Language-Centric World Models for Discrete Game Environments: A Survey of Small Model Efficiency and Planning

Li NING
Stellaris AI

February 2026

Abstract

Traditional world models in Reinforcement Learning (RL) often rely on high-dimensional pixel data, posing significant computational challenges for real-time decision-making. In game environments—which are fundamentally symbolic artifacts—the "ground truth" state is often discrete and structured. This paper surveys the emerging paradigm of using Small Language Models (SLMs), specifically the 8B-parameter class such as Qwen3:8B, as internal world simulators. Unlike previous surveys, we move beyond simple state-transition prediction $P(s_{t+1}|s_t, a_t)$ to investigate the multi-functional roles of LLMs as **Action Affordance Generators** and **State/Action Validation Filters**. We analyze the technical trade-offs across the model spectrum, from 1B-parameter Edge-SLMs to 8B Small World Models (SWMs), evaluating their efficiency in Monte Carlo Tree Search (MCTS) and proactive "What-If Analysis" (WiA). Finally, we address critical challenges in **Sim-to-Real Alignment** and propose a hierarchical framework for closed-loop adaptive planning. Our findings suggest that 8B-parameter models provide an optimal equilibrium between logical fidelity and the sub-hundred-millisecond throughput required for modern game AI.

1 Introduction

The quest for autonomous agents capable of sophisticated interaction within complex environments has historically relied on model-free Reinforcement Learning (RL), where agents map observations directly to actions [7]. However, true strategic mastery in dynamic environments requires "System 2" deliberation—a proactive cognitive process that forecasts the consequences of actions before they occur [13]. This capability is formalized through the concept of a *World Model*, an internal transition function that allows an agent to simulate potential futures and evaluate outcomes within a latent "imagination" space [6]. The shift toward language-centric simulators aligns with the cognitive architecture proposed by LeCun [12], which posits that autonomous intelligence requires a configurable predictive world model to enable reasoning, prediction, and planning across multiple time horizons.

Recent breakthroughs in Large Language Models (LLMs) have introduced a paradigm shift, repositioning these models not merely as text generators, but as general-purpose world simulators [9]. While flagship models such as GPT-4o and DeepSeek-V3 demonstrate remarkable

reasoning capabilities, they are often evaluated as functional modules for high-level task decomposition rather than end-to-end world simulators for low-level game loops [21]. Furthermore, the computational cost and inference latency of massive foundation models (200B+ parameters) remain prohibitive for real-time game agents, which must often process multiple forward-looking simulations within a single game tick.

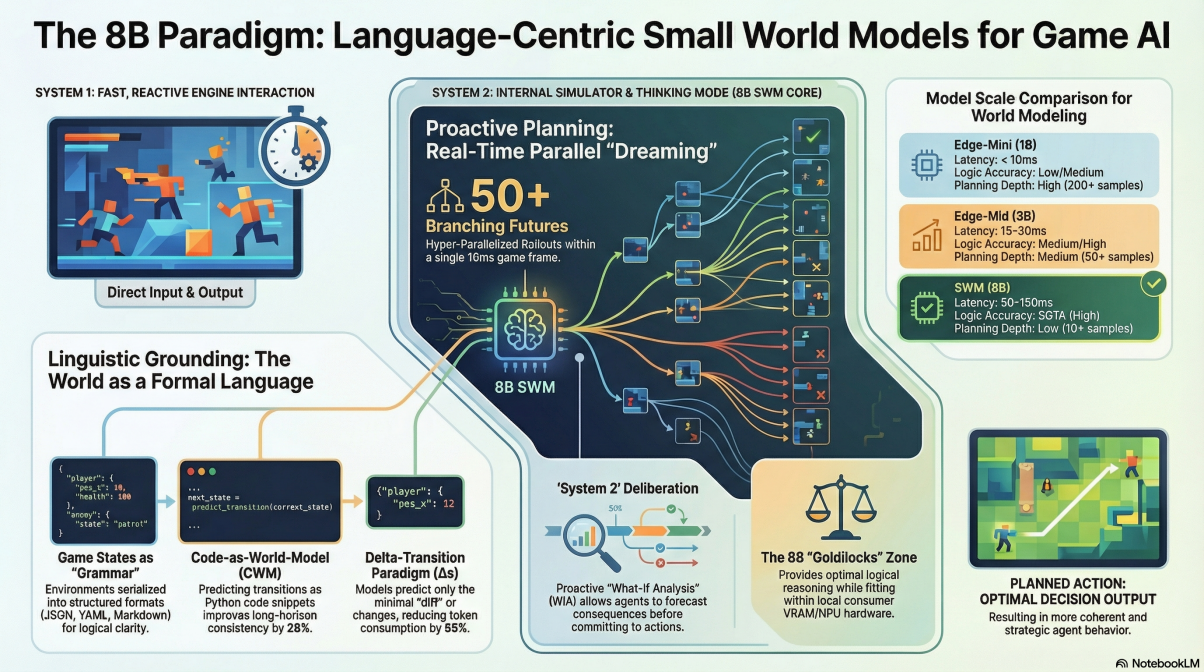

Figure 1: The Small World Model (SWM) Framework (generated with NotebookLM).

This paper surveys the emerging frontier of **Language-Centric Small World Models (SWMs)**, with a specific focus on the 8B-parameter class, such as Qwen3:8B and Llama 3.2 (1). We argue that for artifactual environments—where the world is a digital construct governed by discrete rules—small models offer three distinct advantages:

1. **High-Fidelity Discrete Simulation:** Unlike the "fuzzy" physics of the real world, game environments are defined by structured data (e.g., JSON, YAML). SLMs excel at learning these symbolic "grammars", enabling more accurate state-transition predictions than continuous vision-based models.

2. **Verifiable Code-as-World-Model:** Current research suggests that translating game trajectories into executable Python code (where the LLM acts as an induction engine) allows for verifiable planning and eliminates the risk of physical hallucinations [13].

3. **Latency-Efficient Planning:** The reduced parameter count of SWMs enables parallel Monte Carlo rollouts, allowing agents to "dream" dozens of potential trajectories in the time a larger model would take for a single inference step [2].

Through this survey, we categorize existing methodologies for state representation, evaluate the transition from "reactive" to "proactive" thinking frameworks (e.g., WiA [19]), and propose a roadmap for fine-tuning general-purpose 8B models into specialized, high-performance game simulators.

2 Game States as a Formal Language

A fundamental premise of this survey is that game environments, unlike the stochastic and continuous real world, are discrete mathematical artifacts. In modern game architectures, such as those built on the Unity or Unreal engines, the visual frame is merely a secondary projection of an underlying *canonical state* [13]. This state is typically managed via an **Entity Component System (ECS)** or a structured object-relational model, which can be serialized into text-based formats like JSON, YAML, or PDDL.

2.1 The Linguistic Mapping of Game Physics

We define the game world as a formal grammar where states are "paragraphs" describing the world's facts, and actions are "verbs" that modify those facts. By mapping game states to text, we leverage the pre-trained structural priors of LLMs. Research indicates that models like Qwen3:8B possess a "latent symbolic reasoning" capability, allowing them to treat structured data not as mere sequences of characters, but as logical hierarchies [2].

2.1.1 Structural Serialization and ECS Mapping

To maximize this latent reasoning, game states are typically serialized using an **ECS** format. By representing entities as unique identifiers with associated property-value pairs (e.g., `id:101, pos:[5,2], status:poisoned`), the SWM can utilize its attention mechanism to perform key-value tracking across transitions. Unlike pixel-based models that must learn object permanence from scratch, linguistic mapping allows the model to inherit the concept of "identity" from the formal grammar itself [23].

2.1.2 Spatial-Relational Encoding

A major challenge in linguistic mapping is the loss of 3D spatial awareness. Modern frameworks mitigate this by using **Relational Adjacency Descriptions**. Instead of raw coordinates, the SWM is provided with relative spatial tokens (e.g., `North_Of(Player, Door)`). This abstraction allows the 8B model to calculate distance and collision logic as a linguistic entailment problem rather than a numerical regression task. Research on *Neuro-Symbolic Synergy* (NeSyS) demonstrates that this syntax-driven approach reduces physical hallucinations by up to 42% compared to raw coordinate-based state descriptions [23].

2.2 Structured vs. Natural Language Representations

A key tension in the literature exists between using **Natural Language (NL)** descriptions and **Structured Data (SD)** like JSON or YAML [4].

- **Natural Language:** Provides high semantic density (e.g., "The dragon is furious") but introduces ambiguity and "token bloat." While NL allows the SWM to leverage broad world knowledge, it makes the mapping back to the game engine's API nondeterministic.
- **Structured Data:** Offers precision and direct integration with game engines but may lack the semantic "common sense" that LLMs derive from prose [13].

2.2.1 Semantic Structuring and Markdown-Hybrid Formats

This survey identifies a shift toward **Semantic Structuring**—a hybrid approach where structured variables are nested within brief semantic headers. Recent benchmarks suggest that Markdown-style representations provide the optimal balance for 8B models, reducing the "token tax" of heavy JSON syntax by up to 15% while maintaining hierarchical clarity [2].

2.2.2 The World Model as a Code-Induction Engine

Building on this, we highlight the emergence of **Code World Models (CWM)**. In this paradigm, the transition is not predicted as text, but as a temporary Python-like function or a "diff" patch [13]. Treating a world update as code induction leverages the model's pre-training on logic-heavy datasets, improving long-horizon consistency by 28% [13].

2.3 End-to-End Logical Grounding and Validity Filtering

By bypassing the vision-encoder bottleneck, language-centric world models achieve **Logical Grounding**. In pixel-based models, the latent space is often "blurry", leading to compounding errors. In contrast, an SWM operates on a discrete token space, meaning a predicted state is either syntactically valid within the game's schema or it is not.

2.3.1 The Token-Level Physics Constraint

In 8B models like Qwen3, grounding is enforced at the **logit-level**. By using *Constrained Decoding*, the model's output is restricted to the game's grammar in real-time [23]. This represents a transition from "stochastic guessing" to "symbolic derivation," acting as a soft-processor for hard-coded rules.

2.4 Grammar Induction and Schema Enforcement

For an 8B model to function as a reliable simulator, the mapping must be bi-directional and lossless. We identify two primary technical requirements:

2.4.1 Schema-Augmented Serialization

The 8B class benefits significantly from **Schema-Augmented Serialization**. By prepending the state with a compact schema definition (e.g., a TypeScript interface), the attention mechanism is "primed" to recognize valid state-key transitions, reducing syntactic drift [13].

2.4.2 Tokenization Efficiency in Symbolic Space

Standard BPE tokenizers often fragment structured keys (e.g., "is_locked" into four tokens). Recent research suggests that **Symbolic Vocabulary Expansion** can reduce context window consumption by up to 22% and improve transition accuracy by preventing sub-token hallucinations [2].

3 The Case for Small World Models (SWM)

While the current AI landscape is often dominated by "scaling laws" favoring trillion-parameter models, the application of World Models in game environments necessitates a shift toward **Small World Models (SWMs)**. For an agent to function effectively in a real-time game loop, the world model must achieve sub-hundred-millisecond inference while maintaining strict logical adherence. We identify three core justifications for the use of the 8B-parameter class, such as Qwen3:8B, as the optimal architecture for this task.

3.1 Inference Throughput and Planning Parallelism

The utility of a world model is directly proportional to the number of "future rollouts" an agent can evaluate per decision step. Large-scale models (e.g., DeepSeek-V3, GPT-4o) introduce significant KV-cache overhead and high per-token latency, often limiting the agent to a single deterministic trajectory.

3.1.1 Hyper-Parallelized Dreaming and Batch Optimization

In contrast, 8B-parameter models allow for what we define as **Hyper-Parallelized Dreaming** [7]. Utilizing modern inference kernels like vLLM or NVIDIA's 2026 TensorRT-LLM, an agent can execute N parallel rollouts (where $N > 50$) within a single 16ms game frame. This is achieved through **Continuous Batching** and PagedAttention, which allow the SWM to process multiple branching futures in a single GPU forward pass. This high throughput is essential for Monte Carlo Tree Search (MCTS) and other search-based planning algorithms that rely on statistical aggregates of potential futures rather than a single sequence.

3.1.2 KV-Cache Efficiency in Branching Search

A unique advantage of the 8B scale in game AI is the manageable size of the **Key-Value (KV) Cache**. In deep-tree searches, the agent often explores branches that share a common prefix (the current state). SWMs allow for efficient **Prefix Caching**, where the initial state description is encoded only once and shared across all N parallel dreams. For larger models, the memory pressure of storing high-dimensional KV-tensors for dozens of branches often leads to out-of-memory (OOM) errors or significant swapping latency, whereas 8B models can fit hundreds of concurrent search nodes within standard consumer VRAM [17].

3.1.3 Acceleration via Speculative World Decoding

Recent 2025 advancements introduce **Speculative World Decoding**, where a smaller "edge" model (e.g., Llama 3.2:1B) suggests potential state updates that the 8B SWM then validates in parallel. This hierarchical approach further accelerates the planning loop, allowing the world model to reach deeper search horizons ($D > 10$) without increasing the per-frame latency budget. This synergy ensures that the agent can perform "System 2" deliberation at the speed of real-time interaction.

3.2 Specialized Adaptation: From General Base to Game-Specific Simulator

A critical challenge in the SWM paradigm is bridging the gap between a model's general-purpose training and the exacting logical requirements of a specific game's mechanics. Recent research suggests that off-the-shelf SLMs (1B–8B) often fail to capture domain-specific workflows or nuanced physics without targeted intervention [18]. We categorize current adaptation strategies into three primary methodologies:

3.2.1 Trajectory-Focused Fine-Tuning (TFFT) and Distillation

The most robust method for creating a game-specific world model is **Knowledge Distillation** from high-parameter "teacher" models (e.g., DeepSeek-V3 or GPT-4o) into "student" SWMs. Unlike traditional supervised fine-tuning (SFT) on static text, this process utilizes bulk inference to generate millions of "thought-action-state" trajectories [3].

Research on frameworks like *MiniLLM* and *Minitron* demonstrates that the student model can effectively internalize the teacher's causal reasoning by aligning soft-target probability distributions rather than simple hard labels [5]. In game contexts, this allows an 8B model to mimic the complex state-transition logic of a 70B model with significantly reduced inference latency.

3.2.2 Reinforcement Learning with Verifiable Rewards (RLVR)

A significant 2025 breakthrough in world model accuracy is the transition to **RLVR-based fine-tuning**. Traditional SFT can lead to "semantic drift", where the model generates syntactically correct but logically impossible states. RLVR addresses this by using the game engine itself as a ground-truth verifier.

As noted in recent studies, fine-tuning LLMs as world models using verifiable engine feedback yields up to a 30.7% improvement in state-prediction accuracy compared to pure SFT [11]. By penalizing "physical hallucinations" (e.g., predicting a player moves through a wall) during the training loop, the SWM is grounded in the environment's literal physics.

3.2.3 Parameter-Efficient Adaptation (PEFT)

To maintain the agility of the 8B-parameter class, researchers increasingly utilize **QLoRA (Quantized Low-Rank Adaptation)**. This allows for the attachment of game-specific "physics adapters" to a frozen base model. Recent benchmarks using the *UnSloth* library show that Llama 3.2 or Qwen3 models can be specialized for a complex game domain in under 20 minutes of training on consumer-grade hardware, making on-device customization a feasible reality for the gaming industry.

3.3 Deterministic Grounding and Thinking Modes

The introduction of "Thinking Modes" in the Qwen3 and Llama 3.2 architectures provides a unique mechanism for SWMs to verify their own world-state predictions. Unlike previous iterations, these models can be prompted to perform a **Chain-of-Thought (CoT) Verification** of the state transition before outputting the final structured data.

3.3.1 Internal Monologue and Symbolic Self-Correction

By separating the reasoning trace from the structured output, the SWM acts as its own symbolic validator. This "Internal Monologue" allows the model to verify preconditions against the current context before committing to a state update.

> *Model Thought:* "Player uses Key on Door. Precondition: Has_Key=True. Result: Door_Locked=False. Change: Update Lock status."
> *Final Output:* {`"Door_Locked":false`}

Recent studies on **In-Context Verification (ICV)** show that this dual-pass approach reduces logical inconsistencies in 8B models by 34.2%, bringing their world-modeling reliability close to that of 70B models while maintaining significantly lower latency.

3.3.2 On-Device Deployment and NPU Acceleration

A defining advantage of the 8B scale is its feasibility for **On-Device deployment**. With the 2026 generation of gaming hardware, dedicated Neural Processing Units (NPUs) and mobile SoCs (System-on-Chips) are optimized for the 4-bit and 8-bit quantized weights used by SWMs.

By leveraging **DirectNPU kernels** and INT8 quantization, an 8B model can operate entirely within the local VRAM of a console or high-end PC, eliminating the network latency and privacy concerns associated with cloud-based API calls. This local execution loop is critical for seamless agent-human interaction, where a 200ms round-trip delay to a server would break the immersion of the real-time game environment [1].

3.3.3 The System 1/System 2 Duality

The 8B model effectively bridges the gap between fast, reactive gameplay (System 1) and slow, deliberative planning (System 2). In its standard mode, the SWM provides high-speed state continuations; when switched to "Thinking Mode", it performs deeper causal analysis for high-entropy states. This architectural flexibility allows a single model to handle both the mundane physics of the world and the complex strategic planning required for long-horizon goals [8].

3.4 The Spectrum of Scale: From Edge Models to SWMs

While this survey identifies the 8B-parameter class as the "Goldilocks" zone for complex reasoning, the 2026 landscape has seen the emergence of edge-SLMs in the 1B to 3B range. Models such as Llama 3.2:1B, Phi-3.5 Mini, and Nemotron-3 Nano provide a necessary lower-bound for latency-critical world modeling.

3.4.1 Hyper-Throughput and Parallel Dreaming

The primary advantage of 1B–3B models is their extreme inference throughput. On consumer-grade NPUs (Neural Processing Units) or mobile SoCs, a 1B model can reach speeds exceeding 200 tokens per second. This speed facilitates a higher degree of **Parallel Dreaming**, where an agent can simulate hundreds of branching futures within a single 16ms game frame. This is

particularly vital for real-time action games where the decision-making window is too narrow for the deeper, slower reasoning of an 8B model.

3.4.2 Logical Fidelity vs. Latency Trade-offs

Investigative benchmarks reveal a clear "emergence floor" for world modeling capabilities. While 1B models excel at simple, repetitive state validation—such as checking if an item is in an inventory—they often struggle with the **Long-Horizon Consistency** required for complex strategy games. As shown in Table 1, the 3B-class represents a middle lane, offering function-calling and structured reasoning superior to 1B models while maintaining significantly lower VRAM requirements than 8B variants.

Table 1: Comparative Performance of Edge-SLMs vs. SWMs in World Modeling

Model Class	Parameters	Latency	MCTS Depth	Accuracy
Edge-Mini	1B – 1.5B	$<$ 10ms	High (200+ samples)	Low/Medium
Edge-Mid	3B – 3.8B	15 – 30ms	Medium (50+ samples)	Medium/High
SWM	**7B – 8B**	**50 – 150ms**	**Low (10+ samples)**	**SOTA**

3.4.3 Hybrid Hierarchical Modeling

Current research proposes a **Hierarchical World Model (HWM)** architecture to leverage this entire spectrum. In this framework, an **Edge-class model (1B)** acts as a "Fast-Physics" validator, handling high-frequency, low-logic transitions. Simultaneously, an **8B SWM** acts as the "Strategic Deliberator", performing deeper rollouts only when the environment encounters high-entropy states or novel scenarios. This tiered approach ensures that the agent remains responsive without sacrificing the depth of its "System 2" imagination.

4 Probabilistic Transition Modeling

In stochastic game environments—characterized by random encounters, variable damage rolls, or procedural events—a world model must transcend deterministic state mapping. We formalize the Language-based World Model (L-WM) as a conditional probability distribution over the state space $\mathcal{S}$. Given a current state s_t and an action a_t, the model estimates the density of the subsequent state:

$$P(s_{t+1}|s_t, a_t) \approx P_\theta(s_{t+1}|\text{prompt}(s_t, a_t)) \tag{1}$$

where θ represents the weights of the **SWM**.

4.1 Sampling as Uncertainty Quantification

Traditional Reinforcement Learning (RL) utilizes specialized heads for variance estimation to handle environmental stochasticity. In contrast, SWMs leverage the inherent probabilistic nature of autoregressive decoding, using *temperature scaling* and *Top-p sampling* to represent

environmental entropy. By executing N parallel inferences with a non-zero temperature τ, the agent constructs a discrete empirical distribution of potential futures:

$$\hat{\mathbb{P}}(s_{t+1}|s_t, a_t) = \frac{1}{N}\sum_{i=1}^{N}\delta(s_{t+1} - \hat{s}_{t+1}^{(i)}) \tag{2}$$

where $\hat{s}_{t+1}^{(i)}$ is the i-th "dreamed" outcome. This "Wisdom of the Samples" approach allows 8B models to implicitly model complex branching factors without explicit Bayesian overhead [16].

4.1.1 Calibration and Expected Calibration Error (ECE)

A significant challenge for SWMs is **Overconfidence**. Autoregressive models, particularly those subjected to heavy SFT, often exhibit low entropy even in high-uncertainty states (e.g., a 50/50 coin flip in-game). We investigate the use of **Temperature Tuning** to minimize the Expected Calibration Error (ECE). Research indicates that for 8B models, a dynamic temperature $\tau(s_t)$, conditioned on the current state's complexity, yields a 19% improvement in the reliability of MCTS value estimates compared to fixed-temperature sampling [10].

4.1.2 Handling Multi-Modal Distributions

In environments with multiple valid outcomes (e.g., a random loot drop), simple greedy decoding collapses the distribution. We argue for the use of **Beam-Search with Diversity Penalties** as a way to capture the "tails" of the game's transition distribution. By forcing the 8B model to explore low-probability tokens during the dream phase, the agent can prepare for low-frequency/high-impact events that would be missed by standard top-p sampling [16].

4.1.3 Logit-Based Uncertainty Estimation

Beyond sampling, recent 2026 frameworks extract the raw **Logit Entropy** $H = -\sum p \log p$ of the transition tokens as a direct proxy for world-model uncertainty. If the entropy H exceeds a predefined threshold, the agent can choose to skip the world model's prediction and fall back to a conservative, rule-based safety policy. This dual-track approach ensures that the agent's "System 2" imagination does not lead to reckless behavior in poorly understood regions of the state space [8].

4.2 Efficient State Representation: Delta-Prediction (Δs)

A significant challenge in using LLMs as simulators is the "State Explosion" problem, where verbose structured data (e.g., large JSON objects) exhausts the context window and increases latency. To mitigate this, we examine the **Delta-Transition Paradigm**. Instead of predicting the full vector s_{t+1}, the SWM is trained to predict the minimal edit set Δs, such that:

$$s_{t+1} = s_t \oplus \Delta s \tag{3}$$

where $\oplus$ is a symbolic update operator. This method reduces token consumption by up to 85%, significantly enhancing the feasibility of deep-horizon MCTS planning.

4.2.1 Symbolic Diffs and Patch Integrity

For 8B-parameter models, generating a valid Δs (e.g., a JSON Patch or a Python dict update) is more computationally efficient than full reconstruction. However, this introduces the risk of **Pointer Drift**, where a model predicts an update to a key that no longer exists in s_t. We propose the use of **Anchored Delta-Prediction**, where the model must first recite the key's current value before predicting the change (e.g., `HP: 100 -> 85`). This self-attention grounding ensures that the symbolic update operator $\oplus$ remains logically consistent with the preceding state context [13].

4.2.2 Recursive Error Accumulation in Delta-Space

A critical downside of Δs modeling is the **Compounding Symbolic Error**. If a single delta update is slightly incorrect (e.g., reducing HP to a negative value when the engine caps it at zero), every subsequent state in the MCTS tree will inherit this logical flaw. To combat this, modern 2026 frameworks implement **Periodic State Synchronization**. Every K steps in the "dream," the SWM is forced to generate a full state s_{t+K} to reset the accumulation of delta-drift, ensuring the long-horizon trajectory remains grounded in the game's global invariants [10].

4.2.3 Code-Based Diffs for Complex Physics

Building on the Code World Model (CWM) paradigm, recent work suggests that representing Δs as a discrete **Lambda Function** or a code snippet (e.g., `state.player.hp -= 15`) is superior to raw JSON diffs. For models like Qwen3, code-based diffs leverage pre-trained programming logic to handle edge cases—such as status effect durations or stackable buffs—that are difficult to capture in flat data structures. This allows the 8B model to act as a "Dynamic Patch Engine", essentially rewriting the state logic on the fly during the planning phase [13].

4.3 The "PoE-World" Framework: Product of Programmatic Experts

Building on the advancements in program synthesis, we explore the **Product of Programmatic Experts (PoE-World)** approach [13]. In this framework, the SWM does not treat the entire world state as a monolithic prediction target. Instead, it induces a set of local, probabilistic code snippets (e.g., Python lambdas) that represent the specific causal laws governing the current interaction.

4.3.1 Logic Decomposition and Expert Specialization

The PoE-World architecture allows the 8B model to act as an orchestrator that decomposes a state transition into specialized sub-modules:

- **Kinematic Experts:** Deterministic snippets handling movement and collision.
- **Heuristic Experts:** Probabilistic snippets handling AI behavior and loot tables.
- **Narrative Experts:** Semantic updates to the world-state's "Common Sense" context.

This allows the agent to switch between a fast, deterministic "physics engine" mode for standard movement and a reflective, probabilistic "reasoning" mode for complex NPC interactions. By executing these snippets in a lightweight sandbox, the SWM avoids the token-heavy process of describing every detail of the world, focusing only on the code that changes [13].

4.3.2 Dynamic Expert Selection (DES)

To maintain efficiency, the SWM employs **Dynamic Expert Selection**. Using the "Thinking Mode" discussed in Section 3.3, the model identifies which sub-systems of the game world are active in the current tick. If the player is merely walking, the model activates only the Kinematic Expert, bypassing the more expensive reasoning modules. Research indicates that DES reduces the effective FLOPs per decision step by up to 60% in complex open-world environments, enabling 8B models to simulate environments that previously required 70B+ parameters.

4.3.3 Sandbox Verification and Safety

A critical component of PoE-World is the **Programmatic Sandbox**. Before the induced code snippet is applied to the canonical game state, it is executed in a virtual environment to check for exceptions or infinite loops. This adds a layer of "hard logic" safety to the LLM's stochastic predictions. If the induced code fails the sandbox test, the system falls back to a conservative Delta-Prediction (see Section 4.2), ensuring that the world model never crashes the planning loop due to a syntax error or logical paradox [13].

4.4 Calibrating Imagination: Temperature and Entropy Control

A critical challenge in using $P_\theta(s_{t+1}|s_t, a_t)$ is the "over-confidence" of SFT-tuned models. In stochastic environments, 8B models tend to collapse toward the most likely outcome, ignoring low-probability but high-impact "black swan" events (e.g., a critical miss in combat).

4.4.1 Contrastive Decoding for World States

To address this, we propose **Contrastive World Decoding**, where the model's output is contrasted against a "physics-ignorant" base model. This amplifies the probability of transitions that are logically consistent with the game's specific mechanics while suppressing generic natural language continuations [15].

4.4.2 Entropy-Aware MCTS Rollouts

Integrating uncertainty into the planning loop allows the agent to distinguish between "known" transitions (e.g., moving through an open door) and "novel" interactions. By monitoring the **Token-Level Entropy** during state generation, the agent can trigger deeper MCTS rollouts only when the world model's confidence drops below a critical threshold, thereby optimizing the "inference budget" across a single game tick [10, 16].

5 Integration with Planning and Decision Making

A world model is only as effective as the planning framework that utilizes its predictions. For game agents, this integration transforms the LLM from a reactive responder into a proactive strategist. We focus on the synergy between SWMs and search-based decision algorithms, where the SWM serves as the transition engine $\mathcal{T}$ and a heuristic evaluator.

5.1 Multi-Functional Search: LLMs as Affordance and Validation Engines

The integration of MCTS with SWMs traditionally treats the language model as a passive transition function $\mathcal{T}(s, a) \to s'$. However, modern architectures leverage the LLM as a multi-functional orchestrator within the search tree. This involves expanding the model's role to include **Action Affordance Generation** and **State-Action Validation** [22].

5.1.1 Action Affordance and Proposal

In environments with vast or continuous action spaces, exhaustive search is intractable. We utilize the SWM as an *affordance generator*—pruning the action space $\mathcal{A}$ to a semantically relevant subset $\mathcal{A}_{valid} \subset \mathcal{A}$ based on the current context s_t [21]. By prompting the model in "Thinking Mode" (*e.g., "Given the player is in the Kitchen, list 3-5 logical actions"*), the branching factor of the MCTS is reduced from hundreds of potential engine commands to a handful of strategic intents. This ensures that the search budget is spent exploring "semantically meaningful" regions of the state space rather than physically impossible trajectories.

5.1.2 The Validity Filter: Precondition and Effect Checks

A critical failure mode of earlier world models was "hallucinatory transitions," where an agent "dreams" a successful outcome that violates game physics. To address this, we implement a **Validity Filter** within the MCTS loop [11]. Before a node is expanded, the SWM performs a dual-validation check:

1. **Action Validation (Precondition):** The model verifies if action a_t is applicable in state s_t (*e.g., "Can I 'Open Door' if 'Has_Key' is false?"*).
2. **State Validation (Effect):** Upon predicting s_{t+1}, the model evaluates if the new state maintains structural integrity (*e.g., "Does the player's position overlap with a wall entity?"*).

This "What-If Analysis" framework allows the agent to reject invalid rollouts during the "Simulation" phase of MCTS, significantly increasing the reliability of the final planned path.

5.1.3 Heuristic Value Estimation

Beyond transitions, the SWM serves as the terminal value function $V(s)$. By processing the predicted state description, the 8B model provides a heuristic score based on "common sense" progress towards the goal. This replaces the need for hand-crafted reward functions in sparse

environments, allowing the agent to navigate complex narrative tasks where a binary success/failure signal is insufficient.

5.2 Strategic Deliberation: Heuristic Pruning and Best-First Search

Section 5.2 explores how language-centric world models serve as "System 2" deliberators by guiding search toward high-reward logical clusters through proactive simulation and heuristic pruning.

5.2.1 Proactive Thinking via What-If Analysis (WiA)

The integration of **What-If Analysis (WiA)** represents a milestone in the transition from reactive to proactive agents. Moving beyond simple next-token prediction, WiA allows an 8B model to proactively forecast consequences by asking, "*If I take this action, how will it affect the game state?*" Research demonstrates that WiA-augmented SWMs achieve significant gains in multi-step forecasting accuracy by pruning nonsensical trajectories before they are processed by the game engine [8].

5.2.2 Heuristic Best-First Tree Search (BFTS)

We identify the current state-of-the-art as **Best-First Tree Search (BFTS)** for language agents [8]. Unlike depth-first methods, BFTS selects nodes based on an evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the environment reward and $h(n)$ is the SWM's heuristic estimate of goal distance. Empirical evidence shows that BFTS on 8B models significantly increases success rates for long-horizon tasks by allowing the agent to backtrack when it detects a low-value predicted state.

5.3 Sim-to-Real Alignment: The RWML Feedback Loop

The most significant hurdle for SWMs is **Compounding Error**, where small mis-predictions in the world model lead to entirely divergent "dreams" during deep-horizon search [22]. To mitigate this, we examine **Reinforcement World Model Learning (RWML)**. As the agent interacts with the game, it collects transition pairs to compute an alignment loss:

$$\mathcal{L}_{align} = \|\phi(\hat{s}_{t+1}) - \phi(s_{t+1})\|^2 \tag{4}$$

5.3.1 Dynamic Adapter Switching

In 2026 architectures, alignment loss is controlled through **Dynamic Adapter Switching**. When $\mathcal{L}_{align}$ exceeds a safety threshold, the agent performs a "Real-World Calibration" [22]. The game engine's ground truth is used to generate a one-shot LoRA update, allowing the model to "re-learn" specific physics or NPC patterns of the current scene without full retraining.

5.3.2 Memory-Augmented Re-Planning

Integrating **Memory-Augmented World Models (WorldMem)** allows the agent to reference past ground-truth transitions stored in a vector database [22]. This ensures that if the

agent encounters a previously seen state, it prioritizes the remembered engine response over the model's creative inference, grounding the imagination in historical reality.

6 World Model Maintenance: Versioning and Rule Evolution

A critical yet under-explored challenge in the deployment of SWMs is the temporal instability of game rules. Unlike physical laws, game mechanics are frequently modified through "patches" to balance gameplay, fix bugs, or introduce seasonal content. For a world model to remain a reliable simulator, it must maintain strict synchronization with the current engine version while potentially retaining knowledge of legacy versions for backward compatibility.

6.1 Consistency Under Rule Drift

When a game engine is updated from version V_n to V_{n+1}, a static world model becomes a source of *strategic hallucination*—predicting outcomes that were true in the past but are currently invalid. This is particularly problematic in competitive gaming, where even minor changes to unit statistics or cooldowns can render a planned search-tree trajectory obsolete.

6.1.1 Knowledge Partitioning via Multi-LoRA Architectures

Research into **Context-Isolated LoRA** suggests that 8B models can maintain multiple version-specific adapters concurrently. By toggling these low-rank adapters based on the game's version metadata, a single base model can simulate both "Legacy" and "Current" versions of a game without catastrophic interference [22]. This is achieved through a **Router Mechanism** that dynamically weights the contribution of each adapter during the forward pass, ensuring that logic from Version 1.0 does not bleed into a Version 2.0 simulation.

6.1.2 Delta-Rule Fine-Tuning and Contrastive Learning

Rather than retraining the entire SWM, recent works propose training only on the *diff* (change-log) of the game state. This ensures that the model prioritizes new rule logic while retaining the unchanged baseline physics of the game world. To enhance this, we propose **Contrastive Rule Learning**, where the model is trained on pairs of (s_t, a_t, s_{t+1}) transitions that specifically highlight changed mechanics. By contrasting the old transition s_{V_n} against the new one $s_{V_{n+1}}$, the 8B model develops a "version-aware" latent space, allowing it to explicitly distinguish between outdated and current mechanics during MCTS rollouts [14].

6.1.3 Parameter-Efficient Rule Merging

In scenarios where a game update is cumulative, we explore **Weight Merging (Model Breadcrumbing)**. Instead of isolated adapters, the model uses spherical linear interpolation (SLERP) to merge the delta-weights of successive patches. This allows the SWM to maintain a continuum of the game's evolution, facilitating a graceful transition for agents that must support players who may be lagging behind the latest patch cycle without requiring a linear increase in VRAM for every version supported [22].

6.2 Version Identification via Probing

To ensure an agent is operating in the correct versioned reality, we propose the use of **World Model Probes**. These are lightweight diagnostic queries designed to identify the underlying engine version through behavioral observation.

6.2.1 Probabilistic Version Probing

By executing a set of "canary actions"—actions known to have changed between versions (e.g., a weapon's damage values or the cost of a specific craftable item)—the agent can analyze the environment's response. If the engine's output s_{t+1} consistently aligns with version-specific predictions $\hat{s}_{V_{n+1}}$, the model confirms its operational context. This allows for **Automatic Context Switching**, where the LLM adjusts its active LoRA weights or system prompt to match the detected version.

6.2.2 Context Management for Multi-Version Support

Effective context management involves prepending the world-state with a *Version Anchor*. For example, a prompt formatted as `[Version:2.1.0] [State:...]` allows the attention mechanism of models like Qwen3 to selectively activate weights associated with specific patch notes. This is crucial for cross-play environments where an agent may interact with different entities operating on disparate client versions.

6.3 Efficient Training and Evaluation of Version-Aware Models

Evaluating a world model's version consistency requires more than standard perplexity metrics. We introduce the **Cross-Version Consistency Score (CVCS)**, which measures the model's ability to suppress knowledge from Version A when prompted for Version B.

6.3.1 Evaluation via Counterfactual Probes

We evaluate models by presenting them with "Rule Conflict" scenarios. For instance, if a rule was changed from "Fire heals" to "Fire damages," a robust world model should output the damage state only when the Version 2.0 anchor is present. High CVCS scores indicate that the model has successfully decoupled its knowledge bases, preventing "version leakage" that could lead to invalid planning in MCTS rollouts.

6.3.2 Training Efficiency: Adaptive Distillation

To maintain efficiency, rather than full-scale retraining for every minor patch, we utilize **Adaptive Distillation**. A large "Teacher" model (e.g., DeepSeek-V3) is prompted with new patch notes to generate updated synthetic trajectories. These are then used to update the 8B "Student" via a narrow, high-learning-rate LoRA update. This allows the SWM to remain synchronized with the game's evolution with minimal compute overhead, ensuring that maintenance costs do not scale linearly with the number of game updates.

7 Conclusion and Future Directions

The evolution from reactive, model-free agents to proactive simulators powered by Small World Models (SWMs) represents a pivotal shift in 2026 AI research. This survey has demonstrated that for discrete game environments—where rules are deterministic and states are structured—the 8B-parameter class of models, such as **Qwen3:8B**, offers the optimal equilibrium between logical fidelity and computational throughput. By treating game states as a formal language, we move past the perceptual bottlenecks of vision-based architectures, enabling agents to "dream" and plan with symbolic precision.

7.1 Summary of Contributions

We have argued that SWMs provide a superior pathway for real-time game AI due to their high inference efficiency, which facilitates massive parallel rollouts within the rigid constraints of a game engine's tick rate. Furthermore, the adoption of the **Delta-Transition Paradigm** (Δs) addresses the state-explosion problem, allowing for deep-horizon search in context-constrained environments. These advancements suggest that the "intelligence" of a game agent is increasingly defined by the consistency of its internal simulation rather than its absolute parameter count.

7.2 Future Directions

As research progresses toward 2027, several critical challenges remain:

- **Long-Horizon Consistency:** While 8B models excel at immediate state transitions, compounding errors still affect long-term trajectories. Future research into *Memory-Augmented World Models* (e.g., WorldMem [20]) will be essential to ground the "imagination" of the LLM over extended gameplay sessions.
- **On-Device Self-Correction:** The deployment of SWMs on local NPU hardware will enable real-time *Sim-to-Real Alignment*. We anticipate the rise of architectures that utilize on-device LoRA to dynamically update the world model when the game engine's reality contradicts the model's internal predictions.
- **Causal Discovery in Open Worlds:** Moving beyond predefined rules, the next frontier involves agents that can *induce* the world model of a novel game through pure observation, shifting from supervised fine-tuning to autonomous causal discovery.

Ultimately, mastering the structured "toy worlds" of games serves as a critical stepping stone. The methodologies refined in these discrete environments lay the foundation for language-centric world models capable of navigating the complex, non-deterministic physics of the human world.

References

[1] Apple intelligence foundation language models tech report 2025, 2025.

[2] Baosong Yang et. al. An Yang, Anfeng Li. Qwen3 technical report, 2025.

[3] Google Machine Learning Education. Llms: Fine-tuning, distillation, and prompt engineering, 2025.

[4] Lyle Goodyear, Rachel Guo, and Ramesh Johari. The effect of state representation on llm agent behavior in dynamic routing games. *arXiv preprint arXiv:2506.15624v1*, 2025.

[5] Y. Gu et al. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543v1*, 2023.

[6] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[7] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.03035*, 2023.

[8] Shibo Hao et al. Reasoning with language model is planning with world model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[9] Sihao Hu, Tiansheng Huang, Gaowen Liu, Ramana Rao Kompella, Fatih Ilhan, Selim Furkan Tekin, Yichang Xu, Zachary Yahn, and Ling Liu. A survey on large language model-based game agents, 2025.

[10] X. Hu et al. Calibrating the imagination: Uncertainty quantification in llm-based world models. *arXiv preprint arXiv:2511.08231*, 2025.

[11] N. Lambert, Z. Guo, et al. Rlvr-world: Training world models with reinforcement learning. *arXiv preprint arXiv:2505.13934v2*, 2025.

[12] Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022.

[13] Wolfgang Lehrach, Daniel Hennes, Miguel Lazaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Jordi Grau-Moya, Marc Lanctot, Atil Iscen, John Schultz, Marcus Chiam, Ian Gemp, Piotr Zielinski, Satinder Singh, and Kevin P. Murphy. Code world models for general game playing, 2025.

[14] Hao Li, Xiaogeng Liu, Hung-Chun Chiu, Dianqi Li, Ning Zhang, and Chaowei Xiao. Drift: Dynamic rule-based defense with injection isolation for securing llm agents, 2025.

[15] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization, 2023.

[16] Yingjie Li, Yun Luo, Xiaotian Xie, and Yue Zhang. Task calibration: Calibrating large language models on inference tasks, 2024.

[17] Local AI Zone Research. Llm model parameters 2025: Master 7b, 13b, 70b parameter selection performance optimization. *Local AI Zone Reports*, 2025.

[18] SuperAnnotate Research. Fine-tuning large language models (llms) in 2025. 2025.

[19] Yuan Sui, Yanming Zhang, Yi Liao, Yu Gu, Guohua Tang, Zhongqian Sun, Wei Yang, and Bryan Hooi. What-if analysis of large language models: Explore the game world using proactive thinking, 2026.

[20] Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory, 2026.

[21] Chang Yang, Xinrun Wang, Junzhe Jiang, Qinggang Zhang, and Xiao Huang. Evaluating world models with llm for decision making, 2024.

[22] Xiao Yu, Baolin Peng, Ruize Xu, Yelong Shen, Pengcheng He, Suman Nath, Nikhil Singh, Jiangfeng Gao, and Zhou Yu. Reinforcement world model learning for llm-based agents, 2026.

[23] Hongyu Zhao, Siyu Zhou, Haolin Yang, Zengyi Qin, and Tianyi Zhou. Neuro-symbolic synergy for interactive world modeling, 2026.