GAMEWAM: A WORLD ACTION MODEL FOR VIDEO GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern video games combine first-person perception, rapid visual changes, persistent world state, and heterogeneous native controls. Existing game agents map visual and task context directly to actions but lack explicit world dynamics modeling, whereas interactive game world models predict visual futures from supplied actions but do not serve as task policies. World Action Models (WAMs) unify these objectives, but remain largely unexplored under the dynamics and open-ended interaction of video games. We introduce GameWAM, to our knowledge the first WAM for native closed-loop gameplay and GUI control. GameWAM jointly generates future visual observations and executable keyboard–mouse trajectories through parallel visual and action generative processes with block-causal conditioning and flow matching. To support joint world–action learning, we construct synchronized gameplay and GUI trajectories. To handle heterogeneous native control, GameWAM predicts a gameplay/GUI mode per action step and generates actions with mode-specific prediction distributions and continuous-action normalization. For long-horizon interaction, block-cycle control coordinates prediction, execution, and temporal context: it predicts beyond the committed horizon, executes short action blocks, replans from new observations, and hierarchically structures context from fine-grained within-cycle history to persistent cross-cycle history. Experiments show competitive task success with fewer executed native actions than the compared agents. We further uncover *Low-Frequency Action Source Imprinting* (LASI), in which low-frequency components of the sampled action source systematically steer coarse generated camera motion under fixed conditioning, revealing a source-sensitivity failure mode in generative control.

1 INTRODUCTION

Modern video games demand pixel-based native control over rapidly changing first-person observations, keyboard–mouse inputs, and the consequences of prior actions. Existing solutions span behavioral pretraining, vision-language-action models, hierarchical agent architectures, and large-scale interaction learning (Baker et al., 2022; Lifshitz et al., 2023; Wang et al., 2024a;b; 2026b; 2025). Despite different system designs, their control ultimately maps visual and task context to native or abstract actions, often through discretized, semantic, or temporally compressed representations. While effective for behavior learning, these objectives do not explicitly model how the visual world evolves under executed actions, and their action abstractions may weaken the fine temporal and metric structure needed for concurrent key control and continuous camera motion. Interactive game world models provide the complementary capability: they learn action-conditioned visual dynamics, but rely on actions supplied by a player or external controller rather than selecting task-directed behavior themselves (Bruce et al., 2024; Alonso et al., 2024; Che et al., 2025; Guo et al., 2025; Zhang et al., 2025). Control models therefore choose behavior without explicitly modeling its visual consequences, while interactive world models predict those consequences without serving as task policies. This motivates a unified model of task-directed actions and their visual consequences.

World Action Models (WAMs) jointly generate future visual observations and executable action trajectories, using visual prediction to provide dynamics-aware supervision for control. Recent WAMs have shown promising results mainly in tabletop or bounded indoor robot manipulation (Ye et al., 2026; Yuan et al., 2026), while their behavior under rapidly changing first-person observations, persistent world state, and repeated closed-loop interaction remains less understood. Modern video

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.

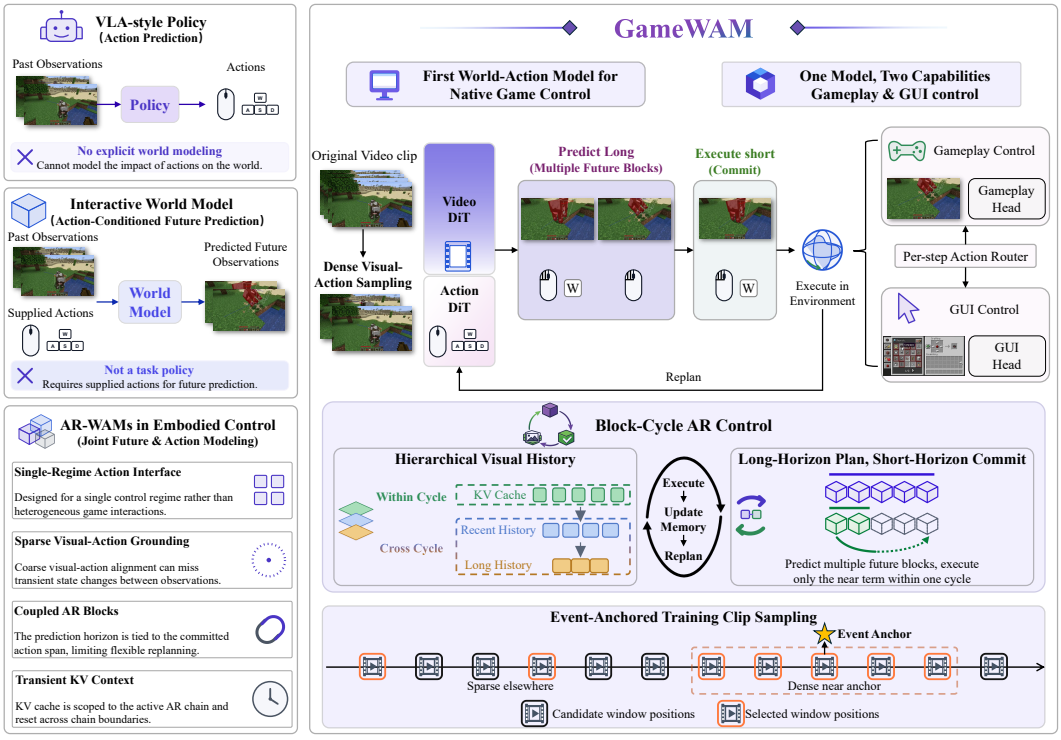


Figure 1: **Overview of GameWAM.** GameWAM jointly models future visual observations and native actions for closed-loop gameplay and GUI control.

games provide a scalable and controllable testbed for this regime: ego-motion, dynamic entities, heterogeneous interaction modes, and persistent state create diverse and abrupt visual changes, while environments remain precisely instrumented, reliably reset, and repeatable under controlled interventions (Fan et al., 2022; Raad et al., 2024). Although games do not reproduce every aspect of physical embodiment, they expose control models to rapidly evolving interaction at a scale difficult to obtain in current physical testbeds (Khazatsky et al., 2024; Pumacay et al., 2024). We therefore introduce GameWAM, to our knowledge the first WAM for native closed-loop gameplay and GUI control. GameWAM jointly generates future visual observations and executable keyboard–mouse trajectories from visual, instruction, and other condition signals.

Realizing a WAM in games introduces challenges along action and temporal dimensions. The action challenge is particularly prominent in native game control: gameplay combines concurrent discrete keys, continuous camera motion, and sparse mouse events, while GUI interaction reuses the same physical channels for cursor motion, clicks, scrolling, and interface shortcuts. Thus, identical action dimensions can have different semantics, scales, and conditional distributions across regimes, and treating them as one undifferentiated distribution can mix incompatible camera and cursor statistics. GameWAM generates continuous and concurrent discrete dimensions within a shared Action DiT flow, while a learned per-action router selects between gameplay- and GUI-specific action-flow predictions at each action timestep. The two regimes use separate normalization for their distinct control distributions while retaining the same underlying physical controls.

The temporal challenge is amplified in rapidly changing interactive environments by visual subsampling relative to native actions. Recent embodied WAMs use temporal subsampling to control video-token cost, but coarse temporal resolution can miss rapid ego-motion, transient targets, and brief interaction events in games. Denser sampling captures these changes, yet under fixed token and cache budgets the same resources cover less interaction time, shortening both future look-ahead and retained past context. This trade-off especially affects chunk-wise autoregressive WAMs like DreamZero (Ye et al., 2026), where prediction, execution, and transient KV accumulation proceed along a finite block chain. On the future side, denser sampling shortens the real-time horizon covered by a fixed chain. Extending the chain to recover longer look-ahead can also lengthen the committed action span and delay feedback from updated observations. GameWAM instead decouples predic-

tion from commitment: each planning unit predicts P actions but commits only an E-step execution block, with $E < P$, before observing and replanning. This predict-long/execute-short organization provides longer action look-ahead while preserving frequent closed-loop feedback. During training, overlapping P-step plans are anchored every E steps, extending supervision beyond each execution block while reusing the same E-spaced temporal organization rather than lengthening the autoregressive chain. On the past side, denser sampling similarly shortens the interaction span represented by a fixed-size K/V cache. Retaining more blocks increases memory, while the transient cache is reset when the finite chain ends, leaving subsequent chains without persistent access to earlier executed observations. GameWAM therefore combines a bounded within-cycle K/V cache with persistent visual history across cycle boundaries: recent executed observations remain explicitly represented, while older information is compressed into long-term history. This preserves fine-grained current-cycle context alongside persistent multi-cycle conditioning under bounded memory.

To support WAM training in this setting, we construct synchronized, WAM-ready gameplay and GUI trajectories. GameWAM models their joint visual and control dynamics with parallel Video and Action DiTs, block-causal attention, and joint flow matching, while event-anchored sampling emphasizes event-rich segments. Across Minecraft and ViZDoom, GameWAM achieves competitive closed-loop performance; in Minecraft, it does so with fewer executed native actions than the compared agents across all evaluated task categories. Our closed-loop evaluation further reveals an unexpected failure mode: when the same sampled action source is reused across replanning steps, some source realizations induce a persistent directional camera bias, including repeated in-place rotation. Controlled frequency-domain interventions reveal that low-frequency components of the sampled action source systematically steer coarse generated camera motion under fixed conditioning. We term this phenomenon *Low-Frequency Action Source Imprinting* (LASI). LASI distinguishes ordinary source-dependent sampling from a pathological low-frequency control bias that can accumulate through repeated closed-loop interaction. Our contributions are:

- We introduce GameWAM, to our knowledge the first World Action Model for native closed-loop gameplay and GUI control. Built on parallel Video and Action DiTs, block-causal attention, and joint flow matching, GameWAM jointly generates future visual observations and executable keyboard–mouse action trajectories.
- We construct synchronized gameplay and GUI trajectories for joint world–action learning, combining standardized gameplay data with generated interface interactions.
- We develop a unified native-action and block–cycle formulation for heterogeneous, long-horizon game interaction. GameWAM supports gameplay and GUI control in a shared keyboard–mouse action space, while block–cycle control decouples long-horizon prediction from short-horizon execution and hierarchically organizes fine-grained within-cycle context and persistent cross-cycle history. Experiments demonstrate competitive closed-loop performance with fewer executed native actions than the compared agents.
- We uncover *Low-Frequency Action Source Imprinting* (LASI), a source-sensitivity failure mode in which low-frequency components of the sampled action source steer coarse generated camera motion under fixed conditioning. Reuse across replanning steps can accumulate into persistent directional bias in closed-loop interaction.

2 Related Work

2.1 Game Agents and Game World Models

Video games provide scalable settings for native control and long-horizon interaction. Earlier work established large-scale game environments, demonstrations, behavioral learning, and open-ended evaluation (Guss et al., 2019; Fan et al., 2022; Baker et al., 2022; Lifshitz et al., 2023; Cai et al., 2024; Zheng et al., 2025; Kempka et al., 2016; Petrenko et al., 2020). Subsequent game agents increasingly emphasize multimodal reasoning, memory, hierarchical control, and generalist pretraining (Wang et al., 2023; 2024a;b; Raad et al., 2024). Recent work more directly addresses action representation and scalable native control. OpenHA (Wang et al., 2026b) systematically studies action abstractions and introduces Chain of Action, using abstract actions as intermediate reasoning steps before executable control. Game-TARS (Wang et al., 2025) scales a unified native keyboard–mouse action space through large-scale continual pretraining across heterogeneous

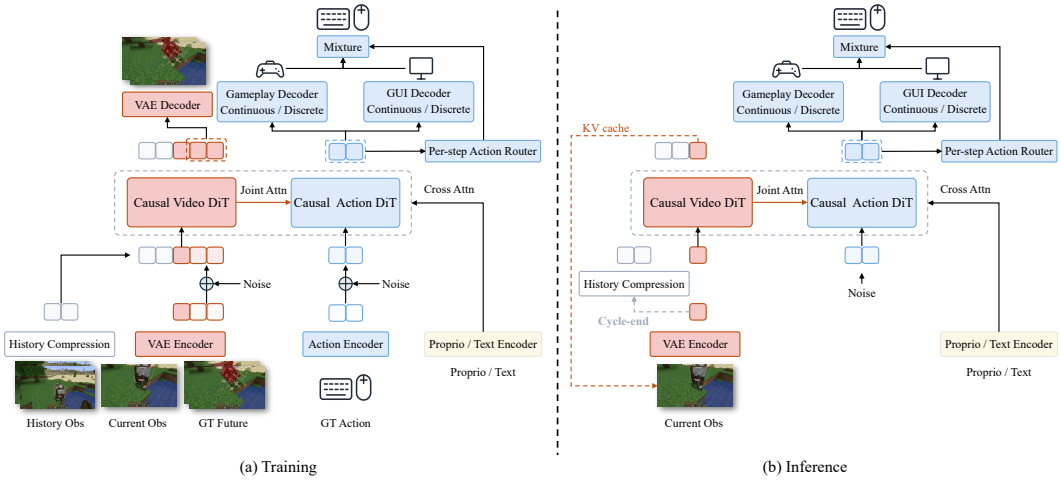


Figure 2: **GameWAM architecture for training and inference.** Parallel Video and Action DiTs jointly model future observations and native actions, with block-causal interaction and gameplay/GUI-specific action generation.

game and computer-use domains, while Lumine (Tan et al., 2025) unifies perception, reasoning, and real-time keyboard–mouse control for long-horizon 3D open-world interaction.

Game world models learn controllable visual dynamics of interactive environments. This line spans latent-action generation, diffusion world modeling, real-time neural game simulation, and increasingly open-world action-conditioned generation (Bruce et al., 2024; Alonso et al., 2024; Valevski et al., 2025; Che et al., 2025; Guo et al., 2025; Zhang et al., 2025). WHAM (Kanervisto et al., 2025) further models gameplay video together with human controller actions. These developments build on broader advances in video diffusion and token-based video generation (Ho et al., 2022; Voleti et al., 2022; Harvey et al., 2022; Villegas et al., 2022; Yu et al., 2023).

2.2 World Action Models

World Action Models connect predictive world modeling with generative control. Learned world models have progressed from latent planning and imagination to scalable transformer and continuous-control formulations (Hafner et al., 2019b;a; 2020; Schrittwieser et al., 2020; Micheli et al., 2023; Zhang et al., 2023; Hansen et al., 2024; Hafner et al., 2025). Generative control has similarly developed structured action prediction through chunking, diffusion, latent actions, and large-scale vision–language–action learning (Zhao et al., 2023; Chi et al., 2023; Lee et al., 2024; Brohan et al., 2023b;a; O'Neill et al., 2024; Ghosh et al., 2024), while video- and latent-action methods connect predicted trajectories to control (Du et al., 2023; 2024; Wu et al., 2024; Ye et al., 2025). Diffusion Transformers and flow-based objectives provide scalable generative backbones (Peebles & Xie, 2023; Ma et al., 2024; Lipman et al., 2023). Related world–action formulations have also been explored in offline model-based RL (Cheng et al., 2025).

More recent generative WAMs directly couple visual prediction with executable action generation and increasingly revisit inference-time execution. DreamZero (Ye et al., 2026) jointly predicts future video and actions for closed-loop control, Fast-WAM (Yuan et al., 2026) retains video cotraining while removing future-video generation at test time, and FFDC-WAM (Wang et al., 2026a) adapts execution by verifying predicted futures against realized observations. GameWAM studies this emerging paradigm in native closed-loop gameplay and GUI control.

3 Data Construction and Training Sampling

Joint world–action learning requires aligned observations and native controls. For Minecraft, we construct three complementary WAM-ready training streams. Regular VPT trajectories (Baker et al., 2022) provide broad naturalistic interaction coverage and long temporal context. From the same recordings, we additionally construct an Event-Anchored VPT dataset by identifying MineStudio-

style interaction events (Cai et al., 2024) from state transitions and forming instruction-aligned event-centered trajectories. We further generate MineStudio-based scripted GUI trajectories to broaden interface-interaction coverage. All streams are standardized on a common interaction timeline into synchronized observations, state, and native keyboard–mouse actions with a control-mode label. Dataset construction details are provided in Section B.

Event-anchored training clip sampling. Dataset construction determines the event-centered trajectories available for training, whereas clip sampling separately determines their temporal sampling density. Long interaction trajectories contain highly uneven supervision density: transitions near an annotated event often contain the visual and action changes most relevant to the instruction, while distant portions may consist largely of traversal, incidental camera motion, or other weakly related behavior. Uniform sampling would therefore devote substantial training capacity to less informative regions. Within the Event-Anchored VPT stream, we instead sample training clips densely around the event anchor and more sparsely away from it, increasing exposure to event-relevant transitions while retaining surrounding interaction context. The sampling configuration is in Section C.

4 METHOD

GameWAM is a block-causal world–action model for native gameplay and GUI control. Figs. 1 to 3 summarize the model and rollout.

4.1 BLOCK-CAUSAL WORLD–ACTION MODELING

At block k of cycle c, define the conditioning context as

$$\Gamma_{c,k} = (C_{c,k}, \ell, H_c, s_{c,k}), \tag{1}$$

where $C_{c,k}$ denotes the clean within-cycle observation context, ℓ the language or task instruction, H_c the persistent cross-cycle visual history, and $s_{c,k}$ the optional proprioceptive state. The complete clean visual prefix used by block-causal attention is

$$\mathcal{P}_{c,k} = H_c \cup C_{c,k}. \tag{2}$$

A world–action block is $B_j = (\mathcal{V}_j, \mathcal{A}_j)$, with video latents $\mathcal{V}_j$ and native actions $\mathcal{A}_j = [\mathcal{A}_j^{\text{cont}}; \mathcal{A}_j^{\text{disc}}]$. For a plan spanning R blocks, GameWAM models the block-causal joint distribution

$$p_\theta(B_{k:k+R-1} \mid \Gamma_{c,k}) = \prod_{r=0}^{R-1} p_\theta(B_{k+r} \mid \Gamma_{c,k}, B_{k:k+r-1}) \,. \tag{3}$$

The attention mask determines how the two modalities exchange information within each conditional factor. We parameterize visual and action processes with joint flow matching (Lipman et al., 2023) using parallel Video- and Action DiT branches (Peebles & Xie, 2023; Ma et al., 2024). Continuous and discrete action coordinates remain components of one native action vector; discrete ones are converted to executable binary decisions only after generation. For modality $m \in \{v, a\}$,

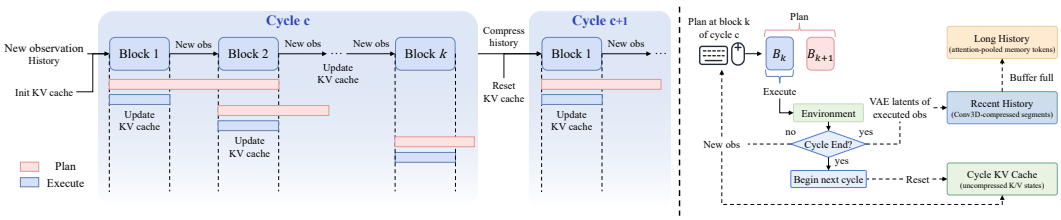
$$X_{\sigma_m}^m = (1 - \sigma_m) X_0^m + \sigma_m \epsilon^m, \qquad U^m = \epsilon^m - X_0^m. \tag{4}$$

The parallel DiTs estimate $(\widehat{U}_\theta^v, \widehat{U}_\theta^a)$ from the noisy modalities, their noise levels, and the causal context. At inference, $X_1^m \sim \mathcal{N}(0, I)$ and the learned ODE is integrated from $\sigma = 1$ to 0.

Block-causal video–action masking. For target block j, let $\mathcal{P}_{c,j}$ denote its causally available clean visual prefix, and let $\mathcal{V}_j$ and $\mathcal{A}_j$ denote the noisy video and action variables of that block. We adopt a Fast-WAM-style (Yuan et al., 2026) modality-decoupled mask, under which both modalities share the same clean prefix while their noisy variables do not condition one another:

$$\text{Vis}(\mathcal{A}_j) = \mathcal{P}_{c,j} \cup \mathcal{A}_j, \qquad \text{Vis}(\mathcal{V}_j) = \mathcal{P}_{c,j} \cup \mathcal{V}_j. \tag{5}$$

The Video DiT encodes $\mathcal{P}_{c,j}$ into layer-wise K/V states shared by both branches, so video and action supervision jointly shape the context used for action generation. Instruction and proprioception remain available through their corresponding conditioning paths. This preserves video co-training while avoiding future-video denoising during action-only online inference. We compare this default design with joint within-block video–action attention in Section D.2.



(a) Block-Cycle Autoregressive Control (b) Hierarchical History Design

Figure 3: **Block–cycle control and hierarchical history.** GameWAM predicts beyond the committed horizon while executing only a short prefix, and combines a transient cycle-local K/V cache with persistent cross-cycle visual history.

Per-action routing. Gameplay and GUI interaction share the same physical keyboard–mouse coordinates but induce different conditional action distributions. GameWAM therefore uses a timestep-wise router with gameplay- and GUI-specific prediction branches. Let $r_\tau^* \in \{0, 1\}$ denote the observed interaction mode and ρ_τ the routing logit. At rollout, the predicted route $\hat{r}_\tau$ selects the action-flow prediction for the entire native action vector at timestep τ:

$$\hat{r}_\tau = \mathbf{1}\left[\mathrm{sigmoid}(\rho_\tau) > \tfrac{1}{2}\right], \qquad \widehat{U}_\tau^a = (1 - \hat{r}_\tau)\widehat{U}_\tau^{\mathrm{game}} + \hat{r}_\tau \widehat{U}_\tau^{\mathrm{gui}}. \tag{6}$$

During training, r_τ^* selects the supervised prediction branch and also provides the routing target; at rollout, the predicted route is used. Continuous action coordinates are normalized with gameplay- or GUI-specific statistics, whereas discrete coordinates retain the shared normalization. Validity masks exclude controls unavailable in a given environment.

4.2 Block-Cycle Control and Hierarchical History

As illustrated in Fig. 3, block–cycle control organizes prediction, execution, and temporal context. A plan contains P actions, whereas only its first E actions form the committed execution block:

$$\widehat{\mathbf{A}}_{c,k}^{\mathrm{plan}} \in \mathbb{R}^{P \times d_a}, \qquad E < P, \qquad \widehat{\mathbf{A}}_{c,k}^{\mathrm{exec}} = \widehat{\mathbf{A}}_{c,k}^{\mathrm{plan}}[1{:}E]. \tag{7}$$

The plan spans $R = \lceil P/E \rceil$ execution intervals. After executing one block, the agent receives a new observation and replans the overlapping $P - E$ suffix. Only observations realized through executed interaction become clean causal context; unexecuted look-ahead is discarded. During training, overlapping P-step plans anchored every E steps are formed from the same trajectory and teacher-forced in parallel under block-causal visibility, avoiding sequential rollout of the additional look-ahead.

GameWAM separates within-cycle realized context from persistent cross-cycle history. Online, both form the clean prefix and its transient K/V cache; persistent history comprises recent and long-term visual memory.

Cycle K/V cache. During online rollout, $\mathcal{P}_{c,k}$ is materialized as layer-wise K/V states. At cycle start, the cache is initialized from cross-cycle history and current context; newly realized observations then extend only its current-cycle portion. The cache is discarded at the cycle boundary and reinitialized from $\mathcal{P}_{c+1,0}$; persistent information survives through H_{c+1}.

Recent history. At a cycle boundary, the realized visual segment produced by executed actions is encoded by the video VAE and compressed by a Conv3D tokenizer:

$$S_c = \mathrm{Conv3D}(\mathrm{VAE}(O_c^{\mathrm{exec}})), \qquad (R_{c+1}, \bar{S}_c) = \mathrm{FIFO}_{K_R}(R_c, S_c). \tag{8}$$

The buffer R_c retains the latest K_R segments; $\bar{S}_c$ denotes the evicted segment when the buffer is full. Recent history therefore remains explicitly addressable at a compressed spatial resolution.

Long-term history. An evicted recent segment updates a fixed set of long-term memory slots through attention pooling and a learned slot-wise gate:

$$\widetilde{M}_c = \mathrm{Attn}(M_c, \bar{S}_c, \bar{S}_c), \qquad M_{c+1} = \mathrm{LN}\left[M_c + g_c \odot (\widetilde{M}_c - M_c)\right]. \tag{9}$$

If no segment is evicted, $M_{c+1} = M_c$. Learned slot and level embeddings assign distinct temporal roles to different memory slots. The resulting persistent history $H_c = [M_c; R_c]$ is retained across interaction-cycle boundaries, unlike the transient cycle-local K/V state.

4.3 Training Objective

Training teacher-forces completed blocks with rollout block-causal visibility. The video flow loss is

$$\mathcal{L}_v = \mathbb{E}\Big[w_v(\sigma_v)\,\|\widehat{U}^v - U^v\|_2^2\Big]. \tag{10}$$

For the selected interaction branch, the Action DiT jointly predicts the complete native action vector. To prevent either continuous or discrete coordinates from dominating, they are reduced separately. For $g \in \{\mathrm{cont}, \mathrm{disc}\}$,

$$\begin{gathered}\mathcal{L}_a^g = \mathbb{E}\left[w_a(\sigma_a)\frac{\sum_{\tau,d\in\mathcal{D}_g} m_{\tau d}(\widehat{U}^a_{\tau d} - U^a_{\tau d})^2}{\sum_{\tau,d\in\mathcal{D}_g} m_{\tau d}}\right], \quad g \in \{\mathrm{cont}, \mathrm{disc}\}; \\ \mathcal{L}_a = \lambda_{\mathrm{cont}}\mathcal{L}_a^{\mathrm{cont}} + \lambda_{\mathrm{disc}}\mathcal{L}_a^{\mathrm{disc}}.\end{gathered} \tag{11}$$

Here $m_{\tau d}$ masks padded or unavailable action coordinates, and $\widehat{U}^a$ is selected by the observed interaction mode during training. Routing is supervised via masked binary cross-entropy,

$$\mathcal{L}_{\mathrm{mode}} = \frac{\sum_\tau \nu_\tau\ \mathrm{BCE}(\rho_\tau, r_\tau^*)}{\sum_\tau \nu_\tau}, \tag{12}$$

where ν_τ indicates whether the interaction-mode label is valid. To keep compressed history informative, we predict the current clean visual feature q_c from persistent history:

$$\mathcal{L}_{\mathrm{hist}} = D_{\mathrm{pred}}(F(\mathrm{Pool}(H_c)), \mathrm{sg}(q_c))\,, \tag{13}$$

where D_{pred} combines cosine and squared-error distances. This regularizes H_c to retain cross-cycle predictive visual information, while stop-gradient keeps the target fixed.

$$\mathcal{L} = \lambda_v\mathcal{L}_v + \lambda_a\mathcal{L}_a + \lambda_m\mathcal{L}_{\mathrm{mode}} + \lambda_h\mathcal{L}_{\mathrm{hist}}. \tag{14}$$

5 Experiments

5.1 Evaluation Setup

We evaluate GameWAM on two complementary closed-loop benchmarks: Minecraft Universe (MCU) and the four-map ViZDoom suite. MCU evaluates long-horizon task completion under native control using task success and executed environment actions, jointly measuring control effectiveness and interaction efficiency. ViZDoom adds fast visual dynamics and continuous interaction, evaluated by average episode reward. Further details are provided in Section C.

5.2 Main Results on MCU

Table 1 shows that GameWAM achieves the highest average success rates (ASR) on both the Mini and full task sets while requiring substantially fewer steps per successful episode across all task categories. The interaction-efficiency gains are particularly pronounced on embodied and GUI tasks.

5.3 Main Results on ViZDoom

Fig. 4 evaluates GameWAM under ViZDoom's distinct dynamics and native control. GameWAM consistently improves over Game-TARS and achieves competitive or leading average rewards relative to the compared multimodal agents across all four scenarios. Together with the MCU results, this demonstrates the formulation's effectiveness across diverse game environments.

5.4 Ablation Analysis

Table 2 shows that removing future-video supervision or using coarser temporal sampling produces the largest performance degradation. Event-anchored clip sampling also contributes substantially, consistent with emphasizing behaviorally informative transitions during training. A unified gameplay/GUI action distribution underperforms the mode-specific formulation, supporting separate prediction and normalization for heterogeneous native controls. Matching the prediction and execution

Table 1: **Evaluation results of Minecraft agents on the MCU benchmark with over 800 tasks.** For a direct WAM-to-WAM comparison, we reproduce Fast-WAM (Yuan et al., 2026) on the same game training data and MCU setup as GameWAM; results for all other baselines are taken from their original reports. For each category, we report average native interaction steps over successful episodes, ASR on the Mini subset (ASR Mini), and ASR over all tasks (ASR All). The *ASR Mini* and *ASR All* average 10 and 5 runs per task, respectively. Game PT. denotes large-scale policy pretraining or continual pretraining on interaction data spanning many game environments; single-game training and generic foundation-model pretraining are excluded. Results highlighted in red and blue denote the best and second-best performance, respectively.

Model	Game PT.	Embodied Tasks			GUI Tasks			Combat Tasks			Avg	
		Steps ↓	ASR Mini ↑	ASR All ↑	Steps ↓	ASR Mini ↑	ASR All ↑	Steps ↓	ASR Mini ↑	ASR All ↑	Mini ↑	All ↑
Instruction-Conditioned Policies												
VPT	×	377	$10.1^{\pm 3.6}$	$6.0^{\pm 11.4}$	398	$0.7^{\pm 0.1}$	$0.8^{\pm 3.3}$	396	$3.6^{\pm 7.7}$	$3.6^{\pm 7.7}$	4.8	3.5
STEVE-1	×	384	$8.4^{\pm 3.0}$	$8.0^{\pm 17.0}$	391	0.0	$3.2^{\pm 8.4}$	395	$4.9^{\pm 1.8}$	$3.9^{\pm 12.0}$	4.4	5.0
ROCKET-1	×	392	$19.2^{\pm 6.1}$	$18.9^{\pm 24.3}$	–	0.0	0.0	320	$29.8^{\pm 9.0}$	$27.9^{\pm 29.3}$	16.3	15.6
JARVIS-VLA	×	305	$31.0^{\pm 12.7}$	$30.0^{\pm 35.4}$	339	$25.3^{\pm 5.7}$	$25.1^{\pm 23.9}$	352	$18.3^{\pm 5.2}$	$18.5^{\pm 22.7}$	24.9	24.5
VLM-based Agents												
LatentHA	×	363	$27.3^{\pm 37.4}$	$24.4^{\pm 31.1}$	393	$3.5^{\pm 8.7}$	$3.0^{\pm 7.5}$	371	$8.2^{\pm 15.6}$	$8.5^{\pm 17.9}$	13.0	12.0
MotionHA	×	336	$31.6^{\pm 10.1}$	$27.4^{\pm 35.2}$	–	0.0	0.0	392	$9.1^{\pm 3.9}$	$4.3^{\pm 10.8}$	13.6	10.6
GroundingHA	×	290	$39.7^{\pm 13.7}$	$37.1^{\pm 38.5}$	380	$3.7^{\pm 2.3}$	$6.7^{\pm 10.8}$	346	$28.2^{\pm 6.2}$	$26.5^{\pm 23.4}$	23.9	23.4
SkillHA	×	365	$13.8^{\pm 7.7}$	$11.3^{\pm 14.5}$	397	$3.4^{\pm 0.8}$	$6.3^{\pm 9.2}$	393	$3.4^{\pm 0.8}$	$6.5^{\pm 9.3}$	6.9	8.0
TextVLA	×	321	$23.9^{\pm 8.9}$	$27.0^{\pm 17.0}$	291	$14.0^{\pm 4.1}$	$25.8^{\pm 14.3}$	317	$27.1^{\pm 11.8}$	$10.0^{\pm 6.1}$	21.7	20.9
OpenHA	×	287	$37.0^{\pm 15.9}$	$30.1^{\pm 13.9}$	314	$33.3^{\pm 13.3}$	$32.5^{\pm 9.2}$	316	$40.0^{\pm 19.6}$	$31.9^{\pm 13.7}$	36.8	31.5
Game-TARS	✓	373	–	$50.4^{\pm 20.7}$	406	–	$39.1^{\pm 27.5}$	372	–	$38.1^{\pm 24.6}$	–	42.5
World Action Models												
Fast-WAM	×	165	$66.0^{\pm 25.4}$	$41.2^{\pm 36.6}$	234	$23.0^{\pm 19.5}$	$18.4^{\pm 22.1}$	261	$31.0^{\pm 29.1}$	$16.3^{\pm 22.4}$	40.0	25.2
GameWAM	×	138	$70.0^{\pm 32.2}$	$47.5^{\pm 36.0}$	155	$43.0^{\pm 32.9}$	$60.0^{\pm 38.6}$	203	$39.0^{\pm 25.9}$	$32.2^{\pm 30.2}$	50.7	46.6

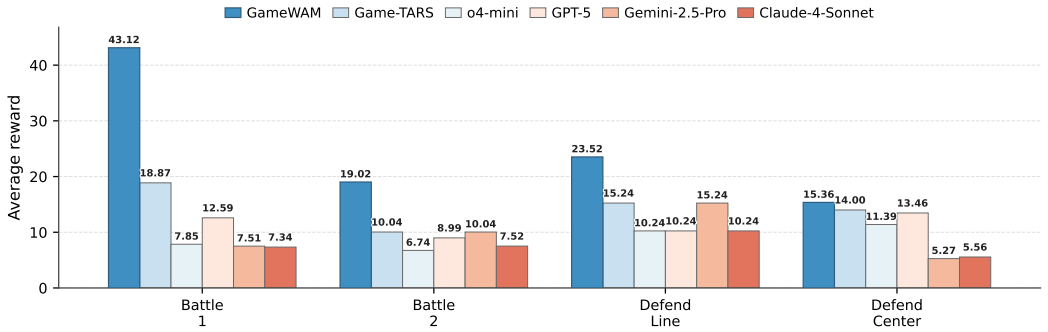


Figure 4: **Evaluation results on the four-map ViZDoom FPS benchmark.** The reported average reward for each map is calculated over 50 episodes.

horizons ($P = E$) further degrades ASR, indicating a benefit from predicting beyond the committed horizon while retaining frequent feedback. Cross-cycle history has a smaller and task-dependent effect: GUI and combat performance declines without it, whereas the embodied point estimate increases slightly, suggesting greater value for tasks that rely on information across replanning cycles.

5.5 Low-Frequency Action Source Imprinting

Our investigation of action-source sensitivity arose from a closed-loop failure observed during evaluation. Reusing the same sampled source across replanning cycles caused some realizations to induce persistent directional camera bias and repeated in-place rotation, in severe cases leaving almost no task completion. Resampling between cycles largely removed this pattern, motivating a controlled study of how source variation shapes generated actions. Source dependence is expected in diffusion and flow-based generation because the sampled source initializes the generative trajectory and selects among possible outputs. LASI is more structured: under fixed conditioning, low-temporal-frequency source components coherently steer coarse camera motion. We therefore fix the visual, historical, and proprioceptive context. Let $X_0, Z \in \mathbb{R}^{H_a \times D_a}$ denote a generated continuous-action

Table 2: **GameWAM ablations on MCU Mini.**

Variant	Embodied	GUI	Combat	Avg
Full GameWAM	$70.0^{\pm 32.2}$	$\mathbf{43.0}^{\pm \mathbf{32.9}}$	$\mathbf{39.0}^{\pm \mathbf{25.9}}$	**50.7**
Action-only supervision	$60.0^{\pm 27.6}$	$34.0^{\pm 35.8}$	$13.0^{\pm 13.5}$	35.7
Coarser temporal sampling	$60.0^{\pm 29.7}$	$34.0^{\pm 40.0}$	$16.0^{\pm 36.7}$	36.7
No event-anchored clip sampling	$64.0^{\pm 28.7}$	$33.0^{\pm 30.0}$	$17.0^{\pm 14.9}$	38.0
Unified action distribution	$59.0^{\pm 25.5}$	$35.0^{\pm 27.7}$	$21.0^{\pm 29.1}$	38.3
Matched prediction–execution horizon ($P = E$)	$63.0^{\pm 33.8}$	$32.0^{\pm 36.8}$	$29.0^{\pm 28.1}$	41.3
No cross-cycle history	$\mathbf{75.0}^{\pm \mathbf{30.1}}$	$31.0^{\pm 29.8}$	$34.0^{\pm 31.0}$	46.7

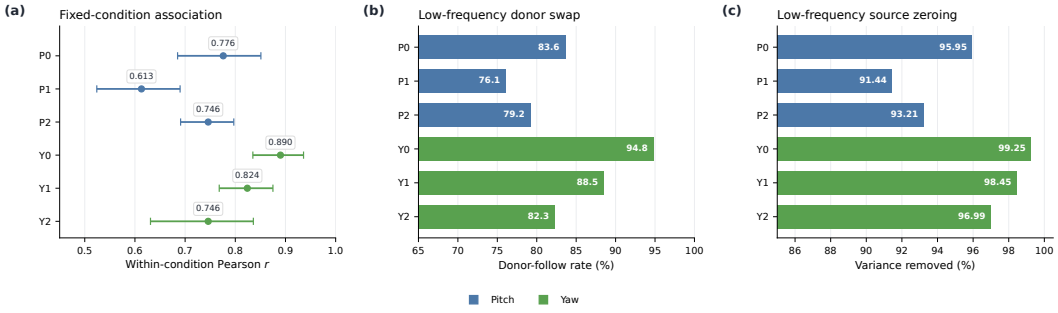


Figure 5: **Controlled evidence for LASI.** Fixed-condition association, low-frequency source replacement, and source zeroing quantify the influence of sampled low-frequency components on generated camera motion.

chunk and its sampled source. We apply an orthonormal DCT along the action horizon,

$$\widetilde{X}_0 = CX_0, \qquad \widetilde{Z} = CZ, \qquad C^\top C = I, \tag{15}$$

where modes 0–2 form the low-frequency band.

Fig. 5 provides three complementary tests. Under fixed conditioning, matching low-frequency source and action coefficients remain strongly associated, with yaw DCT0 reaching $r = 0.890$. Replacing only source modes 0–2 makes the corresponding yaw DCT0 output follow the donor in 94.8% of trials, while zeroing the same band removes 99.25% of the associated output variance. Together, these interventions establish causal evidence that low-frequency action-source components steer coarse generated camera motion rather than a correlation induced by changing conditions.

LASI is therefore distinct from ordinary sampling diversity: a sampled source can acquire a coherent low-frequency control bias that becomes harmful when reused through repeated closed-loop interaction. In our closed-loop evaluation, we consequently resample the action source at each replanning step instead of reusing a fixed source throughout the rollout. This mitigates coherent episode-level accumulation without removing the underlying source sensitivity. Frequency selectivity, single-forward transfer, model-path amplification, and closed-loop alignment are in Section D.5.

6 Conclusion

We introduced GameWAM, a World Action Model for native closed-loop gameplay and GUI control that jointly models future visual observations and executable keyboard–mouse actions. Across Minecraft and ViZDoom, GameWAM achieves competitive closed-loop performance, while using fewer executed native actions than the compared agents in Minecraft. We further identify *Low-Frequency Action Source Imprinting* (LASI), showing that sampled low-frequency action-source structure can induce persistent control bias when reused across replanning steps. Together, these results support joint world–action modeling as a promising direction for interactive game control while exposing a source-sensitivity failure mode specific to generative action policies.

AI USE STATEMENT

Generative AI tools were used for language polishing and editorial assistance, including improving grammar, clarity, concision, and academic phrasing, as well as refining the presentation of author-produced experimental results. The core research questions, method, model design, data construction, implementation, experimental results, and numerical measurements were developed or produced by the authors. All AI-assisted text was manually reviewed and checked against the implementation, experimental records, and underlying results, and was revised by the authors where necessary. The authors take full responsibility for the final content of this work, including all claims, analyses, and artifacts.

ETHICS STATEMENT

This work studies learned closed-loop control in simulated video-game environments. We conduct no new human-subject experiments and collect no sensitive personal data as part of this study. The reported experiments are restricted to digital game environments and do not involve deployment in physical systems. While GameWAM is developed for research on native game control, models capable of operating general keyboard–mouse interfaces could in principle be adapted to other interactive software; deployment beyond the controlled environments studied here should therefore consider authorization, safety, and potential misuse.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model formulation, training objectives, and closed-loop control procedure in the main text and Section A. Section B documents trajectory alignment, native action interfaces, training data construction, and training-sequence semantics, while Section C reports the architecture, temporal and history configurations, training-data mixture and clip-sampling settings, optimization details, objective weights, sampling configuration, and evaluation protocols. Additional quantitative and qualitative analyses are provided in Section D.

We will publicly release the training and evaluation code, data-processing scripts, constructed datasets, trained GameWAM model weights, and associated configuration files. The released resources will include the complete procedures required to reproduce the reported experiments, subject to applicable third-party licensing and redistribution requirements. Together with the detailed methodological descriptions and experimental protocols provided in this paper, these resources are intended to facilitate reproduction of our results and further research on world action models for interactive game control.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023a.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, 2023b. doi: 10.15607/RSS.2023.XIX.025.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first international conference on machine learning*, 2024.

Shaofei Cai, Zhancun Mu, Kaichen He, Bowei Zhang, Xinyue Zheng, Anji Liu, and Yitao Liang. Minestudio: A streamlined package for minecraft ai agent development. *arXiv preprint arXiv:2412.18293*, 2024.

Haoxuan Che, Xuanhua He, Quande Liu, Cheng Jin, and Hao Chen. Gamegen-x: Interactive open-world game video generation. In *International Conference on Learning Representations*, volume 2025, pp. 37546–37593, 2025.

Jie Cheng, Ruixi Qiao, Yingwei Ma, Binhua Li, Gang Xiong, Qinghai Miao, Yongbin Li, and Yisheng Lv. Scaling offline model-based rl via jointly-optimized world-action model pretraining. In *International Conference on Learning Representations*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yilun Du, Sherry Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B. Tenenbaum, Leslie Pack Kaelbling, Andy Zeng, and Jonathan Tompson. Video language planning. In *International Conference on Learning Representations*, 2024.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024.

Junliang Guo, Yang Ye, Tianyu He, Haoyu Wu, Yushu Jiang, Tim Pearce, and Jiang Bian. Mineworld: a real-time and open-source interactive world model on minecraft. *arXiv preprint arXiv:2504.08388*, 2025.

William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *Advances in neural information processing systems*, 35: 27953–27965, 2022.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Anssi Kanervisto, Dave Bignell, Linda Yilin Wen, Martin Grayson, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Tabish Rashid, Tim Pearce, Yuhan Cao, et al. World and human action models towards gameplay ideation. *Nature*, 638(8051):656–663, 2025.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE conference on computational intelligence and games (CIG)*, pp. 1–8. IEEE, 2016.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.120.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26991–27008. PMLR, 2024.

Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36: 69900–69929, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation*. IEEE, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav Sukhatme, and Vladlen Koltun. Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning. In *International Conference on Machine Learning*, pp. 7652–7662. PMLR, 2020.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. THE COLOSSEUM: A benchmark for evaluating generalization for robotic manipulation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.133.

Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, et al. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*, 2024.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Weihao Tan, Xiangyang Li, Yunhao Fang, Heyuan Yao, Shi Yan, Hao Luo, Tenglong Ao, Huihui Li, Hongbin Ren, Bairen Yi, Yujia Qin, Bo An, Libin Liu, and Guang Shi. Lumine: An open recipe for building generalist agents in 3d open worlds. *arXiv preprint arXiv:2511.08892*, 2025.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, volume 2025, pp. 73754–73776, 2025.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in neural information processing systems*, 35:23371–23385, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Rui Wang, Yue Zhang, Jiehong Lin, Kuncheng Luo, Jianan Wang, Zhongrui Wang, and Xiaojuan Qi. When to trust imagination: Adaptive action execution for world action models. *arXiv preprint arXiv:2605.06222*, 2026a.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1894–1907, 2024a.

Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Ma, and Yitao Liang. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents. *Advances in Neural Information Processing Systems*, 37:73278–73308, 2024b.

Zihao Wang, Xujing Li, Yining Ye, Junjie Fang, Haoming Wang, Longxiang Liu, Shihao Liang, Junting Lu, Zhiyong Wu, Jiazhan Feng, Wanjun Zhong, Zili Li, Yu Wang, Yu Miao, Bo Zhou, Yuanfan Li, Hao Wang, Zhongkai Zhao, Faming Wu, Zhengxuan Jiang, Weihao Tan, Heyuan Yao, Shi Yan, Xiangyang Li, Yitao Liang, Yujia Qin, and Guang Shi. Game-TARS: Pretrained foundation models for scalable generalist multimodal game agents. *arXiv preprint arXiv:2510.23691*, 2025.

Zihao Wang, Muyao Li, Kaichen He, Xiangyu Wang, Zhancun Mu, Minghao Liu, Anji Liu, and Yitao Liang. OpenHA: A series of open-source hierarchical agentic models in minecraft. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10459–10469. IEEE, 2023.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. STORM: Efficient stochastic transformer based world models for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yifan Zhang, Chunli Peng, Boyang Wang, Puyi Wang, Qingcheng Zhu, Fei Kang, Biao Jiang, Zedong Gao, Eric Li, Yang Liu, et al. Matrix-game: Interactive world foundation model. *arXiv preprint arXiv:2506.18701*, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Xinyue Zheng, Haowei Lin, Kaichen He, Zihao Wang, Qiang Fu, Haobo Fu, Zilong Zheng, and Yitao Liang. Mcu: an evaluation framework for open-ended game agents. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

APPENDIX CONTENTS

A Extended Method **16**

A.1 Teacher-Forced Training and Online Rollout 16

A.2 Temporal Context Within and Across Interaction Cycles 17

A.3 Cross-Cycle Hierarchical Memory 17

A.4 Block-Causal Multimodal Computation 19

A.5 Interaction-Conditioned Native Action Generation 21

A.6 Supervision Semantics Beyond the Compact Objective 21

B Data and Training Construction **22**

B.1 Trajectory Alignment 22

B.2 Native Action Interfaces 22

B.3 Training Data Sources 23

B.4 Event-Anchored VPT Dataset Construction 24

B.5 Scripted GUI Trajectory Construction 24

B.6 Event-Anchored Training Clip Sampling 24

C Implementation and Evaluation Protocol **25**

C.1 Model Architecture and Initialization 25

C.2 Observation, Planning, and Temporal-Context Configuration 25

C.3 Training Data Mixture and Clip-Sampling Configuration 25

C.4 Optimization and Objective Weighting 26

C.5 Flow-Training and Sampling Configuration 26

C.6 Closed-Loop Evaluation Protocols 26

D Additional Experimental Analysis **27**

D.1 Training Scale and Token Consumption 27

D.2 Cross-Modal Masking: Control, World Modeling, and Inference Efficiency 28

D.3 Qualitative Closed-Loop Rollout Analysis 29

D.4 Cross-Game Zero-Shot Transfer to VoxeLibre 36

D.5 Extended LASI Analysis 37

E Scope, Limitations, and Interpretation Boundaries **42**

E.1 Method and Evaluation Scope 42

E.2 Interpretation of the LASI Diagnostics 43

A Extended Method

The main text defines the GameWAM formulation, block-causal visibility, prediction–execution decomposition, hierarchical history, and learning objective. This section expands their computational realization under teacher-forced training and online interaction. We first clarify how the same block-causal information boundary is instantiated under the two execution settings and how realized context is organized within and across interaction cycles. We then detail the cross-cycle history update, transformer-level multimodal computation and masking variants, timestep-wise gameplay/GUI action routing, and the supervision semantics of the training objective. Concrete architectural dimensions, rollout horizons, memory capacities, and training configurations are reported separately in Section C.

A.1 Teacher-Forced Training and Online Rollout

Teacher-forced training and online rollout share the same block-causal information boundary but differ in how the clean causal prefix becomes available. During training, complete trajectories are available from the dataset. We therefore construct prediction anchors from the E-spaced execution-aligned temporal grid and provide each anchor with its corresponding ground-truth causal prefix. Future visual and action variables within the prediction horizon remain generation targets and are corrupted according to the flow objective, while observations beyond the prediction boundary remain masked. Teacher forcing therefore changes the computational realization of the clean prefix, rather than the conditional information available to the model.

This organization preserves the realized-versus-future separation used during online interaction. Although multiple prediction anchors may be constructed from the same trajectory, each anchor only receives observations that are causally available at its own prediction boundary. Ground-truth observations that occur later in the trajectory do not become additional context for earlier anchors, and predicted future variables from one anchor are never promoted into the clean prefix of another anchor. The resulting training computation therefore preserves the causal semantics of closed-loop execution while allowing supervision to be formed directly from complete trajectories.

Online rollout realizes the same dependency structure sequentially. At each decision point, GameWAM predicts a P-step action plan but commits only the first E actions. The environment is then advanced by this committed execution block, producing a new realized observation for the next decision. The model subsequently replans from the updated causal context, while the unexecuted suffix of the previous prediction is discarded. Consequently, only observations obtained through executed interaction can enter the clean prefix or persistent history.

The prediction horizon and execution horizon play different roles in this process. The block–cycle temporal organization is defined by the execution horizon E: prediction anchors, committed blocks, and interaction updates are aligned to the same E-spaced temporal grid. In contrast, the prediction horizon P determines how far into the future each anchor is supervised. For the j-th prediction anchor,

$$a_j = jE, \qquad Y_j = A[a_j + 1 : a_j + P], \tag{16}$$

where Y_j denotes the corresponding future action target. When $P > E$, neighboring prediction targets overlap by $P - E$ actions. Increasing P therefore extends the predictive supervision attached to each execution-aligned anchor without changing the underlying block or cycle boundaries.

This execution-aligned overlapping supervision increases the predictive coverage extracted from each interaction trajectory. Because the execution-aligned anchors remain fixed by E, increasing P extends the forecast range associated with each anchor rather than requiring additional execution-aligned samples. Realized transitions in overlapping regions can therefore be reused across multiple forecast offsets and supervised from different causally valid prefixes while maintaining the same decision cadence. For example, a transition that is farther in the future for one anchor may become a near-term prediction target for the next anchor. GameWAM therefore combines frequent E-spaced decision points with longer P-step predictive supervision, rather than requiring the training sequence to adopt a longer execution interval.

Because complete trajectories are available during training, these overlapping targets do not require autoregressive rollout of intermediate predicted blocks. Each prediction anchor is supervised using its own ground-truth causal prefix, avoiding the need to generate previous blocks merely to construct

later conditioning context. This distinction does not make larger prediction horizons computationally free: increasing P still introduces additional future targets and associated prediction losses. Instead, the benefit of the organization is that longer-horizon supervision and frequent execution-aligned anchors coexist without extending the autoregressive commitment chain.

During action-only online evaluation, iterative denoising of future visual variables is omitted. The realized visual prefix remains available as the context for action generation, while only the unobserved future video generation process is removed. The resulting computation preserves the same closed-loop execution semantics while removing future-video generation that is not required for immediate control.

A.2 Temporal Context Within and Across Interaction Cycles

The preceding training and rollout schedules organize temporal context at multiple timescales. The cross-cycle state H_c is already defined in the main text and is the historical representation that survives an interaction-cycle boundary. Within an active cycle, we additionally denote by $C_{c,j}$ the realized visual context available before decision j. It contains current-cycle observations that have actually been reached through executed actions. As defined in the main text, the complete clean visual prefix combines persistent cross-cycle history with the realized context accumulated within the active cycle:

$$\mathcal{P}_{c,j} = H_c \cup C_{c,j}. \tag{17}$$

This decomposition is informational rather than architectural: H_c specifies what remains accessible from earlier cycles, whereas $C_{c,j}$ preserves the fine-grained context accumulated inside the current cycle. Predicted but unexecuted future variables belong to neither term.

Under teacher forcing, $C_{c,j}$ is obtained directly from the corresponding ground-truth prefix of the sampled trajectory. During online rollout it can only grow after the environment has returned a new realized observation,

$$C^{\mathrm{inf}}_{c,j+1} = C^{\mathrm{inf}}_{c,j} \cup \{o^{\mathrm{env}}_{c,j+1}\}. \tag{18}$$

Hence a predicted future observation is not treated as history merely because it has been generated; only the observation reached after execution can extend the within-cycle context.

The online K/V cache is a computational representation of the instantiated clean prefix $\mathcal{P}_{c,j}$, rather than a third source of temporal context. At the beginning of a cycle, the persistent cross-cycle history H_c and the initial realized within-cycle context are processed together to form the clean layer-wise K/V states. As interaction continues, only newly realized observations extend the current-cycle portion of this cache. Denoting the retained clean-prefix representation by $\mathcal{Z}^{\mathrm{KV}}_{c,j}$,

$$\mathcal{Z}^{\mathrm{KV}}_{c,j} = \Psi_\theta\big(\mathcal{P}^{\mathrm{inf}}_{c,j}\big) = \Psi_\theta\big(H_c \cup C^{\mathrm{inf}}_{c,j}\big)\,, \tag{19}$$

where Ψ_θ denotes the layer-wise transformation of the clean prefix. During teacher-forced training, the analogous clean representations are formed transiently inside the parallel forward computation. During online inference, they are retained and incrementally extended as new observations are realized. At a cycle boundary the transient K/V state is discarded, while the completed interaction updates the persistent history used to construct $\mathcal{P}_{c+1,0}$ for the next cycle.

Thus, H_c is the persistent state across cycles, $C_{c,j}$ is the fine-grained realized context accumulated within the active cycle, and $\mathcal{Z}^{\mathrm{KV}}_{c,j}$ is their transient processed representation during online computation. Training and inference differ in the computational lifetime of this representation, not in the semantic source of historical information.

A.3 Cross-Cycle Hierarchical Memory

The main text defines the persistent history as $H_c = [M_c; R_c]$, where R_c retains recent executed segments and M_c summarizes older segments evicted from the recent buffer. Here we specify the temporal encoding and long-term update used to construct these two components. As in Eq. 8, the representation

$$S_c = \mathrm{Conv3D}\left(\mathrm{VAE}(O^{\mathrm{exec}}_c)\right) \tag{20}$$

is formed only from observations realized through execution. The Conv3D tokenizer reduces spatial resolution while preserving the temporal ordering within the segment.

Temporally encoded segment tokens. Both recent-history readout and long-term memory updates operate on the same compressed segment representation S_c. Let $s_{c,\tau,k}$ denote the token at temporal position τ and compressed spatial position k. Each token is augmented with embeddings for its within-segment temporal position and spatial token identity, together with a relative-time embedding. For a segment spanning raw temporal indices $[b, e]$ and a reference time t, we use

$$\eta_c(t) = \left[\log\left(1 + \frac{t-b}{T_{\text{cyc}}}\right),\ \log\left(1 + \frac{(t-e)_+}{T_{\text{cyc}}}\right),\ \frac{e-b}{T_{\text{cyc}}}\right], \tag{21}$$

where T_{cyc} is the cycle stride in the native interaction timeline. A small embedding network ψ maps this descriptor to the model dimension. The resulting token representation has the form

$$u_{c,\tau,k}(t) = s_{c,\tau,k} + e^{\text{time}}_{\tau} + e^{\text{space}}_{k} + \psi(\eta_c(t)). \tag{22}$$

Recent-history readout and long-term writing use separate role embeddings and normalization parameters on top of $u_{c,\tau,k}$, while sharing the underlying compressed segment tokens and temporal descriptor. For recent segments, t is the current interaction time, so their relative-age encoding evolves as the rollout proceeds. For a segment entering long-term memory, the reference is its overflow time from the recent buffer.

Recent buffer and overflow. The recent branch retains the latest K_R completed segments explicitly:

$$(R_{c+1}, \bar{S}_c) = \text{FIFO}_{K_R}(R_c, S_c). \tag{23}$$

No long-term write occurs while the buffer has free capacity. Once it is full, the oldest segment $\bar{S}_c$ is evicted and becomes the source of a long-term memory update. Thus, recent observations remain individually addressable until they leave the recent buffer; hierarchical summarization is triggered by overflow rather than applied to every completed cycle.

Multi-timescale long-term update. Let $M_c \in \mathbb{R}^{L \times K_M \times d}$ contain K_M persistent slots at each of L temporal levels. The evicted segment is first converted to long-memory source tokens B_c using the long-term role encoding described above. Each memory slot attends directly to the complete overflow segment:

$$\widetilde{M}_{\ell,k} = \text{Attn}\left(M_{\ell,k} + e^{\text{level}}_{\ell} + e^{\text{slot}}_{k},\ B_c,\ B_c\right). \tag{24}$$

The level and slot embeddings distinguish the roles of different persistent states, while the update rate varies explicitly across temporal levels. Let h_ℓ denote the characteristic half-life of level ℓ and let Δ denote the duration represented by the incoming segment. The corresponding update prior is

$$\alpha_\ell(\Delta) = 1 - 2^{-\Delta/h_\ell}. \tag{25}$$

Shorter-timescale levels therefore place a larger prior weight on a new segment, whereas longer-timescale levels change more conservatively.

The prior is combined with a content-dependent correction computed from the existing slot and its attention candidate:

$$\begin{aligned} g_{\ell,k} &= \text{sigmoid}\left(\text{logit}(\alpha_\ell(\Delta)) + G\Big([M_{\ell,k}; \widetilde{M}_{\ell,k}]\Big)\right), \\ M'_{\ell,k} &= \text{LN}\left(M_{\ell,k} + g_{\ell,k} \odot (\widetilde{M}_{\ell,k} - M_{\ell,k})\right), \end{aligned} \tag{26}$$

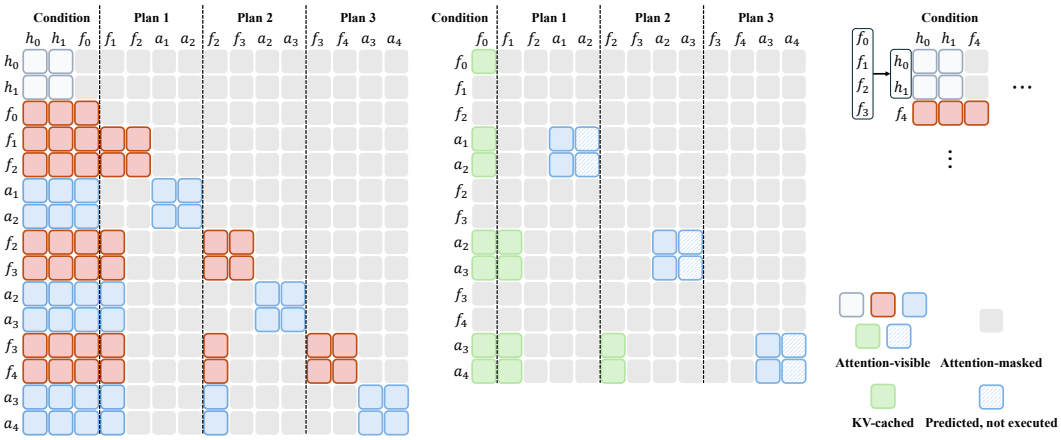
where $g_{\ell,k} \in (0,1)^d$ provides a feature-wise gated interpolation. Consequently, the temporal prior determines the nominal update rate of each level, while the learned correction allows the content of a particular overflow segment to modulate that update.

The long-term branch is itself temporally bounded. Only a fixed number of the most recent overflow segments, determined by the largest configured memory timescale, contribute to the persistent state. Writing this bounded ordered set as $\mathcal{Q}_c$, the effective memory can be viewed as applying Eq. 26 sequentially over $\mathcal{Q}_c$ in temporal order. Hence both the number of explicit recent tokens and the number of long-term memory slots remain independent of episode length.

At readout, the long-term slots are flattened and concatenated with the temporally encoded recent segments to form the persistent history supplied to the next cycle:

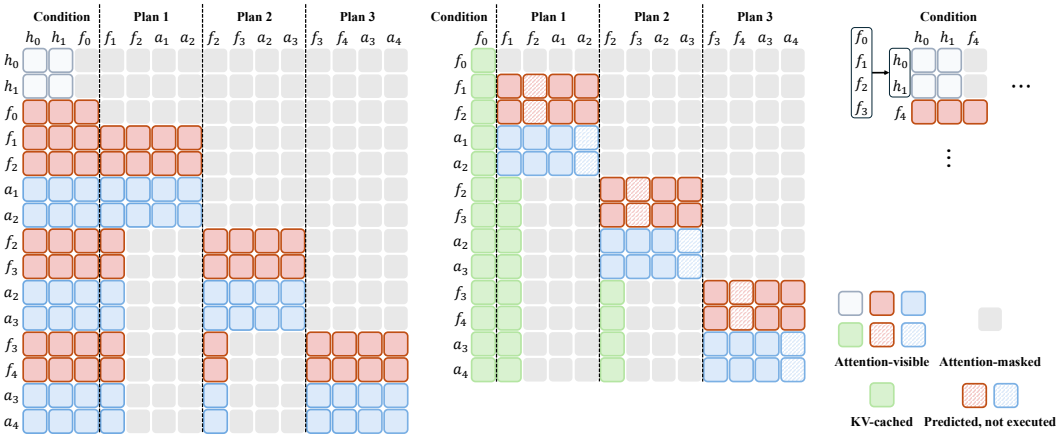
$$H_c = [M_c; R_c]. \tag{27}$$

This construction preserves explicit access to the most recent realized interaction while progressively summarizing older executed context at multiple temporal scales. Predicted but unexecuted observations never enter either branch of the persistent history.



(a) Training (b) Inference

Figure 6: **Block-causal visibility for the default modality-decoupled world–action formulation.** The training view (left) and autoregressive rollout view (right) show three overlapping plans. Visual and action predictions receive the same clean causal prefix, while simultaneously corrupted future visual and action tokens are not used as cross-modal conditions. Completed blocks are teacher-forced during training; at rollout, only observations reached through committed execution enter the clean prefix. Planned but unexecuted suffixes are discarded before the next decision. Colored cells indicate visible entries, gray cells are masked, green cells denote clean retained context, and hatched cells denote predicted but unexecuted variables.



(a) Training (b) Inference

Figure 7: **Block-causal visibility with joint current-block video–action coupling.** The joint alternative allows aligned noisy visual and action variables to exchange information inside the current planning block while retaining the same clean causal prefix and the same exclusion of later blocks. Training and rollout otherwise follow the same teacher-forcing and execution-grounded context rules as in Fig. 6.

A.4 Block-Causal Multimodal Computation

The main text introduces the modality-decoupled block-causal mask used by the reported Game-WAM model. Here we make its transformer-level realization explicit and contrast it with a joint within-block video–action alternative. For target block j, let $\mathcal{P}_{c,j}$ denote the causally available clean visual prefix, and let $\mathcal{V}_j$ and $\mathcal{A}_j$ denote the corresponding noisy video and action variables.

Instruction, proprioceptive context, and other non-prefix conditions remain available through their respective conditioning paths and are omitted below for clarity.

At the transformer level, the Video and Action DiTs retain modality-specific hidden states, prediction parameters, and flow-time modulation. Their interaction is instead determined by the key/value states made visible to each query stream. For modality $m \in \{v, a\}$ at layer ℓ, a generic update can be written as

$$\begin{aligned} \widetilde{h}^{m,\ell} &= h^{m,\ell} + G^{m,\ell}_{\text{attn}}\left(\text{Attn}\left(Q^{m,\ell}, \mathbf{K}^{m,\ell}_{c,j}, \mathbf{V}^{m,\ell}_{c,j}\right); \sigma_m\right), \\ h^{m,\ell+1} &= \widetilde{h}^{m,\ell} + G^{m,\ell}_{\text{ffn}}\left(\widetilde{h}^{m,\ell}; \sigma_m\right), \end{aligned} \tag{28}$$

where the modality-specific gates absorb the corresponding flow-time modulation, and $(\mathbf{K}^{m,\ell}_{c,j}, \mathbf{V}^{m,\ell}_{c,j})$ denotes the layer-ℓ attention context assembled according to the block-causal visibility rule.

For the default modality-decoupled mask,

$$\text{Vis}(\mathcal{A}_j) = \mathcal{P}_{c,j} \cup \mathcal{A}_j, \qquad \text{Vis}(\mathcal{V}_j) = \mathcal{P}_{c,j} \cup \mathcal{V}_j. \tag{29}$$

Importantly, this decoupling applies to the simultaneously corrupted *future* variables; it does not separate action generation from the learned representation of the realized visual world.

Concretely, at each transformer layer the complete clean visual prefix $\mathcal{P}_{c,j}$ is processed by the Video DiT to produce layer-wise clean-prefix key/value states $(\mathbf{K}^{p,\ell}_{c,j}, \mathbf{V}^{p,\ell}_{c,j})$. The same states are reused by both prediction branches. Let $(\mathbf{K}^{v,\ell}_{j}, \mathbf{V}^{v,\ell}_{j})$ and $(\mathbf{K}^{a,\ell}_{j}, \mathbf{V}^{a,\ell}_{j})$ denote the key/value states of the current noisy video and action variables, respectively. Omitting the additional conditioning paths above, the default attention contexts have the schematic form

$$\begin{aligned} \mathbf{K}^{v,\ell}_{c,j} &= \left[\mathbf{K}^{p,\ell}_{c,j}; \mathbf{K}^{v,\ell}_{j}\right], \quad \mathbf{V}^{v,\ell}_{c,j} = \left[\mathbf{V}^{p,\ell}_{c,j}; \mathbf{V}^{v,\ell}_{j}\right], \\ \mathbf{K}^{a,\ell}_{c,j} &= \left[\mathbf{K}^{p,\ell}_{c,j}; \mathbf{K}^{a,\ell}_{j}\right], \quad \mathbf{V}^{a,\ell}_{c,j} = \left[\mathbf{V}^{p,\ell}_{c,j}; \mathbf{V}^{a,\ell}_{j}\right]. \end{aligned} \tag{30}$$

Thus, video queries attend to the realized clean prefix and the current noisy video states, whereas action queries attend to the same realized clean prefix and the current noisy action states. The action branch does not read current-block noisy-video K/V states, and the video branch does not read current-block noisy-action K/V states. The two branches therefore remain coupled through a shared learned representation of the realized clean prefix, rather than through direct exchange between simultaneously denoised future variables.

This distinction is also important for understanding the role of video supervision. The clean-context K/V states are produced by the Video DiT and remain in the training computation graph when they are consumed by both branches. Consequently, gradients from both the video and action objectives propagate through the Video DiT path that constructs this shared context. Video prediction can therefore shape the realized-world representation used by action generation even though the action prediction does not condition on a denoised future video. In this sense, the default formulation couples world and action learning through the shared representation of realized dynamics, while deliberately decoupling their current noisy future variables.

Fig. 6 visualizes this computation under teacher-forced training and autoregressive rollout. During training, completed blocks provide clean causal context, but future targets that are not yet causally available remain excluded from the corresponding clean prefix. During online interaction, only observations reached after committed environment actions are promoted into the realized context; predicted but unexecuted suffixes never become clean conditioning states.

The same dependency structure is retained during action-only online inference. The instantiated clean prefix $\mathcal{P}_{c,j}$ is processed by the Video DiT to construct layer-wise clean-prefix K/V states, which are cached and consumed by Action DiT denoising. As new observations are obtained through execution, only the newly realized within-cycle context extends this cache. What is omitted during action-only inference is the iterative denoising of *future* video variables, not the Video DiT encoding of the realized clean prefix. The default mask therefore preserves the learned visual-context pathway used for action generation while avoiding the additional online cost of future-video denoising.

For the joint within-block alternative,

$$\text{Vis}(\mathcal{A}_j) = \text{Vis}(\mathcal{V}_j) = \mathcal{P}_{c,j} \cup \mathcal{V}_j \cup \mathcal{A}_j. \tag{31}$$

The corresponding attention contexts contain both current noisy modalities:

$$\begin{aligned}\mathbf{K}^{v,\ell}_{c,j} = \mathbf{K}^{a,\ell}_{c,j} &= \left[\mathbf{K}^{p,\ell}_{c,j}; \mathbf{K}^{v,\ell}_{j}; \mathbf{K}^{a,\ell}_{j}\right], \\ \mathbf{V}^{v,\ell}_{c,j} = \mathbf{V}^{a,\ell}_{c,j} &= \left[\mathbf{V}^{p,\ell}_{c,j}; \mathbf{V}^{v,\ell}_{j}; \mathbf{V}^{a,\ell}_{j}\right].\end{aligned} \tag{32}$$

The joint formulation therefore introduces direct bidirectional interaction between the current noisy future-video and action variables during denoising. The temporal causal boundary is otherwise unchanged: both formulations use the same realized clean prefix and exclude information from later blocks.

This additional within-block coupling also changes online computation. Because the action queries depend on the simultaneously denoised future-video states, joint action generation requires the future visual stream to be denoised alongside the action stream. By contrast, the default modality-decoupled model retains the shared realized-context pathway while allowing future-video denoising to be omitted during action-only deployment. We compare the two formulations under matched training and evaluation settings in Section D.2.

A.5 Interaction-Conditioned Native Action Generation

Minecraft gameplay and GUI interaction use the same physical keyboard–mouse interface but induce substantially different conditional control distributions. GameWAM therefore retains a single native action representation while specializing action prediction by interaction mode. The specialization is applied independently at each native action timestep rather than once to an entire prediction horizon, allowing a generated trajectory to transition between gameplay and GUI interaction without switching models or invoking a separate high-level controller.

The Action DiT produces gameplay- and GUI-specific flow predictions together with a timestep-wise routing prediction. During training, the observed interaction-mode label selects which action branch is supervised at each timestep and simultaneously provides the target for the routing loss. During rollout, the predicted route instead selects the branch used to generate the complete native action vector at that timestep. Routing therefore chooses between interaction-specific conditional predictions while preserving a shared action dimensionality and a single low-level control interface.

This distinction is particularly important for continuous control. The same continuous coordinates represent camera motion during gameplay and cursor motion during GUI interaction, but their numerical distributions differ substantially. We therefore normalize continuous action coordinates with interaction-specific statistics before flow modeling and invert the corresponding normalization after generation. Binary keyboard and mouse coordinates retain their shared normalization, while validity masks exclude coordinates unavailable for a particular environment or interaction state. Environment-specific thresholding, hotbar constraints, and coordinate definitions are given in Section B.2.

The router changes the conditional distribution used to predict an action; it does not decompose the controller into independent gameplay and GUI policies. Both branches operate on the same Action DiT representation and predict the same native action vector, and the route can change from one action timestep to the next. Consequently, heterogeneous interaction modes are handled within a single world–action trajectory while retaining mode-appropriate continuous control statistics.

A.6 Supervision Semantics Beyond the Compact Objective

The losses in Section 4.3 already define the optimized quantities. Three details clarify how those terms are applied. First, valid continuous and binary action coordinates are reduced separately before being combined. This normalizes heterogeneous action coordinates but does not imply independent action policies; both groups belong to the same generated native action trajectory. Validity masks exclude padding and coordinates unavailable in a particular environment or interaction state.

Second, teacher forcing applies clean context only to blocks that are causally complete at the prediction boundary. The current block's future visual and action variables remain generation targets and are corrupted independently, while completed visual blocks provide clean conditioning. This preserves the realized-versus-uncertain partition used by online rollout.

Table 3: **Minecraft action interface.** The first two coordinates are continuous camera deltas; the remaining coordinates are binary keyboard or mouse controls.

Index	Control	Type	Index	Control	Type
0	Camera pitch	continuous	11	Drop item	binary
1	Camera yaw	continuous	12	Open inventory	binary
2	Move forward	binary	13	Hotbar slot 1	binary
3	Move backward	binary	14	Hotbar slot 2	binary
4	Move left	binary	15	Hotbar slot 3	binary
5	Move right	binary	16	Hotbar slot 4	binary
6	Jump	binary	17	Hotbar slot 5	binary
7	Sneak	binary	18	Hotbar slot 6	binary
8	Sprint	binary	19	Hotbar slot 7	binary
9	Attack	binary	20	Hotbar slot 8	binary
10	Use/interact	binary	21	Hotbar slot 9	binary

Table 4: **ViZDoom action interface used in our experiments.** A unified 9-D representation covers both map families. Battle maps use continuous horizontal turning, whereas the two defense maps use binary left/right turning.

Index	Control	Type	Used in
0	Fire	binary	all maps
1	Move forward	binary	Battle maps
2	Move backward	binary	Battle maps
3	Strafe left	binary	Battle maps
4	Strafe right	binary	Battle maps
5	Speed modifier	binary	Battle maps
6	Turn left	binary	defense maps
7	Turn right	binary	defense maps
8	Horizontal turn delta	continuous	Battle maps

Finally, the predictive-history objective in Eq. 13 operates only on the bounded cross-cycle history. Its pooled representation predicts the clean visual feature at the current observation boundary, which is treated as a fixed target. This auxiliary constraint discourages information loss during history compression and encourages H_c to retain visual information predictive across cycle boundaries, without changing the main world–action flow objective. Numerical weights are given in Section C.

B Data and Training Construction

B.1 Trajectory Alignment

All data sources are converted to a common observation–state–action timeline. The state paired with an action is the state observed immediately before that action is executed. Binary controls, continuous camera/cursor values, padding indicators, and validity masks for unavailable state or action fields are retained before normalization. The same pre-action convention determines whether a Minecraft step belongs to gameplay or GUI interaction.

For VPT, video observations are synchronized with the corresponding per-frame mouse, keyboard, and Minecraft state records before conversion to the shared trajectory representation. A null-action filtering stage removes redundant null controls together with their paired frames while preserving alignment between retained observations, state, actions, and event metadata. The resulting episodes store aligned action and state records, episode-aligned visual observations, and the metadata required for training.

B.2 Native Action Interfaces

The main text defines GameWAM over hybrid environment actions. Tables 3 and 4 summarize the action coordinates used in our experiments.

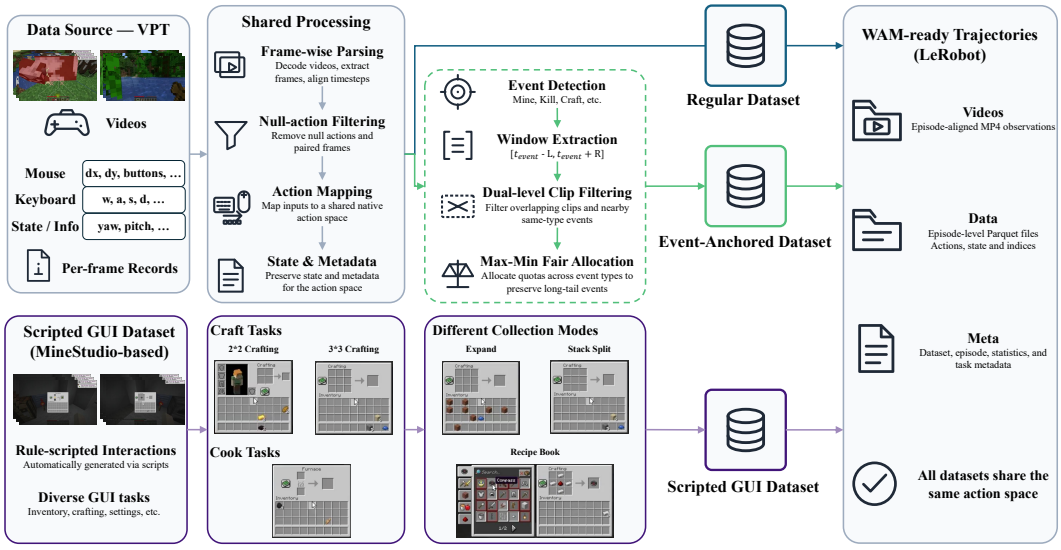


Figure 8: **Construction of WAM-ready game trajectories.** VPT recordings are standardized into regular and Event-Anchored datasets; the event branch detects interaction events, extracts anchored windows, filters temporally redundant candidates, and applies max–min fair allocation across event types. MineStudio-based scripted GUI trajectories broaden interface-interaction coverage. All streams share the same native action space and trajectory representation for joint world–action training.

The camera coordinates are normalized continuously. Binary controls are thresholded after generation, and the hotbar coordinates are constrained to select at most one slot. Gameplay and GUI interaction share the same action layout but use separate normalization statistics and mode-specific decoding. The Minecraft state input contains view orientation, cursor location, selected hotbar slot, and GUI state; unavailable fields are explicitly masked.

For Battle maps, the continuous turn coordinate is normalized to $[-1, 1]$, corresponding to a horizontal rotation of up to 10° per interaction step. No separate continuous pitch control is used in the reported ViZDoom experiments. The proprioceptive input contains planar position, heading, and health, with unavailable fields treated as missing rather than observed zeros.

B.3 Training Data Sources

Minecraft. Minecraft training uses three complementary streams. Regular VPT trajectories (Baker et al., 2022) provide broad naturalistic interaction coverage and long temporal context. From the same source recordings, we construct an Event-Anchored VPT dataset around identifiable MineStudio-style interaction events (Cai et al., 2024), providing instruction-conditioned supervision around behaviorally meaningful transitions. MineStudio-based scripted GUI trajectories form the third stream and broaden interface-interaction coverage. All three streams follow the same observation–state–action alignment and native action representation.

The two VPT streams differ at the dataset level. Regular VPT preserves the standardized long trajectories, whereas the Event-Anchored stream applies the offline event-selection procedure described below to construct anchored sub-trajectories. Event-anchored clip sampling is subsequently applied during training and is therefore distinct from construction of the Event-Anchored Dataset.

ViZDoom. We collect expert trajectories using Asynchronous Proximal Policy Optimization (APPO) (Petrenko et al., 2020) on the two Battle maps and the Defend the Center and Defend the Line maps. The action actually executed at each recorded interaction step is preserved and expressed in the unified 9-D representation of Table 4. Continuous horizontal turning is retained on the

Battle maps rather than discretized. ViZDoom does not require a separate gameplay/GUI interaction mode.

B.4 Event-Anchored VPT Dataset Construction

The Event-Anchored VPT dataset is constructed offline from the same source recordings as Regular VPT. Its construction proceeds through event detection, anchored window extraction, temporal redundancy filtering, and allocation across event types. The concrete construction parameters used for the reported dataset are provided with the sampling configuration in Section C.3.

Event detection. We identify MineStudio-style interaction events from changes in the recorded Minecraft state statistics. The retained event set covers identifiable interactions such as mining, entity interaction, crafting, and related state transitions. Each detected event is associated with its temporal location and corresponding instruction metadata, providing an anchor aligned with the original visual and native-action trajectory.

Anchored window extraction. For an event occurring at temporal index t_i, we extract a surrounding trajectory window

$$W_i = X[t_i - L : t_i + R], \tag{33}$$

where X denotes the aligned observation–state–action trajectory and L and R specify the retained context around the event. The event position is preserved within the extracted sub-trajectory as anchor metadata used by the subsequent filtering and training-sampling stages.

Dual-level clip filtering. Event detection can yield temporally redundant candidates, either because different detected events produce strongly overlapping windows or because multiple occurrences of the same event type appear in close succession. We therefore apply two temporal filtering criteria: one limits overlap among candidate clips, while the other suppresses nearby repetitions of the same event type. This reduces redundant supervision before event-type allocation while retaining distinct interactions and their surrounding context.

Max–min fair allocation. The filtered candidate pool remains imbalanced because event types occur with substantially different frequencies in natural trajectories. We therefore allocate the retained dataset budget across event types using max–min fairness. Event types with fewer available candidates saturate at their available count, while the remaining capacity is redistributed among event types with larger candidate pools. This limits domination by highly frequent events while preserving long-tail interactions whenever sufficient candidates are available.

The retained windows, together with their event anchors, instruction metadata, aligned observations, state, and native actions, constitute the Event-Anchored VPT dataset.

B.5 Scripted GUI Trajectory Construction

The scripted GUI stream is generated in MineStudio through rule-scripted native keyboard–mouse interaction. The collection covers diverse interface procedures and uses multiple collection modes to vary the interaction trajectories. The scripts are used only to generate demonstrations: the resulting training data contains rendered observations, proprioceptive state, and native actions in the same representation used by the VPT streams.

The collected episodes undergo the same frame-wise parsing, state preservation, action mapping, and trajectory alignment as the other Minecraft data sources. Consequently, scripted GUI data broadens interface-interaction coverage without introducing a separate action representation for training.

B.6 Event-Anchored Training Clip Sampling

Event-Anchored Dataset construction determines which anchored sub-trajectories are retained, whereas event-anchored clip sampling separately determines the temporal density of training windows drawn from them.

Long interaction trajectories contain highly uneven supervision density. Transitions near an annotated event often contain the visual changes and actions most directly related to the corresponding instruction, whereas more distant portions may consist largely of traversal, incidental camera motion, or other weakly related behavior. Uniform window sampling can therefore devote substantial training capacity to comparatively less informative regions.

For the Event-Anchored VPT stream, we increase the density of candidate training windows whose observation span overlaps the neighborhood of the stored event anchor, while sampling the remaining regions more sparsely. This increases exposure to event-relevant state transitions while retaining surrounding interaction context. Regular VPT and scripted GUI trajectories use fixed sampling intervals rather than anchor-dependent sampling. The mixture weights, anchor neighborhood, and clip-sampling intervals used in the reported training run are given in Section C.3.

C Implementation and Evaluation Protocol

C.1 Model Architecture and Initialization

The visual predictor is initialized from Wan2.2-TI2V-5B. The Action DiT follows the same 30-layer, 24-head transformer layout but uses a reduced hidden width of 1024, compared with 3072 for the Video DiT. To retain the pretrained video-transformer prior despite this width reduction, we adapt the Wan2.2 Video DiT weights to initialize the Action DiT backbone, while action-specific input/output projections, gameplay/GUI prediction heads, and the interaction-mode router are initialized separately and learned during GameWAM training. Diffusion-time modulation is applied independently to the two streams.

The video VAE is kept frozen, and text-conditioning features are produced by a fixed text encoder. The Video DiT, Action DiT (including the interaction-specific action prediction heads and router), hierarchical history module, and proprioceptive encoder are optimized during GameWAM training.

C.2 Observation, Planning, and Temporal-Context Configuration

Both environments use 224×224 RGB observations. Visual observations are sampled once every two native environment actions. The prediction horizon is $P = 16$ native actions and the execution horizon is $E = 8$, giving an eight-action overlap between neighboring predictions. Three committed execution blocks form one 24-action cycle for the hierarchical temporal context.

For each completed cycle, the cross-cycle representation preserves four latent time positions and projects each position to four spatial tokens, giving 16 tokens per cycle-level segment. The two most recent completed segments are kept explicitly. Older context is summarized at two temporal scales with four memory tokens per scale; the corresponding scale half-lives are two and four completed cycles. The resulting cross-cycle history contains at most 40 tokens. During training, the available cross-cycle history is shortened with probability 0.1 by retaining a randomly selected valid suffix, including the possibility of no cross-cycle history.

The environment-specific action interfaces are given in Tables 3 and 4. Minecraft uses timestep-wise interaction conditioning for gameplay and GUI behavior, whereas the evaluated ViZDoom tasks use a single interaction regime.

C.3 Training Data Mixture and Clip-Sampling Configuration

Event-Anchored VPT construction. The reported Event-Anchored VPT dataset uses 96-frame sub-trajectories. For an event at frame t, the extraction window ends eight frames after the event, giving the interval $[t - 87, t + 8]$ and retaining both substantial pre-event context and a short post-event continuation. Null-action filtering is applied before event-window construction while preserving detected event anchors. During dual-level filtering, retained clip starts are separated by at least 64 frames, and anchors of the same event type are separated by more than 96 frames. The filtered candidates from the VPT partitions are pooled and subject to a global max–min fair allocation with a total budget of 200,000 Event-Anchored sub-trajectories. Candidate selection is shuffled with random seed 42. VPT data are processed at their native 20 FPS.

Training mixture and temporal sampling. The reported Minecraft training run uses sample-mixture weights of 80% for Event-Anchored VPT, 5% for Regular VPT, and 15% for scripted GUI trajectories. These weights specify the training-sample distribution rather than the raw corpus composition.

The three streams use different temporal intervals for constructing training clips. For Event-Anchored VPT, candidate windows whose observation span intersects a radius-8 neighborhood of the stored event anchor are sampled with stride 2, while the remaining regions use stride 16. Regular VPT uses stride 32 throughout the trajectory, whereas scripted GUI trajectories use stride 8. Thus, only the Event-Anchored stream varies its training-window density according to temporal proximity to an interaction event.

C.4 Optimization and Objective Weighting

GameWAM is trained for two epochs using fused AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01, and gradient clipping at a maximum norm of 1.0. The learning rate is linearly warmed up over the first 5% of optimization, corresponding to 1,095 steps, to a peak value of 4×10^{-5}, and then follows cosine decay to a minimum of 4×10^{-7}. Each epoch contains 10,950 optimizer steps, for 21,900 steps in total.

Training uses BF16 precision and DeepSpeed ZeRO-2 on eight NVIDIA H200 GPUs. The per-GPU batch size is 44 with gradient accumulation of one, giving an effective global batch size of 352. The full two-epoch training run takes approximately 22 hours.

For the objective in Eq. 14, the reported model uses $\lambda_v = 1.0$, $\lambda_a = 1.0$, $\lambda_m = 0.05$, and $\lambda_h = 0.5$. Within the grouped action objective in Eq. 11, continuous and discrete action losses are weighted equally with $\lambda_{\text{cont}} = \lambda_{\text{disc}} = 1.0$.

C.5 Flow-Training and Sampling Configuration

The visual and action flow times are sampled independently. For $u \sim \mathcal{U}[0, 1]$, the reported training runs use the shifted time mapping

$$\sigma = \frac{5u}{1 + 4u}. \tag{34}$$

The corresponding flow residuals are weighted by

$$w(\sigma) = \frac{\exp[-2(\sigma - \frac{1}{2})^2] - \exp(-\frac{1}{2})}{Z}, \tag{35}$$

where Z normalizes the mean weight under the sampling schedule. At inference, sampling starts from Gaussian sources and follows the learned vector field from $\sigma = 1$ to $\sigma = 0$ with first-order integration. Standard closed-loop evaluation uses ten denoising steps.

C.6 Closed-Loop Evaluation Protocols

All reported task-performance evaluations use the corresponding final checkpoint after the second training epoch, without validation- or MCU-based checkpoint selection. Unless otherwise stated, the action source is independently resampled at each replanning step rather than reused across an episode. This mitigates coherent episode-level accumulation from repeated reuse of a single source; the corresponding LASI failure mode and source-resampling analysis are detailed in Section D.5.

Minecraft MCU. We use the MCU protocol reported in the main paper, including the Mini subset of 30 tasks (10 mining, 10 crafting, and 10 combat tasks). Success is determined by the environment's task-completion signals. Interaction steps count native environment actions actually executed during closed-loop rollout, excluding internal denoising evaluations. The reported step metric is averaged over successful episodes within each task category and is undefined when no successful episode is observed. The reported task-wise standard deviation is computed across per-task success rates. Except for the reproduced Fast-WAM comparison, the remaining systems in Table 1 differ substantially in training data, task-specific supervision, model class, action representation, and available inputs. Their results should therefore be interpreted primarily as a system-level benchmark comparison rather than a controlled comparison of architecture or data efficiency.

Table 5: **Model scale and reported training-token consumption.** Token categories follow the accounting reported by each work. For two-stage methods, modality-level counts are shown as Stage 1 / Stage 2. A dash indicates that the corresponding count is not separately reported or is not applicable.

	GameWAM	**OpenHA**	**Game-TARS**
Model initialization and scale			
Backbone(s)	Video: Wan2.2-TI2V-5B Action: Wan2.2-initialized Action DiT	Qwen2-VL-7B	Qwen2.5-VL-7B-Instruct
Backbone scale / params.	Video: 5B Action DiT: 1B	7B	7B
Training-token exposure			
Stage 1	*Single stage*	3.40B	526B
Stage 2		0.22B	40B
Video tokens	2.27B	–	–
Image tokens	–	–	208B / 10B
Action tokens	0.37B	–	–
Text tokens	0.15B	–	326B / 29B
Total consumed tokens	**2.79B**	**3.62B**	**566B**

ViZDoom. We use the four-map protocol reported in the main paper and measure average episode reward. The controller uses the action interface in Table 4 and the same $P = 16$, $E = 8$ overlapping-plan geometry as Minecraft.

Online execution frequency. For the inference-efficiency comparison in Section D.2 and Table 6, online execution frequency is evaluated separately from task-performance evaluation on a single NVIDIA H200 GPU under the standard inference configuration. For each model variant, we report the mean frequency over 10 closed-loop episodes. Timing includes all model-side computation performed during online replanning and excludes environment simulation and execution time.

D Additional Experimental Analysis

D.1 Training Scale and Token Consumption

Table 5 places the training scale of GameWAM in context with OpenHA (Wang et al., 2026b) and Game-TARS (Wang et al., 2025). We retain the token categories reported by each method rather than forcing heterogeneous training pipelines into a shared modality definition. GameWAM uses a single training stage, whereas OpenHA and the reported Game-TARS training recipe use two stages.

For GameWAM, the reported counts follow the exact two-epoch sampling schedule rather than an estimate based on average sequence length. Video and action tokens count target-token presentations participating in the training objective, including overlapping autoregressive planning targets. Text tokens count non-padding tokenizer tokens under the same sampling schedule; zero-padded text slots are excluded. Across the two epochs, GameWAM consumes 2.27B video-target tokens, 0.37B action-target tokens, and 0.15B effective text-conditioning tokens, for 2.79B total consumed tokens.

The exact schedule presents 7,708,800 training samples over the two epochs. The 150,723,927 non-padding text tokens therefore correspond to an average of 19.55 effective text tokens per presented sample. The implementation uses a fixed 128-slot text context, corresponding to 986,726,400 text slots over the same schedule, but most of the additional positions are zero padding. We therefore use non-padding tokenizer tokens for the reported text-token consumption rather than counting padded computational slots.

For OpenHA, Stage 1 and Stage 2 correspond to its mixed-action pre-training and CoA fine-tuning stages, respectively. The reported consumed-token counts are 3.40B and 0.22B, giving 3.62B in total. A modality decomposition directly comparable to the video, action, and text accounting used for GameWAM is not separately reported, so the corresponding entries are left unspecified.

Table 6: **Cross-modal masking, closed-loop performance, and online inference efficiency.** We report MCU Mini and MCU All ASR together with executed environment steps averaged over successful episodes under the closed-loop protocols in Section C.6, and separately report online execution frequency.

	Embodied			GUI			Combat			Avg		Exec. Freq.
Mask	Steps ↓	Mini ↑	All ↑	Steps ↓	Mini ↑	All ↑	Steps ↓	Mini ↑	All ↑	Mini ↑	All ↑	(Hz) ↑
Modality-decoupled (default)	**138**	$\mathbf{70.0}^{\pm\mathbf{32.2}}$	$\mathbf{47.5}^{\pm\mathbf{36.0}}$	155	$43.0^{\pm 32.9}$	$\mathbf{60.0}^{\pm\mathbf{38.6}}$	203	$\mathbf{39.0}^{\pm\mathbf{25.9}}$	$\mathbf{32.2}^{\pm\mathbf{30.2}}$	**50.7**	**46.6**	**12.51**
Joint video–action	141	$\mathbf{70.0}^{\pm\mathbf{24.1}}$	$40.3^{\pm 35.5}$	**145**	$\mathbf{45.0}^{\pm\mathbf{34.1}}$	$58.8^{\pm 38.5}$	**175**	$24.0^{\pm 25.0}$	$19.6^{\pm 24.7}$	46.3	39.6	8.12

For Game-TARS, we reproduce the stage-wise quantities in its reported training recipe without redefining their token accounting. Pre-training reports 526B total tokens, including 208B image tokens and 326B text tokens, while post-training reports 40B total tokens, with 10B image tokens and 29B text tokens. We retain these values exactly as reported rather than reconstructing the totals from the modality columns. Our experimental comparison uses the Game-TARS-Dense model, whose reported initialization is Qwen2.5-VL-7B-Instruct; because a separate exact stage-wise token accounting for this Dense variant is not provided, the Game-TARS token entries in Table 5 should be interpreted as the training scale reported for the Game-TARS model family.

At the reported stage level, OpenHA has approximately $1.30\times$ the total token exposure of GameWAM, while the paper-reported Game-TARS recipe exceeds the GameWAM total by more than $200\times$. These quantities provide context for training-data scale rather than a direct measure of computational cost. The models differ in tokenizer design, modality representation, sequence construction, objectives, and architecture, so token exposure should not be interpreted as FLOP-equivalent training compute.

D.2 Cross-Modal Masking: Control, World Modeling, and Inference Efficiency

Section A.4 defines and visualizes the default modality-decoupled mask and the joint within-block alternative in Figs. 6 and 7. We compare the two under otherwise matched model and evaluation settings. This comparison examines whether direct interaction between noisy video and action variables provides a control benefit sufficient to justify the additional computation required by joint video–action denoising.

Table 6 shows that the modality-decoupled design provides stronger overall closed-loop performance. The two variants match on Embodied Mini, while the joint model is slightly higher on GUI Mini. On the broader MCU All evaluation, however, the default model performs better in all three categories, with the largest difference on Combat. Overall ASR increases from 46.3 to 50.7 on MCU Mini and from 39.6 to 46.6 on MCU All with the modality-decoupled mask. The larger difference on MCU All suggests that the advantage of modality decoupling becomes more apparent across the broader task set rather than being confined to the Mini subset.

The step results provide a complementary view of interaction cost. The modality-decoupled model requires fewer steps on Embodied tasks, whereas the joint model reports fewer steps on GUI and Combat. These values should be read together with task success: most notably, the lower Combat step count of the joint model coincides with a substantial reduction in both Mini and All ASR. The step results therefore do not change the overall pattern that the default mask provides substantially stronger Combat control.

The computational advantage of modality decoupling is more pronounced. Joint video–action generation operates at 8.12 Hz under the reported inference configuration, whereas the modality-decoupled model reaches 12.51 Hz, corresponding to approximately a $1.54\times$ higher online execution frequency. This gain follows directly from removing iterative future-video denoising from the online action-generation path: video prediction remains available as a training signal, while deployment does not require the future visual stream to be jointly denoised with the actions.

Interestingly, the optimization dynamics do not simply mirror the downstream control results. During training, the joint model fits the action objective faster, while optimization of the video branch progresses more slowly. Qualitative inspection of validation predictions reveals the same asymmetry: future-video predictions from the joint model are generally less sharp than those produced by

the modality-decoupled model. Faster fitting of the action objective therefore does not translate into stronger overall closed-loop control or better future-video prediction.

The magnitude of this difference also appears to depend on the visual dynamics of the interaction regime. GUI trajectories typically contain comparatively stable viewpoints and relatively limited frame-to-frame visual change, making their future observations easier to predict than gameplay sequences involving ego-motion, moving entities, and rapid scene transitions. Consistent with this distinction, the joint variant shows little degradation on GUI: its GUI Mini ASR is slightly higher than that of the default model (45.0 versus 43.0), while GUI All remains close (58.8 versus 60.0). This may partly explain why joint coupling does not produce the substantial GUI degradation observed in more visually dynamic settings.

The difference is considerably larger on Combat, where rapid camera motion, moving targets, and abrupt changes in scene content make the visual future more difficult to predict. We likewise observe qualitatively that prediction blur becomes more pronounced in fast-motion gameplay sequences, particularly for the joint model. Together with the weaker video-side optimization, this suggests that the consequences of cross-modal coupling may become more significant as the visual prediction problem becomes harder.

One possible interpretation is asymmetric optimization interference between action fitting and visual-dynamics learning. Allowing noisy future-video and action variables to interact directly may make the action objective easier to fit while simultaneously weakening optimization of the visual predictive process, particularly when the visual future is intrinsically difficult to model. This interpretation is consistent with the faster action-loss fitting, slower video-side optimization, blurrier validation predictions, and larger performance degradation in visually dynamic gameplay. We nevertheless treat this as a hypothesis rather than a mechanistic conclusion, since the present experiments do not directly isolate gradient interference or identify where such competition arises inside the model.

This observation may also have implications beyond games. Future visual prediction in physical embodied environments can be challenging under ego-motion, clutter, partial observability, sensor variation, stochastic dynamics, and other sources of visual uncertainty. Strongly coupling action learning to an already difficult visual prediction process may therefore not always be beneficial, and optimization pressure from action learning could potentially interfere with learning useful visual dynamics. The modality-decoupled formulation provides an alternative in which video prediction continues to supply dynamics-aware training supervision without making online action generation directly depend on uncertain future-video variables. Whether a similar optimization asymmetry arises in physical embodied systems remains an important direction for future investigation.

D.3 Qualitative Closed-Loop Rollout Analysis

Aggregate task metrics summarize whether an interaction ultimately succeeds, but provide limited information about how the controller reaches that outcome. We therefore visualize representative closed-loop rollouts from both Minecraft and ViZDoom. These examples are intended as qualitative diagnostics rather than additional benchmark results. They expose behaviors that are compressed by aggregate success rates or episode rewards, including target search, sustained low-level control, persistent tracking, precise GUI manipulation, recovery from intermediate errors, survival-oriented navigation, and reactive combat.

All examples are generated by the same low-level GameWAM policy used in the corresponding quantitative evaluations. The model receives the task instruction and repeatedly replans from newly realized observations; the displayed trajectories therefore reflect closed-loop execution rather than a pre-specified open-loop action sequence.

D.3.1 Minecraft Closed-Loop Rollouts

Local exploration for target acquisition. Fig. 9 visualizes the task *"Mine the white bed."* The target object is not initially visible in the agent's immediate observation. To complete the task, GameWAM first explores the surrounding local environment through native movement and viewpoint adjustment. The agent subsequently enters a nearby structure, where the white bed becomes visible, and then performs the required interaction. This rollout demonstrates that GameWAM can

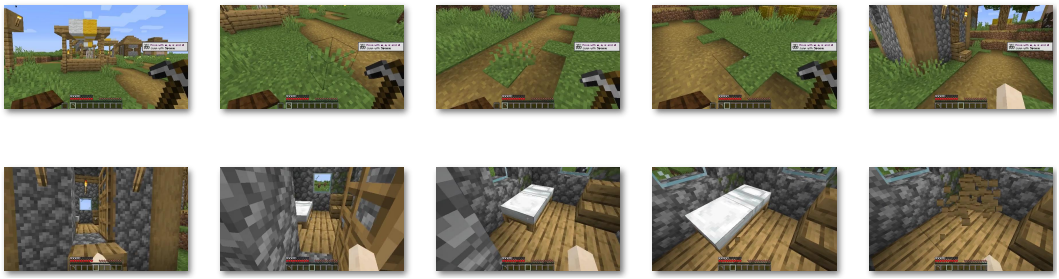


Figure 9: **Closed-loop rollout for *"Mine the white bed."*** The white bed is not visible in the initial observation. GameWAM explores the nearby environment, enters a structure, acquires visual access to the target, and completes the mining interaction. The sequence illustrates visually guided local exploration and target acquisition under closed-loop interaction.

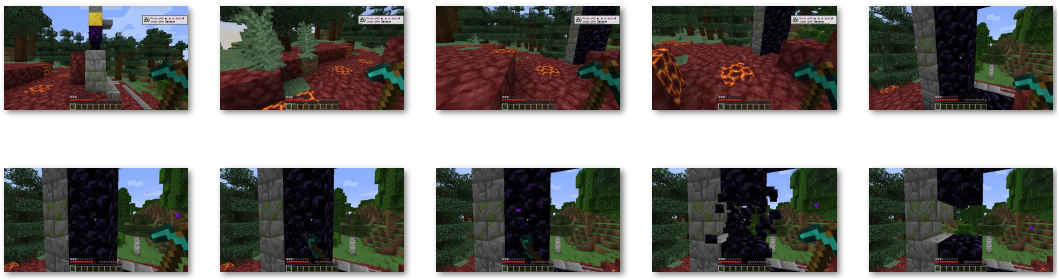


Figure 10: **Closed-loop rollout for *"Collect obsidian for crafting."*** Obtaining obsidian requires a sustained mining interaction rather than a brief action. After aligning with the target block, GameWAM maintains the mining behavior across successive replanning steps until the block breaks, illustrating temporal consistency of low-level control.

acquire targets that are not directly presented in the initial view by continuously coupling action selection with newly received visual feedback.

We interpret this behavior as a form of closed-loop local exploration and target acquisition, rather than evidence of explicit search algorithms or symbolic planning. GameWAM does not maintain an object-level map or a predefined exploration policy; instead, exploratory behavior emerges from repeated native-action prediction conditioned on evolving visual observations.

Maintaining a continuous interaction over an extended duration. Fig. 10 shows *"Collect obsidian for crafting."* Mining obsidian differs from a short attack or click interaction because the mining action must remain consistently applied for a comparatively long interval before the block breaks. A controller that frequently changes its action, loses alignment with the target, or prematurely interrupts the interaction will fail to obtain the block.

GameWAM first approaches and aligns with the obsidian and then maintains the required mining behavior until the block is successfully broken. This rollout therefore highlights temporal consistency at the native-action level: repeated replanning does not necessarily fragment a behavior that must remain stable over many consecutive environment steps.

Long-horizon pursuit of a resilient moving target. Fig. 11 shows the task *"Hunt a horse."* We choose this example because horses combine two challenging properties for low-level control. Compared with common low-health animals such as cows, sheep, and chickens, horses require substantially more successful attacks before task completion. Meanwhile, the target continuously changes its relative position during the encounter. These two factors combine to create a longer interac-



Figure 11: **Closed-loop rollout for *"Hunt a horse."*** The horse requires repeated successful attacks before task completion and changes its relative position throughout the interaction. GameWAM repeatedly reacquires the target, adjusts movement and viewpoint, and continues the interaction until completion. The sequence highlights persistent closed-loop control over an extended moving-target interaction, where consistent task intent must be maintained across multiple observation–action cycles.

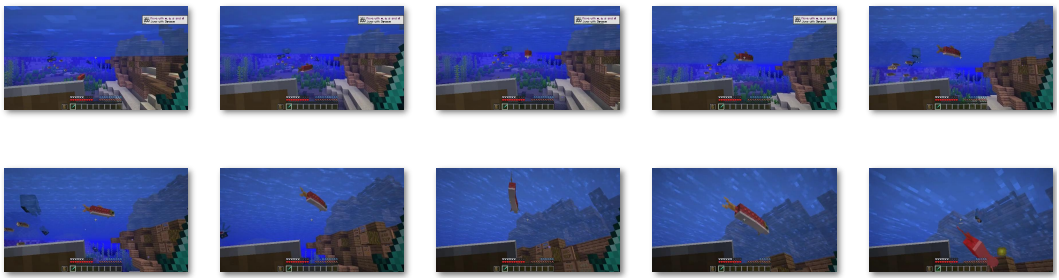


Figure 12: **Closed-loop rollout for *"Kill the salmon."*** GameWAM tracks and attacks a moving target in an underwater environment, where both visual appearance and locomotion differ from ordinary land-based gameplay. The rollout demonstrates successful closed-loop control under a qualitatively different gameplay condition.

tion horizon, requiring the controller to repeatedly update camera orientation, movement, and attack behavior based on newly observed visual feedback rather than relying on a short reactive sequence.

Across the rollout, GameWAM repeatedly reacquires the horse, adjusts its position and viewpoint, and continues attacking until task completion. The example illustrates persistent goal-directed behavior under a changing target state: the high-level objective remains stable while the native actions needed for completion continuously adapt throughout the interaction. This highlights GameWAM's ability to maintain consistent behavior across multiple observation-action cycles.

Control under underwater visual and motion dynamics. Fig. 12 provides a qualitatively different interaction condition through the task *"Kill the salmon."* The entire engagement takes place underwater. Compared with ordinary terrestrial gameplay, the scene has different appearance, depth cues, target motion, and player locomotion. The target also moves through three-dimensional space rather than remaining approximately constrained to a ground plane.

GameWAM remains able to orient toward the salmon, track its changing location, and execute the required attack sequence until completion. This example does not by itself establish broad robust-



Figure 13: **Closed-loop rollout for *"craft item purple concrete powder."*** GameWAM progressively moves the required ingredients from the inventory into their corresponding crafting-grid locations and completes the target recipe. The sequence illustrates precise cursor control and structured multi-step placement through the native GUI action interface.

ness to arbitrary visual distribution shift, but it shows that successful native control is not restricted to the most common land-based interaction pattern represented by the other examples.

Precise spatial placement in the crafting grid. Gameplay and GUI interaction place substantially different demands on the same native keyboard–mouse interface. Gameplay primarily requires camera-centered movement and interaction, whereas crafting requires the cursor to manipulate small spatial targets under recipe-specific placement constraints.

Fig. 13 shows *"craft item purple concrete powder."* Successful execution requires multiple ingredient types to be moved from the inventory into specific locations in the crafting grid. The task therefore depends not only on selecting the correct objects, but also on controlling their relative positions and completing a multi-step placement sequence through continuous cursor motion and discrete mouse actions.

The rollout shows GameWAM progressively constructing the required arrangement and obtaining the target output. This example highlights the role of mode-specific GUI action modeling: the same physical mouse channels used for camera control during gameplay must here support precise cursor displacement, selection, and placement inside a structured interface.

Recovery from an intermediate placement error. Fig. 14 shows *"Create a dispenser in the crafting table."* This trajectory is particularly informative because successful execution is not obtained through a perfectly correct initial sequence. During crafting, GameWAM makes an intermediate placement error that leaves the grid in an incorrect state. Subsequent observations expose that state to the controller, after which the policy modifies the placement and eventually constructs the valid recipe.

The example illustrates a basic but important consequence of closed-loop GUI control. An intermediate low-level error need not irreversibly determine the remainder of the trajectory: because the model replans from the realized interface state, later actions can compensate for an earlier mistake. We refer to this behavior as closed-loop error recovery rather than high-level self-reflection, since no explicit symbolic diagnosis or textual reasoning process is assumed.

D.3.2 ViZDoom Closed-Loop Rollouts

The ViZDoom examples complement Minecraft by examining GameWAM under faster, more reactive combat dynamics. Unlike the Minecraft cases above, these tasks primarily reward continuous

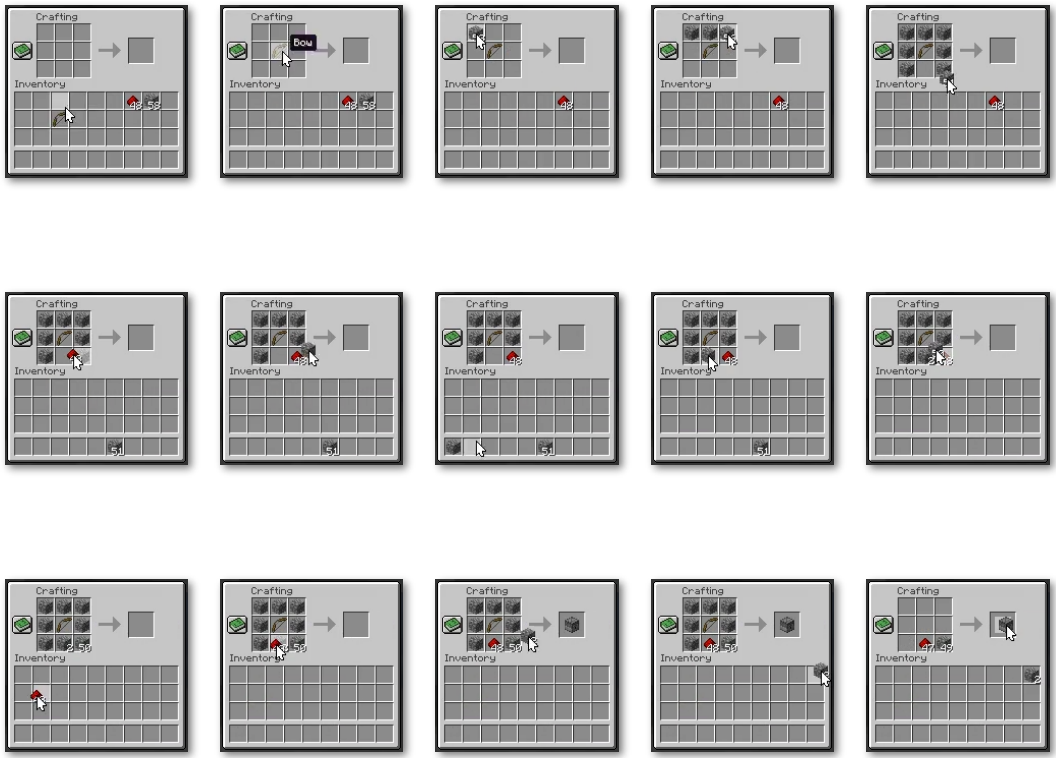


Figure 14: **Closed-loop rollout for *"Create a dispenser in the crafting table."*** GameWAM makes an intermediate placement error while constructing the recipe. After observing the resulting GUI state, subsequent actions adjust the crafting-grid configuration and complete the target item. The rollout illustrates error recovery enabled by repeated closed-loop observation and replanning.

survival and combat effectiveness rather than completion of a discrete object-manipulation goal. The four maps also impose different spatial structures: the two Battle scenarios require locomotion and target engagement through navigable environments, whereas the two defense scenarios emphasize rapid target selection under constrained movement.

Reactive combat and survival in Battle 1. Fig. 15 visualizes the instruction *"Fight off the monsters and stay alive as long as you can."* The rollout alternates between navigation and combat as enemies enter the agent's field of view. GameWAM changes orientation to bring nearby threats into the firing direction and engages them while continuing to move through the environment.

This task differs from a single-target Minecraft combat interaction because there is no terminal target whose elimination completes the instruction. Instead, the controller must repeatedly react to newly encountered threats while preserving its ability to continue the episode. The sequence therefore illustrates sustained closed-loop combat in which movement, target acquisition, and firing are repeatedly recomputed from new observations.

Combat with health-aware recovery in Battle 2. Fig. 16 shows the more spatially complex instruction *"Fight off the monsters in the maze and stay alive as long as you can."* Here the agent must combine combat with navigation through a maze-like environment, where walls and corridors repeatedly change which enemies and resources are visible.

The rollout also exposes an important form of closed-loop adaptation that is not captured by the final episode reward alone. After taking damage during an engagement, GameWAM does not simply continue attacking until failure. It navigates toward an available health pack, collects it to recover health, and then continues through the environment and resumes combat. The sequence therefore

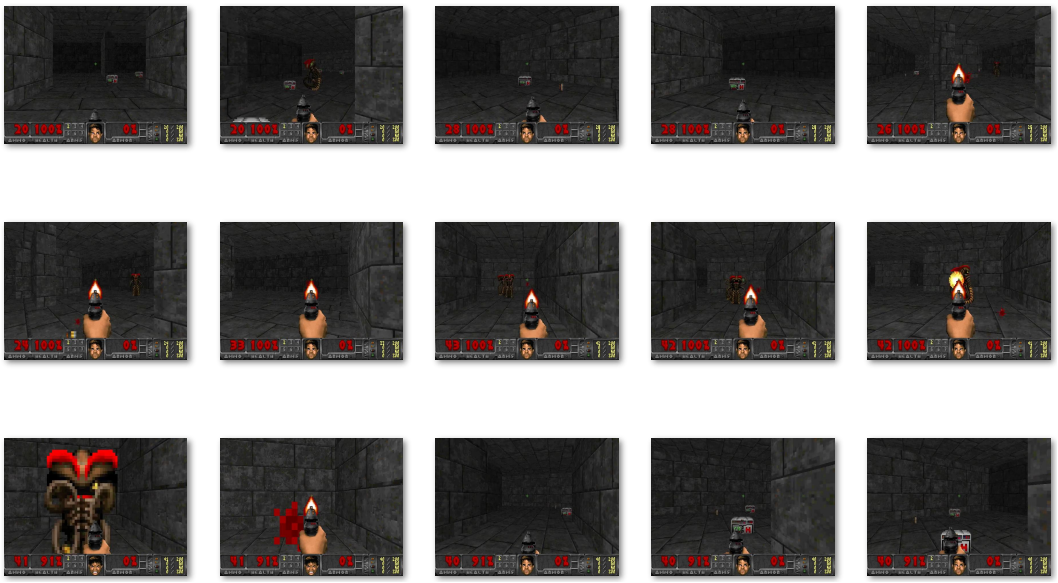


Figure 15: **Closed-loop rollout on ViZDoom Battle 1 for *"Fight off the monsters and stay alive as long as you can."*** GameWAM navigates through the environment, reacts to newly visible enemies, adjusts its view toward threats, and repeatedly engages them while continuing the survival-oriented rollout.

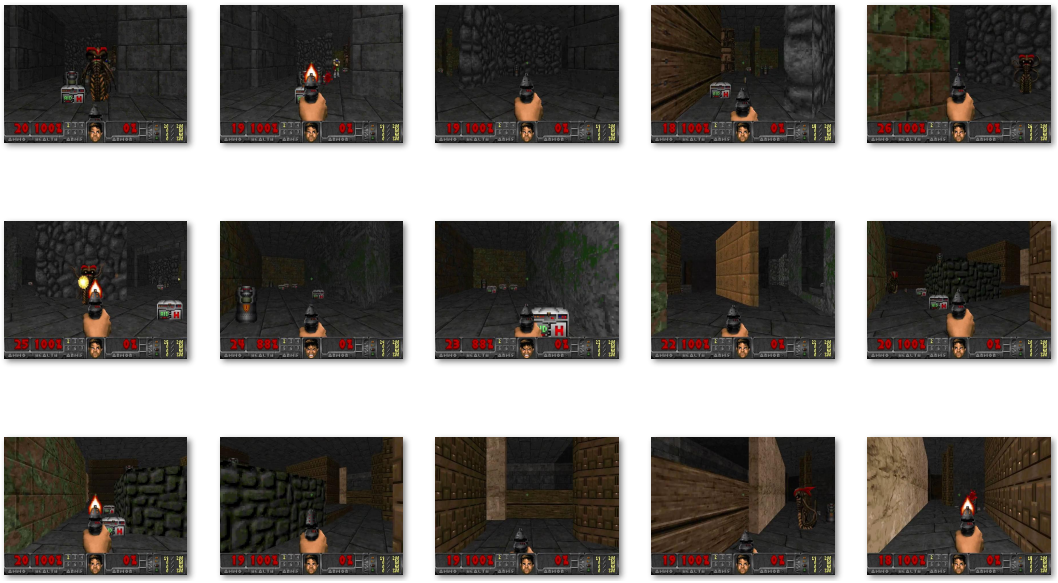


Figure 16: **Closed-loop combat and health recovery on ViZDoom Battle 2 for *"Fight off the monsters in the maze and stay alive as long as you can."*** GameWAM navigates and fights through the maze. After taking damage, the policy approaches and collects a health pack, recovers health, and subsequently continues the survival-oriented combat rollout. The sequence illustrates adaptation to realized agent state in addition to reactive enemy engagement.

demonstrates that the policy can condition its subsequent behavior on the realized consequences of earlier interaction, including changes in its own state.

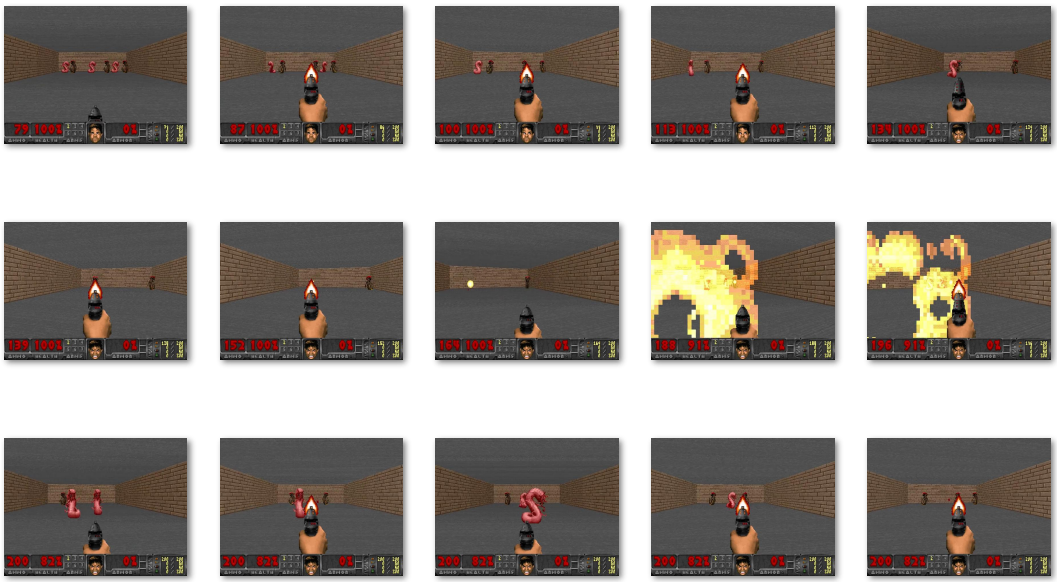


Figure 17: **Closed-loop rollout for *"Defend the line: shoot the monsters advancing toward you from across the room."*** GameWAM repeatedly acquires approaching enemies, adjusts horizontal aim, and fires as the active threat changes. The sequence highlights reactive closed-loop control under a frontal-defense objective.

We interpret this behavior as state-sensitive closed-loop recovery rather than explicit resource planning. No symbolic survival planner is provided; the change in behavior occurs through repeated prediction from the updated visual and proprioceptive state.

Maintaining a frontal defense line. Fig. 17 visualizes *"Defend the line: shoot the monsters advancing toward you from across the room."* Unlike the Battle maps, the spatial objective is comparatively constrained: enemies approach primarily from the forward region, and successful behavior requires repeated target acquisition and firing as successive threats advance.

GameWAM continually adjusts its horizontal aim between approaching enemies and engages targets before they reach the defended position. The rollout illustrates rapid visual–action feedback in a setting where the relevant target can change from one replanning step to the next and delayed correction would allow enemies to continue advancing.

Multi-directional threat response in Defend the Center. Fig. 18 considers *"Defend the center: shoot the enemies closing in from all directions."* The control challenge differs from Defend the Line because threats are not restricted to a single frontal direction. The policy must repeatedly change view orientation to locate and engage enemies appearing around the central position.

Across the rollout, GameWAM redirects its aim between different parts of the arena and fires on enemies as they enter actionable views. This example therefore emphasizes repeated orientation changes and target switching rather than pursuit through the environment. Together with Defend the Line, it shows that the same learned controller can realize qualitatively different combat patterns under different language-conditioned objectives.

Qualitative interpretation. The Minecraft and ViZDoom rollouts expose complementary properties of the same world–action policy. Minecraft emphasizes longer-horizon interaction: searching for an initially non-visible object, maintaining a long mining action, pursuing moving entities, manipulating structured interfaces, and recovering from an intermediate GUI error. ViZDoom instead emphasizes rapid feedback under continuously changing combat states: repeated target acquisition, survival-oriented navigation, adaptation after damage, and switching between threats under different spatial constraints.

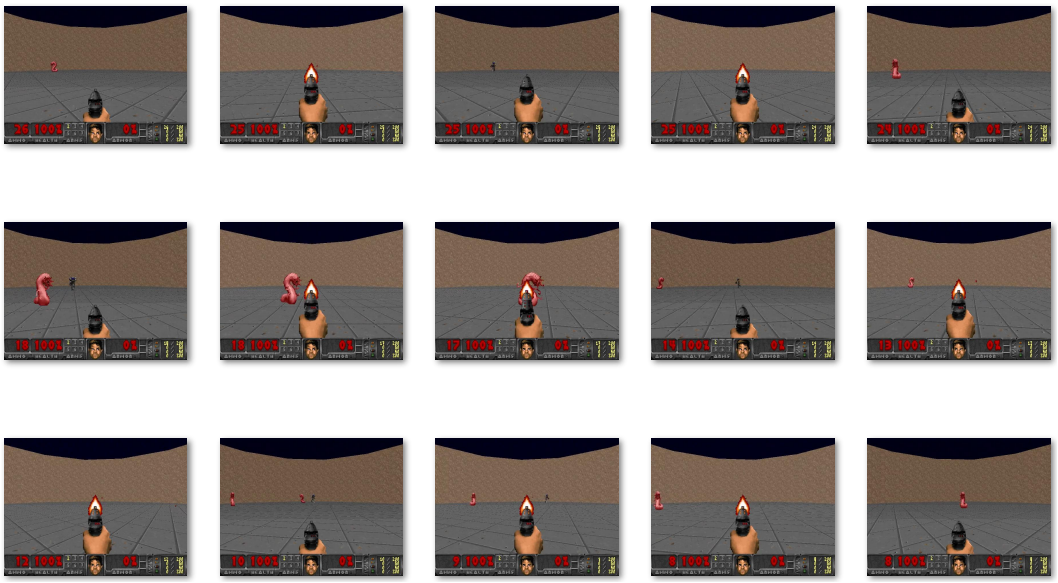


Figure 18: **Closed-loop rollout for *"Defend the center: shoot the enemies closing in from all directions."*** Threats can appear from different directions around the agent. GameWAM repeatedly redirects its view and switches targets as enemies approach, illustrating reactive multi-directional defense from a central position.

These examples also illustrate why closed-loop replanning is important beyond aggregate task success. Useful behavior need not correspond to a single unchanging action pattern. Obsidian mining requires maintaining an action despite repeated replanning, horse hunting requires preserving task intent while continually changing movement and attack commands, the dispenser example requires correcting an earlier interaction, and Battle 2 requires changing behavior after the agent's health state changes. In each case, subsequent actions are conditioned on realized environment feedback rather than on an unexecuted prediction of what should have happened.

As with any qualitative visualization, these trajectories should not be interpreted as estimates of how frequently each behavior occurs. They are selected rollouts intended to expose interaction patterns that are difficult to convey through aggregate success rate or reward alone. Quantitative Minecraft and ViZDoom performance remains reported through the evaluation results in the main paper.

D.4 Cross-Game Zero-Shot Transfer to VoxeLibre

We further probe cross-game transfer by directly deploying the Minecraft-trained GameWAM checkpoint in VoxeLibre without any environment-specific training or fine-tuning. The final checkpoint after the second Minecraft training epoch is used unchanged. Model-side configurations, including the 224×224 observation resolution and temporal settings, remain identical to the Minecraft evaluation. We only apply a deterministic interface mapping between VoxeLibre keyboard–mouse controls and the native action representation expected by GameWAM.

Because the GUI conventions of the two games differ substantially, this diagnostic focuses on basic gameplay interactions rather than interface tasks. We define six simple tasks spanning resource collection, combat, and block placement and provide each through a short natural-language instruction (e.g., "Chop down the oak log."). Each task is evaluated over 20 episodes. The primary metric is success rate; average executed environment steps are reported over successful episodes.

As shown in Table 7, GameWAM succeeds in 71 of 120 episodes, corresponding to a 59.2% micro-averaged success rate. Success rates vary substantially across tasks, ranging from 10.0% on Kill Cow to 85.0% on Chop Tree, with strong transfer also observed on Place Block (80.0%) and Mine

Table 7: **Zero-shot cross-game transfer from Minecraft to VoxeLibre.** The Minecraft-trained GameWAM checkpoint is evaluated without VoxeLibre training or fine-tuning. Each task is evaluated over 20 episodes; steps are averaged over successful episodes.

Task	Success	Rate	Steps / success
Chop Tree	17/20	**85.0%**	61.8
Mine Stone	14/20	70.0%	41.9
Mine Iron Ore	10/20	50.0%	49.4
Kill Cow	2/20	10.0%	46.5
Kill Zombie	12/20	60.0%	112.6
Place Block	16/20	80.0%	67.9
Overall	**71/120**	**59.2%**	–

Stone (70.0%). These results suggest that part of the learned visual–action control behavior transfers to a related but distinct game despite changes in rendering and interaction dynamics.

This experiment is intended as a diagnostic of cross-game generalization rather than as a new benchmark or a claim of broad out-of-domain transfer. Minecraft and VoxeLibre share related voxel-world structure and basic gameplay semantics, and the evaluated tasks deliberately focus on low-level gameplay skills for which a deterministic native-control interface mapping is possible.

D.5 Extended LASI Analysis

We use the LASI notation, source variable, and DCT convention defined in Section 5.5. This section provides additional source-space controls, frequency-resolved analyses, generative-trajectory diagnostics, and closed-loop measurements underlying the main-text findings.

D.5.1 Evaluation Setup and Frequency Decomposition

The qualitative LASI failure pattern was observed across multiple GameWAM checkpoints spanning different training stages, hyperparameter settings, and training configurations. To keep the controlled intervention suite internally consistent, all quantitative diagnostics reported below are performed on the final checkpoint after the second Minecraft training epoch, used here as the representative checkpoint, so every comparison uses the same underlying policy. Controlled source interventions hold the conditioning context fixed and vary only the explicit action source. For analyses that resolve source sensitivity along the generative trajectory, we use a deterministic 20-step integration path, providing finer denoising-time resolution than the standard closed-loop sampling configuration. This diagnostic setting is used only for the LASI analysis and does not alter the reported task-performance evaluation.

The controlled explicit-source matrix contains 24 fixed conditions and 16 action sources per condition ($24 \times 16 = 384$ samples). Low-frequency donor swapping and low-frequency zeroing each use the same controlled matrix; the antithetic analysis uses $24 \times 8 = 192$ source pairs. Confidence intervals reported below use 10,000 condition-cluster bootstrap resamples where available.

For each analyzed eight-step continuous camera chunk, we apply an orthonormal DCT along time. Modes $q = 0, 1, 2$ form the low-frequency band and modes $q = 3, \ldots, 7$ form the high-frequency band. P0–P2 and Y0–Y2 denote pitch and yaw DCT modes 0–2, respectively.

D.5.2 Within-Condition Source Association

For condition c, source j, action dimension d, and DCT mode q, define

$$r_{c,d,q} = \mathrm{Corr}_j \left(\widetilde{\widetilde{Z}}_{c,j,d,q}, \widehat{\widetilde{X}}_{0,c,j,d,q} \right). \tag{36}$$

The six low-frequency modes show substantial within-condition association. Averaging $r_{c,d,q}$ across the fixed conditions, pitch DCT0–2 have mean correlations 0.776, 0.613, and 0.746, while yaw

DCT0–2 have 0.890, 0.824, and 0.746. The yaw DCT0 95% confidence interval is $[0.835, 0.936]$. Because the condition is fixed within each group, this analysis isolates source-associated variation, but correlation alone is not used as the causal claim.

D.5.3 Donor, Zeroing, and Source-Space Controls

For a base source Z^b and donor Z^d, the low-frequency intervention is

$$\widetilde{Z}_q^{\text{swap}} = \begin{cases} \widetilde{Z}_q^d, & q \in \{0,1,2\}, \\ \widetilde{Z}_q^b, & q \in \{3,\dots,7\}. \end{cases} \tag{37}$$

The donor-follow rate asks whether the intervened output moves closer to the corresponding donor output than to the base output. For yaw DCT0, the donor-follow rate is 94.8% (95% CI $[92.2, 97.1]\%$), and the correlation between donor-induced source and output changes is $r = 0.921$.

For low-frequency zeroing, $\widetilde{Z}_q = 0$ for $q \in \{0,1,2\}$. Writing N_c and N_s for the numbers of fixed conditions and sources, respectively, define the condition-averaged within-condition source variances

$$V_{d,q}^{\text{base}} = \frac{1}{N_c}\sum_{c=1}^{N_c} \text{Var}_j\left[\widehat{\widetilde{X}}_{0,c,j,d,q}^{\text{base}}\right], \qquad V_{d,q}^{\text{zero}} = \frac{1}{N_c}\sum_{c=1}^{N_c} \text{Var}_j\left[\widehat{\widetilde{X}}_{0,c,j,d,q}^{\text{zero}}\right]. \tag{38}$$

The fraction of source-induced variance removed is

$$R_{d,q}^{\text{zero}} = 1 - \frac{V_{d,q}^{\text{zero}}}{V_{d,q}^{\text{base}}}. \tag{39}$$

Yaw DCT0 loses 99.25% of its source-induced variance (95% CI $[98.93, 99.50]\%$). The corresponding reductions for pitch DCT0–2 are 95.95%, 91.44%, and 93.21%; for yaw DCT1–2 they are 98.45% and 96.99%.

To compare variation across conditions with variation induced by the source, we additionally define

$$R_{d,q}^{\text{cond/src}} = \frac{\text{Var}_c\left[\frac{1}{N_s}\sum_{j=1}^{N_s} \widehat{\widetilde{X}}_{0,c,j,d,q}^{\text{base}}\right]}{V_{d,q}^{\text{base}}}. \tag{40}$$

We also use antithetic source pairs Z and $-Z$:

$$\widehat{X}_{\text{odd}}(Z) = \tfrac{1}{2}[\widehat{X}(Z) - \widehat{X}(-Z)], \qquad \widehat{X}_{\text{even}}(Z) = \tfrac{1}{2}[\widehat{X}(Z) + \widehat{X}(-Z)]. \tag{41}$$

Let $\mathcal{J}_\pm$ denote the subset of base sources paired with their negatives. The antithetic cancellation ratio for the corresponding DCT coefficient is

$$R_{d,q}^{\text{anti}} = 1 - \frac{\frac{1}{N_c}\sum_c \text{Var}_{j\in\mathcal{J}_\pm}\left[\widehat{\widetilde{X}}_{\text{even},c,j,d,q}\right]}{\frac{1}{N_c}\sum_c \text{Var}_{j\in\mathcal{J}_\pm}\left[\widehat{\widetilde{X}}_{0,c,j,d,q}^{\text{base}}\right]}. \tag{42}$$

Fig. 19 summarizes the source-space controls. The condition/source variance ratio should not be interpreted as source dominance in every mode: pitch DCT0 is 1.062, slightly above one, whereas the remaining reported ratios are 0.940, 0.329, 0.361, 0.239, and 0.086.

D.5.4 Frequency Selectivity

We compare three unit-norm perturbation directions: a low-frequency direction aligned with the leading principal direction of clean camera actions in DCT modes 0–2, an orthogonal direction within the same low-frequency subspace, and a high-frequency direction in modes 3–7. Thus, differences in response are not attributable to perturbation magnitude.

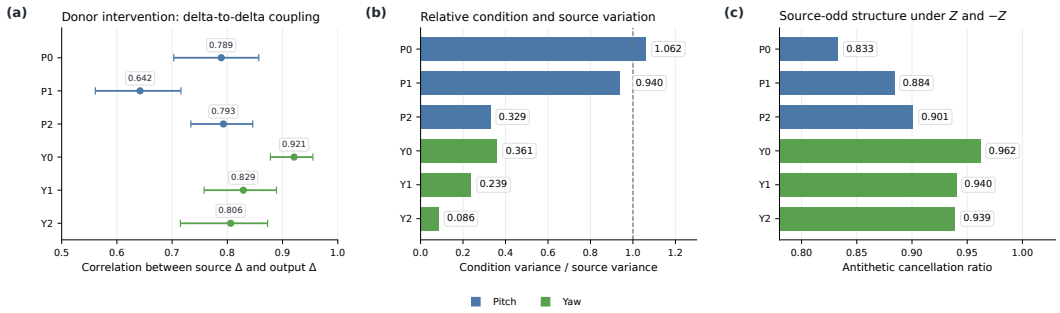


Figure 19: **LASI source-space controls. (a)** Correlation between the change imposed on a low-frequency source coefficient and the corresponding change in the predicted clean-action coefficient. **(b)** Condition-induced variance divided by source-induced variance; values below one indicate larger source variation for the measured mode. **(c)** Antithetic cancellation ratio for paired Z and $-Z$ sources, measuring source-odd structure. P0–P2 and Y0–Y2 denote pitch and yaw DCT modes 0–2.

For a fixed condition c and perturbation direction q, write $\widehat{X}_0(x;\sigma,c)$ for the reconstructed clean-action estimate obtained from noisy action state x at noise level σ. We estimate the local response by the centered finite difference

$$J_{\sigma,c}(q) \approx \frac{\widehat{X}_0(X_{\sigma,c}+\epsilon q;\sigma,c) - \widehat{X}_0(X_{\sigma,c}-\epsilon q;\sigma,c)}{2\epsilon}. \tag{43}$$

We measure the induced low-frequency response magnitude as

$$\rho_{\sigma,c}(q) = \|P_{\text{low}} J_{\sigma,c}(q)\|_2 , \tag{44}$$

where P_{low} projects the continuous camera trajectory onto DCT modes 0–2. The reported comparison uses $\epsilon = 0.05$.

The primary frequency-selectivity statistic compares the aligned low-frequency perturbation with the high-frequency perturbation through the condition-paired log-response difference

$$\Delta_\sigma^{\text{low/high}} = \frac{1}{N_c}\sum_{c=1}^{N_c} [\log \rho_{\sigma,c}(q_{\text{low}}) - \log \rho_{\sigma,c}(q_{\text{high}})] , \qquad G_\sigma^{\text{low/high}} = \exp\left(\Delta_\sigma^{\text{low/high}}\right). \tag{45}$$

Across the tested $\sigma \in \{0.20, 0.40, 0.55, 0.68, 0.80\}$, the geometric low/high response ratios are 4.35, 4.20, 3.53, 4.58, and 4.43. The aligned low-frequency direction is also compared with the orthogonal low-frequency direction; this difference is substantially smaller and depends on noise level. The supported conclusion is therefore strong selectivity between the tested low- and high-frequency perturbations, while evidence for additional selectivity among directions *within* the low-frequency subspace is weaker and noise-dependent.

D.5.5 Single-Forward Source Transfer

To separate one-call source transfer from iterative path effects, we directly construct

$$X_\sigma^{\text{ana}} = (1-\sigma)X_0 + \sigma Z \tag{46}$$

and evaluate the denoiser once at that fixed noise level, without taking an integration step.

To remove between-condition variation, source transfer is measured from within-condition source-pair differences. For two sources j and k under the same condition,

$$\Delta\widetilde{Z}_{c,jk,d,q} = \widetilde{Z}_{c,k,d,q} - \widetilde{Z}_{c,j,d,q}, \qquad \Delta\widehat{\widetilde{X}}^{\text{ana}}_{0,c,jk,d,q} = \widehat{\widetilde{X}}^{\text{ana}}_{0,c,k,d,q} - \widehat{\widetilde{X}}^{\text{ana}}_{0,c,j,d,q}. \tag{47}$$

The source-to-output gain for a DCT coefficient is the regression slope

$$g_{\text{ana}}(\sigma) = \frac{\text{Cov}_{c,j<k}\left(\Delta\widetilde{Z}_{c,jk,d,q}, \Delta\widehat{\widetilde{X}}^{\text{ana}}_{0,c,jk,d,q}\right)}{\text{Var}_{c,j<k}\left(\Delta\widetilde{Z}_{c,jk,d,q}\right)}. \tag{48}$$

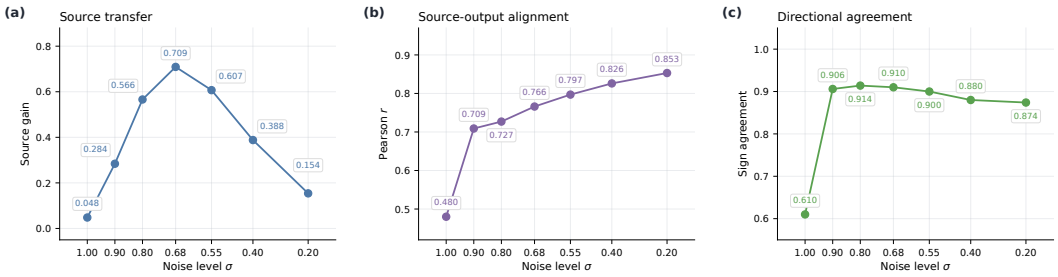


Figure 20: **Single-forward LASI across analytic noise levels.** **(a)** Regression gain from within-condition yaw-DCT0 source differences to the corresponding reconstructed clean-action differences. **(b)** Pearson correlation between the same source and reconstructed action differences. **(c)** Sign agreement of the paired differences. Each point uses one denoiser evaluation at an analytically constructed noisy state, with no integration update before measurement.

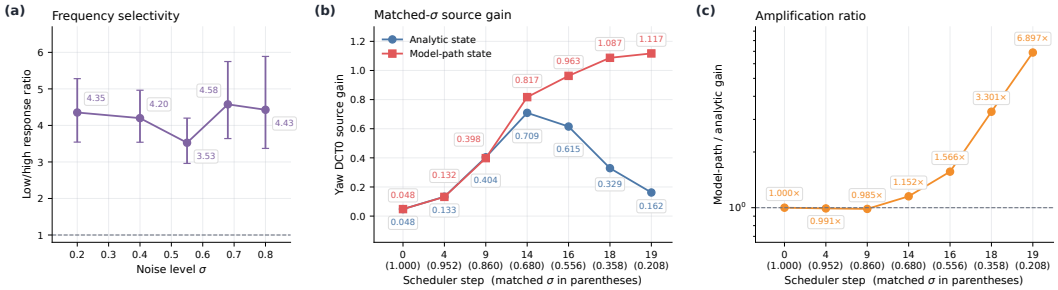


Figure 21: **LASI frequency selectivity and model-path amplification.** **(a)** Geometric response ratio of aligned low-frequency versus high-frequency perturbations over tested noise levels. **(b)** Yaw-DCT0 source gain for an analytically constructed noisy state and for the state reached along the iterative model path at the matched σ shown in parentheses. **(c)** Model-path gain divided by the magnitude of the analytic gain; the endpoint ratio reaches 6.897× at step 19 ($\sigma = 0.208$).

Fig. 20 shows that source transfer is already present before iterative denoising. For yaw DCT0, the gain peaks at 0.709 at $\sigma = 0.68$ in this sweep; the corresponding pairwise Pearson correlation increases from 0.480 at $\sigma = 1$ to 0.853 at $\sigma = 0.2$, while sign agreement is approximately 0.87–0.91 over most measured noise levels after the initial point.

D.5.6 Model-Path Amplification

For a state reached by the iterative model path at the same numeric noise level, define $g_{\text{path}}(\sigma)$ analogously using within-condition source-pair differences. The absolute and relative amplification statistics are

$$A(\sigma) = g_{\text{path}}(\sigma) - g_{\text{ana}}(\sigma), \qquad R_g(\sigma) = \frac{g_{\text{path}}(\sigma)}{|g_{\text{ana}}(\sigma)|}. \tag{49}$$

Fig. 21 combines the frequency-selectivity result with the matched-σ analytic/model-path comparison. At early denoising steps, the two gains are nearly identical. A positive gap appears in the mid/low-noise region; at step 19 ($\sigma = 0.208$), yaw DCT0 has analytic gain 0.162 and model-path gain 1.117, giving a 6.897× ratio.

Fig. 22 reports the same comparison as an absolute difference with uncertainty. The measured gap is essentially zero at steps 0–9, becomes positive at step 14, and increases to 0.955 at step 19.

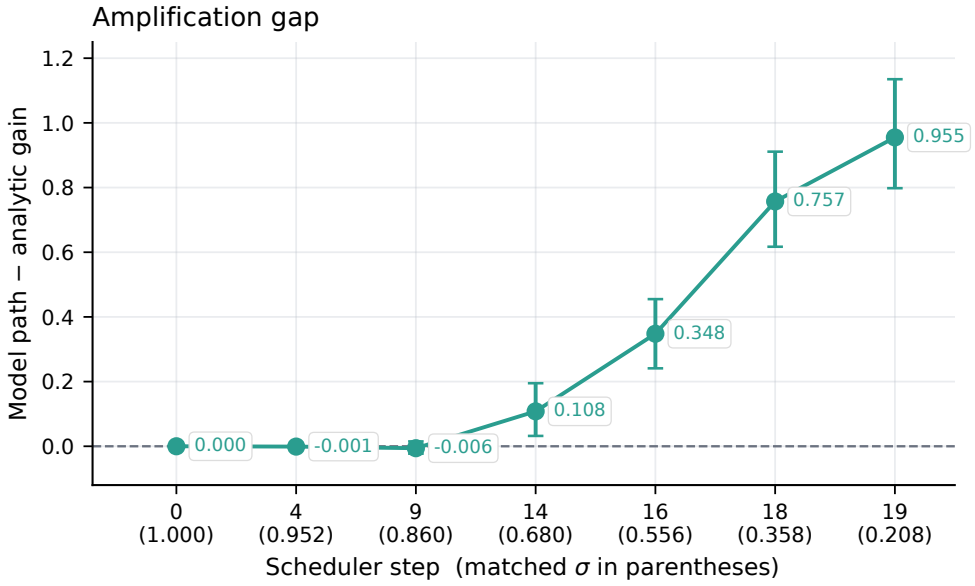


Figure 22: **LASI model-path amplification gap.** Difference $g_{\text{path}} - g_{\text{ana}}$ for yaw DCT0 at matched noise levels. Error bars are 95% confidence intervals; the gap is approximately zero early in the trajectory and increases to 0.955 at the final measured step.

D.5.7 Closed-Loop Alignment

The closed-loop analysis contains 300 traces spanning 30 tasks and 16,707 valid replanning steps. Let $z_{e,t,m}$ and $y_{e,t,m}$ denote the source and executed-action DCT coefficients for episode e, replanning step t, and low-frequency camera mode $m \in \mathcal{M} = \{\text{P0}, \text{P1}, \text{P2}, \text{Y0}, \text{Y1}, \text{Y2}\}$. To remove episode-level offsets, let $\bar{z}_{e,m}$ and $\bar{y}_{e,m}$ denote the corresponding within-episode means and define

$$\text{Align}(z, y) = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \text{Corr}_{e,t} \left(z_{e,t,m} - \bar{z}_{e,m}, y_{e,t,m} - \bar{y}_{e,m} \right). \tag{50}$$

To test whether the action source remains aligned with later behavior rather than merely reflecting a generic trajectory statistic, we circularly shift the source schedule relative to the rollout. For the 75-position replan schedule, define

$$S(\delta) = \text{Align}\left(\{z_{e,(t+\delta) \bmod 75,m}\}, \{y_{e,t,m}\} \right). \tag{51}$$

The unshifted statistic is 0.33763 and ranks first among all 75 circular shifts. As a separate permutation control, we randomly permute the source schedule $B = 999$ times. No permuted statistic exceeds the observed value, and the maximum permutation-null statistic is 0.06466. The corresponding finite-sample permutation value is

$$p = \frac{1 + \sum_{b=1}^{B} \mathbf{1}[S_b^{\text{perm}} \geq S_{\text{obs}}]}{B + 1} = 0.001. \tag{52}$$

For the 197 complete 75-replan episodes, we additionally average each action DCT coefficient across episodes at every replan position and correlate the resulting mean trajectory with the explicit source schedule. The correlations are $0.800, 0.738, 0.727$ for pitch DCT0–2 and $0.803, 0.808, 0.639$ for yaw DCT0–2.

D.5.8 Closed-Loop Accumulation and Source Resampling

The practical failure motivating LASI was most apparent when a single sampled action source was reused across successive replanning steps. Closed-loop behavior was strongly dependent on the reused source realization: for some sources, a persistent directional bias accumulated into repeated

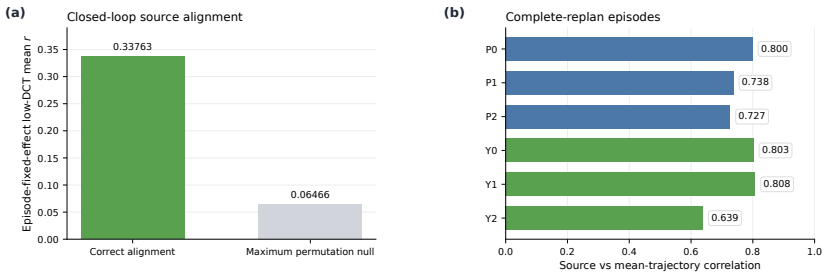


Figure 23: **Closed-loop LASI alignment. (a)** Episode-adjusted low-frequency association at the correct source/replan alignment compared with the maximum permutation-null statistic; the correct alignment additionally ranks first among all 75 circular shifts. **(b)** Correlation between the explicit source schedule and the cross-episode mean trajectory for complete 75-replan episodes, shown for pitch and yaw DCT modes 0–2.

in-place rotation, and in severe cases almost no tasks were completed. Resampling the action source between replanning steps largely removed this episode-level failure pattern.

To characterize why repeated-source reuse can produce such behavior, for a coarse camera quantity Δ_c at replanning step c, consider the decomposition

$$\Delta_c = \mu(\mathcal{C}_c) + b(Z_c) + \varepsilon_c, \tag{53}$$

where $\mu(\mathcal{C}_c)$ collects condition-dependent behavior and $b(Z_c)$ is a source-dependent low-frequency bias. This decomposition is explanatory rather than a claim that the measured trajectory is fully described by these terms.

If one source Z is reused for K cycles, the coherent component contributes

$$\sum_{c=1}^{K} b(Z) = K\, b(Z), \tag{54}$$

so a turning bias can be repeatedly reinforced. By contrast, if sources are resampled independently and $b(Z_c)$ is approximately centered,

$$Z_c \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I), \qquad \mathbb{E}\left[\sum_{c=1}^{K} b(Z_c)\right] \approx 0, \tag{55}$$

and

$$\mathrm{Var}\left[\sum_{c=1}^{K} b(Z_c)\right] = K\,\mathrm{Var}[b(Z_c)]. \tag{56}$$

Thus, source resampling removes the coherent episode-level reuse of a single source bias even though stochastic action variation remains. Consistent with this analysis, our closed-loop evaluation resamples the action source at every replanning step. We treat this as an evaluation-time mitigation of coherent accumulation rather than a mechanism-level correction of the underlying LASI sensitivity.

E Scope, Limitations, and Interpretation Boundaries

E.1 Method and Evaluation Scope

GameWAM is designed as a low-level closed-loop controller rather than a standalone high-level planner. Its policy is conditioned on current visual, language, historical, and proprioceptive context, but it does not explicitly maintain a symbolic task graph, inventory plan, recipe representation, or long-horizon search process. Failures that require semantic decomposition, exploration, or multi-stage resource planning therefore remain outside the intended scope of the block-cycle controller. A

natural extension is to pair GameWAM with a higher-level planner while retaining GameWAM as the native keyboard–mouse policy.

This distinction is particularly relevant to GUI interaction. Crafting, smelting, and inventory manipulation combine low-level cursor and keyboard control with structured procedural knowledge. Higher-level agents can externalize such procedures through symbolic actions, explicit plans, or dedicated executors, whereas GameWAM must realize them through the same low-level native-action policy. The scripted GUI data provides procedural supervision for these interactions, but the current model does not separately represent recipes or explicitly plan multi-step GUI procedures. Future systems could combine GameWAM with structured procedural representations or a higher-level planner while preserving closed-loop visual feedback at the execution level.

The current persistent history is also predominantly visual. Its auxiliary objective encourages the compressed cross-cycle representation to remain predictive of the current clean visual feature, but it does not directly supervise which historical information should be retrieved or used for downstream action generation and task completion. Future work could introduce task- or action-aware history objectives, explicit retrieval supervision, cross-cycle credit assignment, or multimodal memory that jointly represents visual observations, actions, inventory state, and higher-level task context.

Our empirical evaluation is restricted to digital game environments. Minecraft and ViZDoom provide diverse visual dynamics and native action spaces, but they do not establish transfer to physical control, where sensing delay, actuation noise, safety constraints, and contact dynamics differ substantially. The Minecraft results also depend on the available VPT/MineStudio data mixture; the current experiments do not isolate every possible dataset-composition factor. More broadly, current Minecraft Universe (MCU) training and evaluation settings largely focus on atomic task completion and individual interaction events. While useful for measuring specific capabilities, they provide limited supervision and evaluation for temporally extended behaviors where intermediate decisions influence future states and actions. Constructing high-quality chain-of- interaction trajectories with coherent intent, evolving state context, and aligned native actions remains challenging, and represents an important direction for future game-agent datasets and evaluations.

E.2 Interpretation of the LASI Diagnostics

The combined LASI evidence supports four specific statements. First, an explicit intervention on low-frequency source coefficients causally changes the matching low-frequency action output under fixed conditioning. Second, the effect is substantially stronger at low than at high temporal frequencies. Third, source transfer is already measurable in a single analytic forward pass. Fourth, the iterative model path amplifies the effect later along the generative trajectory, and a statistically aligned component remains detectable in closed-loop trajectories.

These results remain diagnostic rather than a complete mechanistic account. They do not identify a unique internal layer or training-time pathway, imply that conditioning information is absent, or establish that source variation explains the complete closed-loop trajectory. Empirically, the clearest LASI effect appears in continuous camera control. A plausible contributor is the pronounced low-frequency structure of camera trajectories in the training data: meaningful camera motion often evolves smoothly over short horizons, potentially making low-frequency source structure especially easy to align with and propagate through the generated action trajectory. This provides a possible data-level explanation for the concentration of the observed effect in camera motion.

Resampling the action source across replanning steps breaks repeated exposure to the same source bias and largely removes its harmful coherent accumulation in closed-loop interaction, but it does not remove the underlying source sensitivity. We therefore also explored training-time interventions aimed at the formation of LASI itself. These included frequency-selective objectives that directly discourage low-frequency source–action coupling, consistency objectives that reduce dependence on particular source realizations under the same conditioning, and additional supervision that strengthens the dependence of generated actions on conditioning. Across these directions, stronger suppression could interfere with normal action learning or degrade closed-loop performance, whereas other variants preserved task behavior but left substantial low-frequency source dependence. None therefore provided a robust mechanism-level correction without an accompanying trade-off.

We consequently use source resampling as an effective deployment-time mitigation while treating mechanism-level removal of LASI as an open problem. A desirable solution should prevent stochastic source structure from becoming a persistent low-frequency control bias without suppressing legitimate smooth motion, condition-compatible action diversity, or closed-loop responsiveness. Developing such objectives remains an important direction for generative action policies.