SHUGYO: ON-POLICY REINFORCEMENT LEARNING OF GAME AGENTS IN A GENERATIVE WORLD MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Game agents trained by supervised imitation on video–action data drift into states that demonstrations rarely cover, where small errors compound into task failure. Reinforcement learning (RL) can provide the missing experience of acting, failing, and recovering, but closed-source commercial games lack the fast dynamics, programmatic resets, and reward signals that RL requires. We present **SHUGYO**, a framework that adapts a pretrained video diffusion model into a learned environment for on-policy RL in the AAA action game *Black Myth: Wukong*. The simulator predicts future frames from rendered observations and native keyboard-and-mouse controls, resets rollouts from specified starting frames, and generates each chunk in four denoising steps. A rubric-guided vision–language judge scores the resulting rollouts, providing rewards without access to internal game state. Starting from a supervised vision–language agent, we perform RL post-training with GRPO and a two-stage curriculum entirely within the simulator, using a load-balanced client–server architecture for scalable rollout generation. Experiments show that the simulator follows native controls, remains visually coherent over extended rollouts, and adapts to a second game with only three hours of data. Per-task success rates in the simulator correlate with those in the real game, and RL post-training in the simulator raises real-game success from 58.6% to 68.6%. These results suggest that action-conditioned video world models can serve as practical training environments for game agents when the real game is not RL-ready. Supplementary videos are available at `https://cool-madeleine-56d762.netlify.app`.

1 INTRODUCTION

Game agents that play commercial action games from raw pixels face a demanding setting: using only the keyboard-and-mouse interface available to human players, they must couple fine-grained control with long-horizon task execution in visually complex AAA game environments. The prevailing approach is supervised imitation on large video–action corpora, and recent VLM-based agents have already learned basic navigation and combat skills in this way Chen et al. (2024). Imitation, however, has a fundamental limitation that more demonstrations cannot remove: once deployed, a behavior-cloned policy drifts into states that expert trajectories rarely visit, where small errors compound into task failure Ross et al. (2011). Game agents therefore need not only examples of successful behavior, but also experience of acting, failing, and recovering.

Reinforcement learning supplies exactly this experience, but it has been most effective in environments that are fast to simulate, programmatically resettable, and equipped with automatic reward signals. RL post-training in mathematics and code, for instance, benefits from automatically verifiable correctness and inexpensive episodes Guo et al. (2025), while standard testbeds such as the Arcade Learning Environment and the DeepMind Control Suite can be stepped, reset, and replicated under controlled conditions Bellemare et al. (2013); Tassa et al. (2018). Closed-source commercial games expose few of these interfaces. Without access to internal state, task progress and completion cannot be reliably detected. Without programmatic resets, trials cannot be repeated from comparable initial conditions, and because each instance runs only in real time, parallel rollout collection is costly. Additional compute can raise throughput, but it cannot supply the missing dynamics, reset, and reward interfaces. Commercial games thus offer abundant offline gameplay but no RL-ready training environment Chen et al. (2026b).

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.

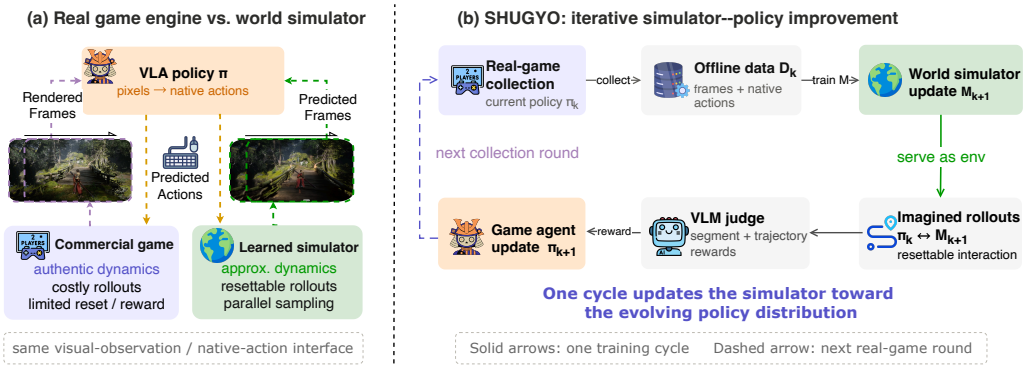

Figure 1: SHUGYO learns an RL-ready environment from offline gameplay. (a) The VLA policy uses the same visual-observation/native-action interface in both the commercial game and the learned simulator, while the simulator enables resettable, parallel, and reward-scored rollouts. (b) Each cycle updates the simulator from real-game data, generates imagined on-policy rollouts, scores them with a vision–language judge, and improves the policy with GRPO.

Learned world models offer a natural substitute. To serve as a training environment for game agents, such a simulator should ideally (i) expose rendered observations to the policy, (ii) support iterative on-policy updates of a standalone policy, and (iii) accommodate the native controls and long horizons of commercial games. Dreamer-style agents train policies entirely within learned dynamics Hafner et al. (2025b). However,their policies are typically learned over the world model's latent representations, which makes it less straightforward to train an off-the-shelf VLM agent acting on pixels. Ctrl-World fine-tunes a policy on successful imagined rollouts from a pixel-level world model Zhu et al. (2026), without iteratively re-rolling the updated policy. On-policy RL in learned simulators has shown promising results in robot manipulation Zhu et al. (2025); Xiao et al. (2025); Zhang et al. (2025) and, through DreamGym Chen et al. (2026b), in textual environments for LLM agents, whereas its application to commercial games remains comparatively limited.

We introduce **SHUGYO**[1], a framework that adapts a pretrained video diffusion model to serve as a learned environment for on-policy RL in a commercial AAA action game. Rather than reproducing the game engine, SHUGYO approximates the three interfaces required by an RL loop. Its dynamics interface predicts future observations from rendered frames and native keyboard-and-mouse controls. Its reset interface initializes rollouts from a specified starting observation, enabling repeated trials from comparable conditions. Its reward interface uses a rubric-guided vision–language judge that scores decoded rollouts against task-specific instruction-following criteria, providing a scalar reward without access to internal game state. A typed action encoder jointly models discrete key presses and continuous mouse motion, while a sink-anchored context helps maintain scene identity over long rollouts. Few-step Self-Forcing distillation Huang et al. (2025) further reduces the number of denoising steps to four per chunk, making on-policy rollouts affordable. Once learned from offline gameplay data, the simulator supports policy optimization without collecting RL rollouts from the real game, and the resulting policy can be deployed there unchanged.

Effective on-policy learning in the simulator still requires informative reward signals and efficient sampling. We train the policy with Group Relative Policy Optimization Shao et al. (2024). When sampled rollouts exhibit limited reward variation, the resulting group-relative learning signal can be weak. We therefore adopt a two-stage curriculum based on the measured success rates of the supervised policy, consolidating atomic skills before introducing low-success and compound tasks. To make simulator interaction scalable, we build on VERL Sheng et al. (2024) and decouple policy optimization from simulator inference through a client–server architecture with load-balanced simulator workers. As shown in Fig. 2, this design allows rollout generation to run in parallel and scale independently of policy updates.

In summary, our contributions are as follows:

[1]Shugyo denotes sustained training through disciplined practice, where competence is developed through experience rather than demonstration.

- We adapt a pretrained video diffusion model into a world simulator for *Black Myth: Wukong* that follows native keyboard-and-mouse controls, remains visually coherent over extended rollouts, generates each chunk in four denoising steps, and adapts to a second game with only three hours of data.
- Building on this simulator, we propose an on-policy RL framework for game agents that combines a rubric-guided VLM reward, GRPO with a two-stage curriculum, and a load-balanced client–server architecture for scalable rollout generation.
- Experiments show that per-task success rates in the simulator correlate strongly with those in the real game (Pearson r=0.88), and that RL post-training in the simulator raises the real-game success of a supervised agent from 58.6% to 68.6%.

2 Related Works

2.1 Action-Conditioned World Models

Whereas pretrained video generators are typically conditioned on text prompts or initial frames, action-conditioned world models condition generation on low-level control signals, such as camera trajectories, discrete keyboard states, and continuous mouse motion, so that predicted observations respond to an agent's actions over time. Early observation-generative models showed that interactive environments can be learned from visual data. GameNGen Valevski et al. (2025) simulates playable DOOM from past frames and player inputs. Genie Bruce et al. (2024) learns latent actions from unlabeled videos, while Matrix-Game 2.0 He et al. (2025) and Genie 3 Parker-Holder & Fruchter (2025) scale this paradigm to real-time, long-horizon interaction. As rollouts grow longer and models condition on their own outputs, control fidelity and temporal consistency become central challenges. Recent work addresses these challenges with more structured representations of actions and state. SCOPE Tong et al. (2026), ActWorld Xiong et al. (2026), and ForgeWM Li et al. (2026) factorize the action space by modeling spatially localized control effects, decoupling camera motion from interaction commands, and separating discrete keyboard states from continuous mouse motion, respectively. Others incorporate explicit state variables or 3D representations to improve mechanical and geometric consistency Lin et al. (2026); Meng et al. (2026). SHUGYO likewise adapts a pretrained video generator to native keyboard-and-mouse control, but assesses the resulting simulator by the policies it trains rather than by visual quality alone.

2.2 Learning Agents with World Models

By offering fast interaction, repeatable resets, and programmatic rewards, learned world simulators can serve as training environments for agents. Latent-imagination methods such as DreamerV3 Hafner et al. (2025a), Dreamer4 Hafner et al. (2025b), and IRIS Micheli et al. (2023) optimize policies over imagined latent or token trajectories rather than decoded observations. Observation-generative approaches instead expose predicted frames to the policy: SimPLe Kaiser et al. (2020) and DIAMOND Alonso et al. (2024) train Atari agents in learned video and diffusion environments, while Ctrl-World Zhu et al. (2026), UniSim Yang et al. (2024), and Interactive World Simulator Wang et al. (2026) use generated, action-conditioned experience for robot-policy learning. Recent work further performs policy optimization inside video world models, adding reward and termination models, imagined rollouts for RL, or mechanisms that mitigate model error and policy-induced distribution shift Xiao et al. (2025); Zhu et al. (2025); Zhang et al. (2025); Liu et al. (2026); Jiang et al. (2026); Zhang et al. (2026); WorldSync Chen et al. (2026a) further improves action grounding under off-expert interventions. Unlike SimPLe and DIAMOND, which alternate between real-environment interaction and imagined training with ground-truth rewards, SHUGYO learns dynamics, reset, and reward interfaces solely from offline gameplay of a closed-source commercial game, enabling on-policy RL without real-game rollouts.

3 Methodology

SHUGYO approximates the real game with a learned, action-conditioned world simulator, which serves as the training environment for on-policy reinforcement learning. We first adapt a pretrained video diffusion transformer to map rendered frames and native keyboard-and-mouse controls to

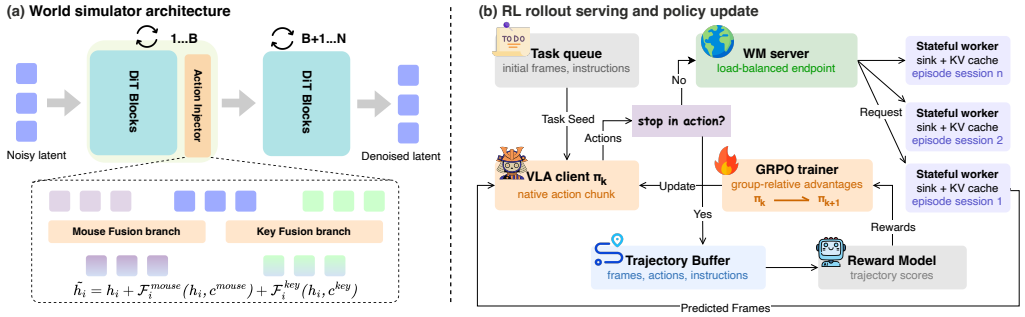

Figure 2: **Overview of SHUGYO.** (a) Native keyboard-and-mouse controls are encoded by a typed dual-branch action encoder and fused into the world simulator. (b) On-policy RL in the simulator, where a VLM judge scores rollouts and GRPO updates the policy.

future frames, and distill it with Self-Forcing into a 4-step simulator for efficient rollout generation. We then roll out an instruction-following policy inside the simulator, score the generated trajectories with a vision–language judge, and optimize the policy with GRPO. Because the simulator preserves the real game's pixel-observation and native-control interface, the learned policy can be deployed in the original game without interface adaptation.

3.1 Problem Setup

We model the underlying game as an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ that is observed only through rendered frames, with hidden game state $s_t \in \mathcal{S}$, native control $a_t \in \mathcal{A}$, transition kernel $P(s_{t+1} \mid s_t, a_t)$, task reward $r(s_t, a_t)$, and discount factor $\gamma \in [0, 1)$. The agent cannot access s_t and instead observes a rendered RGB frame $o_t = g(s_t) \in \mathbb{R}^{H_o \times W_o \times 3}$. Each native control is $a_t = (b_t, m_t) \in \{0, 1\}^{64} \times [-1, 1]^2$, where b_t is a multi-hot vector of simultaneous key states and m_t is the normalized two-dimensional mouse displacement. Given the observation–action history and a task instruction ℓ, the policy emits a K-step control chunk,

$$A_t = (a_t, \ldots, a_{t+K-1}) \sim \pi_\phi(\cdot \mid o_{0:t}, a_{0:t-1}, \ell). \tag{1}$$

The goal is to learn a policy that maximizes return in the source game,

$$J_{\mathcal{M}}(\pi_\phi) = \mathbb{E}_{\tau \sim (\mathcal{M}, \pi_\phi)} \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right]. \tag{2}$$

Direct on-policy optimization of this objective is impractical in a closed-source commercial game because the environment lacks scalable dynamics, reset, and reward interfaces. We therefore learn an action-conditioned world model from offline synchronized observation–action trajectories $\mathcal{D} = \{(o^{(n)}_{0:T_n}, a^{(n)}_{0:T_n-1})\}_{n=1}^{N}$. Given the past observations and a control chunk A_t, the model predicts future observations,

$$\hat{O}^+_t = (\hat{o}_{t+1}, \ldots, \hat{o}_{t+K}) \sim W_\theta(\cdot \mid o_{0:t}, A_t). \tag{3}$$

As shown in Fig. 1, for simulator-based training, π_ϕ and W_θ are chained autoregressively: generated observations extend the history, the policy samples the next control chunk, and the world model generates the next observations. This defines synthetic trajectories $\hat{\tau} \sim (W_\theta, \pi_\phi)$ without interacting with the real game. SHUGYO thus instantiates the three RL interfaces with learned components: W_θ serves as the dynamics interface, starting each rollout from an initial observation o_0 drawn from $\mathcal{D}$ serves as the reset interface, and a task-conditioned reward estimator $\hat{R}(\hat{\tau}, \ell)$ serves as the reward interface. We optimize the surrogate objective in place of $J_{\mathcal{M}}$:

$$J_W(\pi_\phi) = \mathbb{E}_{\hat{\tau} \sim (W_\theta, \pi_\phi)} \left[\hat{R}(\hat{\tau}, \ell) \right] \tag{4}$$

3.2 Action-Conditioned World Simulator

Model architecture and action injection. Native keyboard-and-mouse controls are heterogeneous: key states b_t are sparse, discrete, and multi-label, whereas mouse motion m_t is continuous

and temporally smooth. As shown in Fig. 2, we handle this heterogeneity with a typed dual-branch action encoder that embeds b_t and m_t through separate branches. A lightweight block-causal Transformer models dependencies among these typed action tokens, allowing key and mouse inputs within the same frame to interact while preventing access to future controls. The resulting discrete and continuous action contexts are kept as two separate streams, aligned to the video-latent timeline, and injected into selected DiT blocks through zero-initialized gated residual fusion. This design preserves the pretrained visual backbone at initialization while providing the generator with temporally aligned, per-frame native control signals.

Sink-anchored visual context. Since each chunk observes only a local context, autoregressive generation tends to drift in scene identity, while attending to the full history is prohibitive for long rollouts. We therefore anchor every chunk to a persistent visual reference: the initial observation o_0 is VAE-encoded into a sink latent that all visual tokens can attend to. The sink carries no action conditioning and is excluded from the loss, serving purely as a visual anchor. During training, chunks are sampled at varying offsets from the same anchor, exposing the model to the long anchor-to-chunk distances encountered at inference. Resetting the simulator then amounts to encoding a new o_0 as both the first-chunk condition and the sink.

Action-conditioned fine-tuning. We adapt the pretrained video prior to the target game domain and native-control interface through supervised world-model training on $\mathcal{D}$. We optimize the action-conditioned generator with the standard flow-matching video objective in latent space (Liu et al., 2022; Lipman et al., 2023). The model receives the initial visual condition, the sink anchor, and the aligned control chunk A_t, and learns to predict the flow-matching target for the corresponding video segment. Within each training clip, the video transformer can exploit bidirectional spatiotemporal context to learn high-fidelity local dynamics, while the action encoder remains causal so that control conditioning never uses future actions. This stage turns the pretrained generator into a controllable, bidirectional teacher that maps native controls to visual consequences.

Self-Forcing and autoregressive inference. Policy training requires a simulator that is stable in closed loop and fast enough to generate fresh rollouts at every update. The bidirectional teacher, however, generates each clip jointly from clean real-video context, whereas closed-loop rollout proceeds chunk by chunk from its own outputs. We therefore distill it into a block-causal 4-step student with Self-Forcing (Huang et al., 2025): the student attends only to the sink and preceding chunks via a KV cache, and is unrolled autoregressively during distillation under a distribution-matching objective against the teacher, so that it learns from the self-generated contexts seen at inference. At inference, the first chunk is conditioned on o_0, the sink, and A_0, and each subsequent chunk on the sink and the latent frames of the 6 most recent chunks. The resulting long rollouts support on-policy training (Sec. 3.3), with training details given in Appendix B.

3.3 On-Policy Reinforcement Learning in the Simulator

Behavior-cloned policy initialization. We instantiate the policy for our target environment, *Black Myth: Wukong*, and refer to it as **WukongAgent**. The policy is initialized from Qwen3.5-4B: conditioned on the observation–action history and a textual instruction ℓ, it emits a K-step chunk of native keyboard-and-mouse controls (Eq. 1), matching both the simulator input and the real-game control interface. Supervised fine-tuning on instruction-following demonstrations yields **WukongAgent-SFT**, which serves as the starting point for simulator-based RL. This behavior-cloned policy can execute basic skills, but imitation alone mainly reproduces the demonstrated mapping from instructions to controls and provides little signal for off-demonstration deviations. As small errors accumulate over long horizons, the policy has no learned mechanism for recovery. The distilled simulator supplies this missing interaction without requiring additional real-game RL rollouts.

Scalable simulator rollout. A central bottleneck in on-policy learning with a generative simulator is fresh rollout generation: every update requires newly sampled trajectories from the current policy. As shown in Fig. 2, we build on VERL (Sheng et al., 2024) and decouple simulator serving from policy updates. The policy acts as a *client* that submits action chunks, while the simulator is served as a *stateful worker* that maintains the per-episode sink latent and KV cache and returns the next observation chunk. Each episode is a sequence of requests to the same stateful session, and resetting

a simulated environment reduces to clearing the session state. Requests are dispatched across replicated simulator workers by a load balancer, allowing rollout throughput to scale with the number of workers and improving utilization under variable episode lengths.

Reward from a vision–language judge. A programmatic reward is unavailable in a commercial game whose internal state cannot be queried. We therefore formulate reward assignment as a vision–language judgment task. A rubric-guided vision–language judge, instantiated with Gemini-3.6-Flash (Google DeepMind, 2026), scores each decoded rollout against task-specific instruction-following criteria derived from ℓ, yielding a scalar reward $\hat{R}(\hat{\tau}, \ell)$. This provides a reward interface that requires no game instrumentation. The full rubric and prompt are given in the Appendix D.2.

On-policy group optimization. We optimize the agent with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). A task is specified by an initial frame o_0 and an instruction ℓ. For each task, we sample G trajectories $\{\hat{\tau}^{(i)}\}_{i=1}^{G}$ from the current policy inside the simulator and score each with the judge, obtaining $\hat{R}^{(i)} = \hat{R}(\hat{\tau}^{(i)}, \ell)$. The group-relative advantage is

$$\hat{A}^{(i)} = \frac{\hat{R}^{(i)} - \mathrm{mean}(\{\hat{R}^{(j)}\}_{j=1}^{G})}{\mathrm{std}(\{\hat{R}^{(j)}\}_{j=1}^{G}) + \epsilon}. \tag{5}$$

Two properties make GRPO well suited to this setting. First, it avoids fitting a value function over high-dimensional generated frames, which would be costly and would have to track the simulator-induced state distribution. Second, group normalization emphasizes trajectories that outperform other samples from the same task, amplifying rare successes under sparse rewards.

Two-stage curriculum. Group-relative updates require reward variation within each rollout group: tasks on which the initial policy uniformly fails yield near-zero advantages and inefficiently consume simulator rollouts. Before RL training, we therefore estimate the empirical success rate of WukongAgent-SFT with 8 rollouts per task and use it to construct a fixed two-stage curriculum. The first stage trains on atomic-skill tasks (*i.e.*, `continuous navigation`, `entity-directed navigation`, and `traversal`) whose success rate exceeds a threshold m , so that the policy consolidates the primitives needed by later compound tasks. The second stage adds the remaining low-success-rate tasks and all compound tasks, whose chained sub-goals make them more sensitive to accumulated errors. The resulting agent is denoted **WukongAgent-RL**.

4 Fidelity of the World Simulator

4.1 Experimental Setup

Datasets. We train and evaluate our world simulator in *Black Myth: Wukong*, a commercial AAA action role-playing game (ARPG) with dense third-person camera motion, frequent combat interactions, and visually rich 3D scenes. For training, we collect nearly 190 hours of frame-aligned video–action pairs, where each frame is paired with the native keyboard-and-mouse signal issued by the player. For evaluation, we construct a held-out single-chunk benchmark of 50 initial-frame/action-trajectory pairs from gameplay segments disjoint from the training set. Each example contains one initial frame and a 48-step native action trajectory. Conditioned only on these inputs, the simulator predicts the corresponding future frame sequence, with ground-truth future frames reserved solely for evaluation. To test cross-game transfer, we further adapt the simulator to *Honor of Kings: World* (NGR) using only 3 hours of video–action data, roughly $1/60$ of the Black-Myth training budget. We construct 40 NGR evaluation examples using the same protocol, allowing us to measure action-following fidelity in a new game under a low-data transfer.

Simulator implementation. We use the action-conditioned world simulator described in Sec. 3.2. The simulator is initialized from Wan2.2-TI2V-5B Wan et al. (2025), adapted to the native keyboard-and-mouse control interface, and distilled into a few-step autoregressive model for efficient rollout generation. Full details on the flow-matching adaptation, optimization hyperparameters, chunk-wise rollout protocol, and NGR few-shot fine-tuning are provided in Appendix B.

Table 1: Quantitative evaluation of the world simulator on *Black Myth: Wukong*. Shaded rows are ablations of SHUGYO. Bold marks the best result among the compared models (top block).

Model	FDS↑	EPE↓	CFD↑	Dyn_Deg↑	Mot_Smooth↑
SolarWM (Wan2.2-5B)	0.588	10.44	0.12	**0.88**	0.15
Matrix-Game 3.0	0.638	**8.42**	0.18	0.70	**0.16**
SHUGYO	**0.660**	9.52	**0.20**	0.79	**0.16**
SHUGYO w/o sink latent	0.638	9.90	0.20	0.78	0.13
SHUGYO w/o decoupled fusion	0.629	10.06	0.13	0.72	0.12



Figure 3: Action cloning on *Black Myth: Wukong*. The same native action sequence, applied to different initial frames, produces consistent camera and character motion while preserving each scene's appearance and layout.

4.2 World Simulator Performance Analysis

Metrics. We adopt a set of complementary visual metrics that probe motion agreement, action responsiveness, and temporal quality. Flow Direction Score (FDS) measures the cosine similarity between predicted and ground-truth optical flow, while End-Point Error (EPE) measures their displacement error; FDS isolates directional agreement, whereas EPE additionally penalizes magnitude errors. Counterfactual Distance (CFD) measures action sensitivity by comparing predictions generated from the same initial frame under different action trajectories. Dynamic Degree (Dyn_Deg) quantifies the spatial extent of generated motion, and Motion Smoothness (Mot_Smooth) measures the temporal regularity of the estimated flow field. More details can be found in the Appendix C.

Quantitative analysis. We compare against Matrix-Game 3.0 Skywork AI Matrix-Game Team (2026), a representative game world model conditioned on keyboard-and-mouse actions and camera poses. We also compare against SolarWM Huang et al. (2026), a representative camera-controlled general-purpose world model, using its Wan2.2-5B variant that shares our backbone. As shown in Table 1, SHUGYO achieves the highest FDS and CFD, indicating that its predicted motion is best aligned in direction with the ground truth and most responsive to changes in the native action trajectory. Its EPE is lower than that of SolarWM but higher than that of Matrix-Game 3.0, suggesting that motion magnitude is predicted less accurately than motion direction. SHUGYO matches Matrix-Game 3.0 in Mot_Smooth, so its improved action responsiveness does not come at the cost of temporal stability. Taken together, these results suggest that the simulator learns action-conditioned dynamics rather than merely preserving the appearance prior of the image-to-video backbone.

Ablation study. The shaded rows of Table 1 ablate two components of the simulator. Removing the sink latent leaves CFD unchanged but lowers FDS from 0.660 to 0.638, raises EPE from 9.52 to 9.90, and reduces Mot_Smooth from 0.16 to 0.13, indicating that the persistent visual anchor mainly improves motion fidelity and temporal stability rather than action sensitivity. Replacing the decoupled fusion with a single shared injection stream causes the largest drop in CFD, from 0.20 to 0.13, together with degraded FDS, EPE, and Mot_Smooth. This suggests that injecting key and mouse contexts through separate streams is important for action responsiveness.

Table 2: Cross-game transfer to *Honor of Kings: World* (NGR). Top: zero-shot inference; bottom: training on NGR data. †Reported in clips, as the total duration is not disclosed.

Model	Pre-training data	NGR data	FDS↑	EPE↓	CFD↑	Dyn_Deg↑	Mot_Smooth↑
Matrix-Game 3.0	4.8M clips†	–	0.532	11.34	0.12	0.75	0.19
SHUGYO	190 h (Wukong)	–	0.497	10.21	0.13	0.74	0.19
SHUGYO	–	3 h	0.439	10.50	0.03	0.72	**0.22**
SHUGYO	190 h (Wukong)	3 h	**0.659**	**7.50**	**0.16**	**0.77**	0.20

Action cloning across initial states. We further examine whether the simulator captures reusable control semantics by feeding the same native action trajectory to multiple distinct initial frames. As shown in Fig. 3, the generated sequences exhibit highly consistent action trajectories—for example, comparable camera turns, forward displacement, and interaction timing across different scenes—while retaining scene-specific appearance and layout. This action-cloning behavior suggests that the learned control interface is not a post-hoc visual correlation tied to a single clip, but a transferable mapping from native controls to visual consequences. Such consistency is essential for a simulator intended to serve as a policy-training environment, because the policy must receive predictable visual feedback when issuing the same control pattern in different states.

Cross-game transfer. We test whether the action–visual correspondence learned in *Black Myth: Wukong* transfers to an unseen game (Table 2). In the zero-shot setting, SHUGYO, pre-trained on a single game, obtains lower EPE and slightly higher CFD than Matrix-Game 3.0 but lower FDS, and both models lag far behind the post-trained simulator. Training on the 3-hour NGR data alone yields a near-zero CFD of 0.03, indicating that this data scale is insufficient to learn responsive action conditioning; its higher Mot_Smooth therefore does not reflect better dynamics. Post-training the Wukong-pretrained simulator on the same data achieves the best FDS, EPE, CFD, and Dyn_Deg, raising FDS from 0.439 to 0.659 and CFD from 0.03 to 0.16. This suggests that much of the native-control grounding transfers across titles, with lightweight post-training mainly adapting to the visual appearance and motion statistics of the target game.

5 Policy Evaluation in the World Simulator

Frame-level fidelity (Sec. 4.2) does not by itself ensure that task outcomes in the simulator match those in the real game, and such mismatch would yield misleading learning signals. We therefore ask a prerequisite question: *at what granularity of instruction following can the simulator serve as a reliable interactive environment for a game agent?* We answer it by executing the same agent in both environments.

Evaluation protocol. We build a benchmark of 14 instruction-following tasks in *Black Myth: Wukong*, each defined by a distinct scene and instruction and held out from the agent's training data. The tasks span four categories of increasing difficulty: (i) *continuous navigation* (e.g., following a stream forward), (ii) *entity-directed navigation* (e.g., moving toward the shrine on the front right), (iii) *traversal* (e.g., passing between two pillars on the front left), and (iv) *compound* tasks that chain multiple sub-goals (e.g., passing through the stone gate ahead and then crossing the wooden bridge). Each task is evaluated over 10 trials in both environments, starting from the same initial observation. In both environments, success is determined by the same rubric-guided vision–language judge applied to the rollout video. The task list and per-category statistics are given in the Appendix D.4.

Real environment vs. world simulator. We evaluate **WukongAgent-SFT** (Sec. 3.3) in the real game and in the simulator. As shown in Fig. 4, the per-task success rates in the two environments are strongly correlated (Pearson $r = 0.88$, Spearman $\rho = 0.90$), suggesting that simulator success is indicative of real-game success. The simulator is, however, systematically more pessimistic, yielding a lower success rate on 12 of the 14 tasks. Agreement is tighter on non-compound tasks (i–iii; $r = 0.92$, deviation 20.0%) than on compound tasks ($r = 0.82$, deviation 25.0%), likely because errors accumulate over their longer, multi-stage rollouts. The simulator thus preserves the relative difficulty of tasks, more faithfully for atomic skills than for compound tasks, which motivates the curriculum in Sec. 3.3 and the RL experiments that follow.

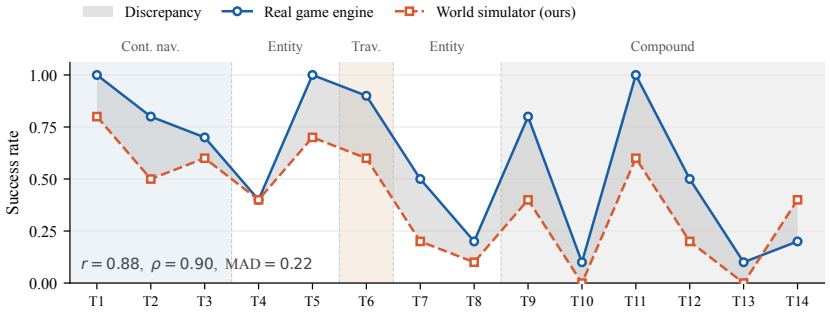

Figure 4: Policy evaluation in the learned world simulator. Per-task success rates of WukongAgent in the real game and in the learned simulator across 14 held-out instruction-following tasks.

Table 3: Real-game success rates (%) by task category. The number of tasks per category is given in parentheses. Each task is evaluated over 10 trials.

Policy	Cont. nav. (3)	Entity nav. (4)	Traversal (1)	Compound (6)	Overall (14)
WukongAgent-SFT	83.3	52.5	90.0	45.0	58.6
WukongAgent-RL	**100.0**	**62.5**	**100.0**	**51.7**	**68.6**

6 Policy Optimization in the World Simulator

We next examine whether the learned world simulator can serve not only as a generator of future observations but also as an effective training environment for game agents. Starting from the supervised policy **WukongAgent-SFT**, we post-train it with GRPO , performing all on-policy RL updates within the simulator. The resulting policy, **WukongAgent-RL**, is then evaluated in the real game environment under the same pixel-observation and native-control interface as the SFT baseline. On *Honor of Kings: World*, RL post-training in the simulator improves overall real-game success by 22% (Details in the Appendix F.2).

Real-game improvement. Table 3 compares WukongAgent-SFT with WukongAgent-RL in the real game, per-task results are given in the Appendix F.1. Post-training WukongAgent-SFT with RL solely in the simulator raises overall real-game success from 58.6% to 68.6%, suggesting that policy updates obtained in the learned simulator transfer to the original game. Gains appear in all four categories and on 8 of the 14 tasks, most notably on approaching the shrine ahead (40% to 90%) and on crossing the muddy ground toward the temple (20% to 70%).

Category-level trends. The gain is largest on continuous navigation (+16.7%, reaching 100%), followed by entity-directed navigation and traversal (+10.0%), and smallest on compound tasks (+6.7%). This ordering is consistent with the curriculum in Sec. 3.3, which first consolidates atomic skills, and with the larger simulator deviation on compound tasks (Sec. 5). Success decreases on three tasks: two compound tasks that begin with a turn, and one entity-directed task that requires approaching a small target. We hypothesize that these behaviors are simulated less accurately, which makes the judge reward less informative.

7 Conclusion

We presented **SHUGYO**, which turns a pretrained video diffusion model into a learned RL environment for closed-source commercial games. An action-conditioned simulator and a rubric-guided vision–language judge approximate the dynamics, reset, and reward interfaces, enabling RL post-training of a supervised agent without real-game rollouts. On *Black Myth: Wukong*, simulator success rates correlate with real-game ones, and RL post-training in the simulator improves real-game success. These results suggest that learned world simulators can bridge abundant offline gameplay data and the resettable, scalable interaction that RL requires in commercial games.

AI USE STATEMENT

We used generative AI tools solely to polish the language of this manuscript. The research ideas, method design, experiments, and conclusions are the authors' own. All experimental data were collected by the authors, and all reported numbers, figures, and claims were checked by the authors against the raw results. The authors reviewed every AI-assisted edit for accuracy and take full responsibility for the content of this work.

ETHICS STATEMENT

This work uses gameplay recordings of *Black Myth: Wukong* and a second commercial game, collected by human players who gave informed consent. The recordings contain only rendered game frames and keyboard-and-mouse inputs, with no personally identifiable information. The game content is used solely for non-commercial research. Formal institutional ethics approval was not sought, as the study poses minimal risk to participants. The learned simulator and agent are designed for research on game agents and are not intended to reproduce or redistribute copyrighted game content.

REPRODUCIBILITY STATEMENT

The main text and appendix describe the world simulator architecture, the typed action encoder, the training and Self-Forcing distillation settings, the evaluation metrics, and the 14 evaluation tasks with per-task results. We also specify the rubric-guided VLM judge, the GRPO hyperparameters, the two-stage curriculum, and the client–server rollout architecture. We plan to release code and model weights upon acceptance, subject to internal review.

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. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.12399.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, et al. Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15391*, 2024. URL https://arxiv.org/abs/2402.15391.

Peng Chen, Pi Bu, Jun Song, Yuan Gao, and Bo Zheng. Can VLMs play action role-playing games? take Black Myth Wukong as a study case. *arXiv preprint arXiv:2409.12889*, 2024.

Sixiang Chen, Jiaming Liu, Jixian Wu, Yichen Guo, Tinghao Wang, Siyuan Qian, Hao Chen, Jiajun Cao, Jian Tang, and Shanghang Zhang. Do robotic world models really follow actions? diagnosing and aligning action-conditioned generation for policy learning. *arXiv preprint arXiv:2608.24885*, 2026a. URL https://arxiv.org/abs/2608.24885. Concurrent work.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Xuefei Cao, Yuanhao Xiong, Haibo Tong, et al. Scaling agent learning via experience synthesis. In *International Conference on Learning Representations*, volume 2026, pp. 121394–121420, 2026b.

Google DeepMind. Gemini 3.6 flash. https://deepmind.google/technologies/gemini/, 2026. Model version X, accessed DATE.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b. URL https://arxiv.org/abs/2509.24527.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, et al. Matrix-game 2.0: An open-source, real-time, and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025. URL https://arxiv.org/abs/2508.13009.

Junchao Huang, Guian Fang, Shengju Qian, et al. Solarwm: Open data and scalable training for long-horizon video world models. *arXiv preprint arXiv:2609.02886*, 2026.

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

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, Yu Wang, Haoran Li, Chao Yu, and Dongbin Zhao. WoVR: World models as reliable simulators for post-training VLA policies with RL. *arXiv preprint arXiv:2602.13977*, 2026. URL https://arxiv.org/abs/2602.13977.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, et al. Model-based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1903.00374.

Xinye Li, Lingshuai Lin, Lei Wang, Liuzhou Zhang, Jialin Cui, Qingshan Li, Guanchu Wang, Qingbin Liu, Xi Chen, Jiang Bian, and Wai Lam. ForgeWM: Progressive causal training for few-step action-conditioned video world models. *arXiv preprint arXiv:2608.14022*, 2026. URL https://arxiv.org/abs/2608.14022.

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

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 (ICLR)*, 2023.

Xiaokang Liu, Zechen Bai, Hai Ci, Kevin Yuchen Ma, and Mike Zheng Shou. World-VLA-Loop: Closed-loop learning of video world model and VLA policy. *arXiv preprint arXiv:2602.06508*, 2026. URL https://arxiv.org/abs/2602.06508.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

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

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.00588.

Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models. Google DeepMind Technical Blog, 2025. URL https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Skywork AI Matrix-Game Team. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory. Technical report, 2026. URL https://github.com/SkyworkAI/Matrix-Game/blob/main/Matrix-Game-3/assets/pdf/report.pdf.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Zizhao Tong, Hongfeng Lai, Zeqing Wang, Zhaohu Xing, Kexu Cheng, Haoran Xu, Zhao Pu, Shangwen Zhu, Ruili Feng, Jian Zhao, Yan Zhang, Hao Tang, Yeying Jin, and Ling Shao. SCOPE: Simulating cross-game operations in playable environments for FPS world models. *arXiv preprint arXiv:2605.23345*, 2026. URL https://arxiv.org/abs/2605.23345.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.14837.

Team Wan, Ang Wang, Baole Ai, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. In *Proceedings of Robotics: Science and Systems*, 2026. URL https://arxiv.org/abs/2603.08546.

Junjin Xiao, Yandan Yang, Xinyuan Chang, Ronghan Chen, Feng Xiong, Mu Xu, Wei-Shi Zheng, and Qing Zhang. World-Env: Leveraging world model as a virtual environment for VLA post-training. *arXiv preprint arXiv:2509.24948*, 2025. URL https://arxiv.org/abs/2509.24948.

Zhexiao Xiong, Yizhi Song, Hao Kang, Qing Yan, Liming Jiang, Jenson Yang, Zhoujie Fu, Stathi Fotiadis, Angtian Wang, Zichuan Liu, Bo Liu, Yiding Yang, Xin Lu, and Nathan Jacobs. Actworld: From explorable to interactive world model via action-aware memory. *arXiv preprint arXiv:2606.17730*, 2026. URL https://arxiv.org/abs/2606.17730.

Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06114.

Jiahui Zhang, Ze Huang, Chun Gu, Zipei Ma, and Li Zhang. Reinforcing action policies by prophesying. *arXiv preprint arXiv:2511.20633*, 2025. URL https://arxiv.org/abs/2511.20633.

Zhilong Zhang, Haoxiang Ren, Yihao Sun, Yifei Sheng, Haonan Wang, Haoxin Lin, Zhichao Wu, Pierre-Luc Bacon, and Yang Yu. Towards practical world model-based reinforcement learning for vision-language-action models. *arXiv preprint arXiv:2603.20607*, 2026. URL https://arxiv.org/abs/2603.20607.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. WMPO: World model-based policy optimization for vision-language-action models. *arXiv preprint arXiv:2511.09515*, 2025. URL https://arxiv.org/abs/2511.09515.

Yanjiang Zhu, Jianglin Wu, Yuxuan Hu, Jiajun Wu, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. In *International Conference on Learning Representations*, 2026.

A World Simulator Architecture Details

Backbone configuration. Our world simulator is built by attaching action-conditioning modules to Wan2.2-TI2V-5B (Wan et al., 2025), a latent video diffusion transformer. The backbone operates on video latents produced by a 3D causal VAE with $4\times$ temporal and 16×16 spatial compression, so that a valid clip of T_{raw} raw frames yields $T_{\text{lat}} = (T_{\text{raw}} - 1)/4 + 1$ latent frames. The DiT backbone contains 30 transformer blocks with hidden dimension 3072 and patch size $[1, 2, 2]$. These pretrained components provide the visual prior for game-scene synthesis, while the added action-conditioning modules provide action controllability.

Action representation and typed encoder. Each native control $a_t = (b_t, m_t)$ is represented as a 66-dimensional vector. The first 64 channels encode simultaneous keyboard and mouse-button states as a multi-hot binary vector b_t, and the last 2 channels encode the normalized mouse displacement $m_t = (\Delta x, \Delta y)$. Mouse displacement is scaled separately along the two axes and clipped to $[-1, 1]$. To preserve the heterogeneous structure of native controls, the action encoder maps b_t and m_t through separate branches into a shared 256-dimensional token space. For each raw frame, the encoder forms a button token and a mouse token, adds learnable type and frame-position embeddings, and interleaves the tokens over time. A 2-layer, 4-head block-causal Transformer then processes the token sequence, allowing the button and mouse tokens of the same frame to attend to each other while preventing any token from attending to future-frame actions. The resulting tokens are pooled to the video-latent timeline, producing discrete and continuous action contexts aligned with the latent generation steps. The first latent position, which corresponds to the anchor frame, uses a learned null action token.

Residual action injection. The action contexts are injected into the pretrained DiT through residual fusion rather than cross-attention. In each selected DiT block, two independent fusion branches combine the hidden visual state with the discrete and continuous action contexts, respectively. Each branch normalizes the hidden state and the action context, concatenates them, applies an MLP, and scales the output by a zero-initialized learnable gate before adding it back to the visual stream. We apply action injection to the first 20 of the 30 DiT blocks. Because the gates start at zero, the model initially reproduces the pretrained backbone and gradually incorporates native-control information during training. At sink positions, the action residual is masked to zero so that the sink remains a purely visual reference.

Sink-anchored context implementation. The initial observation o_0 of each episode is VAE-encoded into a sink latent, which is concatenated along the latent time dimension of every generation chunk. The sink latent participates in DiT attention as a persistent visual reference and is removed before unpatchifying and decoding the predicted chunk. It is assigned the clean anchor timestep, receives no action residual, and is excluded from the video prediction loss. During training, chunk starts are sampled at varying temporal offsets from the same episode anchor. With T_{raw} raw frames per chunk and a maximum chunk offset $C_{\max}$, the sink-to-chunk distance thus ranges up to $C_{\max}(T_{\text{raw}} - 1)$ frames. This exposes the model to the long anchor-to-chunk distances encountered during autoregressive rollout while keeping the anchor fixed across chunks.

B World-Model Training and Rollout Details

Training examples and conditioning. World-model training uses frame-aligned gameplay clips paired with the native keyboard-and-mouse controls issued by the player. Each training example provides an initial observation o_0, a target future video segment, and the corresponding control sequence. The initial observation serves both as the image condition and as the sink anchor described in Appendix A, while the control sequence is converted into typed action contexts by the action encoder and aligned to the video-latent timeline. Initializing from the pretrained image-to-video checkpoint supplies the visual prior required for photorealistic AAA-game scenes, so adaptation can focus on learning control-conditioned dynamics rather than video synthesis from scratch.

Flow-matching adaptation. We first adapt the pretrained generator to the target game with a latent-space flow-matching objective (Liu et al., 2022; Lipman et al., 2023). Let $O = (o_0, \ldots, o_T)$

denote a training video segment, $A = (a_0, \ldots, a_{T-1})$ its aligned control sequence, and z_0 the clean video latent obtained by VAE encoding. For a sampled flow time $\sigma \in [0, 1]$ and Gaussian noise ϵ, we construct a perturbed latent z_σ by the standard flow-matching interpolation and train the DiT to predict the corresponding target velocity $v^\star$. With visual conditioning c_{vis} from the initial frame and sink anchor, and action conditioning c_{act} from the typed action contexts, the world-model loss is

$$\mathcal{L}_{\text{WM}}(\theta) = \mathbb{E}_{(O,A)\sim\mathcal{D},\,\sigma,\,\epsilon}\left[\left\|v_\theta(z_\sigma, \sigma, c_{\text{vis}}, c_{\text{act}}) - v^\star\right\|_2^2\right]. \tag{6}$$

This stage teaches the pretrained video prior to associate native controls with their visual consequences while preserving high-fidelity scene appearance, and yields the bidirectional teacher used for distillation.

Self-Forcing distillation. The bidirectional teacher is trained on clean video context, whereas closed-loop rollout conditions later chunks on histories that contain the model's own generated frames. We therefore distill the teacher into a block-causal 4-step student with Self-Forcing (Huang et al., 2025). During distillation, the student is unrolled autoregressively with a KV cache under the same chunk-wise interface used at inference, so that the latent frames of each generated chunk become context for subsequent chunks. The student is optimized with a distribution-matching objective against the teacher. This exposes the student to the self-generated histories encountered during policy rollout, mitigates exposure bias, and reduces each simulator query to 4 denoising steps per chunk.

Optimization and cross-game transfer. We train the *Black Myth: Wukong* simulator on 190 hours of gameplay at 832×480 resolution, jointly updating the DiT backbone and the action-conditioning modules with AdamW (weight decay 0.01). To preserve the pretrained prior while letting the new modules learn quickly, we use separate learning rates of 2×10^{-5} for the DiT backbone, 5×10^{-5} for the action encoder and fusion branches, and 10^{-3} for the scalar gates, with a linear warmup followed by a constant schedule. Gradients are clipped to a norm of 0.5, and an exponential moving average of the weights with decay 0.9999 is used for evaluation. Training runs for 4000 steps with a global batch size of 128 on 8 NVIDIA GPUs. For *Honor of Kings: World* (NGR), we initialize from the converged *Black Myth: Wukong* checkpoint and fine-tune on 3 hours of NGR video–action data with the same optimization settings and architecture. This protocol measures whether the learned native-control correspondence can be reused in a new visual domain under a limited data budget.

Autoregressive rollout protocol. At inference, the simulator is queried chunk by chunk. The first chunk is conditioned on o_0, the sink, and the first control chunk A_0. Each subsequent chunk is conditioned on the same sink latent and the latent frames of the 6 most recent generated chunks, with the KV cache maintained within each rollout session. The action sequence is sliced by chunk start so that each chunk receives controls aligned with its temporal range. When chunks are decoded into a continuous trajectory, overlapping boundary frames are removed to avoid duplicated observations. Resetting an episode amounts to clearing the rollout state and encoding a new o_0 as both the first-chunk condition and the sink. This protocol defines the simulator interface used for policy training, in which the policy emits a control chunk, the simulator returns the next observations, and the loop repeats without querying the real game. Fig. 5 shows two such rollouts from the same initial observation, in which different sampled control chunks lead to a successful and a failed trajectory.

C World-Model Evaluation Metrics

We report six metrics for single-chunk world-model evaluation. Let $V = (I_0, \ldots, I_T)$ denote a generated video, $V^\star$ the paired ground-truth video when available, and $F_t \in \mathbb{R}^{H\times W\times 2}$ the optical flow from I_t to I_{t+1} estimated by RAFT. For paired evaluation, $F_t^\star$ denotes the corresponding ground-truth-video flow. All metrics are averaged over frames and spatial locations, and then over evaluation examples.

Flow Direction Score (fds). fds measures directional agreement between predicted and ground-truth motion:

$$\text{fds} = \mathbb{E}_{t,x,y}\left[\frac{\langle F_t(x,y), F_t^\star(x,y)\rangle}{\|F_t(x,y)\|_2\,\|F_t^\star(x,y)\|_2 + \epsilon}\right]. \tag{7}$$



Figure 5: Two simulator rollouts for the task *"Pass through the pavilion ahead"*, both starting from the same initial observation. Top: a successful trajectory in which the agent walks straight through the pavilion. Bottom: a failed trajectory in which the agent veers right and leaves the pavilion without passing through it.

It is high when the simulator predicts motion in the same direction as the real video, even if the magnitude differs.

End-Point Error (epe). epe measures the L2 discrepancy between predicted and ground-truth optical flow:

$$\text{epe} = \mathbb{E}_{t,x,y}\left[\|F_t(x,y) - F_t^\star(x,y)\|_2\right]. \tag{8}$$

Unlike fds, epe penalizes both direction and magnitude errors.

Counterfactual Distance (cfd). cfd measures whether the simulator responds to different native controls from the same visual state. Given the same initial frame and two distinct action trajectories A and A', the simulator generates videos V_A and $V_{A'}$. We compute

$$\text{cfd} = \text{DreamSim}(V_A, V_{A'}), \tag{9}$$

using the same frame-sampling protocol for both videos. Higher cfd indicates that different controls produce more distinguishable futures.

Flow Score (flow_score). flow_score measures the average magnitude of generated motion:

$$\text{flow_score} = \mathbb{E}_{t,x,y}\left[\|F_t(x,y)\|_2\right]. \tag{10}$$

It complements cfd by verifying that the generated sequence contains nontrivial motion rather than a static prediction.

Dynamic Degree (dyn_deg). dyn_deg measures the spatial extent of generated motion as the fraction of pixels whose flow magnitude exceeds a threshold τ:

$$\text{dyn_deg} = \mathbb{E}_{t,x,y}\left[\mathbf{1}\{\|F_t(x,y)\|_2 > \tau\}\right]. \tag{11}$$

In our implementation, $\tau = 1.5$ pixels.

Motion Smoothness (mot_smooth). mot_smooth measures temporal regularity of the generated motion field. Let $A_t = F_{t+1} - F_t$ be the optical-flow acceleration between adjacent flow fields. We compute

$$\text{mot_smooth} = \frac{1}{1 + \mathbb{E}_{t,x,y}\left[\|A_t(x,y)\|_2\right]}. \tag{12}$$

Higher values indicate smoother temporal evolution of motion. Because a static video can also obtain a high smoothness score, we interpret mot_smooth jointly with flow_score and dyn_deg.

D REINFORCEMENT LEARNING DETAILS FOR *Black Myth: Wukong*

D.1 TRAJECTORY-LEVEL REWARD DESIGN

For each simulated rollout, Gemini-3.6-Flash evaluates the task instruction together with the temporally ordered actor-decision frames and assigns an integer score $s_i \in \{1, \ldots, 5\}$. The rubric distinguishes complete failure, initial intent without meaningful progress, correct intent with substantial errors, successful completion with minor errors, and precise and fluent completion.

The judge applies task-specific criteria. Continuous-navigation trajectories should follow the specified route without abrupt turns, prolonged obstruction, or excessive deviation; traversal tasks require passage through the designated objects or structures; entity-directed navigation requires termination near the target with the target visible; and compound tasks require all sub-tasks to be completed in the specified order.

In addition to the judge score, we introduce two lightweight behavioral penalties. Let u_i indicate whether SHIFT is used in trajectory i, and let v_i indicate whether the rollout terminates at the step or duration limit rather than through an actor-issued STOP. The final reward is

$$r_i = \max\left(0, \frac{s_i}{5} - 0.1u_i - 0.1v_i\right). \tag{13}$$

The SHIFT penalty discourages sprinting and overshooting, while the termination penalty encourages the policy to recognize task completion and stop autonomously instead of relying on the rollout budget. The two penalties are additive, and the final reward is clipped at zero.

Trajectories reaching the step or duration limit are still evaluated by the vision–language judge because they may contain meaningful task progress; they only receive the additional termination penalty in Eq. 13. By contrast, trajectories terminated through an actor-issued STOP receive no termination penalty, irrespective of their judge score. Trajectories affected by infrastructure failures, empty records, unreadable observations, or more than 50% missing frames are excluded from direct scoring.

Let $\mathcal{V}$ denote the set of valid trajectories within a GRPO group. An excluded trajectory is assigned the mean reward of the valid group members:

$$\widehat{R}_i = \begin{cases} r_i, & i \in \mathcal{V}, \\ \dfrac{1}{|\mathcal{V}|}\displaystyle\sum_{j \in \mathcal{V}} r_j, & i \notin \mathcal{V},\ |\mathcal{V}| > 0, \\ 0, & |\mathcal{V}| = 0. \end{cases} \tag{14}$$

Under group-relative normalization, this imputation gives excluded trajectories a zero-centered advantage, preventing simulator or evaluation failures from being interpreted as poor policy behavior.

D.2 VISION–LANGUAGE JUDGE PROMPT FOR *Black Myth: Wukong*

Prompt. We use Gemini-3.6-Flash as a vision–language judge. It receives the task instruction and the temporally ordered rollout frames. Since the implementation uses a Chinese prompt, we provide its English translation below.

> You are an expert in evaluating the instruction-following performance of game agents in *Black Myth: Wukong*.
>
> Task instruction:
>
> "{instruction}"
>
> You are given an ordered sequence of {n} images showing the agent executing this instruction. The first image is the initial state, and the remaining images form a temporally ordered rollout. Evaluate the entire trajectory according to the following criteria.
>
> General: The agent must follow the instruction accurately.
>
> Continuous navigation: - The character should follow the requested route or direction and steer gradually when the route changes. - Sudden large turns are not allowed. - Reversing, spinning, or remaining stuck

should not persist for more than five consecutive images. The character should navigate around temporary obstacles. - The character should not leave the requested scene or travel excessively far from the initial scene.

Traversal: - The character must pass strictly between the specified objects. - It should avoid prolonged obstruction and should not travel beyond the relevant scene.

Entity-directed navigation: - The character must finish near the target, which should remain visible in the final image. - It should avoid remaining stuck for more than five consecutive images.

Compound tasks: - All constituent actions must be completed in the order specified by the instruction.

Assign exactly one integer score: 1 – No task-related behavior or intent; complete failure. 2 – Initial task-related intent, but no substantive progress. 3 – Correct intent with substantial hesitation or frequent mistakes. Also use 3 if the task is completed but the agent continues acting unnecessarily. 4 – The task is broadly completed, with only minor non-blocking errors. 5 – Immediate, precise, and fluent completion without errors or hesitation.

Briefly analyze the trajectory, and output the score on the final line using exactly the following format, with nothing after it:

SCORE: `<an integer from 1 to 5>`

Design rationale. The judge prompt follows a general, reusable format that maps an instruction and an ordered sequence of rollout images to a scalar reward. Because it relies only on visual observations, it does not require privileged game states, internal engine variables, or task-specific instrumentation. The same reward interface can therefore be instantiated in both open-source and closed-source games, provided that rendered observations are available. Within this general format, we specify visually verifiable criteria for multiple instruction-following task families, including continuous navigation, traversal, entity-directed navigation, and compound tasks. This combination of a shared evaluation structure and task-family-specific criteria enables the judge to assess heterogeneous tasks while reducing ambiguity about successful execution. Evaluating the complete ordered trajectory, rather than only its final frame, also allows the judge to identify route deviation, prolonged obstruction, incorrect action ordering, hesitation, and unnecessary actions after task completion. Finally, the multiple-level rubric distinguishes task intent, partial progress, successful completion, and execution quality. It thus provides a finer-grained trajectory-level learning signal than binary success evaluation, assigning informative rewards even when a rollout does not fully complete the instruction. The constrained output format allows these judgments to be parsed directly and maintains a consistent reward interface across tasks and rollouts.

D.3 Training Task Set

The RL training set contains 48 instruction-following tasks from *Black Myth: Wukong*. Following the taxonomy introduced in the main text, we group these tasks into four categories: 11 continuous-navigation tasks, 7 traversal tasks, 18 entity-directed navigation tasks, and 12 compound tasks. Continuous-navigation tasks require the agent to follow terrain structures such as paths, streams, stairways, and wooden walkways; traversal tasks require it to pass through designated structures or regions; entity-directed tasks require it to approach visually identifiable landmarks or objects; and compound tasks combine multiple spatial actions that must be executed in the specified order. This distribution covers all task families considered in our instruction-following setting and avoids concentrating training on a single type of behavior.

Each training example consists of an initial observation, a natural-language instruction, a rollout step limit, a rollout duration limit, and a unique session identifier. In the current configuration, each rollout is limited to 20 agent steps and a maximum duration of 3,600 seconds. The initial observations are collected from most regions of the game map used in our experiments and cover diverse terrain, routes, structures, and landmarks, including mountain paths, streams, bridges, temples, gates, vegetation, and other interactive scenes. The resulting dataset combines broad spatial coverage with complete task-type coverage and a balanced mixture of atomic and compound behaviors, providing varied initial conditions and behavioral objectives for on-policy training.

Table 4: Distribution of RL training tasks for *Black Myth: Wukong*.

Task category	Number	Proportion
Continuous navigation	11	22.9%
Traversal	7	14.6%
Entity-directed navigation	18	37.5%
Compound	12	25.0%
Total	48	100.0%

D.4 Evaluation Task Set

Task Set. Our evaluation set contains 14 instruction-following tasks in *Black Myth: Wukong*. Following the task taxonomy introduced in the main text, we categorize them as continuous navigation, entity-directed navigation, traversal, or compound tasks according to their primary behavioral requirements. The tasks cover diverse landmarks and terrain, including mountain paths, forest trails, streams, bridges, pillars, shrines, and temples.

Each task is defined by a natural-language instruction and a fixed initial game observation. The evaluation configuration applies rule-based checks to all 14 tasks and additionally enables an image-based judge for five tasks whose completion depends on visually identifying a target landmark. Table 5 lists the complete evaluation set.

Evaluation criteria. Success is determined according to the task category. For continuous-navigation tasks, the agent must travel a sufficient distance along the specified route without visibly turning back, becoming stuck against obstacles, or deviating from the route. For entity-directed navigation tasks, the agent must move toward the specified target and terminate in its vicinity. Traversal tasks require the agent to move from one side of the designated object or structure to the other through the specified passage. For compound tasks, the agent must successfully complete both constituent sub-tasks in the order given by the instruction; a trial is considered successful only if both sub-tasks are completed.

We conduct 10 independent trials for each task and report both per-task and overall success rates. Given a collection of evaluation trials $\mathcal{E}$, its success rate is computed as

$$\mathrm{SR}(\mathcal{E}) = \frac{\sum_{i \in \mathcal{E}} \mathbb{I}[\mathrm{Success}_i]}{|\mathcal{E}|}, \tag{15}$$

where $\mathbb{I}[\mathrm{Success}_i]$ equals one if trial i satisfies the corresponding task-specific criteria and zero otherwise. Thus, the per-task success rate is computed over 10 trials, while the overall success rate is computed over all 140 trials across the 14 evaluation tasks.

D.5 Experimental Setup

We fine-tune the VLA policy with GRPO in a fully imagined closed loop, without interacting with the real game during RL. We remove the critic entirely and initialize both the trainable actor and the frozen reference policy from the supervised fine-tuning checkpoint. For each prompt, we sample a group of $G = 8$ trajectories and normalize each trajectory's return using the mean and standard deviation within its group. The policy is then optimized with the clipped surrogate objective using a clip ratio of $\epsilon = 0.2$ for two PPO epochs per rollout batch. Each rollout batch contains four prompts, yielding $4 \times 8 = 32$ trajectories per policy update. The mini-batch size is four prompts, the per-GPU micro-batch size is one, and the learning rate is 1×10^{-6}.

We regularize the policy toward the frozen SFT initialization using an auxiliary KL loss with a coefficient 10^{-3} and the low-variance KL estimator. The KL term is applied directly to the policy loss rather than being incorporated into the trajectory reward, and no entropy bonus is used. Training is performed in BF-16 under FSDP with gradient checkpointing, padding removal, and FlashAttention-2.

Rollout generation is served by an asynchronous vLLM engine with tensor parallelism of two. The rollout engine is co-located with the trainer using a hybrid engine and is put to sleep during gradient

updates to release its temporary GPU allocations. We allocate a GPU memory fraction of 0.6 to the rollout engine, load policy weights directly from the training side, and enable prefix caching. Trajectories are sampled with temperature 1.0 and top-$p = 1.0$, preserving sufficient within-group diversity for reward normalization. A rollout terminates when the policy emits an explicit STOP token, reaches 20 decision steps, or exceeds a 700-second wall-clock limit.

At each decision step, the policy conditions on a sliding context window of 20 turns of 512×512 visual observations and generates at most 512 tokens. This output budget accommodates the chain-of-thought preamble, which is retained in the training target. The maximum prompt and response lengths are both 16,384 tokens. For the policy update, rollout log-probability recomputation, and reference-policy forward pass, the per-GPU token budget is set to 49,152, ensuring that complete trajectories can be processed without splitting or dropping them. We train for 40 epochs, perform validation every five optimizer steps, and save a checkpoint every four optimizer steps.

Table 5: Evaluation tasks and their initial observations. The task categories follow the taxonomy defined in the main text.

Task category	Instruction	Initial observation
Continuous navigation	Move straight ahead along the current mountain path between the rocks.	
	Run forward along the forest dirt path.	
	Continue walking forward along the shallow stream.	
Entity-directed navigation	Move forward and approach the shrine ahead.	
	Run straight ahead to the wooden gate.	
	Move toward the glowing *lingzhi* mushroom ahead-left.	
	Approach the stone pillar ahead-left.	
Traversal	Move straight ahead and cross the wooden bridge.	

Continued on the next page

Table 5 continued

Task category	Instruction	Initial observation
Compound	Move toward and approach the temple ahead-right.	
	Turn around and cross the wooden bridge.	
	Turn toward the front-right and continue along the current path.	
	Turn toward the front-left and pass between the two stone pillars.	
	Jump down the cliff ahead-left, and then proceed straight along the stream.	
	Cross the muddy ground ahead, and then move toward the temple to the front-left.	

E Reinforcement Learning Details for *Honor of Kings: World*

E.1 Reward Design

The *Honor of Kings: World* follows the same general trajectory-level reward framework as the *Black Myth: Wukong*. A vision-language judge evaluates the instruction together with the ordered rollout frames, and invalid trajectories are handled at the GRPO-group level as described above. We therefore focus here on the environment-specific differences.

Since all training tasks in this world are entity-directed, the reward prompt is specialized to target identification, movement in the correct direction, arrival within the target vicinity, final target visibility, and timely stopping. Gemini-3.8-Flash assigns a score $s_i \in \{1, 3, 5\}$ using a hierarchical rubric. A score of 1 indicates that at least one main requirement is not satisfied; 3 indicates that all main requirements are satisfied but some execution-quality requirements are violated; and 5 indicates that both the main and additional requirements are fully satisfied.

As in the *Black Myth: Wukong* reward, we apply penalties for sprinting and for failing to terminate autonomously. Let u_i indicate whether SHIFT is used in trajectory i, and let v_i indicate whether the rollout reaches its step or duration limit rather than terminating through an actor-issued STOP. The reward is

$$r_i = \frac{s_i}{5} - 0.2u_i - 0.2v_i, \qquad s_i \in \{1, 3, 5\}. \tag{16}$$

Compared with the *Black Myth: Wukong* setting, both behavioral penalty coefficients are increased from 0.1 to 0.2. The stronger SHIFT penalty discourages sprinting and overshooting the target, while the stronger termination penalty encourages the policy to recognize arrival and emit STOP instead of relying on the rollout budget. The penalties are additive, and the resulting reward is not clipped at zero. This design retains the shared visual reward interface while adapting the scoring resolution and behavioral regularization to the target-approach and stopping requirements of the *Honor of Kings: World* tasks.

E.2 Reward Prompt

Reward prompt. For the *Honor of Kings: World*, we use a reward prompt specialized for entity-directed navigation. Gemini-3.8-Flash receives the instruction and the ordered trajectory frames, including the initial observation, and evaluates whether the agent approaches and stops near the

specified target. Since the implementation uses a Chinese prompt, we provide its English translation below.

> You are an expert in evaluating the instruction-following performance of a game AI agent.
>
> Task instruction:
>
> "{instruction}"
>
> You are given an ordered sequence of {n} images showing the agent executing this instruction. The first image is the initial state, and the remaining images form a temporally ordered rollout. Evaluate the complete trajectory according to the following criteria.
>
> Main requirements: - The agent should not rotate the camera substantially or for a prolonged period at the beginning of or during movement. - Identify the target and the required movement direction from the instruction and initial observation. The agent must move toward the target rather than in an incorrect direction. - The agent must not stop before reaching the vicinity of the target.
>
> Additional requirements: - After reaching the target, the agent should not continue moving for more than four images or travel far beyond the target region. - The target must remain visible in the final image, and the agent must stop near it. - Reversing, spinning, or remaining stuck against an obstacle should not persist for more than five consecutive images. The agent should navigate around temporary obstacles.
>
> Briefly assess each main and additional requirement, and assign exactly one of the following scores:
>
> 1 – The execution does not fully satisfy the main requirements. 3 – The execution fully satisfies the main requirements but does not fully satisfy the additional requirements. 5 – The execution fully satisfies both the main and additional requirements.
>
> Output the score on the final line using exactly the following format, with nothing after it:
>
> SCORE: <1, 3, or 5>

The prompt separates task completion from execution quality. The main requirements determine whether the agent identifies and reaches the correct target, whereas the additional requirements assess camera stability, timely stopping, final target visibility, and robustness to obstacles. This hierarchical rubric assigns partial credit to trajectories that reach the target but exhibit undesirable behavior, providing a more informative learning signal than a binary success indicator.

E.3 Training task set.

The training set contains 20 entity-directed navigation tasks. Each example consists of an initial observation, a natural-language instruction, a maximum rollout length, a duration limit, and a unique session identifier. All examples use a limit of 20 agent steps and a maximum duration of 3,600 seconds:

$$\mathcal{D}_{\text{HoK}} = \left\{ (o_0^{(i)},\, \ell^{(i)},\, T_{\max}^{(i)},\, D_{\max}^{(i)},\, q^{(i)}) \right\}_{i=1}^{20}, \tag{17}$$

where $o_0^{(i)}$ is the initial image, $\ell^{(i)}$ is the instruction, $T_{\max}^{(i)} = 20$ is the step limit, $D_{\max}^{(i)} = 3600$ seconds are the duration limit, and $q^{(i)}$ is the session identifier.

The tasks require the agent to navigate toward visually identifiable targets under diverse relative-direction descriptions, such as *ahead*, *ahead-left*, and *ahead-right*. The targets include stone formations, colored rocks, gates, buildings, crates and chests, pillars, tables, steles, trees, and mechanisms. This diversity varies both the visual appearance of the goal and its spatial relation to the agent while retaining a consistent entity-directed objective. It therefore provides multiple initial conditions for learning target grounding, directional movement, proximity judgment, and timely stopping.

E.4 Evaluation Tasks in the *Honor of Kings: World*

The *Honor of Kings: World* evaluation set contains 10 entity-directed navigation tasks. Each task is specified by a natural-language instruction and an initial game observation. The agent is required to identify the referenced target from the initial observation, move in the specified direction, and terminate in the vicinity of the target. Several tasks further require the agent to traverse stairs before approaching the target, thereby testing whether it can follow an intermediate spatial constraint while preserving the final entity-directed objective.

Table 6: Evaluation tasks and initial observations in the *Honor of Kings: World*. The first five tasks are shown on the left and the remaining five on the right.

Instruction	Initial observation	Instruction	Initial observation
Approach the glowing yellow stone on the right front.		Approach the yellow rock ahead-right.	
Walk to the blue mechanism ahead.		Descend the stairs and approach the table ahead-right.	
Step up onto the stairs on the right front and approach the gate.		Approach the golden mechanism ahead-right.	
Ascend the stairs and approach the blue device.		Move ahead-right to the yellow gate.	
Approach the blue chest ahead-left.		Move ahead-left, ascend the stairs, and approach the yellow rock.	

The test set covers diverse visually identifiable targets, including colored rocks, mechanisms, gates, chests, tables, and other devices. It also varies the relative target direction and elevation, including targets located ahead, ahead-left, ahead-right, above stairs, and below stairs. These variations jointly evaluate visual target grounding, directional navigation, terrain-aware movement, proximity judgment, and timely stopping. The complete evaluation set is shown in Table 6.

F Experimental Results

F.1 *Black Myth: Wukong*

Fig. 6 reports the per-task and category-level real-game success rates of WukongAgent-SFT and WukongAgent-RL. Both models are evaluated on 14 tasks, with 10 independent trials for each task.

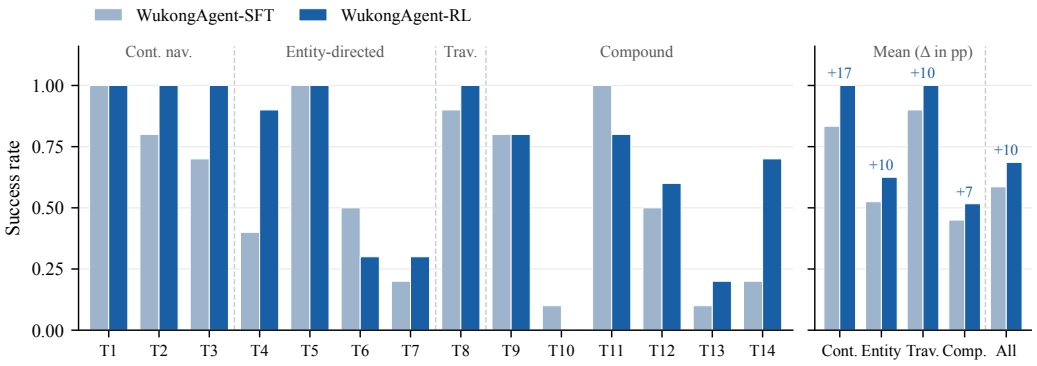

Figure 6: Per-task real-game success rates of WukongAgent-SFT and WukongAgent-RL (left), and category-level means with absolute gains in percentage points (right). Each task is evaluated over 10 trials.

Overall, RL increases the average success rate from 58.6% to 68.6%, an absolute gain of 10.0 percentage points. Performance improves on 8 of the 14 tasks, remains unchanged on 3 tasks, and decreases on 3 tasks. The largest improvements occur on Task 4 (approaching the shrine) and Task 14 (crossing muddy terrain and then approaching the temple), both of which improve by 50

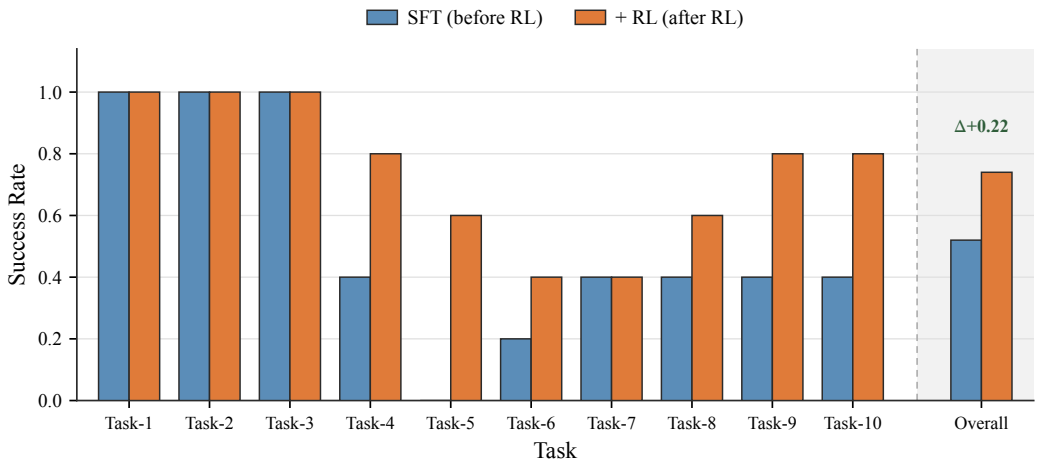

Figure 7: Per-task and overall success rates before and after reinforcement learning in the *Honor of Kings: World* . Each task is evaluated over multiple independent trials. RL improves the overall success rate from 52% to 74%, corresponding to an absolute gain of 22 percentage points.

percentage points. Tasks 2, 3, 7, 8, 12, and 13 also benefit from RL, whereas Tasks 6, 10, and 11 exhibit regressions.

At the category level, RL improves all four task families. Continuous navigation increases from 83.3% to 100%, entity-directed navigation from 52.5% to 62.5%, traversal from 90% to 100%, and compound tasks from 45.0% to 51.7%. The gains in route following, target approach, and multi-stage navigation indicate that optimization in the learned world model transfers to real-game execution. Compound tasks remain the most challenging, particularly when they require substantial reorientation before traversal. Nevertheless, the consistent category-level gains and the 10-point overall improvement demonstrate that on-policy RL provides an effective learning signal beyond supervised fine-tuning.

F.2 *Honor of Kings: World*

We evaluate the effect of reinforcement learning in the *Honor of Kings: World* by comparing the policy before RL (SFT) with the policy after on-policy training (SFT+RL). The evaluation uses the 10 tasks described in Appendix E.4. Each task is evaluated over multiple independent trials with different models. We report both per-task and overall success rates.

As shown in Figure 7, RL increases the overall success rate from 52% to 74%, corresponding to an absolute improvement of 22 percentage points. Performance improves on six of the ten tasks and does not decrease on any task. The three tasks already solved reliably by the SFT policy (Tasks 1–3) retain a 100% success rate after RL, indicating that on-policy training preserves the target-approach capabilities acquired during supervised fine-tuning.

The largest improvement occurs on Task 5, which requires approaching the blue chest ahead-left, where the success rate increases from 0% to 60%. Tasks 4, 9, and 10 improve from 40% to 80%. Tasks 4 and 10 require the agent to ascend stairs before approaching the target, suggesting that RL is particularly beneficial when entity-directed navigation involves additional spatial constraints. More moderate gains are observed on Task 6 ($20\% \rightarrow 40\%$) and Task 8 ($40\% \rightarrow 60\%$).

Task 7 remains at 40%, showing that descending stairs and subsequently approaching the target remains challenging. Nevertheless, the absence of degradation across all ten tasks, together with improvements on both direct target-approach tasks and tasks involving stairs, demonstrates that the trajectory-level visual reward provides an effective learning signal beyond supervised imitation. After RL, eight of the ten tasks achieve success rates of at least 60%, compared with only three tasks before RL.