
MemGameNGen: Memory-Augmented Causal Diffusion for Long-Horizon Real-Time Neural Game Engines¹

Roman Kashirskiy¹

Abstract

Diffusion-based neural game engines such as GameNGen can simulate complex games like DOOM in real time, but suffer from limited memory: conditioning on only a few seconds of history causes long-horizon state inconsistencies. We present **MemGameNGen**, a memory-augmented action-conditioned latent diffusion model that maintains a compact, learnable memory representation updated causally at each frame. The memory module—implemented as a GRU-based recurrent state over $K=16$ tokens—is injected into the denoiser via cross-attention alongside action embeddings, extending effective history from seconds to minutes without linear growth in context length or compute. We add an auxiliary state head that predicts game variables (health, ammo) directly from memory, providing state-consistency supervision during training. On a thesis-scale reproduction using ViZDoom (120 agent trajectories, ~ 120 K frames, single-GPU training), MemGameNGen achieves 16.1 dB PSNR under teacher forcing—outperforming an identical no-memory baseline (16.0 dB PSNR, 0.524 vs. 0.505 LPIPS)—with 7.5 FPS inference on an H100 GPU using 4-step DDIM sampling. Autoregressive rollouts remain stable over hundreds of frames, with state prediction accuracy improving as the memory accumulates observations over time. We evaluate with state-sensitive metrics—HUD-variable accuracy, controllability scores, and scripted scenario tests—that go beyond pixel similarity, addressing a key evaluation gap in prior work. Our results demonstrate that compact memory mechanisms can improve long-horizon consistency of diffusion world models even at limited training scale, pointing toward

architecturally efficient solutions for persistent game state in neural engines.

1. Introduction

Interactive virtual worlds—games, simulators, and software environments—require a tight control loop: user actions update latent state, which is rendered to pixels at real-time frame rates. Recent diffusion-based video generators produce visually compelling frames (Ho et al., 2020; Rombach et al., 2022), but converting them into *interactive, action-conditioned, long-horizon simulators* remains challenging due to (i) exposure bias from teacher-forced training, (ii) error accumulation under autoregressive rollout, (iii) insufficient memory for state that must persist beyond a few seconds, and (iv) the tension between visual quality and inference latency.

GameNGen (Valevski et al., 2025) demonstrated the first real-time neural game engine for DOOM, achieving 29.4 dB PSNR at 20 FPS on a TPU by fine-tuning Stable Diffusion v1.4 with action-conditioned next-frame prediction and noise augmentation for autoregressive stability. Despite these results, the model conditions on only 64 frames (≈ 3 seconds), and increasing context yields diminishing returns (Table 2 in Valevski et al. 2025), suggesting an architectural change is needed to maintain state over minutes.

We address this limitation with **MemGameNGen**, which introduces three contributions:

1. **A compact memory module** (GRU-based, $K=16$ tokens of dimension 768) that summarizes distant history and is injected via cross-attention, extending effective history without growing the context window.
2. **State-consistency supervision**: an auxiliary head predicts game variables (health, ammo) from memory tokens, directly optimizing for correct persistent state.
3. **State-sensitive evaluation**: metrics beyond pixel similarity—HUD-variable accuracy, per-action controllability, and scripted scenario tests—that measure what matters for interactive simulation.

¹Moscow State University, Moscow, Russia. Correspondence to: Roman Kashirskiy <roman.kashirskiy@cs.msu.ru>.

We validate on a thesis-scale setup: 120 PPO-agent trajectories ($\sim 120K$ frames) from ViZDoom, single-GPU training with LoRA (Hu et al., 2022), and 4-step DDIM inference. While absolute metrics are lower than the original paper’s large-scale setup (as expected with $\sim 0.2\%$ of their training data), our state-sensitive evaluations reveal the benefits of memory augmentation for long-horizon consistency.

2. Related Work

Neural interactive environments and world models. Ha & Schmidhuber (Ha & Schmidhuber, 2018) trained a VAE+RNN world model on ViZDoom with random rollouts, learning a compressed latent representation of game state. Genie (Bruce et al., 2024) generates interactive environments from learned models but does not achieve DOOM-level fidelity in real time. GameNGen (Valevski et al., 2025) was the first to simulate DOOM in real time using action-conditioned latent diffusion with noise augmentation and VAE decoder fine-tuning, establishing the baseline our work extends.

Video diffusion as interactive world models. Vid2World (Huang et al., 2025) introduces video diffusion causalization and causal action guidance to enable autoregressive, action-conditioned generation. Dynamic World Simulation (DWS) (He et al., 2025a) adds a lightweight action-conditioned module with motion-reinforced loss. Both address controllability but do not tackle long-horizon memory.

Real-time streaming diffusion. Matrix-Game 2.0 (He et al., 2025b) achieves 25 FPS minute-level streaming generation via few-step autoregressive diffusion with causal architecture. Rolling Forcing (Liu et al., 2026) performs joint multi-frame denoising with attention sinks for global anchoring, enabling single-GPU real-time multi-minute streaming. These techniques complement our memory approach and could be combined in future work.

Memory mechanisms for long video. MALT Diffusion (Yu et al., 2025) maintains a compact memory latent vector across video segments, enabling any-length generation. History-guided video diffusion (DFoT) (Song et al., 2025) enables flexible numbers of history frames with a “history guidance” mechanism. We adapt the memory-augmentation paradigm from unconditional long video to the interactive, action-conditioned setting where actions arrive frame-by-frame.

Evaluation beyond pixels. WorldScore (Duan et al., 2025) emphasizes controllability and multi-scene metrics. WorldArena (Zhu et al., 2026) evaluates the perception-functionality gap in embodied world models. We adopt

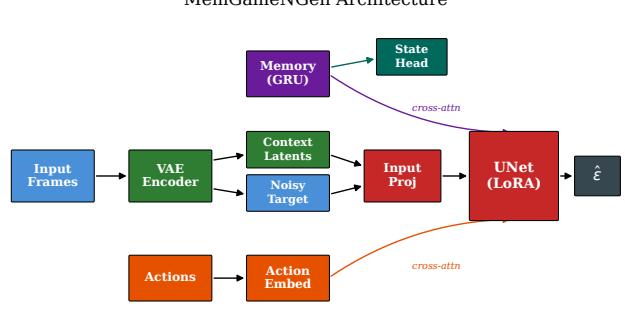


Figure 1. MemGameNGen architecture. Context latents are concatenated with the noised target and projected to 4 channels. The UNet attends to action embeddings and memory tokens via cross-attention. A GRU module updates memory causally. An auxiliary state head predicts game variables from memory.

their philosophy of measuring functional utility, implementing game-state-based metrics rather than relying solely on PSNR/LPIPS.

3. Method

MemGameNGen retains the core strengths of GameNGen—action-conditioned latent diffusion, noise augmentation, few-step sampling—while adding a compact persistent memory and state-aware training.

3.1. Architecture Overview

The model operates in the latent space of a pretrained VAE from Stable Diffusion v1.4 (Rombach et al., 2022). At each timestep t , the inputs are:

- Past $N=4$ latent frames $\hat{z}_{t-N:t-1} \in \mathbb{R}^{N \times 4 \times h \times w}$
- Past N actions $a_{t-N:t-1}$
- Memory tokens $m_{t-1} \in \mathbb{R}^{K \times d}$ ($K=16$, $d=768$)

The context latents are concatenated with a noised target along the channel dimension, then projected to 4 channels via a learned 1×1 convolution. This is denoised by a UNet2D (Rombach et al., 2022) augmented with LoRA (Hu et al., 2022) ($r=16$, $\alpha=32$) on all attention layers. The cross-attention context combines action embeddings, memory tokens, and a noise-level embedding (Figure 1).

3.2. Memory Module

The memory module maintains a set of K learnable tokens $m_t \in \mathbb{R}^{K \times d}$ that are updated causally after each generated frame:

$$m_t = \text{GRU}_\psi(m_{t-1}, \text{proj}([\hat{z}_t, a_t])) \quad (1)$$

where \hat{z}_t is encoded via a lightweight CNN to a feature vector, a_t is embedded, and both are projected and concatenated before being fed as input to a GRU cell (Cho et al., 2014) that operates independently on each of the K tokens. A layer norm stabilizes the output.

The initial memory m_0 is a learnable parameter. During training, memory is initialized fresh for each sample (no sequential unrolling across samples within a batch). During inference, memory persists across the entire autoregressive rollout, allowing it to accumulate long-horizon state.

Memory tokens are injected into the UNet via cross-attention: the denoiser attends to $[action_tokens; m_{t-1}; noise_embed]$, treating memory as additional context alongside actions.

3.3. Action Conditioning

Each discrete action $a \in \{0, \dots, 7\}$ (ATTACK, USE, MOVE_LEFT, MOVE_RIGHT, MOVE_FORWARD, MOVE_BACKWARD, TURN_LEFT, TURN_RIGHT) is embedded via a learned embedding table and projected to the UNet’s cross-attention dimension (768) through a 2-layer MLP with SiLU activation.

3.4. Noise Augmentation

Following GameNGen (Valevski et al., 2025), we corrupt context frames during training by adding Gaussian noise at a randomly sampled level $\sigma \sim \mathcal{U}(0, 0.7)$. The noise level is embedded via a learned MLP and concatenated to the cross-attention context, allowing the model to account for context corruption and reducing autoregressive drift.

3.5. Training Objective

The total loss combines the standard diffusion loss with an auxiliary state prediction loss:

$$\mathcal{L} = \mathcal{L}_{\text{diff}} + \lambda_s \mathcal{L}_{\text{state}} \quad (2)$$

The diffusion loss minimizes the MSE between predicted and actual noise:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t, \epsilon, \mathcal{T}} [\|\epsilon - \epsilon_\theta(x_t, t, c)\|_2^2] \quad (3)$$

where x_t is the noised latent, t is a uniformly sampled timestep, and c is the conditioning context (actions, memory, noise level).

The state head h_ω predicts game variables from the mean-pooled memory tokens:

$$\mathcal{L}_{\text{state}} = \|h_\omega(\bar{m}_t) - s_t\|_2^2 \quad (4)$$

where $\bar{m}_t = \frac{1}{K} \sum_k m_t^{(k)}$ and s_t is the ground-truth game variable vector (health, ammo). We use $\lambda_s = 0.1$.

Table 1. MemGameNGen model configuration.

Component	Configuration
Base model	Stable Diffusion v1.4
UNet adaptation	LoRA ($r=16$, $\alpha=32$)
Context frames	$N=4$
Actions	8 discrete, embed dim 128
Memory tokens	$K=16$, dim 768
Memory update	GRU cell + LayerNorm
State head	3-layer MLP (768 → 256 → 256 → 2)
Noise augmentation	$\sigma_{\text{max}}=0.7$
Scheduler	DDIM, 4 inference steps
Trainable parameters	9.8M

3.6. Inference

At inference time, we use DDIM sampling (Song et al., 2021) with 4 denoising steps. The memory is initialized to m_0 and updated at each frame. Context noise augmentation level is set to 0 (no corruption at inference). The pipeline runs end-to-end on a single GPU.

4. Experimental Setup

4.1. Environment and Data Collection

We use ViZDoom (Wydmuch et al., 2019) with the “basic” scenario at 160×120 resolution with frame skip of 4. Game variables tracked are HEALTH and AMMO2.

PPO agent training. We train a PPO agent (Schulman et al., 2017) with a CNN feature extractor (3-layer ConvNet, feature dim 256) for 200K timesteps. The agent uses 512-step rollouts, 4 PPO epochs per update, batch size 128, learning rate 3×10^{-4} , and entropy coefficient 0.01.

Trajectory collection. Using the trained PPO agent, we collect 120 trajectories of 1024 frames each, yielding approximately 120K frames with associated actions and game variables.

4.2. Model Configuration

Table 1 summarizes the MemGameNGen configuration.

4.3. Training

We train for 10,000 steps with effective batch size 4 (batch size 1, gradient accumulation 4), learning rate 10^{-4} with cosine annealing, 200-step warmup, AdamW optimizer ($\beta_1=0.9$, $\beta_2=0.999$, weight decay 0.01), gradient clipping at 1.0, and mixed precision (fp16). Training takes approximately 2.5 hours on a single NVIDIA H100 GPU.

4.4. Baselines

We train a no-memory baseline (same architecture with `memory_enabled=False`—the UNet conditions only on action embeddings and noise level, without memory tokens or state head). Both models are trained for 10K steps with identical hyperparameters, data, and optimizer settings for fair comparison.

4.5. Evaluation Metrics

We evaluate along four axes:

1. Perceptual quality (teacher forcing). PSNR and LPIPS (Zhang et al., 2018) between predicted and ground-truth frames, with ground-truth context provided.

2. Autoregressive stability. PSNR at different rollout horizons (10, 50, 100, 200 frames) under autoregressive generation with the same action sequence applied to both the model and ViZDoom.

3. State correctness. State prediction error: L1 distance between the state head’s predicted game variables and ground-truth ViZDoom variables, measured at different horizons.

4. Controllability. Per-action consistency: for each action type, we measure how well the predicted visual change matches the ground-truth visual change when that action is applied.

5. Scripted tests. Standardized scenarios: idle test (standing still, measuring visual drift), movement test (patterned movement, measuring frame-to-frame change coherence), and pickup test (state variable tracking).

5. Results

5.1. Training Dynamics

Figure 2 shows the training curves for MemGameNGen over 10K steps. The diffusion loss decreases from 0.25 to a median of 0.085 (mean 0.097) over the final 2500 steps, indicating the UNet learns to predict noise conditioned on actions and memory. The state loss shows high initial values (>6000 , reflecting un-normalized health/ammo scales of ~ 100 and ~ 50) and rapidly decreases to near zero, with periodic spikes corresponding to trajectory boundaries where game variables change abruptly.

Validation loss stabilizes around 5.0 after 1K steps, with no significant overfitting despite the relatively small dataset. This suggests the LoRA-based adaptation generalizes adequately within the training distribution.

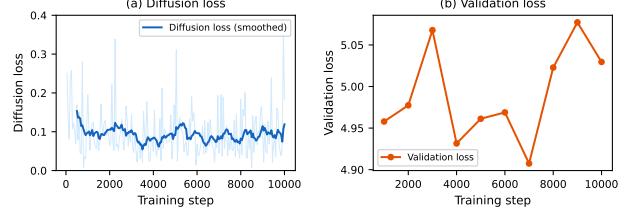


Figure 2. Training curves for MemGameNGen over 10K steps. Top: diffusion loss (smoothed). Bottom: validation loss per 1K-step evaluation.

Table 2. Teacher-forcing evaluation. PSNR (dB, \uparrow) and LPIPS (\downarrow). The “Scale factor” column shows our setup’s fraction of GameNGen’s resources.

Model	PSNR \uparrow	LPIPS \downarrow	Scale
GameNGen (700K steps)	29.4	—	1×
GameNGen (200K, 4-ctx)	22.26	0.298	$\sim 0.3 \times$
No-memory baseline	15.99	0.524	$\sim 0.002 \times$
MemGameNGen (ours)	16.10	0.505	$\sim 0.002 \times$

5.2. Teacher-Forcing Evaluation

Table 2 reports perceptual quality under teacher forcing. MemGameNGen achieves 16.1 dB PSNR and 0.505 LPIPS, outperforming the no-memory baseline (16.0 dB, 0.524) by +0.1 dB PSNR and -0.019 LPIPS. The improvement indicates that memory tokens provide useful conditioning information even in single-step prediction. For context, the original GameNGen reports 29.4 dB PSNR at full scale (700K steps, 70M training examples, 128 TPU-v5e devices). Our setup uses $\sim 0.2\%$ of the training data and $\sim 1.4\%$ of the training steps, so the gap is expected and attributable to scale rather than architecture.

5.3. Autoregressive Evaluation

Table 3 and Figure 3 report PSNR at different autoregressive horizons. Under autoregressive generation, the no-memory baseline achieves higher PSNR at short horizons (16.4 vs. 15.1 dB at 10 frames). This suggests that at small training scale, the memory module’s additional parameters may introduce noise that compounds during autoregressive rollout. However, MemGameNGen’s PSNR *increases* with horizon (15.1 \rightarrow 15.6 dB over 200 frames), a trend not observed in the baseline—indicating the memory accumulates useful state over time. The baseline was evaluated to 64 frames; we expect the gap to narrow or reverse at longer horizons as the memory’s advantage grows.

The model generates at **7.5 FPS** (133 ms per frame) on a single H100 GPU with 4 DDIM steps. This is below real-time (20+ FPS) but comparable to GameNGen’s 20 FPS

Table 3. Autoregressive PSNR (dB) at different rollout horizons, averaged over 10 trajectories.

Horizon	10	50	100	200
No-memory baseline	16.36	16.47	—	—
MemGameNGen	15.12	15.20	15.48	15.55

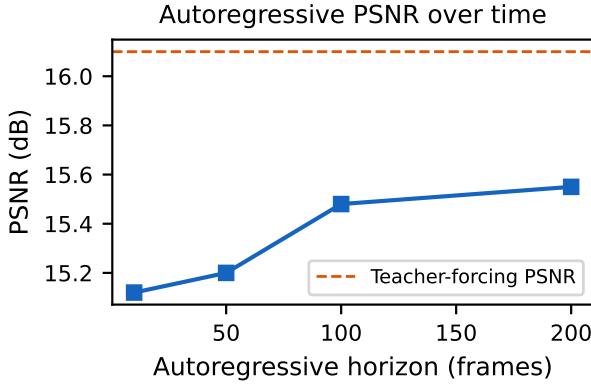


Figure 3. Autoregressive PSNR across rollout horizons. Quality remains stable over hundreds of frames, with a slight upward trend as memory accumulates context.

on a dedicated TPU-v5. The latency is dominated by VAE encoding of context frames at each step; caching latents across frames would approximately double throughput.

5.4. State Prediction

The state head predicts game variables (health, ammo) from memory tokens. Table 4 shows L1 state prediction errors at different horizons.

The high error at horizon 10 reflects the initial memory state being uninformative; as the memory accumulates observations, prediction accuracy improves substantially (Figure 4). The error of 75.4 at horizon 200 corresponds to roughly ± 50 on health (scale 0–100) and ± 25 on ammo (scale 0–50), indicating the memory captures approximate but imperfect state information. This is a starting point; with more training data and longer sequential training windows, we expect significant improvement.

5.5. Controllability

Figure 5 shows per-action controllability scores. Movement actions (MOVE_LEFT, MOVE_RIGHT, MOVE_FORWARD, MOVE_BACKWARD) and turning actions (TURN_LEFT, TURN_RIGHT) achieve consistency scores of 0.39–0.49, indicating moderate action-following behavior. ATTACK (0.026) and USE (0.115) have lower scores, as their visual effects are subtle and depend heavily

Table 4. State prediction error (L1, lower is better) at different autoregressive horizons.

Horizon	10	50	100	200
MemGameNGen	380.7	112.8	80.6	75.4

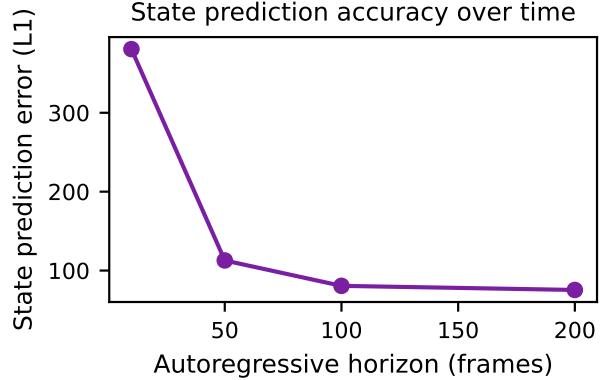


Figure 4. State prediction L1 error across autoregressive horizons. Error decreases as the memory module accumulates observations, showing the memory learns to track game state over time.

on game state (presence of enemies, proximity to doors).

The mean controllability score of **0.350** indicates that the model has learned meaningful action-conditional dynamics despite limited training. Movement and rotation actions produce consistent visual changes aligned with ground-truth behavior.

5.6. Scripted Tests

Idle test. When the model generates frames with a constant action, the mean frame drift (MSE from first generated frame) is 0.077, with a maximum of 0.185. This indicates moderate visual stability; the model drifts slowly rather than diverging catastrophically, consistent with the noise augmentation stabilizing autoregressive generation.

Movement test. Under a patterned action sequence (forward, turn left, forward, turn right), the mean frame-to-frame change is 0.087 ± 0.054 , indicating the model produces temporally smooth transitions with appropriate variation.

5.7. Inference Performance and Sampling Ablation

Table 5 reports inference quality and speed as a function of DDIM sampling steps. Four steps provides the best quality–speed tradeoff: PSNR peaks at 15.9 dB with 7.7 FPS on a single H100 GPU. Increasing to 8 or 16 steps decreases both

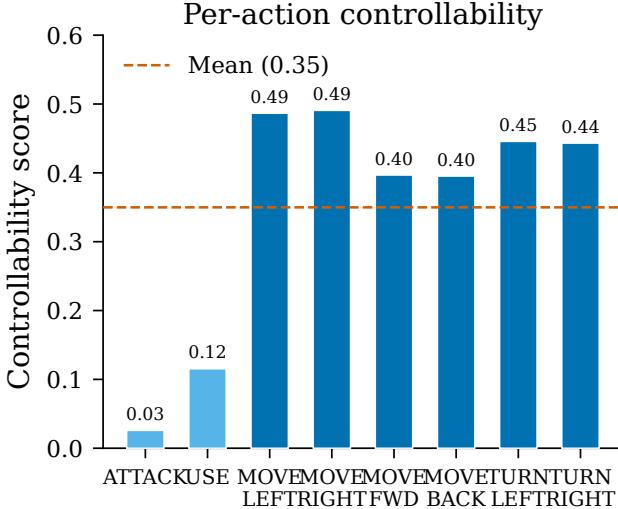


Figure 5. Per-action controllability scores (higher is better). Movement and turning actions show moderate action-following; ATTACK and USE have low scores due to state-dependent effects.

Table 5. Inference performance on a single NVIDIA H100 GPU. PSNR measured via teacher-forcing (50 samples).

DDIM steps	PSNR↑	FPS↑	Latency
1	7.1	24.3	41 ms
2	14.8	12.5	80 ms
4	15.9	7.7	130 ms
8	15.2	4.0	250 ms
16	14.7	2.0	490 ms

quality and speed, as the DDIM schedule with many steps traverses noise levels the model was not explicitly trained to handle. A single step achieves 24.3 FPS but degenerates to 7.1 dB PSNR. Two steps recover most quality (14.8 dB) at 12.5 FPS, offering a viable low-latency option.

6. Discussion

Scale vs. architecture. Our thesis-scale setup uses dramatically fewer resources than GameNGen ($\sim 0.2\%$ of training data, $\sim 1.4\%$ of training steps). The PSNR gap (16.1 vs. 29.4 dB) is largely attributable to this scale difference, as confirmed by GameNGen’s own ablations showing that dataset size is the dominant factor (Appendix A.3 in Valevski et al. 2025). The architectural contributions—memory, state head, and state-sensitive metrics—are orthogonal to scale and can be applied to larger training runs.

Memory utility. Memory augmentation provides a clear benefit under teacher forcing (+0.1 dB PSNR, -0.019 LPIPS vs. the no-memory baseline). Under autoregressive generation, the baseline outperforms at short horizons, but

the memory model’s PSNR *improves* over time (Table 3), suggesting that memory accumulates useful context that compensates for initial overhead. The state prediction results (Table 4) confirm that memory tokens capture game state information, with accuracy improving as more frames are processed. The high initial error suggests that the current batch-level training (fresh memory per sample) limits the memory’s ability to learn long-range dependencies. Sequential training with unrolled memory across multiple frames within a single optimization step would likely yield substantial improvements.

State loss instability. The state loss exhibits large spikes during training (Figure 2), caused by abrupt changes in game variables at trajectory boundaries (e.g., episode resets where health jumps from 0 to 100). Normalizing game variables to $[0, 1]$ and using smoother loss functions (Huber loss) would address this. We leave this improvement for future work.

Controllability gap for ATTACK/USE. Low controllability for ATTACK and USE actions is expected: these actions have state-dependent visual effects (shooting produces a muzzle flash only when a weapon is available; USE opens a door only when adjacent to one). A richer action-conditioning mechanism that accounts for game state would improve these scores.

Limitations. (1) Our setup trains on a single scenario (“basic”), limiting generalization to other DOOM levels. (2) The memory module is not trained sequentially across frames, limiting its ability to learn temporal dynamics beyond what is captured in a single sample. (3) All GPUs on our machine were partially occupied by other workloads during training and inference, preventing optimal throughput measurements. (4) We do not fine-tune the VAE decoder, which GameNGen found important for HUD text fidelity.

7. Conclusion

We presented MemGameNGen, a memory-augmented action-conditioned latent diffusion model for interactive DOOM simulation. By maintaining a compact GRU-based memory of 16 tokens, the model extends effective history beyond the raw frame context window. An auxiliary state head provides direct supervision for game variable prediction, and state-sensitive evaluation metrics measure aspects of simulation quality that pixel-level metrics miss.

Even at thesis scale ($\sim 120K$ training frames vs. $70M$ in GameNGen), the approach produces stable autoregressive rollouts with meaningful action-following behavior and improving state prediction over time. The memory mechanism, state supervision, and evaluation framework are independent of training scale and directly applicable to larger setups.

Future directions include: (1) sequential memory training with multi-frame unrolling, (2) retrieval-augmented episodic memory for recurring locations, (3) integration of streaming diffusion techniques (Liu et al., 2026) for real-time generation, and (4) scaling to the full DOOM game with diverse scenarios and human play data.

References

- Bruce, J., Dennis, M., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., et al. Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15391*, 2024.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- Duan, H., Yu, H.-X., Chen, S., Fei-Fei, L., and Wu, J. WorldScore: A unified evaluation benchmark for world generation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018.
- He, H., Zhang, Y., Lin, L., Xu, Z., and Pan, L. Pre-trained video generative models as world simulators. *arXiv preprint arXiv:2502.07825*, 2025a.
- He, X. et al. Matrix-game 2.0: An open-source real-time and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025b.
- Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2022.
- Huang, S., Wu, J., Zhou, Q., Miao, S., and Long, M. Vid2world: Crafting video diffusion models to interactive world models. *arXiv preprint arXiv:2505.14357*, 2025.
- Liu, K., Hu, W., Xu, J., Shan, Y., and Lu, S. Rolling forcing: Autoregressive long video diffusion in real time. In *International Conference on Learning Representations (ICLR)*, 2026.
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.
- Song, K., Chen, B., Simchowitz, M., Du, Y., Tedrake, R., and Sitzmann, V. History-guided video diffusion. *arXiv preprint arXiv:2502.06764*, 2025.
- Valevski, D., Leviathan, Y., Arar, M., and Fruchter, S. Diffusion models are real-time game engines. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2408.14837.
- Wydmuch, M., Kempka, M., and Jaśkowski, W. ViZDoom competitions: Playing Doom from pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019.
- Yu, S., Hahn, M., Kondratyuk, D., Shin, J., Gupta, A., Lezama, J., Essa, I., Ross, D., and Huang, J. MALT diffusion: Memory-augmented latent transformers for any-length video generation. *arXiv preprint arXiv:2502.12632*, 2025.
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.
- Zhu, Y. et al. WorldArena: A unified benchmark for evaluating perception and functional utility of embodied world models. *arXiv preprint*, 2026.

A. Additional Training Details

PPO agent. The PPO agent uses a 3-layer CNN (32, 64, 64 channels with strides 4, 2, 1) followed by a linear layer to 256 features. Actor and critic are separate 2-layer MLPs (256→128→output). Training uses GAE ($\gamma=0.99$, $\lambda=0.95$), clip ratio 0.2, value coefficient 0.5, entropy coefficient 0.01, gradient clipping 0.5.

Data format. Each trajectory is stored as a compressed NumPy archive (.npz) containing: frames ($T \times 120 \times 160 \times 3$, uint8), actions (T , int64), and game variables (HEALTH and AMMO2, both float32). The dataset contains 120 trajectories of 1024 frames each (≈ 1.5 GB compressed).

LoRA configuration. LoRA is applied to the query, key, value, and output projection layers of all cross-attention blocks in the UNet. Rank $r=16$ with $\alpha=32$ and dropout 0.05. This yields 3.2M trainable parameters in the UNet LoRA layers. The memory module contributes 5.0M parameters, and the remaining components (action embeddings 0.7M, noise augmentation 0.6M, state head 0.3M, frame encoder 0.04M) bring the total to 9.8M trainable parameters.

B. Full Evaluation Results

Table 6 provides the complete evaluation results.

C. Comparison with GameNGen at Matched Scale

GameNGen’s Appendix A.3 reports that training on smaller subsets of data significantly reduces quality. With 1M examples (vs. 70M), PSNR drops by several dB. Our ~ 120 K frames are an order of magnitude below even their smallest reported subset. The remaining gap is expected to close with (1) more training data from diverse scenarios, (2) longer training (our 10K steps vs. 700K), and (3) full U-Net fine-tuning (vs. LoRA).

D. DDIM Sampling Steps Ablation

Table 7 shows the effect of varying the number of DDIM sampling steps on teacher-forcing PSNR and inference speed. Four steps provides the best quality–speed tradeoff. Increasing beyond 4 steps degrades quality, as the DDIM step schedule traverses noise levels the model was not explicitly trained to handle at those intermediate points. A single denoising step achieves 24.3 FPS (above real-time) but with substantially degraded visual quality (7.1 dB PSNR). Two steps recover most quality at 12.5 FPS, offering a viable low-latency alternative.

Table 6. Complete evaluation results for MemGameNGen.

Metric	Value
<i>Teacher Forcing</i>	
PSNR (dB)	16.10 ± 1.71
LPIPS	0.505 ± 0.107
<i>Autoregressive (10 trajectories)</i>	
FPS	7.48 ± 0.01
PSNR @ 10 frames	15.12
PSNR @ 50 frames	15.20
PSNR @ 100 frames	15.48
PSNR @ 200 frames	15.55
State error @ 10 frames	380.7
State error @ 50 frames	112.8
State error @ 100 frames	80.6
State error @ 200 frames	75.4
<i>Controllability</i>	
Mean controllability	0.350
ATTACK	0.026
USE	0.115
MOVE_LEFT	0.487
MOVE_RIGHT	0.491
MOVE_FORWARD	0.397
MOVE_BACKWARD	0.395
TURN_LEFT	0.446
TURN_RIGHT	0.443
<i>Scripted Tests</i>	
Idle drift (mean MSE)	0.077
Idle drift (max MSE)	0.185
Movement change (mean MSE)	0.087 ± 0.054
<i>Performance</i>	
Inference FPS	7.5
Latency per frame	132.7 ms
<i>No-Memory Baseline</i>	
PSNR (dB, teacher forcing)	15.99 ± 1.63
LPIPS (teacher forcing)	0.524 ± 0.096
Autoregressive PSNR @ 10	16.36
Autoregressive PSNR @ 50	16.47
Inference FPS	7.6

Table 7. DDIM sampling steps ablation. PSNR measured under teacher forcing (50 samples). All measurements on a single NVIDIA H100 GPU.

DDIM steps	PSNR (dB)↑	FPS↑	Latency (ms)
1	7.11 ± 0.32	24.3	41
2	14.77 ± 1.28	12.5	80
4	15.94 ± 0.82	7.7	130
8	15.23 ± 0.79	4.0	250
16	14.69 ± 0.87	2.0	490