GameDojo: Benchmarking Real-Time Interactive Agents in Video Games

Kuan Zhang[1,*], **Zhihao Yang**[2,*], **Yukun Chen**[2,*], **Xingyiliang Yin**[3], **Mingyu Ouyang**[4], **Yue Su**[5], **Xiangnan Wu**[2], **Jiaming Li**[2], **Longze Chen**[2], **Run Luo**[4], **Zirong Chen**[1], **Lu Wang**[1,†], **Yiming Li**[1,†]

[1]*Tsinghua University* [2]*University of the Chinese Academy of Sciences*
[3]*University of Macau* [4]*National University of Singapore* [5]*University of Hong Kong*

[*]Equal contribution. [†]Co-corresponding authors.

Interactive intelligence requires agents to act on observations and adapt to the consequences of actions. Studying these abilities calls for diverse, challenging tasks under controlled conditions. We introduce GAMEDOJO, a controllable interactive gameplay benchmark of 120 boss encounters across 20 games for evaluating model capabilities, learning algorithms, and agent harnesses in interaction. We have the coding agents reference classic games and scenes to build the environment and write gameplay scripts to collect gameplay data automatically, producing GAMEDOJO-Data with 600 hours of high-quality gameplay trajectories. Experiments reveal substantial challenges for frontier models, especially in real-time interaction where all models win 0 score. Training on GAMEDOJO-Data brings GAMEDOJO-4B to frontier performance, with the strongest real-time performance. We also find joint training improves individual tasks, and learned policies transfer across bosses. Together, these findings establish GAMEDOJO as a challenging and valuable interactive gameplay benchmark for the community.

Project Page GitHub Benchmark & Dataset Model

1 Introduction

Generalist agents must achieve goals through interaction with diverse environments (Reed et al., 2022; SIMA Team, 2024, 2025; Wang et al., 2025; Tan et al., 2025a). This requires interactive intelligence: understanding the current situation, anticipating the consequences of actions, and continually adjusting decisions based on feedback (Yao et al., 2023; Shinn et al., 2023; Wang et al., 2024). Adversarial tasks concentrate these demands. An opponent continually changes the situation and opportunities for action, requiring agents to anticipate its behavior, choose a response, and act at the right moment through precise control (Vinyals et al., 2019; de França et al., 2019; Chen et al., 2025; Wang et al., 2026a). These tasks provide a concrete setting for examining how perception, prediction, decision-making, and control work together to produce effective behavior.

Building diverse, repeatable adversarial tasks in the physical world is costly and can pose safety risks, making video games a safe and controllable environment (Bellemare et al., 2013). However, whole-game evaluations combine navigation, puzzle solving, and combat (Zhang et al., 2025; Park et al., 2026), making it difficult to locate specific bottlenecks. Open-source environments often feature older or small games (Kempka et al., 2016; Cobbe et al., 2020), while modern games are typically closed-source (Tan et al., 2025b; Schuck, 2023). Boss encounters offer a focused task unit in which recognizing attack cues, timing actions, and adjusting strategies directly affect progress (Chen et al., 2025; Schuck, 2023). Implementing such encounters as independent envi-

Email:zhang-k26@mails.tsinghua.edu.cn; yimingli9702@mail.tsinghua.edu.cn



Figure 1: GAMEDOJO: a controlled and challenging arena for interactive agents. The benchmark spans 20 games and 120 boss encounters in diverse interaction patterns, supporting the study of model capabilities, algorithms, and agent harnesses.

ronments preserves their interactive challenges while making task conditions controllable and outcomes measurable.

We introduce GAMEDOJO, a challenging benchmark of 120 boss encounters across 20 popular and classic games, together with GAMEDOJO-Data and GAMEDOJO-4B. Gameplay controllers collect a 600-hour trajectory dataset, GAMEDOJO-Data, where we train GAMEDOJO-4B. Built on Gymnasium, GAMEDOJO supports custom experiments and provides a unified evaluation interface for studying model capabilities, learning algorithms, and agent harnesses on interactive gameplay tasks.

Our contributions are threefold:

- **A diverse, controllable and challenging benchmark for interactive intelligence.** We introduce GAMEDOJO, with 120 boss encounters across 20 diverse game environments. Configurable mechanics, action interfaces, timing modes and difficulties support controlled studies of model capabilities, learning algorithms, and agent harnesses.
- **An efficient and automated pipeline to collect a high-quality gameplay dataset.** We build a pipeline in which coding agents read game source code and write the behavior-tree controllers to efficiently collect gameplay data in parallel. This produces GAMEDOJO-Data, with 600 hours of high-quality gameplay trajectories covering all 120 encounters at three difficulty levels, training generalist agents and specialist policy network.
- **Extensive experiments on interactive performance, learning, and transfer.** GPT-6.1-Sol wins only 4% of encounters at medium difficulty with time paused and 0% in real time. Our GAMEDOJO-4B approaches frontier performance and achieves the best in real-time. Joint training improves individual tasks, and learned skills generalize across bosses and games. These findings establish GAMEDOJO as a challenging and valuable interactive gameplay benchmark for the community.

2 Related Work

Interactive decision-making and control. Interactive intelligence requires agents to connect perception, decisions, and actions through feedback. Policy learning spans optimization and sequence modeling (Schulman et al., 2017; Chen et al., 2021), while Gato and NitroGen share a policy across tasks and embodiments (Reed et al., 2022; Magne et al., 2026). ReAct and Reflexion organize reasoning and memory around feedback (Yao et al., 2023; Shinn et al., 2023), showing how harness design shapes interaction. SayCan and RT-2 connect language with robotic skills and

Table 1: Comparison of related benchmarks. Games counts supported environments. Tasks counts benchmark objectives or boss encounters; Config. denotes configurable scenario sets. For game source, ✓ denotes fully available source and state access, ◗ partial availability, and ✗ closed-source games. Modern AAA indicates whether the benchmark includes AAA and modern commercial titles or tasks based on them; GAMEDOJO uses independently reconstructed encounters. Core capabilities summarizes task demands and interaction coverage.

Benchmark	Games	Tasks	Game source	Modern AAA	Core capabilities
General game benchmarks					
VideoGameBench (Zhang et al., 2025)	23	23	✗	✗	Navigation, puzzles, combat, memory
Orak (Park et al., 2026)	12	12	◗	✓	Navigation, puzzles, combat, memory
GameVerse (Zhang et al., 2026c)	15	15	◗	✓	Navigation, puzzles, combat, memory
OmniGameArena (Lin et al., 2026)	12	12	✓	◗	Navigation, combat, cooperation
GameWorld (Ouyang et al., 2026a)	34	170	✓	✗	Navigation, puzzles, combat, memory
PlayerBench (Jiang et al., 2026)	50	287	◗	✗	Navigation, puzzles, combat, memory
Adversarial environments					
Gym-μRTS (Huang et al., 2021)	1	Config.	✓	✗	1: real-time strategy
ViZDoom (Kempka et al., 2016)	1	Config.	✓	✗	1: first-person shooting
EvoMan (de França et al., 2019)	1	8	✓	✗	1: action platforming
SoulsGym (Schuck, 2023)	1	1	✗	✓	1: action adventure and role-playing
DSLE (Gezgin et al., 2026)	1	22	✗	✓	1: action adventure and role-playing
GAMEDOJO (Ours)	**20**	**120**	✓	✓	**9 interaction patterns** (Table 2)

actions (Ichter et al., 2023; Zitkovich et al., 2023). Game-TARS, SIMA-2 and GameScaling extend generalist control to virtual worlds (Wang et al., 2025; SIMA Team, 2025; Zhang et al., 2026a). Our testbed evaluates specialist policy networks and agents built on general-purpose language models, alongside their algorithms and harnesses.

Game benchmarks and combat. Games provide repeatable challenges for evaluating interactive capabilities (Zhang et al., 2026d). VideoGameBench, Orak, GameVerse, and GameWorld cover diverse games and objectives (Zhang et al., 2025; Park et al., 2026; Zhang et al., 2026c; Ouyang et al., 2026a), while Gym-μRTS and ViZDoom provide configurable strategy and shooter environments (Huang et al., 2021; Kempka et al., 2016), EvoMan studies generalist strategies across boss encounters in one game (de França et al., 2019), and SoulsGym and DSLE focus on boss encounters in commercial games (Schuck, 2023; Gezgin et al., 2026). Video-Arena examines state tracking from gameplay videos (Zhang et al., 2026b). Other focused work covers arcade fighting (Palmas, 2022; Li et al., 2024) and visual combat control in 3D action RPGs (Chen et al., 2025). GAMEDOJO provides independently reconstructed boss encounters with configurable mechanics and difficulty across 20 games in nine interaction patterns.

Training in virtual environments. Virtual environments make repeated interaction practical for studying how agents learn. ALE provides a shared suite of Atari games (Bellemare et al., 2013), while AlphaGo combines learning with search in Go and AlphaStar uses multi-agent learning for competitive play in StarCraft2 (Silver et al., 2016; Vinyals et al., 2019). VPT learns a Minecraft policy from video with inferred action labels (Baker et al., 2022). OpenAI Gym standardizes environment interfaces (Brockman et al., 2016), Procgen tests policy generalization across generated levels (Cobbe et al., 2020), and Isaac Gym enables parallel simulation for robot learning (Makoviychuk et al., 2021). GameScaling studies data, diversity, and model scaling in gameplay pretraining (Zhang et al., 2026a), while Player-1 combines imitation and reinforcement learning in synthetic environments (Ouyang et al., 2026b). GameHorizon Suite provides gameplay data and evaluation at multiple timescales (Wang et al., 2026b). Our environments expose states, rewards, and mechanics for parallel training and standardized evaluation at fixed difficulty settings.

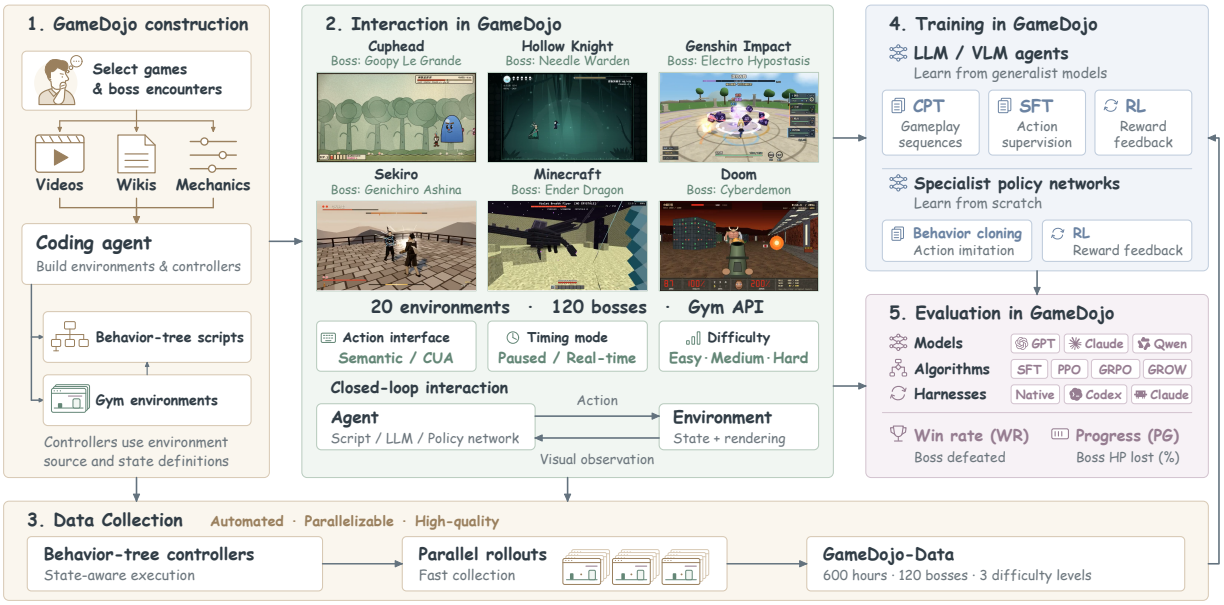


Figure 2: Overview of GAMEDOJO. (1) Human-selected games and bosses guide coding agents that build environments and behavior-tree controllers from videos, wikis, and mechanics descriptions. (2) Scripts, LLM agents, and policy networks interact through configurable action interfaces, timing modes, and difficulty settings. (3) Parallel rollouts produce high-quality GAMEDOJO-Data efficiently without any manual gameplay. (4) The suite supports learning from generalist models and training specialist policies from scratch. (5) Shared tasks evaluate model capabilities, algorithms, and agent harnesses.

3 GAMEDOJO Environments

3.1 Task Definition and Notation

Each task starts at a boss encounter, with the goal of defeating the boss. At decision step t, the environment renders its internal state s_t into a visual observation o_t. The agent selects an action $a_t \sim \pi(\cdot \mid o_t, h_t)$ from its observation and optional history $h_t = (o_0, a_0, \ldots, a_{t-1})$. The action interface executes a_t, and the environment advances to s_{t+1} under the selected timing mode and renders the next observation. This loop requires the agent to recognize attack cues, anticipate the boss's behavior, and adjust its strategy and control in response to feedback of the boss and environment.

3.2 GAMEDOJO Construction

GAMEDOJO contains 120 boss encounters across 20 environments: 10 environments and 60 encounters each in 2D and 3D. We select representative games and bosses as references, covering the nine main interaction patterns in Table 2, from action platforming and bullet-hell dodging to first-person shooting and racing. Differences in viewpoints, movement systems, and combat mechanics call for different strategies and forms of control. Table 1 compares benchmarks by environment count, task count, source availability, and core interactions. General game benchmarks combine navigation, puzzle solving, combat, and memory in broader tasks. The specialist adversarial environments in the table each focus on one interaction family in a single game. GAMEDOJO uses individual boss encounters as a common task unit across nine interaction patterns.

For each selected reference game and boss in Table 2, we prepare the gameplay videos, wikis, and mechanics descriptions. The videos show attack cues, action timing, and phase changes; the wikis and mechanics descriptions detail hit rules, resource constraints, and other mechanics. The agent first uses gameplay videos, wikis, and mechanics descriptions to identify the core

elements to retain, define simplifying assumptions, and specify implementation parameters. It then implements the encounters as lightweight, standalone and high-quality environments by Claude-Opus-5.5 (Anthropic, 2026b) with Claude Code at `max` reasoning effort.

3.3 Environment Details

Each GAMEDOJO encounter is wrapped as a Gym environment (Brockman et al., 2016) and uses a fixed rate of 60 Hz for both rendering and simulation ticks. The wrapper exposes `reset()` to start an episode and return the initial observation, and `step(action)` to apply an action and return the next observation, reward, and episode termination status. GAMEDOJO has the environment source code and can expose internal states, observations, controls, and mechanics parameters as needed, or customize resets, simulation timing, and rewards. In principle, GAMEDOJO enables developers to conduct any custom experiment. The standard GAMEDOJO evaluation uses two action interfaces, two timing modes, three difficulty settings, and a common reward definition.

Action interfaces. **CUA** accepts keyboard and mouse button presses and releases, as well as mouse movement. The model outputs a structured action chunk a_t that the interface parses into low-level device commands (Tan et al., 2025a; Lin et al., 2026). Each chunk covers 200 ms and uses the following format.

$$a_t = \texttt{<|action_start|>}\ \Delta x_t\ \Delta y_t\ \Delta z_t;\ K_t^{(1)};\ \ldots;\ K_t^{(6)}\ \texttt{<|action_end|>}, \tag{1}$$

where $(\Delta x_t, \Delta y_t) \in [-1000, 1000]^2$ are discretized relative mouse moves, $\Delta z_t \in [-5, 5]$ is the scroll delta, and each $K_t^{(j)}$ contains 0-4 keys or mouse buttons held for 33 ms.

Semantic maps parameterized primitives such as movement, attacks, dodges, and skills to low-level controls. For example, the model can issue a single 200 ms action that combines moving left, jumping, and shooting.

$$a_t = \texttt{\{"move": "left", "vert": "jump", "attack": "shoot 135"\}}. \tag{2}$$

Both interfaces share the same visual observations and task objectives so that we can compare performance across levels of action abstraction.

Timing modes. In **real-time mode**, the simulation continues during inference. In **time-paused mode**, it freezes during each decision and advances after receiving the action. Both interfaces use a standard action interval of 0.2 seconds. Each action advances 0.2 seconds of simulation time in time-paused mode; in real-time mode, inference latency affects how often the agent acts. Two modes compare decision-making and execution under different time constraints.

Difficulty and reward. Each encounter has fixed Easy, Medium, and Hard parameter sets, calibrated as described in Section 3.4. All encounters use the same reward definition: given the boss's remaining health H_t in state s_t and its total health $H_{\max}$ in the current configuration, the step reward is $r_t = (H_t - H_{t+1})/H_{\max}$. Evaluation reports average win rate and average progress: whether the boss is defeated and the fraction of its health removed, respectively (Section 4.1).

3.4 Script Design and Dataset Collection

We use behavior-tree controllers that read internal environment state as reference policies for calibrating difficulty. We give Claude-Opus-5.5 (Anthropic, 2026b) with Claude Code the environment source code, state information, and task instructions, and have it write a behavior tree to control the player character in each encounter. We add a fixed 200 ms delay to the controller's responses to approximate human reaction time.

We test all three configurations for each encounter with the same script and tune the parameters

Table 2: Task coverage: nine interaction patterns and reference games used in environment design. The grid shows one boss encounter from each of the 20 reconstructed environments.

Interaction pattern	Reference games
Action platforming	Hollow Knight; Kirby; Mega Man; Super Mario 64
Run-and-gun combat	Cuphead
Fighting	The King of Fighters '97
Top-down combat	Battle City; Soul Knight; The Binding of Isaac
Bullet-hell dodging	Touhou: Embodiment of Scarlet Devil
First-person shooting	Doom; Quake II; Hexen II
Action adventure and role-playing	Sekiro; Genshin Impact; Tomb Raider I; The Legend of Zelda: Ocarina of Time
Sandbox survival combat	Minecraft; Terraria
Racing	Racing environment



against target defeat times: about 20 seconds on Easy, about 60 seconds on Medium, and several minutes on Hard. At higher difficulties, bosses deal more damage, have more health and skills, and are more likely to initiate attacks with less reaction time.

Prior gameplay datasets have relied heavily on human recordings, which can include failed attempts and actions unrelated to the target task. These lower-quality demonstrations can hurt imitation learning. On Q*bert, behavior cloning on the top-scoring 5% of human episodes outperformed training on the full dataset (Kanervisto et al., 2020). Cleaning such recordings at scale takes substantial manual effort. Yue et al. (Yue et al., 2026) manually review flagged clips and spot-check additional recordings, a process that can still miss low-quality segments.

We collect GAMEDOJO-Data by running the behavior-tree controllers in parallel. Once the controllers are in place, adding simulator instances increases collection throughput without recruiting more players. Each controller targets a specific boss, so we can record demonstrations aligned with the encounter objective and verify episode outcomes from the game state. We collected 600 hours of trajectories for fine-tuning across all 120 encounters and three difficulty levels, averaging 100 minutes per encounter and difficulty.

Table 3: Frontier-model and baseline performance at medium difficulty. Entries report WR / PG (%) over 120 bosses, averaging three episodes per boss for models and five for the script. Average is the mean across Semantic and CUA. The best and second-best model results per metric and column are in **bold** and underlined.

Medium Difficulty	Time-paused (300 steps, 1 gameplay min)			Real-time (3 physical min)		
Model	Semantic	CUA	Average	Semantic	CUA	Average
Baselines						
Idle	0.0 / 0.3	0.0 / 0.3	0.0 / 0.3	0.0 / 0.3	0.0 / 0.3	0.0 / 0.3
Random	0.1 / 8.5	0.0 / 6.8	0.1 / 7.7	0.0 / 8.6	0.0 / 6.8	0.0 / 7.7
Boss-specific script	19.8 / 86.5	29.0 / 90.3	24.4 / 88.4	99.2 / 99.7	99.8 / 100.0	99.5 / 99.9
Closed-source models						
GPT-6.1-Sol	**5.8** / **45.0**	**3.3** / **44.6**	**4.6** / **44.8**	**0.0** / **2.5**	**0.0** / **2.9**	**0.0** / **2.7**
GPT-6-Luna	0.0 / 23.1	0.0 / 28.1	0.0 / 25.6	0.0 / 1.8	0.0 / 1.5	0.0 / 1.6
Claude-Opus-5.5	0.0 / 32.0	0.0 / 30.4	0.0 / 31.2	0.0 / 1.2	0.0 / 1.2	0.0 / 1.2
Claude-Haiku-5.5	0.0 / 16.3	0.0 / 15.1	0.0 / 15.7	0.0 / 1.2	0.0 / 1.3	0.0 / 1.3
Claude-Haiku-4.5	0.0 / 9.6	0.0 / 9.0	0.0 / 9.3	0.0 / 0.8	0.0 / 0.6	0.0 / 0.7
Gemini-3.8-Flash	0.0 / 31.1	0.8 / 21.0	0.4 / 26.1	0.0 / 1.8	0.0 / 1.8	0.0 / 1.8
Open-source models						
DeepSeek-V4.1-Flash	0.0 / 20.6	0.0 / 20.6	0.0 / 20.6	0.0 / 1.0	0.0 / 1.0	0.0 / 1.0
GLM-5.3-Flash	0.0 / 18.9	0.0 / 23.4	0.0 / 21.2	0.0 / 0.4	0.0 / 0.5	0.0 / 0.5
MiniMax-M3	0.0 / 24.3	0.0 / 25.4	0.0 / 24.8	0.0 / 0.5	0.0 / 0.5	0.0 / 0.5
Qwen3.5-35B-A3B	0.0 / 12.7	0.0 / 10.3	0.0 / 11.5	0.0 / 0.6	0.0 / 0.5	0.0 / 0.6
Qwen3.5-4B	0.0 / 7.7	0.0 / 1.9	0.0 / 4.8	0.0 / 0.7	0.0 / 1.2	0.0 / 1.0
Generalist game model						
GameScaling-9B-S3	-	0.0 / 5.3	0.0 / 5.3	-	0.0 / 1.5	0.0 / 1.5
Gamer-35B-A3B	-	0.0 / 3.3	0.0 / 3.3	-	0.0 / 2.0	0.0 / 2.0

4 Main Experiments

We evaluate GAMEDOJO through both existing capabilities and learning from interaction. We first compare frontier models across comprehensive evaluations, then examine how GAMEDOJO-Data improves agent performance and how specialist policies learn from scratch.

4.1 Experimental Settings

Models. We evaluate eleven frontier models: GPT-6.1-Sol (OpenAI, 2026b), GPT-6-Luna (OpenAI, 2026a), Claude-Opus-5.5 (Anthropic, 2026b), Claude-Haiku-5.5 (Anthropic, 2026a), Claude-Haiku-4.5 (Anthropic, 2025), Gemini-3.8-Flash (Google DeepMind, 2026), DeepSeek-V4.1-Flash (DeepSeek-AI, 2026), GLM-5.3-Flash (Z.ai, 2026), MiniMax-M3 (MiniMax, 2026), and Qwen3.5-35B-A3B and Qwen3.5-4B (Qwen Team, 2026). We also include GameScaling-9B-S3 (Zhang et al., 2026a) and Gamer-35B-A3B (Luo et al., 2026) trained on diverse gameplay trajectories as two generalist game model comparison. We include three baselines: Idle issues no actions, while Random samples actions from the available action space at each decision step. Boss-specific script uses the state-aware controllers written by coding agents in Section 3.4, providing a privileged upper-bound reference. GAMEDOJO-4B denotes Qwen3.5-4B after supervised fine-tuning on GAMEDOJO-Data. Transfer experiments independently fine-tune Qwen3.5-4B on trajectories from specified source bosses. We also evaluate PPO visual policies trained from scratch (Schulman et al., 2017), following

Table 4: Performance across difficulty levels using CUA, with a native-control script reference. Entries report WR / PG (%) over 120 bosses; Average is the mean across difficulty levels. Models use three episodes per boss; the script uses five.

CUA Mode	Time-paused (300 steps, 1 gameplay min)				Real-time (3 physical min)			
Model	Easy	Medium	Hard	Average	Easy	Medium	Hard	Average
Baselines								
Random	9.7 / 19.0	0.0 / 6.8	0.0 / 2.3	3.3 / 9.4	9.4 / 19.2	0.0 / 6.8	0.0 / 2.3	3.1 / 9.4
Boss-specific script	100.0 / 100.0	29.0 / 90.3	0.0 / 32.9	43.0 / 74.4	100.0 / 100.0	99.8 / 100.0	35.8 / 93.2	78.6 / 97.7
Closed-source models								
GPT-6.1-Sol	**58.3** / **78.1**	3.3 / **44.6**	**0.0** / **13.2**	20.6 / **45.3**	2.5 / 13.7	**0.0** / 2.9	**0.0** / 0.7	0.8 / 5.8
GPT-6-Luna	25.8 / 58.6	0.0 / 28.1	0.0 / 8.2	8.6 / 31.6	0.0 / 7.1	0.0 / 1.5	0.0 / 0.5	0.0 / 3.0
Open-source models								
GameDojo-4B	52.2 / 76.7	**11.9** / 40.3	**0.0** / 11.8	**21.4** / 42.9	**6.4** / **26.3**	**0.0** / **8.2**	**0.0** / **2.5**	**2.2** / **12.3**
Qwen3.5-4B	0.3 / 6.7	0.0 / 1.9	0.0 / 0.6	0.1 / 3.0	0.0 / 2.8	0.0 / 1.2	0.0 / 0.4	0.0 / 1.5
Generalist game models								
GameScaling-9B-S3	4.2 / 16.2	0.0 / 5.3	0.0 / 1.8	1.4 / 7.8	0.0 / 5.8	0.0 / 1.5	0.0 / 0.5	0.0 / 2.6
Gamer-35B-A3B	4.2 / 12.9	0.0 / 3.3	0.0 / 1.0	1.4 / 5.7	0.8 / 6.6	0.0 / 2.0	0.0 / 0.5	0.3 / 3.1

DSLE (Gezgin et al., 2026).

Tasks and interaction. Agents must avoid death and deal as much damage as possible to the boss within a fixed interaction budget. Time-paused episodes allow 300 steps and can take tens of minutes end to end. Real-time episodes allow 180 s including inference. Generalist models receive the two most recent frames and actions, while specialist policy networks observe only the current frame.

Reasoning budget. Language-model agents use `low` reasoning where supported. Each model retains the same reasoning setting across all four interaction conditions. The supplementary material records the settings and output limits.

Metrics. We report average win rate (WR) and average progress (PG). For case i, let $w_i = 1$ if the boss is defeated and 0 otherwise; progress is the fraction of total boss health removed. For a fixed model, interface, timing mode, and difficulty, we average over all N cases:

$$\mathrm{WR} = \frac{1}{N}\sum_{i=1}^{N} w_i, \qquad \mathrm{PG} = \frac{1}{N}\sum_{i=1}^{N} \frac{\Delta H_i}{H_i^{\mathrm{total}}}. \tag{3}$$

Here, ΔH_i is the boss health removed and H_i^{total} is its total health. WR measures how often agents defeat the boss; PG captures partial progress within an encounter. Even when WR is zero, PG distinguishes models by how far they progress through an encounter. All cases receive equal weight and difficulty levels are reported separately. Tables report WR / PG in percentages; training curves show episode progress in $[0, 1]$. More details are in supplement material.

4.2 Frontier Models across Interaction Conditions

We begin by evaluating frontier models on all 120 bosses in GameDojo at medium difficulty. Each of the eleven models runs under time-paused Semantic, time-paused CUA, real-time Semantic, and real-time CUA conditions, yielding 48 model configurations. Table 3 groups results by timing mode and interface, with WR / PG over all 120 bosses.

Strong models make progress but rarely win. In paused mode, GPT-6.1-Sol and Claude-Opus-5.5

achieve the highest average PG at 44.77% and 31.23%, respectively, compared with 7.66% for Random, but they only win 4.6% and even 0%, showing the difficulty to completely beat a boss. GPT-6.1-Sol survives until the end in 47.6% and 1.7% of encounters in time-paused and real-time settings, showing that it can avoid boss attacks when having time to reason. It still cannot match the scripts' ability to dodge precisely while maximizing damage output, which explains the large gap in scores. The results show that GAMEDOJO remains far from saturation even in paused mode.

Weaker models perform close to or below Random. Claude-Haiku-4.5 achieves 9.28% average PG in paused mode, only slightly above Random's 7.66%, with zero WR. Under paused CUA, Qwen3.5-4B reaches just 1.94% PG, below Random's 6.81%. Even GameScaling-9B-S3, a generalist model pretrained on gameplay trajectories (Zhang et al., 2026a), reaches only 5.28% PG and zero WR in the same setting. Boss encounters remain challenging for smaller generalist models, even for those already trained on diverse gameplay.

Most models make similar progress with Semantic and CUA. In paused mode, the median absolute PG gap between the two interfaces is 1.6 percentage points across the eleven frontier models. Gemini-3.8-Flash reaches 31.1% PG with Semantic and 21.0% with CUA, while GLM-5.3-Flash reaches 23.4% with CUA and 18.9% with Semantic.

All models fall below Random in real time. In real time, all evaluated models score below Random in PG and achieve zero WR. GPT-6.1-Sol's average PG drops from 44.77% to 2.70%, while Opus-5.5 drops from 31.23% to 1.19%. In a medium-difficulty CUA diagnostic with one episode per boss, switching from time-paused to real-time interaction reduces the mean number of recorded decisions per episode from 222.3 to 6.7 for GPT-6.1-Sol and from 229.1 to 5.4 for Claude-Opus-5.5. Both fall well below Random. GameScaling-9B-S3 also reaches only 1.50% PG. The sharp decline across both interfaces shows that acting effectively while the environment continues to change remains a major challenge. The state-aware boss-specific script reaches 99.83% WR and 99.99% PG, showing the large gap to strong task-specific control.

4.3 Learning from Gameplay Trajectories

After assessing frontier capabilities, we examine how trajectory supervision improves interactive performance. We obtain GAMEDOJO-4B by supervised fine-tuning Qwen3.5-4B on 240 hours sampled uniformly from GAMEDOJO-Data: 40 minutes for each boss at each of three difficulty levels.

Table 4 shows large gains from trajectory supervision across all three difficulty levels. In paused mode, GAMEDOJO-4B raises average PG from 3.04% to 42.92% and average WR from 0.09% to 21.39%. GAMEDOJO-4B reaches performance comparable to GPT-6.1-Sol: leading 0.83% in WR while losing 2.36% in PG, respectively. In real time, GAMEDOJO-4B achieves the best performance among all models at every difficulty level, reaching 26.26%, 8.18%, and 2.48% on easy, medium, and hard encounters. Across the three difficulty levels and both timing modes, the share of recorded decisions that pass CUA action format validation rises from 8.94%–11.88% before SFT to 99.96%–99.98% after SFT. These gains show that scripted gameplay trajectories therefore provide a promising source of training data, and offers an effective way to improve interactive agents.

4.4 Learning with Specialist Policies

Specialist policy networks offer another view of task difficulty by learning a mapping from visual observations to actions directly from environment feedback. We use Terraria and Doom as representative 2D and 3D environments. For each environment, we train six separate single-boss PPO policies and one shared policy on a mixture of all six bosses. Both training and evaluation use easy difficulty, time-paused interaction. Outputs use each game's Semantic interface; rewards and action intervals follow GAMEDOJO.

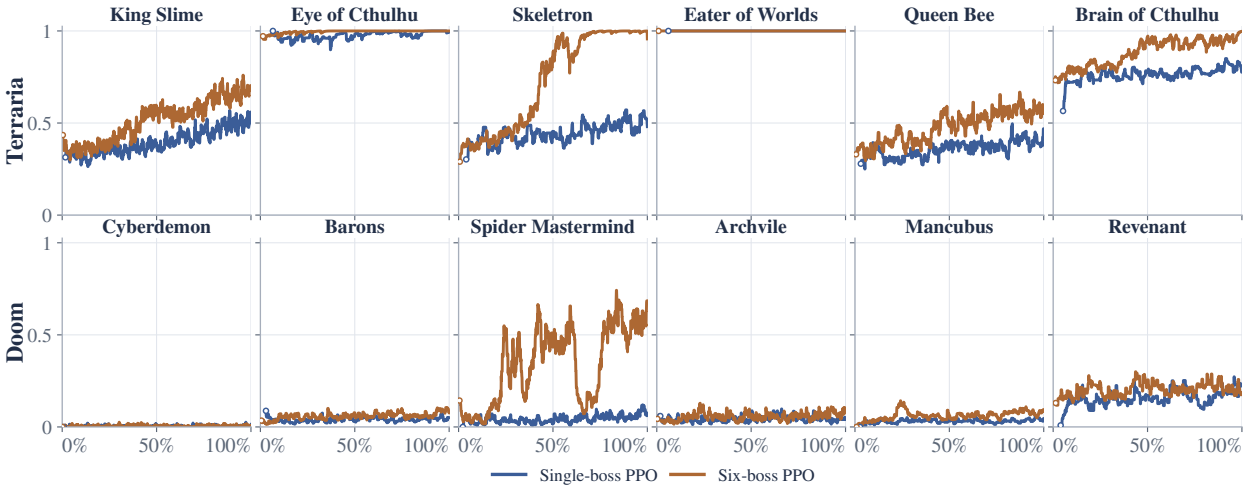


Figure 3: PPO learning with single-boss and six-boss training. Terraria and Doom occupy the top and bottom rows, respectively. Blue denotes a policy trained on the plotted boss, and orange denotes a policy trained jointly on all six bosses in the same game. Both settings allocate 100K steps per boss (600K steps for the joint policy). The axes show training progress and the fraction of boss health removed.

Boss encounters pose distinct learning challenges. Figure 3 shows different learning curves across bosses even with the same initialization. In Terraria, PG approaches 100% on Eye of Cthulhu and Eater of Worlds, while King Slime and Queen Bee approache 50%. In Doom, PG on Cyberdemon stays near 0%, and Barons, Archvile, and Mancubus remain below approximately 10% late in training. These differences reveal a range of encounter difficulties under a shared embodiment: even the easy configuration remains challenging for specialist policies trained with PPO.

5 Further Analysis

We extend the evaluation by examining the difficulty separation, the generalization across the embodiment, boss and game, and the comparison of separate and joint training.

5.1 Performance across Difficulty Levels

We examine the difficulty gradient across easy, medium, and hard configurations on the same tasks. Table 4 compares different models on all 120 bosses through CUA.

Performance drops consistently with difficulty. PG decreases from easy to medium to hard for every evaluated model in both timing modes. For example, GAMEDOJO-4B's paused PG falls from 76.70% to 40.32% and 11.75% and GPT-6.1-Sol falls from 78.08% to 44.59% and 13.17%. All reported model WRs on hard are zero in both modes, showing that the difficulty gradient extends to encounters that remain unsolved even after trajectory supervision.

Real-time interaction remains demanding even on easy. On easy encounters, switching to real time reduces GPT-6.1-Sol's PG from 78.08% to 13.65% and GAMEDOJO-4B's from 76.70% to 26.26%. On hard encounters, the highest real-time PG among learned agents is only 2.48%, close to Random's 2.31%. The boss-specific script reaches 35.83% WR and 93.23% PG under the same 180-second budget, exposing a large gap to the privileged reference. Effective control under continuous environment changes remains challenging across difficulty levels.

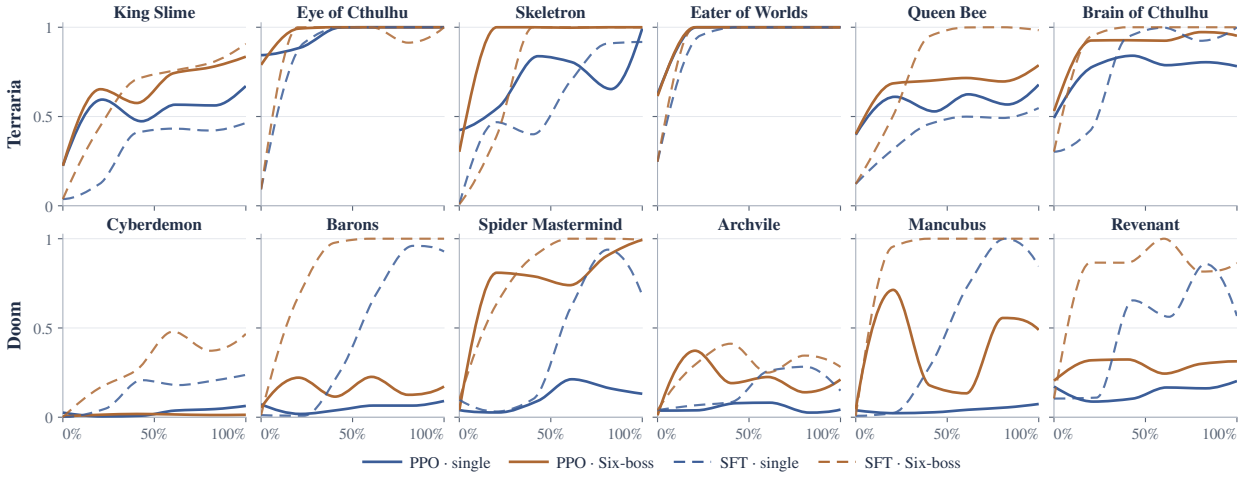


Figure 4: Effect of joint training on individual bosses. For each target boss in Terraria and Doom, we compare training on that boss alone (blue) with joint training on all six bosses in the same game (orange). Solid curves show PPO evaluated with greedy argmax, and dashed curves show SFT. Each curve interpolates six evaluations from 0% to 100% of training in 20% increments. The vertical axis shows the fraction of boss health removed.

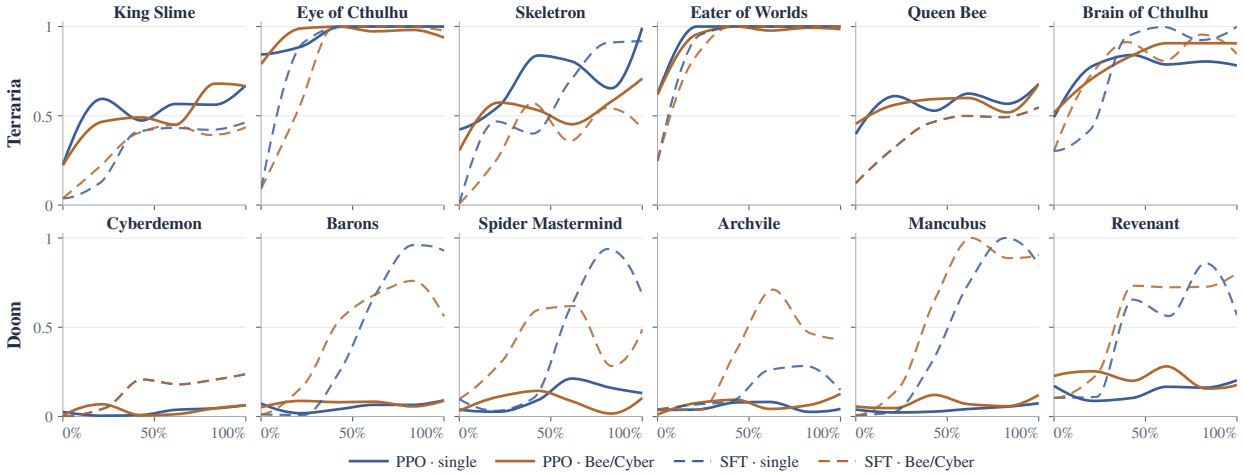


Figure 5: Transfer from a fixed training boss. For each target boss, we compare training on that boss (blue) with training on a fixed source (orange), Queen Bee in Terraria and Cyberdemon in Doom. Solid curves show PPO evaluated with greedy argmax, and dashed curves show SFT. The curves interpolate evaluations every 20% of training from initialization to completion. The fixed-source panels are shown for reference. The transfer averages cover only the five other bosses in each game.

5.2 Learning and Transfer across Bosses

We compare six-boss joint training and fixed-source transfer in Terraria and Doom using PPO and SFT. The joint and single-boss runs use the same training budget per boss within each method (100K online steps for PPO and 2 hours of demonstrations for SFT).

Joint training improves mean progress on individual bosses. At the final checkpoint in Figure 4, six-boss joint PPO raises mean PG over separate training by 7.51 percentage points in Terraria and 26.46 in Doom. Across all twelve bosses, joint PPO raises mean PG from 47.7% to 64.7%. Joint SFT raises it from 69.5% to 87.5%.

Single-boss training transfers within each game. We train on Queen Bee in Terraria or Cyberdemon in Doom and evaluate on the other five bosses in the same game (Figure 5). From initialization to the final checkpoint, fixed-source PPO raises mean PG by 34.96 percentage points in Terraria and 4.81 in Doom, going from 28.3% to 48.2% across the ten held-out bosses. The larger joint-training

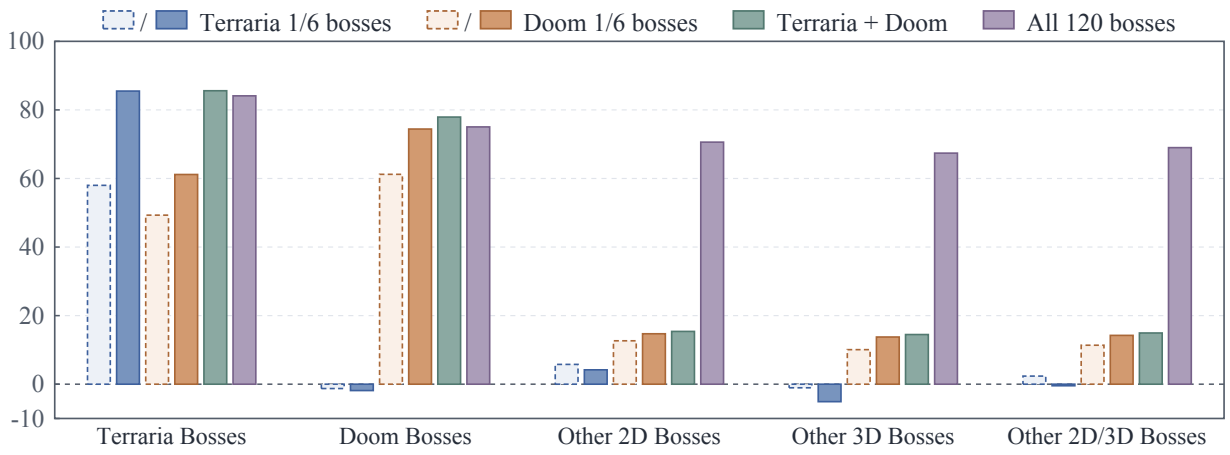


Figure 6: Evaluation on the performance of 6 models on 5 boss sets. Mean progress gains (percentage points) over Qwen3.5-4B after SFT on GAMEDOJO-Data, evaluated with time-paused CUA on easy difficulty. Training uses one or six bosses from Terraria or Doom, both games, or all 120 bosses (GAMEDOJO-4B); single-boss sources are Queen Bee and Cyberdemon, respectively. The five target groups contain 6, 6, 54, 54, and 108 bosses; the last three exclude Terraria and Doom. All settings use the same 2-hour training data per boss.

gain occurs in Doom, and the larger fixed-source gain occurs in Terraria. Across the same ten bosses, fixed-source SFT raises mean PG from 9.5% to 68.7%, finishing 6.8 percentage points below direct training on each target boss.

5.3 Transfer across Games

Fixed-source SFT reaches 73.65% mean PG on the other five Terraria bosses and 63.81% on the other five Doom bosses (Figure 5). We fine-tune Qwen3.5-4B on GAMEDOJO-Data from one or six bosses in Terraria or Doom, or twelve bosses together. GAMEDOJO-4B provides the all-boss comparison. Figure 6 reports PG gains over Qwen3.5-4B for six model settings on five boss sets.

Joint training yields the largest mean PG gains. Combining both games extends these gains beyond the source environments: joint SFT outperforms the four single-game variants in all five evaluation sets. PG improves by 15.39 and 14.48 percentage points on unseen 2D and 3D games, respectively, averaging 14.94 points over their 108 bosses.

Transfer depends on the source environment. Both Doom-only models gain across all five groups, including unseen 2D games. Terraria-only models gain 4.19–5.77 points on other 2D games but lose 1.05–5.10 points on other 3D games. Adding Terraria bosses improves within-game performance without removing this directional difference, highlighting the value of diverse source environments for broader transfer.

6 Conclusion and Limitations

Conclusion. We present GAMEDOJO, a controllable and challenging interactive benchmark of 120 boss encounters across 20 games, together with the 600-hour GAMEDOJO-Data, the automatic data collection pipeline and the trajectory-supervised GAMEDOJO-4B. These resources support controlled studies of model capabilities, learning algorithms, and agent harnesses for interactive intelligence.

Limitations. Reconstructing encounters from videos, wikis and mechanics descriptions may omit implicit rules and visual details of the original games. Evaluation therefore measures performance in the reconstructed tasks; transfer to the original games and physical environments requires extra evaluation in those settings.

Copyright and use. All game titles, characters, artwork and other intellectual property referenced on this page belong to their respective owners and publishers. GameDojo is not affiliated with, endorsed or sponsored by any of them. The encounters are independent reconstructions made for non-commercial scientific research only; game covers and screenshots are shown solely to identify the reference titles. During reconstruction, the models used public gameplay videos, wikis and descriptions of game mechanics as references and had no access to the original games' source code. We did not reverse engineer or crack the original games. GameDojo research materials are for non-commercial use.

References

Anthropic. Introducing Claude Haiku 4.5, 2025. Official release, accessed November 15, 2026.

Anthropic. Introducing Claude Haiku 5.5, 2026a. Official release or model documentation, accessed November 9, 2026.

Anthropic. Introducing Claude Opus 5.5, 2026b. Official release or model documentation, accessed November 9, 2026.

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. In *Advances in Neural Information Processing Systems*, volume 35, pages 24639–24654, 2022.

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. doi: 10.1613/jair.3912.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016. arXiv:1606.01540.

Capcom. Mega Man. Video game, 1987.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097, 2021.

Peng Chen, Pi Bu, Yingyao Wang, et al. CombatVLA: An efficient vision-language-action model for combat tasks in 3D action role-playing games. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10919–10928, 2025.

ChillyRoom. Soul Knight. Video game, 2017.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2048–2056. PMLR, 2020.

Core Design. Tomb Raider. Video game, 1996.

Fabricio Olivetti de França, Denis Fantinato, Karine Miras, A. E. Eiben, and Patricia A. Vargas. EvoMan: Game-playing competition, 2019. arXiv:1912.10445.

DeepSeek-AI. DeepSeek-V4.1-Flash: Smarter, faster, more efficient, 2026. Official release or model documentation, accessed November 9, 2026.

EA Canada. Need for Speed: Most Wanted. Video game, 2005.

FromSoftware. Sekiro: Shadows Die Twice. Video game, 2019.

Derin Gezgin, Jim O'Connor, Tanner Goodwin, and Gary B. Parker. DSLE: A learning environment for Dark Souls boss encounters, 2026. arXiv:2608.09902. Presented at the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2026).

Google DeepMind. Gemini 3.8 Flash: Model Card, 2026. Official release or model documentation, accessed November 9, 2026.

HAL Laboratory. Kirby's Dream Land. Video game, 1992.

Shengyi Huang, Santiago Ontañón, Chris Bamford, and Lukasz Grela. Gym-μRTS: Toward affordable full game real-time strategy games research with deep reinforcement learning. In *2021 IEEE Conference on Games (CoG)*, 2021.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2023.

id Software. Doom. Video game, 1993. See also Doom II (1994) in the linked collection.

id Software. Quake II. Video game, 1997.

Xirui Jiang, Yuxi Chen, Rui Yang, Feng Luo, Pengcheng Wang, Bingxuan Li, Zihao Zhao, Yushan Li, Han Wang, Manling Li, Yang Zhang, Shiyu Chang, Tong Zhang, and Huan Zhang. Game2Player: Scaling gameplay experience for visual agents through environment–player co-development, 2026. arXiv:2611.24154.

Anssi Kanervisto, Joonas Pussinen, and Ville Hautamäki. Benchmarking end-to-end behavioural cloning on video games. In *2020 IEEE Conference on Games (CoG)*, pages 558–565, 2020. doi: 10.1109/CoG47356.2020.9231600.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2016. doi: 10.1109/CIG.2016.7860433.

Wenzhe Li, Zihan Ding, Seth Karten, and Chi Jin. FightLadder: A benchmark for competitive multi-agent reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 27653–27674. PMLR, 2024.

Mingxian Lin, Shengju Qian, Yuqi Liu, Yi-Hua Huang, Yiyu Wang, Wei Huang, et al. OmniGameArena: A unified UE5 benchmark for VLM game agents with improvement dynamics. *arXiv preprint arXiv:2606.09826*, 2026.

Run Luo, Kuan Zhang, Qi Deng, Yukun Chen, Zhihao Yang, Yufei Zhang, Renrong Hu, Yi Zhang, Guiyu Ma, Tianyu Fan, Longze Chen, Hao Sun, Jiaming Li, Jiaming Yang, Xiaohao Liu, Tao Tan, Minzheng Wang, Hangtian Jia, Shuai Yuan, Lvdi Wang, Xiao Liu, Chao Ma, Yuting Zou, Yinhe Zheng, Xiaobo Xia, Dawei Jiang, and Shaoqing Ren. Gamer: A generalist vision-language-action foundation model for long-horizon real-time game agent. *arXiv preprint arXiv:2611.18397*, 2026.

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 (CVPR)*, pages 21511–21521, 2026.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, et al. Isaac Gym: High performance GPU based physics simulation for robot learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Edmund McMillen and Florian Himsl. The Binding of Isaac. Video game, 2011.

miHoYo. Genshin Impact. Video game, 2020.

MiniMax. MiniMax M3, 2026. Official model documentation, accessed November 14, 2026.

Mojang. Minecraft. Video game, 2011.

Namco. Battle City. Video game, 1985.

Nintendo. Super Mario 64. Video game, 1996.

Nintendo. The Legend of Zelda: Ocarina of Time. Video game, 1998.

OpenAI. GPT-6 Luna Model, 2026a. Official release or model documentation, accessed November 9, 2026.

OpenAI. Introducing GPT-6.1 Sol, 2026b. Official release or model documentation, accessed November 9, 2026.

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

Mingyu Ouyang, Yiran Wang, Siyuan Hu, Xingyilang Yin, Huiqiang Sun, Guangzhi Wang, Junfu Pu, Cheng

Cheng, Honghao Chen, Ying Shan, Mike Zheng Shou, and Hwee Tou Ng. Player-1: Learning to play from synthetic experience, 2026b. arXiv:2610.20584.

Alessandro Palmas. DIAMBRA Arena: A new reinforcement learning platform for research and experimentation, 2022. arXiv:2210.10595.

Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, Byeong-Uk Lee, Jaeyoung Hwang, Jaewoo Ahn, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Pritam Biswas, Yoshi Suhara, Kangwook Lee, and Jaewoong Cho. Orak: A foundational benchmark for training and evaluating LLM agents on diverse video games. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026.

Qwen Team. Qwen3.5: Towards Native Multimodal Agents, 2026. Official release or model documentation, accessed November 9, 2026.

Raven Software. Hexen II. Video game, 1997.

Re-Logic. Terraria. Video game, 2011.

Scott Reed, Konrad Żołna, Emilio Parisotto, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

Martin Schuck. SoulsGym, 2023. Software, version 1.1.0.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. arXiv:1707.06347.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

David Silver, Aja Huang, Chris J. Maddison, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. doi: 10.1038/nature16961.

SIMA Team. Scaling instructable agents across many simulated worlds, 2024. arXiv:2404.10179.

SIMA Team. SIMA 2: A generalist embodied agent for virtual worlds, 2025. arXiv:2512.04797.

SNK. The King of Fighters '97. Video game, 1997.

Studio MDHR. Cuphead. Video game, 2017.

Weihao Tan, Xiangyang Li, Yunhao Fang, Heyuan Yao, Shi Yan, Hao Luo, et al. Lumine: An open recipe for building generalist agents in 3D open worlds. *arXiv preprint arXiv:2511.08892*, 2025a.

Weihao Tan, Wentao Zhang, Xinrun Xu, et al. 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*, pages 58658–58725. PMLR, 2025b.

Team Cherry. Hollow Knight. Video game, 2017.

Team Shanghai Alice. Touhou: Embodiment of Scarlet Devil. Video game, 2002.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019. doi: 10.1038/s41586-019-1724-z.

Dawei Wang, Di Zhao, Jianran Yu, et al. ROAR: Benchmarking LMM agents under real-time constraints in multiplayer action-shooter games, 2026a. arXiv:2610.24282.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Yiran Wang, Xingyilang Yin, Junfu Pu, Guangzhi Wang, Kaifeng Li, Mingyu Ouyang, Huiqiang Sun, Lingen Li, Cheng Cheng, Wangbo Yu, Honghao Chen, Xiaodong Cun, Chi-Man Pun, Zhiguo Cao, and Ying Shan. GameHorizon Suite: Multi-horizon data and evaluation in gameplay, 2026b. arXiv:2609.25001.

Zihao Wang, Xujing Li, Yining Ye, et al. Game-TARS: Pretrained foundation models for scalable generalist multimodal game agents, 2025. arXiv:2510.23691.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

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, 2026. arXiv:2601.04575.

Z.ai. GLM-5.3-Flash: Frontier Intelligence, Flash Cost, 2026. Official release or model documentation, accessed November 9, 2026.

Alex L. Zhang, Thomas L. Griffiths, Karthik R. Narasimhan, and Ofir Press. VideoGameBench: Can vision-language models complete popular video games?, 2025. arXiv:2505.18134.

Kuan Zhang, Yukun Chen, Zhihao Yang, Yue Su, Xiangnan Wu, Zirong Chen, Run Luo, Jinkun Hou, Tao Tan, Yinhe Zheng, and Yiming Li. Scaling in games: Continued pre-training for embodied agents in diverse virtual worlds, 2026a. arXiv:2610.28698.

Kuan Zhang, Yukun Chen, Zhihao Yang, Yue Su, Xiangnan Wu, Zirong Chen, Run Luo, Tao Tan, Yinhe Zheng, and Yiming Li. Video-Arena: A scalable agentic video benchmark for spatiotemporal capability, 2026b. arXiv:2610.14158.

Kuan Zhang, Dongchen Liu, Qiyue Zhao, Jinkun Hou, Xinran Zhang, Qinlei Xie, Miao Liu, and Yiming Li. GameVerse: Can vision-language models learn from video-based reflection? In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 157348–157420. PMLR, 2026c.

Kuan Zhang, Dongchen Liu, Qiyue Zhao, Tianyu Xin, Yue Su, Haisheng Wang, Han Yin, Hongbo Ma, Peize Li, Tianjun Gu, Xiangnan Wu, Xinran Zhang, Yongxuan Li, Zirong Chen, and Yiming Li. Towards generalist game agents: A survey of foundation models in virtual worlds. *Transactions on Machine Learning Research*, 2026d. ISSN 2835-8856.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 2023.

Appendix

A Environment Construction and Design

A.1 Games and Boss Encounters

GAMEDOJO spans nine interaction patterns across ten 2D and ten 3D environments, with six encounters per game.

Table S1: Full GAMEDOJO Environment Catalog. All 20 games (10 2D and 10 3D) and their 6 boss encounters (120 encounters in total).

Game	Viewpoint & Genre	Six Boss Encounters
2D Environments (10 Games, 60 Encounters)		
Hollow Knight	2D Action Platformer	Hollow Vessel, Iron Bulwark, Lumen Moth, Needle Warden, Pale Vessel, Sickle Sisters
Kirby	2D Side-Scrolling Platformer	Kaboola, King Dedede, Kracko, Lololo & Lalala, Poppy Bros. Sr., Whispy Woods
Terraria	2D Sandbox Action	Brain of Cthulhu, Eater of Worlds, Eye of Cthulhu, King Slime, Queen Bee, Skeletron
The King of Fighters '97	2D Fighting	Iori, Orochi, Orochi Chris, Orochi Iori, Orochi Shermie, Orochi Yashiro
Battle City	2D Top-Down Tank Defense	Brick Front, Jungle Ambush, River Delta, River Ice, Steel Fort, Steel Maze
Cuphead	2D Run-and-Gun Boss Rush	Cagney Carnation, Goopy Le Grande, Hilda Berg, Ribby & Croaks, The Root Pack, Wally Warbles
Touhou	2D Vertical Bullet Hell	Cirno, Hong Meiling, Patchouli Knowledge, Remilia Scarlet, Rumia, Sakuya Izayoi
Soul Knight	2D Top-Down Dungeon Crawler	Christmas Treant, Devil's Snare, Goblin Priest, Grand Slime, Varkolyn Leader, Volcanic Sandworm
Mega Man	2D Precision Platformer	Bomb Man, Cut Man, Elec Man, Fire Man, Guts Man, Ice Man
The Binding of Isaac	2D Top-Down Twin-Stick	Duke of Flies, Envy, Famine, Gemini, Larry Jr., Monstro
3D Environments (10 Games, 60 Encounters)		
Minecraft	3D First-Person Sandbox	Ender Dragon, Iron Golem, Skeleton Archer, Warden, Wither, Zombie Leader
Doom	3D First-Person Shooter	Arch-vile, Barons of Hell, Cyberdemon, Mancubus, Revenant, Spider Mastermind
Sekiro	3D Third-Person Sword Action	Demon of Hatred, Genichiro, Guardian Ape, Gyoubu, Isshin, Lady Butterfly
Genshin Impact	3D Third-Person Party Action	Andrius, Electro Hypostasis, Maguu Kenki, Oceanid, Ruin Guard, Signora
Quake II	3D First-Person Shooter	Gladiator, Hornet, Makron, Mutant, Supertank, Tank Commander
Hexen II	3D First-Person Fantasy	Airborne Bone Rider, Axe-Hurling Rider, Blinking Tether Rider, Boar Archer Rider, Orb-Warded Serpent, Star-Volley Sorcerer
Super Mario 64	3D Platformer	Big Bully, Bowser in the Dark World, Eyerok, King Bob-omb, Whomp King, Wiggler
Tomb Raider	3D Third-Person Action	Cannon Quadruped, Giant Lizard, Legless Giant, Lone Gunman, Twin Pistols, Winged Matriarch
Zelda: Ocarina of Time	3D Third-Person Lock-On Action	Armored Spider Queen, Burrowing Fire Dragon, Lava Lizard King, Specter Rider, Twin Flame-Frost Witches, Water-Core Amoeba
Need for Speed: Most Wanted	3D Arcade Racing	Drag, Knockout, Pursuit, Speedtrap, Street Champion, Tollbooth

(a) Battle City (Namco, 1985). In this top-down tank game, terrain shapes the firing lanes. Players line up shots from cover and move before enemy fire cuts off their route.
Encounters. Brick Front (brick cover), Jungle Ambush (concealed approaches), River Delta (water crossings), River Ice (icy lanes), Steel Fort (steel fortifications), Steel Maze (steel-walled corridors).

(b) Cuphead (Studio MDHR, 2017). This run-and-gun game pairs sustained fire with jumps and dashes through patterned attacks. In aerial encounters the player can also change altitude while firing.
Bosses. Cagney Carnation (projectile-spitting flower), Goopy Le Grande (bouncing slime), Hilda Berg (aerial transformations), Ribby & Croaks (frog duo), The Root Pack (vegetable trio), Wally Warbles (airborne bird).

(c) Hollow Knight (Team Cherry, 2017). This 2D action platformer centers on close-range sword combat. Players read windups and use jumps or dashes to get into striking range as an attack ends.
Bosses. Hollow Vessel (vessel duelist), Iron Bulwark (armored heavyweight), Lumen Moth (luminous flier), Needle Warden (needle-wielding duelist), Pale Vessel (pale swordfighter), Sickle Sisters (paired sickles).

(d) Kirby's Dream Land (HAL Laboratory, 1992). This side-scrolling platformer lets players hover to change their height and avoid attacks. Grounded and airborne bosses call for different approaches to getting in range.
Bosses. Kaboola (armed blimp), King Dedede (hammer wielder), Kracko (lightning cloud), Lololo & Lalala (block-pushing duo), Poppy Bros. Sr. (bomb thrower), Whispy Woods (apple-dropping tree).

(e) The King of Fighters '97 (SNK, 1997). In these one-on-one bouts, players manage spacing, block or evade attacks, and counter during recovery.
Bosses. Iori (purple flames), Orochi (screen-wide blasts), Orochi Chris (flame pressure), Orochi Iori (fast swipes), Orochi Shermie (lightning), Orochi Yashiro (grappling).

(f) Mega Man (Capcom, 1987). Players in this 2D action platformer jump over attacks while lining up horizontal shots. Each Robot Master changes when and where the player can safely fire.
Bosses. Bomb Man (arcing bombs), Cut Man (returning cutters), Elec Man (electric beams), Fire Man (flame waves), Guts Man (thrown boulders), Ice Man (ice projectiles).

(g) Soul Knight (ChillyRoom, 2017). This top-down shooter requires players to aim while moving out of projectile lanes. Boss summons and moving hazards can change which part of the arena is safe.
Bosses. Christmas Treant (charging tree), Devil's Snare (projectile-spitting plant), Goblin Priest (spell volleys), Grand Slime (jumping slime), Varkolyn Leader (poison attacks), Volcanic Sandworm (burrowing ambushes).

(h) The Binding of Isaac (McMillen and Himsl, 2011). In this top-down shooter, movement and firing direction are independent. Players can retreat while shooting, but splitting enemies and sweeping attacks can take away the space they need to dodge.
Bosses. Duke of Flies (fly swarm), Envy (splitting enemy), Famine (mounted charges), Gemini (paired attackers), Larry Jr. (segmented pursuit), Monstro (leaps and projectile bursts).

(i) Terraria (Re-Logic, 2011). These 2D sandbox combat encounters require players to use vertical space and fire while moving. The boss's trajectory determines whether the player should keep distance, change elevation, or cross the arena.
Bosses. Brain of Cthulhu (teleporting brain), Eater of Worlds (segmented burrower), Eye of Cthulhu (charging eye), King Slime (leaping and teleporting slime), Queen Bee (dashes and stingers), Skeletron (head and independently attacking hands).

(j) Touhou: Embodiment of Scarlet Devil (Team Shanghai Alice, 2002). In this vertical bullet-hell shooter, players thread gaps in dense barrages while keeping their shots on the boss.
Bosses. Cirno (ice barrages), Hong Meiling (colorful spreads), Patchouli Knowledge (elemental barrages), Remilia Scarlet (scarlet patterns), Rumia (darkness attacks), Sakuya Izayoi (knife volleys).

(k) Doom (id Software, 1993). This first-person shooter combines continuous aiming with strafing around projectiles and moving enemies. The encounters vary how much time and space players have to stay on target.
Bosses. Arch-vile (flame attack), Barons of Hell (paired plasma throwers), Cyberdemon (rocket volleys), Mancubus (spread fireballs), Revenant (homing missiles), Spider Mastermind (rapid gunfire).

(l) Genshin Impact (miHoYo, 2020). These third-person action RPG encounters alternate between avoiding attack patterns and closing in to deal damage. Changing phases, summons, and protected targets alter when the player can attack effectively.
Bosses. Andrius (wolf lunges), Electro Hypostasis (exposed core), Maguu Kenki (phantom sword strikes), Oceanid (water summons), Ruin Guard (spinning automaton), Signora (ice and fire phases).

(m) Hexen II (Raven Software, 1997). In this first-person fantasy action game, players aim at mobile targets while dodging spells and thrown weapons. Flying, mounted, and teleporting enemies change where the player needs to look next.
Bosses. Airborne Bone Rider (aerial rider), Axe-Hurling Rider (thrown axes), Blinking Tether Rider (teleporting tether attacks), Boar Archer Rider (mounted archery), Orb-Warded Serpent (protective orbs), Star-Volley Sorcerer (star volleys).

(n) Minecraft (Mojang, 2011). These first-person sandbox encounters require players to track mobs in three dimensions and control their distance. Some enemies can be engaged at range, while others force the player to reposition before getting a clear attack.
Bosses. Ender Dragon (aerial passes), Iron Golem (heavy melee swings), Skeleton Archer (ranged arrows), Warden (sonic attack), Wither (explosive skulls), Zombie Leader (undead melee pressure).

(o) Need for Speed: Most Wanted (EA Canada, 2005). The racing environment requires players to coordinate steering, braking, and acceleration. Its events change what counts as progress, from holding speed through checkpoints to finishing ahead of a rival or escaping pursuit.
Events. Drag (short sprint), Knockout (last-place elimination), Pursuit (police evasion), Speedtrap (speed at checkpoints), Street Champion (rival race), Tollbooth (timed checkpoints).

(p) Quake II (id Software, 1997). In this first-person shooter, players track moving targets while strafing or breaking line of sight. The encounters range from fast lunging enemies to large machines with sustained fire.
Bosses. Gladiator (railgun shots), Hornet (flying gunship), Makron (heavy weapon fire), Mutant (leaping melee), Supertank (armored gun platform), Tank Commander (armored ranged attacker).

(q) Sekiro (FromSoftware, 2019). This third-person sword combat game rewards deflecting at the moment of impact. Players read attack sequences to decide when to deflect, dodge, or counter.
Bosses. Demon of Hatred (wide fire attacks), Genichiro (sword and bow), Guardian Ape (sweeping attacks), Gyoubu (mounted spear), Isshin (multi-weapon duel), Lady Butterfly (aerial attacks).

(r) Super Mario 64 (Nintendo, 1996). These 3D platforming encounters make position and momentum part of the attack. Players may need to get behind an enemy, redirect it toward a hazard, or land on a moving target.
Bosses. Big Bully (knockback near lava), Bowser in the Dark World (tail-grab throws), Eyerok (eye-bearing stone hands), King Bob-omb (grab-and-throw duel), Whomp King (ground-pound opening), Wiggler (stomp and pursuit).

(s) The Legend of Zelda: Ocarina of Time (Nintendo, 1998). These third-person encounters revolve around finding weak points and acting while they are exposed. Players reposition through arena hazards to reach the right angle or range.
Bosses. Armored Spider Queen (armored arachnid), Burrowing Fire Dragon (emerging from below), Lava Lizard King (lava-arena lizard), Specter Rider (mounted apparition), Twin Flame-Frost Witches (fire and ice attacks), Water-Core Amoeba (waterborne core).

(t) Tomb Raider I (Core Design, 1996). This third-person action adventure requires players to move while firing and keep enemies in view. Grounded and flying opponents differ in how they close distance and pressure the player.
Bosses. Cannon Quadruped (four-legged gunner), Giant Lizard (reptilian attacker), Legless Giant (torso mutant), Lone Gunman (solo shooter), Twin Pistols (dual-pistol shooter), Winged Matriarch (aerial attacker).

Table S1 lists each game, its viewpoint and genre, and all six boss encounters.

A.2 Coding Agent Game Reconstruction

Macro reconstruction workflow. We construct all 20 games from scratch using Claude-Opus-5.5 through Claude Code with `max` reasoning effort. For each target game, we gather public gameplay videos, wiki pages, and mechanics summaries, then run a three stage workflow:

1. *Specification writing*: The coding agent writes a structured mechanics document covering player movement, attacks, cooldowns, resource rules, boss attack phases, hitboxes, visual cues, and win or loss conditions for all six bosses.
2. *Engine and boss implementation*: The agent implements the game simulation, arena geometry, boss state machines, and visual renderer using lightweight assets that keep combat cues readable without copying commercial files.
3. *Three round verification loop*: Every new game passes three test and fix rounds. Round 1 checks complete combat mechanics and boss phase transitions; Round 2 reviews visual clarity, attack warnings, and controls against reference material; Round 3 tests multi seed stability, exact replay, public Gym compliance, and frame rate. Any remaining bugs are fixed and retested before a game enters the benchmark.

Gym packaging and rendering backends. All 20 games are exposed through a single Gymnasium wrapper (`reset(seed, options)` and `step(action)`) that hides three lightweight runtime backends: (1) a shared 2D Canvas engine (`BB.Case`) for 7 classic 2D games (*Hollow Knight*, *Kirby*, *Terraria*, *KOF '97*, *Battle City*, *Cuphead*, and *Mega Man*), (2) standalone browser based JavaScript engines for 12 games using HTML5 Canvas 2D (*Touhou*, *Soul Knight*, *Isaac*), CPU raycasting (*Doom*), or offline Three.js WebGL 3D (*Sekiro*, *Genshin Impact*, *Quake II*, *Hexen II*, *Super Mario 64*, *Tomb Raider*, *Zelda: OoT*, and *NFS: Most Wanted*), and (3) a Python and NumPy simulation engine with headless hardware accelerated ModernGL perspective rendering for *Minecraft*. Visual effects and camera rendering read the simulation state without modifying physics or random number streams. For all 3D games, the wrapper enforces true perspective rendering and raises an error if hardware acceleration is unavailable, preventing software top down fallbacks from entering evaluation or training. The public interface returns only RGB frames and public control descriptions; internal health values, hitboxes, boss move tables, and reward verifiers stay hidden inside the environment wrapper.

Fixed game parameters. Every environment shares the following fixed runtime parameters:

- **Resolution**: 854×480 pixels (16:9), `uint8` RGB array of shape $(480, 854, 3)$.
- **Simulation and rendering clock**: Fixed 60 Hz tick rate (16.67 ms per tick), unified across all 20 games.
- **Decision step**: 200 ms per environment step (5 Hz), advancing 12 simulation ticks divided into 6 equal input substeps of 33.3 ms (2 ticks per substep).
- **Episode limit**: Hard engine timeout of 300 s (18,000 ticks), ending early upon player death or boss defeat on the exact terminal tick.
- **Determinism**: Seeded pseudo random number generators and full engine state hashing guarantee bit exact replay given the same seed, difficulty, and action sequence.

- **Step reward**: Normalized boss health reduction per step, $r_t = (H_t - H_{t+1})/H_{\max}$, with cumulative progress clipped to $[0, 1]$ at episode end.

A.3 Action Spaces

Every environment supports two action interfaces over the same 200 ms decision step: a unified Computer Use Agent (**CUA**) keyboard and mouse interface and a game specific **Semantic** discrete interface.

CUA action format. A CUA action is a single line token sequence specifying mouse motion and six 33.3 ms keyboard substeps for the next 200 ms:

$$\texttt{<|action_start|>}\ X\ Y\ Z\ ;\ K^{(1)}\ ;\ \ldots\ ;\ K^{(6)}\ \texttt{<|action_end|>}$$

Here $X, Y \in [-1000, 1000]$ are integer relative mouse displacements ($X > 0$ right, $Y > 0$ down), and $Z \in [-5, 5]$ is the scroll wheel delta ($Z > 0$ scroll up). Each substep $K^{(m)}$ ($m = 1, \ldots, 6$) lists 0 to 4 simultaneously held controls separated by spaces from the shared 14-control set {`W`, `A`, `S`, `D`, `Space`, `Shift`, `J`, `K`, `L`, `Q`, `E`, `R`, `LB`, `RB`}, where `LB` and `RB` denote left and right mouse buttons. An empty substep releases all keys while keeping its semicolon separator. A key appearing in consecutive substeps (or across adjacent 200 ms packets) remains continuously held rather than triggering a fresh press; a new press fires only on a released-to-held transition. Mouse motion follows one of three per game modes: `None` (mouse motion ignored), `Pointer` (2D crosshair movement normalized to screen coordinates), or `Camera` (3D yaw and pitch camera rotation).

Semantic action format. A Semantic action is a multi-head discrete choice written as a JSON dictionary mapping each action head to one named option, for example `{"move": "left", "vert": "jump", "attack": "shoot 135"}` in *Terraria* or `{"move": "forward", "turn": "left_10", "attack": "fire"}` in *Doom*. Any omitted head defaults to option 0 (`"none"`), and every combination translates deterministically into one 200 ms CUA packet (held across the full step). Across 19 games, heads are defined in `compact_actions.json` such that separate heads never share keys or mouse axes and the total number of combinations stays strictly below 1,000. *Terraria* uses a three-head layout (`move` $\times$ `vert` $\times$ `attack` $= 3 \times 3 \times 19 = 171$) covering 16 directional bow angles at 22.5° increments plus two sword swings. Malformed strings or out of range values in either interface execute a neutral no op packet and increment the format error counter.

Table S2 details both action spaces for all 20 games.

Table S2: Action Spaces Across All 20 Games. For each game, we list the CUA mouse mode, active keyboard and mouse bindings with their gameplay functions, and the Semantic multi-head discrete structure with total action count.

Game	Mouse	CUA Controls (Key: Function)	Semantic Heads & Total Size
Battle City	None	WASD: 4-way tank drive; J/K/Space/LB: fire shell	move(57) × fire(3) = 171
Cuphead	None	WASD: move/aim/crouch; Space: jump/parry; Shift: dash; J/LB: fire; K/RB: EX shot; L: lock aim; Q: swap plane gun/bomb; E: super	move(28) × act(29) = 812
Doom	Camera	WASD: move/strafe; Shift: run; Q/E: turn L/R; J/LB: fire; K/RB: next weapon; L: prev weapon	move(9) × turn(9) × attack(5) = 405
Genshin Impact	Camera	WASD: move; Space: jump; Shift: dodge dash; J: normal attack; LB: charge attack; K/RB: elemental skill; L: burst; Q/E: swap character; R: lock-on	move(9) × action(12) × view(9) = 972
Hexen II	Camera	WASD: move/strafe; Shift: run; Space: jump; L: crouch; Q/E: turn L/R; J/LB: attack; K/RB: next weapon; R: prev weapon	move(9) × turn(9) × pitch(3) × attack(4) = 972
Hollow Knight	None	A/D: move L/R; W/S: up/down slash aim; Space: jump; Shift: dash; J/LB: nail slash; K: focus heal	move(5) × jump(9) × action(22) = 990
Binding of Isaac	None	WASD: 8-way move; J/K/L/Shift: fire tears L/D/R/U; E: place bomb	move(9) × fire(5) × bomb(2) = 90
Kirby	None	A/D: walk/fly L/R; W: float up; S: duck; Space: jump; J/LB: inhale / spit star	mouth(10) × walk(9) × vertical(11) = 990
KOF '97	None	A/D: move/block; W/Space: jump; S: crouch; J/K: light punch/kick; L/Q: heavy punch/kick; Shift: roll; E: MAX mode; R: super helper	direction(28) × button(23) = 644
Super Mario 64	Camera	WASD: move; Space: jump; Shift/RB: crouch/slide/pound; J/LB: punch/dive/grab; K: slow walk; Q/E: rotate camera 45°; R: center camera; L: zoom	move(90) × action(11) = 990
Mega Man	None	A/D: move L/R; Space: variable jump; J/LB: buster fire; Q/E: prev/next weapon	move(7) × jump(15) × fire(4) = 420
Minecraft	Camera	WASD: move/strafe; Space: jump; Shift: sprint; J/LB: sword swing; RB: draw/release bow	move(6) × action(6) × aim(27) = 972
NFS: Most Wanted	None	W/S: throttle/brake; A/D: steer L/R; Space: handbrake; Shift: nitrous; J: slow-mo; K/L: shift up/down; R: reset car	pedal(4) × steer(13) × nitrous(2) × press(6) = 624
Quake II	Camera	WASD: move/strafe; Shift: run; Space: jump; L: crouch; Q/E: turn L/R; J/LB: fire; K/RB: next weapon; R: prev weapon	move(9) × turn(7) × pitch(3) × action(5) = 945
Sekiro	Camera	WASD: move; Space: jump; Shift: dash/dodge; J/LB: slash/thrust; K/RB: guard/deflect; Q: heal gourd; E: lock-on; R: revive	move(12) × act(73) = 876
Soul Knight	Pointer	WASD: 8-way move; J/LB: attack; K/RB: class skill; Q: switch weapon	move(9) × aim(9) × act(4) = 324
Terraria	Pointer	A/D: move L/R; Space: jump; S: drop through platform; J/LB: use weapon; Q: swap sword/bow; E: bow shot at cursor; RB: direct sword swing	move(3) × vert(3) × attack(19) = 171
Tomb Raider	Camera	W/S: forward/back-hop; A/D: tank turn; Q/E: sidestep; Space: jump; Shift: walk; J/LB: fire; K/RB: draw guns; L: 180° roll; R: medipack	move(5) × turn(15) × act(11) = 825
Touhou	None	WASD: 8-way move; Shift: focus slow mode; J/LB: fire; K/RB: bomb; Q/E: switch shot type	move(9) × action(6) = 54
Zelda: OoT	Camera	WASD: move; Space: roll/jump; Shift: lock-on; J/LB: sword slash; K/RB: shield block; L: boss item; Q/E: turn camera 45°; R: reset camera	lock(2) × move(24) × act(20) = 960

A.4 Scripted Controller Design

Source aware controller synthesis. Because GAMEDOJO provides full access to each game's source code, the coding agent writes a dedicated behavior tree controller for every boss encounter. The agent reads the boss state machine, attack timers, projectile positions, player cooldowns, and the game's legal CUA action space, then implements a deterministic policy hook (exposed through `BB.GAME.scripted`, `Module.scripted`, or `mc_scripted.scripted`). Each controller observes live game state strictly in read-only mode, checked before and after every call by verifying that the engine state hash does not change. To keep demonstrations realistic and aligned with learned agents, every scripted controller runs at the same 200 ms decision step (introducing a natural 200 ms reaction delay) and outputs only legal CUA action packets through the public Gym `step` function. In addition, we verify each game's Semantic action space by snapping scripted CUA packets to the nearest Semantic action and confirming that bosses remain winnable.

Difficulty calibration from scripted controllers. We use the verified scripted controllers as a consistent skill anchor to calibrate three difficulty levels per encounter. For each boss, the coding agent tunes numeric difficulty parameters such as boss health, movement speed, attack cooldown, projectile speed, and attack damage, then runs multi seed scripted evaluations until average clear times fall cleanly into three target bands: **Easy** (20–30 s), **Medium** (45–90 s), and **Hard** (150–240 s). This procedure ensures that every difficulty tier reflects a smooth progression in combat length and execution pressure while remaining reliably solvable by a strong policy across random seeds.

Two stage trajectory collection. We collect GAMEDOJO-Data from the scripted controllers using a two stage parallel pipeline across all 120 bosses and all three difficulty tiers:

1. *Headless rollout and filtering*: Parallel workers run the scripted controllers in headless symbolic mode (`time_mode=paused`) across distinct engine seeds to record `actions.jsonl` and `replay.json`. Only winning episodes that pass exact replay state-hash matching and basic engagement checks (movement ratio ≥ 0.05, at least 2 distinct key combinations, and at least 10 visited arena positions) are kept.
2. *Replay rendering and dataset assembly*: Accepted action traces are replayed tick-by-tick through the visual renderer to capture pre-action 854×480 RGB frames at 60 fps, encoded with H.264 (`libx264`, CRF 18, `veryfast`, `yuv420p`). Each video passes strict decode checks (exact frame count matching tick count, SHA-256 verification, and zero corrupt or all-black frames). We remove duplicate action sequences by action-trace hash and partition episodes by `(boss, seed)` with zero cross-split overlap.

The resulting GAMEDOJO-Data corpus contains 44,368 winning episodes totaling 604.6 h: 39,937 training episodes (543.9 h), 2,193 validation episodes (29.7 h), and 2,238 test episodes (31.0 h).

B Training Details

B.1 GAMEDOJO-4B Training Details

Training datasets. For GAMEDOJO-4B (`S1`), we draw a balanced subset from the training split of GAMEDOJO-Data by sampling complete winning episodes until every boss reaches at least 40 minutes at each of the three difficulty tiers ($120 \times 3 \times 40$ min $= 240$ h nominal, yielding 17,859 complete episodes and 244.9 h total). To study cross-game transfer (Figure 6 in the main paper) and own-boss learning curves (Figure 4), we extract matched single-boss (2 h each), six-boss (12 h each), and twelve-boss (24 h) subsets from the exact same balanced pool using identical episode selection rules. Table S3 summarizes the dataset statistics and compute budgets for GAMEDOJO-4B and every SFT variant.

Table S3: GAMEDOJO-4B and SFT Variant Training Summary. All runs fine-tune Qwen3.5-4B for 1 epoch with AdamW, peak learning rate 1×10^{-5} (cosine decay to 1×10^{-6}), and 4.0 s (up to 20 step) causal windows. In the main paper, S1 supports Table 4 and S1–S6 support Figure 6; S5, S6, S2, S3, and the 10 additional single-boss 2 h runs (S7–S16) support Figure 4.

ID	Training Bosses	Episodes	Hours	4 s Windows	CUA Actions	Batch	Steps	GPUs	A800 GPU-h
S1 (GAMEDOJO-4B)	All 120 bosses (20 games)	17,859	244.9	229,423	4,415,457	512	449	48	344.2
S2	Terraria (all 6 bosses)	933	12.2	11,488	220,189	64	180	16	16.0
S3	Doom (all 6 bosses)	877	12.3	11,455	221,089	64	179	16	16.2
S4	Terraria + Doom (12 bosses)	1,810	24.5	22,943	441,278	64	359	16	33.2
S5	Terraria: Queen Bee (1 boss)	163	2.0	1,902	36,527	64	30	16	4.5
S6	Doom: Cyberdemon (1 boss)	152	2.1	1,925	37,090	64	31	16	4.5
S7–S11 (5 runs)	Other 5 Terraria bosses (each 1 boss)	∼155/run	∼2.0/run	1,906–1,931	36,390–36,949	64	30–31	16	∼4.5/run
S12–S16 (5 runs)	Other 5 Doom bosses (each 1 boss)	∼150/run	∼2.0/run	1,882–1,932	36,520–37,210	64	30–31	16	∼4.5/run

Training hyperparameters and compute budget. Every SFT run starts from the pretrained `Qwen3.5-4B` vision-language backbone (24 linear-attention `GatedDeltaNet` layers interleaved with 8 full-attention layers). Each episode is segmented into non overlapping 4.0 s causal multi-turn windows of up to 20 consecutive (frame, CUA action) turns paired with the exact English system prompt used during evaluation. All frames are passed at native 854×480 resolution (`min_pixels=65536`, `max_pixels=414720`). Because scripted trajectories contain no chain-of-thought text, every assistant turn begins with a closed empty thinking prefix (`<think>\n\n</think>`) immediately followed by the CUA action string and end-of-turn token, matching our non-thinking inference template (`enable_thinking=False`). Loss is computed exclusively on the assistant action tokens and end-of-turn tokens (system prompts, images, user headers, and empty thinking tags are masked out) using token-averaged FP32 cross-entropy over each global batch.

We train all models for 1 epoch with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01, gradient norm clipping 1.0), a peak learning rate of 1×10^{-5}, linear warmup over the first $\sim 5\%$ of updates (22 steps for S1, 2–18 steps for variants), and cosine decay to 1×10^{-6}. To keep throughput high without cross-sample attention leakage, multiple 4.0 s windows are packed up to 12,000 tokens per GPU with strict boundary isolated attention masks and per segment position IDs (each window spans $\sim$ 2,500–10,500 tokens). Training runs under PyTorch FSDP2 (`fsdp_size=8`, `bfloat16` activations and compute, FP32 master weights and gradient reductions) on NVIDIA A800-SXM4-80GB GPUs. Total SFT compute is 344.2 GPU hours for GAMEDOJO-4B (S1, 48 GPUs), 74.4 GPU hours for the five transfer variants (S2–S6, 16 GPUs each), and $\sim$ 45.0 GPU hours for the ten additional single-boss curve runs (S7–S16, 16 GPUs each).

Complete set of SFT experiments. Our SFT experiments comprise three groups:

1. *All boss generalist training* (S1, GAMEDOJO-4B): trained on 244.9 h across all 120 bosses and evaluated on all 120 bosses across Easy, Medium, and Hard in both time-paused and real-time CUA modes (Table 4 of the main paper).
2. *Cross game and within game transfer* (S2–S6, compared against S1 and base Qwen3.5-4B): five source configurations spanning 1, 6, and 12 bosses from *Terraria* and *Doom*, evaluated on all 120 bosses grouped into Terraria, Doom, other 2D games, other 3D games, and all 120 bosses (Figure 6 of the main paper).
3. *Own-boss SFT learning curves* (S5, S6, S7–S16 for single-boss 2 h runs, plus S2 and S3 for 6-boss 12 h joint runs): checkpoints saved at 20%, 40%, 60%, 80%, and 100% of 1 epoch to trace own-boss progress against PPO in Figure 4 of the main paper.

B.2 PPO Training Details

Training rollouts and per run summary. We train 14 visual PPO policies from scratch across *Terraria* (2D) and *Doom* (3D): six single-boss specialists (`P1`–`P6` in *Terraria*, `P8`–`P13` in *Doom*) and one six-boss joint policy per game (`P7` in *Terraria*, `P14` in *Doom*). During training rollouts, the policy samples an action from the categorical distribution of each Semantic action head. Each single-boss run uses 100,000 environment steps (33 PPO updates, 12,500 Adam minibatch steps), while each six-boss joint run uses 600,000 environment steps (196 PPO updates, 75,000 Adam minibatch steps), matching the exact exposure of 100,000 steps per boss. Table S4 lists every PPO run, its action space, rollout steps, completed training episodes, wall clock duration, and GPU hours.

Table S4: All 14 PPO Training Runs. Every run trains a shared Nature-CNN policy-value network from scratch on Easy difficulty using the game's Semantic action space (0.2 s decision interval) and signed step-damage reward on a single NVIDIA A800-SXM4-80GB GPU.

ID	Game	Training Boss(es)	Sem. Size	Env Steps	Updates	Episodes	Minutes	A800 GPU-h
`P1`	*Terraria*	King Slime	171	100,000	33	1,085	15.6	0.26
`P2`	*Terraria*	Eye of Cthulhu	171	100,000	33	306	15.6	0.26
`P3`	*Terraria*	Skeletron	171	100,000	33	436	15.6	0.26
`P4`	*Terraria*	Eater of Worlds	171	100,000	33	361	15.6	0.26
`P5`	*Terraria*	Queen Bee	171	100,000	33	483	15.6	0.26
`P6`	*Terraria*	Brain of Cthulhu	171	100,000	33	365	15.6	0.26
`P7`	*Terraria*	All 6 Terraria bosses (joint)	171	600,000	196	3,603	93.6	1.56
`P8`	*Doom*	Cyberdemon	405	100,000	33	1,524	15.8	0.26
`P9`	*Doom*	Barons of Hell	405	100,000	33	433	15.6	0.26
`P10`	*Doom*	Spider Mastermind	405	100,000	33	687	15.7	0.26
`P11`	*Doom*	Archvile	405	100,000	33	749	15.7	0.26
`P12`	*Doom*	Mancubus	405	100,000	33	344	15.7	0.26
`P13`	*Doom*	Revenant	405	100,000	33	277	15.6	0.26
`P14`	*Doom*	All 6 Doom bosses (joint)	405	600,000	196	3,970	93.7	1.56
Total	**2 Games**	**14 PPO Policies**	–	**2,400,000**	**788**	**14,623**	**374.9**	**6.25**

Architecture, hyperparameters, and compute budget. Following DSLE (Gezgin et al., 2026), each policy uses a shared Nature-style CNN encoder: three convolutional layers (`Conv2d(1, 32, 8, stride=4)`, `Conv2d(32, 64, 4, stride=2)`, `Conv2d(64, 64, 3, stride=1)`) followed by a 512-unit fully connected layer with ReLU activations. The 512-dimensional feature vector feeds a scalar value head and one independent linear classification head per Semantic action dimension ($3+3+19$ logits in *Terraria*, 1,691,322 parameters; $9+9+5$ logits in *Doom*, 1,690,296 parameters). Observations are single frame (`frame_stack=1`) 84×84 `uint8` grayscale images converted from RGB via $0.299R + 0.587G + 0.114B$ and adaptive area resizing. Training runs on Easy difficulty ($d = 0.2$ in *Terraria*, $d = 0.15$ in *Doom*) with the signed per step normalized boss-damage reward $r_t = (H_t - H_{t+1})/H_{\max}$ and no extra player-damage, time, or terminal bonus terms.

We use 24 parallel environment slots (6 worker processes $\times$ 4 environments per worker), a rollout horizon of 128 steps (3,072 transitions per update), 4 PPO epochs per update, and minibatch size 32. Optimization uses Adam with initial learning rate 2.5×10^{-4} linearly annealed to 0, $\epsilon = 10^{-5}$, discount $\gamma = 0.99$, GAE $\lambda = 0.95$, clipping ratio 0.1, entropy coefficient 0.01, value loss coefficient 0.5, maximum gradient norm 0.5, no value clipping, and FP32 precision with TF32 disabled. Each run uses one NVIDIA A800-SXM4-80GB GPU, averaging 106.4–106.9 environment steps per second. A single-boss run finishes in 15.6–15.8 minutes (0.26 GPU-h) and a six-boss joint run finishes in 93.6–93.7 minutes (1.56 GPU-h), totaling 6.25 A800 GPU hours for all 14 models.

Complete set of PPO experiments. We conduct three evaluations over the 14 trained policies:

1. *Single-boss learning curves* (`P1`–`P6`, `P8`–`P13`): track episode progress over 100,000 steps on each boss (Figure 3).

2. *Single-boss vs. joint training* (`P7` vs. `P1`–`P6` in *Terraria*; `P14` vs. `P8`–`P13` in *Doom*): compare shared six-boss training against dedicated single-boss training at equal per boss budget (Figure 4 and Section 5.2).
3. *Zero-shot cross-boss transfer*: freeze each of the 12 single-boss checkpoints and evaluate on all 6 bosses in its own game, yielding 30 unseen source-to-target pairs per game (60 pairs total) compared against randomly initialized PPO policies. Figure 5 of the main paper shows the fixed-source comparison.

C Evaluation Details

C.1 Benchmark Evaluation Protocol

Time-paused vs. real-time modes. GAMEDOJO evaluates every environment under two timing protocols that share the exact same 60 Hz physics engine and 200 ms decision step:

- **Time-paused mode (TP)**: The environment renders one 854 × 480 RGB frame and immediately freezes its 60 Hz clock while the agent runs inference. Only after the action arrives does the simulation advance by exactly one 200 ms step (12 ticks), after which it freezes again for the next observation. Inference speed therefore has zero effect on episode outcomes.
- **Real-time mode (RT)**: The simulation clock stays coupled to wall clock inference time. If a model call takes Δt seconds from receiving the observation to returning an action, the environment first advances by $\lfloor \Delta t / 0.2\,\mathrm{s} \rfloor$ neutral no op steps (200 ms each, with all keys released and zero mouse displacement) before applying the returned action to the resulting state. Latency below 200 ms incurs zero lag steps. If the player dies or the boss is defeated during the no op lag steps, the episode ends on that exact tick and the late action is discarded.

Evaluation forms across agent families. All three agent families are scored by the same encounter-level win rate (`WR`, %) and boss-health progress (`PG`, %), averaged first over episodes within each boss and then with equal weight over all evaluated bosses:

- **VLM and LLM agents**: Receive 854 × 480 RGB frames, public control instructions, and a 2 step sliding history of recent frames and actions. Frontier API models are evaluated on all 120 bosses with a 300 step cap (60 s game time) in TP mode and a 180 s wall clock budget in RT mode. Self-hosted `Qwen3.5-4B` and GAMEDOJO-4B checkpoints run 5 independent seeds per boss (600 episodes per difficulty and timing mode) with a 180 s (900 step) budget.
- **PPO specialist policies**: Receive only the current 84 × 84 grayscale frame with no text prompt or frame history, outputting Semantic actions at 0.2 s steps up to the 300 s engine limit. Each policy is evaluated for 20 episodes per boss using test seeds held out from training.
- **Scripted controllers**: State-aware behavior trees running through the same CUA step interface as a privileged upper-bound reference, evaluated over 5 seeds per boss (600 episodes per difficulty) under matched 60 s (300 step, TP comparison) and 180 s (900 step, RT comparison) game-time budgets.

C.2 VLM Evaluation Details

Shared evaluation harness and prompt specification. All frontier, open-source, and generalist game models run through a unified OpenAI compatible evaluation harness (`bossbench_llm`). At episode start, the harness builds a deterministic system prompt by concatenating five fixed sections: (1) role and 200 ms step timing rule, (2) exact output syntax for either CUA (6-substep token string) or Semantic (named JSON dictionary), (3) the game overview and the active controls for the interface, (4) concise boss mechanics (excluding numeric health values, which must be read from the screen), and (5) a short footer with output formatting or observation instructions. At each

decision step, the current 854 × 480 frame is JPEG-encoded (quality 90, base64) and appended to the conversation while retaining a 2 step FIFO window of the two most recent (image, action) pairs. Invalid actions count as format errors and execute a neutral no-op. The harness retries temporary network or HTTP failures up to three times and excludes those failures from model error counts. Figure 7 shows the CUA and Semantic system prompts for the King Slime encounter in *Terraria*.

CUA System Prompt (Example: *Terraria* / King Slime)

You are an experienced PC game player, proficient in keyboard and mouse operations. You are playing "Pixel Arena: Sword and Bow". Based on the current game screen and the corresponding historical trajectory, plan the next 200ms of actions, consisting of 6 steps. Each step is spaced 33ms apart. Every step lasts 33ms from its start time until the next step begins.

Output Format

<|action_start|>X Y Z ; k1 k2 ; k3 ; k4 ; k5 ; k6 ; k7<|action_end|>

Explanation

1. **Mouse Movement**: Relative integers X, Y in [-1000, 1000] (X>0 right, Y>0 down) and scroll Z in [-5, 5] (Z>0 up).
2. **Key Sequence**: 6 groups separated by ;; space-separated keys within each group (at most 4 keys/group). Leave an empty group blank while keeping ;. Keys listed in consecutive groups stay held; unlisted keys are released.
3. Output only one plain line matching the format, with no line breaks or quotes.

Key Naming Rules: Capital letters (W, A), capitalized names (Space, Shift), mouse buttons (LB, RB).

Game: Side-view 2D arena with two wooden platforms. Empty the boss's health bar within 300 s without losing all health.
Controls (only these keys are active): A/D move left/right; Space jump/fly; S drop through platform; J/LB use weapon; Q swap sword/bow; E shoot bow at crosshair; RB swing sword.
Mouse: X/Y move crosshair (1000 = full screen width/height); bow shoots toward crosshair.
Boss: King Slime: Giant gel creature that leaps to crush you, sheds small blobs when hit, and teleports near you. Moves: 1. Leap; 2. Teleport; 3. Gel Spawns.
Health bars and other status are shown on the screen; read them yourself.

Semantic System Prompt (Example: *Terraria* / King Slime)

You are an experienced PC game player. You are playing "Pixel Arena: Sword and Bow". Based on the current game screen and the corresponding historical trajectory, choose the action for the next 200ms.

Output Format

Output one JSON object on a single line that picks one option for each action group, for example:
{"move": "left", "vert": "jump", "attack": "shoot 135"}
- All groups act at the same time for the next 200ms.
- Any omitted group takes its first option (none = do nothing).
- Use option names exactly as listed; do not output indices or any other text.

Action Groups

- move (horizontal): none (stop); left (move left); right (move right).
- vert (vertical): none; jump (jump/fly); drop (drop through platform).
- attack (weapon): none; shoot <angle> (bow arrow at 0–359°, 0°=right, 90°=up, rounded to 22.5°); sword_left; sword_right.

Game: Side-view 2D arena with two wooden platforms. Empty the boss's health bar within 300 s without losing all health.
Mechanics: Flight lasts ∼1.5 s and refills on landing; bow fires every 0.33 s and sword swings every 0.38 s; both destroy boss projectiles; brief invulnerability after taking a hit.
Boss: King Slime: Giant gel creature that leaps to crush you, sheds small blobs when hit, and teleports near you. Moves: 1. Leap; 2. Teleport; 3. Gel Spawns.
Health bars and other status are shown on the screen; read them yourself.

Figure 7: System Prompt Templates for CUA (left) and Semantic (right) Interfaces. Illustrated with *Terraria* (King Slime). The examples show the CUA key sequence and the Semantic action groups. Both prompts state the 300 s arena limit. The benchmark evaluation budgets are given in Section C.1.

Closed source models. Table S5 lists the reasoning effort setting, average output tokens per decision step (including reasoning tokens when produced), and average per step latency for all six closed-source API models evaluated in Table 3 of the main paper. per step token statistics are measured across the completed time-paused CUA benchmark runs (21,622–27,496 decisions per model; 14,421 decisions on Hard for Claude-Haiku-5.5). Where per step API wall clock latency was not logged in the aggregated run summary, we mark the entry with -.

Open-source and generalist game models. Table S6 summarizes reasoning mode, average output tokens per step, average latency per step, and deployment settings for all open-source VLMs and generalist game models. For our self-hosted Qwen3.5-4B and GameDojo-4B evaluations in Table 4 of the main paper, we serve exported bfloat16 checkpoints with vLLM on single NVIDIA A800-SXM4-80GB GPUs (tensor_parallel_size=1, max_model_len=16384, prefix caching enabled, limit_mm_per_prompt={"image":3,"video":0}). Inference uses the aligned non-thinking chat template (enable_thinking=False), prefilling <think>\n\n</think> so the model emits CUA tokens directly (temperature=1.0, top_p=1.0, top_k=-1, max_tokens=256, skip_special_tokens=False), with streaming abort as soon as <|action_end|> is generated. Under 8 concurrent environment

Table S5: Closed-Source Model Evaluation Settings. Average output tokens per step include emitted reasoning tokens measured on time-paused CUA runs. - denotes fields not recorded in the aggregated API logs.

Model	Reasoning Effort	Avg. Out Tok/Step	Avg. Latency
GPT-6.1-Sol	low	42.9	-
GPT-6-Luna	low	227.9	-
Claude-Opus-5.5	low	67.9	-
Claude-Haiku-5.5	low	183.4	-
Claude-Haiku-4.5	low (thinking on)	478.3	-
Gemini-3.8-Flash	low	75.6	-

workers sharing one A800 replica, GAMEDOJO-4B averages 32.8–33.5 output tokens per step, 0.72–0.77 s mean round trip latency in RT mode (0.88–0.92 s in TP mode), and 99.96%–99.98% action format validity across all difficulties.

Table S6: Open Source and Generalist Game Model Evaluation Settings. Average output tokens per step include reasoning tokens where thinking is enabled. `stop tag` indicates streaming termination upon emitting `<|action_end|>`

Model	Reasoning	Out Tok/Step	Latency (s)	Serving & GPU	Decoding Setup
Open Source API Models					
DeepSeek-V4.1-Flash	On	881.0	-	Hosted API	API default
GLM-5.3-Flash	On	1941.1	-	Hosted API	API default
MiniMax-M3	On	1482.7	-	Hosted API	API default
Self-Hosted Open Source Models					
Qwen3.5-35B-A3B	On	1357.1	-	Local vLLM (BF16)	2,048-tok thinking budget
Qwen3.5-4B (Main Table 3)	On	1819.7	-	Local vLLM (BF16)	2,048-tok thinking budget
Qwen3.5-4B (Main Table 4)	Off	46.6	0.90	Local vLLM (BF16)	None-thinking
GAMEDOJO-4B (S1)	Off	32.8	0.74	Local vLLM (BF16)	Non-thinking
Generalist Game Models (CUA Only)					
GameScaling-9B-S3	Off	-	-	Local vLLM (BF16)	Non-thinking
Gamer-35B-A3B	Off	-	-	Local vLLM (BF16)	Non-thinking

C.3 PPO Evaluation Details

Average inference latency. Because the specialist PPO policy is a compact 1.69M-parameter Nature CNN operating on $1 \times 84 \times 84$ grayscale inputs, policy inference time is very small compared with the 200 ms decision step. In our NVIDIA A800-SXM4-80GB throughput benchmark (8 parallel browser actor processes $\times$ 8 environment slots = 64 concurrent environments), batched policy forward passes and action selection take 7.14 s across 43,465 decisions in *Terraria* (0.16 ms per decision, 776.4 end to end steps/s) and 12.87 s across 70,544 decisions in *Doom* (0.18 ms per decision, 575.0 end to end steps/s). Even single frame forward passes execute in well under 1 ms on GPU, which is more than two orders of magnitude below the 200 ms decision interval and therefore introduces zero real-time lag steps.

Action selection. All PPO evaluations use greedy `argmax`. At each decision step, the policy selects the option with the highest logit in each Semantic action head and combines the selected options into one action.