ability, monster, wound and branch needs its own line, with enough variation to avoid becoming wallpaper. This is the main expense of the whole stance, and it is writing, not code.

Validation: two playtest questions

Neither requires a UI mockup.

1. After a fight, stop the player: **"How hurt are you, and what is wrong with you?"** Compare their answer to actual state. If they cannot answer, the feedback vocabulary failed.
2. **"Would you fight another one of those right now?"** Compare their answer to the real odds. If willingness does not track actual risk, the representation is lying to the player.

The two catch different failures. (1) catches a vocabulary that cannot describe the state. (2) catches a vocabulary that describes it inaccurately.

Where this collides with existing decisions

- **Open Question 6** (UI/information design) is now governed by this page. The knowledge-gated ASCII question stands, but any answer must satisfy Rules 1 and 2, which means a numberless player-facing UI in the target state and numbers available behind a dev flag during development.
- **Open Question 13** previously allowed the 0-1 roll to be shown as 1-100 percentages "for readable tuning". Rule 2 disallows numeric display in the **player-facing UI**. Percentages remain legitimate in dev tooling and tuning logs, where the audience is the designer and the seeded replay log rather than the player.
- Downstream consequences: monster knowledge display, wound vocabulary, level-up signalling, and how ASCII glyphs convey phase state all become children of this page.

Related

- Combat System > Tick Engine and PER Cycle
- Combat System > Cancellation and Readability (readability is diegetic signalling, the same principle applied to intent)
- Planning > Open Questions (6, 13)
- Overview (pillars)