Monster AI

Ammann action profiles, knowledge state, decision models

- Action Profiles (Ammann Framework)
- Knowledge State and 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.