ZEngine

Unified Design Manifest

Version 0.2 · February 27, 2026

Social ecology simulator / party-based roguelike

Pre-implementation · Design complete · Handoff-ready

Author: Zachery · Solo indie development

§1 Agent Orientation

Read this section first. It contains everything needed to orient an AI coding agent before touching any ZEngine code. Design philosophy, rationale, and post-MVP content follow in later sections.

1.1 Project Identity

ZEngine is a party-based (1–3 PCs) wilderness exploration game set in a hostile world punctuated by Points of Light — template-grounded, procedurally populated settlements on a vitality spectrum from flourishing to collapsed ruin.

Experiential references: Caves of Qud / Fallout 76 (open hostile wilderness, deep simulation legibility) · Divinity: Original Sin 2 / Baldur's Gate 3 (consequential social interaction) · Diablo / ARPG (kinetic combat feel) · Retro terminal / roguelike aesthetics (meaning carried by system density).

1.2 Confirmed Tech Stack

Component	Decision
Language	Python 3.14.3 (stable Feb 3 2026)
ECS	python-tcod-ecs
Renderer	tcod terminal renderer
Spatial math / lighting	NumPy — scoped to spatial layer + alpha gradient lighting ONLY
Data validation / event models	Pydantic v2
Event bus	Bespoke pub-sub on Pydantic v2 (see engine/combat.py)
Chronicle storage	JSONL (append-only, write-on-dispatch)
Templates / abilities / grammar	TOML
Annotations	Python 3.14 deferred annotations (PEP 649)

1.3 Directory Structure

```
zengine/
├── data/
│   ├── abilities/            # TOML ability definitions
│   ├── grammar/              # TOML grammar tables (authorial — NEVER agent-
generated)
│   ├── templates/            # TOML chunk / Point of Light structural templates
│   └── chronicle_significance.toml
├── engine/
│   ├── chronicle.py          # Chronicle write/query interface
│   ├── social_state.py       # Social State schema and transition logic
│   ├── equilibrium.py        # Vitality, conversion rate, conduction
│   ├── combat.py             # Turn resolution, action economy, event bus
(canonical)
│   └── ecs/                  # ECS component and system definitions
```

```
├── world/
│   ├── generator.py          # WFC room placement, cyclic topology
│   └── wilderness.py         # Encounter density, negative space generation
├── ui/
│   └── renderer.py           # tcod terminal renderer
├── sessions/
│   ├── chronicle.jsonl       # Active Chronicle (append-only)
│   └── spatial_snapshot.toml
├── CONTEXT.md
├── DESIGN_VARIABLES.md       # All design variables with current defaults
├── FUTURE.md                 # Post-MVP deferred systems
└── DO_NOT_TOUCH.md           # Locked files per phase
```

1.4 Phase Gate Table

Phase	Goal	Entry Condition	Exit Criteria
0	CONTEXT.md ritual, opening/closing prompt templates	v0.2 manifest complete	CONTEXT.md committed, templates tested
1	Extract COMPONENTS.md, SYSTEMS.md, EVENTS.md contracts	Phase 0 complete	All three contract files committed, no code written
2	Static terminal render loop with ECS skeleton	Phase 1 complete	Entity renders to terminal, event bus dispatches
3	WFC generation with cyclic topology validation	Phase 2 complete	Room graph generates, cycles validated
4	Markov history generator producing 10–30 events	Phase 3 complete	Chronicle populated with fabricated prehistory
5	Lore objects with lacunae principle	Phase 4 complete	Lore objects generate from Chronicle fragments
6	Integration + full MVP validation + save/load	Phase 5 complete	All systems talk, session persistence verified

1.5 Design Variable Quick-Reference

If you encounter a configurable parameter with no specified value, log it in DESIGN_VARIABLES.md with the default used. Never hardcode silently.

Variable	MVP Default	Notes
Social Layer simulation mode	Background tick at session boundary	→ Real-time daemon post-MVP
Chronicle mode	Passive record-keeper	Schema identical to active mode; activation is additive

Chronicle significance threshold	2	Range 1–5
Reputation thresholds	Ostracization −0.3 / cooperation +0.4	Expose as config
Stress exodus threshold	0.7	float 0.0–1.0
Stress passive decay rate	0.0	0.0 = no passive decay
Equilibrium Taper base resistance	40	Range 20–80
Conduction coefficient	0.3	0.0 disables conduction
Conduction attenuation factor	0.6 per distance unit	Range 0.1–0.9
Energy threshold (turn eligibility)	100.0	float
AP pool size	TBD — register before Phase 2	
Movement allocation	TBD — register before Phase 2	Speed-derived
Combat roll display	Outcome category	hit / graze / miss / critical / fumble
Social Layer catch-up ticks	TBD — register before Phase 1	
Vitality cache	NOT IMPLEMENTED MVP	Stub field present; activate post-MVP

1.6 Agent Hard Limits

These apply across all phases and override any other instruction.

1. No direct inter-layer state mutation. All Social Layer changes from the Dungeon Layer must pass through a Chronicle event first.
2. No Chronicle entry modification after inscription. Corrections are new entries with a supersedes reference.
3. No vitality caching during MVP. The cached_vitality stub field must not be populated or read.
4. No hardcoded ability or event behavior. All effects expressed as TOML data and tag subscriptions.
5. No NumPy outside spatial layer and lighting.
6. Grammar tables are never agent-generated. Agents may read, never create or modify.
7. Design variables are documented before use. Log undocumented parameters in DESIGN_VARIABLES.md.
8. Post-MVP systems are not architected toward during MVP phases. Note in FUTURE.md and implement the simpler version.
9. Never mutate Combatant.hp directly. Always call Combatant.apply_damage().
10. Never use raw strings as event keys. Use EVT_* constants from engine/combat.py.
11. Never instantiate a global EventBus singleton. Pass the instance at construction.

§2 System Contracts

Contract 1: World Chronicle

Append-only JSONL event store. Single source of historical truth. Bridges both simulation layers and all sessions. All inter-layer communication passes through the Chronicle. No exceptions.

Schema:

```
ChronicleEntry:
  event_id:       UUID
  timestamp:      {era: Ancient|Middle|Recent, cycle: int, tick: int}
  provenance:     witnessed | fabricated
  legibility:     transparent | obscured
  actor_handle:   abstract reference (living | Legacy | faction | place)
  payload:        {event_type, verb, object, modifier}
  confidence:     float 0.0–1.0  (default: 0.9 witnessed, 0.4 fabricated)
  citation_count: int  (dormant MVP; active post-MVP reconciliation)
  significance:   int 1–5  (minimum inscription threshold: 2, configurable)
```

Epistemic layering:

- witnessed + transparent = unambiguous record
- witnessed + obscured = real history rendered fragmentary
- fabricated + transparent = acknowledged myth
- fabricated + obscured = default state of procedural prehistory

Prime directives:

- Append-only. Corrections are new entries referencing the superseded event.
- All inter-layer communication passes through the Chronicle.
- Provenance is assigned at inscription and never changed.
- Write-on-dispatch. Session boundary markers: session.opened and session.closed.

Contract 2: Social State

Per-actor persistent record. Carries reputation, moral weight, stress, and retirement history.

Schema:

```
SocialState:
  actor_handle:    Chronicle-registered reference
  reputation:      float -1.0 to 1.0  (default 0.0)
  moral_weight:    float 0.0 to 1.0   (default 0.5)
  stress:          float 0.0 to 1.0   (default 0.0)
  resilience:      float 0.0 to 1.0   (default 1.0)
  consequence_log: List[event_id]
  retirement:      RetirementRecord | None

LegacySocialState (extends SocialState):
  legacy_handle:           actor_handle  — immutable at retirement
  origin_handle:            actor_handle  — bidirectional lineage link
  retirement_chronicle_id: UUID
```

Field distinctions:

- reputation — written by social interaction systems; drives behavioral consequences and Apathy Exodus
- moral_weight — written by Chronicle reconciliation only; governs lore description register
- stress — written by Chronicle events tagged stress_delta; third independent field
- resilience — absorbs stress deltas (DOS2 binary armor adapted for stress); depletes under pressure, recovers at rest in high-vitality nodes

Retirement transition (5-step sequence):

12. Retirement Chronicle event inscribed (carries final reputation, moral weight, cause)
13. Legacy Actor handle created; LegacySocialState populated with bidirectional link
14. Living actor flagged retired; removed from all active simulation pools
15. Legacy Actor enters archaeological record; begins accreting citation counts
16. Adjacent actors may receive Social State modifications per relationship tags

Prime directives:

- Reputation and moral weight have distinct producers. Never conflate.
- No system may act on the retirement flag without querying the retirement Chronicle event for cause context.
- Legacy Actor records are immutable after creation.

Contract 3: Equilibrium Dynamics

Governs node vitality, Legacy conversion rate, and inter-node social conduction.

Node Vitality (computed on demand — never cached MVP):

Vitality Range	Node Behavior
0.6–1.0 (flourishing)	Migration inflows. Rich population. High Chronicle rate. Dense living NPCs, sparse lore.
0.2–0.6 (stable)	Equilibrium. Moderate migration. Nominal Chronicle rate.
−0.2–0.2 (declining)	Apathy Exodus active. Accelerated Legacy conversion.
−1.0–−0.2 (collapsing)	Rapid Legacy conversion. Outflows only. Collapse events inscribed.
Collapse threshold	Ruin state. No living NPCs. Maximum lore density. Permanent within session.

Equilibrium Taper formula:

```
BASE_RESISTANCE = 40  # configurable
taper_threshold = BASE_RESISTANCE + (living_count * vitality_score)
migration_occurs = random(1, 100) > taper_threshold
```

```
# No node becomes a population attractor regardless of vitality.
```

Inter-node Social Conduction:

```
conducted_delta = source_delta * CONDUCTION_COEFFICIENT * (ATTENUATION_FACTOR **
distance)
# Applied to faction-affiliated actors in target node matching subject_faction_id
# CONDUCTION_COEFFICIENT default: 0.3
# ATTENUATION_FACTOR default: 0.6 per distance unit
# Cross-faction conduction: post-MVP only
```

Contract 4: Combat System (Stub)

Canonical implementation: engine/combat.py. All contracts below are enforced in that file.

Turn resolution:

Energy-based. Each actor carries action_energy (float) accumulating at rate = speed attribute. At ENERGY_THRESHOLD (default 100.0), actor is eligible to act. Three dispatch points per actor activation:

- combat.turn_started — actor activation
- combat.action_resolved — per action (attack, ability, item, etc.)
- combat.turn_ended — end of actor's turn; modifier expiry fires here
- combat.round_ended — after all actors complete turns; fires on both combatants

Resolution mechanic:

- d20 + attack_bonus vs BASE_HIT_DC + defense_bonus
- Outcome categories: fumble / miss / graze / hit / critical
- Graze: missed DC by 1–4 — deals half damage
- Critical: natural 20 — max damage (6 + damage_bonus)
- Display defaults to outcome category; raw roll is configurable flag

Event Bus Contract:

- EventBus is Pydantic v2–typed. All events are CombatEvent (BaseModel) subclasses.
- Wildcard key "*" receives every emitted event (used by Chronicle).
- Per-handler errors are swallowed and logged to stderr — emission always continues.
- Modifier expiry is event-driven (Option C). Modifiers declare expires_on: list[str]. Engine loop never manages modifier duration.
- max_triggers controls multi-hit modifier effects (e.g., absorb-3-hits shield).

Canonical event keys (EVT_* constants in engine/combat.py):

Constant	Key String	Emitter	Description
EVT_TURN_STARTED	combat.turn_started	CombatEngine	Actor activation
EVT_ACTION_RESOLVED	combat.action_resolved	CombatEngine	Per action result; Chronicle-ready payload
EVT_TURN_ENDED	combat.turn_ended	CombatEngine	End of actor's turn;

			modifier expiry
EVT_ROUND_ENDED	combat.round_ended	CombatEngine	After all actors; fires on both combatants
EVT_ON_DAMAGE	combat.on_damage	Combatant	HP mutation; carries amount, hp_remaining
EVT_ON_DEATH	combat.on_death	Combatant	HP ≤ 0; carries final_hp
EVT_MODIFIER_ADDED	combat.modifier_added	Combatant	Modifier registration
EVT_MODIFIER_EXPIRED	combat.modifier_expired	Combatant	Modifier self-expiry
EVT_SOCIAL_STRESS_SPIKE	social.stress_spike	Combatant (stub)	No-op Phase 0; wired Phase 1
EVT_SOCIAL_DISPOSITION_SHIFT	social.disposition_shift	SocialState (stub)	No-op Phase 0; wired Phase 1

Prime directives:

- No direct health or Social State mutation from combat systems. All changes via Chronicle events.
- Height mechanics: excluded MVP and post-MVP.
- Multi-agent NPC planning: excluded. NPC behavior is Social State-driven event subscriptions only.

Contract 5: Save / Load / Continue

Within-session: Chronicle writes to disk on every dispatch. No additional save action required.

Continue prompt:

On application launch, scan Chronicle for most recent session.opened without corresponding session.closed. If found, offer Continue / New Game prompt.

Load reconstruction (two sources):

17. Chronicle query → world seed, party composition, Social State records
18. Spatial snapshot → lightweight TOML/JSON written at session close containing dungeon map seed, room graph, party positions. Disposable cache, not authoritative. Discarded on new session start.

Fallback: If spatial snapshot absent or corrupted, regenerate dungeon from seed (deterministic generation is a Phase 3 validation requirement — this is free).

Cross-session persistence: Post-MVP. MVP Chronicle is session-persistent only.

§3 Two-Layer Simulation Model

Social Layer — outer simulation loop. Runs continuously. Manages all living actors and their Social States, governs faction dynamics and population equilibrium, produces Chronicle events, converts retired actors to Legacy Actors. Advances via background tick at session boundary (MVP default).

Dungeon Layer — inner experience loop. Player observation interface. Generates spatial cross-section of current world state. Writes back to Social Layer exclusively through Chronicle events. No direct inter-layer state mutation — ever.

§4 Grammar Tables

Location: data/grammar/ — TOML files, one per semantic category.

Authorial status: First-class deliverables. Never generated algorithmically. Never modified by an agent without explicit author instruction.

Minimum entry schema:

```
[[entries]]
value = "string to be drawn"
weight = 10            # relative frequency, positive int
tags = ["mercantile", "faction_guild"]
era_affinity = ["Ancient", "Middle"]  # optional
```

History generator draws via weighted random selection filtered by Chronicle event era and subject actor faction tags. Tag filtering is the only coherence mechanism — all other coherence is authorial.

Related config: data/chronicle_significance.toml — maps event types to significance integers 1–5. Minimum inscription threshold: 2 (configurable).

§5 Design Philosophy & Rationale

5.1 Foundational Principles

- Two-layer simulation / Chronicle-only inter-layer interface: Prevents tight coupling that collapses the simulation into a dungeon crawler with social flavor. The Chronicle's provenance model is only meaningful if it is the genuine single source of truth.
- Pydantic v2 event bus over framework: Keeps the event bus legible to agents across sessions. Pydantic validation at dispatch catches malformed events at the boundary. Bespoke implementation avoids framework archaeology.
- Energy-based turn order: Exposes a clean interrupt/reaction surface without requiring a separate interrupt resource type. Reactions consume AP from the reacting actor's existing pool.
- Stress as third Social State field (not karma modifier): Allows stress to interact with future systems independently of reputation. Conflating them into a single float would prevent decoupling.
- Resilience as stress absorption buffer: DOS2 binary armor principle applied to stress. Players understand they must deplete resilience before stress consequences manifest.
- Stress release requires agency (passive decay = 0.0 default): Stress should feel earned in both directions. Passive decay trivializes the system.
- Grammar tables as authorial deliverables: The project's aesthetic identity lives in these tables. Algorithmic generation collapses the lacunae principle and the fabricated/witnessed epistemic distinction.
- Node vitality on-demand computed (never cached, MVP): Stale vitality produces incorrect equilibrium behavior. Cache post-MVP with cached_vitality stub present from day one.
- Chronicle writes to disk on every dispatch: Crash-safe by architecture. Session recovery reads Chronicle to rebuild in-memory state.
- Danger/narrative reward structural coupling: The most historically rich dungeon sites are collapsed nodes deep in hostile wilderness. This emerges from simulation dynamics, not designer spawn table placement.
- Event-driven modifier expiry (Option C): Modifiers self-expire on named events. Engine loop never manages modifier duration. Modifier system becomes a first-class citizen of the event pipeline, not a parallel loop.

5.2 Closed Doors

Exclusion	Rationale
Node.js as primary language	Python 3.14.3 for ECS library alignment (python-tcod-ecs) and Chronicle JSONL tooling.
NumPy in Social Layer calculations	Equilibrium math is simple enough for plain Python floats. NumPy scoped to spatial layer and lighting only for debuggability.
Height mechanics in combat	Inappropriate overhead for terminal-based roguelike. Excluded MVP and post-MVP unless spatial model expands.
Multi-agent NPC planning architecture	Confuses AI coding agents with in-game NPC agents. NPC behavior is Social State-driven event subscriptions only.

Direct Social State mutation from combat systems	All changes via Chronicle events. No exceptions.
Chronicle entry modification after inscription	Append-only is inviolable. Corrections are new entries.
Cached vitality during MVP phases	Stub field only. Do NOT implement cache logic during MVP.
Cross-session persistence (MVP)	Chronicle is session-persistent only. Full persistence is post-MVP.
Cover / line-of-sight bonus mechanics	tcod FOV handles LOS. Cover as mechanical bonus excluded at MVP.
Explicit reputation-tracking UI	World responds, Chronicle records why. No reputation dashboard.
d20 replaced by 2d8 prototype	d20 + modifiers is canonical (Contract 4). The zengine_combat_v0.1.py prototype is superseded.

§6 Post-MVP Deferred Systems

Do not implement during any MVP phase. Do not architect toward these systems in ways that add MVP complexity. Log anticipated integration points in FUTURE.md only.

- Chronicle Active Epistemology — reconciliation daemon, confidence weight adjustment by citation frequency
- Faction Hooking — lore object reading updates party faction standing
- Tag-Based Casting (Wildermyth model) — personality/relationship tags drive narrative actor selection
- Ancestor Worship / Healing Legacy — martyr IDs generate spatial aura effects near tomb locations
- Gift Economy — altruism-driven item transfers between high-reputation and high-stress actors
- Third Gender / Custom Caste System — non-binary actor castes; architecturally trivial given actor_handle abstraction
- Blue Prince Mode — player-facing blueprint drafting for Points of Light layout
- Cross-Session World Persistence — full Chronicle accumulation across runs
- Ruin Recovery — collapsed node regeneration via player-driven Chronicle events
- Cross-Faction Social Conduction — reputation propagation beyond faction scope
- NPC AI Goal-and-Tactic Planning — meaningful only after Social Layer is stable at scale
- EventPayload.data typed dicts per event type (Phase 1 hardening task)

§7 Open Threads

These must be resolved and registered in DESIGN_VARIABLES.md before the indicated phase begins.

Item	Required By	Status
AP pool size	Phase 2	TBD — register in DESIGN_VARIABLES.md before Phase 2 begins
Movement allocation (speed-derived distance)	Phase 2	TBD — register in DESIGN_VARIABLES.md before Phase 2 begins
Social Layer catch-up tick count at session boundary	Phase 1	TBD — register before Phase 1 begins
Combat roll display toggle (raw vs. category)	Phase 6 UI pass	Default confirmed: outcome category. Raw roll as configurable flag.
EventPayload.data typed dicts per event	Phase 1 hardening	Open dict in Phase 0; typed dicts are Phase 1 task