WAM-Mini: A Fully Configurable Causal Environment for Training Local World Action Models

Christopher Tardy
University of British Columbia
Vancouver, BC, Canada
ctardy@student.ubc.ca

Frank Wood
University of British Columbia
Vancouver, BC, Canada
fwood@cs.ubc.ca

Abstract

World action models (WAMs) jointly predict future observations and the actions that produce them, with interesting applications such as robotics and 3D simulation. But the action-labelled data needed to train them is scarce and expensive. Compute constraints also limit debugging end-to-end and re-encoding data at scale. We present *WAM-Mini*, a 3D environment that generates video with wall clock-aligned action labels and use it to train a joint video-and-action diffusion transformer on a single 11 GB VRAM GPU. A diffusion model trained in this environment learns an action policy and responds to its own generated actions by predicting frames consistent with the physics of the world. Preliminary results show that model behaviour verified in WAM-Mini transfers to Minecraft, a more complex environment.

1 Introduction

A *world action model* (WAM) is a generative model trained over interleaved observations and actions, so it can both simulate the consequences of an action and propose the next one (Black et al., 2024). WAMs are a promising route to generalized robot policies, as in NVIDIA's DreamZero (Ye et al., 2026), agents that complete long-horizon tasks (Hafner et al., 2023) and games simulated in real time (Valevski et al., 2025). To train these models, action-labelled data is needed but not widely available. Games are often used as training data and while videos are abundant, videos paired with the actions that produced them are not. VPT (Baker et al., 2022) trained an inverse dynamics model (IDM) on about 2,000 hours of labelled Minecraft play and used it to pseudo-label 70,000 hours of online video for behavioural cloning. MIRA (Hu et al., 2026) trains on bot-played Rocket League matches. GameNGen (Valevski et al., 2025) trained an RL agent inside DOOM and collected action-labelled data from its gameplay. Trying to address this shortage, He et al. (2025) created a dataset of over 10,000 hours of human gameplay data for Minecraft.

This work started from a concrete problem: a ~1.5B-parameter multimodal WAM (video, audio, keyboard and mouse) was being trained by the same team as He et al. (2025) with 32 H100 GPUs. With limited compute and data, a team can only test a few hypotheses. The most informative run is a "hero" run that the team can afford only once. When that run produces poor samples, the cause could be the autoencoder, the positional encodings, the attention mask, a hyperparameter or any piece of the architecture that one run cannot tell apart.

This motivated an environment in which runs are cheap enough to act as end-to-end tests. WAM-Mini makes action labels nearly free: a scripted policy plays, and the engine records every frame with the exact keyboard and mouse actions that produced it. It generates about 30 hours of labelled play in about five minutes, at roughly 500 MB per hour of footage.

WAM-Mini renders at any dimension and aspect ratio and comes with 13 validation tasks placed on action boundaries. A video diffusion transformer, VDT (Lu et al., 2024), trained on it generates video and actions jointly. From 10 frames of context, it reconstructs the next 10 frames while learning an action policy whose predicted keys closely match the ground truth (Jaccard distance below 0.1).

Contributions.

- *WAM-Mini*, a fully configurable 3D environment (built in Panda3D) in which every action has a known, deterministic effect, with infrastructure for parallel offscreen data generation and for in-memory online data generation during training.
- A VDT-based WAM that learns both an action policy and a state transition function, generating 10 frames and their actions from 10 frames of context.
- A validation set and metrics that test whether predictions depend causally on the model's inputs: action-boundary tasks, action-swap tests, and a paired test in which only a player embedding and visual cue change.

2 WAM-Mini

Scene and rendering. Rather than recording HD footage and compressing it with a VAE, the Panda3D (Goslin and Mine, 2004) engine renders straight into a low-resolution offscreen buffer (24×40 by default). The scene, coloured pillars on a rainbow-gradient floor seen in first person, is about as visually complex as the Atari games used by DIAMOND (Alonso et al., 2024).

Actions and policy. The action space mirrors the keyboard-and-mouse interface of Minecraft: W/A/S/D, jump, crouch, left/right click (which brighten or darken the scene) and continuous mouse $(\Delta x, \Delta y)$ for the camera. A scripted exploration policy samples an action from a weighted menu and

Table 1. Five of the 13 validation tasks. The action starts on the boundary between observed and generated frames.

Task	Action on the boundary	Rollout (actions)	Rollout (frames)
Walk forward	W pressed	W held	Camera advances; pillars grow
Strafe left	A pressed	A held	Camera slides left; scene shifts right
Jump	Space pressed	Space released	Camera rises, lands within 8 frames
Look up	Mouse moves up	Mouse up, then back down	Camera pitches up, then levels out
Turn around	Mouse sweep after walking	Horizontal mouse sweep	Camera turns to face backwards

runs it for a configurable duration, from a single key press to a short sequence such as walking forward, then turning around. Every event is logged with millisecond timestamps in an SQLite database alongside the rendered frames. Training sets can be generated offline or online every epoch. The frame rate is configurable as well to enable compatibility with different encoders, for example.

Validation tasks. During data generation, 20 5-minute sessions are held out and 13 validation windows are drawn from them, one per task (examples in Table 1). In each 20-frame window, a new action starts exactly on the boundary between the 10 observed and 10 generated frames. Because the world is deterministic, a model that ignores its actions, or follows the wrong action trajectory, diverges from the ground truth on exactly these windows.

3 The World Action Model

We use the VDT video diffusion transformer (Lu et al., 2024) and the codebase of Yoo et al. (2024). The model works directly on normalized pixels, with no VAE or other autoencoder: DIAMOND (Alonso et al., 2024) shows that at low resolution every pixel matters, and in early experiments an off-the-shelf VAE visibly degraded heavily downsampled frames. Each frame $x \in \mathbb{R}^{H\times W\times C}$, with H=24, W=40, C=3, is cut into 4×4 patches, giving 60 video tokens per frame. The frame's action is appended to the video tokens, and all tokens receive spatial and temporal sinusoidal embeddings. Each frame is paired with the action that caused it, so frame s_{t+1} shares a step with a_t and the model learns

$$p(a_t, s_{t+1} \mid s_{\le t}, a_{<t}), \tag{1}$$

the next action together with the frame it produces. This follows GameNGen (Valevski et al., 2025), which denoises each new frame given the action that led to it, $q(o_n \mid o_{<n}, a_{\le n})$.

Factorized attention. Full attention over all tokens of a 20-frame window is expensive, so factorized attention splits attention into a spatial and temporal process. With $X \in \mathbb{R}^{T\times N\times D}$ the tokens of a window, a temporal attention first mixes each token position n across frames, then a spatial attention mixes the tokens within each frame t, followed by a feed-forward layer:

$$X_{:,n} \leftarrow X_{:,n} + W_\tau \,\mathrm{Attn}\big(\mathrm{LN}(X_{:,n})\big), \tag{2}$$

$$X_{t,:} \leftarrow X_{t,:} + \alpha_1 \odot \mathrm{Attn}\big((1+\gamma_1)\odot \mathrm{LN}(X_{t,:}) + \beta_1\big), \tag{3}$$

$$X_{t,:} \leftarrow X_{t,:} + \alpha_2 \odot \mathrm{FFN}\big((1+\gamma_2)\odot \mathrm{LN}(X_{t,:}) + \beta_2\big). \tag{4}$$

Both attentions are bidirectional. The spatial attention and feed-forward layer are modulated by adaptive layer norm with zero-initialized gates (Peebles and Xie, 2023), computed from a conditioning vector c (the timestep embedding, plus optionally the player embedding in §5):

$$(\gamma_1, \beta_1, \alpha_1, \gamma_2, \beta_2, \alpha_2) = \mathrm{MLP}(c). \tag{5}$$

An observation mask (Lu et al., 2024) splits each window into 10 context and 10 target frames. Context frames and actions are clean; target frames and actions start as Gaussian noise and are denoised jointly, with bidirectional attention across the window.

Joint generation. Action vectors are diffused and denoised alongside video, so the model can predict actions as well as condition on them. Training follows DDPM (Ho et al., 2020) with a 1000-step linear schedule, ϵ-prediction and uniform timesteps; sampling uses Heun's sampler with 50 steps (Karras et al., 2022). Other settings are Adam at 10^{-4}, gradient clipping at 1.0, batch size 6, EMA rates 0.999/0.9999 and fp32.

Action encodings. *Raw* uses 8 binary key channels plus a separate 2-d mouse vector with its own loss. *Raw-fused* uses one 10-d vector. *FSQ* uses 36-d codes from a pretrained keyboard-and-mouse tokenizer (Mentzer et al., 2024), with its loss re-weighted by dimension (Eq. 6).

3.1 Loss function

Video and actions each have a mean-squared error on the predicted noise over the target frames, and the two are added. Because each term is a mean, adding them unweighted gives the 10 action values per frame as much weight as its $d_v = 24 \cdot 40 \cdot 3 = 2{,}880$ pixel values. Transfusion (Zhou et al., 2025) balances its two losses with a tuned coefficient λ, and DreamZero (Ye et al., 2026) uses one loss over concatenated video latents and actions, where the compressed latent keeps the mismatch small. In pixel space, equal weighting let the action gradient dominate, and the video predictions collapsed. We instead weight every value equally:

$$\mathcal{L} \;=\; \mathrm{MSE}\big(y^v, \hat{y}^v\big) \;+\; w\,\frac{d_a}{d_v}\,\mathrm{MSE}\big(y^a, \hat{y}^a\big), \tag{6}$$

with both errors averaged over the target frames and d_a the action dimension.

Compute. Each model was trained on an 11 GB RTX 2080 Ti with a peak allocated memory of 9.94 GiB, at about 0.50 s/step for one week.

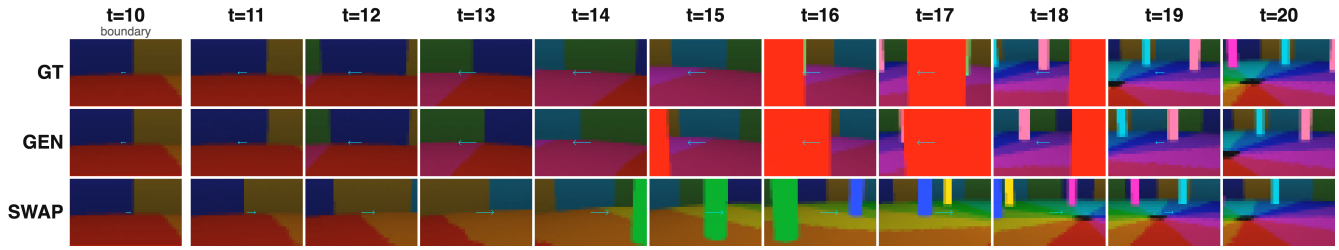


Figure 1. Rollout for the "turn around" task. Columns: the action boundary (t=10) and all 10 generated frames. Rows: ground truth (GT) and predictions under the true (GEN) and swapped (SWAP) actions; SWAP receives an opposite action at frame 10.

Table 2. Held-out validation results. Key J.: Jaccard distance of predicted keys.

	Raw	Raw-fused	FSQ
Action dim.	8+2	10	36
Steps	1.73M	1.29M	1.75M
1-step PSNR / SSIM ↑	**35.1** / 0.961	32.8 / 0.935	34.0 / **0.966**
1-step LPIPS ↓	0.050	0.070	**0.049**
Rollout PSNR ↑	26.3	**26.5**	25.6
Rollout SSIM ↑	0.841	0.834	**0.843**
Rollout LPIPS ↓	0.162	0.161	**0.151**
Key J. (1-step / roll) ↓	**0.00** / 0.096	0.077 / 0.058	**0.00** / **0.047**
Mouse L1, 1-step (trivial)	0.020 (0.89)	**0.010** (0.89)	0.095 (0.98)
Mouse L1, roll (trivial)	**0.138** (0.69)	0.261 (0.69)	0.191 (0.78)
Swap L_2 true vs. swap ↑	**0.091**	0.083	0.072

4 Results

Metrics. Video is scored with PSNR, SSIM and LPIPS (Zhang et al., 2018) on generated frames only: *1-step* on the first generated frame and *rollout* on all 10. Keys are scored with the Jaccard distance, which counts mismatches over all 8 keys, ignoring keys correctly left unpressed. Most keys are not held down most of the time, so a model that never presses anything scores 90% for Hamming distance, hence the need for Jaccard distance. Mouse movement is scored by L_1 error, with the error of a trivial baseline that predicts no movement in parentheses in Table 2. The *swap test* flips the boundary action to its counterpart (W↔S, A↔D, mouse direction negated), lets the model generate the rest, and measures the L_2 distance to the true-action rollout (Fig. 1).

5 Player Embeddings

Different users may follow different action policies $\pi(a \mid s)$, or even experience different transition functions $T(s, a)$, in the same environment. We therefore test whether the model can condition on a specific user.

Setup. Each training session belongs to one of two randomly assigned players. The model sees the player as a learned label embedding $y \in \{y_0, y_1\}$ added to the timestep embedding, so $c = t + y$ in Eq. 5, and as a coloured nametag in every frame. Player 0 (red nametag) tints the scene pink on a left click and green on a right click; player 1 (blue nametag) tints it yellow and blue. No tint is shared between players, so rendering the right one requires knowing the player (Fig. 2).

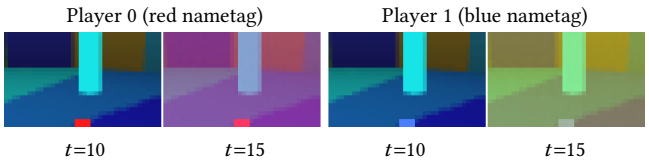


Figure 2. The same left click under each player label. The boundary frames (t=10) differ only in the visual cue and a player embedding.

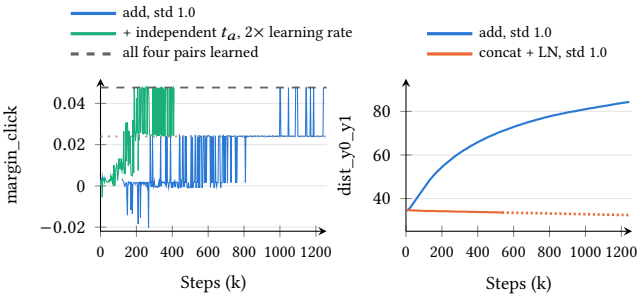


Figure 3. Left: margin_click (Eq. 7) per validation step. The dashed line marks when the model correctly predicts all 4 tint colours. Right: embedding distance between both learned player embeddings over time (dotted: projected).

Paired validation. Two held-out windows (left and right click) are rendered once per player, identical except for the nametag and label. The click starts on the context boundary, so the tint cannot be copied. For each player p, margin_click takes the L_2 distance d over the clicked frames between its rollout $\hat{x}_p$ and the other player's ground truth $x_{\bar{p}}$, minus the distance to its own ground truth x_p:

$$\text{margin_click} = \text{mean}_{\text{windows},\,p}\left[d(\hat{x}_p, x_{\bar{p}}) - d(\hat{x}_p, x_p)\right]. \quad (7)$$

It is near zero when both players are rendered alike and highest when each rollout shows its own player's tints. We also track dist_y0_y1 $= \|y_0 - y_1\|_2$.

What made it learn. During initial experiments, the distance between the learned embeddings collapsed, essentially causing the model to ignore them. Three choices were necessary. *(i) Do not apply LayerNorm.* Adding y to the timestep embedding lets the embeddings move apart, whereas a LayerNorm'd concatenation $[t; y]$ keeps their distance flat and the label unused (Fig. 3, right). *(ii) Initialize at least at std 1.0*, so the label starts large enough to matter next to the timestep embedding. An alternative is initializing the embeddings as orthogonal and scaling them up (we used 25 and 100). *(iii) Independent denoising timesteps.* Sampling the action's noise level separately from the video's, with its own timestep embedding (Chen et al., 2024), potentially provides training examples with action and video generation tasks isolated. Empirically, it was noted that just action conditioning and no generation was a much easier task for the VDT model to learn. With (i) and (ii) alone the players separate slowly; adding (iii), in a run that also doubled the learning rate to 2×10^{-4}, reaches the ceiling much sooner (Fig. 3, left).

6 Transfer to a Real Dataset (PLAICraft)

To test whether behaviour verified in WAM-Mini carries over to real gameplay, we trained the same architecture on PLAICraft (He et al., 2025), a dataset of human Minecraft play, using a variational autoencoder (VAE) latent space for video instead of raw pixels. Figure 4 shows hand-picked validation rollouts. The ego-motion tests in WAM-Mini transfer: conditioned on a forward (W) input at the context boundary, the model moves the scene past the camera and reproduces the bobbing of held items, although frames are blurrier than the ground truth (Fig. 4a). Behaviour that WAM-Mini does not exercise does not transfer: a sheep walking out of frame in the ground truth stays in view in the generated rollout (Fig. 4b). We attribute this to dynamics modelling rather than the VAE, but a test on raw pixels would confirm this. This points to a concrete next step for the debug environment: adding objects that move independently of the player, similar to mobs in Minecraft.

7 Related Work

Game and world models. DreamerV3 (Hafner et al., 2023) learns latent dynamics for control, while GameNGen (Valevski et al., 2025), DIAMOND (Alonso et al., 2024) and Genie (Bruce et al., 2024) generate interactive game frames with diffusion or autoregressive transformers. VDT (Lu et al., 2024) unifies video prediction through mask modelling in a diffusion transformer and, like DiT (Peebles and Xie, 2023), conditions through adaptive layer norm. Diffusion forcing (Chen et al., 2024) gives each token its own noise level, an idea adapted to action tokens in our model.

World action models. Vision-language-action policies such as π_0 (Black et al., 2024) generate actions with flow matching, and DreamZero (Ye et al., 2026) shows that models

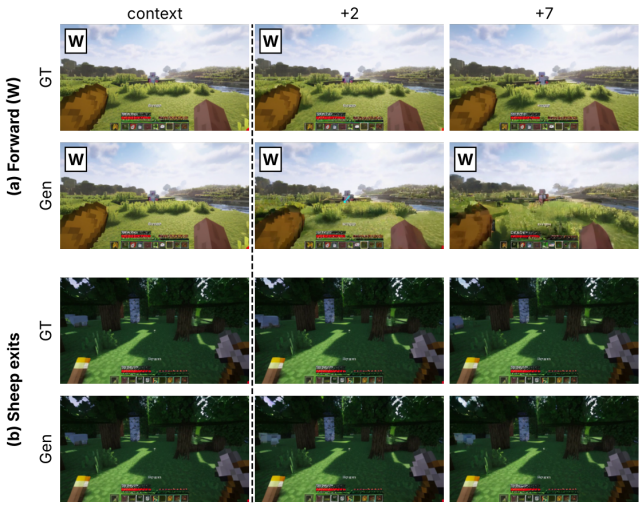


Figure 4. Hand-picked PLAICraft rollouts: last context frame, then frames +2 and +7 generated after it. (a) A forward (W) input at the context boundary produces forward motion and item bobbing. The model also predicts further W keypresses. (b) A sheep (left edge) walks out of frame in the ground truth but stays in view in the generated rollout.

trained jointly over video and actions can act as zero-shot policies. This paper does not propose a new architecture; it provides a controlled setting in which such models can be tested and trained cheaply.

8 Limitations and Outlook

WAM-Mini is deliberately simple: low-entropy visuals, a scripted policy and a deterministic environment. A model that performs well here should not be assumed to perform as well in a more complex environment, even when scaled up on more powerful hardware. However, failure in a complex environment can be addressed by altering the debug environment to be more complex, such as the "sheep test" in Section 6.

The environment's value is as an end-to-end test of a WAM architecture at a scale that many research groups can afford: training data is easily configurable, fast to generate and trains on one consumer GPU. This makes hyperparameter sweeps practical and lets developers run ablation studies on their model before an expensive large-scale run.

It would be useful in the future to test ablations on multiple seeds, providing a confidence interval of performance.

Acknowledgments

We thank the members of the UBC PLAI group for discussions, mentorship, code and compute. We would also like to thank the University of British Columbia for funding the first author's summer research award (SURE) and AML-TN for additional award funding.

References

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. 2024. Diffusion for World Modeling: Visual Details Matter in Atari. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. 2022. Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kevin Black, Noah Brown, Danny Driess, et al. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164* (2024).

Jake Bruce, Michael Dennis, Ashley Edwards, et al. 2024. Genie: Generative Interactive Environments. In *International Conference on Machine Learning (ICML)*.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. 2024. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mike Goslin and Mark R. Mine. 2004. The Panda3D Graphics Engine. *Computer* 37, 10 (2004), 112–114.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104* (2023).

Yingchen He, Christian D. Weilbach, Martyna E. Wojciechowska, Yuxuan Zhang, and Frank Wood. 2025. PLAICraft: Large-Scale Time-Aligned Vision-Speech-Action Dataset for Embodied AI. *arXiv preprint arXiv:2505.12707* (2025).

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Anthony Hu, Václav Volhejn, Adrien Ramanana Rahary, et al. 2026. MIRA: Multiplayer Interactive World Models with Representation Autoencoders. General Intuition, Kyutai and Epic Games. https://github.com/mira-wm/mira.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Haoyu Lu, Guoxing Yang, Nanyi Fei, Yuqi Huo, Zhiwu Lu, Ping Luo, and Mingyu Ding. 2024. VDT: General-purpose Video Diffusion Transformers via Mask Modeling. In *International Conference on Learning Representations (ICLR)*.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2024. Finite Scalar Quantization: VQ-VAE Made Simple. In *International Conference on Learning Representations (ICLR)*.

William Peebles and Saining Xie. 2023. Scalable Diffusion Models with Transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. 2025. Diffusion Models Are Real-Time Game Engines. In *International Conference on Learning Representations (ICLR)*.

Seonghyeon Ye et al. 2026. World Action Models are Zero-shot Policies. *arXiv preprint arXiv:2602.15922* (2026).

Jason Yoo, Yingchen He, Saeid Naderiparizi, Dylan Green, Gido M. van de Ven, Geoff Pleiss, and Frank Wood. 2024. Lifelong Learning of Video Diffusion Models From a Single Video Stream. *arXiv preprint arXiv:2406.04814* (2024).

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. 2025. Transfusion: Predict the Next Token and Diffuse Images with One Multi-Modal Model. In *International Conference on Learning Representations (ICLR)*.