Layering the World

History, State, and Appearance in Persistent Generative Worlds

Flyxion
Independent Researcher

September 2026

Abstract

Learned world models and generative game engines are usually classified by a single question: does the system maintain an explicit state with a transition model, or does it predict frames directly? This essay argues that the binary is one layer short. A durable computational world requires three layers, rendered appearance, operative state, and historical inscription, and the third is absent from nearly the entire learned-world-model literature, including the systems that possess explicit state. The essay then addresses the architectural question the third layer forces: which layer is authoritative. Neither a state-primary design, in which history is a side-log, nor a naively log-primary design, in which every solver step is a constitutive event, is adequate. The proposed resolution divides authority by scale: history is authoritative at commitment boundaries and state is authoritative between them. The consequences are worked out in detail: three temporal scales, a single append-only history yielding several certified folds (operative state and provenance state among them), rendering as a pure function of those folds, attestation rather than observation as the criterion for an event, refusals and replay discrepancies as first-class records, and a directed acyclic history whose distributed fold is well-defined by the residue algebra already required of such worlds.

1 Introduction

A generative system can produce a wall that looks continuous across frames and yet is not the same wall from one frame to the next. This is the problem that motivates the persistent-world program: the difference between a picture that is regenerated and a place that persists. The usual diagnosis is that frame-predicting systems lack state, and the usual prescription is to give them one, a latent variable with a learned transition function that a planner or policy can consume. The diagnosis is correct as far as it goes. This essay argues that it stops one layer short.

A system with an explicit state and a transition model can still fail to be the same world over long horizons, because it reconstructs rather than remembers. Two systems can hold extensionally identical states and nonetheless be different worlds, because one of them was built, damaged, repaired, owned, and witnessed in ways the other was not. Nothing in the state encodes this; it lives in the history of how the state came to be. The question that follows is not whether a world needs a history but which of the two, state or history, has authority when they disagree. That is the layering question, and it has a definite answer.

The essay proceeds in three movements. The first reads the learned-world-model literature through a three-layer classification and shows that the third layer is missing even where the second is present. The second states and defends the layering principle: history is authoritative at commitment boundaries; state is authoritative between them. The third works out the consequences for what counts as an event, how rendering consumes history without scanning it, how refusals and replay discrepancies are recorded, and how the history behaves when it is a graph rather than a line.

2 Three layers, not two

The literature on learned world models divides cleanly, and the division is informative. On one side are latent-state predictive models built for control. These commit to a state s_t, a learned transition $p(s_{t+1} \mid s_t, a_t)$, and a consumer of that structure, whether a planner searching over imagined trajectories or an actor-critic trained inside the model's imagination. PlaNet plans in a learned latent space; Dreamer propagates value gradients through the compact latent state of its learned model; MuZero learns an iterable model that predicts policy, value, and reward without access to the true dynamics. IRIS, DIAMOND, and the later Dreamer versions belong to the same family. In these systems the state space is a space of possible states and the transition model is an admissible-transition relation. Whatever their limits, they possess a stated s_t.

On the other side are neural and generative game engines: GameNGen, the Genie lineage, Oasis, GameGen-X, the Matrix and Matrix-Game systems, and their relatives. These are autoregressive frame predictors conditioned on past frames and actions. What they call a latent is, in GameNGen's own description, a compression latent inherited from an image autoencoder, a representation that supports reconstruction and prediction of pixels but does not commit to a world variable that a planner could act on. A recent taxonomy names this gap directly, contrasting stateless video architectures with state-centric world-model theory. Genie is the hinge case, pairing a dynamics model with a latent action model and thereby sitting closer to the first family than the rest of its lineage, but the general division holds.

The report that produced this classification adopted a binary: controllable state versus rendered content only. The binary is correct and incomplete. Consider what the state-holding systems do not have. None of them maintains an append-only record of committed events from which the current state could be re-derived. None distinguishes an event that changed the world from a solver step that advanced it. None records who caused a change, whether the change was permitted, whether it was later contested, or whether a subsequent state is original or a repair. Persistence in these systems is a property of the current latent's ability to carry information forward, and when that ability is exhausted the world silently ceases to be the same world. The persistence-focused papers make the gap visible by working at it: WorldMem stores memory frames together with poses and timestamps, binding rendered evidence to structured addressable state; Genie 3 is described as holding roughly a minute of visual memory; Oasis acknowledges limited memory over long horizons. These are the beginnings of inscription, not instances of it. Poses and timestamps are not yet a historical ontology.

So the classification should be three-way. Rendered appearance x_t supports reconstruction and prediction of what an observer receives. Operative state s_t supports controlled transition. Historical inscription $H_{\leq t}$ supports identity, provenance, replay, and accountability. Written as a chain of non-identities,

$$z_t^{\text{image}} \not\equiv s_t^{\text{world}} \not\equiv H_{\leq t}^{\text{history}},$$

with the transitions

$$s_{t+1} \sim P(s_{t+1} \mid s_t, a_t), \qquad H_{t+1} = H_t \,\|\, e_{t+1},$$

and rendering treated as a function of the other two rather than as the thing itself. The frame-predicting systems occupy the first layer. The control-oriented systems occupy the first and second. Almost nothing in the learned-world-model literature occupies the third. The empirical findings reinforce the point: video diffusion models fail out of distribution by case-based imitation of nearest training examples, a failure that scaling does not fix, and physical understanding turns out to be unrelated to visual realism. Photorealistic recurrence is not evidence of a coherent world model, and successful re-rendering is not evidence of preserved identity.

One refinement is needed before going further. Within the third layer, the record itself and the historically active relations it supports are worth distinguishing, since they are consumed differently. Ownership, attribution, access rights, repair lineage, warrants, sanctions, and unresolved claims are all historical in origin, but a running system needs them as a current state, not as a log to be searched. The essay therefore works with four

roles within one architecture, summarized in Table 1. They are not four worlds and need not use four storage systems. They differ by function and, above all, by the kind of authority each carries.

Role	Primary object	Authority	Characteristic failure
Rendering	observer-relative appearance x_t	none beyond presentation	visual continuity mistaken for world continuity
Operative state	fields, objects, variables M_t	provisional dynamics	snapshot treated as complete identity
Provenance state	ownership, permissions, attributions Q_t	certified historical projection	context recomputed informally, or lost
Committed history	typed event DAG $H_{\leq t}$	identity and admissibility	past silently patched after divergence

Table 1: Four roles within one architecture. Only the last is authoritative over identity; the two middle rows are certified folds over it.

3 Why state is not enough

The argument that a third layer is required, rather than merely useful, runs through the observer. Suppose rendering is a function of state alone, $x_t = R(s_t, \omega_t)$, where ω_t parameterizes the observer. Then two worlds with identical states must render identically to identical observers. But this is false of any world in which objects have histories that matter. A wall that was built by a particular person, damaged in a particular event, repaired with a particular authorization, and owned under particular terms is encountered differently from a wall that merely shares its current geometry and texture. The difference is not always visible in the field; the repair may be seamless, the ownership invisible. Yet a world in which the difference is unavailable to any observer under any circumstances is a world in which the wall is not the same wall in any sense that survives the next regeneration.

This can be stated without appeal to human perception. Let two worlds have the same operative state, $M_t^{(1)} = M_t^{(2)}$. If the worlds are the same, then every admissible future of one is an admissible future of the other. But admissibility depends on permissions, and permissions depend on history: who may alter the wall depends on who owns it, which depends on how it was acquired. Two worlds with identical fields and different histories therefore have different admissible futures, and a world is at least its admissible futures. So identity of state does not imply identity of world:

$$M_t^{(1)} = M_t^{(2)} \not\Rightarrow W_t^{(1)} = W_t^{(2)}.$$

This is not a failure of Markov completeness in the physical dynamics. The field may well

be Markov with respect to its own evolution. It means that M_t is the operative physical state and not the total ontological state, and that the total state must include the history:

$$W_t = (M_t, H_{\leq t}).$$

The point generalizes the argument of *Inscription Before Rendering*, which showed that a preserved artifact functions as a witness against later authority only when provenance, recoverability, redundancy, and recurrence hold together, and that a system can preserve appearance while severing every one of them. A generative world that keeps state but not history has done exactly this internally: it can regenerate the wall indefinitely while having lost the wall.

Computational mechanics makes the same point from the other direction and thereby sharpens it. A causal state, in Crutchfield's sense, is an equivalence class of histories that share the same distribution over futures; it is the minimal sufficient statistic of the past for prediction. Two histories in the same causal state are, for every predictive purpose, interchangeable. The claim of this essay is that predictive interchangeability is not identity. Two walls whose histories fall in the same causal state will behave identically under every future action, and are nonetheless different walls if one was built by the person who owns it and the other was seized. The causal state is exactly the right object for M_t, which is why the control-oriented world models are right to seek it. It is exactly the wrong object for W_t, because it discards by construction the part of history that does no predictive work and all the identity-bearing work.

4 Two designs that fail

Once history is admitted as a layer, the question is what authority it has, and there are two designs that answer badly in opposite directions.

The first is state-primary. The field is the world; history is a log written as a side effect of commits. The update

$$M_{t+1} = \Pi_C\big(M_t + \Delta\psi\big)$$

is performed, and afterward an event describing it is appended. This is efficient and natural for physics-style dynamics, and it is the design the persistent-world book adopted in its drafted chapters. Its defect is structural. The log is a witness to the world rather than constitutive of it, and nothing prevents the two from diverging. A snapshot can be edited; a log entry can be lost; a replay can disagree with the stored state and the stored state wins by default because it is what the system runs on. This is precisely the failure mode that platforms exhibit at scale: a mutable present decorated with a history that has no authority

over it. The design reproduces, inside a single system, the separation of rendering from inscription that makes the recent past unrecoverable on a social feed.

The second is naively log-primary. The append-only history is the world, and the state is a cache derived by folding the history from an initial condition. This is event sourcing, and it is the design that Spherepop OS commits to and that Haplopraxis implements at the level of its game mechanics, where a flare, once fired, never disappears and becomes a permanent trace that later players encounter. The design has the virtues the first lacks: replay, provenance, and the impossibility of silent revision come for free. Its defect appears when the world contains continuous fields. If every numerical integration step is an ontologically constitutive event, then the history records solver arithmetic rather than meaningful commitment, the log grows with the timestep rather than with what happened, and the distinction between a fluctuation and an event is lost. The design confuses physical evolution with historical commitment.

Each design gets one thing right. The state-primary design is right that dynamics between meaningful events are provisional and should be cheap. The log-primary design is right that whatever has become part of the world's identity must be recorded in a form that nothing else can overrule. The resolution is to keep both, separated by scale.

5 The principle

History is authoritative at commitment boundaries; state is authoritative between them.

The principle assigns authority by temporal scale rather than by layer. Within an interval between commitments, the field evolves under its own dynamics and the state is authoritative: it may be integrated, refined, re-integrated at a different resolution, or discarded, and the history has nothing to say about any of this. At a commitment boundary, the provisional state is collapsed into an event, the event is appended to the history, and from that point the history is authoritative: the committed state is whatever the history folds to, and any cached state that disagrees is wrong.

This is not a compromise between two ontologies. It is a claim about what kind of thing each layer is. The field is the world's continuously evolving operative condition. The history is the world's committed identity. They are not peers, and neither reduces to the other. The field may outrun the history provisionally, since dynamics happen before they are committed. It may never contradict the history silently, since anything committed has authority over any state that claims otherwise.

6 Three temporal scales

The principle produces three scales, and keeping them distinct is most of the work. Between commitments the field evolves provisionally:

$$\widetilde{M}_{\tau+\delta\tau} = F_{\delta\tau}\big(\widetilde{M}_\tau; \theta, \xi\big),$$

where θ parameterizes the dynamics, ξ is whatever stochastic forcing the solver draws, and the tilde marks the state as provisional. Every step of this evolution is disposable numerical work. It has the same ontological status as a rendered frame: it can be recomputed and nothing depends on which particular computation produced it.

A commitment occurs when one of a small number of conditions holds. An admissibility boundary is crossed, so that the field's provisional trajectory has reached a region where the constraint manifold C intervenes. An identity-bearing distinction is created, so that something now exists in the world that did not before, or something that existed has ceased. An irreversible interaction takes place between agents or between an agent and the field. Or an observation is attested as evidence that constrains what may subsequently occur. When any of these holds, the provisional state is collapsed:

$$e_{n+1} = \mathrm{COLLAPSE}\big(H_{\le n}, \widetilde{M}_\tau, C, \mathcal{O}\big), \qquad H_{\le n+1} = H_{\le n} \,\|\, e_{n+1},$$

where $\mathcal{O}$ carries whatever attested observations are part of the event. The index n counts commits; the index τ counts solver time. The two are not aligned, and the misalignment is the point.

Between these lies a middle scale. Not every solver step is a commit, but not every solver step is equally uninteresting either. A semantic event is a candidate change: a detection that a boundary is near, a proposal $\Delta\psi$ that has been generated but not yet projected and committed, an interaction that has begun but not yet resolved. Semantic events are where the system decides whether a commitment is warranted. They are the middle term:

$$\text{solver ticks} \;\longrightarrow\; \text{semantic events} \;\longrightarrow\; \text{committed history}.$$

Solver ticks are numerical work. Semantic events identify candidates. Commits determine what has become part of the world's identity. In Spherepop terms, not every fluctuation is a POP, and not every provisional interaction reaches COLLAPSE. The middle scale is exactly where the four Spherepop primitives operate: a candidate is popped from the provisional dynamics, may be refused, may be bound to an existing identity, and, if it survives, collapses into a committed event.

Commit frequency is task-relative and not a property of the solver. A collision, a transfer

of ownership, a public attestation, a branching decision, or an irreversible phase change may deserve a commit even when it falls inside a much finer numerical integration, and thousands of stable substeps may remain provisional. The history records the distinctions that matter to recurrence and accountability. It is not a transcript of computation, and a design that lets the integrator's step size set the commit rate has confused the two.

7 One history, several folds

The authoritative current state is not stored; it is derived. The physical state is a fold over the history from an initial condition:

$$M_t = \mathrm{fold}_{\mathrm{phys}}(H_{\le t}) = \mathrm{fold}\big(\mathrm{commit}_{\mathrm{phys}}, M_0, H_{\le t}\big).$$

This raises the obvious objection that a continuous field cannot be re-derived from a log of discrete events without re-running the solver, and re-running the solver is expensive and possibly nondeterministic. The objection is met by specifying what an event records. A field-level event does not record the arithmetic. It records a seed for ξ, the operator version that implements F, the boundary conditions in force, the admissibility constraint and projection rule applied at the commit, the interval covered, and a checkpoint hash of the resulting state. Folding then means re-running the recorded operator with the recorded inputs over the recorded interval and checking the result against the recorded hash. The fold reproduces the state, not the computation.

A snapshot is a stored result of a fold up to some commit. It is permitted, and for any practical system it is necessary, as a replay accelerator. But its status is fixed by the principle: a snapshot is a certified checkpoint inside history, an event that records a state together with the hash that certifies it, and not an independent present that can overrule the history it was derived from. If a snapshot and a fold disagree, the fold is authoritative and the disagreement is recorded. Concretely, a checkpoint binds a folded state to the history frontier it was folded from, together with everything a refold would need to check it:

$$C_k = \big(\mathrm{frontier}_k,\ M_k,\ Q_k,\ \mathrm{versions}_k,\ \mathrm{seeds}_k,\ \mathrm{hash}_k\big).$$

A checkpoint without its frontier is a snapshot in the bad sense, a state with no stated history behind it.

The frontier is also what makes folding affordable. A fold is never run from M_0 in ordinary operation; it is run from the latest certified checkpoint whose frontier is an ancestor of the current one, over the events since. Invalidation is by frontier rather than by time: a cached fold is valid exactly when its frontier equals the history frontier for the region it

covers, and interest management, which already partitions a large world into regions with their own frontiers, gives each region its own checkpoint cadence. The cost of the architecture is therefore the cost of incremental folding from recent certified checkpoints, which is the cost any event-sourced system already pays, plus the cost of certifying checkpoints, which is what a snapshot-based system omits and should not.

The same history supports more than one fold, and this is what allows rendering to consult history without scanning it. Provenance, ownership, permissions, attribution, and the record of what has been attempted against an object are not physical quantities and do not belong in M_t. They are folded separately:

$$Q_t = \mathrm{fold}_{\mathrm{prov}}(H_{\le t}) = \mathrm{fold}\big(\mathrm{commit}_{\mathrm{prov}}, Q_0, H_{\le t}\big).$$

The two folds consume the same events with different commit functions. A physical event updates M_t and may update Q_t; an ownership transfer updates Q_t and leaves M_t untouched; a refused action, as discussed below, updates Q_t and not M_t. Rendering is then a pure function of two certified folds and an observer:

$$x_t = R(M_t, Q_t, \omega_t).$$

The renderer never reads the log. It reads two current states that the log produces, and the total world is one authoritative history together with a family of folds over it:

$$W_t = \big(H_{\le t};\ M_t,\ Q_t,\ \ldots\big).$$

Further folds may be added without changing the architecture: a fold that maintains spatial indices, a fold that maintains each agent's accessible region, a fold that maintains the set of unresolved semantic events. Each is a cache, each is subordinate to the log, and each can be discarded and rebuilt.

8 What counts as an event

The principle is only as good as the criterion for commitment, and the criterion has three parts that are easy to get wrong.

First, looking at the world does not create history. If the act of rendering committed an event, then history would grow with the number of observers and the frame rate rather than with what happened, and the system would be back at the naively log-primary design by another route. Rendering has the same status as a solver tick. An observation enters history only when it is attested, that is, when it is entered as evidence E_{t+1} that narrows

the admissible set of subsequent states or interpretations. This is the distinction *Epistemic Immutability* draws between a record and the constraint it later supports: the record is preserved, and a later observation does work only when it is registered as a constraint on what the record may be taken to mean. In a world engine the same distinction separates an observer who is watching from an observer who is testifying. Written out, the two operations have different signatures:

$$\begin{aligned} \text{looking:} \quad & R(M_t, Q_t, \omega_t) \;\to\; x_t \qquad \text{(no commit)} \\ \text{attesting:} \quad & (x_t, \text{witness}, \text{rule}) \;\to\; E_{t+1} \;\to\; \text{narrower admissible set.} \end{aligned}$$

Sensors may sample continuously while only selected, authenticated, or threshold-crossing measurements become attestations. A screenshot need not alter a world; a signed measurement used to authorize a repair does. The decisive property is not perceptual occurrence but constraining work. Attesting is an event. Looking is not.

This raises the question of who may attest, and the architecture answers it without a new mechanism. Standing to attest is a historical relation, and so it lives in Q_t: which witnesses are warranted for which kinds of evidence is a fold over past events, revised by the same typed vocabulary as everything else. An attestation therefore passes through the same gate as any other candidate. One offered without warrant is a refused attestation, and is logged as such; one offered by a witness whose warrant was later withdrawn remains in the history with its warrant as it stood at the time. Conflicting attestations from different witnesses are not adjudicated by overwrite. They produce an unresolved claim in Q_t, a recorded conflict in the sense of an event structure, which later evidence may constrain but which no fold may quietly settle by keeping one and dropping the other. And in a distributed world every attestation names the causal frontier its witness held when it attested, so that two witnesses reporting incompatible things from different cuts through the history have not contradicted each other. They have recorded a divergence between cuts, which is the same kind of event as a replay discrepancy and is handled the same way.

Second, rejected actions enter history. A candidate change that reaches an admissibility boundary and fails there does not alter M_t, and so $\text{fold}_{\text{phys}}$ ignores it. But it may alter provenance, permissions, or the record of attempts, and so $\text{fold}_{\text{prov}}$ need not ignore it. A world in which refusals leave no trace cannot distinguish an object that has never been contested from one that has been contested repeatedly and successfully defended, and that distinction matters to what may happen next. The world remembers what was attempted against it. The admissibility-gated training loop that logs each step as admitted, rejected, or repaired is the same discipline applied to a learning system; here it is applied to a world. The event vocabulary should therefore be typed, with at least the outcomes proposed,

admitted, refused, repaired, attested, checkpointed, and replay-discrepant. A candidate passes through a gate that reads all three current objects,

$$c \longrightarrow A\big(c \mid M_t, Q_t, H_{\le t}\big) \longrightarrow \{\text{admit, refuse, repair}\},$$

and repair deserves the same visibility as refusal. Replacing a failed proposal with a corrected one must not erase the distinction between what was first attempted and what was finally admitted; otherwise the history records only the world's successes, and a history of successes cannot explain why the world's constraints are where they are.

Third, replay divergence is itself an event. Nondeterministic solvers, hardware differences, and library updates will eventually produce a fold that fails its checkpoint hash. The tempting engineering response is to repair the checkpoint. The principle forbids it. The discrepancy is appended as a new record that names the event it disagrees with and the result the refold produced; the earlier record is never corrected in place. This is the discipline stated for provenance generally: preserve the commitment, append the correction, record the discrepancy, and correct the continuation rather than the record. A world that patches its own checkpoints has made its present authoritative over its history, and the principle has been abandoned at exactly the moment it was needed. What the policy separates is epistemic immutability from operational rigidity. The system may improve its solver, restore service, or adopt a better reconstruction, and may certify a new checkpoint that supersedes the failed one. What it may not do is present the improved result as though it had always been the original.

Replay itself must be graded rather than treated as one claim. Bitwise recurrence, numerical recurrence within a tolerance policy, semantic recurrence of the committed distinctions, causal recurrence of the dependency structure, and provenance-preserving recurrence of ownership and attribution are five different equivalences, and a discrepancy at one level does not imply a discrepancy at the others. A refold that differs in the last bits of a floating-point field while reproducing every committed event and every attribution has failed bitwise recurrence and nothing else. Collapsing the five into a single pass-or-fail replay claim is how a system ends up either refusing to certify anything or certifying too much.

9 Distributed history

In a multiplayer world the history is not a sequence. Agents act concurrently, their actions are committed at different sites, and the order in which independent commits arrive at any one site is not the order in which they were made. Interest management gives

each participant a partial and sometimes stale projection of the world; network delay produces concurrent proposals; rollback and reconciliation revise provisional states after the fact. The distributed-systems literature settled the relevant limits long ago. State-machine replication shows that deterministic operations applied in an agreed order define a replicated service, which is the multi-site form of the fold. Jefferson's Virtual Time supplies rollback, antimessages, and fossil collection, which is the multi-site form of the provisional interval: what has not yet been committed may be undone. And Chandy and Lamport's distributed-snapshot result marks an epistemic boundary that the principle must respect: a globally consistent snapshot need not correspond to any instantaneous state that every process jointly occupied. There is, in general, no global present. There is only a consistent cut through the history, which is another way of saying that history is primary and the present is a fold.

History is therefore a directed acyclic graph of commits, each naming the causal frontier it depends on, and the fold must be well-defined over a partial order.

The condition for this is already required elsewhere. The residue algebra of the persistent-world program specifies that the residue operator $\oplus$ commutes over causally independent commits and need not commute over causally dependent ones:

$$a \perp b \implies R_a \oplus R_b = R_b \oplus R_a.$$

This was introduced to make a conservation theorem go through. It is also exactly the condition under which independent commits may be reordered or merged without changing the fold, while causally dependent commits may not. Distributed folding is therefore not an additional mechanism layered on top of the architecture. It is a consequence of the residue algebra as stated. A site that receives independent commits in a different order folds to the same state; a site that receives a dependent commit before its predecessor must wait, or must record the out-of-order arrival as a semantic event awaiting resolution. Either way the history remains authoritative and the folds remain caches. The condition cuts in both directions. When two actions compete for one object, draw on the same scarce resource, or alter each other's permissions, the system must order, refuse, branch, or repair them. It may not declare them independent for convenience, and independence should be certified from the recorded dependencies rather than assumed from the absence of an obvious conflict.

10 Precedents and differences

None of this is without precedent. Event sourcing in software architecture stores an append-only sequence of events and derives current state by replay, with snapshots as accelerators.

Git stores an immutable directed acyclic graph of commits, derives a working tree by checkout, and distinguishes the graph, which never changes, from the branch references that select a history through it. Append-only ledgers make the same commitment in a financial setting. The persistent-world program has drawn on all three, and the principle stated here is recognizably in their family.

The differences are what make the world case harder. Event-sourced systems assume that replay is deterministic and cheap; a continuous field with stochastic forcing and a numerical solver is neither, which is why events must record seeds and operator versions and why checkpoint hashes are not optional. Git assumes that a commit is a discrete change to discrete content; a field evolves continuously between commits, which is why the three-scale distinction is needed and why the criterion for what counts as an event cannot be borrowed. And none of the precedents distinguishes between an observer who watches and an observer who attests, because none of them has observers inside the system whose observations might or might not constrain it. The world case inherits the discipline of its precedents and adds three requirements: recorded reproducibility, a commitment criterion, and attestation.

11 What the surrounding literature establishes

The layering principle is a construction, not a discovery of something already in the literature, but each of its parts has a home, and it is worth saying where. Formal transition systems supply the second layer with rigor. Structural operational semantics defines the admissible transitions as exactly those derivable from rules, which is what Π_C enforces. Partially observable Markov decision processes supply state-dependent action sets, belief states, and noninvertible transition kernels, so that admissibility and partial observation are both already first-class. Viability theory studies the set of initial states from which a constraint can continue to be satisfied, and reachability theory the set of states attainable under controlled dynamics; together they describe the future volume that admissibility carves out, and one of the open questions below asks whether that volume can be defined over (M_t, Q_t) jointly rather than over M_t alone.

Two formalisms come unusually close to the third layer. Ceptre, a language for modeling generative interactive systems, represents world state as a multiset of resources, transitions as rules that consume and produce them, and permanent facts as a separately marked class; its execution traces carry causal-dependency structure. That is a working instance of a typed event history with a distinguished irreversible fraction. Event structures, from concurrency theory, give irreversibility an even cleaner home: a configuration can only grow, and conflict between events permanently excludes alternatives. A refusal in the vocabulary above is a

recorded conflict; a committed event is one that has entered the configuration and can no longer leave it.

Artificial life and generative social simulation, from Schelling and Boids through Sugarscape, Tierra, Avida, Neural MMO, and generative agents, establish something the persistent-world program depends on methodologically: that local rules and per-agent state produce persistent global organization, and that a simulated world is therefore an instrument for discovering consequences not transparent in its rules. This licenses the use of computational worlds as epistemic laboratories. It does not license, and the present essay does not make, the metaphysical claim that physical reality is a simulation.

Finally, the operational formalisms of quantum theory, generalized probabilistic theories, quantum instruments, categorical quantum mechanics, monotone information metrics, and Markov categories, provide developed languages for states, transformations, interventions, distinguishability, and composition. They establish no quantum account of game engines or persistent worlds, and none is claimed. Their legitimate role is comparative: quantum theory is a highly developed operational geometry, and its methods show how a theory can define admissible states and transformations without committing to a hidden classical substrate. Of these, Markov categories are the strongest bridge to the ordinary stochastic transition systems that a world engine actually runs.

12 The observer's architecture

The precedents above are architectures of the record. There is also a precedent for the other side of the rendering equation, the observer ω_t, and it is worth placing against the principle because it comes from an unusual direction. De Gyurky and Tarbell's *The Autonomous System* (2014) sets out to build an autonomous software agent by translating the cognitive architecture of Kant, Hegel, and Schopenhauer into system-engineering terms, on the stated premise that the fundamental science in computer science is the science of thought and that software is an abstraction of the human thought system. The result is a constellation of interfacing subsystems, each given a chapter: Will, Reason, Intellect, Presentation, Understanding, Sensory, Decision, and Thought. The Sensory system is divided into a Phenomenon subsystem and a treatment of the noumenon, and the Presentation system carries Schopenhauer's *Vorstellung* directly into the design. The authors are explicit that the philosophers did not write with implementation in mind and that the work of the book is to build a model, a viewable object rather than a text, on the ground that text conveys only half of what an engineering design requires.

What is striking is how closely the constellation reproduces the first two of the three layers argued for here, on the side of the agent rather than the world. Presentation is

appearance: the world as rendered to a subject. The noumenon, the thing in itself that the Sensory system is built to work with, is the state beneath appearance that presentation is derived from and does not exhaust. The Kantian and Schopenhauerian inheritance guarantees that these will be kept apart, since the whole point of that tradition is that presentation is not the thing presented. In the notation of this essay, the Presentation system is what receives x_t, and the noumenal machinery is the agent's model of M_t. An architecture that separates these is already ahead of any frame-predicting world model, which has only the first.

The question is whether the constellation has a third layer, and the book does name a place for it. The Intellect system is described as the librarian of the mind: its custodian, repository, and producer of knowledge, with three subsystems, Abstraction, Experience, and Knowledge. It receives the two kinds of information the design recognizes, sorts and stores them, builds databases from them, and continuously expands, updates, and modifies those databases through interaction with the rest of the constellation. Its primary product is what the authors call practical knowledge, which they define carefully. Abstract knowledge is acquired secondhand, through reading, lectures, and viewing without participation. Experiential knowledge is acquired through personal or group participation, repetition, and practice. Practical knowledge is what results when one has been validated by the other: the abstract confirmed by experience, or experience confirmed by the abstract. The book is severe about the failure case. An Intellect stocked only with abstract data, however extensive, holds academic knowledge and little else.

This taxonomy is closer to the present argument than its vocabulary suggests. Abstract knowledge is a declared record: what someone else has asserted, received without the receiver having witnessed it. Experiential knowledge is a witnessed record. Practical knowledge is the declared constrained by the witnessed, or the witnessed given its place by the declared. That is the same structure *Epistemic Immutability* formalized as constraint without erasure: a commitment is preserved, later evidence narrows what it may be taken to mean, and neither is discarded in favor of the other. The Intellect's validation step is, in this reading, an attestation. A piece of abstract knowledge becomes practical when an experience is entered as evidence that constrains it, and the book's insistence that the two kinds must not be confused or substituted for one another is the insistence that declared and witnessed history are different things.

Where the design falls short of the principle is exactly where the essay's test bites. The librarian's databases are continuously updated and modified in place. Validation produces practical knowledge, but the design as described does not preserve the unvalidated abstract claim, the validating experience, and the discrepancy between them as separate committed records that a later state cannot silently overwrite. It keeps the result. A preference structure

or knowledge base that is updated in place is a fold without a log. The Intellect as described is a custodian with a current state and no authoritative history behind it, which is the state-primary design with the roles moved from world to agent. Schopenhauer supplies an unintended commentary on this: the book follows him in locating the Intellect in the brain, so that it dies with the organ, while the Will, the thing in itself, does not. A librarian whose entire holding perishes with its host has custody and no redundancy. The knowledge it has produced is exactly as durable as the one repository that holds it, which is the condition *Inscription Before Rendering* identified as possession mistaken for preservation.

The point is not to fault a design of this vintage for lacking a distinction drawn here. It is that the distinction applies on both sides of $R(M_t, Q_t, \omega_t)$, and that the book has already done most of the work of establishing it on the observer's side. It separates appearance from state. It separates declared from witnessed knowledge. It makes validation, which is attestation, the central productive act of the mind's repository. What it does not do is give that repository a history with authority over its current contents. The world needs a history that its state cannot overrule; the observer needs one too, if the observer is to attest rather than merely look. An agent that enters an observation into the world's history as a constraint is only as reliable as its own record of what it observed, and an Intellect that updates its databases in place cannot produce that record on demand. Schopenhauer's epigraph, that a system of thoughts must always have an architectural structure, is right, and the structure has three layers on the observer's side for the same reason it has three on the world's.

It does not follow that the world must audit the observer's ledger. The world's history records attestations, each with a witness identity, a warrant, and a frontier; it does not record, and cannot verify, the internal record from which the witness attested. If that internal record is corrupted or disputed, what the world can do is what it does with any commitment later found wanting: preserve the attestation as made, append the evidence against it, and let the witness's standing in Q_t carry the consequence. An observer whose attestations are repeatedly contradicted loses warrant; an observer whose ledger is shown to have been altered loses it faster. The adjudication is not of the observer's memory but of the observer's standing, and it is done by the same folds and the same discipline as everything else.

13 The wall's two memories

The wall that runs through the persistent-world program can now be described exactly. It has two memories, and the principle separates them.

Material memory is what the field carries: cracks, discoloration, stress, the accumulated deformation that the dynamics have written into M_t. This memory is physical, it is Markov

with respect to the field's own evolution, and it is what a state-holding world model can in principle preserve. Historical memory is who built the wall, what happened at it, whether a given repair is original or a later intervention, who owns it, and who may alter it. This memory lives in $H_{\leq t}$ and is read through Q_t. It is not physical, it is not recoverable from the field, and no amount of state fidelity will supply it.

Rendering consults both because an observer does not perceive instantaneous matter. An observer encounters an historically situated object, and the encounter is different when the history is different even where the matter is the same. This is why R takes Q_t as an argument, and it is why a world engine that renders from M_t alone will produce walls that are indistinguishable from one another and therefore, over any long horizon, indistinguishable from their own regenerations.

The two memories are separate folds, not separate worlds, and they interact. A repair is one event consumed by both commit functions: it changes the field, and it changes the repair lineage. A crack is a physical fact that may also become an attribution once someone attests to what caused it. The coupling runs in definite directions. Q_t constrains what may happen to M_t, since the gate reads permissions before admitting a physical change; M_t supplies the evidence from which attestations into Q_t are made. Neither fold is derived from the other. Both are derived from the history, and the history is where their consistency is guaranteed, because a single event cannot be admitted to one fold and refused by the other; it is admitted or refused once, and each fold consumes the outcome.

14 Persistence is engineered

The most useful empirical finding in the learned-world-model literature is that persistence is always engineered and never free. Systems that persist better do so by binding rendered evidence to addressable state, by storing poses and timestamps alongside frames, by extending the horizon of what the latent can carry. None of them gets persistence from scaling the renderer, and the finding that physical understanding is unrelated to visual realism says why: appearance is the wrong layer to engineer persistence in.

The principle stated here is the same finding carried to its conclusion. Persistence is engineered at the commitment boundary, in the criterion for what becomes an event and in the discipline that makes the history authoritative once it does. A world that engineers persistence in the state alone will persist as long as the state carries what is needed and no longer. A world that engineers it in the history persists as long as the history is preserved and can be folded, which is a condition on custody and redundancy rather than on model capacity. The three layers are not three degrees of the same thing. Rendering predicts appearances. State constrains transitions. Inscription preserves worlds.

Stated as a hierarchy of increasingly strong claims about a system: renderability means an appearance can be generated; controllability means actions produce structured transitions; recurrence means a state can be reproduced under declared conditions; provenance means the legitimacy and lineage of the state can be recovered; persistence means identity survives change because commitments and refusals remain addressable. The frame-predicting engines have the first. The control-oriented models have the first two. A checkpointed engine with a deterministic solver has the third. Nothing in the surveyed literature has the last two, and the last two are what the word persistent should mean.

15 Consequences

For a world engine, the consequences are concrete. The event schema must record what a fold needs to reproduce a state, not what a solver did to produce it. Snapshots must be typed as certified checkpoints and must fail loudly when a refold disagrees. Rendering must be factored as a function of folds, and the folds must be enumerable so that new derived views can be added without touching the history. The commitment criterion must be explicit, must distinguish attestation from observation, and must admit refusals as events. And the history must be a graph from the start, with the residue algebra's commutativity condition enforced at merge.

These reduce to a short set of design rules. Keep rendering pure: a render call may read certified folds and may not write history because a view was produced. Commit attestations, not perceptions: evidence enters history only when an identified procedure uses it to constrain admissibility. Use typed events, distinguishing proposals, admissions, refusals, repairs, attestations, checkpoints, and replay discrepancies. Derive current authority: physical and provenance states are folds or certified checkpoints tied to explicit history frontiers. Make refusal durable: a failed proposal may leave physical state unchanged while still affecting the audit and permission folds. Never repair the past in place: a corrected reconstruction or checkpoint is appended with its relation to the one that failed. Record causal parents so that independent events can commute and conflicting ones cannot. And separate recurrence levels, so that bitwise, numerical, semantic, causal, and provenance equivalence are never collapsed into one replay claim.

For the persistent-world book, the consequence is a revision rather than a rewrite. The fields describe the world's continuously evolving operative condition. The append-only history defines its committed identity. Chapters that speak of M_t as the world should be read, and where necessary revised, so that M_t is the physical fold of the world and W_t is the world. The chapters on the write operator, projection, and commit need the three-scale distinction inserted between projection and commit. The chapter on causal residue should

recognize that part of what it currently stores in the field belongs to the provenance fold. The treatment of identity as a persistent orbit should say that identity is a property of W_t and not of M_t alone. The multiplayer chapter should state the graph claim. And the wall, wherever it appears, should be given its two memories by name.

16 Open questions

The construction leaves a number of questions open, and they are better stated than hidden.

1. What is the smallest event vocabulary sufficient to reconstruct both the operative and the provenance fold without making every solver step historical?
2. Which admissibility predicates can be evaluated against current folds, and which require direct reference to an historical witness, so that the folds alone are provably insufficient?
3. When may two events commute in a distributed history, and how can independence be certified from recorded dependencies rather than assumed?
4. How should replay equivalence be graded across bitwise, numerical, semantic, causal, and provenance-preserving recurrence, and which grade should a certification claim?
5. Can a learned world model acquire persistent identity without an explicit event ledger, or does apparent persistence merely relocate the ledger into opaque memory where it cannot be audited?
6. How should a refusal affect future permissions without allowing adversarial attempts to inflate or poison the authoritative history?
7. Can viability kernels be defined over the combined folded state (M_t, Q_t), making institutional and provenance constraints part of the reachable future volume rather than an overlay on it?
8. Which information metric best measures the loss incurred when a historically distinct world is collapsed into an extensionally identical snapshot?

17 Conclusion

The learned-world-model literature has correctly identified that a world needs more than appearance. It has not yet identified that a world needs more than state. The third layer, historical inscription, is what makes a world the same world over a horizon longer than its

state can carry, and it is absent from nearly every system in the field, including the ones with explicit latent dynamics. Admitting the layer raises the question of authority, and the answer is neither to subordinate history to state nor to make every computation an event. History is authoritative at commitment boundaries; state is authoritative between them. Everything else in this essay, the three scales, the family of folds, the attestation criterion, the recording of refusals and discrepancies, the graph-structured history, follows from holding that line. A world built this way can still be rendered, still be simulated, still be scaled. What it cannot do is quietly stop being itself. A world persists not because it renders the same scene again, but because it can state what was committed, what was refused, and why the present is entitled to count as a continuation of its past.

Selected references

Aubin, J.-P. (1991). *Viability Theory*. Birkhäuser.

Barrett, J. (2007). Information processing in generalized probabilistic theories. *Physical Review A*, 75, 032304.

Chandy, K. M., & Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1), 63–75.

Coecke, B., & Kissinger, A. (2017). *Picturing Quantum Processes*. Cambridge University Press.

Crutchfield, J. P., & Young, K. (1989). Inferring statistical complexity. *Physical Review Letters*, 63, 105–108.

de Gyurky, S. M., & Tarbell, M. A. (2014). *The Autonomous System: A Foundational Synthesis of the Sciences of the Mind*. Wiley.

Epstein, J. M. (1999). Agent-based computational models and generative social science. *Complexity*, 4(5), 41–60.

Fritz, T. (2020). A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370, 107239.

Ha, D., & Schmidhuber, J. (2018). World Models. arXiv:1803.10122.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). Learning latent dynamics for planning from pixels. *Proceedings of ICML*.

Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2020). Dream to control: Learning behaviors by latent imagination. *Proceedings of ICLR*.

Jefferson, D. R. (1985). Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3), 404–425.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2), 99–134.

Martens, C. (2015). Ceptre: A language for modeling generative interactive systems. *Proceedings of*

AIIDE.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach. *ACM Computing Surveys*, 22(4), 299–319.

Schrittwieser, J., et al. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588, 604–609.

Shalizi, C. R., & Crutchfield, J. P. (2001). Computational mechanics: Pattern and prediction, structure and simplicity. *Journal of Statistical Physics*, 104, 817–879.

Winsberg, E. (2010). *Science in the Age of Computer Simulation*. University of Chicago Press.