BEYOND PIXELS: GROUNDING VIDEO WORLD MODELS WITH STATE-CENTRIC INFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Video world models are trained under the paradigm of predicting future video frames from a stream of input actions, with the hope of unlocking fast, interactive and physics-accurate simulation environments from video data only. Many state-of-the-art world models rely on off-the-shelf VAE encoders of text-to-video generative models, thus inheriting from their ad-hoc latent space, which is also hard to interpret or control, and is not built with the task of world modeling in mind. In contrast, in settings where hard-coded physics simulations are standard (e.g. autonomous driving or robotics), world models often rely on explicit 3D, sensor information or dedicated logic engines. In this work, we study how privileged state information can be used alongside video data to unlock new properties of video world models: We first show that a modular training recipe enables improved efficiency, while modeling both the states and video streams jointly induces a natural measure of uncertainty of the full world state, which may only be partially observable in the video. We also exploit the structure of the state space to infuse LLM priors during training as a form of data augmentation to correct policy bias or introduce novel actions. Finally, we design state-based metrics for the Rocket Science dataset to evaluate the world model's physics understanding.

1 INTRODUCTION

World models are rapidly arising as a mean to build general intelligence from visual signal, departing from the language-based understanding of vision-language models: Given an action and a world state, typically input as a video frame, a world model's task is to predict the next world state. A core appeal of this paradigm is that training world models only requires raw videos with actions sequences as inputs, no dense language annotations nor task-specific physics simulations, etc. The growing amount of video data available has unlocked world modeling for several applications such as navigation (Bar et al., 2025; Hafner et al., 2024), robotics (Yang et al., 2024; Hou et al., 2026) and video games (Bruce et al., 2024; AlayaWorld Team et al., 2026). Nevertheless, video world models exhibit two drawbacks: **First,** they rely on a learned autoencoder to build a latent space in which the transition model can then operate. Thus training a world model is often a complex two-stage process which is inherently suboptimal as the encoder is typically trained for reconstruction: There is no guarantee that the resulting latents are amenable to world modeling. In fact, defining the properties of a good latent space for world model is an open problem (Maes et al., 2026; Garrido et al., 2026) and the literature displays a wide range of solutions such as VLMs (PAN Team et al., 2026), generative video models encoder (Shin et al., 2026b; Zhang et al., 2026a) or representation autoencoders (Zheng et al., 2026; Porcher et al., 2026). **Second** the resulting latent space is neither interpretable nor controllable as-is, as they lack any prior that language-based modeling could bring. This also makes evaluating world models difficult as we cannot easily assess their understanding of physics or other world properties from pixels: The only reliable evaluation becomes training a policy in the world model and evaluating its performance, which adds another complex phase to the training pipeline. To address those issues, a growing body of work is advocating for state-centric world models, incorporating explicit structure in the latent space in the form of 3D representations (Liu et al., 2024; Meng et al., 2026b) or game states (Menapace et al., 2024; Lin et al., 2026) for example, which can later be controllable by a human or program (Shang et al., 2026).

In this work, we build on MIRA (Hu et al., 2026), a recent real-time multiplayer world model of the highly dynamic environment of the game Rocket League. We leverage game state information

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.

across four different axes to improve performance on modeling, efficiency, controllability and interpretability. To foster reproducibility, we also plan to release our models and newly generated synthetic datasets for this task. Our contributions are as follows:

- **Modeling;** We explore two paradigms to combine explicit states with video streams: First, a state-centric world model that decouples state transitions from rendering. Second, we propose a dual-stream model which jointly trains states and pixel frames as two separate flow matching processes which communicate via cross-attention. We show through varied experiments that these two models enable different trade-off in terms of performance and controllability.
- **Efficiency;** Decoupling pixel rendering from learning state transitions allows for modeling the dynamics of the world with a much smaller model with little loss of performance. This enables faster training as we only need to train the transition model and faster inference at large batch sizes.
- **Interpretability and Evaluation;** Evaluating the generated video for physics grounding or temporal consistency of objects is difficult and often requires extracting side information such as 3D or depth maps (Upadhyay et al., 2026). Modeling the states allows us to define explicit metrics in that space, with a clear interpretation on how they contribute to the plausibility of the current world.
- **Controllability;** Foundation models exhibit growing knowledge of the world such as 3D modeling or understanding of physics (Dai et al., 2026; Zhang et al., 2026b). We show that they can be used to guide synthetic data generation in our simple structured game state, enabling novel action synthesis as well as data augmentations to combat policy bias.

2 Related Work

The term *world model* was popularized by Ha & Schmidhuber (2018), who learn a latent transition model of an environment and train a controller entirely inside its imagined rollouts. Its usage has since broadened to predicting in pixel space (Brooks et al., 2024), in a learned latent space (Hafner et al., 2024), or in a structured state space (Meng et al., 2026a). This work spans all three categories.

Video World Models as Game Simulators. A large body of work study world models in the video games domain, by learning to render the game conditioned on player input actions (Bruce et al., 2024; Alonso et al., 2024; Decart & Etched, 2024; Li et al., 2025; Robbyant Team et al., 2026; Wang et al., 2026b). In this regime, efficiency is often a key constraint as the end model can be used as either an interactive simulation for human users (real-time constraint) or a simulation environment to train policies (the faster the better). Hence recent efforts mostly target real-time generation and long-horizon consistency (Robbyant Team et al., 2026; Wang et al., 2026b; Qian et al., 2026), and multiplayer settings such as two-player Minecraft (Savva et al., 2026) or four-player Rocket League (Hu et al., 2026). In all these models, the game state (e.g. positions, scores) is implicitly captured in the latent space, which makes it hard to inspect, edit or evaluate.

State-based World Models. Several works make the state of the world explicit, similar to our SCWM, but differ in how much of the state they model, or how they predict and render it. For instance in Magpie (Zhan et al., 2026), the dynamics are not learned at all: A conventional game engine runs the state transitions; Only the rendering is generative. Another line of work relies on dedicated 3D machinery: Promptable Game Models (Menapace et al., 2023) render player poses with a compositional NeRF, and Marionette (Meng et al., 2026a) predicts 3D skeletons, converts them to geometry with a handcrafted renderer, then paints the appearance with video diffusion. Both are tied to the geometry of their domain, while our renderer is a generic network that decodes frames directly from a low-dimensional state vector. Other works make only part of the state explicit and leave the rest to a pixel-based world model, e.g. the state needed for NPC decisions (Deng et al., 2026), scores and game events (Chen et al., 2025a), a global state used to retrieve past frames (Chen et al., 2025b), or an editable level map (Po et al., 2026). Closest to our SCWM is a recent commercial model, Agora-1 (Odyssey, 2026), which learns the dynamics of the full internal state of a multiplayer shooter (e.g. positions, health) and renders it with a separate diffusion model. Finally, explicit states also make world models easier to evaluate: Shin et al. (2026a) use them to show that video world models lose track of hidden state over long horizons, even when their outputs look plausible.

World-action and multi-stream Models. Our DSWM models video and game states as two time-aligned streams, a design shared with world-action models (WAMs), which predict future ac-

tions jointly with future frames, integrating the world model with a policy (Kim et al., 2026; Zhu et al., 2025; Wu et al., 2024). In the taxonomy of Wang et al. (2026a), our DSWM falls under joint diffusion-based multi-stream models such as UWM, except its second stream models game states rather than actions. Parallel streams are also used outside of video and action conditioning for instance for joint audio-video generation in LTX-2 (HaCohen et al., 2026), speech-text models (Défossez et al., 2024), jointly generating video with motion (Chefer et al., 2025).

3 Grounding a world model with state-centric information



Figure 1: **A SCWM rollout.** The transition model generates states (**bottom**) while the neural renderer generate video rollouts (**top**) conditioned on the state stream and past video frames.

We study the problem of modeling video paired with a stream of corresponding explicit physical state, conditioned on actions. In our experimental setting, we use the Rocket Science dataset (Hu et al., 2026) consisting of clips from the game Rocket League. The videos are obtained by pitting 4 Nexto bots (Rolv-Arild, 2021) against each other in 2v2 matches. The game state is a 117-dimensional vector and includes the position, velocity, and orientation of each car and the ball (see Appendix A). Notably, the video stream is only a partial view of the game state as all information may not be visible from the camera of the *ego car*, i.e. the player whose view is recorded.

Formally, given an input video with frames extracted at a fixed interval, $x_{1:t}$ and corresponding physical states $s_{1:t}$, our goal is to learn a *world model* predicting the next frame and state in the sequence given an input action from one player (the ego car) $p(x_{t+1}, s_{t+1} \mid x_{\leq t}, s_{\leq t}, a_{\leq t})$. Under these notations, we distinguish three types of world models:

Video World Model (VWM). A VWM only models video, without using explicit states. The prevalent approach is to learn world dynamics directly in a learned latent space:

$$z_0 = \mathrm{enc}(x_0), \qquad z_{t+1} \sim w_V(\cdot \mid z_{\leq t}, a_t), \qquad x_{t+1} = \mathrm{dec}(z_{t+1}) \tag{1}$$

The model is trained in two stages: First, train the codec ($\mathrm{enc}(\cdot)$ and $\mathrm{dec}(\cdot)$) as an autoencoder. Then, freeze it and train the latent world model w_V.

State-Centric World Model (SCWM) (Section 3.1). In this approach, we explicitly separate the task of modeling pixels from modeling the dynamics of the world in the known explicit state space:

$$s_{t+1} \sim w_T(\cdot \mid s_{\leq t}, a_t), \qquad z_{t+1} \sim r(\cdot \mid z_{\leq t}, s_{t+1}), \qquad x_{t+1} = \mathrm{dec}(z_{t+1}) \tag{2}$$

where w_T is a transition model predicting the dynamics directly in the state space, r is a learned renderer mapping a state to a frame given past latent frames, and $\mathrm{dec}(\cdot)$ is defined as in Equation (1). At training, r sees latent frames and states, while w_T only sees states and actions. At inference, we only assume a short prompt $s_0 \dots s_n$ is known; w_T then generates all subsequent states in the sequence while r renders them to pixels.

Dual-Stream World Model (DSWM) (Section 3.2). Building on the video world model, we introduce a dual-stream world model where the states and frames are modeled as two generative processes that are trained jointly and communicate via cross-attention:

$$z_t = \mathrm{enc}(x_t), \qquad (s_{t+1}, z_{t+1}) \sim w_D(\cdot, \cdot \mid s_t, z_t, a_t), \qquad x_{t+1} = \mathrm{dec}(z_{t+1}) \tag{3}$$

where $\mathrm{enc}(\cdot)$ and $\mathrm{dec}(\cdot)$ are the same as in Equation (1).

We follow MIRA's architecture choices for the video world model: First, for both DSWM and VWM we use DINOv3 (Siméoni et al., 2026) with a light projector head as the encoder enc, while the decoder dec is trained from scratch. The resulting encoder yields a $32\times$ spatial and $2\times$ temporal compression rate: A 512×288 video at 20 fps is encoded as 16×9 latents at 10 fps. Second, the VWM is a flow matching model (Lipman et al., 2023) trained using diffusion forcing (Chen et al., 2024). Unlike MIRA, we train on single-view data rather than four player views: In single-view, the input and output spaces become smaller, but the amount of hidden information increases, as the other cars are not visible on the video stream most of the time, increasing the difficulty of the task.

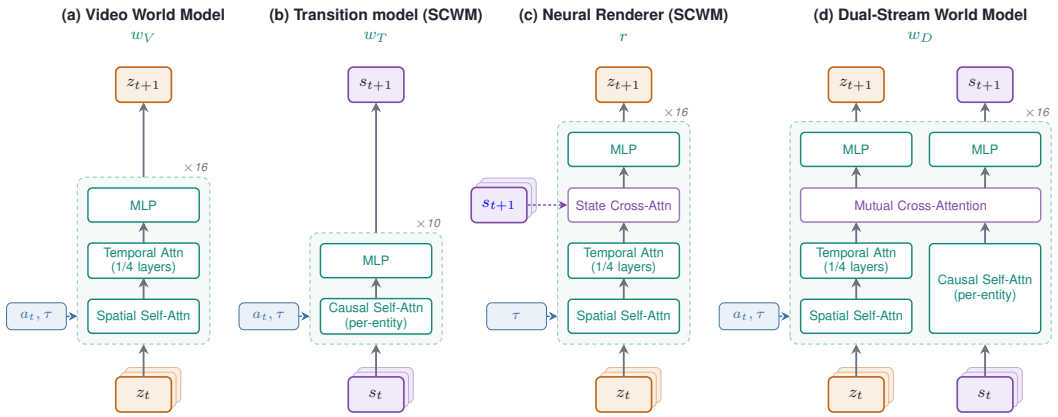


Figure 2: **Illustration of the inner transformer mechanisms for all models.** (a) In VWM, spatial attention (within one frame) and temporal attention (over a sliding window of past frames) are decoupled for efficiency purposes. The player action is fed via AdaLN in the spatial self-attention. For SCWM, the transition model (b) only models temporal attention over each dimension of the state independently and is also action-conditioned via AdaLN; The neural renderer (c) uses the same spatial-temporal decoupling and a dedicated cross-attention layer for conditioning on the state. (d) Finally, DSWM models both states and frames as two parallel streams interacting via cross-attention.

3.1 State-Centric World Model

SCWM decouples the rendering process from modeling the state dynamics. There are two core challenges: **(i)** For the rendering, some aspects may not be contained in the state (e.g. appearance of background decor) which raises the questions of how to model this uncertainty while preserving temporal coherence of the video; **(ii)** For world modeling, it is not clear what is the optimal way to encode the state information into a single latent and how to inject it in the transition model.

Modeling transitions in the state space (Figure 2 (b)). The transition model w_T is trained using flow matching; It takes as input the current state s_t and predicts the *difference* $s_{t+1} - s_t$: As shown in Section 4.1, we find that this yields better performance than directly predicting s_{t+1}. We first normalize the state difference vector using component-wise statistics computed on the training dataset. Flow matching noise is then applied to this normalized vector, and a small module tokenizes the vector into four tokens, one per car. Each token carries information about the state of its respective car, the global state of the game and the state of the ball. Finally, a prediction head decodes each car token independently into the corresponding states. The ball state and the global score are independently decoded from each car token and averaged to obtain the final prediction (Appendix C). The sequence of states is then fed into a DiT (Peebles & Xie, 2023) conditioned on actions via AdaLN-Zero. As shown in Section 4.1, teacher forcing performs better than diffusion forcing for training the transition model. This means that, in each training step, we concatenate the clean states $s_{1:T}$ and the noised states $\delta^{\tau}_{1:T}$, where $\delta_t = s_t - s_{t-1}$ is the normalized state difference and δ^{τ}_t is its noised version at flow matching time τ. Each timestep δ^{τ}_t attends to the clean past $s_{1:t-1}$ and to itself.

Neural renderer (Figure 2 (c)). The renderer r is a flow-matching model trained with diffusion forcing to generate a frame, conditioned on the game state and past frames. For videos, we work in the latent space of DINOv3, same as for our VWM baseline. To tokenize the states, the neural renderer uses a slightly different module than the transition model, yielding seven tokens per timestep:

Four for cars, one for the ball, and two for global information. The conditioning on states then enters the renderer via cross-attention in every block, while the diffusion timestep is fed through AdaLN-Zero. As in the VWM, self-attention is decoupled as spatial attention within one frame, and causal temporal attention over a sliding context window. At inference time, the neural renderer takes as input the state trajectories predicted by the transition model. We also apply RoPE-like positional embedding to the keys of the state tokens during cross-attention. See Appendix C.3 and Table 2.

3.2 Dual-Stream World Model

DSWM (Figure 2 (d)) can be seen as an end-to-end combination of the state-centric world model and its renderer, trained with diffusion forcing to jointly denoise a sequence of (state, frame) pairs. The video and states are processed in separate streams and communicate via cross-attention layers.

Encoding the states. Like the transition model of SCWM, the state stream predicts state differences $s_{t+1} - s_t$. For its input, we investigate two designs: Absolute state s_t, like in SCWM, or keeping the relative formulation throughout the stream, $s_t - s_{t-1}$. In the latter case, we also provide the model with the clean past state s_{t-1} as *anchor state* in order to be able to recover the absolute state. The anchor approach can be thought of as a compressed version of the teacher-forcing used in the transition model, except the noisy state at step t is co-located with the clean state for step $t-1$ rather than being separate. As shown in Table 3, the relative approach outperforms the absolute one.

Diffusion vs Teacher forcing. We train using diffusion forcing rather than teacher forcing, as the additional memory cost incurred by teacher forcing is prohibitive for the larger DSWM model. Therefore the model sees past frames and states with various noise levels at training. At inference time, we can leverage this property to increase the stability of rollouts: After generating a (frame, state) pair, instead of directly inserting it into the KV cache, we first re-noise it to noise levels $1 - \tau_{frame}$ and $1 - \tau_{state}$ respectively, where 0 is clean and 1 is pure noise (Appendix G). Intentionally noising the past forces the model to trust it less, reducing the compounding error. In practice, we use a noise level of 0 for the frame stream and 0.3 for the states.

3.3 Data Generation: RoLeSynth

Game states yield a simple structure which is much easier to manipulate than the video frames. Recent foundation models handle structured data well and are showing increasing progress in spatial and 3D understanding, which raises the question of how to leverage them to manipulate and augment such data. This approach has two benefits: **(i)** It allows us to generate novel data cheaply, without access to a game or physics engine, many of which are private or require engineering effort to reproduce, **(ii)** it indirectly injects general knowledge from the foundation model into novel data synthesis. Unlike a real physics simulation, however, we have no guarantee that the generated trajectories match the original world. We further address these points in Appendix E.1; Appendix E.2 also compares the final RoLeSynth with the open-source RocketSim engine.

Building on this idea, we introduce RoLeSynth, a lightweight rule-based physics engine built with a LLM agent's (Claude Opus) assistance using the Rocket Science dataset as a starting point; It can be seen as a simplified game engine, reverse engineered from ground-truth trajectories. The engine was refined over several iterations between which we verify its outputs against the Rocket Science data. Internally, RoLeSynth defines physical constants fitted to the recorded game states. Other constants that short recorded clips cannot measure, such as the arena dimensions and speed caps, were set by the LLM itself from documented game knowledge. RoLeSynth outputs a match as a trajectory of game states. A 5-minute match takes only about 1.4 s to simulate on a single CPU core (220× real time). Ground-truth actions are generated alongside the states for world model training, by reading off the motion that is produced at every frame. RoLeSynth does not receive action as inputs and is therefore not interactive by itself; Even when given an action interface, it is 14% less accurate than a learned SCWM (Appendix E.4). In addition to the engine, we also define *player profiles* which can be seen as a lightweight form of policies. We first define a SynthRoSc player calibrated to mimic the original policy induced by the Rocket Science dataset. We also generate a diverse random zoo of player profiles, SynthZoo (Appendix E.3), with two goals in mind: **(i)** To add more trajectory diversity, as a form of data augmentation and reduce the aforementioned policy bias (Section 4.2), and **(ii)** to introduce novel actions in the state space, relying on the LLM agent's knowledge to generate out-of-distribution behaviors (Appendix F.1).

4 EXPERIMENTS

4.1 MAIN EXPERIMENTS

We train on the public Rocket Science (RoSc) dataset, consisting of ∼15.8k 2v2 Rocket League matches totaling roughly 2k hours, hence 8k hours of single-player data. We train our models at 512×288 resolution, with state and actions sampled at 10 fps; The video itself runs at 20 fps but is compressed to 10 fps latents by the encoder before entering the model backbone.

Main results. We train a VWM, SCWM and DSWM with comparable model size of 1.2B parameters. The models outputting video frames (VWM, DSWM and neural renderer) were trained on the same amount of videos; The transition model w_T in SCWM is only 200M parameters, yielding very fast training times and low memory footprint, and was trained on 9 times more (state only) data. Despite the larger amount of data, it finishes training 3 times faster than the neural renderer. At inference, each model auto-regressively unrolls generated trajectories of 12 seconds from an initial context of 4 seconds, sampled from the held-out test split of RoSc. The rollouts are then evaluated on the image quality (gFID), trajectory fidelity (DINO drift, gPSNR) and game state fidelity (RMSE on car positions, Ball position error, measured in unreal units, uu) of those unrolled sequences compared to the ground-truth video and states. Table 1 compares the three model families on video- and state-based metrics. Exploiting state information improves performance on almost all metrics: both SCWM and DSWM outperform the video world model baseline, the only exception being the gFID of DSWM. As we show in the remainder of the text, it also enables additional efficiency, uncertainty estimation (Section 4.3) and novel action synthesis (Appendix F.1). Qualitative samples for all experiments are linked in Appendix B supported by this anonymous github-hosted webpage.

Model	Params	gFID ↓	gFVD ↓	DINO drift ↓	gPSNR ↑	Pos. RMSE ↓	Ball pos. err. ↓
VWM	1.2B	10.60	104.7	0.222	15.04	–	–
SCWM	1.2B	**9.40**	**85.3**	0.214	15.39	**1953**	**2743**
DSWM	1.2B	10.89	96.2	**0.213**	**15.48**	2083	3118

Table 1: **Main comparison of world model types.** SCWM is built with a 200M transition model and a 1B neural renderer. The parameter count omits the 900M-param DINOv3 encoder used by VWM, DSWM and the neural renderer of SCWM. Exploiting state information improves performance across the board, the only exception being the gFID of DSWM. As we show in the remainder of the section, it also enables additional properties and better efficiency.

Ablation 1: Parameter efficiency of the state-centric world model. In Figure 3, we vary the latent width of SCWM and VWM, and evaluate their gFID drift at different horizons. Notably, the SCWM retains almost the same performance with 31M parameters as with 1B, suggesting it is easier to predict world dynamics in the explicit state latents. Using self-forcing Huang et al. (2025) (see Table 5 in the appendix) further improves the action-following performance of the SCWM, but does not improve image quality. In contrast, VWM heavily suffers from drifting at small model sizes, and only stabilizes at around 600M parameters. Even at its best, the VWM only catches up with the 31M SCWM after 25 s of rollout, with 20× more parameters. In terms of compute, the SCWM produces a frame about 4× faster than the VWM at batch size 1, and its latency does not depend on its size as the cost is dominated by the decoder for both SCWM and VWM. For larger batch sizes, the SCWM is 6.6× cheaper per player than the smallest VWM, and 9–10× cheaper than the VWMs of matching quality. This cheap transition model also allows higher frame rates: With `torch.compile`, one step takes 8.2ms, within a 60fps budget, and training at 60fps does not substantially degrade quality (Appendix F.2). Memory gains are smaller: The SCWM has an overall smaller parameter count, but the neural renderer keeps recent frames at full resolution, incurring a memory overhead. Detailed metrics for every model size at a fixed 12 s horizon are given in Table 4.

Ablation 2: State encoding in SCWM. In Table 2 (left), we ablate the positional embedding of the renderer (Section 3.1): applying RoPE only to the ego car and the ball tokens brings the most improvement. This echoes with the findings of a previous line of work (Xiang et al., 2026; Wu et al., 2026; Qiao et al., 2026) which uses RoPE to encode camera variables: the ego car drives the camera, which always points towards the ball.

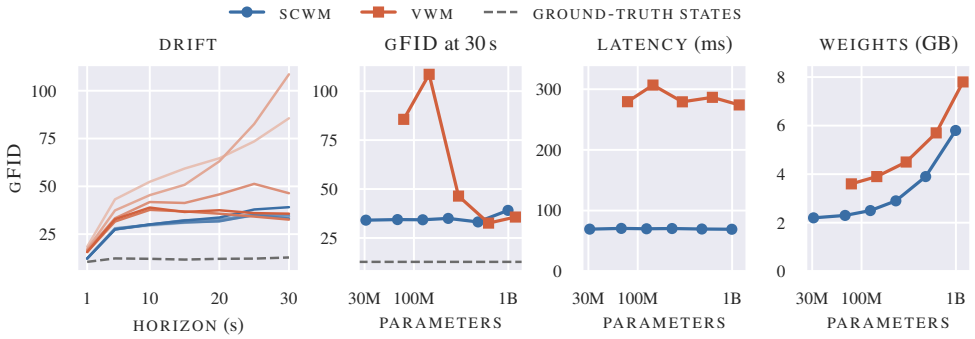


Figure 3: **SCWM and VWM performance against model size.** On the left, we plot the gFID over a 30 s rollout for every world model size (lighter = smaller). On the right, we display the gFID at the end of the rollout, the latency and the memory taken by the total model weights, all as a function of the number of parameters, to show the scaling behavior with model size. SCWM has a smaller latency and its performance is robust to model size, while VWM at small sizes suffers from high drift: VWM only recovers performance as good as the smallest SCWM at around 600M parameters.

In Table 2 (right) we also report ablations on teacher vs diffusion forcing, and whether to predict absolute or relative states in the transition model. The results confirm that teacher forcing with relative encoding of states is the best setting for SCWM. Intuitively, diffusion forcing also corrupts the past tokens the prediction is conditioned on: Video frames tolerate it better thanks to their redundancy, but state tokens each carry exclusive information.

Neural renderer	gFID ↓	gFDD ↓	Drift ↓	gPSNR ↑
No RoPE	6.39	0.241	0.062	21.10
Ego car	6.41	0.236	0.062	21.51
Ego car + ball	**6.31**	**0.232**	**0.045**	23.07
All cars + ball	**6.31**	0.234	**0.045**	**23.08**

Transition model	gFID ↓	gFDD ↓	gFVD ↓	Drift ↓	Pos. ↓	Ball ↓
TF, delta	**9.40**	**0.471**	**85.3**	**0.214**	**1953**	**2743**
TF, absolute	10.47	0.550	104.7	0.219	2102	2978
DF, absolute	11.84	0.588	137.4	0.229	2148	3295

Table 2: **State encoding ablations.** On the left, we compare SCWM performance when applying RoPE to different game state tokens in the neural renderer (Section 3.1). The car which drives the camera and the ball account for the main gains.

Ablation 3: State encoding in DSWM Similarly, we evaluate alternative ways to encode the game states in DSWM. As described in Section 3.2, our main model uses relative states encoding, and RoPE for the positions of the ego car and the ball similar to the neural renderer. The results in Table 3 show that departing from either of those design choices degrades the performance of the model.

State	Anchor	Pos. RoPE	gFID ↓	gFVD ↓	DINO drift ↓	gPSNR ↑	Pos. RMSE ↓	Ball pos. err. ↓
Absolute	yes	no	10.98	94.2	0.219	15.21	2139	3367
Absolute	yes	yes	10.90	**93.1**	0.218	15.25	2097	3274
Relative	yes	no	11.06	100.6	0.216	15.36	2120	3231
Relative	yes	yes	**10.89**	96.2	**0.213**	**15.48**	**2083**	**3118**

Table 3: **State encoding designs in DSWM.** All models build on the baseline DSWM from Table 1. Dropping either of (a) the relative state encoding or (b) RoPE in the states stream hurts performance.

4.2 Measuring and Correcting Policy Bias

In the Rocket Science (RoSc) dataset, all cars follow the same policy (Nexto bot). As a result, the public MIRA contains notable biases leaking from the policy: For instance, at the start of the game cars often rush towards the ball no matter what action the user inputs.

Such biases are much easier to quantify in state space than in pixel space: we measure statistics on state trajectories, e.g. how many cars rush the ball at kickoff, how often the ball is hit or how much

the steering varies (full list in Table 11). As shown in Figure 4 (left), collecting more data from RoSc does not change these statistics: The bias comes from the data content. The clearest example is the kickoff: in RoSc, all four cars rush the ball in 85% of kickoffs (Figure 4, middle). Part of it does not even depend on the input, as we verify through a small counterfactual experiment: When one car's actions are removed after kickoff in a SCWM trained on SynthZoo, it still goes for the ball in 25% of kickoffs, against 43% with its own ground-truth actions (Figure 4, middle). However, since the transition model in SCWM is trained on states only, we can augment our initial corpus with synthetic trajectories generated with RoLeSynth (Section 3.3). To analyze the effect of augmented data, we train five SCWMs with the same recipe and change only the training data: RoSc trajectories, RoLeSynth data using the SynthRoSc policy built to emulate RoSc, RoLeSynth matches generated from the policy zoo (SynthZoo), and the respective mixtures of RoSc with these two synthetic data. Each model is then tested on the three data sources (RoSc, SynthRoSc and SynthZoo) and results are shown in Figure 4, right. All three single-source models learn the car physics of their own data equally well, but the main insights are about what biases transfer across policies.

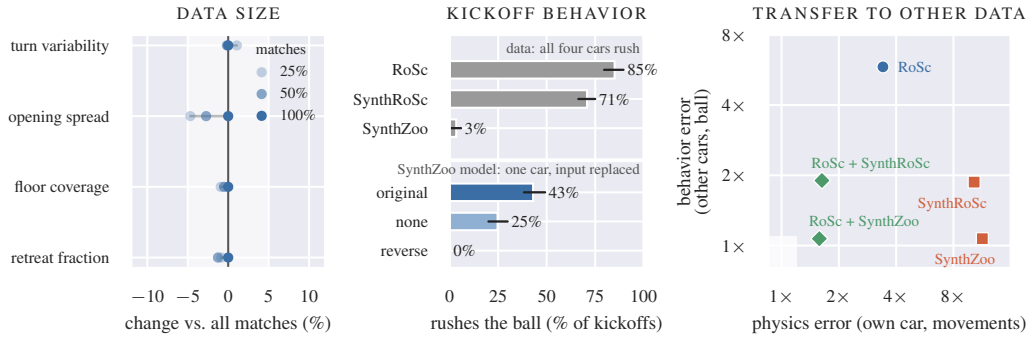


Figure 4: **Policy bias lives in the RoSc dataset.** *Left:* Behavior statistics when training on 25%, 50% and 100% of the RoSc matches barely differ. *Middle, top:* share of kickoffs in which all four cars rush the ball, in the training data: 85% in RoSc, 71% in SynthRoSc, and 3% in SynthZoo. In RoSc, the game freezes all cars for a short countdown at the start of each kickoff, so a rush is measured from the moment the cars are released rather than from the reset; the simulated data has no countdown, so both conventions agree there. Note that the top rows count kickoffs in which *all four* cars rush, whereas the bottom rows follow a *single* car whose input is replaced, so the two groups are not directly comparable. *Middle, bottom:* a SCWM trained on SynthZoo predicts the 3 s after a kickoff while the input to one car is replaced (the other cars keep their original input). With no input, the car still rushes the ball in a quarter of kickoffs; a reverse input stops it completely. 95% intervals from bootstrap resampling. *Right:* Five SCWMs that differ only in their training data: Rocket Science (RoSc), simulated matches from a policy built to imitate RoSc (SynthRoSc) or from SynthZoo, or a mix. Each model is tested on all three sources and compared with the model trained on that single source; Each marker is the model's worst case over the three test sources. RoSc transfers physics, simulated data transfers behavior, and mixing them gets close to both.

To measure this, we separate two kinds of prediction accuracy: *Physics* is the motion of the car the model is conditioned on: Its actions are given, so predicting where it goes only requires knowing how the game responds to them. We measure it with the next-frame position error on the ego car; *Behavior* is the motion of the other three cars and the ball, whose actions are not given: Predicting them means guessing what the other players will do, which is precisely where a policy bias may leak. For each model, we measure its prediction error normalized by that of the model trained on the test data (therefore, $1\times$ means equally good as the model trained on single-source data) and plot its worst case physics and behavior accuracy over the three test sources. The model trained on RoSc learns physics that transfers better than when trained on synthetic data. However, behavior transfers the other way (vertical axis): For instance, the model trained on SynthZoo predicts the other cars and the ball as well as the model trained on either dataset, while the model trained on RoSc is up to $5.8\times$ worse on domains where their policy bias no longer applies.

In other words, the transition model w_T learns the behavior contained in its data rather than the world (policy bias), thus introducing new data policy changes the world behavior w_T predicts. Finally,

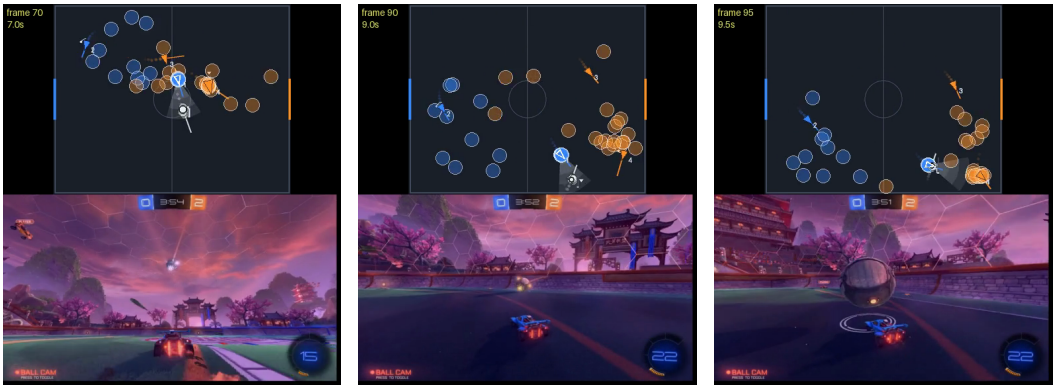
mixing RoSc with SynthZoo recovers most of both worlds: It is the only model close to the bottom-left corner, meaning both behavior (1.1× worse) and physics (1.6×) transfer well.

No handcrafted physics simulation was needed to correct the original RoSc bias: SynthZoo was created from the LLM agent's prior knowledge. As proprietary LLMs are increasingly better at spatial and physics understanding, this opens interesting perspectives on on-demand data generation for world modeling. Appendix F.1 uses the same property to teach the SCWM and DSWM new MAGNET and HOVER actions.

4.3 Estimating uncertainty with DSWM

In DSWM, the flow matching noise τ is drawn independently for each timestep and each of the modality stream by design, meaning that for certain frames, denoising the state might be an easier task than denoising the frame or vice versa. This makes DSWM robust to a missing modality, so it can also train on unpaired data (Appendix F.3), and enables three inference modes: **(i)** Jointly denoising both the states and video, as in Figure 2; **(ii)** Using the DSWM as a neural renderer by teacher-forcing the state and predicting the video; **(iii)** Using it as a streaming *inverse* renderer by teacher-forcing the video and predicting the game state.

The latter models uncertainty over hidden information: By sampling several state predictions from one video, we obtain confidence estimates for every state dimension, including the positions of cars the ego car does not see. Figure 5 illustrate this qualitatively: Position prediction for cars in the ego car's field of view exhibit little variation, clustering tightly around the truth; For off-screen cars, the samples spread out, and they tighten again within a few frames once the car reappears. Measured on 8 held-out clips with 10 samples per frame, the spread is about 35× wider for cars the camera does not show, and it matches the error the model actually makes: The ratio of error to spread is 0.96–1.10 for the ego car, the visible cars and the hidden cars alike. Appendix H further describes the details of the uncertainty setup.



(a) (b) (c)

Figure 5: **DSWM is an inverse renderer, mapping video to states.** Here we sample 10 predictions from the state stream to obtain uncertainty estimates. (a) The orange car is visible in the top left of the screen and the sampled estimates of its position cluster tightly around the ground-truth; (b) After two seconds of the car being out of view, the model correctly identifies the general area the car is located in, but higher uncertainty; (c) After the car re-enters the view, the prediction tightens again.

5 Conclusions

We study how explicit states can be used alongside video to build interactive world models, on the video game Rocket League. Decoupling state transitions from rendering makes the dynamics cheap to learn: A state-centric world model with a 31M transition model matches video world models twenty times larger, and serves many players at a fraction of their cost. Modeling states and video jointly in a dual-stream model instead gives us a natural measure of uncertainty over the parts of the game hidden from the camera, and lets the same model act as an inverse renderer. Since the state space is simple and structured, we can leverage an LLM coding agent to generate novel data which we use to reduce the policy bias with SynthZoo and to teach the model entirely new actions.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing, drafting parts of the supplementary material, execution of research ideas and generating synthetic datasets. We have not used generative AI tools for retrieval and discovery of related work or proving mathematical claims are not applicable to this work. We have reviewed all AI-assisted work: For example, we have proofread all corrections, feedback and other typos made by AI. The draft sections for the supplementary materials were only used as supporting facts and rewritten by one of the authors. LLM generated code was reviewed by authors (e.g. standard PR review process between codebase contributors) and tested in different settings. Finally, the synthetic dataset generation assisted by a LLM coding agent (Claude Opus) is further detailed in Appendix E, and was tested against the ground-truth data at several point in times to verify its consistency. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We do not believe our submission raises significant ethics concerns. Nonetheless, our work on controllability (RoLeSynth) indirectly attempts to reverse engineer an existing game engine by prompting a LLM to reproduce existing game trajectories. While far from perfectly reproducing the game engine (see Appendix E for a comparison to existing open-source engines), this experiment suggests foundation LLMs are getting increasingly better at reproducing existing, possibly proprietary, code with simple prompting.

REPRODUCIBILITY STATEMENT

For the synthetic data generation, Appendix E contains some details on the physics constants used in the engine. The Rocket Science dataset we use as our basis is already publicly available. For models, architecture and training details are explained either in MIRA, our baseline, or in our Section 3 as well as Appendix C. We also plan to release the following artifacts: The code of RoLeSynth engines, alongside the generated state-only trajectories. The model architecture and inference code, alongside the model weights for our main results tables.

REFERENCES

AlayaWorld Team, Kaipeng Zhang, Chuanhao Li, Yifan Zhan, Yongtao Ge, Yuanyang Yin, Jiaming Tan, Kang He, Liaoyuan Fan, Ruicong Liu, Xiaojie Xu, Xuangeng Chu, Zhen Li, Zhengyuan Lin, Zhixiang Wang, Zian Meng, and Zihui Gao. Alayaworld: Long-horizon and playable video world generation. *arXiv preprint 2607.06291*, 2026. URL https://arxiv.org/abs/2607.06291. 1

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J. Storkey, Tim Pearce, and François Fleuret. Diffusion for World Modeling: Visual Details Matter in Atari. In *NEURIPS*, 2024. URL https://arxiv.org/abs/2405.12399. 2

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *CVPR*, 2025. URL https://arxiv.org/abs/2412.03572. 1

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. https://openai.com/index/video-generation-models-as-world-simulators/, 2024. URL https://openai.com/index/video-generation-models-as-world-simulators/. 2

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative Interactive Environments. In *ICML*, 2024. URL https://arxiv.org/abs/2402.15391. 1, 2

Hila Chefer, Uriel Singer, Amit Zohar, Yuval Kirstain, Adam Polyak, Yaniv Taigman, Lior Wolf, and Shelly Sheynin. VideoJAM: Joint appearance-motion representations for enhanced motion generation in video models. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v267/chefer25a.html. 3

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *NEURIPS*, volume 37, pp. 24081–24125, 2024. URL https://arxiv.org/abs/2407.01392. 4

Jingye Chen, Yuzhong Zhao, Yupan Huang, Lei Cui, Li Dong, Tengchao Lv, Qifeng Chen, and Furu Wei. Model as a Game: On Numerical and Spatial Consistency for Generative Games. In *ICCV Workshops*, 2025a. URL https://arxiv.org/abs/2503.21172. 2

Taiye Chen, Xun Hu, Zihan Ding, and Chi Jin. VRAG: Learning World Models for Interactive Video Generation. In *NEURIPS*, 2025b. URL https://arxiv.org/abs/2505.21996. 2

Guangzhao Dai, Qi Wu, and Bin Zhu. How far can gpt-6-astra go? evaluating capabilities in zero-shot vision-and-language navigation. *arXiv preprint 2609.20116*, 2026. URL https://arxiv.org/abs/2609.20116. 2

Decart and Etched. Oasis: A Universe in a Transformer. https://oasis-model.github.io/, 2024. URL https://oasis-model.github.io/. 2

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint 2410.00037*, 2024. URL https://arxiv.org/abs/2410.00037. 3

Zhiyang Deng, Boran Zhang, Danze Chen, and Yeying Jin. WorldMind: Decoupled Game World Model for State-Aware NPC Behavior. *arXiv preprint 2608.21439*, 2026. URL https://arxiv.org/abs/2608.21439. 2

Quentin Garrido, Tushar Nagarajan, Basile Terver, Nicolas Ballas, Yann LeCun, and Michael Rabbat. Learning latent action world models in the wild. *arXiv preprint 2601.05230*, 2026. URL https://arxiv.org/abs/2601.05230. 1

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *NEURIPS*, 2018. URL https://arxiv.org/abs/1803.10122. 2

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, Eitan Richardson, Guy Shiran, Itay Chachy, Jonathan Chetboun, Michael Finkelson, Michael Kupchick, Nir Zabari, Nitzan Guetta, Noa Kotler, Ofir Bibi, Ori Gordon, Poriya Panet, Roi Benita, Shahar Armon, Victor Kulikov, Yaron Inger, Yonatan Shiftan, Zeev Melumian, and Zeev Farbman. LTX-2: Efficient Joint Audio-Visual Foundation Model. *arXiv preprint 2601.03233*, 2026. URL https://arxiv.org/abs/2601.03233. 3

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint 2301.04104*, 2024. URL https://arxiv.org/abs/2301.04104. 1, 2

Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, Oier Mees, Marc Pollefeys, Zhuang Liu, Jiajun Wu, Pieter Abbeel, Jitendra Malik, Yilun Du, and Jianfei Yang. World model for robot learning: A comprehensive survey. *arXiv preprint 2605.00080*, 2026. URL https://arxiv.org/abs/2605.00080. 1

Anthony Hu, Václav Volhejn, Adrien Ramanana Rahary, Chris Mulder, Aditya Makkar, Alyx Liao, Amélie Royer, Manu Orsini, Adam Jelley, Eloi Alonso, et al. Multiplayer interactive world models with representation autoencoders. *arXiv preprint 2607.05352*, 2026. URL https://arxiv.org/abs/2607.05352. 1, 2, 3, 15, 18

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *NEURIPS*, 2025. URL https://arxiv.org/abs/2506.08009. 6

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. In *ICLR*, 2026. URL https://arxiv.org/abs/2601.16163. 3

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. *arXiv preprint 2506.17201*, 2025. URL https://arxiv.org/abs/2506.17201. 2

Zijun Lin, Zeqing Wang, Cheston Tan, Bihan Wen, and Yeying Jin. Stateplay: State-aware game world models for mechanics-consistent generation. *arXiv preprint 2607.26754*, 2026. URL https://arxiv.org/abs/2607.26754. 1

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023. 4

Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. Physgen: Rigid-body physics-grounded image-to-video generation. *arXiv preprint 2409.18964*, 2024. URL https://arxiv.org/abs/2409.18964. 1

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023. 16

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint 2603.19312*, 2026. URL https://arxiv.org/abs/2603.19312. 1

Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi Fan. NitroGen: An open foundation model for generalist gaming agents. In *CVPR*, pp. 21511–21521, 2026. 21

Willi Menapace, Aliaksandr Siarohin, Stéphane Lathuilière, Panos Achlioptas, Vladislav Golyanik, Sergey Tulyakov, and Elisa Ricci. Promptable Game Models: Text-Guided Game Simulation via Masked Diffusion Models. *TOG*, 2023. URL https://arxiv.org/abs/2303.13472. 2

Willi Menapace, Aliaksandr Siarohin, Stéphane Lathuilière, Panos Achlioptas, Vladislav Golyanik, Sergey Tulyakov, and Elisa Ricci. Promptable game models: Text-guided game simulation via masked diffusion models. *ACM Transactions on Graphics*, 43(2):1–16, Jan 2024. ISSN 1557-7368. doi: 10.1145/3635705. URL http://dx.doi.org/10.1145/3635705. 1

Zian Meng, Zhen Li, Chuanhao Li, Qiang Li, and Kaipeng Zhang. Marionette: Predicting World States, Rendering Geometry, Painting Appearance. *arXiv preprint 2608.14530*, 2026a. URL https://arxiv.org/abs/2608.14530. 2

Zian Meng, Zhen Li, Chuanhao Li, Qiang Li, and Kaipeng Zhang. Marionette: Predicting world states, rendering geometry, painting appearance. *arXiv preprint 2608.14530*, 2026b. URL https://arxiv.org/abs/2608.14530. 1

Odyssey. Agora-1: The Multi-Agent World Model. https://odyssey.systems/introducing-agora-1, 2026. URL https://odyssey.systems/introducing-agora-1. 2

PAN Team, Zihan Liu, Yi Gu, Mingkai Deng, Guangyi Liu, Zeyu Feng, Qiyue Gao, Yiyan Hu, Benhao Huang, Yichi Yang, Kun Zhou, Jiannan Xiang, Zhiting Hu, Zhengzhong Liu, and Eric P. Xing. Pan: A world model for general, actionable, and long-horizon world simulation. *arXiv preprint 2511.09057*, 2026. URL https://arxiv.org/abs/2511.09057. 1

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 4

Ryan Po, David Junhao Zhang, Amir Hertz, Gordon Wetzstein, Neal Wadhwa, and Nataniel Ruiz. MultiGen: Level-Design for Editable Multiplayer Worlds in Diffusion Game Engines. *arXiv preprint 2603.06679*, 2026. URL https://arxiv.org/abs/2603.06679. 2

Francois Porcher, Nicolas Carion, Karteek Alahari, and Shizhe Chen. Flow matching in feature space for stochastic world modeling. *arXiv preprint 2606.29059*, 2026. URL https://arxiv.org/abs/2606.29059. 1

Runjia Qian, Zile Wang, Jihai Zhang, Kai Zou, Wei Yu, Jiaxing Li, Zexiang Liu, Yaokun Li, Fei Kang, Kaichen Huang, Mengyin An, Haobo Zhang, Biao Jiang, Jiahua Wang, Haofeng Sun, Yang Liu, and Yangguang Li. Matrix-Game 3.5: Enhancing Real-Time Streaming Interactive World Models with Patch Memory. *arXiv preprint 2608.29910*, 2026. URL https://arxiv.org/abs/2608.29910. 2

Zhijian Qiao, Xinjiang Wang, Jiajie Chen, Haoming Huang, Meng Li, Chih-Chung Chou, Jing Wang, and Shaojie Shen. Merope: Metric rotary position embedding for camera-controlled video generation. *arXiv preprint 2609.01252*, 2026. URL https://arxiv.org/abs/2609.01252. 6

Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, Yihang Chen, Jie Liu, Yansong Cheng, Yao Yao, Jiayi Zhu, Yihao Meng, Kecheng Zheng, Qingyan Bai, Jingye Chen, Zehong Shen, Yue Yu, Xing Zhu, Yujun Shen, and Hao Ouyang. Advancing Open-source World Models. *arXiv preprint 2601.20540*, 2026. URL https://arxiv.org/abs/2601.20540. 2

Rolv-Arild. Necto. https://github.com/Rolv-Arild/Necto, 2021. URL https://github.com/Rolv-Arild/Necto. 3

Georgy Savva, Oscar Michel, Daohan Lu, Suppakit Waiwitlikhit, Timothy Meehan, Dhairya Mishra, Srivats Poddar, Jack Lu, and Saining Xie. Solaris: Building a Multiplayer Video World Model in Minecraft. *arXiv preprint 2602.22208*, 2026. URL https://arxiv.org/abs/2602.22208. 2

Shuyao Shang, Yuqi Wang, Ruopeng Gao, Xu Chen, Tieniu Tan, Lue Fan, and Zhaoxiang Zhang. Phizero: A world model built around physical language. *arXiv preprint 2607.28624*, 2026. URL https://arxiv.org/abs/2607.28624. 1

Joonghyuk Shin, Yicong Hong, Jaesik Park, and Xun Huang. Can Video World Models Track Unobserved World States? *arXiv preprint 2608.30692*, 2026a. URL https://arxiv.org/abs/2608.30692. 2

Joonghyuk Shin, Zhengqi Li, Richard Zhang, Jun-Yan Zhu, Jaesik Park, Eli Shechtman, and Xun Huang. Motionstream: Real-time video generation with interactive motion controls. *arXiv preprint 2511.01266*, 2026b. URL https://arxiv.org/abs/2511.01266. 1

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *TMLR*, 2026. URL https://openreview.net/forum?id=2NlGyqNjns. 4

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 17

Rishi Upadhyay, Howard Zhang, Jim Solomon, Ayush Agrawal, Yunhao Ba, Alex Wong, Celso M de Melo, and Achuta Kadambi. Worldbench: Benchmarking physical understanding of world models by isolating physics concepts. *arXiv preprint 2601.21282*, 2026. URL https://arxiv.org/abs/2601.21282. 2

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, Mike Zheng Shou, Xuanjing Huang, Xipeng Qiu, and Yu-Gang Jiang. World action models: The next frontier in embodied ai. *arXiv preprint 2605.12090*, 2026a. URL https://arxiv.org/abs/2605.12090. 3

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. *arXiv preprint 2604.08995*, 2026b. URL https://arxiv.org/abs/2604.08995. 2

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. iVideoGPT: Interactive VideoGPTs are Scalable World Models. In *NEURIPS*, 2024. URL https://arxiv.org/abs/2405.15223. 3

Yu Wu, Minsik Jeon, Jen-Hao Rick Chang, Oncel Tuzel, and Shubham Tulsiani. Rayrope: Projective ray positional encoding for multi-view attention. *arXiv preprint 2601.15275*, 2026. URL https://arxiv.org/abs/2601.15275. 6

Chendong Xiang, Jiajun Liu, Jintao Zhang, Xiao Yang, Zhengwei Fang, Shizun Wang, Zijun Wang, Yingtian Zou, Hang Su, and Jun Zhu. Geometry-aware rotary position embedding for consistent video world model. *arXiv preprint 2602.07854*, 2026. URL https://arxiv.org/abs/2602.07854. 6

Sherry Yang, Yilun Du, Seyed Kamyar Seyed Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *ICLR*, 2024. URL https://openreview.net/forum?id=sFyTZEqmUY. 1

Xiaoyu Zhan, Xinyu Wang, Xiaohong Zhang, Huanjie Zhu, Tengjiao Sun, Pengcheng Fang, Jiaxing Yu, Yanwen Guo, and Dongjie Fu. Magpie: Real-Time World Renderer for Interactive Games. *arXiv preprint 2608.27168*, 2026. URL https://arxiv.org/abs/2608.27168. 2

Jiahan Zhang, Muqing Jiang, Nanru Dai, TaiMing Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal M. Patel, Paul Pu Liang, Daniel Khashabi, Cheng Peng, Rama Chellappa, Tianmin Shu, Alan Yuille, Yilun Du, and Jieneng Chen. World-in-world: World models in a closed-loop world. In *ICLR*, 2026a. URL https://openreview.net/forum?id=yDmb7xAfeb. 1

Wenbo Zhang, Kaixuan Wang, Yutao Ouyang, Xiaoyu Huang, Liyang Li, Kailun Su, Weiyang Jin, Wenhao Chai, Haotian Liang, Zhiyang Dou, Yue Chen, and Tianxing Chen. An unexpected robot policy: Early evaluations of gpt-6 astra on robodojo and beyond. *arXiv preprint 2609.24170*, 2026b. URL https://arxiv.org/abs/2609.24170. 2

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, volume 2026, pp. 35791–35820, 2026. 1

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified World Models: Coupling Video and Action Diffusion for Pretraining on Large Robotic Datasets. In *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2504.02792. 3