PLAYER-1: LEARNING TO PLAY FROM SYNTHETIC EXPERIENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Computer-use agents have demonstrated strong capabilities in interpreting screens and executing keyboard and mouse actions. However, games, with their visually rich worlds and diverse mechanics, remain challenging and offer a demanding next step toward general digital control. Current game-agent learning pipelines often rely on large-scale demonstrations with limited coverage of dynamic game states, while scalable environments for interactive training remain difficult to obtain. We introduce **PLAYER-1**, a post-training framework for learning game control beyond demonstrations through synthetic experience. We extend the computer-use action space with relative pointer motion, sustained inputs, and composite actions. For supervised initialization, we fine-tune computer-use models on 6,044 curated gameplay trajectories that pair recorded actions with task instructions and visual reasoning. We construct **PLAYER-1-GYM**, sixteen synthetic 2D and 3D environments designed around control demands shared across games. Programmatically verifiable rewards and randomized observations and control dynamics support multi-environment reinforcement learning with PPO. The resulting Player-1-9B and Player-1-35B-A3B improve MineDojo-300 success over their SFT policies by **27.8 and 17.7 percentage points**, respectively. Player-1-9B reaches **38.33%** success compared with **12.63%** for the strongest evaluated baseline, with further gains over SFT on ViZDoom-10 and GameWorld. Further analyses show that broader synthetic environment coverage improves transfer, while the use of composite actions that combine multiple keyboard and mouse inputs increases during RL. We demonstrate that learning from synthetic experience substantially improves game control beyond supervised initialization, with gains that generalize beyond the RL training suite.

1 INTRODUCTION

Foundation models have developed broad capabilities in visual understanding, reasoning, and instruction following (OpenAI, 2024; Anthropic, 2025; Gemini Team, 2025). Computer-use agents bring these capabilities to digital interfaces, interpreting screens and executing keyboard and mouse actions (Wang et al., 2025a; Bai et al., 2025; Lu et al., 2026). Games offer a demanding next step toward general digital control: their visually complex environments and diverse mechanics require precise, sustained, and concurrent control under dynamic and rapidly changing visual feedback (Luo et al., 2026a).

Recent game agents learn from large-scale gameplay demonstrations and interactive experience (Wang et al., 2025b; Magne et al., 2026; SIMA Team, 2025; Shi et al., 2026). Yet demonstrations cover only a limited portion of the states an agent may encounter in open-ended tasks, while scalable environments for interactive training remain difficult to obtain. Despite differences in appearance and rules, we find that many games share recurring control demands such as precise aiming, sustained movement, and coordinated inputs. This motivates our central question: ***Can synthetic experience advance game-control capabilities beyond supervised initialization, with gains that generalize to games outside the RL training suite?***

Three challenges shape this approach. First, gameplay demonstrations recorded with heterogeneous input conventions and temporal resolutions must be converted into a shared action space, and paired with task instructions and visual reasoning that reflect player's intent. Second, learning beyond

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.

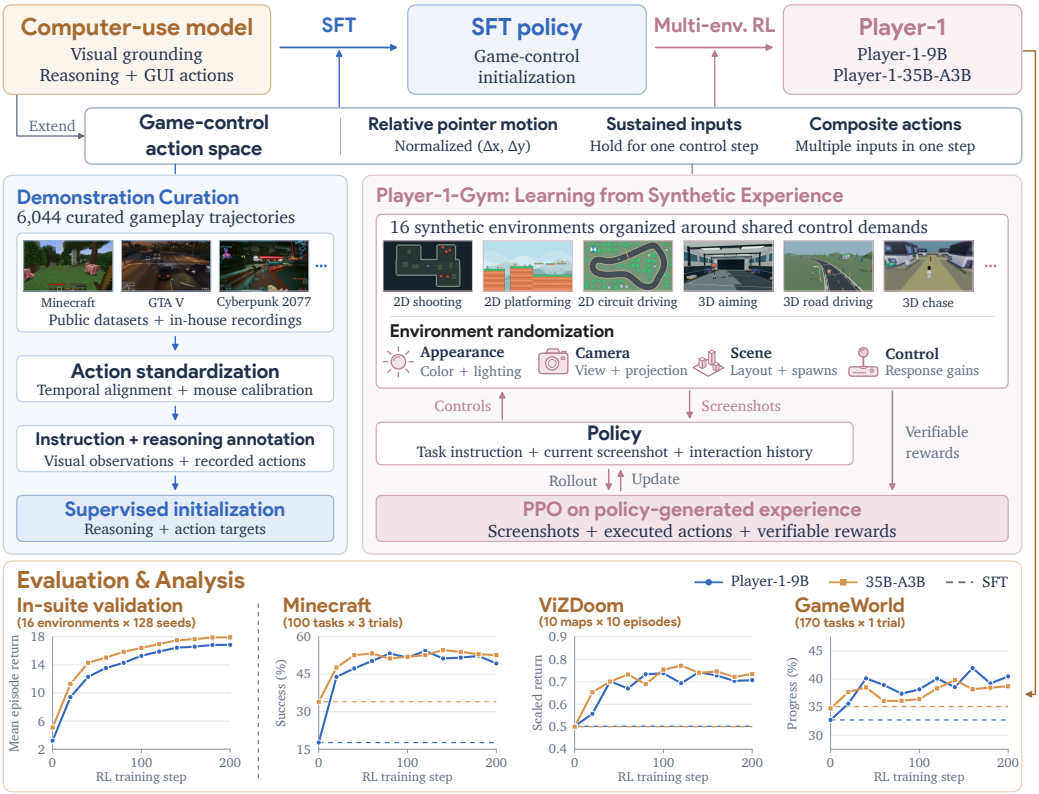


Figure 1: Overview of **PLAYER-1**: learning game control beyond demonstrations. A shared game-control action space connects supervised initialization from gameplay demonstrations with multi-environment RL in **PLAYER-1-GYM**. Learning from synthetic experience with verifiable rewards advances control capabilities beyond SFT, with gains that generalize beyond the RL training suite.

demonstrations requires feedback on states reached by the policy's own actions, but implementing programmatic rewards across diverse commercial games is difficult to scale. Third, experience from environments with different control demands, dynamics, and reward scales must be incorporated into a shared policy that generalizes beyond the training suite. These challenges motivate our supervised initialization from curated demonstrations, followed by learning in controllable synthetic environments with verifiable rewards.

We present **PLAYER-1**, a post-training framework for learning game control beyond demonstrations through synthetic experience, as illustrated in Figure 1. We extend the computer-use action space with relative pointer motion, sustained inputs, and composite actions. Curated gameplay demonstrations pair recorded actions with task instructions and visual reasoning to initialize game control. **PLAYER-1-GYM** then provides sixteen synthetic 2D and 3D environments designed around recurring control demands across aiming, combat, driving, navigation, and UI interaction. Programmatic game-state rewards and randomization of observations and control dynamics support multi-environment reinforcement learning with verifiable rewards (RLVR), yielding Player-1-9B and Player-1-35B-A3B. Table 1 summarizes the differences and focuses of representative game-agent learning pipelines.

Our results demonstrate that learning from synthetic experience improves game-control capabilities, with gains extending beyond the RL training suite. On MineDojo-300, Player-1-9B improves from 10.54% success after SFT to 38.33%, exceeding the strongest evaluated external baseline at 12.63% under our evaluation protocol. The same policy also improves on ViZDoom-10 and GameWorld, both outside the RL training suite. We summarize our main contributions as follows:

- We present **PLAYER-1**, a post-training framework that builds on supervised initialization to learn transferable game-control capabilities from synthetic experience. A shared action space supports relative pointer motion, sustained inputs, and composite actions, while 6,044 curated gameplay trajectories provide supervised initialization for subsequent interactive learning.

Table 1: Comparison of representative game-agent learning pipelines, including supervision, RL environments, evaluation scope, and model-weight availability.

Method	Learning focus	Supervision	RL environments	Evaluation	Weights
Game-TARS	Large-scale pretraining	Reasoning + action	—	Held-out games	✗
SIMA 2	Embodied assistance	Reasoning + action	Games + world model	Held-out games	✗
NitroGen	System-1 generalist	Action	—	Games + adaptation	✓
Odysseus	Long-horizon RL	Reasoning + action	5 Mario levels	Mario games	✓
PLAYER-1	**Beyond demonstrations**	Reasoning + action	**16 synthetic envs**	**RL held-out games**	✓ (with paper)

- We introduce **PLAYER-1-GYM**, sixteen synthetic 2D and 3D environments designed around recurring control demands across games. Programmatically verifiable rewards, together with randomized observations and control dynamics, provide diverse experience for multi-environment reinforcement learning beyond demonstrations.
- We demonstrate that **learning from synthetic experience advances game-control capabilities beyond supervised initialization**: Player-1-9B and Player-1-35B-A3B gain 27.8 and 17.7 percentage points in MineDojo-300 success, respectively, with further gains on ViZDoom and GameWorld outside the RL training suite. Further analyses show improved transfer with larger training suites.

2 RELATED WORK

Generalist and Computer-Use Agents. Foundation models provide broad language, reasoning, and multimodal capabilities that underpin general-purpose agents (OpenAI, 2024; Anthropic, 2025; Gemini Team, 2025; Yang et al., 2025). Vision-language models (VLMs) connect visual perception with language, supporting scene understanding and spatial grounding (Bai et al., 2025). Computer-use agents (CUAs) specialize these capabilities for screenshot-conditioned interaction, using keyboard and mouse actions to complete multi-step tasks (Hu et al., 2024; Qin et al., 2025; Wang et al., 2025a; Lu et al., 2026).

Game Agents, Development, and Environments. Game-agent research spans planning-based systems (Tan et al., 2025b), learned visual policies and vision–language–action (VLA) models (SIMA Team, 2024; Tan et al., 2025a; Li et al., 2025; Wang et al., 2025b; Magne et al., 2026; Yue et al., 2026), and joint video–action models (Guo et al., 2026). Reinforcement learning further supports self-improvement and long-horizon control (SIMA Team, 2025; Shi et al., 2026). Most recently, Jev demonstrates low-latency game control through a text-only interface, relying on structured game-state representations rather than screenshot observations (Almeida, 2026). Complementary benchmarks evaluate gameplay (Ouyang et al., 2026) and agentic game development (Chi et al., 2026; Luo et al., 2026b). Code world models synthesized from rules and trajectories support planning (Lehrach et al., 2026), while game engine development provides trajectory data for world-model training (Zhou et al., 2026), connecting game creation to learning (Luo et al., 2026a).

Agent Post-Training with Verifiable Rewards. Post-training with verifiable rewards enables agents to learn from the outcomes of their own actions. Environment-synthesis approaches expand training coverage by constructing interactive tasks with verifiable rewards (Dong et al., 2026; Wang et al., 2026a). Adaptive approaches learn which environments to generate or jointly improve the policy, environment, and reward model (Liu et al., 2026; Wang et al., 2026b). In games, SIMA 2 combines supervised initialization with online RLVR and further demonstrates self-improvement transfer across Genie 3 environments (SIMA Team, 2025). **PLAYER-1** combines supervised initialization with learning from synthetic experience in environments organized around shared control demands, demonstrating game-control gains that generalize beyond the RL training suite.

3 METHOD

PLAYER-1 connects supervised initialization with learning from synthetic experience through a shared game-control action space. We first extend computer-use actions to support dynamic game

control, and initialize the policy using curated gameplay demonstrations. We then optimize the policy across **PLAYER-1-GYM**'s synthetic environments using PPO with verifiable rewards.

3.1 EXTENDING THE COMPUTER-USE ACTION SPACE

We extend the computer-use action space for dynamic game control while retaining screenshot-conditioned decision making. At turn t, the policy receives task instruction q, screenshot I_t, and interaction history $C_H(h_{<t})$ (see Section 3.2):

$$(z_t, \hat{a}_t) \sim \pi_\theta(\cdot \mid q, I_t, C_H(h_{<t})).$$

Here z_t is reasoning and $\hat{a}_t$ the proposed action. The game-control action space extends the base model's computer-use actions with relative pointer motion, sustained inputs, and composite actions.

Relative pointer motion. For the mouse actions, CUAs typically use normalized absolute coordinates to select visible targets by specifying their on-screen positions. In game-play scenarios, to control the camera without a visible pointer, we extend this action space with relative displacements:

$$\mathcal{A}_{\text{pointer}} = \underbrace{\{\text{absolute}(x, y)\}}_{\text{Retained CUA coordinates}} \cup \underbrace{\{\mathbf{relative}(\boldsymbol{\Delta x}, \boldsymbol{\Delta y})\}}_{\textbf{Relative-motion extension}}.$$

Absolute coordinates satisfy $x, y \in [0, 1000]$, while relative displacements $\Delta x, \Delta y \in [-1000, 1000]$ are normalized by screen width and height. Positive Δx denotes rightward motion, and positive Δy denotes downward motion. The resulting view rotation depends on the game's camera gain.

Sustained inputs. We extend taps and clicks with holds lasting one full control step of duration Δt. For a key or mouse button b, the press or long-hold action space is

$$\mathcal{A}_{\text{press}}(b) = \underbrace{\{\text{tap/click}(b)\}}_{\text{Retained CUA taps/clicks}} \cup \underbrace{\{\mathbf{hold}(\boldsymbol{b}; \boldsymbol{\Delta t})\}}_{\textbf{Sustained extension}}.$$

Repeating holds across consecutive control steps sustains movement and interaction.

Composite actions. We define a composite action as two or more compatible keyboard or mouse inputs executed concurrently within one control step. The step action space is

$$\mathcal{A}_{\text{step}} = \underbrace{\{\text{wait}\} \cup \mathcal{A}_{\text{keyboard}} \cup \mathcal{A}_{\text{mouse}}}_{\text{Individual CUA inputs}} \cup \underbrace{\{\mathbf{input_1} \parallel \cdots \parallel \mathbf{input}_{\boldsymbol{m}}\}}_{\textbf{Composite-action extension}}.$$

Each $\text{input}_i \in \mathcal{A}_{\text{keyboard}} \cup \mathcal{A}_{\text{mouse}}$; $\parallel$ denotes concurrent execution and $m \geq 2$. These combinations support coordinated movement and combat that is crucial for game control. For example, holding 'W' while tapping 'Space' enables running jumps, and pointer motion with a right-button hold and left-button click enables concurrent aiming and firing.

3.2 SUPERVISED INITIALIZATION FOR GAME CONTROL

The extended action space in Section 3.1 defines the available controls, but the policy must learn when and how to use them. In this section, we describe how we use SFT on a compact collection of curated gameplay demonstrations to initialize computer-use models for subsequent learning from synthetic experience.

Demonstration trajectory curation. We curate 6,044 gameplay trajectories comprising 295,097 interaction turns from public datasets (Choi et al., 2026; Markov AI, 2026; Yue et al., 2026; Baker et al., 2022) and in-house recordings, mapping recorded controls to the shared game-control action space defined in Section 3.1. Controls are aligned to shared action intervals, and source-specific mouse calibration expresses relative inputs in normalized, pixel-equivalent coordinates. After filtering incomplete or redundant recordings, we use a VLM (Google DeepMind, 2026) to annotate the recorded actions with task instructions and per-turn visual reasoning. For reasoning annotation, the VLM conditions on the task instruction, current and past observations, and the action target. Figure 2 illustrates input coverage and an annotated demonstration. Appendix B details the sources, alignment, calibration, and annotation procedures.

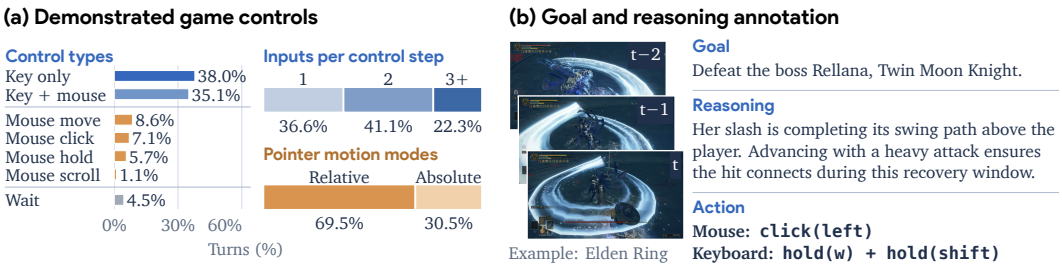


Figure 2: Curated gameplay demonstrations for supervised initialization. (a) Recorded controls summarized by control type, number of inputs per control step, and pointer-motion mode in our demonstration trajectories. Two or more concurrent inputs form a composite action. (b) A demonstration sample pairing recorded actions, with an annotated task instruction and visual reasoning.

Supervised fine-tuning. We construct each SFT example as $(q, I_t, C_H(h_{<t}), z_t, a_t)$, where the task instruction q, current screenshot I_t, and retained history $C_H(h_{<t})$ form the policy input, while visual reasoning z_t and action a_t provide supervision. Following Wang et al. (2025b), SFT, RL, and evaluation share a $2 + 2$ history: two recent visual–action turns and two earlier action-only turns, with historical reasoning omitted. We fine-tune the policy to predict (z_t, a_t) conditioned on the task instruction, current screenshot, and retained history. We use action-balanced sampling to increase exposure to less frequent controls and composite actions.

3.3 PLAYER-1-GYM: LEARNING FROM SYNTHETIC EXPERIENCE

We construct **PLAYER-1-GYM**, a suite of synthetic environments organized around control demands shared across games. Programmatically verifiable rewards, together with randomized observations and control dynamics, support our multi-environment PPO training of the SFT-initialized policy.

Verifiable environment construction. We identify recurring control demands across games and organize training tasks into five families: aiming, combat, driving, navigation, and UI interaction. For example, aiming tasks require pointer-based or camera-relative target tracking, while combat tasks combine movement with attack timing. Building upon our own shared 2D and 3D engines, reusable assets, and scene templates, we implement sixteen environments that require these forms of control through different game mechanics, viewpoints, and spatial layouts, as shown in Figure 3(a–b). A unified runtime provides screenshot observations and executes the shared game-control action space. This shared foundation facilitates extending the suite with new environments while maintaining consistent observation and action interfaces.

To provide verifiable training feedback, each environment defines game- and task-specific scoring rules over changing game states. Score increments supply step-level rewards, providing dense and verifiable feedback without a learned judge, while privileged state is not exposed to the policy.

Environment randomization. To further broaden training experience within each environment, we randomize four complementary dimensions in **PLAYER-1-GYM**: **(1) Appearance**: colors, lighting, and assets; **(2) Camera**: projection, placement, and initial orientation; **(3) Scene**: layouts, spawns, and task-specific configurations; and **(4) Control**: input-response gains for camera rotation, movement, driving, and steering where applicable. All four dimensions are jointly randomized during training where supported. Examples are provided in Figure 3(c). Appendix C specifies the distributions and per-environment settings.

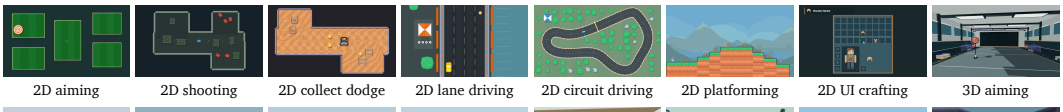
Multi-environment reinforcement learning. We jointly optimize the SFT-initialized policy with PPO across randomized environments with different control demands, dynamics, and reward scales. To limit the effect of reward-scale differences on policy-update magnitudes, we apply *per-environment advantage normalization* to generalized advantage estimates (GAE):

$$\widetilde{A}_{i,j} = \frac{A_{i,j} - \mu_e}{\sqrt{\sigma_e^2 + \epsilon}}, \quad \mu_e = \frac{\sum_{(i,j)\in\mathcal{S}_e} A_{i,j}}{N_e}, \quad \sigma_e^2 = \frac{\sum_{(i,j)\in\mathcal{S}_e} (A_{i,j} - \mu_e)^2}{N_e - 1}. \tag{1}$$

(a) Task families and control demands Player-1-Gym: 16 synthetic environments

Task family →	Control demands →	Environments
Aiming (2)	Align the pointer or camera to hit targets.	2D aiming; 3D aiming
Combat (5)	Coordinate movement, aiming, and attack timing.	2D shooting; 3D combat arena; 3D spear duel; 3D sword duel; 3D melee arena
Driving (4)	Control vehicle direction and position along paths.	2D lane driving; 2D circuit driving; 3D road driving; 3D lane driving
Navigation (4)	Move toward goals, avoid obstacles, and time jumps.	2D collect dodge; 2D platforming; 3D navigation; 3D chase
UI interaction (1)	Select and manipulate interface elements.	2D UI crafting

(b) Environment views



(c) Environment randomization



Figure 3: **PLAYER-1-GYM**: synthetic experience with verifiable rewards. (a) Shared control demands across five task families. (b) Sixteen 2D and 3D environments. (c) Appearance, camera, and scene randomization; control randomization is not visualized.

Here i indexes turns and j response tokens. For each environment e, $\mathcal{S}_e$ contains its response-token indices in the rollout batch, with $N_e = |\mathcal{S}_e|$. We normalize each $A_{i,j}$ for $(i,j) \in \mathcal{S}_e$ using the corresponding advantage mean μ_e and sample variance σ_e^2. Return-to-go targets accumulate rewards from subsequent turns, allowing later outcomes to inform earlier control decisions. Appendix D.2 further details reward construction, cross-turn credit, loss aggregation, and optimization.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Training setup. We initialize Player-1-9B and Player-1-35B-A3B from respective Qwen3.5-9B and Qwen3.5-35B-A3B backbones. Both undergo two epochs of SFT on the full demonstration dataset with action balanced sampling. We then apply PPO in the sixteen-environment **PLAYER-1-GYM** suite. Both model sizes train on the full sixteen-environment suite with a budget of 200 RL steps. Each training step collects 256 episodes, allocated equally across environments, with at most 50 interaction turns per episode. Appendix D provides the remaining hyperparameters.

Evaluation. We construct two evaluation suites from existing benchmarks: MineDojo-300, comprising 300 tasks selected from the MineDojo task collection in Minecraft (Fan et al., 2022), and ViZDoom-10, comprising ten maps from the twelve available maps (Kempka et al., 2016). We also evaluate browser-game interaction on GameWorld, covering 170 tasks across 34 games (Ouyang et al., 2026). Our standardized protocols use 8 trials per MineDojo task, 50 per ViZDoom map, and 3 per GameWorld task. Per-trial budgets are 1,000 agent decisions for MineDojo-300, 150–2,100 for ViZDoom-10 depending on the map, and 400 per role for GameWorld. Appendix E details benchmark composition and budgets. External baselines comprise JARVIS-VLA (Li et al., 2025), UI-TARS-1.5-7B (ByteDance Seed, 2025), NitroGen (Magne et al., 2026), Open-P2P (Yue et al., 2026), and GameWAM (Guo et al., 2026), retaining their preprocessing and inference protocols. We also compare the Qwen3.5 backbones, our SFT initialized checkpoints, and the final **PLAYER-1** models to separate the gains from SFT and those from synthetic experience.

Table 2: **Task success and coverage on MineDojo-300.** Results use 8 trials per task. Coverage counts tasks meeting the success thresholds; brackets show 95% task-bootstrap confidence intervals.

	Success rate (%) ↑				Coverage by successes ($n/300$) ↑		
Method	**Combat**	**Harvest**	**Overall**	**[95% CI]**	**≥1 of 8**	**≥2 of 8**	**≥4 of 8**
JARVIS-VLA	7.56	5.00	6.71	[5.38, 8.08]	95/300	41/300	5/300
UI-TARS-1.5-7B	0.56	0.00	0.38	[0.13, 0.67]	8/300	1/300	0/300
NitroGen	0.00	0.00	0.00	[-]	0/300	0/300	0/300
Open-P2P-1.2B	0.44	0.00	0.29	[0.08, 0.50]	7/300	0/300	0/300
GameWAM	14.00	9.88	12.63	[10.79, 14.54]	155/300	79/300	16/300
Qwen3.5-9B	11.75	4.13	9.21	[7.50, 11.00]	106/300	59/300	13/300
Qwen3.5-35B-A3B	8.69	6.00	7.79	[6.25, 9.46]	99/300	50/300	8/300
Player-1-9B-SFT	12.50	6.63	10.54	[8.67, 12.50]	120/300	63/300	20/300
Player-1-35B-A3B-SFT	30.31	9.50	23.38	[20.63, 26.21]	172/300	133/300	76/300
Player-1-9B	52.31	10.38	38.33	[35.13, 41.58]	209/300	173/300	136/300
Player-1-35B-A3B	55.56	12.00	41.04	[37.83, 44.29]	224/300	182/300	142/300

Table 3: **First-person control on ViZDoom-10**, with 50 trials per map. We report the mean scaled return. Map-specific reward settings and scaling factors are provided in Appendix E.4.

Method	**Precision shooting**	**Goal navigation**	**Mobile combat**	**Survival**	**Overall**
JARVIS-VLA	–0.6270	–0.1459	0.1340	0.1565	–0.1458
UI-TARS-1.5-7B	–0.6648	–0.1064	0.0587	0.1444	–0.1743
NitroGen	–0.3877	–0.0732	0.0740	0.1471	–0.0793
Open-P2P-1.2B	–0.5385	–0.0174	0.6493	0.1525	0.0602
GameWAM	0.1175	–0.1285	1.1860	0.1424	0.3938
Qwen3.5-9B	0.2257	0.1155	0.4340	0.3822	0.2975
Qwen3.5-35B-A3B	0.3376	0.1147	0.5627	0.4909	0.3912
Player-1-9B-SFT	0.4403	0.1839	0.5293	0.5423	0.4361
Player-1-35B-A3B-SFT	0.7376	0.3019	0.5380	0.6065	0.5644
Player-1-9B	1.0772	0.5974	0.3033	0.7927	0.6922
Player-1-35B-A3B	1.0423	0.2628	0.5960	0.8979	0.7236

Table 4: **Browser-game control on GameWorld**, with 3 trials for all 170 tasks of 34 games.

Method	**Success (%)**	**Progress (%)**
JARVIS-VLA	6.47	17.61
UI-TARS-1.5-7B	5.49	19.71
NitroGen	8.82	23.41
Open-P2P-1.2B	2.35	10.37
GameWAM	5.88	16.91
Qwen3.5-9B	10.39	29.22
Qwen3.5-35B-A3B	13.14	33.49
Player-1-9B-SFT	13.14	31.43
Player-1-35B-A3B-SFT	15.49	34.60
Player-1-9B	15.88	34.45
Player-1-35B-A3B	17.25	38.79

4.2 Main Results

Task-directed control in Minecraft. On MineDojo-300, SFT raises success from 9.21% to 10.54% for 9B and from 7.79% to 23.38% for 35B-A3B. Interactive learning raises 9B success to 38.33% on the same 300-task evaluation panel, a gain of 27.79 percentage points over SFT. Combat success increases from 12.50% to 52.31%, and Harvest from 6.63% to 10.38%. The number of tasks completed at least four of eight trials increases from 20 to 136, showing gains in both task coverage and repeatability.

Generalization beyond the RL training suite. The same policy improves performance on external games. On ViZDoom-10, Player-1-9B increases scaled return from 0.4361 after SFT to 0.6922. On GameWorld's 510 episodes, success rises from 13.14% after SFT to 15.88%, and progress from 31.43% to 34.45%. Player-1-35B-A3B achieves an overall ViZDoom return of 0.7236. Together, these results show that control experience acquired in **Player-1-Gym** transfers to a distinct first-person shooting and browser games, beyond the task implementations used for RL. These gains support using synthetic experience to develop transferable control capabilities without online RL in each target game.

Case Example: Control in a Unseen Commercial Game We integrate our **Player-1** to commercial games through screenshots and keyboard–mouse actions. Figure 4 shows a sequence in *Watch Dogs 2*, a game unseen during both SFT and RL, spanning on-foot navigation, target approach, vehicle interaction, and driving.

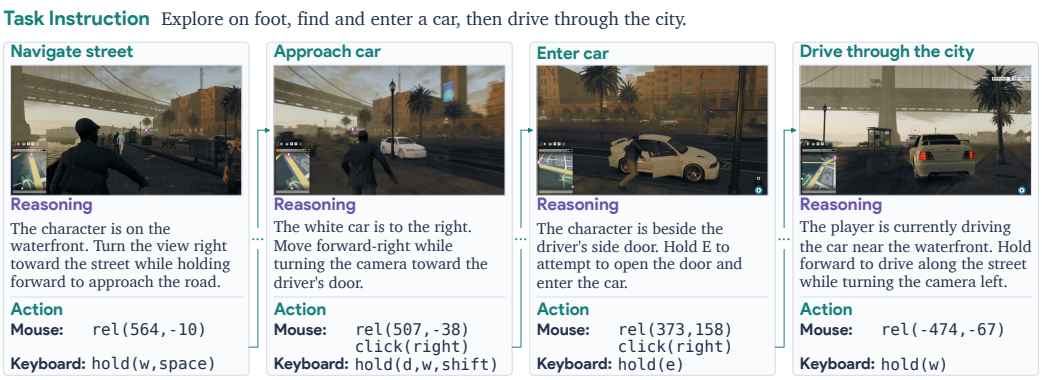


Figure 4: Player-1-9B in held-out commercial game *Watch Dogs 2*: on-foot navigation, car approach, entry, and city driving. Reasoning is lightly edited; action labels incorporate manual corrections.

5 Analysis

5.1 Learning Dynamics

The training panels show SFT-initialized 9B and 35B-A3B policies over 200 RL steps. Figure 5(a,b) separates penalized episode returns from unpenalized game scores, with each metric averaged equally across sixteen environments.

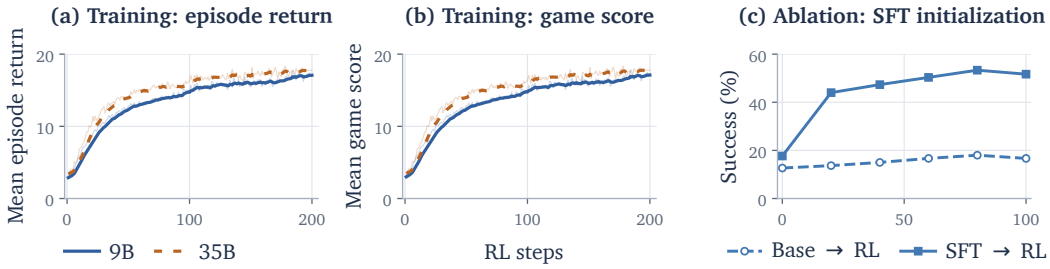


Figure 5: Training dynamics and supervised initialization. (a,b) Mean episode returns and game scores across sixteen environments for 9B and 35B-A3B over 200 RL steps; faint traces show step-level values and bold curves show EMA ($\alpha = 0.2$). (c) Base→RL and SFT→RL for 9B model.

Learning beyond SFT. Both mean episode return and mean game score rise across the displayed training horizon. Gains are largest early in training and become more gradual later; the 35B-A3B curve finishes slightly above 9B on both measures. These training panels describe progress within the training suite; transfer evaluations are shown separately in Figure 1.

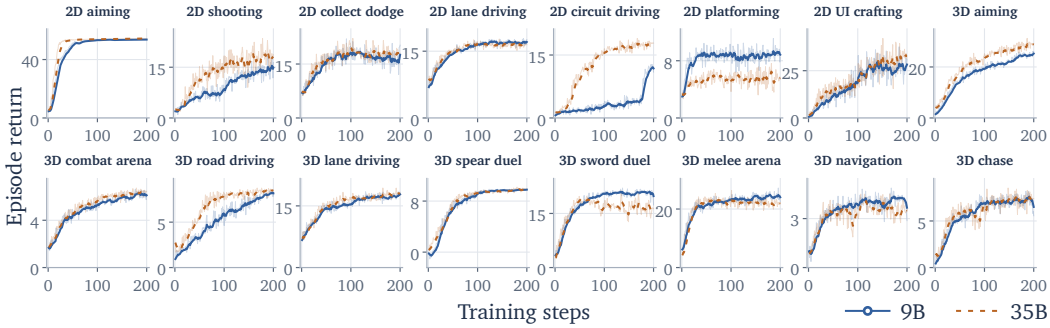


Figure 6: Per-environment episode returns for 9B and 35B-A3B over 200 RL steps. Learning progress, return scales, and model ordering vary across diverse environments.

Effect of supervised initialization. Figure 5(c) contrasts the displayed Base→RL and SFT→RL trajectories for 9B over 100 RL steps. SFT→RL rises from 17.67% to 51.67% Minecraft success, while Base→RL rises from 12.67% to 16.67%, leaving a 35.00-percentage-point gap at step 100.

Both trajectories peak at step 80 and decline slightly at the final point. The panel illustrates the role of supervised initialization separately from the training dynamics in panels (a,b).

Learning across environments. Figure 6 breaks down episode returns across all sixteen training environments using the same EMA as Figure 5(a,b). Both models show broad net gains, but learning profiles might differ: 2D aiming improves rapidly and levels off, whereas 9B improves much later in 2D circuit driving.

5.2 Environment Scaling

Expanding the training suite. The scaling comparison allocates a fixed episode budget across 2, 4, 8, or 16 environments, starting from the same 9B SFT checkpoint. Each configuration uses 100 RL steps and 256 episodes per step, allocated equally; Appendix D.4 specifies the subsets and protocol. Minecraft success on 100 tasks with three trials each increases from the 17.67% SFT reference to 39.33%, 44.67%, 48.00%, and 51.67%, respectively (Figure 7).

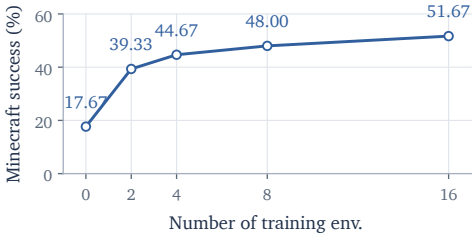


Figure 7: Environment scaling for 9B at a fixed episode budget of 100 RL steps and 25,600 episodes. Zero environments denotes the SFT reference.

5.3 Evolution of Action Composition

Figure 8 shows increased use of composite actions during RL. Across Player-1-9B's in-suite validation trajectories, the share of turns with composite actions rises from 64.7% at the start of RL to 85.4% and 93.2% after 50 and 100 RL steps, respectively. This increase occurs without an explicit reward for action composition, suggesting that task-driven RL reinforces coordinated inputs common to game control, such as aiming while firing or moving while adjusting the camera.

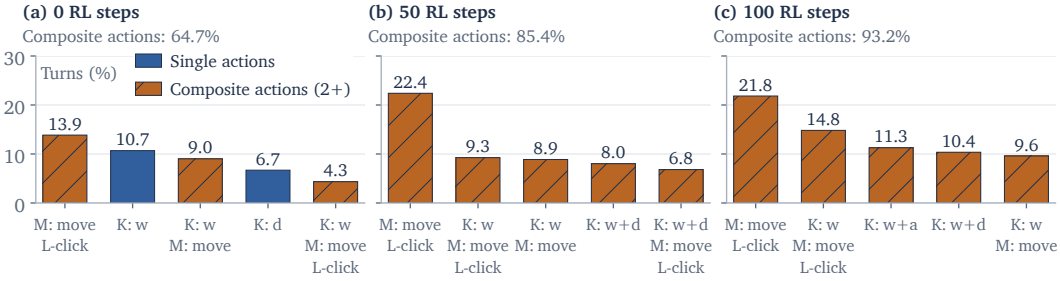


Figure 8: Evolution of action composition during **Player-1-Gym** validation. Composite-action use rises from 64.7% to 93.2% over 100 RL steps. Bars show each checkpoint's five most frequent actions as shares of all recorded turns. M: mouse movement; K: held key; L-click: left click.

6 Conclusion

We presented **Player-1**, a post-training framework for learning game control beyond demonstrations through synthetic experience. Curated gameplay demonstrations initialize computer-use models in a shared game-control action space. **Player-1-Gym** then provides sixteen synthetic environments organized around shared control demands, with verifiable rewards and randomized observations and control dynamics for multi-environment RL. Evaluations on MineDojo-300, ViZDoom-10, and GameWorld demonstrate that learning from synthetic experience advances game-control capabilities beyond SFT, with gains that generalize beyond the RL training suite. Further analyses show improved transfer with larger training suites. Together, these findings establish synthetic experience as an effective resource for learning transferable game control without online training in each target game.

AI USE STATEMENT

We used a vision–language model to annotate gameplay demonstrations with task instructions and visual reasoning, with data checks described in Appendix B. Generative AI tools also assisted with software development, methodological discussion, results analysis, figure preparation, and manuscript drafting and editing. The authors take responsibility for the final content, experimental results, and associated artifacts.

REFERENCES

Diogo Almeida. Introducing System One Models & Jev. TypeSafe AI Blog, September 2026. URL https://typesafe.ai/blog/introducing-system-one-models-and-jev.

Anthropic. Introducing Claude 4. Technical announcement, 2025. URL https://www.anthropic.com/news/claude-4.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): Learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022.

ByteDance Seed. UI-TARS-1.5-7B, 2025. URL https://huggingface.co/ByteDance-Seed/UI-TARS-1.5-7B. Model release.

Wayne Chi, Yixiong Fang, Arnav Yayavaram, Siddharth Yayavaram, Seth Karten, Qiuhong Anna Wei, Runkun Chen, Alexander Wang, Valerie Chen, Ameet Talwalkar, and Chris Donahue. GameDevBench: Evaluating agentic capabilities through game development. *arXiv preprint arXiv:2602.11103*, 2026. URL https://arxiv.org/abs/2602.11103.

Suhwan Choi, Jaeyoon Jung, Haebin Seong, Minchan Kim, Minyeong Kim, Yongjun Cho, Yoonshik Kim, Yubeen Park, Youngjae Yu, and Yunsung Lee. D2E: Scaling vision-action pretraining on desktop data for transfer to embodied AI. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.05684.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, et al. Agent-World: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

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. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18343–18362, 2022.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Technical report, Google DeepMind, 2025. URL https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf.

Google DeepMind. Gemini 3.8 Flash. Model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/.

Yuncheng Guo, Zhanqiu Zhang, Yiwen Guo, and Weijia Li. GameWAM: A world action model for video games. *arXiv preprint arXiv:2608.26200*, 2026. URL https://arxiv.org/abs/2608.26200.

Siyuan Hu, Mingyu Ouyang, Difei Gao, and Mike Zheng Shou. The dawn of GUI agent: A preliminary case study with Claude 3.5 Computer Use. *arXiv preprint arXiv:2411.10323*, 2024. URL https://arxiv.org/abs/2411.10323.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pp. 341–348, 2016. doi: 10.1109/CIG.2016.7860433. URL https://arxiv.org/abs/1605.02097.

Wolfgang Lehrach, Daniel Hennes, Miguel Lazaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Marc Lanctot, Atil Iscen, John Schultz, Marcus Chiam, Ian Gemp, Piotr Zielinski, Satinder Singh, and Kevin Murphy. Code world models for general game playing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1UoB7IWiku.

Muyao Li, Zihao Wang, Kaichen He, Xiaojian Ma, and Yitao Liang. JARVIS-VLA: Post-training large-scale vision language models to play visual games with keyboards and mouse. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17878–17899, 2025. doi: 10.18653/v1/2025.findings-acl.920.

Bo Liu, Simon Yu, Yiding Jiang, Ao Qu, Andrew Zhao, Zichen Liu, Junsu Kim, Zijian Zhou, Seungone Kim, Tongzheng Ren, Mickel Liu, Hanfei Yu, Zhaorun Chen, Weiyan Shi, Paul Pu Liang, Luke Zettlemoyer, Yejin Choi, and Natasha Jaques. SPADE: Self-play in adaptive synthetic executable environments. *arXiv preprint arXiv:2608.19197*, 2026.

Dunjie Lu, Shuai Bai, Tianyi Bai, Sicheng Fan, Chang Gao, Jian Guan, Feng Hu, Mianqiu Huang, Xingyang Huang, Yizhen Jiang, et al. Qwen-CUA: Native computer use for (almost) everything. *arXiv preprint arXiv:2608.02352*, 2026. URL https://arxiv.org/abs/2608.02352.

Meng Luo, Yanlin Li, Hao Li, Hongzhan Lin, Pengfei Zhou, Tianjie Ju, Ran Zhang, Yeying Jin, Mong-Li Lee, and Wynne Hsu. AI for games in the foundation model era. *arXiv preprint arXiv:2609.16679*, 2026a. URL https://arxiv.org/abs/2609.16679.

Tongxu Luo, Rongsheng Wang, Jiaxi Bi, Chenming Xu, Zhengyang Tang, Jianlong Chen, Juhao Liang, Ke Ji, Shuqi Guo, Yuhao Du, Fan Bu, Wenyu Du, Xiaotong Zhang, Kyle Li, Shaobo Wang, Linfeng Zhang, Yuxuan Liu, Xin Lai, Chenxin Li, Yiduo Guo, Zhexin Zhang, Xinyuan Wang, Tianyi Bai, Ziniu Li, and Benyou Wang. GameCraft-Bench: Can agents build playable games end-to-end in a real game engine? *arXiv preprint arXiv:2606.17861*, 2026b. URL https://arxiv.org/abs/2606.17861.

Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi Fan. NitroGen: An open foundation model for generalist gaming agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21511–21521, 2026.

Markov AI. Gaming 500 hours. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/markov-ai/gaming-500-hours.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. URL https://arxiv.org/abs/2410.21276.

Mingyu Ouyang, Siyuan Hu, Kevin Qinghong Lin, Hwee Tou Ng, and Mike Zheng Shou. GameWorld: Towards standardized and verifiable evaluation of multimodal game agents. *arXiv preprint arXiv:2604.07429*, 2026.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, et al. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Chengshuai Shi, Wenzhe Li, Xinran Liang, Yizhou Lu, Wenjia Yang, Ruirong Feng, Seth Karten, Ziran Yang, Zihan Ding, Gabriel Sarch, Danqi Chen, Karthik Narasimhan, and Chi Jin. Odysseus: Scaling VLMs to 100+ turn decision-making in games via reinforcement learning. *arXiv preprint arXiv:2605.00347*, 2026.

SIMA Team. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*, 2024. URL https://arxiv.org/abs/2404.10179.

SIMA Team. SIMA 2: A generalist embodied agent for virtual worlds. *arXiv preprint arXiv:2512.04797*, 2025. URL https://arxiv.org/abs/2512.04797.

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*, 2025a. URL https://arxiv.org/abs/2511.08892.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering foundation agents towards general computer control. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 58658–58725, 2025b. URL https://proceedings.mlr.press/v267/tan25h.html.

Bowen Wang, Dunjie Lu, Junli Wang, Tianyi Bai, Shixuan Liu, Zhipeng Zhang, Haiquan Wang, Hao Hu, Tianbao Xie, Shuai Bai, Dayiheng Liu, Que Shen, Junyang Lin, and Tao Yu. CUA-Gym: Scaling verifiable training environments and tasks for computer-use agents. *arXiv preprint arXiv:2605.25624*, 2026a.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, Wanjun Zhong, Yining Ye, Yujia Qin, et al. UI-TARS-2 technical report: Advancing GUI agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025a.

Yinjie Wang, Tianbao Xie, Ke Shen, Mengdi Wang, and Ling Yang. RLAnything: Forge environment, policy, and reward model in completely dynamic RL system. *arXiv preprint arXiv:2602.02488*, 2026b.

Zihao Wang, Xujing Li, Yining Ye, Junjie Fang, Haoming Wang, Longxiang Liu, et al. Game-TARS: Pretrained foundation models for scalable generalist multimodal game agents. *arXiv preprint arXiv:2510.23691*, 2025b. URL https://arxiv.org/abs/2510.23691.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Yuguang Yue, Irakli Salia, Samuel Hunt, Chris Green, Wenzhe Shi, and Jonathan J. Hunt. Scaling behavior cloning improves causal reasoning: An open model for real-time video game playing. *arXiv preprint arXiv:2601.04575*, 2026. URL https://arxiv.org/abs/2601.04575.

Pengfei Zhou, Hexin Wang, Zhengfeiyang Zhang, Yixing Ma, Zhenglin Wan, Kaipeng Zhang, Wangbo Zhao, and Yang You. Agentic game development as a verifiable trajectory data engine for scaling world models. *arXiv preprint arXiv:2608.25518*, 2026. URL https://arxiv.org/abs/2608.25518.

A GAME-CONTROL INTERFACE

A.1 COORDINATES AND COMPOSITE ACTIONS

Table 5 specifies the execution semantics of the actions introduced in Section 3.1. For an observation of width W and height H, normalized absolute positions and relative displacements map to pixels as

$$(x, y) \mapsto (xW/1000, yH/1000), \qquad (\Delta x, \Delta y) \mapsto (\Delta xW/1000, \Delta yH/1000).$$

The origin is the upper-left corner, and positive displacement points right and down. One movement uses either absolute or relative coordinates. A key cannot be both held and tapped in the same step, nor can one mouse button be both held and clicked.

Table 5: Game-control semantics. Inputs in a composite action execute concurrently within one step; bindings are restricted to those supported by the environment.

Component	**Semantics**
Key hold / tap	Hold a set of keys throughout the step, or briefly press and release them.
Mouse movement	Move to an absolute screen position or apply a relative displacement.
Mouse hold / click	Hold one button throughout the step, or click one button; different buttons may execute concurrently.
Mouse scroll	Apply one upward or downward scroll step.
Wait	Advance one step without keyboard or mouse input.

A.2 EXECUTION AND VISUAL FEEDBACK

Relative motion requests input displacement, not a universal camera angle. For per-pixel camera gain g, requested yaw and pitch increments $\delta\psi$ and $\delta\phi$ have magnitudes

$$|\delta\psi| = gW|\Delta x|/1000, \qquad |\delta\phi| = gH|\Delta y|/1000,$$

before native camera limits; mouse displacement is not multiplied by action duration. The seven mouse-controlled 3D **PLAYER-1-GYM** environments use a nominal gain $g_0 = 0.00196$ radians per observation pixel. With Control randomization, the effective gain is $g = g_0 L$, where L is defined in Appendix C.2. Observation geometry and camera constraints remain fixed within a checkpoint comparison. Benchmark-specific settings are given in Appendix E.

A.3 OBSERVATION AND INTERACTION HISTORY

SFT, RL, and evaluation all use two recent visual–action turns and two earlier action-only turns (2+2). In the earlier action-only turns, screenshots are replaced by `<image omitted>`; reasoning in all retained historical turns is replaced by `<thinking omitted>`. The current screenshot and executed historical actions are retained.

B GAMEPLAY DEMONSTRATIONS: CONSTRUCTION AND ANNOTATION

B.1 SOURCES AND STATISTICS

Both model sizes use the collection in Figure 9: 6,044 trajectories and 295,097 turns from 2,544 source-qualified recording sessions and 172 game groups. Session counts are deduplicated, and game coverage uses a fixed alias mapping. Sampled control hours equal the turn count $\times$ 200 ms, or 16.4 hours.

Quality selection. We automated validity, activity, and repetition checks complement source-level and manual review of recording completeness, observation–action alignment, and sequence diversity. The collection contains 5,360 continuous-gameplay trajectories with total 268,000 turns and 684 UI trajectories with total 27,097 turns.

Action statistics. Figure 2(a) counts each current action once, before training-time resampling. Keyboard–mouse composites take precedence. Mouse-only turns are assigned by click, hold, scroll, then movement, so button and scroll categories may include pointer motion. Inputs per step count each key tap/hold and mouse operation separately, with wait as one no-op. Pointer-mode shares use all turns containing `mouse_move`, including keyboard–mouse composites.

(a) Sources & scale

Source	Traj.	Turns	Hours
Public datasets			
Desktop to Embodied AI (2026)	1,005	49,883	2.8
Gaming 500 Hours (2026)	985	48,652	2.7
Open Pixel2Play (2026)	1,140	57,000	3.2
Video PreTraining (2022)	1,153	56,431	3.1
In-house recordings			
In-house Gameplay Recordings	1,761	83,131	4.6
Total	**6,044**	**295,097**	**16.4**

(b) Game distribution



Game	Traj.	Turns	Share
Minecraft	1,236	60,545	20.5%
Grand Theft Auto V	400	18,478	6.6%
Cyberpunk 2077	185	8,636	3.1%
Apex Legends	165	8,161	2.7%
Red Dead Redemption	150	7,158	2.5%
Valorant	144	6,921	2.4%
PUBG	143	6,957	2.4%
Palworld	142	6,806	2.3%
Elden Ring	119	5,849	2.0%
WutheringWaves	113	5,506	1.9%
Other (162)	3,247	160,080	53.7%

Figure 9: Sources and game distribution of the 6,044 gameplay demonstrations used for SFT. (a) Sources and scale, with sampled control hours. (b) Game distribution.

B.2 Action Standardization

Temporal alignment. For continuous gameplay, source timestamps define contiguous intervals of 150–250 ms, projected onto a 200-ms target grid using cumulative boundaries. Relative input is summed within each source interval. Keyboard presses shorter than 100 ms become taps, and longer presses become holds on overlapping target steps. Mouse presses lasting at most 300 ms become clicks. Each target summarizes compatible recorded inputs for concurrent execution within one control step.

UI trajectories also follow recorded operation boundaries. Clicks retain their recorded targets, key sequences expand into individual presses, drags use start and endpoint controls with an explicit release, and scroll gestures become directional steps. Execution uses the shared 200-ms action space. Sampled control hours use target execution duration, while UI trajectories retain their recorded, operation-dependent timing.

Spatial alignment. Native relative-mouse input scales vary across data sources and recording setups. We therefore calibrate these inputs to pixel-equivalent displacements using freely moving GUI pointers. The mapping also applies to pointer-locked recordings. Camera response is defined in Appendix A.2. For native-input sums (r_x, r_y), coefficients (k_x, k_y), and logical recording size (W_m, H_m), the normalized target is

$$\Delta x = \mathrm{round}(1000 k_x r_x / W_m), \qquad \Delta y = \mathrm{round}(1000 k_y r_y / H_m).$$

Mouse-move Calibration. In-house and D2E calibration uses GUI windows with stable focus and geometry, aligned timestamps, and no pointer clipping or recentering. For paired native input r_i and pointer displacement p_i, window and pooled session estimates are

$$k_i = \frac{r_i^\top p_i}{r_i^\top r_i}, \qquad \hat{k} = \frac{\sum_i r_i^\top p_i}{\sum_i r_i^\top r_i}.$$

D2E uses window-equal estimates for a source-level coefficient; in-house coefficients are recording-specific. Pixel2Play uses per-axis Huber regression over at least two stable, nonoverlapping two-second GUI blocks. Out-of-range projections are rejected rather than clipped. Table 6 lists the source mappings.

Table 6: Source-specific mouse-action mapping. Relative inputs use calibration coefficients bound to recording geometry; UI operations use absolute endpoints.

Source	Mapping
In-house	Recording-specific calibration coefficients.
D2E	Source-level $k_x = k_y = 0.66$ for relative input; absolute endpoints for UI operations.
Pixel2Play	Hierarchical selection: session GUI fit $\rightarrow$ metadata profile $\rightarrow$ display scale.
VPT	$k_x = k_y = 2$ in logical 1280×720 coordinates for relative input; absolute endpoints for UI operations.
Gaming 500h	Recorded absolute positions projected using capture geometry.

B.3 Goal and Reasoning Annotation

We use VLM to build reusable profiles of rules, roles, inputs, and examples for each games in the trajectory, and check them for recorded-control coverage and action validity. It assigns goals from

each profile, full-trajectory storyboard, and action timeline. The same game from different data sources should share the same assembled profile. For per-turn reasoning annotation, it uses the goal, fixed action target, and current and past frames.

C PLAYER-1-GYM: TASKS, REWARDS, AND RANDOMIZATION

C.1 TASKS AND VERIFIABLE REWARDS

By primary task, the sixteen environments comprise aiming (2), combat (5), driving (4), navigation (4), and UI interaction (1). Tables 7 and 8 specify their tasks, score components, and API state fields. Interaction tests, score checks, and human review verify controls, observable transitions, and reward events against human-defined rules.

The shared Three.js browser runtime provides seeded resets, keyboard–mouse execution, screenshots, and rewards, with fixed-step simulation independent of inference latency. Each environment defines a cumulative score $S_e(s)$ and supplies the increment $r_{e,t} = S_e(s_{t+1}) - S_e(s_t)$ as reward. Reward computation uses privileged state; the policy receives only screenshots and task/control text. Episodes end on native completion or failure, or at the turn cap.

Table 7: 2D environments, controls, and verifiable score components. State fields are read from `gameGymAPI.getState().metrics`; `primary_score` gives the resulting cumulative score.

Environment	Task and control	Score components	State fields (API)
2D aiming	Track and click moving targets	Hits $+0.1\times$ accumulated precision	`targets_collected`, `precision_total`
2D shooting	Move, aim, and shoot	Enemies defeated	`kills`
2D collect dodge	Collect while avoiding hazards	Coins collected	`coins_collected`
2D lane driving	Time lane changes	Coins collected	`coins_collected`
2D circuit driving	Steer through a route	Checkpoints reached	`checkpoints_passed`
2D platforming	Run, jump, and dash	Coins, cleared gaps/spikes, and flags	`coins_collected`, `gaps_cleared`, `spikes_cleared`, `flags_reached`
2D UI crafting	Consult, craft, and equip	Recipe consultations + material reward $+3\times$ crafts + selections $+2\times$ equips	`recipe_book_rewards`, `correct_material_reward`, `target_crafts`, `correct_equipment_selections`, `equipment_equipped`

Table 8: 3D environments, controls, and verifiable score components. State fields are read from `gameGymAPI.getState().metrics`; `primary_score` gives the resulting cumulative score.

Environment	Task and control	Score components	State fields (API)
3D aiming	Track and hit targets	Target hits	`hits`
3D combat arena	Coordinate movement, aim, and fire	Enemies defeated	`kills`
3D road driving	Follow bends and checkpoints	Checkpoints reached	`checkpoints_passed`
3D lane driving	Collect and avoid with lane changes	Coins collected	`coins_collected`
3D spear duel	Track, charge, release, and dodge	Spear hits $-0.5\times$ damage received	`spear_hits`, `damage_taken`
3D sword duel	Approach, aim, and strike	Attack hits + defeats $-0.5\times$ damage received	`attack_hits`, `kills`, `damage_taken`
3D melee arena	Coordinate facing, range, and movement	Attack hits + defeats $-0.5\times$ damage received	`attack_hits`, `kills`, `damage_taken`
3D navigation	Reach successive flags	Goals reached	`goals_reached`
3D chase	Pursue around obstacles	Targets reached	`targets_reached`

C.2 Randomization and Sampling

Table 9 specifies the parameters, distributions, and applicable environments for Appearance, Camera, Scene, and Control. Appearance preserves collision geometry and functional roles; Scene varies layouts, spawns, and subsequent entity generation, potentially changing task difficulty. Appearance and Camera are sampled at reset and fixed within each episode, with triangular modes at native values. Background lightness and saturation, ambient intensity, and key-light intensity share one sampled quantile, and linked asset components receive compatible color changes. Orthographic zoom is relative to the native view, including 2D platforming. Initial yaw/pitch is distinct from camera-rig pitch.

Control-response sampling. Control varies input response independently of camera projection and scene configuration. For each supported response parameter k, we independently sample

$$c_{e,k} \sim \mathrm{Tri}(a_{e,k}, 1, b_{e,k}), \qquad c_{e,k}(t) = c_{e,k}(0) \quad \text{throughout the episode.} \tag{2}$$

The control seed is independent of domain randomization; separate parameter substreams set values at reset, unchanged through respawns. The narrower chasing, platforming, and jump ranges preserve pursuit-speed and traversal constraints. Collision rules, gravity, attack timing, control bindings, and reward definitions remain fixed. Table 10 specifies the training mixture, and Figure 10 shows individual and joint Appearance, Camera, and Scene variation.

Table 9: Appearance, camera, scene, and control-response randomization configurations. Multipliers are relative to each game's native setting. U denotes uniform sampling and Tri is parameterized by lower bound, mode, and upper bound. Rig parameters affect only camera components used by the game. Scene variation is task-specific and may already be present in native settings.

(a) Appearance

Visual attribute	Native	Perturbed distribution / choice	Applies to
Entity hue shift	$0°$	Uniform over $\{0, 60, 120, 180, 240, 300\}°$	All games
Entity saturation	1	$U(0.95, 1.25)$	All games
Entity lightness	1	$U(0.96, 1.15)$	All games
Background hue shift	$0°$	$\mathrm{Tri}(-10, 0, 10)°$	All games
Background saturation	1	$\mathrm{Tri}(0.55, 1, 1.05)$	All games
Background lightness	1	$\mathrm{Tri}(0.32, 1, 1.04)$	All games
Color temperature	Neutral	Warm / neutral / cool: $0.2/0.6/0.2$	All games
Ambient intensity	1	$\mathrm{Tri}(0.35, 1, 1.15)$	3D games
Key-light intensity	1	$\mathrm{Tri}(0.45, 1, 1.20)$	3D games
Key-light azimuth	$0°$	$-120°/0°/120°$: $0.2/0.6/0.2$	3D games
Shadow strength	1	$\mathrm{Tri}(0.78, 1, 1)$	3D games
Visual assets	Native	Compatible non-native variants	All games

(b) Camera

Camera parameter	Native	Perturbed distribution	Applies to
Orthographic zoom	1	$U(0.80, 1.00)$	2D games
FOV offset	$0°$	$U(-8, 8)°$	3D games
Rig height	1	$U(0.90, 1.10)$	3D games
Follow distance	1	$U(0.85, 1.15)$	3D road/lane driving, melee arena, chase
Rig pitch offset	0	$U(-0.05, 0.05)$ rad	3D except driving
Initial yaw: aiming	0	$U(-0.10, 0.10)$ rad	3D aiming, 3D combat arena
Initial yaw: combat	0	$U(-0.05, 0.05)$ rad	3D duel/melee games
Initial yaw: navigation	0	$U(-0.20, 0.20)$ rad	3D navigation, 3D chase
Initial pitch	0	$U(-0.05, 0.05)$ rad	All three groups

(c) Scene

Environment	Task-specific variation
2D aiming	Target positions, motion, and spatial layouts
2D shooting	Connected walls, enemy regions, and spawns
2D collect dodge	Connected layouts, coins, and moving obstacles
2D lane driving	Initial lane and object sequences
2D circuit driving	Circuit geometry, direction, scale, and mirroring
2D platforming	Platforms, gaps, hazards, and route widths
2D UI crafting	Recipes, quantities, distractors, and inventory layouts
3D aiming	Target directions, distances, and spawns
3D combat arena	Connected rooms, cover, and spawns
3D road driving	Road curvature, phase, and mirroring
3D lane driving	Initial lane and object sequences
3D spear duel	Arena geometry and opponent spawns
3D sword duel	Platforms, cover, and spawns
3D melee arena	Connected courtyards, gates, and spawns
3D navigation	Obstacles, starts, and goal positions
3D chase	Building geometry, starts, and target routes

(d) Control

Response multiplier	Native	Perturbed distribution	Applies to
Look gain L	1	$\mathrm{Tri}(0.8, 1, 1.2)$	3D except driving
Movement M	1	$\mathrm{Tri}(0.8, 1, 1.2)$	2D shooting, collect dodge, and lane driving; 3D combat arena, lane driving, spear duel, sword duel, melee arena, and navigation
Chasing movement M	1	$\mathrm{Tri}(0.9, 1, 1.1)$	3D chase
Platform movement M	1	$\mathrm{Tri}(0.95, 1, 1.05)$	2D platforming
Driving D	1	$\mathrm{Tri}(0.8, 1, 1.2)$	4 driving games
Steering T	1	$\mathrm{Tri}(0.8, 1, 1.2)$	3D road driving
Jump impulse J	1	$\mathrm{Tri}(0.975, 1, 1.025)$	2D platforming; 3D combat arena

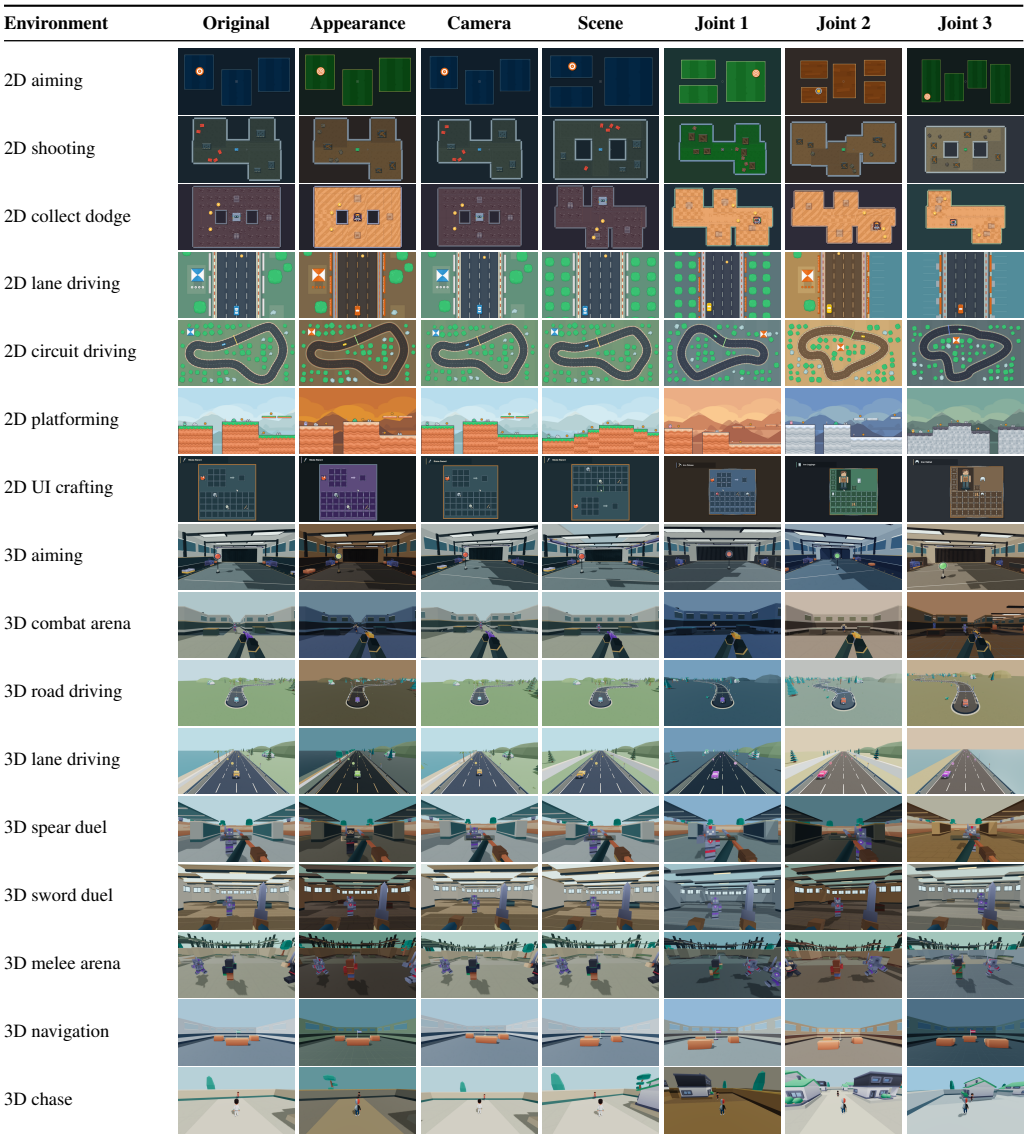


Figure 10: Visual examples of **PLAYER-1-GYM** randomization: 16 environments × seven views. Original uses default profiles; Appearance, Camera, and Scene change one domain; Joint 1–3 combine these three dimensions. Control-response variation is omitted because its effects require action sequences rather than static screenshots.

D POST-TRAINING DETAILS

D.1 SUPERVISED FINE-TUNING

Both model sizes use token-level cross-entropy on the current turn's reasoning and action, with the history in Appendix A.3.

Action-balanced sampling. We classify each turn by its current recorded action into overlapping pools: keyboard, mouse movement, click, hold, any mouse button, keyboard–mouse, movement–click, keyboard–click, and keyboard–movement–click. Keyboard includes held and tapped keys; keyboard–mouse includes any mouse input, including scrolling. Other contains the remaining turns, including wait and scroll-only actions. Each draw selects a pool proportionally to its weight (1 for each control pool; 0.1 for Other), then samples a turn uniformly with replacement.

D.2 Multi-Environment Reinforcement Learning

We use synchronous PPO with a dense critic initialized from Qwen3.5-9B for both actor sizes. Rollouts retain the observations, prompts, responses, masks, and behavior log probabilities used at generation, preserving the same conditioning during optimization.

Reward and cross-turn credit. For turn t in environment e, we define the penalized step reward and undiscounted return-to-go as

$$\widetilde{r}_{e,t} = r_{e,t} - p_{e,t}^{\text{invalid}} - p_{e,t}^{\text{length}}, \qquad G_{e,t} = \sum_{u=t}^{T} \widetilde{r}_{e,u}, \tag{3}$$

where $r_{e,t}$ is the game-score increment (Appendix C.1) and T the terminal turn. Training curves average episode returns $G_{e,1}$. The penalties are

$$p_{e,t}^{\text{invalid}} = \mathbf{1}[\text{invalid action}], \qquad p_{e,t}^{\text{length}} = \min\{1,\ 0.005 \max(0, n_t^{\text{reason}} - 64)\},$$

where n_t^{reason} counts generated reasoning/prose tokens. An invalid action executes the runtime's wait fallback. We apply no additional terminal-failure penalty or task-wise reward rescaling. With $\gamma = \lambda = 1$, return targets are computed over complete trajectories, preserving cross-turn credit.

Advantage normalization and loss reduction. We normalize advantages globally in GAE, then within environments using Equation (1), with $N_e \geq 2$ and $\epsilon = 10^{-8}$; critic return targets are unchanged. For minibatch $\mathcal{B}$, let $\mathcal{T}_i$ be turn i's nonempty response-token set selected by the loss mask. The clipped PPO loss averages within turns, then across turns:

$$\mathcal{L}_{\text{policy}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{1}{|\mathcal{T}_i|} \sum_{j \in \mathcal{T}_i} \ell_{i,j}^{\text{PPO}}(\theta; \widetilde{A}_{i,j}). \tag{4}$$

The critic uses the same reduction. Normalization is token-weighted within environments, whereas losses weight turns equally. Each environment therefore contributes in proportion to its included turns.

D.3 Training Implementation

Table 10 summarizes the shared recipe for both model sizes. For each model size, SFT uses 32 NVIDIA H200 GPUs and takes approximately 24 hours, while a 200-step RL run uses 64 H200 GPUs and takes approximately 36 hours. Both durations include validation. Browser rollouts use a cluster-wide pool of 32 workers, each managing up to 16 isolated Playwright browser contexts, for a total capacity of 512 concurrent environments. Browser processes and contexts are reused across episodes to amortize startup costs.

D.4 Environment-Scaling Protocol

For Section 5.2, 9B policies start independently from the same SFT checkpoint, sharing the PPO, history, reward, and randomization settings in Table 10. Table 11 gives the suites and matched episode budgets, while interaction lengths and compute may vary. Comparisons share Minecraft development tasks and trial seeds, with SFT as the zero-environment reference.

Table 10: Post-training configuration. Both model sizes share the settings below except for actor architecture. Each RL step includes rollout collection and PPO updates; critic-only warmup is excluded.

Setting	Player-1-9B	Player-1-35B-A3B
Actor / critic	Dense 9B / dense 9B	MoE 35B-A3B / dense 9B
SFT		
Training trajectories	6,044	
History (visual–action + action-only)	$2 + 2$	
Sampling	Pool weights: nine action pools 1 each; Other 0.1	
Epoch budget / global batch	2 / 512 turns	
Learning rate / warmup fraction	5×10^{-7} / 0.03	
Interactive learning		
Algorithm / discount / GAE parameter	PPO / $\gamma = 1$ / $\lambda = 1$	
RL training steps	200	
Critic-only warmup	5 iterations	
History (visual–action + action-only)	$2 + 2$	
Observation resolution	480P	
Rollout batch / PPO minibatch	256 episodes / 64 turn responses	
Episode cap	50 turns	
Output cap per turn	256 tokens	
Actor learning rate	10^{-6}	
Critic learning rate	10^{-5}	
Optimizer / learning-rate schedule	AdamW / constant	
AdamW β_1, β_2 / weight decay	$(0.9, 0.999)$ / 0.01	
PPO epochs per rollout batch	1	
Gradient-norm clip / PPO clip	1.0 / 0.2	
Reward rescaling	None	
Advantage normalization / loss	Per environment / token mean, then turn mean	
Environment allocation	Equal across 16 environments	
Randomization mixture	100% Appearance + Camera + Scene + Control	

Table 11: Environment subsets for 9B scaling. All runs use 100 RL steps and 25,600 episodes (256 per step), allocated equally across environments.

Envs.	Environment subset	Episodes / env. / step
2	2D shooting; 3D navigation	128
4	2-env. subset + 2D circuit driving; 3D aiming	64
8	4-env. subset + 2D UI crafting; 2D platforming; 3D sword duel; 3D road driving	32
16	Full **PLAYER-1-GYM** suite	16

E EVALUATION DETAILS AND SUPPLEMENTARY RESULTS

Table 12 lists the full panels and budgets. Methods share each panel's tasks, trial seeds, and decision budgets while retaining model-specific preprocessing and inference schedules. Periodic RL evaluation uses 100 Minecraft tasks with three trials each, ten ViZDoom maps with ten episodes each, and 170 GameWorld tasks with one trial each. Qwen3.5, Player-1-SFT, and final Player-1-9B use the history in Appendix A.3. Across game benchmarks, sampling uses temperature 1.0, top-p 0.95, and top-k 20, with action-format retries disabled.

Table 12: Evaluation panels, budgets, and reporting units. Decision limits are per episode (per role in GameWorld). The three external benchmarks use environments outside **PLAYER-1-GYM**.

Panel	Tasks / maps	Trials	Budget	Metrics
PLAYER-1-GYM	16 games	128 / game	50 decisions	Mean episode return
MineDojo-300	300 tasks	8 / task	1,000 decisions	Success; task coverage counts
ViZDoom-10	10 maps	50 / map	150–2,100 decisions	Scaled return
GameWorld	170 tasks; 34 games	3 / task	400 decisions	Success; progress

Evaluation cost. Our full MineDojo-300, ViZDoom-10, and GameWorld protocols allocate up to 3.14 million agent decision turns per model, counting each controlled role separately. Our in-suite validation records average roughly 3,000 input tokens and 100 output tokens per turn, including visual

context and generated reasoning. Using these observed lengths as a workload proxy, a full-budget evaluation would cost approximately US$110,000 at GPT-6 Astra's standard API rates of US$10 and US$50 per million input and output tokens, respectively, before caching discounts.[1] Actual costs depend on model-specific tokenization, reasoning, and episode lengths. This estimated cost motivates our focus on representative released game-control and computer-use policies rather than proprietary API models.

E.1 Baseline Inference and Action Adapters

JARVIS-VLA uses its released action-token decoder and two history entries to predict one action per decision. UI-TARS-1.5-7B uses three previous screenshots, deterministic decoding, and a 400-token output limit. Its adapter maps pixel-coordinate actions to one benchmark interval, with drag and double-click unsupported. NitroGen aggregates an 18-action chunk into six benchmark decisions (three native actions per decision), refreshing the image after each chunk. GameWAM predicts 16 actions and executes eight before replanning, using its Minecraft checkpoint outside Doom and its ViZDoom checkpoint for Doom. Open-P2P uses visual-only conditioning and session-specific recurrent state. All temporal inference state is isolated between episodes.

Model inputs retain released preprocessing: Minecraft is rendered at 864×480. We resize the screenshots to 640×360 for JARVIS-VLA and UI-TARS, 256×256 for NitroGen, 224×224 for GameWAM, and 192×192 for Open-P2P. JARVIS-VLA and GameWAM retain their native degree-based Minecraft camera actions, while Open-P2P uses an adapter gain of 0.15 degrees per raw mouse pixel. Browser-game adapters project native actions onto per-game and per-role bindings. Benchmark action intervals remain fixed, so a native action chunk can span different simulated durations across benchmarks.

E.2 **Player-1-Gym** In-Suite Validation

In-suite validation uses the seeded panel in Table 12 with the training randomization settings. For an episode of T turns in environment e, the return is

$$G_{e,1} = \sum_{t=1}^{T} \left(r_{e,t} - p_{e,t}^{\text{invalid}} - p_{e,t}^{\text{length}} \right).$$

Here $r_{e,t} = S_e(s_{t+1}) - S_e(s_t)$ is the native game-score increment (Appendix C.1), and the invalid-action and length penalties are defined in Appendix D.2. We report mean episode return, averaging trials within each environment and then equally across the sixteen environments.

E.3 MineDojo-300

MineDojo-300 contains 200 Combat and 100 Harvest tasks from MineDojo, ported to MineStudio on Minecraft 1.16.5. World and task seeds are deterministically derived from each task and trial. Section 5 averages success over the 100-task development subset of 60 Combat and 40 Harvest, using three trials per task across comparisons.

Success is verified after initialization from inventory acquisition (Harvest) or target kills (Combat), rather than shaping reward. We average success over trials within each task, then equally over tasks. One decision advances a 50-ms simulator tick. **Player-1**'s camera adapter uses a nominal gain of 0.00196 radians per observation pixel, subject to native constraints.

Task coverage. We report $C_m = \sum_{i=1}^{N} \mathbf{1}[s_i \geq m]$ for $m \in \{1, 2, 4\}$, where s_i counts successes among eight trials for task i. Entries C_m/N, with $N = 300$, count tasks meeting each threshold.

Overall success intervals use 100,000 percentile task-cluster bootstrap replicates, stratified by Combat and Harvest while retaining all eight trials per task and the panel's category weights. The 95% intervals quantify within-panel task-sampling uncertainty. For all-zero outcomes, we report [-].

[1] `https://developers.openai.com/api/docs/models/gpt-6-astra`, accessed Sep 24, 2026.

E.4 VIZDOOM-10

Table 13 lists the ten selected maps, control categories, native rewards, and fixed return scales. ViZDoom 1.3.0 provides 640×480 RGB observations. Each decision advances two simulator tics. Basic permits 150 decisions, Deathmatch 2,100, and the other maps 1,050, subject to earlier native termination. **PLAYER-1**'s camera adapter uses 0.00196 radians per observation pixel with native pitch constraints and no additional horizontal-turn cap.

From raw return $R_{m,s}$ and map-specific scale $c_m > 0$, we compute map, category ($\mathcal{M}_c$), and overall scaled returns:

$$S_m = \frac{1}{N}\sum_{s=1}^{N}\frac{R_{m,s}}{c_m}, \qquad S_c = \frac{1}{|\mathcal{M}_c|}\sum_{m\in\mathcal{M}_c} S_m, \qquad S_{\text{overall}} = \frac{1}{10}\sum_{m=1}^{10} S_m.$$

Evaluation uses $N = 50$ fixed seeds per map, with identical scales across models. The survival scale of 2,100 corresponds to the native time-reward budget, with death penalties retained. Observation geometry, timing, and camera constraints are matched within each backbones.

Table 13: ViZDoom-10 map categories, native reward rules, and fixed return scales. Rewards accumulate over an episode; a tic is one simulator tick. Scaled Return divides the raw episode return by c_m without clipping.

Category	**Map**	**Raw reward rule**	**Scale c_m**
Precision shooting	Basic	$+106$ for eliminating the target; -5 per shot; -1 per tic.	100
	Defend the Center	$+1$ per kill; -1 on death.	10
	Defend the Line	$+1$ per kill; -1 on death.	10
Goal navigation	Deadly Corridor	Forward position along the corridor; $+1000$ for collecting armor; -100 on death.	1000
	My Way Home	$+1$ for reaching the goal; -0.0001 per tic.	1
Mobile combat	Battle 1	$+1$ per kill.	10
	Battle 2	$+1$ per kill.	10
	Deathmatch	Enemy-dependent kill bonuses, with 1, 3, 3, 4, 3, and 10 for the six enemy types.	10
Survival	Health Gathering	$+1$ per tic alive; -100 on death.	2100
	Health Gathering Supreme	$+1$ per tic alive; -100 on death.	2100

E.5 GAMEWORLD

GameWorld uses five tasks per game with trial seeds zero, one, and two (Table 12). The policy receives screenshots, rules, and instructions, while evaluator state is hidden. Action duration, observation geometry, and admissible controls follow each game's configuration, and multi-role games keep separate histories.

Success is the fraction of trials reporting completion. Objective progress is the achieved fraction from its initial value to its target, clipped to $[0, 1]$. The evaluator averages objectives and retains the best task progress reached. Reported success and progress average trials within each task, then equally across tasks.