PROGRAMMABLE WORLD MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent video world models generate increasingly realistic and interactive visual experiences, yet lack reliable mechanisms for maintaining persistent world state and enforcing programmable rules over extended interactions. We introduce **Programmable World Model**, a framework that decouples world-state evolution from visual observation generation. An agent translates natural-language instructions into executable programs that specify entity states and state-transition rules, enabling direct control over individual entities and their interactions. A lightweight engine executes these programs to update and maintain an explicit, persistent global world state, including off-screen entities and non-visual attributes. To connect world state with visual generation, we introduce state-augmented 3D oriented bounding boxes (OBBs) as an intermediate representation. This representation, together with the target camera trajectory, is deterministically compiled into pixel-aligned spatiotemporal conditioning signals for a pretrained video model serving as the generative renderer. This design allows users to create playable games with predefined mechanics, direct control over individual entities, and persistent world state throughout gameplay. We further introduce COMBATSTATEBENCH, a benchmark for evaluating programmable world models. On COMBATSTATEBENCH, our method achieves 94% Count Accuracy and 98% State Accuracy, substantially outperforming existing interactive video world models while supporting coherent long-horizon generation. These results demonstrate the effectiveness of separating explicit state evolution from generative rendering for building persistent, programmable worlds.

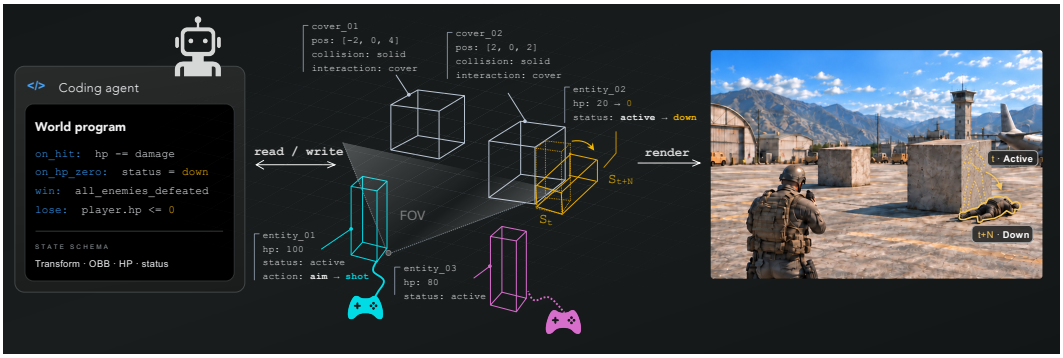


Figure 1: **Overview of the programmable world model.** Given a single reference image and a user description, a coding agent generates executable programs that specify entity states and interaction rules. A lightweight engine executes these programs to maintain and evolve an explicit world state, represented by state-augmented 3D OBBs. This representation is compiled into conditioning signals for a generative renderer, which produces realistic visual observations consistent with the world state. This design allows users to create playable games by defining game mechanics in advance, directly controlling individual entities, and maintaining persistent world state throughout gameplay.

1 INTRODUCTION

Recent video world models (Ball et al., 2025; Sun et al., 2025; Team et al., 2026a;b; Wang et al., 2026) seek to build interactive world engines on top of generative video models Wan et al. (2025); HaCohen et al. (2024). By predicting subsequent observations from visual histories and user actions,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

they synthesize increasingly realistic and responsive environments, opening new possibilities for immersive world creation.

However, turning video world models into interactive world engines requires capabilities beyond generating plausible observations. First, existing control interfaces primarily specify cameras, actions, or high-level prompts, offering limited support for directly addressing and manipulating individual entities. Second, these models generally lack an explicit, persistent global state that can be accessed and updated independently of the current view. Such state must account for off-screen entities and nonvisual information, including inventory, task progress, and interaction history, which is also important for a multiplayer world engine. Third, users cannot readily program the rules governing world evolution, such as specifying the conditions under which a door opens or how an interaction affects other entities. Prompting a desired outcome does not establish an executable rule that consistently governs subsequent interactions.

We introduce **Programmable World Model**, a framework that decouples world-state evolution from visual observation generation (Figure 1). A coding agent translates user instructions into executable programs that define entity states and world rules, while a generative renderer produces observations conditioned on the evolving state. Connecting these components requires an interface that is compact and directly editable by programs, yet sufficiently expressive to guide visual generation. Our central question is therefore: *what representation supports programmable world-state evolution while preserving the spatial and semantic structure needed for visual generation?* We analyze representation trade-offs in Appendix A.

Following these requirements, we represent each entity using a *state-augmented 3D oriented bounding box (OBB)*. Each OBB specifies an entity's position, extent, and orientation in a shared world coordinate system using a compact, fixed set of geometric parameters, without requiring a mesh, material, skeleton, or predefined animation. We augment this geometry with persistent identity, semantic category, dynamic state, and appearance information. Inspired by advances in generative rendering Huang et al. (2026b); Liang et al. (2025), we communicate this structured spatial information through rasterized controls. While approaches such as RenderFormer Zeng et al. (2025) directly map structured scene tokens to images, we explicitly resolve camera projection and entity-to-pixel correspondence through a deterministic *state compiler*. The compiler projects the evolving OBBs along the target camera trajectory and rasterizes them into pixel-aligned spatiotemporal controls. These controls guide entity identity, semantics, orientation, motion, and state changes, while the video model synthesizes the fine geometry, appearance, articulation, and secondary dynamics left unspecified by the representation.

Given a single reference image and a natural-language description, a VLM-based coding agent constructs the initial entity layout, assigns states and attributes, and writes programs defining interactions, event triggers, and objectives. During interaction, a state executor applies player actions and events according to these rules. The state compiler converts the updated world state into rendering controls, and the generative renderer produces the next observations. Users can revise world behavior through further instructions to the coding agent. To train the renderer, we develop a video data curation pipeline that recovers the spatial and semantic supervision required by this interface.

We introduce COMBATSTATEBENCH, a controlled benchmark for evaluating consistency between generated videos and engine-maintained world states. Through combat scenarios with diverse camera and entity motion, it measures whether generated videos preserve visible alive-character counts and visually realize death states.

Our contributions are threefold:

- We introduce **Programmable World Model**, which decouples executable world-state evolution from generative rendering to enable entity-level control, persistent state management, and user-programmable world rules. State-augmented 3D OBBs and a deterministic state compiler connect this editable world state to video generation through view-consistent, pixel-aligned spatiotemporal controls.

- We develop a data curation pipeline that extracts spatial and semantic supervision from videos, providing a practical path toward scaling training data for programmable generative worlds.

- We introduce COMBATSTATEBENCH, a controlled benchmark for evaluating consistency between generated videos and engine-maintained world states, focusing on visible alive-character counts and the visual realization of death states under diverse camera and entity motion.

2 RELATED WORK

2.1 INTERACTIVE VIDEO WORLD MODELS

Recent video world models have made rapid progress toward interactive visual environments, supporting action-conditioned generation (Bruce et al., 2024; Yu et al., 2025b), controllable camera motion (Li et al., 2025a; Ren et al., 2025), real-time interaction (Valevski et al., 2024; He et al., 2025; Team et al., 2026b), and increasingly long-horizon rollouts (Hong et al., 2025; Sun et al., 2025; Mao et al., 2025a; Yang et al., 2025a). These models are typically trained primarily with observation-generation objectives, which supervise whether the predicted observations are visually plausible but do not explicitly require the model to maintain a persistent, structured world state over time. As a result, entities, attributes, relations, and interaction outcomes may be represented only implicitly in the generative context, without being organized into an authoritative state that must remain consistent across occlusion, camera motion, or long interaction horizons. Recent systems introduce temporal or spatial memory to improve visual consistency over extended rollouts (Xiao et al., 2025; Li et al., 2025b; Wu et al., 2025; Po et al., 2025; Yu et al., 2025a; Team et al., 2026a; Wang et al., 2026; Yi et al., 2026), but such memory is still optimized primarily for observation generation rather than for preserving executable world facts. Our work instead separates these two responsibilities: a lightweight engine explicitly maintains and advances the canonical world state, while the video model serves as a generative renderer conditioned on the resulting state.

2.2 EXPLICIT-STATE WORLD MODELING

Recent work has begun to move beyond purely observation-centric world modeling by explicitly representing the internal state of interactive environments (Deng et al., 2026; Fang et al., 2026; Duan et al., 2026; Gu et al., 2026a; Li et al., 2026; O'Mahony et al., 2025). StatePlay (Lin et al., 2026b) jointly predicts visual observations and game-state variables, allowing the predicted state to guide visual generation and improve mechanics consistency. However, because the state itself is predicted by the model, state errors can directly lead to incorrect world updates and may accumulate over long interaction horizons. MASS (Cai et al., 2026) further introduces an authoritative typed state for multiplayer world modeling and separates state dynamics from view rendering. Its shared state, however, is advanced by a learned Logic Engine, so the authoritative state evolution still depends on learned transition dynamics and remains susceptible to transition prediction errors. Unlike StatePlay and MASS, our framework maintains a canonical world state in a lightweight engine and executes state transitions according to explicit rules, making the world state directly programmable, editable, and verifiable.

2.3 GENERATIVE RENDERING

Generative rendering leverages learned generative models to synthesize photorealistic visual observations from structured scene conditions, reducing reliance on conventional graphics pipelines (Zeng et al., 2024; 2026; Beisswenger et al., 2025). Different approaches adopt renderer-facing representations with varying levels of geometric explicitness (Cai et al., 2024; Kim et al., 2025; Gu et al., 2026b; 2025; Chen et al., 2026). DiffusionRenderer (Liang et al., 2025) performs diffusion-based forward and inverse rendering using G-buffers that encode geometric and appearance-related scene attributes. The AlayaRenderer series (Huang et al., 2026b; Lin et al., 2026a) develops this paradigm for dynamic world rendering. AlayaRenderer (Huang et al., 2026b) focuses on high-fidelity generative rendering of dynamic worlds, while AlayaRenderer-Flash (Lin et al., 2026a) further improves generation efficiency for real-time interactive rendering. These works demonstrate that generative priors can synthesize rich appearance, material, lighting, and dynamic details from explicit structural rendering signals.

Moving toward lighter structural conditioning, Coarse-to-Real (Gomez-Nogales et al., 2026) synthesizes dynamic scenes from coarse 3D proxies, using simplified geometry to constrain scene layout,

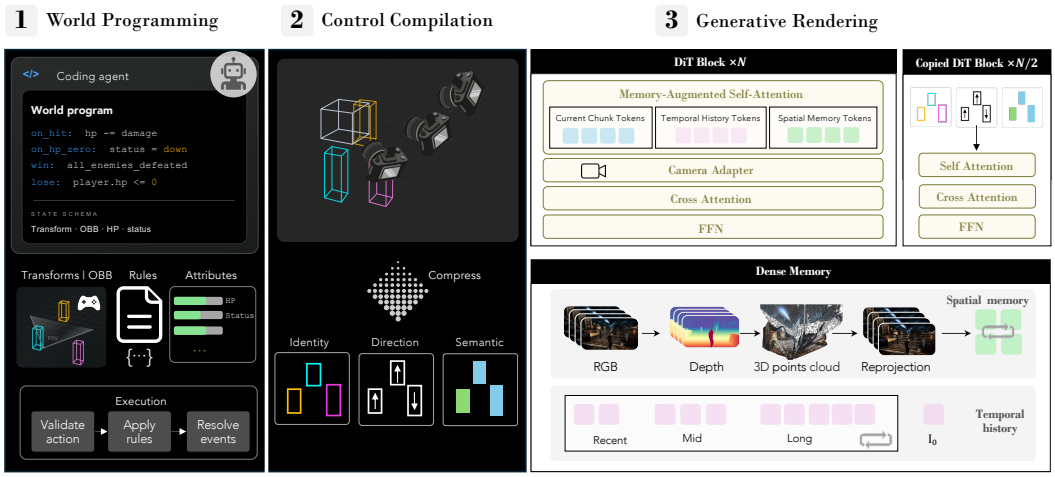


Figure 2: **Architecture of the programmable world model.** (1) Agent-orchestrated world programming starts from natural-language specifications to instantiate an executable canonical world state, which is maintained and advanced by a lightweight engine according to player actions and world rules. (2) Control compilation projects state-augmented 3D OBBs under the target camera into pixel-aligned identity, semantic, and motion-direction controls. (3) Generative rendering conditions a pretrained camera-controlled video generation model on these structured controls and visual history to synthesize the next video chunk. Completed chunks are incorporated into temporal and geometry-aligned spatial memories to support long-horizon generation.

camera motion, and object trajectories while leaving fine-grained geometry, appearance, articulation, and secondary dynamics to the generative model. Together, these works illustrate a spectrum of generative rendering interfaces, where more explicit representations provide stronger structural constraints, while lighter representations delegate a larger fraction of visual and dynamic realization to the generative prior (Dominici et al., 2026).

3 Method

3.1 Problem Formulation

We consider an interactive generative world initialized from a visual observation I_0. At interaction step t, a player takes an action a_t, and the system maintains a canonical world state s_t that records the persistent information required to execute world interactions. The overall interaction loop is

$$s_t \xrightarrow[a_t]{F} s_{t+1} \xrightarrow[C_{t+1}]{P} M_{t+1}^{\text{ctrl}} \xrightarrow{G} I_{t+1}, \tag{1}$$

where F denotes the state transition executed by the lightweight engine, C_{t+1} is the target camera at the next interaction step, P is the deterministic state compiler that projects the updated world state under C_{t+1} into camera-aligned spatial controls, and G denotes the generative renderer that synthesizes the corresponding visual observation I_{t+1}. We detail the representation, programming, and state transition F of the programmable world in Section 3.2, the state compiler P in Section 3.3, and the generative renderer G in Section 3.4. Figure 1 provides an overview of the framework.

3.2 Agent-Orchestrated Box World

World representation and initialization. We represent the world as persistent entities grounded by 3D oriented bounding boxes (OBBs), which specify their positions, extents, and orientations. Each entity has a persistent identifier, a semantic category, and functional attributes such as health and faction membership. The world also stores inter-entity relations, such as hostility and ownership, and executable rules governing state changes under actions and events. The canonical world state at interaction step t is

$$s_t = (\mathcal{E}_t, \mathcal{A}_t, \mathcal{Q}_t; \mathcal{R}_t), \tag{2}$$

where $\mathcal{E}_t$ contains the persistent entities and their 3D OBBs, $\mathcal{A}_t$ their semantic and functional attributes, $\mathcal{Q}_t$ their relations, and $\mathcal{R}_t$ the executable rules.

Given the initial observation I_0, an off-the-shelf 3D detector estimates the visible entities' OBBs. We assign these entities persistent identifiers and initialize $\mathcal{E}_0$ from the detections. The agent orchestrator specifies the remaining attributes, relations, and rules.

Agent-orchestrated world programming. The orchestrator combines the recovered 3D layout with the player's natural-language description q to generate an engine-readable world program. Grounded in persistent entity identifiers, the program specifies initial entity states and relations, supported actions, and state-transition rules, with optional global constraints, event triggers, and objectives. The engine executes this program to instantiate s_0.

For a combat scenario, the orchestrator assigns factions, initializes health and combat states, specifies valid attack relations, and defines shooting effects such as damage, death, and objective updates. This lightweight box world uses OBBs as its geometric scaffold and structured states, relations, and rules to encode interaction semantics.

Engine execution. The engine advances the canonical state according to the world program and player action a_t, using the transition function F introduced in Eq. (1):

$$s_{t+1} = F(s_t, a_t). \tag{3}$$

At each step, the engine validates the action against the current attributes, relations, and rules $\mathcal{R}_t$, applies its effects, and resolves triggered events. Changes to entities, attributes, relations, and, when applicable, rules are recorded in s_{t+1}.

The engine also explicitly maintains variables such as health, inventory, and faction membership, which may not be directly visible but affect future transitions and rendered outcomes. The updated state specifies the interaction outcome in world coordinates; the next stage converts its visually relevant content into renderer-facing spatial controls aligned with the target camera.

3.3 Control Compilation

From world states to spatial controls. The state compiler P projects renderer-relevant variables from the updated canonical state s_{t+1} under the target camera C_{t+1} without modifying the underlying state:

$$M_{t+1}^{\text{ctrl}} = P(s_{t+1}, C_{t+1}). \tag{4}$$

Projected OBBs specify each entity's spatial support, augmented with identity, semantic, and object-motion attributes. Explicit object-motion states complement projected displacement, which may arise from object motion, camera motion, or both.

Structured projected-box controls. We encode these attributes in three spatially aligned control maps.

The identity map M_{t+1}^{id} encodes persistent instance correspondence across frames, including occlusion and view re-entry. Each entity identifier k_i is assigned a fixed slot in a bank of K learnable identity embeddings:

$$\mathcal{E}^{\text{id}} = \left\{ q_j^{\text{id}} \right\}_{j=1}^{K}. \tag{5}$$

The selected embedding is rasterized over the visible OBB projection, providing a spatial key for the entity's appearance reference. During training, slots are sampled uniformly without replacement within each sample, with assignments fixed across its frames but randomized across samples. This discourages slot-specific associations with category, appearance, or viewpoint. At inference, assignments remain fixed across interaction steps.

The semantic map M_{t+1}^{sem} encodes each entity's category label y_i from the canonical state using a pretrained text encoder:

$$q_i^{\text{sem}} = E_{\text{text}}(y_i). \tag{6}$$

Each embedding is rasterized over the visible OBB projection, with zero background features and depth-aware rasterization resolving box overlaps. Entities of the same category share semantic embeddings but remain distinguishable through their identity slots. This provides category guidance for entities absent from the initial observation or lacking a reliable appearance reference.

For the direction map M_{t+1}^{dir}, the compiler rotates each entity's world-space velocity into target camera coordinates and quantizes the result into seven motion states: FORWARD, BACKWARD, LEFT, RIGHT, STATIC, UP, and DOWN. These states are encoded by learnable embeddings:

$$\mathcal{E}^{\mathrm{dir}} = \left\{ q_\ell^{\mathrm{dir}} \right\}_{\ell=1}^{7}. \tag{7}$$

The selected embedding is rasterized over the visible OBB projection, while background and invalid regions receive zero features. Because the motion state derives from world-space velocity rather than image-space displacement, a static entity retains the STATIC state even when camera motion changes its projected position. Together with the target camera trajectory, this encoding helps distinguish object motion from camera-induced apparent motion.

Finally, the three maps are concatenated along the channel dimension:

$$M_{t+1}^{\mathrm{ctrl}} = \mathrm{Concat}\left(M_{t+1}^{\mathrm{id}}, M_{t+1}^{\mathrm{sem}}, M_{t+1}^{\mathrm{dir}} \right). \tag{8}$$

Together with the target camera trajectory, these maps provide spatial, identity, semantic, and object-motion guidance to the generative renderer.

3.4 Generative Renderer

3.4.1 Conditional Video Generation

We build the renderer on LingBot-World-v1 (Team et al., 2026b), retaining its pretrained camera-conditioning pathway for global viewpoint control. Each rendering window generates T frames under a target camera trajectory $\mathcal{C} = \{C_t\}_{t=1}^{T}$, which is also used to compile the projected-box controls.

We attach a trainable Structured Spatial ControlNet to the backbone. Following Eq. (8), the compiler constructs

$$M_t^{\mathrm{ctrl}} = \mathrm{Concat}\left(M_t^{\mathrm{id}}, M_t^{\mathrm{sem}}, M_t^{\mathrm{dir}} \right), \tag{9}$$

yielding the control sequence $M_{1:T}^{\mathrm{ctrl}} = \{M_t^{\mathrm{ctrl}}\}_{t=1}^{T}$. These controls specify the spatial support, persistent identity, semantic category, and motion direction of engine-maintained entities. Motion directions are derived from world-space object velocities expressed in target camera coordinates, rather than image-space displacement, complementing the camera-conditioning pathway.

The control sequence is encoded at the video latent's spatial resolution and processed by the ControlNet together with the noisy video latent. Layer-wise outputs are projected to the backbone feature dimension, yielding control residuals $r^{(\ell)}$ that are added to the corresponding main blocks:

$$h_{\mathrm{main}}^{(\ell)} \leftarrow h_{\mathrm{main}}^{(\ell)} + r^{(\ell)}. \tag{10}$$

During training, the pretrained backbone parameters θ, including the inherited camera-conditioning pathway, remain frozen, while the spatial-control branch parameters ϕ are optimized.

The fixed-window rendering process is

$$I_{1:T} = G_{\theta,\phi}\left(I_0, \mathcal{C}, M_{1:T}^{\mathrm{ctrl}} \right), \tag{11}$$

where $I_{1:T} = \{I_t\}_{t=1}^{T}$ is the generated clip and I_0 provides the initial visual context. The pretrained generative prior synthesizes details not explicitly specified by the controls, including shape, texture, material, articulation, illumination, and secondary motion.

3.4.2 Chunk-Autoregressive Long-Horizon Rendering

We extend the renderer through chunk-autoregressive generation, with bidirectional denoising within each chunk and causal conditioning across chunks. Let $I^{(n)}$, $\mathcal{C}^{(n)}$, and $M^{\mathrm{ctrl},(n)}$ denote the generated frames, target camera trajectory, and compiled controls of the n-th chunk, each spanning L frames.

The first chunk uses I_0 through the original image-to-video pathway, with zero-filled temporal history and no spatial memory conditioning. Subsequent chunks additionally receive temporal history $\mathcal{H}^{(n)}_{\text{temp}}$ and geometry-aligned spatial memory $\mathcal{H}^{(n)}_{\text{spa}}$:

$$I^{(n)} = G_{\theta,\phi,\psi}\left(\mathcal{C}^{(n)}, M^{\text{ctrl},(n)}, \mathcal{H}^{(n)}_{\text{temp}}, \mathcal{H}^{(n)}_{\text{spa}}\right), \qquad n \geq 2, \tag{12}$$

where ψ denotes the trainable parameters for cross-chunk visual conditioning. Both memory inputs use only observations available before the current chunk.

Temporal history. For subsequent chunks, we maintain a bounded latent history organized into recent, mid-range, and long-range context. Recent latents are projected into tokens at finer spatiotemporal resolution, while older latents use progressively coarser resolutions, extending context beyond a single chunk. The latent representation of I_0 is retained as a persistent anchor.

During training, history latents are constructed from preceding source-video frames and stochastically degraded through noise perturbation, feature corruption, and partial history dropping to improve robustness to imperfect context. At inference, each completed chunk contributes its generated latents to the history.

Geometry-aligned spatial memory. To retain visual context beyond the bounded temporal history, we adopt AlayaWorld's geometry-aligned spatial memory (Team et al., 2026a), supporting consistency across camera motion, occlusion, and viewpoint revisitation.

Completed RGB frames are lifted into a world-space memory using estimated depth and corresponding camera parameters. Before each new chunk, relevant past observations are retrieved, reprojected into the target views, and encoded as spatial latent tokens. The memory is updated only from completed past chunks, preserving causal generation.

4 Experiments

4.1 Experimental Setup

Training Data. We collect HUD-free gameplay videos from *Cyberpunk 2077*, *Forza Horizon 6*, and *Grand Theft Auto V*. The *Cyberpunk 2077* footage is captured from a first-person perspective, whereas *Forza Horizon 6* and *Grand Theft Auto V* are recorded from third-person perspectives using diverse camera viewpoints. The collected videos are subsequently processed using the Data Engine described in Sec. B to construct paired video-control training data.

Benchmark. We construct CombatStateBench, a controlled benchmark comprising 50 clips for evaluating whether generated videos faithfully reflect engine-maintained world states. For each scene, the Data Engine described in Sec. B reconstructs the initial 3D layout from the first frame, including the entities, their 3D boxes, semantic attributes, and camera parameters. Conditioned on this initial layout and the extracted game rules, an AI agent autonomously evolves a short combat scenario, producing a complete sequence of box-based world states. The scenarios cover diverse combinations of camera and entity motion, including interactions involving entities that initially lie outside the camera view. When a death event occurs, the target transitions to the corresponding state and remains in the scene thereafter.

Each benchmark sequence contains synchronized 3D boxes, entity states, camera parameters, projected entity controls, and instance masks. An automatic verifier checks initial-frame reprojection, metric depth consistency, box geometry, ground contact, temporal continuity, prescribed camera and entity motion, and the persistence of state transitions. Only sequences that satisfy all consistency checks are retained for evaluation.

4.2 Quantitative Results

Baselines. We compare our method with LingBot-World-V2 (Gao et al., 2026) and YUME (Mao et al., 2025b), two representative interactive video world models. We evaluate both methods using the same initial observations and benchmark transitions while retaining their native conditioning

Table 1: **Video quality and world-state evaluation on 50 CombatStateBench clips.** All values are reported as percentages. Count Accuracy is evaluated over 400 sampled frames (eight per clip), and State Accuracy over 50 death events, with three post-transition frames sampled per event.

Method	Imaging	Subject Cons.	Background Cons.	Temporal Stability	Count Acc.	State Acc.
LingBot-World-V2 (Gao et al., 2026)	67.46	81.87	91.89	96.85	40.75	8.00
YUME (Mao et al., 2025b)	64.10	92.35	93.63	98.76	32.00	58.00
Ours	**67.62**	**94.74**	**96.98**	**99.00**	**94.00**	**98.00**

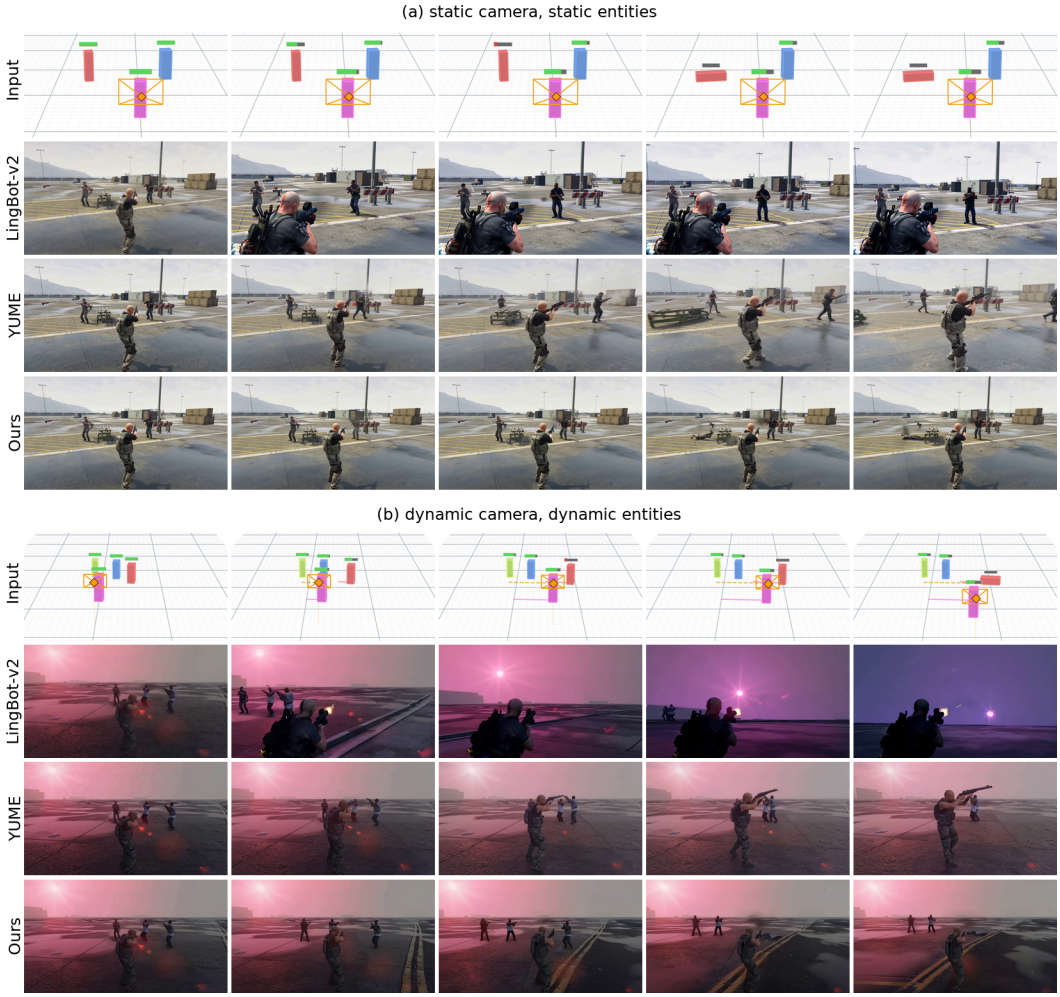


Figure 3: **Qualitative comparison of entity-death interactions over five generated frames.** We compare LingBot-World-v2 and YUME with our method under (a) a static camera and (b) a dynamic camera with moving entities. The input rows visualize the structured control signals provided to our renderer. Our method follows the specified death events while preserving the remaining entities and the scene evolution.

pathways. Since neither method exposes an external instance-level state interface for explicitly maintaining entity identities, counts, and states, we communicate state transitions through prompt switching. At each transition, the text condition is updated as

```
{environment description}. {action description}.
```
(13)

For example, when a specified NPC is killed, the action description is updated to instruct the model to render the corresponding event. These baselines allow us to evaluate whether engine-maintained state and structured spatial controls provide more reliable interaction than the implicit state representations of existing video world models.

VLM-based Evaluation. We use `Qwen3.6-27B` as the VLM judge, providing only generated RGB frames without ground-truth boxes, identities, expected counts, death locations, or other privileged annotations. To compare methods without requiring instance-level correspondence, we use two permissive global metrics of agreement with the engine state: *Count Accuracy* and *State Accuracy*.

Count Accuracy. We randomly sample eight frames per video and ask the VLM to count visibly alive characters. Predictions are compared with the corresponding engine-recorded counts:

$$\mathrm{CountAcc} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{n}_i = n_i^{\mathrm{eng}}\right], \tag{14}$$

where N is the total number of sampled frames, and $\hat{n}_i$ and n_i^{eng} are the predicted and engine-recorded counts, respectively.

State Accuracy. For each engine-recorded death event, we uniformly sample three frames after the transition. The event passes the visual check if the VLM detects at least one dead character in any sampled frame:

$$\mathrm{StateAcc} = \frac{\text{number of death events passing the visual check}}{\text{total number of death events}}. \tag{15}$$

This metric checks dead-character presence without matching the identity or location of the recorded event.

As shown in Table 1, our method achieves 94.00% Count Accuracy and 98.00% State Accuracy, exceeding LingBot-World-V2 by 53.25 and 90.00 percentage points, and YUME by 62.00 and 40.00 points, respectively. These results show stronger agreement with the engine state under both global metrics, although neither directly evaluates instance-level correspondence.

Video Quality. We evaluate perceptual and temporal quality using four VBench Huang et al. (2024) metrics: Imaging Quality, Subject Consistency, Background Consistency, and Temporal Stability. The last metric uses the temporal-flickering score, where higher values indicate less flicker.

As shown in Table 1, our method achieves the highest scores across all four metrics. Subject and background consistency improve by 12.87 and 5.09 percentage points over LingBot-World-V2, and by 2.39 and 3.35 points over YUME, respectively. These results indicate stronger entity and scene consistency while maintaining high visual quality and temporal stability.

4.3 Qualitative Results

Figure 3 compares our method with the prompt-switching baseline under the same initial observations and interaction sequences. Although the baseline can often generate a visually plausible response to an action prompt, it does not reliably preserve the resulting world state. For example, characters that should remain alive may disappear, killed characters may continue moving, and the number of visible characters may change without a corresponding engine event. In contrast, our method more faithfully reflects the engine-maintained state while retaining the visual realism of the underlying video generator. Additional qualitative results are shown in Figure 5 in Appendix C.

5 Conclusion

We introduced Programmable World Model to explore a simple premise: a generative world should not rely on visual generation alone to remember what is true about the world. By maintaining world state explicitly and using the video model primarily as a renderer, our framework makes persistent interactions more reliable without sacrificing generative flexibility. Our results show that this separation leads to substantially stronger state consistency and remains effective across long-horizon generation, unseen entities, visual styles, and interaction domains. We believe this points toward a broader direction for world models in which state is executable and verifiable, while appearance remains generative.

AI USE STATEMENT

Generative AI tools were used in the research workflow and manuscript preparation. A VLM-based coding agent generates executable world programs from user descriptions. For COMBAT-STATEBENCH, an AI agent generates sequences of box-based world states conditioned on reconstructed scene layouts and game rules. Qwen3-VL generates video captions and identifies countable object categories in the training-data processing pipeline. Qwen3.6-27B serves as an RGB-only VLM judge for entity-count and death-state evaluation. GPT Image 2 was used to generate the initial images for examples (a), (c), and (d) in Figure 5.

As described in Section 4.1, the generated benchmark sequences are filtered by an automatic verifier that checks initial-frame reprojection, metric depth consistency, box geometry, ground contact, temporal continuity, prescribed camera and entity motion, and the persistence of state transitions.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, Christos Kaplanis, Marjorie Limont, Matt McGill, Yanko Oliveira, Jack Parker-Holder, Frank Perbet, Guy Scully, Jeremy Shar, Stephen Spencer, Omer Tov, Ruben Villegas, Emma Wang, Jessica Yung, Cip Baetu, Jordi Berbel, David Bridson, Jake Bruce, Gavin Buttimore, Sarah Chakera, Bilva Chandra, Paul Collins, Alex Cullum, Bogdan Damoc, Vibha Dasagi, Maxime Gazeau, Charles Gbadamosi, Woohyun Han, Ed Hirst, Ashyana Kachra, Lucie Kerley, Kristian Kjems, Eva Knoepfel, Vika Koriakin, Jessica Lo, Cong Lu, Zeb Mehring, Alex Moufarek, Henna Nandwani, Valeria Oliveira, Fabio Pardo, Jane Park, Andrew Pierson, Ben Poole, Helen Ran, Tim Salimans, Manuel Sanchez, Igor Saprykin, Amy Shen, Sailesh Sidhwani, Duncan Smith, Joe Stanton, Hamish Tomlinson, Dimple Vijaykumar, Luyu Wang, Piers Wingfield, Nat Wong, Keyang Xu, Christopher Yew, Nick Young, Vadim Zubov, Douglas Eck, Dumitru Erhan, Koray Kavukcuoglu, Demis Hassabis, Zoubin Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025.

Ole Beisswenger, Jan-Niklas Dihlmann, and Hendrik P. A. Lensch. FrameDiffuser: G-Buffer-conditioned diffusion for neural forward frame rendering, 2025. URL https://arxiv.org/abs/2512.16670.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments, 2024. URL https://arxiv.org/abs/2402.15391.

Shengqu Cai, Duygu Ceylan, Matheus Gadelha, Chun-Hao Huang, Tuanfeng Wang, and Gordon Wetzstein. Generative rendering: Controllable 4D-guided video generation with 2D diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2312.01409.

Ziqi Cai, Siqi Yang, Yimu Wang, Zixian Gao, Yunheng Liu, Shuchen Weng, Erwin Wu, Kaipeng Zhang, and Boxin Shi. Mass: Multiplayer world models with authoritative shared state, 2026. URL https://arxiv.org/abs/2608.06257.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719.

Junhao Chen, Mingjin Chen, Henghaofan Zhang, Minglin Chen, Liaoyuan Fan, Boran Zhang, Saining Zhang, Mingze Sun, Hao Zhao, Ruqi Huang, Zhihao Li, and Yufei Wang. Video models as native 4D renderers: World-grounded conditioning from animated mesh, 2026. URL https://arxiv.org/abs/2608.00094.

Zhiyang Deng, Boran Zhang, Danze Chen, and Yeying Jin. WorldMind: Decoupled game world model for state-aware NPC behavior, 2026. URL https://arxiv.org/abs/2608.21439.

Edoardo A. Dominici, Thomas Deixelberger, Konstantinos Vardis, and Markus Steinberger. Control-DINO: Feature space conditioning for controllable image-to-video diffusion. In *European Conference on Computer Vision*, 2026. URL https://arxiv.org/abs/2604.01761.

Zicheng Duan, Jiatong Xia, Zeyu Zhang, Wenbo Zhang, Gengze Zhou, Chenhui Gou, Yefei He, Feng Chen, Xinyu Zhang, and Lingqiao Liu. LiveWorld: Simulating out-of-sight dynamics in generative video world models, 2026. URL https://arxiv.org/abs/2603.07145.

Zixun Fang, Yawen Shao, Kai Zhu, Jie Xiao, Shihan Chen, Yu Liu, Xueyang Fu, Yang Cao, Wei Zhai, and Zheng-Jun Zha. Code plans, diffusion renders: Open-ended generative world modeling, 2026. URL https://arxiv.org/abs/2609.26458.

Zelin Gao, Qiuyu Wang, Jiapeng Zhu, Jingye Chen, Zichen Liu, Qingyan Bai, Jiahao Wang, Yufeng Yuan, Hanlin Wang, Yichong Lu, et al. Infinite worlds with versatile interactions. *arXiv preprint arXiv:2607.07534*, 2026.

Gonzalo Gomez-Nogales, Yicong Hong, Chongjian Ge, Peiye Zhuang, Marc Comino-Trinidad, Dan Casas, and Yi Zhou. Coarse-to-real: Generative rendering for populated dynamic scenes. *arXiv preprint arXiv:2601.22301*, 2026.

Bohai Gu, Taiyi Wu, Yueyang Yuan, Jian Liu, Xiaocheng Lu, Dazhao Du, Jie Zhang, Jinxiang Lai, Shuai Yang, Xiaotong Zhao, Alan Zhao, and Song Guo. WorldCraft: From camera navigation to object manipulation in interactive video world models, 2026a. URL https://arxiv.org/abs/2605.25077.

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, Wenping Wang, and Yuan Liu. Diffusion as shader: 3D-aware video diffusion for versatile video generation control. In *ACM SIGGRAPH 2025 Conference Papers*, 2025. doi: 10.1145/3721238.3730607. URL https://arxiv.org/abs/2501.03847.

Zekai Gu, Shuoxuan Feng, Yansong Wang, Hanzhuo Huang, Zhongshuo Du, Chengfeng Zhao, Chengwei Ren, Peng Wang, and Yuan Liu. GO-Renderer: Generative object rendering with 3D-aware controllable video diffusion models, 2026b. URL https://arxiv.org/abs/2603.23246.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv: 2501.00103*, 2024.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Size Wu, Wei Li, Xuchen Song, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 2.0: An open-source real-time and streaming interactive world model, 2025. URL https://arxiv.org/abs/2508.13009.

Yicong Hong, Yiqun Mei, Chongjian Ge, Yiran Xu, Yang Zhou, Sai Bi, Yannick Hold-Geoffroy, Mike Roberts, Matthew Fisher, Eli Shechtman, Kalyan Sunkavalli, Feng Liu, Zhengqi Li, and Hao Tan. Relic: Interactive video world model with long-horizon memory, 2025. URL https://arxiv.org/abs/2512.04040.

Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, et al. Vipe: Video pose engine for 3d geometric perception. *arXiv preprint arXiv:2508.10934*, 2025.

Weikai Huang, Jieyu Zhang, Sijun Li, Jiafei Duan, Yunqian Cheng, Jaemin Cho, Matthew Wallingford, Rustin Soraki, Chris Dongjoo Kim, Shuo Liu, et al. Wilddet3d: Scaling promptable 3d detection in the wild. 2026a.

Zheng-Hui Huang, Zhixiang Wang, Jiaming Tan, Ruihan Yu, Yidan Zhang, Bo Zheng, Yu-Lun Liu, Yung-Yu Chuang, and Kaipeng Zhang. Generative world renderer. *arXiv preprint arXiv:2604.02329*, 2026b.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Geonung Kim, Janghyeok Han, and Sunghyun Cho. VideoFrom3D: 3D scene video generation via complementary image and video diffusion models. In *SIGGRAPH Asia 2025 Conference Papers*, 2025. doi: 10.1145/3757377.3763871. URL https://arxiv.org/abs/2509.17985.

Jiaqi Li, Junshu Tang, Zhiyong Xu, Longhuang Wu, Yuan Zhou, Shuai Shao, Tianbao Yu, Zhiguo Cao, and Qinglin Lu. Hunyuan-GameCraft: High-dynamic interactive game video generation with hybrid history condition, 2025a. URL https://arxiv.org/abs/2506.17201.

Runjia Li, Philip Torr, Andrea Vedaldi, and Tomas Jakab. VMem: Consistent interactive video scene generation with surfel-indexed view memory, 2025b. URL https://arxiv.org/abs/2506.18903.

Zhen Li, Zian Meng, Shuwei Shi, Wenshuo Peng, Yuwei Wu, Bo Zheng, Chuanhao Li, and Kaipeng Zhang. WildWorld: A large-scale dataset for dynamic world modeling with actions and explicit state toward generative ARPG, 2026. URL https://arxiv.org/abs/2603.23497.

Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Chih-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, et al. Diffusionrenderer: Neural inverse and forward rendering with video diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26069–26080. IEEE, 2025.

Guixu Lin, Zheng-Hui Huang, Siqi Yang, Ming-Hsuan Yang, Kaipeng Zhang, and Zhixiang Wang. Generative world renderer at the speed of play, 2026a. URL https://arxiv.org/abs/2607.18703.

Zijun Lin, Zeqing Wang, Cheston Tan, Bihan Wen, and Yeying Jin. Stateplay: State-aware game world models for mechanics-consistent generation, 2026b. URL https://arxiv.org/abs/2607.26754.

Xiaofeng Mao, Zhen Li, Chuanhao Li, Xiaojie Xu, Kaining Ying, Tong He, Jiangmiao Pang, Yu Qiao, and Kaipeng Zhang. Yume-1.5: A text-controlled interactive world generation model, 2025a. URL https://arxiv.org/abs/2512.22096.

Xiaofeng Mao, Shaoheng Lin, Zhen Li, Chuanhao Li, Wenshuo Peng, Tong He, Jiangmiao Pang, Mingmin Chi, Yu Qiao, and Kaipeng Zhang. Yume: An interactive world generation model. *arXiv preprint arXiv:2507.17744*, 2025b.

Felix O'Mahony, Roberto Cipolla, and Ayush Tewari. VDAWorld: World modelling via VLM-directed abstraction and simulation, 2025. URL https://arxiv.org/abs/2512.11061.

Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models, 2025. URL https://arxiv.org/abs/2505.20171.

Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. GEN3C: 3D-informed world-consistent video generation with precise camera control, 2025. URL https://arxiv.org/abs/2503.03751.

Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling. *arXiv preprint arXiv:2512.14614*, 2025.

AlayaWorld Team, Kaipeng Zhang, Chuanhao Li, Yifan Zhan, Yongtao Ge, Yuanyang Yin, Jiaming Tan, Kang He, Liaoyuan Fan, Mingliang Zhai, et al. Alayaworld: Interactive long-horizon world modeling–full technical report. *arXiv preprint arXiv:2607.18367*, 2026a.

Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, et al. Advancing open-source world models. *arXiv preprint arXiv:2601.20540*, 2026b.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines, 2024. URL https://arxiv.org/abs/2408.14837.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv: 2503.20314*, 2025.

Zile Wang, Zexiang Liu, Jiaxing Li, Kaichen Huang, Baixin Xu, Fei Kang, Mengyin An, Peiyu Wang, Biao Jiang, Yichen Wei, Yidan Xietian, Jiangbo Pei, Liang Hu, Boyi Jiang, Hua Xue, Zidong Wang, Haofeng Sun, Wei Li, Wanli Ouyang, Xianglong He, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory, 2026. URL https://arxiv.org/abs/2604.08995.

Tong Wu, Shuai Yang, Ryan Po, Yinghao Xu, Ziwei Liu, Dahua Lin, and Gordon Wetzstein. Video world models with long-term spatial memory, 2025. URL https://arxiv.org/abs/2506.05284.

Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. WorldMem: Long-term consistent world simulation with memory, 2025. URL https://arxiv.org/abs/2504.12369.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. LongLive: Real-time interactive long video generation, 2025a. URL https://arxiv.org/abs/2509.22622.

Xindi Yang, Baolu Li, Yiming Zhang, Zhenfei Yin, Lei Bai, Liqian Ma, Zhiyong Wang, Jianfei Cai, Tien-Tsin Wong, Huchuan Lu, et al. Vlipp: Towards physically plausible video generation with vision and language informed physical prior. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12360–12370. IEEE, 2025b.

Jung Yi, Minjae Kim, Paul Hyunbin Cho, Wooseok Jang, Sangdoo Yun, and Seungryong Kim. Worldkv: Efficient world memory with world retrieval and compression, 2026. URL https://arxiv.org/abs/2605.22718.

Jiwen Yu, Jianhong Bai, Yiran Qin, Quande Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. Context as memory: Scene-consistent interactive long video generation with memory retrieval, 2025a. URL https://arxiv.org/abs/2506.03141.

Jiwen Yu, Yiran Qin, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. GameFactory: Creating new games with generative interactive videos, 2025b. URL https://arxiv.org/abs/2501.08325.

Chong Zeng, Yue Dong, Pieter Peers, Hongzhi Wu, and Xin Tong. Renderformer: Transformer-based neural rendering of triangle meshes with global illumination. In *ACM SIGGRAPH 2025 Conference Papers*, 2025.

Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. RGB↔X: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*, 2024. doi: 10.1145/3641519.3657445. URL https://arxiv.org/abs/2405.00666.

Zheng Zeng, Marco Salvi, Lifan Wu, Jan Novák, Daqi Lin, Saeed Hadadan, Yichen Sheng, Robert Pottorff, Shiqiu Liu, Ravi Ramamoorthi, Ling-Qi Yan, and Miloš Hašan. RGBX-Next: Towards realistic generative rendering from G-Buffers, 2026. URL https://arxiv.org/abs/2608.13929.

A REPRESENTATION TRADE-OFFS

The choice of representation shapes what a system can explicitly control, the cost of achieving that control, and the potential mismatch between representations extracted from training observations and those constructed programmatically at inference time. As illustrated in Figure 4, candidate representations span a spectrum from lightweight text descriptions to detailed 3D structures. Increasing structural detail enables finer-grained control over entity geometry, spatial relationships, and their evolution over time. However, richer structural supervision is more costly to obtain, and more structural degrees of freedom must be explicitly maintained and evolved during inference. More importantly, representations extracted from observed dynamics during training may differ from those constructed programmatically from high-level state transitions at inference time, creating a potential training–inference mismatch. We therefore seek an intermediate level of abstraction that supports explicit controllability and training–inference alignment while keeping training-data acquisition and inference-time evolution tractable.

Lightweight representations offer limited explicit control over world-space geometry. Text descriptions can convey entity categories, attributes, and high-level states, but do not by themselves impose geometric constraints on an entity's position, extent, or orientation in a shared world coordinate system. 2D bounding boxes and masks provide more direct spatial control, but operate in image space: their positions, scales, and visible regions change with the camera viewpoint. These representations therefore constrain individual observations without defining an underlying world-space structure that can be deterministically reprojected across views. For a programmable world, it is more natural to specify entity geometry in a shared world coordinate system and derive its projection in each observation from the target camera.

At the other end of the spectrum, greater structural detail does not necessarily yield a more suitable representation. Complete 3D scenes, articulated entity models, and dynamic 3D geometry encode finer-grained structure than entity-level bounding boxes, while G-buffers provide dense, view-dependent surface information. Such detail supports more precise control but also introduces additional costs. First, training requires richer and more accurate structural supervision, increasing data acquisition and processing costs and making it harder to scale to open-domain settings. Second, the system must explicitly specify and evolve more structural degrees of freedom during inference, taking responsibility for geometric and dynamic details that a coarser representation would leave to the generative model.

Consider a character transitioning from standing to falling. A coarse representation may only need to describe changes in global position, orientation, and spatial occupancy, whereas a complete articulated or dynamic 3D representation must additionally characterize high-dimensional structural changes such as body pose, limb motion, and local deformation. As the representation becomes more detailed, the system gradually moves from specifying *what changes in the world* to specifying *how that change unfolds geometrically and dynamically*. In conventional 3D environments equipped with complete animation systems or physics simulators, such details can be produced by existing simulation mechanisms. Our focus, however, is on open-domain programmable generative worlds, where we aim to rely on generative models to realize visual dynamics rather than reconstruct a complete 3D animation and simulation stack.

Under this setting, the same representation is also obtained through fundamentally different paths during training and inference. In the training data, the dynamic process has already been realized, and the corresponding structural representation therefore describes an already realized evolution:

$$\text{Training: realized dynamics} \longrightarrow \text{structural representation.} \tag{16}$$

Inference proceeds in the opposite direction. The system first receives a high-level state transition and must actively produce the corresponding time-varying structural representation before the visual outcome is generated. The generative model then realizes these structural constraints as visual dynamics:

$$\text{Inference: state transition} \longrightarrow \text{structural representation} \longrightarrow \text{generated dynamics.} \tag{17}$$

This training-inference asymmetry means that a representation that can be obtained during training is not necessarily equally easy to produce and evolve at inference time. Finer-grained representations provide more precise explicit control, but also require the system to determine more structural

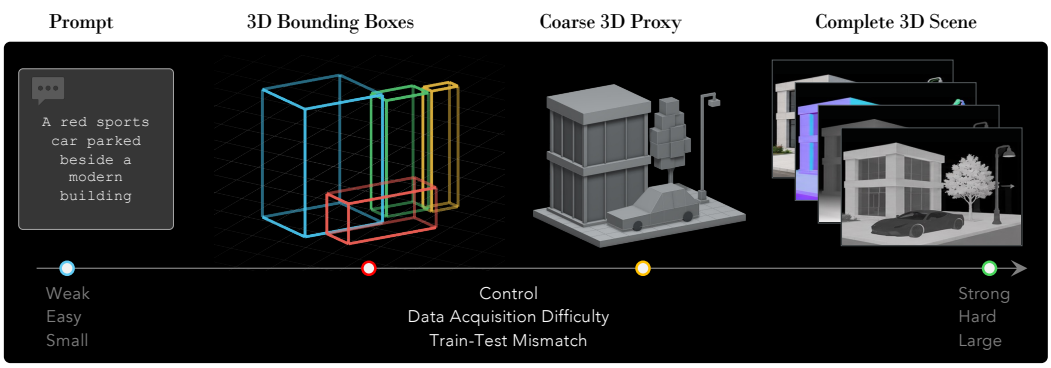


Figure 4: **State representation trade-offs.** Candidate representations range from lightweight semantic descriptions to detailed geometric structures Huang et al. (2026b). Greater structural detail enables finer-grained control but increases the cost of training-data annotation and inference-time state construction and evolution. Moreover, representations extracted from observations or recorded from engines during training may differ from those constructed programmatically at inference time, creating a potential training–inference mismatch. We adopt state-augmented 3D OBBs as an intermediate abstraction that supports explicit world-space control while leaving fine geometry, articulation, and secondary visual dynamics to the generative renderer.

degrees of freedom and their temporal evolution from high-level state transitions. As the representation approaches complete articulated motion or dynamic 3D geometry, this process increasingly resembles the high-dimensional dynamic realization problem addressed by conventional animation, motion generation, or physical simulation.

From this perspective, representation choice determines the boundary between *explicit structural control* and *generative dynamic completion*. A representation that is too weak leaves entity positions, world-space structure, and state-dependent changes that should be explicitly constrained to the generative model. A representation that is too strong not only requires more demanding training supervision, but also forces the system to explicitly determine large amounts of low-level geometric and dynamic detail at inference time. Pretrained video models already provide strong visual and dynamic priors for completing precise shape, texture, material, articulation, local deformation, illumination, and secondary motion. We therefore explicitly represent only the structure necessary for entity persistence, world-space consistency, and state-dependent changes, while leaving finer-grained dynamic realization to the generative model.

Following this principle, we adopt *state-augmented 3D oriented bounding boxes (OBBs)* as an intermediate representation. A 3D OBB specifies an entity's position, extent, and orientation in a shared world coordinate system, and is associated with persistent identity as well as relevant semantic and dynamic states. Compared with text, 2D bounding boxes, and masks, OBBs provide a view-independent and reprojectable world-space scaffold. Compared with more explicit representations such as G-buffers, complete 3D scenes, articulated models, or dynamic 3D geometry, they restrict the structure that must be explicitly represented and evolved to a compact entity-level space. Time-varying OBBs can express coarse structural changes in entity position, orientation, and spatial occupancy, while internal articulation, local deformation, and other fine-grained dynamics are completed by the generative renderer under these constraints.

Representation choice shapes explicit controllability, the cost of constructing and evolving world state, and the potential mismatch between training and inference conditions. As illustrated in Figure 4, text descriptions are easy to author but do not by themselves impose geometric or other dense constraints, while 2D boxes and masks provide image-space control tied to particular viewpoints (Yang et al., 2025b). Detailed 3D scenes and structural proxies (Gomez-Nogales et al., 2026), together with dense rendering conditions such as G-buffers (Huang et al., 2026b; Liang et al., 2025), enable finer-grained control but demand richer supervision and more complex inference-time construction. Crucially, training representations are extracted from observed dynamics, whereas inference requires constructing structural trajectories from desired state transitions. Specifying that a character falls, for example, is simpler than generating its detailed body poses and limb trajectories.

More detailed representations therefore require the explicit system to resolve additional motion and geometry, and the resulting controls may differ from those extracted during training.

B Automatic Training Data Pipeline

Training our model requires structured annotations of camera geometry, persistent instance identities, semantic information, and object motion. Since such annotations are rarely available for in-the-wild videos, we develop an automatic data engine that recovers camera parameters, semantic labels, instance tracks, and object trajectories from unlabeled videos. These annotations are subsequently converted into camera-aligned conditioning maps, while the estimated camera trajectory is provided separately to the renderer.

Camera-parameter estimation. Given an input video $\mathcal{V}$, we use ViPE Huang et al. (2025) to estimate the camera intrinsics, camera poses, and metric depth map for each frame. The depth and intrinsics allow image-space observations to be lifted into 3D, while the camera poses establish a shared world coordinate system across frames. Together, they support per-frame 3D object annotation and compensate for camera motion when recovering object trajectories.

Semantic-category discovery. An agent powered by Qwen3-VL Bai et al. (2025) analyzes each video to generate a global caption describing its overall content and identify the discrete and countable object categories relevant to the scene. The resulting category set defines the semantic vocabulary used for subsequent instance segmentation. It can also be replaced with a predefined vocabulary when constructing a domain-specific dataset.

Instance segmentation and tracking. Given the discovered category set, SAM3 Carion et al. (2025) performs video instance segmentation and tracking, producing temporally associated instance masks, 2D bounding boxes, semantic labels, and persistent track identities. Each instance inherits the semantic category used to prompt SAM3, allowing objects of the same category to share a semantic label while retaining distinct track identities. Masks with insufficient visible area are discarded to suppress unreliable annotations caused by extremely small or heavily occluded objects.

Object-trajectory recovery. For each visible instance, we provide its tracked 2D bounding box to WildDet3D Huang et al. (2026a), together with the corresponding RGB frame, metric depth map, and camera intrinsics, to estimate a per-frame 3D oriented bounding box:

$$b_i^t = \left(\mathbf{p}_i^t, \mathbf{d}_i^t, \mathbf{R}_i^t, \ell_i, k_i\right), \tag{18}$$

where $\mathbf{p}_i^t$, $\mathbf{d}_i^t$, and $\mathbf{R}_i^t$ denote the box center, dimensions, and orientation, respectively. The semantic label ℓ_i specifies the object category, while k_i associates the box with its persistent instance track. Using the estimated camera poses, we transform the per-frame boxes into a shared world coordinate system. Boxes associated with the same track identity then form a temporally consistent object trajectory.

We estimate object motion from the displacement between consecutive world-space box centers along each trajectory. Computing the displacement in the shared world coordinate system removes apparent motion caused by camera movement. We then rotate the displacement into the current camera coordinate system and quantize it into seven states: static, left, right, up, down, forward, and backward. A small displacement threshold is used to identify static objects.

Conditioning-map generation. Finally, we project the estimated 3D OBBs into each frame using the corresponding camera parameters. The rasterization produces identity, semantic, and direction maps, with a z-buffer resolving visibility when multiple projected boxes overlap. Based on the persistent track identities, semantic labels, and quantized motion states, we further construct the identity, semantic, and object-motion maps used to condition our model.

C Additional Qualitative Results

Figure 5 presents qualitative results across diverse visual styles, camera motions, object categories, and interaction dynamics. For (b) and (e), we sample first frames from GTA V data that are unseen

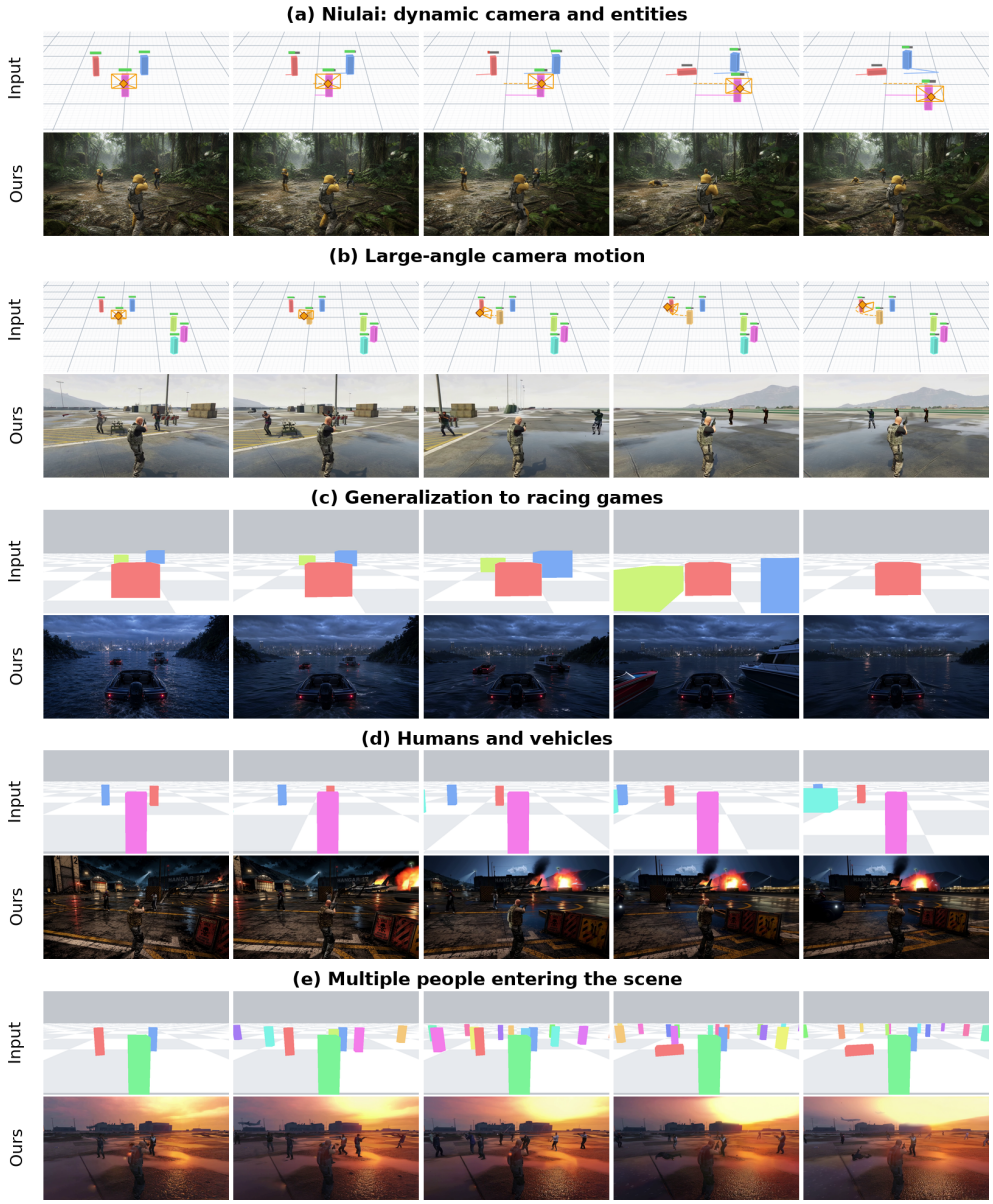


Figure 5: **Additional qualitative results shown at five selected time steps.** (a) Generalization to a novel minotaur scene with dynamic camera and entity motions. (b) Under a large-angle camera rotation, the model preserves scene consistency and correctly reveals characters located behind the initial camera view, following the persistent world state maintained by the engine. (c) Generalization to a racing game with moving cameras and vehicles. (d) Joint rendering of heterogeneous object categories, including humans and vehicles. (e) Selected frames from an 897-frame autoregressive sequence in which many NPCs progressively enter the scene. The input rows visualize the global scene state in (a)–(b) and camera-view 3D OBB maps with ground-grid references in (c)–(e). The generated results consistently follow the specified camera motion, spatial layout, and entity states.

during training, whereas the first frames in (a), (c), and (d) are generated using GPT Image 2. In (a) and (b), the control signals are visualized in the global scene coordinate system, explicitly showing the engine-maintained entity states, spatial layout, and camera motion. In (c)–(e), we instead visualize the camera-view 3D OBB maps together with a ground-grid reference. Across both representations, the generated videos closely follow the specified camera motion and object configurations. Specifically, (a) depicts a minotaur scene with dynamic characters, demonstrating generalization to

novel character appearances and environments. In (b), the camera undergoes a large-angle rotation, revealing three characters initially placed behind the first-frame view. The model preserves the scene structure throughout the camera motion and correctly renders these previously unobserved entities according to the persistent world state maintained by the engine. Example (c) extends the system to a racing scenario, showing that the proposed framework can support a different game genre rather than overfitting to a particular game or type of interaction. Example (d) demonstrates the joint rendering of heterogeneous object categories, including humans and vehicles, within the same scene. Finally, (e) shows an autoregressive sequence in which many NPCs progressively enter the scene. Despite the increasing number of simultaneously visible entities, the model maintains reasonable visual and temporal stability, suggesting that the system can support more complex and evolving worlds.