ByteDance | Seed

Lumine: An Open Recipe for Building Generalist Agents in 3D Open Worlds

ByteDance Seed

See Contributions section for a full author list.

Abstract

We introduce Lumine, the first open recipe for developing generalist agents capable of completing hours-long complex missions in real time within challenging 3D open-world environments. Lumine adopts a human-like interaction paradigm that unifies perception, reasoning, and action in an end-to-end manner, powered by a vision-language model. It processes raw pixels at 5 Hz to produce precise 30 Hz keyboard–mouse actions and adaptively invokes reasoning only when necessary. Trained in Genshin Impact, Lumine successfully completes the entire five-hour Mondstadt main storyline on par with human-level efficiency and follows natural language instructions to perform a broad spectrum of tasks in both 3D open-world exploration and 2D GUI manipulation across collection, combat, puzzle-solving, and NPC interaction. In addition to its in-domain performance, Lumine demonstrates strong zero-shot cross-game generalization. Without any fine-tuning, it accomplishes 100-minute missions in Wuthering Waves and the full five-hour first chapter of Honkai: Star Rail. These promising results highlight Lumine's effectiveness across distinct worlds and interaction dynamics, marking a concrete step toward generalist agents in open-ended environments.

Date: November 13, 2025
Correspondence: weihao001@ntu.edu.sg, shiguang.sg@bytedance.com
Project Page: https://www.lumine-ai.org/

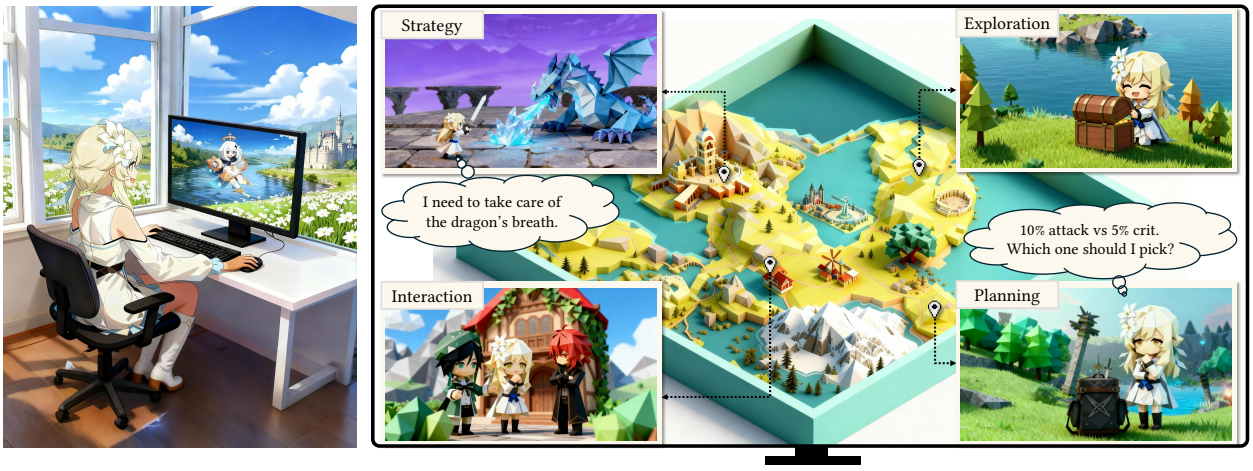


Figure 1 Lumine, the first AI agent to complete hours-long missions in real time within expansive 3D open worlds.

Contents

1 Introduction . . . 3

2 Related Work . . . 4

3 Environment . . . 7

4 The Lumine Model . . . 9

5 Data Curation and Training Recipe . . . 11
- 5.1 Pre-Training . . . 12
- 5.2 Instruction Following . . . 12
- 5.3 Reasoning . . . 13
- 5.4 Training Details . . . 13

6 Inference . . . 14
- 6.1 Context Management . . . 14
- 6.2 Real-Time Optimization . . . 14

7 Benchmark . . . 16

8 Experimental Results . . . 18
- 8.1 Scaling Results . . . 18
- 8.2 Instruction Following Performance . . . 20
  - 8.2.1 Non-History Performance . . . 20
  - 8.2.2 Benefits of History . . . 22
- 8.3 Reasoning Performance . . . 24
  - 8.3.1 Performance in Genshin Impact . . . 24
  - 8.3.2 Generalization to New Games . . . 30

9 Discussion and Future Work . . . 31

A Gameplay Collection . . . 39
- A.1 Participant Recruitment . . . 39
- A.2 Annotation Tasks . . . 39
- A.3 Gameplay Recording . . . 40
- A.4 Post-Process . . . 41

B Instruction Following Data Curation . . . 42

C Reasoning Data Curation . . . 44

D Benchmark Introduction . . . 45
- D.1 Collection . . . 45
- D.2 Combat . . . 46
- D.3 NPC Interaction . . . 47
- D.4 Puzzle . . . 48

E Prompt . . . 49

1 Introduction

Building generalist autonomous agents that can **perceive**, **reason**, and **act** at human-level within **open worlds** has long been the north star for artificial general intelligence research [2, 24, 25, 43, 51, 64, 66]. Over the past decades, remarkable progress has been achieved in constrained domains: agents can match or surpass humans in games such as Atari [50], Go [69], Dota II [10], StarCraft II [73] and Gran Turismo [82]. Despite these achievements, such agents remain confined to closed environments, optimizing a single, well-defined objective with explicitly shaped rewards via large-scale reinforcement learning in a perceive-then-act paradigm. This specialization yields mastery but brittle intelligence with limited abstraction, weak transfer, and poor adaptability to the ambiguity and diversity inherent in open-world scenarios.

A promising path to broader generalization is to ground agents in natural language, leveraging it as a universal medium for knowledge and capabilities across modalities and domains. The recent success of large language foundation models [1, 27, 33, 71] reinforces this view through their superior common-sense understanding and reasoning. These models have begun to extend beyond static text processing into interactive environments such as robotics [3, 20, 35, 93] and video games [29, 70, 75, 78]. While these agents have yet to achieve human-level performance in accuracy and efficiency, these developments signal a promising direction for building versatile, language-grounded agents capable of perception, reasoning and acting in complex worlds.

In this work, we take a step toward this goal by exploring the recipe for building generalist agents in 3D open-world environments. We systematically summarize the six core challenges in achieving this goal:

- **Scalable Environments.** The selection and design of environments that are both rich and diverse, providing compositional dynamics that challenge agents to interact, learn, and generalize, while remaining standardized, scalable, and reproducible [8, 9, 14, 62, 70].
- **Multimodal Perception.** The ability to fuse and interpret heterogeneous sensory streams, including embodied 3D vision, 2D graphical user interfaces (GUIs), textual information and other modalities, to construct an actionable understanding of both the external world and the agent's state [4, 20, 63].
- **High-Level Planning.** The ability to generate self-motivated, long-horizon plans that adapt as environmental dynamics evolve, incorporating reflection and self-revision to refine strategies while balancing competing objectives and environmental feedback [3, 42, 70, 75, 88].
- **Low-Level Control.** The skill to ground abstract intentions into precise and executable actions that enable coherent behavior across diverse embodiments [12, 93].
- **Memory.** The capacity to maintain and leverage experience over various timescales, providing context for consistent decision-making to enable coherent exploration and test-time adaptation [15, 28, 30, 56, 75].
- **Real-Time Inference.** The ability to operate under strict latency constraints, balancing computational deliberation with timely responses and managing asynchronous interactions to avoid missing critical opportunities [13, 40].

We introduce Lumine, a comprehensive and scalable recipe together with its prototype model for addressing these challenges as a concrete step toward general-purpose agents. We select Genshin Impact, a globally popular 3D open-world game, as our primary testbed. To the best of our knowledge, Lumine is the first agent capable of completing hours-long missions in real time within such an extremely challenging environment. Lumine adopts a human-like interaction paradigm that unifies perception, reasoning and action. Built upon the Qwen2-VL-7B-Base model [77], Lumine perceives the game world directly from raw pixels at 5 Hz, and autoregressively generates textual keyboard and mouse actions at 30 Hz using action chunking [91]. The standardized human-like interface and sufficient interaction frequency enable Lumine to operate seamlessly across a wide range of video game environments. Additionally, Lumine adopts a hybrid thinking strategy, allowing it to adaptively enter a thinking mode to produce inner-monologue reasoning before generating executable actions when necessary, thereby avoiding redundant computation and latency without compromising decision quality. We further design a three-stage training curriculum to empower the base model with these capabilities: i) 1731 hours of human gameplay for pre-training to master action primitives; ii) 200 hours of instruction following data to ground control in language; and iii) 15 hours of reasoning data to enable hybrid

thinking. To ensure consistency across long horizon, Lumine dynamically maintains up to 20 recent steps in context as short-term memory and preserves reasoning steps as long-term memory. Finally, an end-to-end optimization yields a 25.3× overall latency reduction, enabling real-time inference and smooth task execution.

During pretraining, we observe a distinct progression of emergent capabilities. Lumine first masters object interaction, then develops basic combat and GUI manipulation, and finally acquires an understanding of game-specific mechanisms and navigation skills, all essential for effective exploration in open-world environments. These emergent behaviors indicate that structured visuomotor competence can naturally arise from large-scale imitation of human gameplay, even without explicit supervision.

Building upon these foundational abilities, instruction-following fine-tuning enables Lumine to demonstrate robust short-horizon control, successfully completing a wide range of tasks lasting from 10 seconds to several minutes, with a success rate exceeding 80% and generalizing effectively to unseen objectives and scenarios.

Following the final reasoning fine-tuning stage, Lumine achieves expert human-level efficiency in completing Act I of Mondstadt's main storyline, a mission lasting about one hour. To evaluate its reasoning generalization, we further assess performance on the remaining main storyline, Acts II and III, which are excluded from the reasoning dataset but included in the pretraining dataset. Lumine continues to demonstrate comparable performance on these missions, which together typically require about four hours for human players. Beyond the training domain, Lumine exhibits strong zero-shot generalization. It manages to navigate to the new region, Liyue, complete the initial one-hour mission, and reach the Adeptus hidden within the mountains, despite no exposure to such content during training. Lumine's capabilities further extend across entirely different games, demonstrating cross-game generalization without any additional fine-tuning. It successfully completes a 100-minute mission in Wuthering Waves and the five-hour first chapter of Honkai: Star Rail, showcasing its ability to transfer visuomotor and reasoning competence to unseen environments.

These results establish Lumine as the first agent capable of real-time long-horizon task completion and cross-environment transfer in open-world settings, demonstrating the effectiveness of the Lumine recipe in developing generalist agents for complex 3D worlds.

2 Related Work

Lumine aims to advance the development of general-purpose agents capable of solving diverse, long-horizon tasks. Although we primarily validate this concept within video game environments, the underlying design naturally shares common principles with GUI agents and robotic vision-language-action (VLA) models. We discuss these connections across the following dimensions and compare Lumine with other representative game agents in Table 1.

Agent. Traditional agents [7, 10, 36, 50, 57, 73, 82] optimize policy networks from scratch via supervised learning or reinforcement learning in a system 1 style [38]. Such agents often generalize poorly to unseen scenarios and struggle to incorporate prior knowledge for temporal adaptation. Moreover, their limited language grounding fundamentally constrains their application in modern interactive environments saturated with textual and symbolic information. The recent rise of large language models (LLMs) and vision–language models (VLMs) [1, 6, 26] has demonstrated strong capabilities in language understanding and commonsense reasoning. Even without additional training, prompt-based agents built upon LLMs and VLMs have shown that such models can serve as powerful foundation models for general-purpose agents, achieving impressive results in complex, long-horizon tasks across diverse domains, ranging from web navigation [76, 80, 89, 92] and robotics [3, 20, 31, 32] to video games [49, 70, 75, 78]. While effective at high-level reasoning and planning, these agents struggle with domain-specific yet essential challenges such as generating precise low-level actions and recognizing fine-grained visual patterns. Moreover, their inefficient inference leads to high latency, making it difficult to meet the real-time requirements of interactive environments. Beyond prompt-based methods, data-driven training approaches have also been explored. Continue training with large-scale robotic data, VLMs can be converted into VLA models [12, 35, 39, 68, 93], which are capable of following instructions and performing a wide range of robotic tasks, demonstrating strong generalization capabilities. A similar paradigm is also applied in GUI agents [5, 17, 54, 60, 81, 87]. More recently, this idea has been extended to game environments [16, 46]. These works typically rely on pretraining with high-quality instruction-following

Table 1 Comparison between Lumine and other representative game agents in terms of environment open-endedness, the longest task duration achievable (measured by the average time required for a human player to complete), multimodal understanding (vision and text), ability to follow instructions to accomplish diverse tasks, reasoning capability, real-time inference, and the interface used for interaction (K&M denotes keyboard and mouse).

Method	Open-World	Task Horizon	Multimodal Understanding	Instruction Following	Reasoning	Real-Time	Interface
DQN [50]	✗	5 mins	✗	✗	✗	✓	APIs
AlphaStar [73]	✗	15 mins	✗	✗	✗	✓	APIs
OpenAI Five [10]	✗	45 mins	✗	✗	✗	✓	APIs
VPT [7]	✓	20 mins	✗	✗	✗	✓	K&M
Voyager [75]	✓	20 mins	✗	✓	Stepwise ☹	✗	APIs
Cradle [70]	✓	1 hr	✓	✓	Stepwise ☹	✗	K&M
SIMA [62]	✓	10 secs	✓	✓	✗	✓	K&M
CombatVLA [16]	✗	1 min	✓	✗	Stepwise ☹	✗	K&M
JAVIS-VLA [46]	✓	10 secs	✓	✓	✗	✗	K&M
Lumine (Ours)	✓	5 hrs	✓	✓	Adaptive ☺	✓	K&M

or reasoning datasets, annotated by human labelers, but this leaves open the risk for continued scaling. In this work, Lumine also applies a similar VLA setting but aims to provide a more efficient and scalable training recipe for general-purpose agents.

Environment. Video games have long served as popular environments for developing AI agents, primarily due to their efficient and low-cost interactions while providing rich and diverse dynamics. Traditional game environments are typically built on games that expose APIs for accessing internal states and actions [8–10, 19, 21, 22, 37, 44, 55, 65, 73]. However, most environments are limited to fixed maps with constrained dynamics and minimal textual content. The limited volume restricts their applicability for developing broader aspects of intelligence. Moreover, each environment adopts its own conventions for encapsulating observation and action spaces, making it difficult to develop general-purpose agents transferable across environments. It further limits scalability to more games, particularly commercial games, which constitute the vast majority of the market but typically do not provide API access. Commercial games, especially AAA games, however, often feature more realistic physics engines, richer content, and more diverse gameplay, making them especially valuable yet challenging testbeds for general-purpose agents. Recent advances [62, 70] demonstrate the feasibility of interacting with arbitrary PC games via human-like interfaces (monitor, mouse and keyboard), substantially extending the reach of AI agents. Building upon this direction, Lumine aims to provide a general solution to develop agents in these challenging video games.

Task. Commercial video games are usually built around carefully designed missions that mirror human learning curricula, progressing from simple to complex challenges and comprehensively evaluating agents across multiple levels of competence, from fundamental skills such as navigation, interaction, and combat to higher-level abilities such as adapting to dynamic environments, leveraging newly acquired knowledge, and composing learned skills to solve novel problems. These missions frequently include long-horizon objectives that may span hours or even days to complete, providing natural and richly structured testbeds for investigating long-term planning and compositional intelligence. Traditional RL agents [10, 36, 50, 73, 82] typically optimize a single objective within closed environments, exhibiting limited multitask capability and poor open-ended exploration ability. In contrast, VLA models [12, 39, 62, 93] can follow instructions to perform diverse tasks, yet remain limited to short horizons of only a few seconds to minutes. Recently, prompt-based reasoning agents have achieved remarkable progress, successfully completing one-hour main storyline missions in Red Dead Redemption 2 [70] and full playthroughs of Pokémon Red [29, 90]. These systems exhibit task composition, reflection, and contextual reasoning over extended periods of time. Building upon these advances, Lumine aims to combine the instruction-following versatility of VLA models with the long-horizon autonomy demonstrated by prompt-based agents, pursuing a unified framework capable of accomplishing diverse, extended missions through reasoning-driven planning and adaptation.

Reasoning. Some robotics VLAs adopt a hierarchical architecture, where one model performs high-level reasoning at a low frequency, and another model generates low-level actions based on that reasoning [11, 23, 67]. Although this structure provides temporal abstraction, it is challenging to optimize both stages jointly and stably due to non-stationarity [34]. In contrast, other approaches [29, 35, 60, 87, 90] follow the ReAct paradigm [88], where the agent performs explicit reasoning and outputs an action at every step. While straightforward, this design can be computationally inefficient and prone to hallucinations in continuous, high-frequency control settings. To combine the strengths of both paradigms, we draw inspiration from hybrid thinking [29, 61], where LLMs can flexibly decide whether to perform explicit reasoning or directly output an action based on the context. Lumine is trained in an end-to-end manner and is capable of generating reasoning only when necessary, seamlessly coordinating reasoning and control.

Memory. Recent VLA models [12, 23, 39, 93] and data-driven game agents [7, 47, 62] typically operate in a purely reactive, single-step manner that consumes only the current observation. While this design simplifies both training and inference, it inherently suffers from partial observability, limiting temporal coherence and hindering performance on long-horizon tasks. In contrast, prompt-based agents [70, 75, 78, 90] exploit extended context windows to retain historical information and periodically summarize trajectories into natural language, enabling stronger long-horizon competence. We posit that this paradigm can benefit data-driven approaches as well. Lumine makes an initial step in this direction: it maintains recent observations as short-term memory and leverages reasoning to summarize the past and plan future goals as long-term memory. This "context as memory" design allows us to study how an extended context window influences temporal coherence and action consistency without introducing specialized memory modules.

Interface. To interact with arbitrary video games and software applications, a human-style interface, receiving pixel inputs from the screen and using mouse and keyboard for control, serves as the most unified and standardized approach [70]. Typical GUI agents [60, 86, 87] adopt this paradigm, but often oversimplify input modeling. For instance, most agents ignore mouse movement traces: they couple movement with clicks using absolute positioning, effectively teleporting the cursor to the target location before issuing a click. While sufficient for typical websites or desktop applications, this abstraction fails in video games, where the trajectory and dynamics of motion are critical. In 3D first- or third-person games, for example, the mouse directly controls the camera, making relative movement indispensable. Similarly, mouse trajectories may themselves carry meaning, such as drawing gestures or simulating physical interactions, that absolute teleportation cannot capture. A comparable limitation also exists in keyboard input. GUI agents usually support only coarse-grained operations like press or type, without modeling finer-grained events such as key down, key up, or hold. Yet in video games, these distinctions are essential: holding a key versus tapping it can trigger entirely different actions, while combinations of key states underlie complex mechanics such as sprinting, crouching, charging, or chaining skill sequences. Without this level of expressivity, agents cannot faithfully reproduce the rich and continuous interaction patterns demanded by gaming environments. While some recent game agents [16, 46, 70] attempt to address the above issues, their actions are still represented in code-like formats, resulting in inefficiencies. Lumine aims to provide an efficient and accurate solution for modeling keyboard and mouse operations while covering the full spectrum of functionalities.

Inference. Real-time inference poses a great challenge for VLM-based agents, especially in fast-paced video games requiring high-frequency interactions. Efficient real-time inference for VLM-based GUI or game agents with keyboard and mouse control remains largely unexplored. It usually takes GUI agents several seconds to produce a single executable action in an autoregressive manner [5, 54, 60, 81, 87]. In robotics, VLAs employ techniques such as action chunking [91], flow matching [12], and action tokenization [58] to accelerate policy learning and action generation. While these methods are promising to be adapted to game agents, the semantic nature of actions must be carefully considered. Unlike robotics, where low-level actions are usually with limited semantic meaning and trained from scratch, mouse movements and key presses carry clear semantic intent that is naturally interpretable by VLMs. Lumine provides a practical solution by combining these techniques with traditional LLM inference optimization strategies and efficient action modeling, enabling autoregressive models to achieve real-time inference in gaming environments.

Figure 2 Overview of the gameplay environment in *Genshin Impact*. The game combines large-scale open-world exploration and multi-level reasoning challenges within a richly interactive 3D environment. Players can freely traverse diverse regions, glide, swim, dive, and interact with characters while engaging in quests, puzzles, and combat.

3 Environment

Agents must learn from interaction with diverse dynamics to achieve general-purpose capabilities. Video games have emerged as popular platforms for developing such agents due to their efficient interaction and rich dynamics. Among these, commercial video games stand out since they typically feature high-quality graphics, offer expansive open worlds and complex systems that require nuanced and timely decision-making. While these games provide ideal environments for developing general-purpose agents, they remain largely unexplored due to their closed-source nature. To bridge this gap, we select Genshin Impact, a globally popular commercial video game, as a representative case, aiming to present a general and scalable solution for developing agents in such challenging, content-rich environments, shown in Figure 2.

As an online, third-person, action role-playing open-world game with hundreds of hours of playthrough content, Genshin Impact has incorporated most of the common challenges found in video games, making it both an ideal and demanding testbed for our vision:

Open-World Exploration. Genshin Impact offers players a vast and highly interactive open world that blends high-fidelity natural beauty, rich cultural landscapes, and immersive gameplay. Powered by high-quality graphics, realistic physics, and richly detailed environments, the world draws inspiration from real-world geography and features diverse landforms, such as towering mountains, tropical forests, expansive deserts, frozen tundras, and vibrant coastlines, combined with dynamic weather and lighting conditions, all crafted with fine-grained environmental modeling. Players are granted extensive freedom of movement, including running, jumping, climbing, gliding, swimming, sailing and diving, requiring sophisticated spatial understanding and 3D navigation. Fully traversing and understanding the environment typically requires weeks or even months of continuous exploration, reflecting the game's exceptional scale and structural complexity. Beyond its natural terrains, Genshin Impact features multiple nations and urban regions, each characterized by distinct cultural aesthetics, architectural styles, and environmental motifs. This diversity enriches the overall world structure and contributes to the game's function as a comprehensive testbed for open-world exploration and embodied intelligence research.

Long-Horizon Progression. In the game, players assume the role of a traveler undertaking an extensive journey

across seven nations in search of their lost sibling. This adventure spans hundreds of hours through main story missions and countless side quests, many of which demand sustained effort and long-term engagement. Character progression and team development evolve gradually over weeks or even months, requiring continuous planning, strategic resource allocation, and deliberate decision-making. These long-horizon tasks stand in contrast to the short-term objectives typically emphasized in traditional research benchmarks [8, 9, 41, 59, 65], positioning Genshin Impact as an ideal environment for examining extended strategic planning and complex decision-making.

Diverse Gameplay. As a role-playing game, Genshin Impact encompasses the full spectrum of core gameplay elements characteristic of the genre: expansive 3D open-world exploration, real-time combat, map navigation, dungeon challenges, interactive NPC dialogue, quests completion, team composition, character progression, equipment collection, resource management, and crafting systems. Beyond these foundation components, the game continuously broadens its scope through a wide range of mini-games and special events, ranging from housing systems, card battles and auto-chess to rhythm games, tower defense, and hide-and-seek, effectively covering most common forms of both 3D and 2D interaction found in modern video games. This extensive gameplay diversity presents an exceptionally rich environment for generalization research, allowing for the study of how agents can acquire, transfer, and adapt skills across heterogeneous tasks within a single, coherent virtual world. Consequently, Genshin Impact serves as a promising platform for exploring the development of general-purpose intelligence that extends beyond domain-specific competence.

Rich Puzzles. A distinct characteristic of Genshin Impact lies in the richness and variety of its puzzles distributed throughout the game's expansive world. These puzzles extend far beyond conventional logic-based challenges, integrating elements of exploration, observation, environmental interaction, and strategic reasoning. Players must interpret terrain and environmental clues, manipulate mechanisms, and master elemental mechanics to activate or combine triggers. Many puzzles emphasize temporal and spatial coordination, requiring players to activate devices or eliminate enemies within limited time windows, glide or climb to reach distant targets. Others encourage close observation and memory, as players must recall previously encountered symbols, device states, NPC hints or environmental patterns that serve as clues for subsequent actions. Larger multi-stage puzzles, which are often embedded within ancient ruins or region-specific questlines, demand reasoning across extended spatial and causal structures, such as understanding dependencies among multiple devices or predicting the effects of sequential activations. This multi-layered, multidimensional design constitutes a comprehensive test of perception, logic, and execution, providing a challenging yet structured environment for studying embodied reasoning, spatial cognition, long-term memory and adaptive problem-solving.

Comprehensive Guidance. As a globally popular video game, Genshin Impact offers a well-structured onboarding experience with detailed tutorials and beginner-friendly guidance. Whenever players encounter a new gameplay mechanic or system for the first time, the game provides clear and accessible instructions directly on screen. These tutorials are also archived within the game for easy reference, allowing players to revisit key information as needed. This built-in system not only supports players but also greatly benefits agents designed to learn from or interact with the game. Furthermore, the difficulty curve is carefully calibrated: challenges progress gradually, and gameplay features are unlocked step by step, making the overall experience highly approachable for beginners starting from scratch and providing a natural curriculum for agents to acquire skills in a staged and progressive manner.

Large Population Base. As a globally popular game, Genshin Impact has cultivated a massive and diverse global player base, facilitating large-scale participant recruitment and offering abundant opportunities for gameplay data collection and content annotation. The game further benefits from a vibrant online community that actively produces a wide array of guides, walkthroughs, tutorials and analytical discussions. These resources not only support human players but also constitute valuable auxiliary data sources for agent learning.

While the in-game content already presents diverse content and sufficient challenges, developing autonomous agents within such commercial video games introduces additional systematic difficulties [62]:

- Commercial video games usually do not expose APIs for interaction. Agents must rely on the standard interface, i.e., receiving raw pixel images from the screen and issuing keyboard and mouse inputs for

control.

- Unlike traditional research environments [8, 9, 22, 41, 59, 65], where environments are frozen during action generation, agents must act under strict latency constraints and execute actions asynchronously.
- Internal states and reward signals are usually inaccessible. Moreover, reliance on GPU rendering limits scalability to thousands of parallel rollouts. These challenges make reinforcement learning particularly difficult and highlight the need for more efficient approaches.

4 The Lumine Model

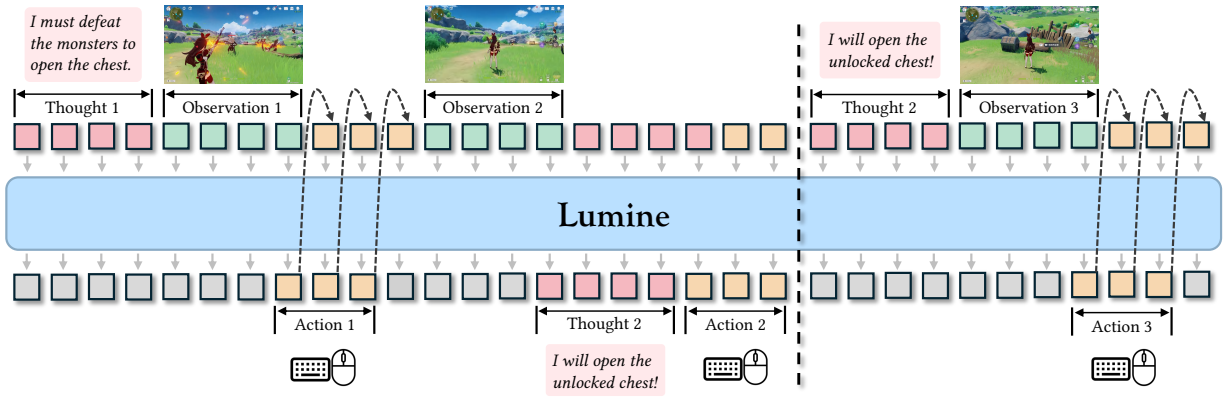


Figure 3 Overview of the Lumine model. Built upon a VLM, Lumine receives pixel inputs along with historical context, such as previous actions and reasoning, and outputs textual keyboard and mouse actions. It employs a hybrid reasoning strategy, generating new reasoning steps only when necessary; otherwise, it directly produces actions for efficient real-time control.

As illustrated in Figure 3, we introduce Lumine, a 7B-parameter model designed to process sequences of raw pixel images as inputs and generate executable keyboard and mouse operations, accompanied by interpretable intermediate reasoning. Lumine builds upon Qwen2-VL-7B-Base [77], inheriting its strong multimodal understanding and generation capabilities gained from large-scale pretraining on diverse web data. By augmenting this foundation with an explicit reasoning and action prediction mechanism, Lumine is capable of closed-loop visual decision-making within interactive environments.

At each timestep t, the model first determines whether to enter a thinking mode to generate reasoning r_t, conditioned on its historical visual observations $o_{\leq t}$ as well as prior reasoning traces $r_{<t}$ and actions $a_{<t}$. If the model decides not to reason explicitly, r_t is set to null. Subsequently, it predicts the next executable action a_t. Formally, the model π_θ captures the joint distribution over reasoning and actions as:

$$\pi_\theta(a_t, r_t \mid o_{\leq t}, r_{<t}, a_{<t}) = \pi_\theta(a_t \mid o_{\leq t}, r_{\leq t}, a_{<t})\, \pi_\theta(r_t \mid o_{\leq t}, r_{<t}, a_{<t}), \tag{1}$$

This factorization reflects the model's perceive-reason-action paradigm, where intermediate reasoning provides an explicit latent structure that guides the subsequent action generation. Next, we provide a concrete definition of Lumine's observation and action space.

Observation Space. Lumine perceives continuous visual input from the game environments. Each frame is resized to 1280×720 (720p), a resolution chosen to balance UI text legibility and computational efficiency. To align with human visual reaction time of roughly 200–250 ms [79] and to avoid missing critical timing events, Lumine processes one observation frame every 200 ms (5 Hz). In addition to visual inputs, Lumine maintains a history of past reasoning and actions as contextual information, preserved in the form of multi-turn dialogues interleaved across model outputs with the visual inputs.

Hybrid Thinking. As shown in Table 2, at every step, Lumine first optionally produces intermediate reasoning as inner monologue and then generates the executable keyboard and mouse actions. Lumine adopts a hybrid

Table 2 Examples of Lumine switching between thinking and non-thinking modes by generating <|thought_start|> or <|action_start|> as the first token at each step. Historical information is not shown for simplicity.

Thinking Mode	Non-Thinking Mode
<|im_start|>user {image} <|im_end|> <|im_start|>assistant <|thought_start|> {reasoning_content} <|thought_end|> <|action_start|> {action_content} <|action_end|><|im_end|>	<|im_start|>user {image} <|im_end|> <|im_start|>assistant <|action_start|> {action_content} <|action_end|><|im_end|>

thinking mode, engaging in explicit reasoning only when necessary, while continuously predicting actions that align with its ongoing thought process. Each reasoning phrase is enclosed by the special tokens <|thought_start|> and <|thought_end|>, explicitly marking the boundaries between the model's inner monologue and actions. The reasoning acts both as a reflection on previous behavior and as a plan for subsequent steps. Reasoning typically emerges at critical transitions, such as when sudden environmental changes cause prior plans invalid and adjustments are required, or when a task has been completed and new goals need to be proposed.

Keyboard and Mouse Modelling. While prior works often introduce additional action heads [12, 35] or redefine the vocabulary [46, 93] to represent actions, they fail to exploit the inherent semantics of keyboard and mouse operations, which are well captured by LLMs. Other approaches [16, 46, 60, 70] model such interactions in code formats, which tend to be verbose and inefficient for high-frequency interactions. Motivated by these limitations, Lumine introduces a concise and efficient action representation that allows models to autoregressively generate both high-level reasoning and low-level control signals without modifying the model architecture or vocabulary, enabling seamless integration across different LLM-based agents. We unify all keyboard and mouse actions within the language space, covering the full spectrum of functionalities, which we formally define as follows:

$$\underbrace{\Delta X\ \Delta Y\ \Delta Z}_{\text{Mouse movements}}\ ;\ \underbrace{K_1\ ;\ K_2\ ;\ K_3\ ;\ K_4\ ;\ K_5\ ;\ K_6}_{\text{Key presses}}$$

- **Format.** Each action is enclosed by special tokens <|action_start|> and <|action_end|>. Internally, the action consists of a semicolon-separated sequence of components that specify mouse movements ($\Delta X\ \Delta Y\ \Delta Z$), and a series of key presses (K_1 ; K_2 ; K_3 ; K_4 ; K_5 ; K_6).
- **Mouse Movement.** We discretize the mouse action space. Lumine predicts the *relative displacement* $(\Delta X, \Delta Y)$ as integer values within the range $(-1000, 1000)$, along with an associated scroll value ΔZ as an integer in $[-5, 5]$ representing the number of scroll steps. The full predicted movement is then executed smoothly over a 200 ms interval to ensure execution efficiency.
- **Key Presses.** To capture fine-grained dynamics while maintaining computational efficiency, Lumine adopts an action chunking strategy [91]. At each step, the model predicts six consecutive action chunks over a 200 ms window, with each chunk lasting 33 ms, resulting in a 30 Hz interaction frequency. An action chunk K_t specifies zero to four keys, including both keyboard inputs and mouse buttons, that are pressed during this interval. Any keys not listed in the chunk are automatically released. Keys that appear in consecutive chunks retain their key-down state and are not pressed again. The choice of a 33 ms granularity is supported by empirical evidence: we observe that the minimal keyup–keydown interval is around 40 ms, which is consistent with the reported lower bound of approximately 60 ms for inter-key intervals (keydown+keyup) [18]. This action modeling allows Lumine to faithfully mimic player behavior and accurately execute a wide range of complex interactions, such as long and short key presses, key combinations, rapid clicking, drag-and-drop gestures, or simultaneous multi-key actions

typical in fast-paced scenarios such as combat. For efficient inference, each key is represented by a single token, shown in Appendix Table 6.

A possible instantiation of such an action could be: "92 0 0 ; Shift W ; Shift W ; Shift W ; F W ; F W ; F". This action depicts the agent dashing (`Shift` & `W`) toward a treasure chest on the right (mouse turn right `92` units), then stopping upon reaching it and attempting to open the chest (`F`).

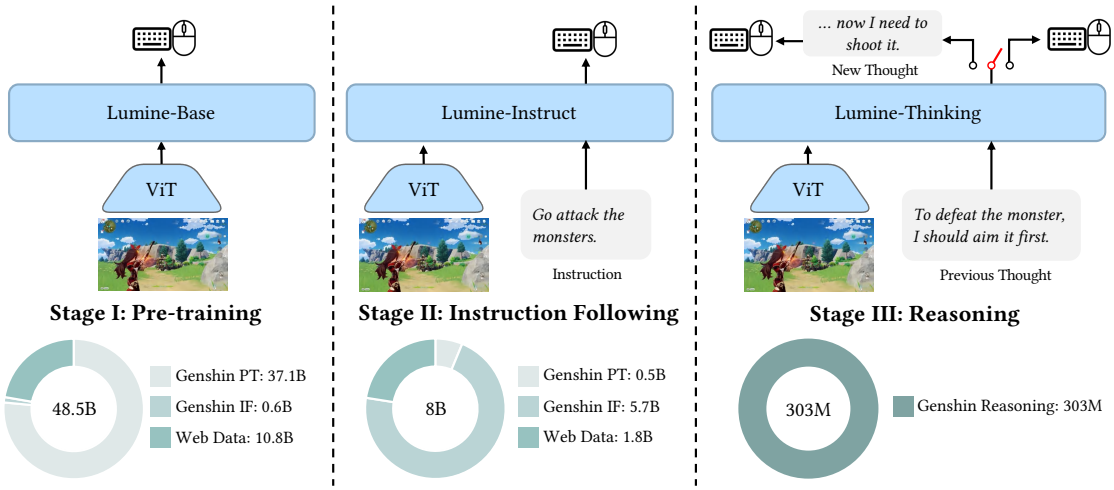


Figure 4 Overview of Lumine's three-stage training recipe. In the first pre-training stage, Qwen2-VL-Base is trained on large-scale image–action data to learn fundamental action primitives, resulting in the Lumine-Base model. In the second instruction-following stage, Lumine-Base is further trained on instruction–image–action triplets for language grounding, producing the Lumine-Instruct model. In the final reasoning stage, the instruction input is replaced with a thought, and an optional new thought is prepended before the action sequence, yielding the Lumine-Thinking model.

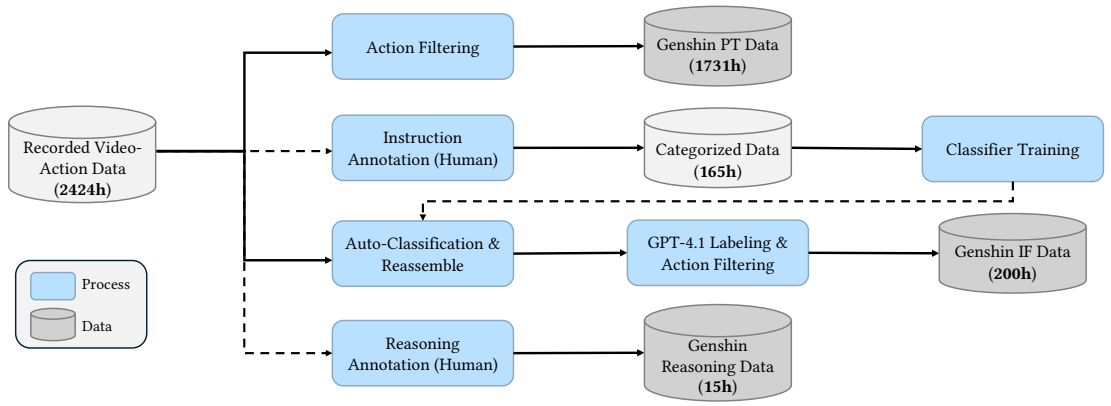


Figure 5 Overview of the data processing pipeline from raw gameplay recordings to curated datasets for pre-training, instruction following, and reasoning. i) Starting from 2424 hours of synchronized video-action data, we first apply rule-based filtering to produce a 1731-hour dataset for pre-training. ii) A subset of 165 hours is human-annotated for instruction-level activities, used to train a classifier that auto-labels all the raw data, further refined into 200 hours of high-quality instruction following data via GPT-4.1 captioning and action filtering. iii) Meanwhile, 15 hours of manually annotated reasoning data support the training of Lumine's hybrid thinking. Together, this multi-stage curation pipeline enables scalable, structured curriculum learning from human demonstrations.

5 Data Curation and Training Recipe

In this section, we present an efficient and scalable data curation pipeline and multi-stage training recipe, designed to enable Lumine to perceive, reason, and act as a generalist agent in the challenging 3D open-world

environment of Genshin Impact.

Training Stages. As illustrated in Figure 4, we employ a multi-stage training procedure for our model. In the first pre-training stage, the model learns to act from raw observations across diverse scenarios, enabling robust and reactive control behaviors. In the second instruction-following stage, the agent's actions are grounded in natural language, allowing it to follow textual instructions to complete short-horizon tasks. Finally, in the third reasoning stage, the model learns to perform explicit reasoning that guides subsequent action generation, supporting complex, long-horizon decision making.

Raw Data. Figure 5 demonstrates that we start from a collected dataset comprising 2424 hours of human gameplay, consisting of synchronized video streams paired with recorded keyboard and mouse operations. Contractors were tasked to start with a brand-new account and progress through the entirety of Mondstadt (the first nation in Genshin Impact), completing its main storyline and achieving over 80% map exploration using only system-provided characters. On average, this process requires approximately 30 hours of playtime for a human player. Comprehensive details on data collection and curation are provided in Appendix A.

5.1 Pre-Training

Unlike prior works [12, 39, 46, 62] that rely on large-scale instruction-following data for pre-training, our approach focuses on exposing models to diverse in-game dynamics by primarily training on image–action pairs without additional labels. This design choice is motivated by two key considerations. i) Human annotation is both costly and difficult to scale, while automatic labeling using VLMs remains unreliable due to their limited long-horizon understanding and insufficient domain knowledge. In contrast, raw video–action data are naturally abundant and straightforward to collect. ii) In open-world exploration settings, it is inherently difficult to assign precise labels to every gameplay segment, as players may wander aimlessly or perform suboptimal and sometimes even confusing actions. These seemingly irregular behaviors, however, capture valuable corner cases that significantly enhance the model's robustness and generalization.

Based on these considerations, we apply rule-based filtering to remove 95% of idle actions and clips dominated by camera jitter, resulting in 1731 hours of high-quality gameplay data. The remaining data are then fully utilized for Lumine's pre-training. To preserve the broad perceptual and reasoning capabilities of the base model during this action-centric training, which are essential for downstream instruction following and reasoning generation, we incorporate an approximately 20% mixture of multimodal web data to retain general knowledge. Additionally, we include a small amount of instruction-following data, which is reserved for evaluation purposes. This stage yields the Lumine-Base model, which serves as the foundation for subsequent language grounded.

5.2 Instruction Following

Building on the action primitives learned during pre-training, we align action prediction with language through a modest amount of instruction-following data. To achieve this, we adopt a label-then-augment strategy that transfers the generalization capability of large vision-language models into the action domain.

We first collect 165 hours of human-annotated instruction data from raw gameplay. Annotators were asked to identify the start and end timestamps of 38 predefined activity categories within 20-second video clips. These annotations are used to fine-tune a classifier based on Qwen2-VL-2B, which enables scalable auto-labeling of the all the raw data.

Empirically, we observe that the Lumine-Base model can instinctively interact with nearby objects, NPCs, and enemies, yet struggles with situations that require breaking its current behavioral inertia, for example, navigating to specific target locations. This is precisely where natural language instructions play a critical role, guiding the model to overcome local behavioral priors and execute goal-directed behaviors. To address this, we identify transition points between adjacent gameplay segments that are assigned with different labels by the classifier, typically indicating a shift in task context or objective. Around each transition point, we extract a 20-frame (4s) snippet and prompt GPT-4.1 [52] to generate diverse, context-aware instructions based on the labeled categories. While the provided category labels supplement GPT-4.1's limited understanding of

game mechanics and objectives, the model also acts as a verifier, detecting and discarding mislabeled samples when inconsistencies are found.

After applying the same action filtering as in pre-training, we obtain 200 hours of high-quality instruction-following data. These are mixed with multimodal web data in the same ratio as used during pre-training, while retaining a minimal portion of action-only data to preserve behavioral diversity. The resulting model, denoted as Lumine-Instruct, can generate contextually grounded actions in response to textual instructions and serves as the foundation for the subsequent reasoning-stage training.

5.3 Reasoning

Building upon Lumine-Instruct's ability to follow textual guidance, we further enhance the model with explicit reasoning skills for autonomous exploration and long-horizon decision-making. To achieve this, we curate a specialized dataset of human-annotated inner monologues. We select the first act of Genshin Impact's main storyline in Mondstadt, *Prologue: Act I - The Outlander Who Caught the Wind*, as a testbed and sample 27 gameplay videos from the raw data in which human players are engaged in this mission. Annotators are provided with consecutive 10-second clips and instructed to identify key decision points at the frame level, then write first-person thoughts that articulate the underlying rationale behind actions in a concise and accurate tune. This process yields a high-quality dataset of 15 hours gameplay, containing 15K reasoning traces, with an average interval of 3.2 seconds between consecutive thoughts. Each reasoning sequence contains an average of 37.4 ± 11.7 tokens, achieving a balance between detail and brevity.

To better align with real inference scenarios, we do not apply any action filtering, enabling Lumine to learn to wait appropriately at critical decision points. We then fine-tune Lumine-Instruct on this dataset, resulting in Lumine-Thinking, an autonomous model to complete hours-long missions without human intervention.

Table 3 Hyperparameters and computational resources used by Lumine during the three-stage training process under both non-history and history settings. We apply VeOmni [48] as our training framework, which dynamically packs batches to match the target sequence length for efficient training.

	Non-history			History		
Hyperparameters	**Pre-training**	**Instruction Following**	**Reasoning**	**Pre-training**	**Instruction Following**	**Reasoning**
LLM Learning Rate	2e-5	2e-5	1.83e-5	2e-5	2e-5	1.64e-5
ViT Learning Rate	7e-6	7e-6	-	7e-6	7e-6	-
LR Scheduler	Constant	Cosine	Cosine	Constant	Cosine	Cosine
Gradient Norm Clip	1.0	1.0	1.0	1.0	1.0	1.0
Optimizer	AdamW($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1.0 \times 10^{-8}$)			AdamW($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1.0 \times 10^{-8}$)		
Warm-up Ratio	0.05	0.05	0.05	0.05	0.05	0.05
Batch Packing Length	32768	32768	32768	32768	32768	32768
Batch Size	128	128	64	128	128	64
Training Epochs	1	2	3	3	3	3
GPU Num (H100)	64	32	64	64	32	64
Training Time	3.5 Days	1.3 Days	1 Hour	12.4 Days	2.2 Days	1 Hour
GPU Hours (H100)	5376	960	64	19008	1664	64

5.4 Training Details

We explore both non-history (single-frame input) and history (multi-turn input) settings. The data organization varies slightly across the three training stages.

- **Pre-training.** In the non-history setting, each sample consists of a single image paired with its corresponding action. In the history setting, each sample consists of 20 interleaved image-action pairs, resembling the multi-turn conversational structure used in VLMs.
- **Instruction Following.** In the non-history setting, each trajectory is decomposed into single-step samples, where each step includes an instruction, an image, and the corresponding action. In the history setting, training is performed on the entire trajectory sequence.

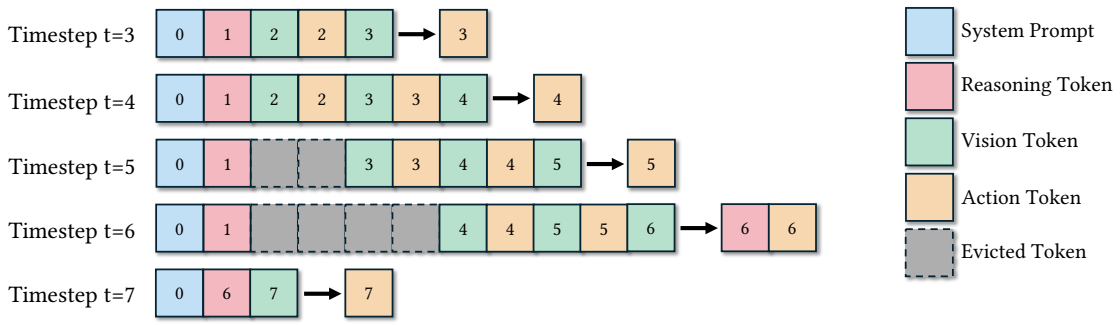


Figure 6 Visualization of the strategy Lumine uses for context management during inference. Lumine maintains a sliding window within the context to preserve image–action pairs across interaction steps, with a maximal window length of 2, as shown in the example. The context begins with the system prompt and previous reasoning, which guide subsequent action generation. When the number of image–action pairs exceeds the threshold, the oldest pair is discarded while retaining the system prompt and reasoning. Upon generating new reasoning, the context is flushed and re-accumulated from that point onward.

- **Reasoning.** In the non-history setting, most samples follow the same format as instruction-following data, where the model receives the previous reasoning and current visual observation as input. Only a small subset of data contains segments that generate new reasoning. For history training, a trajectory starts from the first frame after generating a new reasoning and ends upon the next reasoning generation, or earlier if it exceeds 20 frames in length.

We adopt VeOmni [48] as our training framework, and Table 3 presents the hyperparameters and computational resources used across the three-stage training process under both the non-history and history settings. The ViT backbone is kept frozen during the final reasoning stage. Empirically, we observe that during the pre-training stage, multiple training epochs continue to yield improvements in the history setting, whereas the non-history setting tends to overfit after the second epoch. We report the results corresponding to the number of training epochs that achieved the best performance on our benchmark.

6 Inference

In this section, we first introduce our inference strategy for context management, followed by the optimizations applied to achieve real-time inference.

6.1 Context Management

Figure 6 illustrates how Lumine manages context during inference using a sliding-window mechanism. In the history setting, the sliding window length is set to 20, whereas in the non-history setting, it is set to 1. At the beginning of the context are the system prompt and previous reasoning, which together guide subsequent action generation. These are followed by image–action pairs maintained in a multi-turn dialogue format. A first-in-first-out (FIFO) policy is applied: when the number of pairs exceeds the maximum threshold, the oldest image–action pair is discarded while retaining the system prompt and reasoning. When new reasoning is generated, the context is flushed and re-accumulated from that point onward. By applying this strategy, Lumine effectively uses previous reasoning steps as long-term memory and a 20-frame context as short-term memory, thereby maintaining the consistency and coherence of the model's behavior during continuous interactions. In this work, we preserve only the most recent reasoning within the context, while the mechanism can be easily extended to maintain multiple reasoning segments if needed. We set the temperature to 1 and `top_p` to 1 for all inference settings.

6.2 Real-Time Optimization

Real-time operation is essential for continuous, closed-loop control within dynamic environments that require a high frequency of interaction. Achieving real-time interaction between Lumine and the game presents

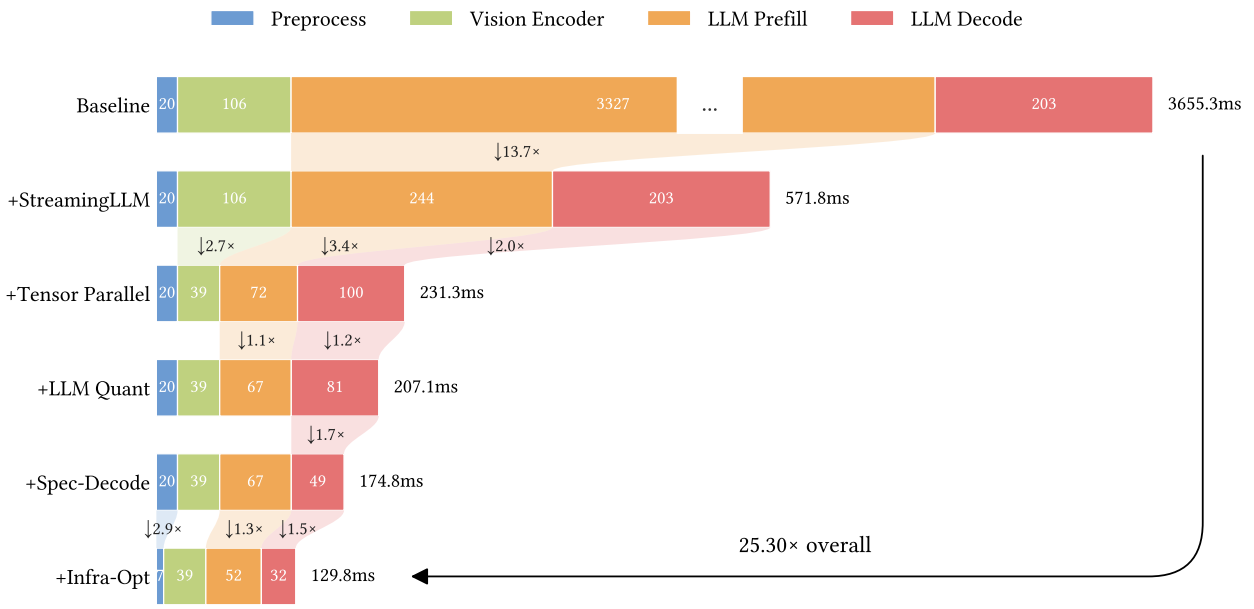


Figure 7 Latency breakdown by stage with corresponding ribbons and overall improvement. The figure shows time latency under different strategies for a typical 20-token action without reasoning generation and with the full context of 20 frames. Infra-Opt denotes the remaining infrastructure-level optimizations. The overall optimization yielding a 25.3× speedup compared with the baseline.

significant challenges for a 7B-parameter model. A typical interaction involves two machines: a local Windows host running the game client and a remote server responsible for model inference. Communication between the two machines is conducted over the network.

At an interaction rate of 5 Hz, the system must complete the following within 200 ms in each control cycle: i) the host captures a screenshot of the game and sends an inference request to the model server; ii) upon receiving the request, the server performs inference and returns an action string; and iii) the host parses the action string into corresponding keyboard and mouse events and executes them. To achieve this level of responsiveness, we developed a series of optimizations across communication and model inference (prefill & decoding). Figure 7 provides a detailed latency breakdown with different optimization strategies.

Communication. We employ a streaming output mechanism for timely action execution. Lumine autoregressively generates an entire action sequence composed of six continuous chunks, each corresponding to a 33 ms executable segment of keyboard and mouse operations. Once a complete chunk is produced, signaled by a terminating semicolon, it can be executed immediately without waiting for the full sequence to finish. This substantially relaxes the strict timing constraint: as long as each action chunk is generated within 33 ms, the remaining 200 ms interval can be fully utilized for processing stages. Additionally, to reduce data payload, image observations are compressed into JPEG format, and then Base64-encoded before being transmitted. A persistent TCP connection is maintained to avoid repeated handshaking costs.

Prefill. In the history setting, we maintain and reuse the historical key–value (KV) cache from prior interactions for efficiency. When the context window becomes saturated and the earliest turns need to be dropped, we adopt the StreamingLLM technique [85] to maximize KV-cache reuse. An attention sink is anchored to the system prompt to prevent collapse, enabling a stable attention window for efficient inference. Note that we observed a performance degradation in long-horizon tasks with the use of StreamingLLM, which is effectively alleviated by our context management strategy that clears the context when a new reasoning appears.

Decoding. As discussed in Section 4, we minimize the number of decoding steps by designing a compact action space. Additionally, we implement a draft-model-less speculative decoding strategy [45] to further reduce token decoding, leveraging fixed delimiters across generation stages. Specifically, we observe that ΔX

and ΔY end with a space, ΔZ and $K_1, \cdots K_5$ end with a semicolon, and K_6 ends with the `<|action_end|>` token. We use a simple state variable to track the generator's stage to dynamically select the appropriate delimiter as a draft token. Standard reject sampling is then applied to ensure the final output matches the same distribution.

Infrastructure. We further optimize the low-level computational framework to maximize hardware utilization and throughput: i) *Tensor Parallelism:* Model weights are partitioned across four GPUs. Since Qwen2-VL-7B exposes only four KV heads, we deploy the server on four NVIDIA H20 GPUs with a tensor-parallel (TP) degree of 4, assigning one KV head per GPU. This contributes the a remarkable acceleration. ii) *Quantization:* We apply W8A8 (8-bit weights and activations) quantization using SmoothQuant [84] to reduce computation and memory bandwidth demands during both the ViT and LLM prefill stages. iii) *Kernel and Graph Optimizations:* We perform search-based tuning of GEMM kernels for the ViT, prefill, and decode stages, and introduce a custom one-shot all-reduce kernel to enhance communication efficiency during decoding. With speculative decoding enabled, we further cut CPU latency by capturing a single CUDA graph that fuses the forward pass and rejection sampling process. Finally, image preprocessing is offloaded to the GPU to accelerate data handling and minimize CPU–GPU transfer overhead.

The combined effect of these optimizations significantly reduces end-to-end latency across preprocessing, vision encoding, prefill, and decoding stages, as illustrated in Table 4. When reasoning generation is not invoked, the delay before producing the first action chunk is approximately 110 ms, well below the 200 ms threshold. Even in the worst case, when generating the longest action chunk (four keys and a semicolon), the latency remains only 12 ms, comfortably under the 33 ms threshold. This enables Lumine to interact with its environment seamlessly. However, the system is still affected by an asynchrony issue, as it perceives visual inputs that are roughly 200 ms old, which can introduce side effects in highly time-sensitive scenarios. Additionally, during steps where the model outputs reasoning, the latency may exceed 200 ms, resulting in a brief idle period. Nonetheless, due to the low frequency of reasoning events, we empirically observe no noticeable impact on the visual experience, maintaining smooth and stable gameplay. This further highlights the importance of hybrid thinking for balancing efficiency and responsiveness.

Table 4 Inference time of Lumine at each stage with the combined optimizations. In addition to latency, we also report the statistical averages of token usage and forward steps. Due to speculative decoding, the number of forward steps is typically smaller than the number of generated tokens.

Stage	Time (ms)	Token	Forward Step
Network latency	6	-	-
Preprocessing	6.8	-	-
Vision encoder	39	1196	1
LLM prefill	52	1209	1
Decode latency per token	3.1	1	-
First action chunk w/o reasoning	113.9	8.4	4.7
First action chunk w/ reasoning	234.0	46.8	43.1
Action chunk (average)	3.1	1.8	1.02
Action chunk (max)	12.4	5	4

7 Benchmark

To systematically evaluate Lumine's performance in open-world environments, we constructed a comprehensive benchmark of 141 language-conditioned tasks. These tasks are grouped into four categories as shown in Figure 8.

- **Collection.** Agents collect items scattered across varied terrains, such as fruits on trees, plants by rivers, flowers on cliffs, treasure chests hidden in grass, and Oculi in the air. Success requires precise object recognition, strong 3D spatial reasoning, and robust navigation despite environmental distractions.
- **Combat.** Agents defeat varied enemy groups to unlock guarded treasure chests. They must adapt to enemy traits and terrain by strategically coordinating their team's skills and synergies. For example,

Figure 8 Overview of the benchmark comprising 141 tasks across four categories, Collection, Combat, NPC Interaction, and Puzzle. Each category includes simple, hard, and unseen tasks to comprehensively assess agents' various abilities in open-world gameplay.

using ranged attacks against elevated foes or exploiting elemental reactions like freezing water-based monsters with ice abilities. After combat, agents must also retrieve the unlocked treasure chest, which may be easy to overlook during the battle.

- **NPC Interaction.** Agents locate designated NPCs within crowds and engage in dialogues to complete quests or access services such as shops and crafting materials. This task combines OCR capacity for target detection and precise GUI manipulation.
- **Puzzle.** Agents solve diverse challenges, such as activating elemental mechanisms, stepping on pressure plates, lighting torches in sequence, completing time-limited trials, and uncovering hidden pathways. Unlike collection or combat tasks, puzzles require careful observation, logical reasoning, and strategic use of elemental interactions, while requiring precise spatial awareness and fine-grained control.

We categorize tasks into three difficulty levels: simple, hard, and unseen. **Simple tasks** are short-horizon challenges, typically completed within 10 seconds (except for combat, which may last up to 2 minutes). These tasks evaluate fundamental skills, involving continuously visible and ground-level objects that require basic navigation and interaction. **Hard tasks** demand more advanced capacities, such as a nuanced understanding of gameplay mechanics, sophisticated 3D spatial reasoning, and precise low-level control. They often involve distractors, temporally hidden targets, or objectives requiring vertical or aerial navigation and combat against elite enemies. Both simple and hard tasks are situated in Mondstadt, the first nation in Genshin Impact, allowing us to evaluate in-distribution performance. **Unseen tasks** are designed to measure out-of-distribution capacities. These tasks are set in new environments, like Liyue and Inazuma, and feature novel objects, items, NPCs, or enemies not encountered during training. We exclude puzzle tasks from this category, as they often depend on region-specific mechanics that the agent has not been exposed to.

Baselines. We benchmark Lumine against state-of-the-art VLMs: GPT-5 [53], Gemini 2.5 Pro [26], Grok4 [83], Doubao1.6-Vision [74] and Qwen3-VL-235B-A22B-Thinking [72]. All thinking models are with default thinking budgets. To adapt these general-purpose models for gameplay, we integrate them into the Cradle framework [70]. Interaction is enabled through a predefined set of skills in code format (e.g., *turn(degree)*,

move_forward(duration) and *attack()*), that models can invoke via function calls. Models maintain at most five recent steps as historical information in the context as input, which we found empirically to yield the best performance.

Evaluation Setting. In simple and hard tasks, all agents operate under identical early-game conditions, limited to the four starter characters, Traveler, Amber, Kaeya, and Lisa, at level 20 with default weapons in World Level 1. For unseen tasks in Liyue and Inazuma, the same four characters are used at level 40 and with appropriate equipment that aligns with the expected game progression. At the start of each task, the agent is placed at a predefined location and given textual instructions specifying the objective. We use human evaluation to assess task performance. Unless stated otherwise, each task is run five times. Since API responses can take as long as 30 seconds to return, we pause the game during inference following the setup in Cradle [70].

8 Experimental Results

We design our experiments to systematically evaluate the effectiveness of our modeling approach and the contribution of each training stage, focusing on four key questions:

- **Q1:** What does Lumine-Base learn during large-scale pre-training, and how do its core abilities emerge in this phase?
- **Q2:** How well does Lumine-Instruct follow natural language commands? Can it reliably complete both in-domain and unseen tasks given textual instructions, and does incorporating history introduce any benefits?
- **Q3:** After the full three-stage training, can Lumine-Thinking perform complex, long-horizon tasks and generalize to out-of-domain or even entirely new games?

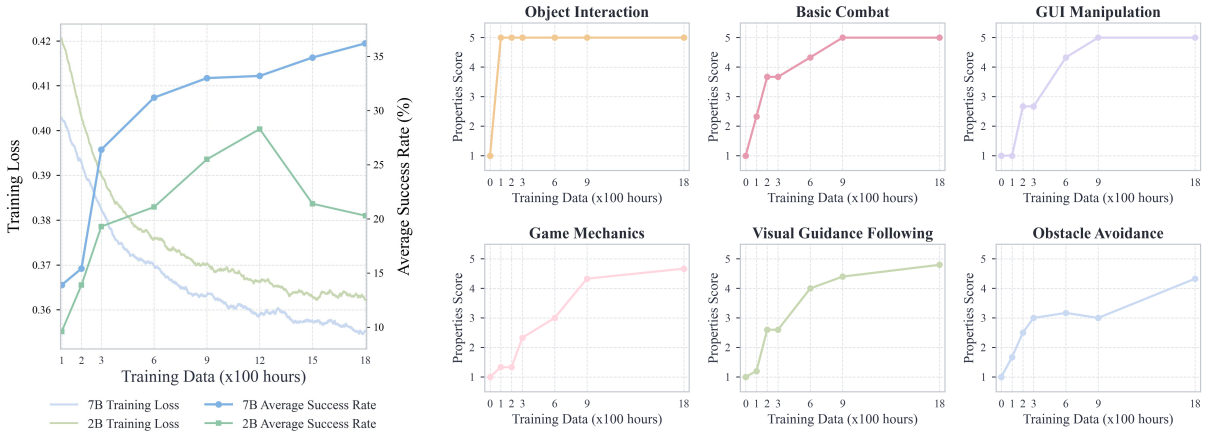
We first evaluate Lumine under the non-history setting and then extend the evaluation to the history setting. By default, Lumine-Base, Lumine-Instruct, and Lumine-Thinking refer to models trained under the history setting, whereas their non-history counterparts are denoted as Lumine-Base-NonHis, Lumine-Instruct-NonHis, and Lumine-Thinking-NonHis.

8.1 Scaling Results

In this section, we investigate the scaling behavior of Lumine without history during the first epoch of pre-training, focusing on two model sizes: 2B and 7B. As shown in Figure 9a, before the 1200 hours of data training, both models exhibit steadily decreasing training loss alongside consistent improvements in benchmark performance. Notably, the 7B model consistently achieves lower loss than the 2B model. However, beyond 1200 hours, a divergence emerges: while the 2B model's loss continues to decline, its benchmark performance begins to degrade, revealing the limited volume of smaller models. In contrast, the 7B model maintains stable improvements across both loss and benchmark metrics. Based on these observations, we adopt the 7B model as our primary base. These findings highlight the effectiveness of our pre-training strategy and provide strong evidence that further scaling will yield additional performance gains.

Atomic Ability Evaluation. To better understand the sources of the improvements of the 7B model during the pre-training, we conduct a study on the model's behaviors from an atomic perspective. The primary objective is to verify that the agent has learned context-appropriate reactions to various in-game scenarios. To this end, we designed a test suite of controlled scenarios to rigorously measure a set of core capabilities essential for effective gameplay. These atomic capacities include:

- **Object Interaction.** Assesses the agent's capacity to interact with nearby objects within reach, such as picking flowers, opening treasure chests, or engaging with NPCs.
- **Basic Combat.** Measures the agent's proficiency in combat scenarios, including executing basic attacks, switching characters to perform combos and identifying as well as engaging elevated enemies with ranged characters.



(a) Training dynamics of base models.

(b) Capability emergence of 7B base model.

Figure 9 Scaling analysis of Lumine-Base trained under non-history setting during pre-training stage. The left figure presents loss curves and corresponding success rates on the full set of benchmark tasks as the amount of Genshin training data increases in the first epoch of pre-training for both 2B and 7B base models, while the right figure illustrates the progressive improvement of key abilities for 7B base model, highlighting distinct scaling behaviors across different capability types.

- **GUI Manipulation.** Tests the agent's ability to navigate and control graphical user interfaces meaningfully, especially for mouse movement, e.g., clicking options to continue dialogue, reviving characters, and navigating menu systems.
- **Game Mechanism.** Uses puzzles to examine agents' understanding of core game mechanics, such as switching to the appropriate elemental character to activate an elemental monument, or using a Pyro character to light a torch.
- **Visual Guidance Following.** Evaluates the agent's ability to follow visual directional cues, such as golden quest markers shown in the overworld. Specifically, it tests whether the agent can rotate the camera to center the marker in its view and move forward along the indicated direction.
- **Obstacle Avoidance.** Evaluates the agent's spatial awareness and locomotion skills across four aspects: keep following the path road without deviation, avoiding obstacles such as trees and walls, stopping safely at the edge of cliffs or rivers, and deploying a glider promptly during falls.

The evaluation was conducted through human analysis of the models' gameplay videos, with performance rated on a 5-point scale:

- **Score 1**: The agent exhibits no task-relevant behavior or intent, resulting in complete failure.
- **Score 2**: The agent shows initial intent but fails to make meaningful progress or complete the task.
- **Score 3**: The agent displays relevant behaviors but with significant hesitation and frequent errors, leading to a low success rate.
- **Score 4**: The agent demonstrates generally appropriate behavior with only occasional mistakes, achieving a high success rate.
- **Score 5**: The agent fully masters the capability, performing immediate and accurate actions to complete the task with proficiency.

Figure 9b shows that all core capabilities strengthen as scale increases, but at notably different rates, reflecting the relative difficulty of mastering each skill.

- **Immediate Interaction (< 100 hours).** Simple behaviors such as basic object interaction emerge quickly. At this stage, agents can consistently gather nearby resources, whether local specialties found in the wild or items dropped by enemies, and interact with NPCs in their vicinity by pressing key `F`.
- **Extended Interaction (∼ 1,000 hours).** More complex behaviors, such as basic combat and UI manipulation, become smooth and reliable. Agents can switch characters to chain skill combos, execute ranged attacks against elevated or distant targets. They also manage to handle common basic GUI events, from moving the cursor to select dialogue options, to clicking the close button in the top-right corner, or choosing Confirm or Cancel buttons when responding to interface prompts, such as during a character revival.
- **Game Mechanics (> 1,800 hours).** Game mechanics pose remarkable challenges. Agents can recognize game-specific puzzle elements and exhibit reasonable reactions, but their sparse occurrence in raw gameplay, coupled with the wide variety present in Genshin Impact, makes these mechanics significantly harder for agents to master and complete the full task.
- **Navigation (> 1,800 hours).** Navigation, the most essential component for open-world exploration and quest progression, requires significantly more data. By this point, agents exhibit robust road sense: they tend to follow the in-game roads to proceed, avoid obstacles such as trees and walls, halt at cliff edges, use the wind glider to prevent fall damage, and follow quest markers efficiently. This reliable navigation ability establishes a solid foundation for longer-horizon tasks. It sometimes exhibits hesitation, inefficient stamina management and mistimed actions, leaving room for further improvement.

8.2 Instruction Following Performance

After large-scale pretraining, Lumine-Base-NonHis has already developed the fundamental capabilities needed to accomplish a wide range of tasks. We next investigate whether Lumine-Instruct-NonHis can effectively follow human instructions and how its performance improves through alignment with language supervision. Our evaluation begins with the model under the non-history setting, followed by an extension to the history setting.

8.2.1 Non-History Performance

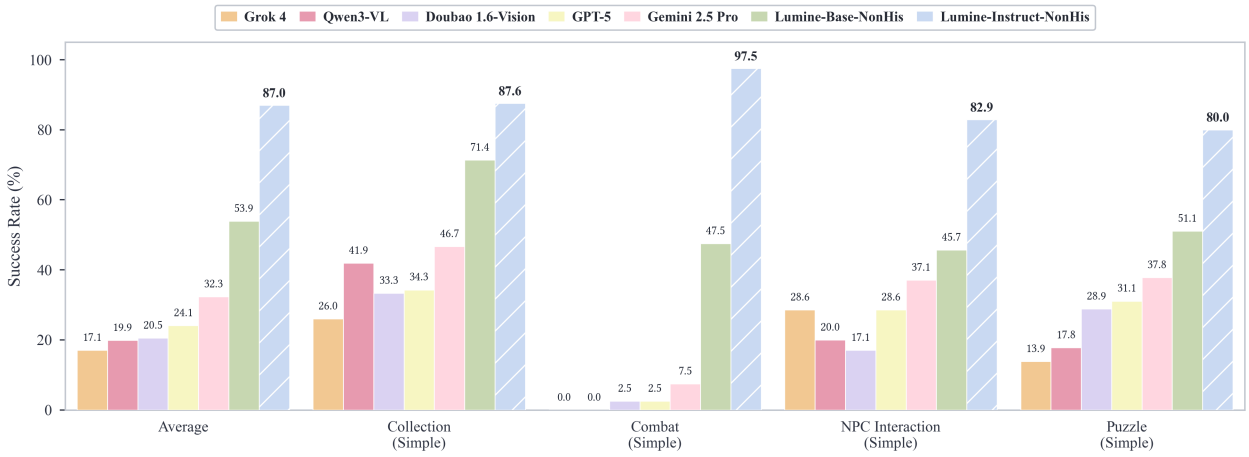


Figure 10 Average success rate of agents on the benchmark simple tasks by categories. Lumine-Instruct-NonHis achieves over 80% success across all four categories, significantly outperforming its base model and all baseline methods.

Baselines Comparasion. As shown in Figure 10, we evaluate both Lumine base and instruct models trained under non-history setting, against multiple baseline models on the benchmark's simple tasks. While Lumine-Base-NonHis already significantly outperforms all baselines, the instruct model further achieves a 61% performance gain, reaching over 80% success across all four categories. The largest improvement occurs in *combat* tasks, where the instruct model doubles the success rate of the base model. The instruct model

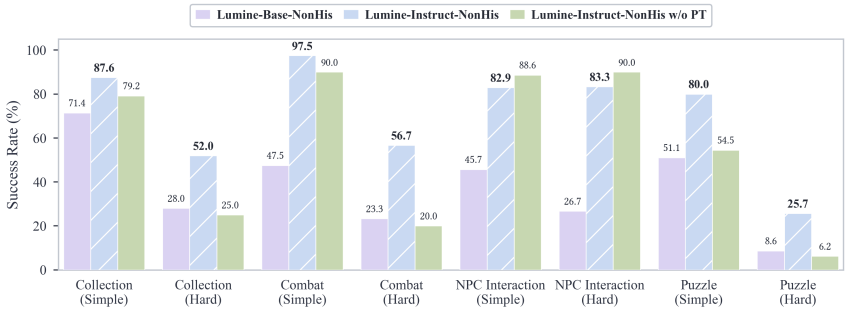


Figure 11 Performance of Lumine-Base, Lumine-Instruct and Lumine-Instruct without pre-training on simple and hard tasks under non-history setting.

Figure 12 Comparison of Lumine-Instruct-NonHis performance in simple and unseen tasks.

demonstrates more consistent goal-conditioned behavior. For example, after battles, it actively adjusts the camera to locate and open the unlocked treasure chests, a step often neglected by the base model, which instead tends to just move forward. For *NPC Interaction* and *Puzzle* tasks, Lumine-Instruct-NonHis achieves more than a 50% gain in performance, showing more consistent engagement with targets. Even in Collection tasks, where the base model already performs strongly, the instruct model still provides a 23% improvement. These findings demonstrate Lumine-Instruct-NonHis's robust ability to follow language instructions, maintain behavioral consistency, and reliably complete diverse common short-horizon tasks.

Performance on Hard Tasks. We further evaluate Lumine-Instruct-NonHis on hard tasks shown in Figure 11. The model exhibits consistent performance in *Interaction* tasks, highlighting its robustness: even when surrounded by multiple non-target NPCs acting as distractors, it can still successfully engage with the designated NPC. More pronounced limitations appear in *Combat* and *Collect*. Although the model manages to defeat tough elite enemies such as *Ruin Guard*, *Debt Collector*, and *Fatui Electro Cicin Mage*, it sometimes fails due to an insufficient understanding of game mechanics and poor dodging responses, often resulting in timeouts or even full-party eliminations. The model exhibits a zero success rate against the Eye of the Storm on elevated terrain, a flying enemy that lands only briefly. While the agent correctly switches to the ranged character Amber, its limited efficiency and precision in moving-target aiming, coupled with delayed evasive actions, often lead to significant damage or character death. The elevated terrain further increases difficulty by punishing movement errors with the risk of falling. These challenges demand precise timing, spatial awareness, and adaptive strategy, underscoring the need for further improvement in combat capabilities. The drop of *Collection* tasks also expose deficiencies in spatial reasoning and fine-grained control, particularly when retrieving items that are not located on flat ground. Performance on *Puzzle* tasks drops the most, reflecting their comprehensive requirement for the abilities above and emphasizing the importance of a deeper understanding of in-game mechanisms.

Figure 11 also shows the impact of pre-training through an ablation study. As expected, models trained exclusively on instruction-following data exhibit overall lower performance, particularly on hard tasks. Interestingly, however, the model achieves slightly better performance in *NPC Interaction* tasks. This suggests that the pre-training data introduces a notable bias, where many trajectories involve players merely passing by NPCs without engaging in interaction. The instruction-following data helps to mitigate this bias by reinforcing goal-directed behaviors and promoting more deliberate interactions with the environment.

Performance on Unseen Tasks. We further evaluate Lumine on unseen tasks. As shown in Figure 12, the model exhibits strong generalization capabilities in *NPC Interaction*, achieving performance comparable to in-domain settings. This indicates that such interaction abilities can be effectively transferred to new scenarios and entities. In contrast, we observe a moderate but acceptable performance drop in *Collection* tasks. Unlike *NPC Interaction*, where unseen NPCs can still be identified by the names displayed above them, new collectibles must be recognized solely by their visual appearance, which the model has never encountered during training. Consequently, the agent must approach potential targets closely until their names are revealed to confirm correctness, significantly increasing task difficulty. Performance in *Combat* tasks also declines, as Lumine-Instruct-NonHis is unfamiliar with previously unseen enemy attack patterns

Initial State	General Instruction	Success Rate	Detailed Instruction	Success Rate
	收集位于空中的风神瞳。 Collect the Anemoculus floating in the air.	20%	爬上右侧的石柱，到达最高处后收集位于左侧空中的蓝色风神瞳。 Climb the stone pillar on the right and, once you reach the top, collect the blue Anemoculus floating in the air on the left.	100%
	收集前方浮在水面上的风神瞳。 Collect the Anemoculus floating on the water ahead.	0%	切换角色为凯亚，不断释放E技能冻结水面，以收集前方浮在水面上的风神瞳。 Switch to Kaeya, continuously use his Elemental Skill (E Skill) to freeze the water surface, and collect the Anemoculus floating ahead.	20%
	收集铁矿石。 Collect Iron Ore.	40%	击打铁矿石，拾取掉落的铁矿石。 Hit the Iron Chunk and collect the dropped Iron Chunk.	100%
	收集位于风障内的宝箱。 Collect the chest within the Wind Barrier.	40%	收集风种子以激发风场，进入风障内，开启宝箱。 Collect the Wind Anemograna to activate a Wind Current, then enter the Wind Barrier to open the chest.	80%

Figure 13 Case study of the in-context abilities of Lumine-Instruct-NonHis. When provided with additional contextual details that are relevant to the instruction, Lumine demonstrates improved performance and is able to complete previously low-performing tasks more effectively. The instructions given to Lumine were originally in Chinese; their English translations are provided here for reference.

and combat mechanics, preventing it from reasoning about and reacting appropriately to new behaviors. Overall, while some degradation is observed, the results remain within an acceptable range, demonstrating Lumine's strong generalization to new scenarios and objects.

Complex Instruction Following. We observe that, beyond additional training, Lumine's performance can further be improved through in-context learning. As shown in Figure 13, providing more detailed instructions, incorporating prior knowledge or task decomposition that breaks a complex objective into a sequence of manageable steps, enables the model to successfully complete tasks that previously had low or even zero success rates. This demonstrates the strong generalization ability introduced by language grounding and establishes a solid foundation for subsequent reasoning training.

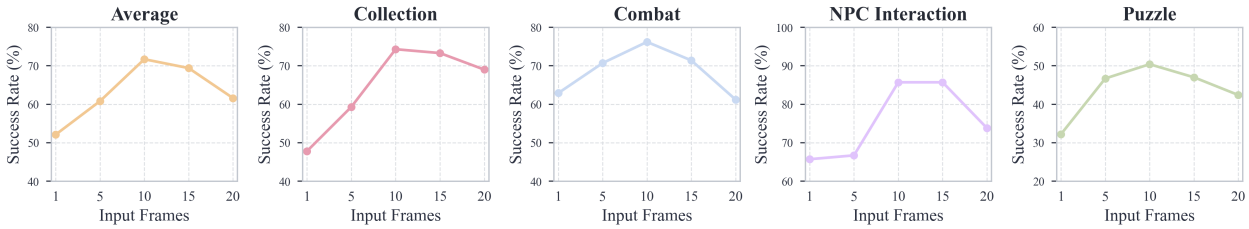


Figure 14 Performance of Lumine-Instruct preserving different lengths of frames in the context as historical information on the full set of benchmark.

8.2.2 Benefits of History

Since Lumine-Instruct-NonHis already demonstrates strong performance across a wide range of tasks, we next investigate whether incorporating historical information can further enhance model effectiveness.

As shown in Figure 14, models that preserve multiple historical frames and actions in the context achieve

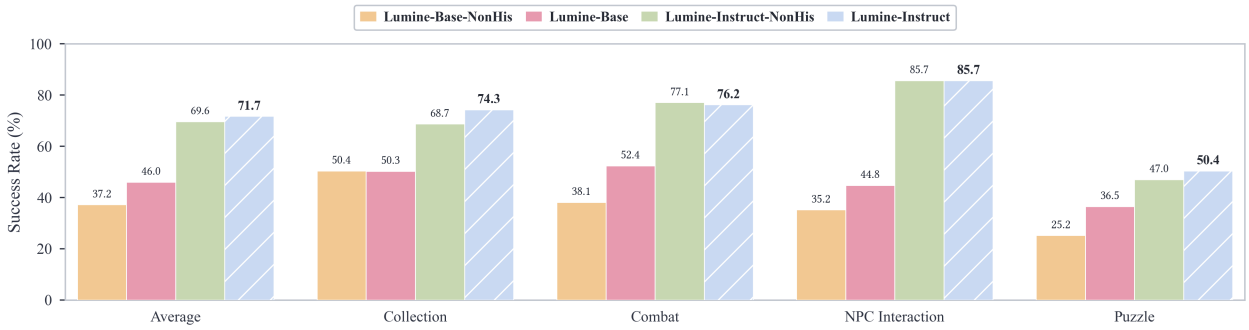


Figure 15 Comparison of Lumine trained under non-history and history settings on the full set of benchmark tasks.

substantially higher performance across all tasks compared to those limited to a single frame. Performance peaks when the model maintains 10 frames in context and starts to drop with more frames in the context. This decrease may be related to the data distribution: after the action filtering operation, which removes 95% noop and trivial movements, the data segments typically less than 20 frames, making it harder for the model to learn long-term dependencies effectively at longer range.

We then evaluate the best-performing configuration, the history model with 10 frames of context, on the full set of benchmark tasks, comparing it against the non-history baseline. Figure 15 shows that history model exhibits a clear advantage in *Collection* and *Puzzle* tasks, achieving overall better performance.

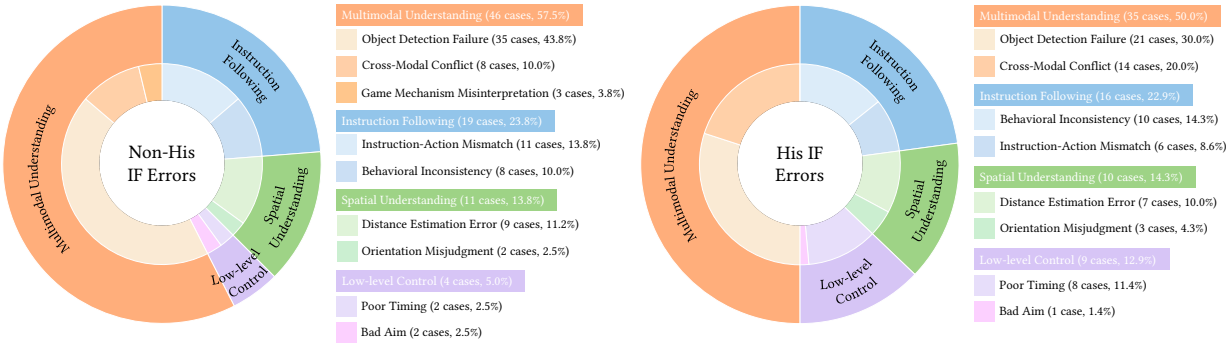


Figure 16 Error analysis of Lumine instruct models trained under non-history and history settings on the full set of benchmark tasks.

Error Analysis. As shown in Figure 16, we conduct a comprehensive error analysis of the Lumine-Instruct models to better understand their behaviors across the full set of benchmark tasks. For tasks involving multiple mistakes, we report only the primary error that prevents further progress.

Our analysis reveals several key failure modes. The most significant is limited **Multimodal Understanding**, which accounts for nearly half of all errors. Agents frequently fail to detect target objects within the visual scene, particularly small, gatherable items that blend into the environment. Once the agent incorrectly infers that the target is not in view, it often rotates by a large angle to search, inadvertently causing already visible objects to move out of sight, making recovery difficult. History models show significantly fewer such errors, indicating that history improves dynamic tracking ability. Another common issue in multimodal understanding arises from conflicts between language and vision modalities. In these cases, the agent is overly influenced by vision, neglecting textual information embedded in images. For instance, when interacting with NPCs, agents may choose the wrong character despite the correct name being clearly visible. Interestingly, models without history make fewer of these modality-conflict errors, suggesting that historical context encourages a stronger focus on temporal consistency at the expense of single-frame understanding. The second major failure mode is **Instruction Following**. Both models occasionally demonstrate inconsistency when executing

tasks. For example, after defeating two of three required *Anemo Slimes*, an agent may abruptly abandon the objective and wander elsewhere. Agents also sometimes ignore fine-grained instructions, such as "pick up the flower on the left", instead turning away or moving forward in search of alternative targets. The third source of errors is **Spatial Understanding**, where agents misjudge their own position or fail to estimate distances accurately. This leads to poor navigation and timing errors, such as jumping too early or too late, thereby missing critical opportunities for progress. Finally, **Low-Level Control** contributes the smallest portion of failures. Both models exhibit unstable shooting performance and occasionally fail to interact with or collect objects, often due to delayed key presses. However, the history model makes noticeably more of these errors than the non-history model, which explains its lower performance on combat-related tasks. Overall, these findings highlight the diverse weaknesses across perception and control layers, leaving substantial room for future improvement.

8.3 Reasoning Performance

Lumine-Instruct models exhibit strong performance in language following and can successfully complete a wide range of short-horizon tasks across the world of Genshin Impact. Building on this foundation, we investigate whether Lumine-Thinking can address more challenging long-horizon tasks through its adaptive thinking capability. For evaluation, we primarily use the game's main storyline as a testbed, an experience that typically takes human players hours to complete and encompasses most major events and gameplay mechanics. The storyline evolves rapidly, with both scenarios and objectives changing frequently, requiring the agent to react adaptively rather than relying on a fully pre-planned task decomposition. Such extremely long tasks serve as a comprehensive test of the agent's capabilities and represent a critical benchmark for evaluating true autonomy.

Setting. Lumine trained under history setting is set with a maximum of 20 historical frames in the context. Thinking models will be forced to enter thinking mode when it does not generate a new reasoning for more than 100 steps, which is useful to help it recover from a stuck state. All the experiments are run in a real-time setting. Each model is provided with "Complete the main storyline mission" as an instruction in the prompt at the beginning of each task, which will be overridden by the following reasoning. Throughout the evaluation, agents are restricted to the four default system-provided characters, Traveler, Amber, Kaeya and Lisa, with their progression aligned to what a typical human player would reasonably have achieved at that stage.

Figure 17 Visualization of the in-domain evaluation mission, the main storyline of Mondstadt, *Prologue: Act I - The Outlander Who Caught the Wind*, which is divided into five subtasks. The left figure illustrates the agent's geographical trajectory during task completion. Red lines denote the character's movement path, while blue lines indicate teleportation jumps between distant locations. The right figure presents the complete trajectory and corresponding timestamps for Lumine-Thinking, who completed the mission in **56** minutes, compared with fresh human players with an average of **78** minutes and expert human players with an average of **53** minutes.

8.3.1 Performance in Genshin Impact

In-domain Performance. We first evaluate Lumine-Thinking on missions that are covered in both the pretraining and reasoning datasets. As illustrated in Figure 17, we select the main storyline of Mondstadt *Prologue: Act I - The Outlander Who Caught the Wind* as the primary testbed, where players are required

Table 5 Success rate of Lumine on the five subtasks of the in-domain mission. Each task is run three times.

Model	Overall	Task 1	Task 2	Task 3	Task 4	Task 5
Lumine-Instruct-NonHis	6.6%	0/3	0/3	0/3	1/3	0/3
Lumine-Thinking-NonHis	53.4%	1/3	2/3	2/3	2/3	1/3
Lumine-Instruct	66.8%	2/3	**3/3**	2/3	2/3	1/3
Lumine-Thinking	**93.4%**	**3/3**	2/3	**3/3**	**3/3**	**3/3**

to create a new account and progress entirely from scratch. This mission evaluates a broad spectrum of agent capabilities, including long-horizon navigation to quest locations, NPC interaction, instruction following, combat, boss fights, puzzle solving, domain exploration, and GUI operations such as character development, map teleportation, and dialog continuation. Completing this mission typically takes human players about one hour. To systematically evaluate agent performance, we divide the mission into five sequential subtasks:

- **Task 1:** The Traveler follows Paimon across Starfell Lake and through the forest toward Mondstadt. Along the way, they meet Amber, join forces to defeat monsters, and ultimately reach the city.
- **Task 2:** The Traveler helps defend Mondstadt during Stormterror's sudden attack, after which they are invited to the Knights of Favonius Headquarters to offer assistance against the escalating dragon threat.
- **Task 3:** The Traveler teams up with Amber to investigate and clear the *Temple of the Falcon*.
- **Task 4:** The Traveler meets Kaeya and explores the *Temple of the Wolf* under his guidance.
- **Task 5:** The Traveler joins Lisa in delving into the *Temple of the Lion*.

As shown in Table 5, thinking models achieve significantly better performance than instruct models, highlighting both the effectiveness and necessity of high-level reasoning. While instruct models can handle certain subtasks, they often fail to complete the full task chain. This failure typically arises from distractions encountered along the way, such as puzzles or enemies, which cause the model to lose track of the main objective. Once disoriented in the overworld, the model tends to wander aimlessly and struggles to recover the correct progression path. In contrast, thinking models are able to reflect and set appropriate goals for the current context and remain focused, avoiding such distractions. Lumine-Thinking with history demonstrates robust performance across all tasks, with only a single failure observed in Task 2. In that case, the model accidentally fell off a platform while traveling to the target quest location, Knights' Headquarters and crucially failed to generate intermediate reasoning steps to invoke the in-game quest guidance. Subsequently, it roamed the city in search of a way back, resulting in a timeout. Nevertheless, given additional time, the model would likely have recovered its route and completed the task successfully. Overall, Lumine-Thinking trained under history setting successfully completed the entire act in **56** minutes, outperforming fresh human players, whose average completion time was **78** minutes, and performing comparably to expert human players, who averaged **53** minutes. The expert group had substantial prior experience with the game and had played through the same story segment at least once within the preceding week.

On the other hand, models with history exhibit a clear advantage under real-time conditions, particularly in GUI manipulation and navigation. By contrast, non-history models are significantly affected by the 200ms delay introduced by asynchronous execution. Specifically, when a non-history model attempts to move the mouse to a designated position, the visual input it receives corresponds to the pre-execution state of the previous action. As a result, the model incorrectly believes that the cursor has not yet moved and generates another mouse movement action in the same direction. This leads to overshooting the target and makes precise clicking on GUI elements considerably more difficult. Since models with history have access to previous actions and trajectories in context, they are far less affected by latency and can perform operations much more smoothly and stably. The same phenomenon also occurs in navigation when turning directions.

Error Analysis. As shown in Figure 18, we conducted an error analysis of the reasoning quality generated by Lumine under both non-history and history settings. During the completion of the entire mission, the history model generated 593 reasoning instances with an error rate of 8.8%, which is significantly lower than

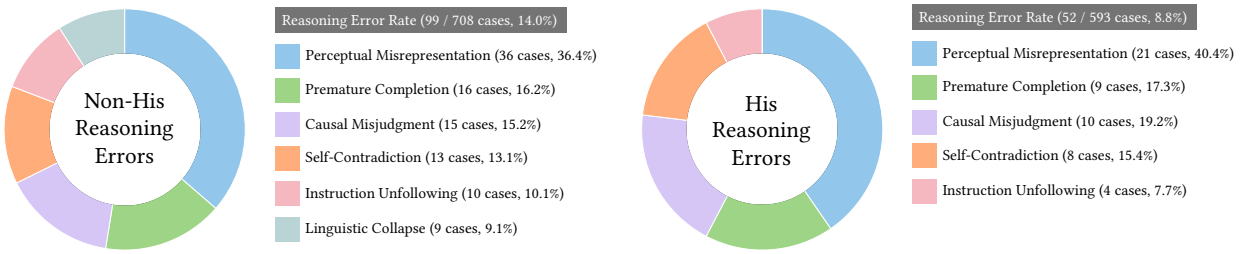


Figure 18 Error analysis of reasoning quality generated by Lumine during the completion of the entire in-domain mission. We investigate both non-history and history settings.

that of the non-history model (708 instances, 14.0% error rate) in both quantity and quality. These results demonstrate the advantage of the history model, which produces reasoning that is not only more efficient but also more accurate. It is worth noting that low-quality reasoning does not necessarily lead to task failure.

In terms of error composition, both settings show similar contributing factors and relative proportions. The most frequent error type in both settings is **Perceptual Misrepresentation**, which occurs when the model describes objects or scenarios that do not exist in the observation or misinterprets the character's current status (e.g., health or stamina). This reflects the model's limited situational understanding of the game environment. The second is **Premature Completion**, a type of hallucination in which the model incorrectly assumes that a proposed goal has already been achieved. Based on this false premise, it continues to generate reasoning that builds upon the nonexistent completion, thereby compounding the initial error. The third category, **Causal Misjudgment**, refers to instances where the model incorrectly believes that a given action or strategy will accomplish the intended outcome. These errors often arise from an incomplete understanding of the system's causal mechanisms or underlying dynamics. **Self-Contradiction** describes reasoning that is internally inconsistent. For instance, the model sometimes claims, "I closed the task interface before... so I did not close the task interface," producing an inherently impossible statement that exposes the base model's limited capacity for logical reasoning. We also observed several instances of **Instruction Unfollowing**, where the model's generated actions failed to align with its own reasoning. Finally, we occasionally observed **Linguistic Collapse** in the non-history model where the model produced unreadable or incoherent outputs. This suggests that reasoning based solely on single-frame information imposes greater cognitive strain, making the model more prone to collapse. In contrast, the model trained under the history setting demonstrates greater robustness and stability.

Generalization to Reasoning OOD Missions. As Lumine demonstrates superior performance in trained tasks, we then investigate whether the model's reasoning abilities can generalize to previously unseen scenarios and the model's performance in extremely challenging tasks. We apply our best-performing model, Lumine-Thinking, to the remaining main storyline of Mondstadt *Prologue: Act II - For a Tomorrow Without Tears* and *Prologue: Act III - Song of the Dragon and Freedom*, where the Traveler and their companions overcome various difficulties along the way in search of a way to purify the corrupted dragon, Dvalin. They eventually confront Dvalin in his lair and succeed in purifying him, thereby lifting the threat to Mondstadt. This part of the gameplay is included in the pre-training data but excluded from the reasoning data. To ensure uninterrupted progression, we prepare a game account with an Adventure Rank of 18, allowing the agent to access and complete the quests without being blocked by rank requirements.

Lumine successfully completed the two acts consecutively within **4.7** hours, compared to an average of **3.6** hours for expert human players. During the course of these missions, Lumine also attempted to solve puzzles, open monster-guarded chests, and collect Oculus encountered along the way. It is encouraging to observe that Lumine consistently stayed on the right track, allowing the mission progress to advance smoothly, which indicates reasoning abilities manage to transfer to unseen scenarios, though with more hallucinations as expected. Reasoning tends to be precise in NPC interactions, GUI operations, common navigation, and combat, but becomes less reliable in unfamiliar mechanisms and puzzles, such as the wind wall in Stormterror's Lair, where Lumine relies more on its low-level control. Interestingly, as shown in Figure 20, in one of the dungeons of Act II, there is a floating moving stone platform with a mechanism similar to that found in Lisa's

(A-1) Prologue: Act II - *For a Tomorrow Without Tears*

(A-2) Prologue: Act III - *Song of the Dragon and Freedom*

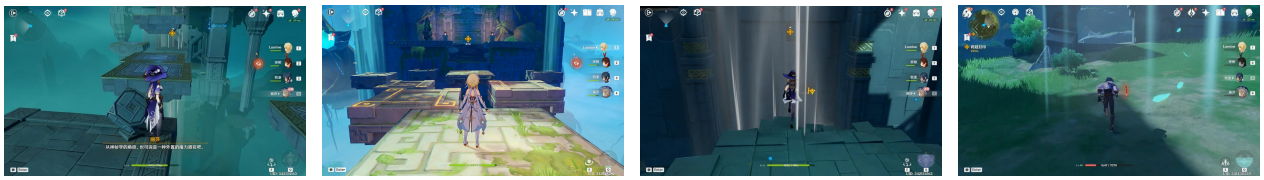
(B) Chapter I: Act I - *Of the Land Amidst Monoliths*

Figure 19 The two missions on the left, Acts II and III of Mondstadt's main storyline, are included in the pre-training data but excluded from the reasoning data. Lumine successfully completed the two acts consecutively within **4.7** hours, compared to an average of **3.6** hours for expert human players. While Liyue's mission on the right is entirely new to Lumine, it still manages to reach Liyue Harbor and visit the Adeptus dwelling deep within the mountains. Due to space limitations, we are unable to present the full process of Lumine's journey from Mondstadt's Windrise to Liyue. The red dashed line indicates a round trip between two locations.

dungeon in Act I. Despite the completely different layout and background, Lumine successfully recognized the stone platform and explicitly reasoned that we should wait for the platform to move before proceeding. The wind current also tells the same story. This demonstrates the strong generalization capability of the VLM-based agent.

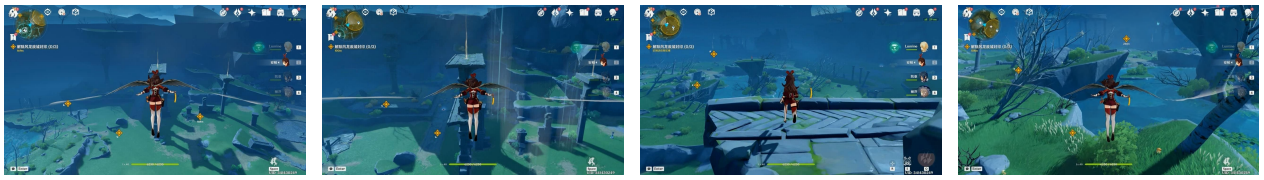
From Lumine's gameplay, we identify five key factors that account for the primary inefficiencies preventing it from achieving expert human-level performance in these missions:

- **Lack of proactive fast travel.** Lumine rarely makes use of teleportation for fast travel, instead following quest markers on foot even across long distances. However, we did observe a few successful cases of teleportation, suggesting that the capability already exists but is not triggered properly. This is mainly due to the lack of such a pattern in the first hour of gameplay in the reasoning data.
- **Limited understanding of the minimap.** While Lumine shows some awareness of the minimap, its use is unreliable, leading to heavy dependence on the golden quest marker for navigation. If the quest marker disappears and Lumine fails to generate a new reasoning to recall it again, the agent is likely to move in the wrong direction and stray far from the intended target.
- **Lack of proactive health recovery.** Lumine does not actively restore the health of party members. This omission is closely related to the first issue, as the agent does not teleport to Statues of The Seven

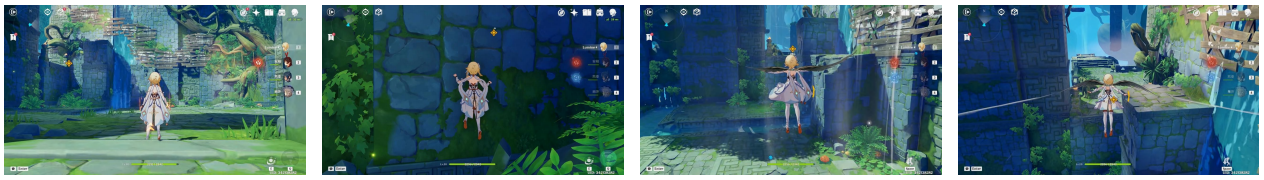


(a) Moving platform (ID) **(b)** Moving platform (OOD) **(c)** Wind current (ID) **(d)** Wind current (OOD)

Figure 20 Two examples of same mechanisms encountered by Lumine in both in-domain (ID) and out-of-domain(OOD) scenarios. Even across vastly different contexts and scenarios, Lumine-Thinking demonstrates strong generalization ability, successfully recognizing mechanisms in OOD settings and generating reasoning that effectively leverages them.



(a) Three golden quest markers appear simultaneously, each pointing to a different location. When the agent reaches one region of them, the corresponding marker disappears, leaving the other two active. The agent is often drawn toward these remaining markers and repeatedly moves back and forth among the three regions.



(b) When following the golden quest marker directly, the agent encounters a wall and must take a detour using the wind current on the right. Lumine often abandons the detour midway and returns to the starting point, drawn back by the quest marker's signal since it has forgotten that the direct path is blocked.

Figure 21 Two examples illustrating the importance of long-term memory for successful task completion.

(which can heal the party) nor open the inventory to consume food items for healing. These recovery patterns are also notably absent from the reasoning data.

- **Limited memorization.** As shown in Figure 21, in Act II and III, certain missions present multiple simultaneous quest markers shown in the image, and Lumine is easily distracted, oscillating between different targets. Additionally, in cases where the quest marker is directly ahead but requires a detour due to obstacles, Lumine often abandons the detour midway and returns to the starting point, drawn back by the quest marker's signal. Although Lumine demonstrates diverse strategies, strong exploration ability, and eventually manages to get out of such situations, these behaviors highlight the limitation of its four-second (20 frames) memory span and underscore the need for more effective long-term memory mechanisms.
- **Combat proficiency to be improved.** Although Lumine achieves a high success rate of combat across various combinations of enemies in the benchmark, its efficiency remains limited in aspects such as skill coordination, aiming accuracy, dodging timing and mechanics understanding. These shortcomings become more pronounced in multi-wave encounters and boss fights, where Lumine performs noticeably below the level of expert human players.

Generalization to Fully OOD Mission. We further extend the evaluation to missions entirely absent from the training data, thereby testing the agent's ability to generalize to completely novel scenarios. After successfully completing the full Mondstadt main storyline, the next mission directs the player to travel to the nation of Liyue. As shown in Figure 19, Agents are then seamlessly tasked with journeying from Mondstadt to Liyue

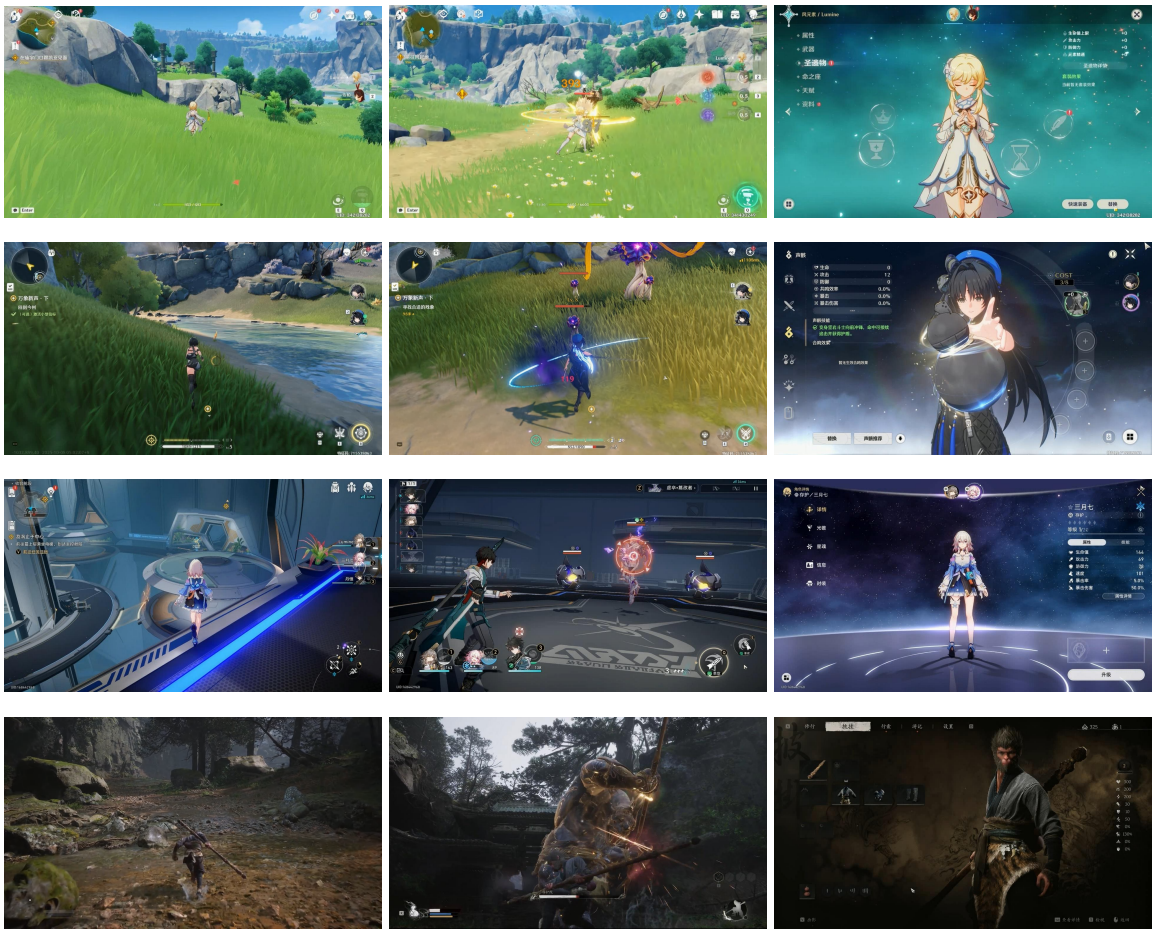


Figure 22 Demonstration of overworld navigation, combat, and UI in four games, Genshin Impact, Wuthering Waves, Honkai: Star Rail, and Black Myth: Wukong. Similar to Genshin Impact, Wuthering Waves is also an open-world ARPG, while Honkai: Star Rail is a turn-based RPG that combines strategic combat with a hub-based world design. Black Myth: Wukong is a hub-based ARPG but features a more realistic visual rendering style.

Harbor and advancing as far as possible in the Liyue main storyline. Since Liyue is entirely excluded from the training data, it presents a challenging OOD environment with new regions, stories, NPCs, enemies, and puzzles. Compared to the relatively flat terrain of Mondstadt, the mountainous landscape of Liyue poses significantly greater challenges for the agent.

Lumine impressively demonstrates in-domain level efficiency during the first hour of gameplay, where it must complete an extremely long-distance navigation from Mondstadt to Liyue Harbor, evade capture by the Millelith, and locate Tartaglia hidden on the second floor. Though less efficient, Lumine even manages to find the Adeptus dwelling deep within the mountains, after a long journey across rugged terrain and rivers with highly noticeable ups and downs. On the way to visit the second Adeptus, Lumine accidentally cancels quest tracking due to a hallucination. When attempting to reactivate it, the model outputs a left-click command combined with mouse movement, which the game misinterprets as a drag action, causing the operation to fail. However, the agent is overconfident that the task had been activated and subsequently closed the quest menu, leading to aimless exploration of the map for approximately two hours. Eventually, it manages to reopen the quest interface, activate quest tracking, locate the second Adeptus, rescue the person trapped in amber, and complete the mission.

8.3.2 Generalization to New Games

Finally, we investigate whether Lumine, though trained exclusively on Genshin Impact gameplay data, can generalize and be deployed in other unseen games without additional training and without any prompt modification. As shown in Figure 22, we carefully select three games, *Wuthering Waves*, *Honkai: Star Rail* and *Black Myth: Wukong*, that represent diverse genres, mechanics, and levels of similarity to Genshin Impact.

Wuthering Waves. Similar to Genshin Impact, Wuthering Waves is also an online 3D open-world ARPG that shares many structural similarities in terms of control schemes, combat loops, and gameplay rhythm. This makes it a natural testbed for measuring how well Lumine adapts to games that are mechanically aligned but distinct in content and setting. For our evaluation, Lumine began with a fresh account and started from the very beginning of the game. We would also like to mention that the launch date of Wuthering Waves was later than the knowledge cutoff of Qwen2-VL, so Lumine is unlikely to leverage prior knowledge about this game from the base model pretraining.

In this unseen game, Lumine impressively demonstrated in-domain-level efficiency during the first two main storyline missions, *First Resonance* and *Echoing Marche*, where Lumine completes these missions in **107** minutes, compared to an average of **101** minutes for fresh human players.[1] Lumine's diverse capabilities generalize seamlessly to these unseen game missions, in both 3D overworld and 2D GUI, enabling it to follow in-game guidance to reach quest target location, perform key presses and icon selections according to hints, defeat enemies, unlock fast travel points, interact with objects and NPCs, level up characters, manage equipment and explore the open world.

While generally efficient, several mistakes were observed. A notable issue arises from the game's distinctive rendered style: the agent occasionally misinterprets the on-screen prompt to press key `F` for interaction as key `E`. Since `E` triggers skill usage, this misinterpretation can cause gameplay stalls, highlighting the importance of robust OCR capabilities. Other errors come from domain bias and hallucination in reasoning, where Lumine borrows terminology from Genshin Impact to describe entities in Wuthering Waves. For instance, it refers to monsters as Hilichurls, the protagonist Rover as Traveler, and the red-clad girl in the team as Amber. Interestingly, such cross-game naming confusion is also commonly seen among human players.

Honkai: Star Rail. Different from the previous two games, Honkai: Star Rail is an online 3D turn-based strategy role-playing game. Instead of a fully open-world structure, it adopts a hub-based exploration design, in which the game world is divided into discrete zones connected through hubs rather than presented as a seamless world. Movement is also more restricted, as characters cannot jump or climb over obstacles. While the game retains some control schemes and interface conventions from Genshin Impact, its core combat system emphasizes turn-based strategy over real-time action.

Although less efficient than Wuthering Waves due to the significant domain gap, Lumine still surprisingly demonstrates reasonable performance in 3D navigation, NPC interaction, and GUI manipulation, abilities that are essential for progression. Lumine successfully completed the entire first chapter, *Today is Yesterday's Tomorrow*, in the Herta Space Station, win the Boss fight of Doomsday Beast, cleared the Simulated Universe tutorial, and even progressed into the second chapter, reaching a new planet, Jarilo-VI, using **7** hours, compared to the **4.7** hours typically taken by fresh human players.

The main bottlenecks hindering smooth gameplay arise in navigation and combat, primarily due to domain bias. In several scenarios, Lumine attempts to jump across gaps toward a quest marker on the opposite platform, only to be blocked by an invisible wall. While such behavior works in Genshin Impact, it is impossible in Honkai: Star Rail, which lacks a jump mechanic and is not an open-world. Although Lumine sometimes adapts by taking an alternate route, it more often returns to the same spot, repeatedly drawn by the quest marker and constrained by limited memorization. Encouragingly, after enough attempts, Lumine eventually manages to get unstuck and reach the correct location, despite the distraction.

[1]The estimated average playtime is based on five human players and should be regarded only for reference. All the players have extensive gaming experience, but none of them played this game before. Players have different playstyles: some tend to skip all the dialogues and the story, while others prefer to immerse themselves fully in the narrative. Some focus primarily on the main storyline, whereas others take time to clear enemies, solve puzzles, and collect chests along the way. As a result, gameplay length can vary significantly.

Combat presents even greater challenges. While Lumine can reliably distinguish allies from enemies, the turn-based combat system differs drastically from real-time combat in Genshin Impact and causes the greatest delays. Fortunately, the two games share some overlapping keybinds: in Genshin Impact, normal attacks use mouse clicks, Elemental Skills use `E`, Elemental Bursts use `Q`, and characters can be swapped with numeric keys `1-4`. In Honkai: Star Rail, however, `Q` is used for normal attacks, `E` for skills, and `1-4` for each character's ultimate ability. This overlap allows Lumine to stumble through combat using its prior Genshin experience, but targeted skills pose a serious obstacle. It fails to understand that pressing `E` requires selecting a target, or that using an ultimate requires confirming with the spacebar. Still, Lumine occasionally presses the correct keys by chance, allowing battles to continue. These difficulties were most pronounced in the final boss fight against the Doomsday Beast, where repeated misunderstandings led to multiple party wipes. Eventually, Lumine chose to lower the game's difficulty and, by the narrowest margin, secured victory in the final battle.

Black Myth: Wukong. Finally, we evaluate Lumine in a much more challenging game, Black Myth: Wukong, a single-player 3D action role-playing game with highly realistic graphics, in contrast to the more stylized, animated visuals of the previous titles. While Lumine demonstrates basic competence in both navigation and combat, several factors prevent it from completing longer tasks.

Firstly, similar to Honkai: Star Rail, the game uses a hub-based exploration design rather than an open world. The lifelike scenery combined with pervasive invisible walls often causes Lumine to get stuck, mistakenly assuming it can pass through. Secondly, during navigation, the overworld UI elements such as health and status indicators will automatically hide. The cinematic visuals lead Lumine to misinterpret the scene as a pre-rendered cutscene (CG), at which point it keeps outputting noop actions and waits for the "CG" finish. Thirdly, Lumine struggles to correctly recognize the health bar and has no understanding of how to restore health. This limitation becomes especially punishing in the game's high-difficulty, Souls-like combat, where survival requires enduring multiple waves of enemies. Taken together, these challenges make it particularly difficult for a zero-shot agent to play the game.

Conclusion. Lumine demonstrates strong generalization abilities across different games, even as the similarity between game genres decreases. This indicates that despite variations in visual style and gameplay mechanics, the core skills of navigation, combat, and 2D GUI manipulation can transfer across games, making it a promising foundation model across games. Remarkably, even when trained on a single game, Lumine is able to adapt, highlighting its promising scalability for broader applications.

9 Discussion and Future Work

We proposed Lumine, an open recipe that spans the entire lifecycle of developing generalist agents in 3D open world environments, from environment selection, data collection and preprocessing to interaction frequency, model design, training procedures, and inference optimization. Lumine offers a unified framework that organically integrates perception, reasoning, and action. The success of Lumine demonstrates that with only 2400 hours of raw gameplay data and 64 H100 GPUs, a small open-source VLM can be seamlessly and efficiently transformed into a powerful agent capable of following natural language instructions in real time to perform diverse tasks and complete hours-long missions in complex 3D open-worlds, without the need to modify model structures or loss functions. Experimental results also reveal substantial potential for further scaling. Lumine's remarkable zero-shot generalization to unseen missions and even entirely unseen games suggests that the model acquires transferable meta-skills, such as 3D navigation and 2D manipulation, that can be readily applied to other domains. This underscores the promise of Lumine's recipe as a pathway toward developing general-purpose decision foundation models.

Lumine is not without its limitations, which also point to several promising directions for future improvement:

- **Scaling.** To experimentally validate the effectiveness of the Lumine recipe, the pre-training data was limited to the Mondstadt region of Genshin Impact, and the reasoning data covered only the first hour of gameplay. There is significant potential to scale both the pre-training and reasoning datasets, not only within Genshin Impact but also across other games and domains, to enhance the model's generalization and robustness.

- **Long-Term Memory.** Lumine currently employs a straightforward memory management mechanism, using previous reasoning steps for long-term memory and a 20-frame context window for short-term memory. While this setup proves effective in most cases, it remains inadequate for complex, long-horizon missions. Future work should explore more sophisticated approaches that enable efficient memory retrieval and management over thousands turns of interactions.
- **Online Learning.** Lumine primarily learns from offline data, which is efficient but limits its ability to surpass human-level performance. Building upon the strong foundation of the existing Lumine model, integrating online reinforcement learning could enable autonomous exploration and continuous self-improvement, further enhancing its performance beyond static, offline learning.
- **Real-Time Inference.** To meet the strict latency requirements of real-time interaction, Lumine applies tensor parallelism across multiple GPUs for acceleration and still suffers from the inference delay. More efficient inference strategies are needed to reduce computational overhead. Improved efficiency would not only facilitate real-time responsiveness but also benefit reinforcement learning through faster rollout generation.

Beyond its research-oriented contributions, Lumine demonstrates strong potential for practical applications. With its generalist perception–reasoning–action capabilities, Lumine can autonomously explore game environments or interpret natural-language instructions to execute gameplay tasks, supporting debug detection, quality assurance, and large-scale usability evaluation in game development. It can identify inconsistencies and ambiguities in tutorials, quest logic, and interaction design, helping developers reduce manual testing costs and improve overall development efficiency. Meanwhile, Lumine also opens up possibilities for new forms of interactive entertainment, such as AI-driven game assistants and streaming where intelligent agents are capable of playing, commenting, and engaging with audiences in real time. These applications highlight Lumine's potential to blur the boundary between player, developer, and audience, paving the way for new human–AI co-experiences in digital worlds.

Ethics Statement

This work strictly adheres to the ethical standards in research involving artificial intelligence systems, human gameplay data, and virtual environments.

All gameplay data used in this study were collected from consenting adult participants who were compensated at fair market rates. Participants were informed that their keyboard, mouse, and screen recordings would be used solely for research on AI agents.

We strictly complied with the terms of service of all games used in this research, and we emphasize that our system is developed for academic research only. Lumine is not intended for cheating or competitive advantage in any commercial product. We explicitly oppose any use of this technology that undermines fair play, game integrity, or player experience.

While we acknowledge the potential risk that such AI systems could be misused as game cheats or unauthorized automation tools, we also recognize that automation and general intelligence represent an inevitable and transformative direction for the future, one that may not only reshape video games but also fundamentally influence how humans and intelligent agents coexist across digital and physical domains.

Therefore, instead of merely preventing misuse, we call for an open, collaborative dialogue within the research, developer, and player communities to explore new frameworks and governance models that balance innovation, creativity, and fairness. We believe that by working together, the community can establish shared ethical standards and technological safeguards that enable the positive and responsible evolution of intelligent agents and game development.

Acknowledgement

We thank Yi Lin, Tianheng Cheng, Yudong Liu, Jingjia Huang for providing valuable suggestions on VLM training. We thank Faming Wu, Xiaojun Xiao, Junjie Fang, Junda Zhang, Yuxing Yao, Yang Yang, and

Xiaoying Jia for their assistance in resource allocation, storage management, model training and inference optimization. We thank Zihao Wang for discussions on model design. We thank Xinrun Wang for discussions on paper writing. We thank Bo Zhou, Zili Li, Yu Miao, Woyu Lin, and Qiaoyu Dong for their contributions to annotator recruitment and management. We thank Heng Zhang and Xiaoyu Tang for their help with local evaluation. We also thank Chao Wu for preparing game accounts used in demo recordings.

Contributions

Authors Weihao Tan[2,*,†], Xiangyang Li[3,*,†], Yunhao Fang[1,*,‡], Heyuan Yao[3,*,†], Shi Yan[3,*,†], Hao Luo[3,*,†], Tenglong Ao[1], Huihui Li[1], Hongbin Ren[1], Bairen Yi[1,‡], Yujia Qin[1], Bo An[2], Libin Liu[3], Guang Shi[1]

Affiliations [1]ByteDance Seed, [2]Nanyang Technological University, [3]Peking University

[*] Core contributors
[†] Work was done during their internship at ByteDance Seed
[‡] Work performed while at ByteDance Seed

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

[2] Philip E. Agre and David Chapman. What are plans for? In MIT AI Memo 1050A / Robotics and Autonomous Systems, 1990.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691, 2022.

[4] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. Advances in neural information processing systems, 35:23716–23736, 2022.

[5] Anthropic. Developing a computer use model. https://www.anthropic.com/news/developing-computer-use, 2024. Accessed: 2025-09-22.

[6] Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet, 2025. Accessed: 2025-05-10.

[7] 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, 2022. URL https://arxiv.org/abs/2206.11795.

[8] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. arXiv preprint arXiv:1612.03801, 2016.

[9] 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.

[10] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680, 2019.

[11] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734, 2025.

[12] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164, 2024.

[13] Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. arXiv preprint arXiv:2506.07339, 2025.

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

[15] 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. Advances in neural information processing systems, 34:15084–15097, 2021.

[16] Peng Chen, Pi Bu, Yingyao Wang, Xinyi Wang, Ziming Wang, Jie Guo, Yingxiu Zhao, Qi Zhu, Jun Song, Siran Yang, et al. Combatvla: An efficient vision-language-action model for combat tasks in 3d action role-playing games. arXiv preprint arXiv:2503.09527, 2025.

[17] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. arXiv preprint arXiv:2401.10935, 2024.

[18] Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. Observations on typing from 136 million keystrokes. In Proceedings of the 2018 CHI conference on human factors in computing systems, pages 1–12, 2018.

[19] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. arXiv preprint arXiv:1611.01779, 2016.

[20] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023.

[21] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Nicolaus Foerster, and Shimon Whiteson. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2023. URL https://openreview.net/forum?id=5OjLGiJW3u.

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

[23] Figure AI. Helix: A vision-language-action model for generalist humanoid control. https://www.figure.ai/news/helix, 2025. Accessed: 2025-09-22.

[24] Pascale Fung, Yoram Bachrach, Asli Celikyilmaz, Kamalika Chaudhuri, Delong Chen, Willy Chung, Emmanuel Dupoux, Hongyu Gong, Hervé Jégou, Alessandro Lazaric, et al. Embodied ai agents: Modeling the world. arXiv preprint arXiv:2506.22355, 2025.

[25] Joaquín M. Fuster. Upper processing stages of the perception–action cycle. Trends in Cognitive Sciences, 8(4): 143–145, 2004.

[26] Google. Gemini 2.5: Our most intelligent ai model. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking, 2025. Accessed: 2025-05-10.

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

[28] David Ha and Jürgen Schmidhuber. World models. arXiv preprint arXiv:1803.10122, 2(3), 2018.

[29] David. Hershey. Claude plays pokemon twitch stream. https://www.twitch.tv/claudeplayspokemon, 2025. Accessed: 2025-09-09.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[31] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. arXiv preprint arXiv:2311.17842, 2023.

[32] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608, 2022.

[33] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. arXiv preprint arXiv:2410.21276, 2024.

[34] Matthias Hutsebaut-Buysse, Kevin Mets, and Steven Latré. Hierarchical reinforcement learning: A survey and open research challenges. Machine Learning and Knowledge Extraction, 4(1):172–221, 2022.

[35] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. arXiv preprint arXiv:2504.16054, 2025.

[36] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. Science, 364(6443):859–865, 2019.

[37] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The Malmo platform for artificial intelligence experimentation. In Ijcai, pages 4246–4247, 2016.

[38] Daniel Kahneman. Thinking, fast and slow. macmillan, 2011.

[39] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. arXiv preprint arXiv:2406.09246, 2024.

[40] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11):1238–1274, 2013.

[41] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474, 2017.

[42] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. Advances in neural information processing systems, 29, 2016.

[43] John E Laird, Allen Newell, and Paul S Rosenbloom. Soar: An architecture for general intelligence. Artificial intelligence, 33(1):1–64, 1987.

[44] Joel Z Leibo, Edgar A Dueñez-Guzman, Alexander Vezhnevets, John P Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charlie Beattie, Igor Mordatch, and Thore Graepel. Scalable evaluation of multi-agent reinforcement learning with melting pot. In International conference on machine learning, pages 6187–6199. PMLR, 2021.

[45] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In International Conference on Machine Learning, pages 19274–19286. PMLR, 2023.

[46] Muyao Li, Zihao Wang, Kaichen He, Xiaojian Ma, and Yitao Liang. Jarvis-vla: Post-training large-scale vision language models to play visual games with keyboards and mouse. arXiv preprint arXiv:2503.16365, 2025.

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

[48] Qianli Ma, Yaowei Zheng, Zhelun Shi, Zhongkai Zhao, Bin Jia, Ziyue Huang, Zhiqi Lin, Youjie Li, Jiacheng Yang, Yanghua Peng, et al. Veomni: Scaling any modality model training with model-centric distributed recipe zoo. arXiv preprint arXiv:2508.02317, 2025.

[49] Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Runji Lin, Yuqiao Wu, Jun Wang, and Haifeng Zhang. Large language models play starcraft ii: Benchmarks and a chain of summarization approach. Advances in Neural Information Processing Systems, 37:133386–133442, 2024.

[50] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. nature, 518(7540):529–533, 2015.

[51] Vernon B Mountcastle. An organizing principle for cerebral function: the unit module and the distributed system. The neurosciences. Fourth study program, pages 21–42, 1979.

[52] OpenAI. Introducing gpt-4.1 in the api. https://openai.com/research/gpt-4-1, 2025. Accessed: 2025-05-10.

[53] OpenAI. Introducing gpt-5. https://openai.com/index/introducing-gpt-5/, 2025. Accessed: 2025-10-24.

[54] OpenAI. Operator. https://openai.com/index/introducing-operator/, 2025. Accessed: 2025-09-22.

[55] Philip Paquette. Super mario bros. in openai gym. https://github.com/ppaquette/gym-super-mario, 2016.

[56] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th annual acm symposium on user interface software and technology, pages 1–22, 2023.

[57] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In International conference on machine learning, pages 2778–2787. PMLR, 2017.

[58] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. arXiv preprint arXiv:2501.09747, 2025.

[59] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8494–8502, 2018.

[60] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. arXiv preprint arXiv:2501.12326, 2025.

[61] Qwen Team. Qwen3: Think deeper, act faster. https://qwenlm.github.io/blog/qwen3/, 2025. Accessed: 2025-09-22.

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

[63] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. arXiv preprint arXiv:2205.06175, 2022.

[64] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 25(27):79–80, 1995.

[65] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. arXiv preprint arXiv:1902.04043, 2019.

[66] Jürgen Schmidhuber. One big net for everything. arXiv preprint arXiv:1802.08864, 2018.

[67] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. arXiv preprint arXiv:2502.19417, 2025.

[68] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. arXiv preprint arXiv:2506.01844, 2025.

[69] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. nature, 550(7676):354–359, 2017.

[70] Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Gang Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. In NeurIPS 2024 Workshop on Open-World Agents, 2024.

[71] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805, 2023.

[72] Qwen Team. Qwen3-vl: Sharper vision, deeper thought, broader action. https://qwen.ai/blog?id=99f0335c4ad9ff6153e517418d48535ab6d8afef&from=research.latest-advancements-list, 2025. Accessed: 2025-10-24.

[73] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. nature, 575(7782):350–354, 2019.

[74] Volcengine. doubao-seed-1.6-vision. Volcengine, 2025. URL https://www.volcengine.com/docs/82379/1799865. Accessed: 2025-10-24.

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

[76] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. arXiv preprint arXiv:2406.01014, 2024.

[77] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191, 2024.

[78] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. arXiv preprint arXiv:2302.01560, 2023.

[79] David L Woods, John M Wyma, E William Yund, Timothy J Herron, and Bruce Reed. Factors influencing the latency of simple reaction time. Frontiers in human neuroscience, 9:131, 2015.

[80] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. OS-copilot: Towards generalist computer agents with self-improvement. arXiv preprint arXiv:2402.07456, 2024.

[81] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. arXiv preprint arXiv:2410.23218, 2024.

[82] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. Nature, 602(7896):223–228, 2022.

[83] xAI. Grok 4. https://x.ai/news/grok-4, July 2025. Accessed: 2025-10-24.

[84] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In International conference on machine learning, pages 38087–38099. PMLR, 2023.

[85] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. arXiv preprint arXiv:2309.17453, 2023.

[86] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. Advances in Neural Information Processing Systems, 37:52040–52094, 2025.

[87] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. arXiv preprint arXiv:2412.04454, 2024.

[88] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR), 2023.

[89] Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. UFO: A UI-focused agent for Windows OS interaction. arXiv preprint arXiv:2402.07939, 2024.

[90] J. Zhang. Gemini plays pokemon twitch stream. https://www.twitch.tv/gemini_plays_pokemon/about, 2025. Accessed: 2025-09-09.

[91] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. arXiv preprint arXiv:2304.13705, 2023.

[92] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In ICLR, 2024.

[93] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Conference on Robot Learning, pages 2165–2183. PMLR, 2023.

Appendix

A Gameplay Collection

In this section, we describe the recruitment and annotation process for collecting gameplay data. All annotators were native Chinese and played the Chinese version of the game, in which all in-game text and dialogues were displayed in Chinese. Consequently, all curated data, including instruction-following and reasoning annotations, were also produced in Chinese.

A.1 Participant Recruitment

To ensure the collection of high-quality gameplay data, we recruited annotators with proficient gameplay abilities and strong comprehension skills. The specific criteria were as follows:

1. **Game Experience:** Possess an in-game account with an Adventure Rank of 45 or higher in Genshin Impact; or be a frequent player of PC-based 3D games, such as MMORPGs, 3D RPGs, or other popular titles.
2. **Age:** Between 18 and 40 years old.
3. **Education:** An associate degree or higher.
4. **Hardware:** To ensure smooth gameplay and stable graphics rendering, participants' devices must meet or exceed the following specifications:
    - Operating System: Windows 10 or Windows 11
    - CPU: Intel Core i5-12400 or better
    - Memory: 16 GB or higher
    - GPU: NVIDIA GeForce GTX 1060 (6 GB) or better
    - Network: In-game latency below 100 ms

A.2 Annotation Tasks

Annotators are asked to start with a brand-new account and sequentially complete the following objectives:

- **Task 1:** Complete Act I of the Mondstadt main storyline. (≈1h)
- **Task 2:** Progress through the remianing main storyline (Act II & III) of the Mondstadt region. (≈15h)
- **Task 3:** Achieve over 80% map exploration in each region of Mondstadt. (≈14h)

On average, this process requires approximately 30 hours of gameplay for each annotator, and one annotator can complete the whole process up to five times. Furthermore, we imposed several constraints on the annotators' in-game behavior and operating habits to ensure consistency and data quality: (a) use only system-provided characters, including the Traveler, Amber, Kaeya, and Lisa; (b) do not modify the default game settings or key bindings; (c) remain strictly within the Mondstadt region and avoid entering other regions; (d) follow all in-game tutorial prompts without skipping them; (e) avoid rapid or repetitive camera movements and refrain from unnecessary or meaningless in-game actions; (f) do not remain idle for long periods or switch to other applications during gameplay; (g) do not check online guides or walkthroughs while playing, although reviewing them before starting the annotation session is allowed; and (h) do not spend Primogems through any means, including gacha pulls or converting them into resin.

Before commencing the formal recording tasks, all qualified annotators underwent standardized training and received detailed documentation describing the task objectives, procedures, and quality standards. During the annotation phase, continuous supervision and quality control were maintained. Annotators were required to submit daily progress reports through questionnaires. Regular manual inspections and automated tools were employed to assess data quality. Participants whose performance failed to meet the required standards were

dismissed, and their corresponding data were excluded from the final dataset. To further ensure practical competency, a secondary verification step was implemented. Candidates who failed to complete Task 1 within one hour, were disqualified from further participation.

Ultimately, 70 qualified participants were retained for the recording tasks, yielding a total of 2,424 hours of raw gameplay footage. The data collection took place from March to May 2025, covering three consecutive versions of Genshin Impact from 5.4 to 5.6.

A.3 Gameplay Recording

During annotation tasks, annotators' on-screen activities, along with their keyboard and mouse interactions, are recorded. Although this process appears straightforward, variations in participants' customized hardware and software settings present significant challenges for standardized data processing. For example, an identical mouse movement might appear scaled differently in the final gameplay output due to variations in players' sensitivity or configuration settings. Similarly, differences in screen aspect ratios may result in black bars or other display artifacts. To minimize the influence of such factors, we systematically examined potential sources of variability in game recording and developed an integrated game data recording software to standardize gameplay data across participants.

System Configuration Standardization. To ensure consistent gameplay resolution, all annotators are required to set their monitors to 1080p with a 100% scaling ratio and to play the game in Seamless Fullscreen mode instead of Windowed Mode. Before formal recording begins, the software automatically verifies whether these conditions are met. In addition, to avoid interruptions during recording (e.g., chat pop-ups), annotators must close all unnecessary applications in advance.

Video Capture. Gameplay footage is recorded using OBS[2], which is configured to automatically locate the Genshin Impact process and capture gameplay at *1080p* and 60 fps with a bitrate of 10,000 kbps. The recording is saved in `mkv` format to ensure that the recorded content remains unaffected in the event of an unexpected interruption.

Keyboard and Mouse Input Logging. Since participants are not allowed to modify in-game key bindings, no discrepancies in keyboard mappings are observed during recording. Mouse behavior, however, is considerably more complex. Previous studies typically record mouse input as either absolute positions or relative displacements, but neither method alone can fully capture real mouse movements. While absolute positioning works well in conventional GUI scenarios, it struggles to reflect 3D navigation, where mouse movement controls the in-game camera and implementation details vary across games. For example, in Red Dead Redemption II, although the cursor is invisible in the overworld, it can move freely across the screen according to player input. When the cursor reaches the edge of the screen (e.g., the right boundary) and the player continues turning right, the cursor position remains fixed, yet the turning action still occurs, an event that absolute-position recording would miss but relative-movement logging would capture. In contrast, Genshin Impact periodically re-centers the mouse cursor to the middle of the screen during overworld gameplay without the player's awareness. This system-induced motion is mistakenly recorded as user input in absolute coordinates, however, this automatic repositioning is not captured by relative movement. Thus, absolute positions alone are unreliable indicators of player intent, making it necessary to log relative-movement events.

Thus, absolute positions alone are unreliable indicators of player intent, making it necessary to log relative-movement events. However, relative measurements introduce their own challenges: factors such as display scaling ratios can proportionally distort movement values. To mitigate this, all recordings are standardized by unifying player resolution and display scaling ratios. An additional complication arises from Windows *Enhance Pointer Precision* feature, which applies nonlinear acceleration based on the magnitude of relative motion. This feature is enabled by default; although annotators can be instructed to disable it, compliance is difficult to verify and infeasible for existing public datasets. Empirically, we found that in Genshin Impact, this feature only affects GUI interactions and does not influence 3D overworld movement, making it complementary to absolute-position recording. Consequently, both absolute cursor positions and relative displacements are recorded simultaneously to ensure complete and reliable capture of mouse input.

[2]Open Broadcaster Software.

To achieve this, two complementary methods are implemented. For keyboard keypress events and absolute mouse coordinates, we employed the Win32 API `SetWindowsHook` to register a custom global hook function for capturing low-level input events. For relative mouse coordinates, we utilized the DirectX *DirectInput* API to actively poll the mouse state and relative displacement at a frequency of one query every 5 ms. Since timestamps from hook-captured input events rely on `GetTickCount()` and thus lack sufficient precision, we additionally retrieved high-resolution timestamps upon each event using the Win32 `GetSystemTimePreciseAsFileTime`.

A.4 Post-Process

After obtaining both gameplay videos and detailed keyboard/mouse input logs from annotators, the subsequent step is to process and standardize these data into a unified frame-action pair format. The core of this procedure involves two tasks: i) aligning keyboard and mouse events with corresponding video frames based on precise timestamps, and ii) reconciling the two types of mouse motion data, absolute cursor positions and relative displacements, into a consistent representation.

For time alignment, although both recordings are triggered simultaneously via the same hotkey, the video and keyboard recording processes each require some time to initialize. This can lead to a temporal misalignment of about 800 ms to 2 seconds (depending on the hardware), causing potential information leakage during the training. To address this, we parse the absolute timestamps of the first video frame from the OBS log and compare it with the start of input logging. If the video starts too early, we discard a few frames until its first frame is synchronized with the input events, or vice versa, thus eliminating time misalignment.

After aligning the timestamps of the video frames with those of the keyboard and mouse logging, we extract the visual frame and the corresponding user actions at a given time t. For the visual content, we simply retrieve the i-th frame corresponding to time t, leveraging process-level parallelism to accelerate video decoding.

For the keyboard, we reconstruct the state of each key at time t from a series of discrete events in the form of "key W pressed at time t_1." This reconstruction requires maintaining a full keyboard state and simulating the sequence of events in chronological order.

For the mouse, we aim to standardize the representation as relative movements. This involves deciding between computing the difference of absolute positions and summing the reported relative movement events. Our empirical observations indicate that, in the overworld environment of Genshin Impact, the mouse primarily controls the camera, and its movement magnitude depends solely on the relative motion events, while the absolute position may be modified by the game engine. Therefore, for non-GUI scenes, we sum all relative movement values between two consecutive frames to represent the mouse movement for that frame.

In contrast, in GUI scenes, the effective mouse movement is a function of the reported relative motion, influenced by factors such as scaling and the Windows *Enhance Pointer Precision* feature. In such cases, we compute the mouse movement as the difference in absolute positions between two consecutive frames. In practice, we employ template matching techniques to detect whether the current frame corresponds to a UI interface, dynamically selecting the appropriate method for mouse-action computation.

To reduce the complexity of model learning, we discretize mouse movement values using units of 5 pixels along the X-axis and 4 pixels along the Y-axis. Keys not listed in Table 6 are discarded. To decrease computational load during inference, each key press and the preceding space are represented as a single token, e.g., "ĠKey." Keys that require multiple tokens, such as F1–F12 and 0–9, are remapped accordingly.

To verify the accuracy of both the recording and data processing systems, we replay the processed data segments in the game and compare them against the originally recorded gameplay footage. We observe that in most cases, game events can be faithfully reproduced over short durations of a few seconds. However, over longer durations of more than ten seconds, or during sequences involving large camera rotations, a certain degree of drift emerges due to stochastic in-game mechanics, such as collision-avoidance adjustments in camera movement.

Table 6 Mapping between keyboard and mouse inputs and their corresponding token representations. Each key press and its preceding space are represented as a single token (e.g., "ĠKey") to reduce inference complexity. Keys requiring multiple tokens, such as F1–F12 and numeric keys 0–9, are remapped to single-token forms to reduce computational load during inference. LMB, RMB and MMB are for left mouse button, right mouse button and middle mouse button.

Key	Token	Key	Token	Key	Token	Key	Token	Key	Token	Key	Token
LMB	LB	7	seven	H	H	R	R	F2	Two	F12	Twelve
RMB	RB	8	eight	I	I	S	S	F3	Three	Esc	Esc
MMB	MB	9	nine	J	J	T	T	F4	Four	Tab	Tab
0	zero	A	A	K	K	U	U	F5	Five	Caps	Caps
1	one	B	B	L	L	V	V	F6	Six	Shift	Shift
2	two	C	C	M	M	W	W	F7	Seven	Ctrl	Ctrl
3	three	D	D	N	N	X	X	F8	Eight	Alt	Alt
4	four	E	E	O	O	Y	Y	F9	Nine	Space	Space
5	five	F	F	P	P	Z	Z	F10	Ten		
6	six	G	G	Q	Q	F1	One	F11	Eleven		

B Instruction Following Data Curation

The curation of instruction following dataset starts with human annotation. As shown in Figure 23, we first design a three-level hierarchical taxonomy to capture both broad and detailed aspects of gameplay. The first level, *Game Scene*, distinguishes among core gameplay environments: Overworld, Domain, and GUI Interface. The second level, *Game Content*, describes the player's activities within these scenes. For Overworld and Domain, the categories include Exploration, Visual Guidance Following, Collection, Combat, and Puzzle. For the GUI Interface, the categories cover Character Map, Quests, Inventory, Tutorials, and other interface-related activities. To better handle large variations within certain categories, we introduce a third level for fine-grained distinctions. For example, Puzzle-Solving is further divided into specific tasks such as *Following a Seelie*, *Activating Elemental Monuments*, and *Opening a vine-wrapped chest*. In total, this hierarchical design produce 38 distinct categories.

To ensure annotation accuracy and consistency, each annotator is provided with clear examples and definitions for every category. Annotators then segment each 20-second video clip exhaustively, identifying all relevant game content categories and marking the precise start and end timestamps of each segment. We also implement a parallel annotation and quality inspection workflow to maintain data integrity. The team is divided into two groups: one focus on labeling, while the other perform quality checks. The inspection group flags clips with incorrect classifications or misaligned timestamps. These clips are returned for re-annotation until all errors are resolved. Through this iterative process, we obtain 165 hours of high-quality, fine-grained gameplay classification data.

To facilitate further scaling and enable autonomous annotation, we then develop a specialized video classifier. The primary function of this classifier is to automatically categorize gameplay content by analyzing sequences of video frames. The model processes a fixed-length input of five consecutive frames and outputs a single category label describing the in-game activity.

The training dataset is constructed from 165 hours of 720p gameplay footage, sampled at 5 fps, with human annotators assigning a class to every frame. We first merge consecutive frames sharing the same label into variable-length video clips. From each segment, we uniformly sample five frames to form one training instance (segments with fewer than five frames are padded by repeating the final frame). To mitigate class imbalance, we employ a random down-sampling strategy, capping the number of data points for any single category at 1,000, and then split the balanced set 90%/10% into train/test. Each training sample is formatted as a multi-image prompt with five image slots and an associated ground-truth label.

We fine-tune the Qwen2-VL Base model, specifically experimenting with the 2B and 7B parameter variants. The training is conducted as a full supervised fine-tuning of the language model components, while keeping ViT and projector layers frozen to maintain visual feature extraction capabilities and improve efficiency. We

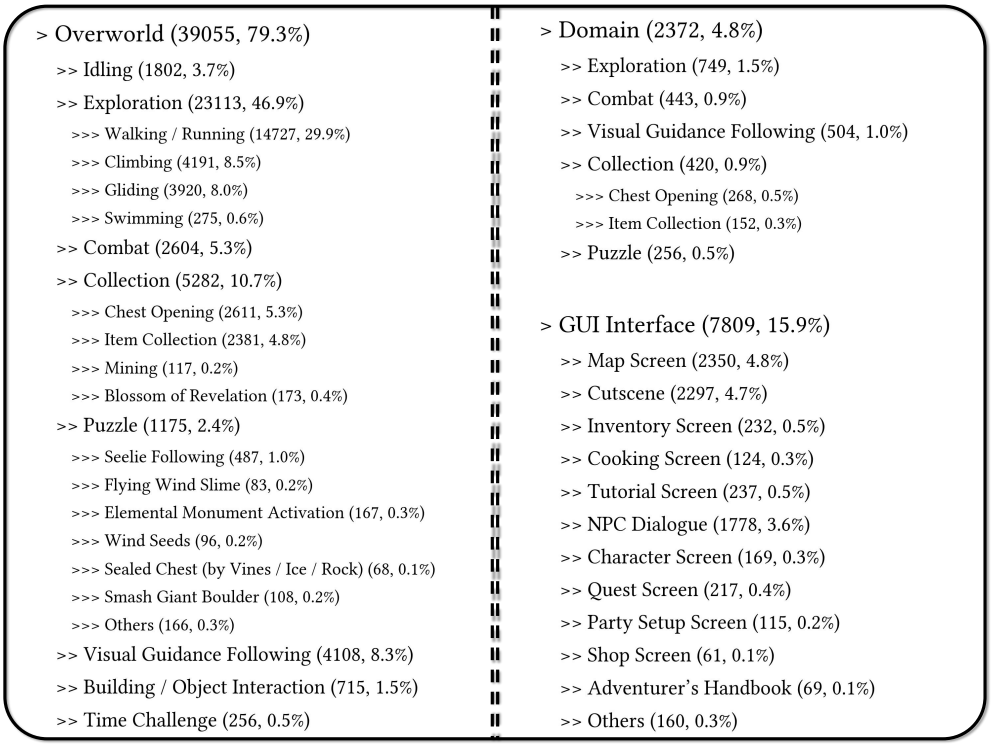


Figure 23 Distribution of gameplay categories in the Instruction Following dataset. A total of 39,055 annotated clips (≈165 hours) are organized into a three-level hierarchical taxonomy, consisting of *Game Scene*, *Game Content*, and fine-grained activity types. The numbers in parentheses after each category indicate the number of clips and their percentage proportion within the full dataset.

set the learning rate to 1e-5 with a warmup ratio of 0.1 over 500 training steps.

Evaluation on the test set reveals excellent performance for this classification task. As shown in Table 7, both the 2B and 7B models achieve a high average precision of over 80%. Crucially, the performance difference between the 2B and 7B models is marginal. Given the comparable accuracy, we opt to use the Qwen2-VL 2B model for our production inference pipeline to maximize processing throughput and efficiency.

Table 7 Accuracy comparison across different model sizes.

Accuracy	Overworld	UI Interface	Dungeon	Average
Model Size 2B	83.32%	87.87%	72.65%	83.04%
Model Size 7B	84.37%	88.09%	76.92%	84.32%

We then use this classifier to label all the raw gameplay data and identify transition points between adjacent gameplay segments that are assigned with different labels by the classifier, typically indicating a shift in task context or objective. Around each transition point, we extract a 20-frame (4s) snippet and prompt GPT-4.1 [52] to generate diverse, context-aware instructions based on the labeled categories. While the provided category labels supplement GPT-4.1's limited understanding of game mechanics and objectives, the model also acts as a verifier, detecting and discarding mislabeled samples when inconsistencies are found. After applying the same action filtering as in pre-training, we obtain 200 hours of high-quality instruction-following data.

C Reasoning Data Curation

To curate high-quality reasoning data, we recruited experienced Genshin Impact players and instructed them to annotate the first hour of gameplay. Annotators were asked to review the recorded footage, identify key decision-making frames, and describe the player's thought process from a first-person perspective.

Keyframes were defined according to the following criteria:

- Multiple options are presented in the current situation (e.g., at a fork in the road or within an inventory interface containing various items and buttons).
- A noticeable change occurs (e.g., mission completion, entering a dialogue interface, receiving new instructions, or encountering danger) that prompts the player to respond.
- The player's current reasoning is either complete or no longer applicable to the situation, necessitating a determination of the next goal.

To maintain consistent annotation standards and ensure quality control, we restructured the task around first-person keyframe cognition. Each keyframe was treated as a snapshot of the player's immediate thoughts, reconstructed as if the player were thinking in real time. For every keyframe, annotators need to follow a three-part structure to capture the flow of thought naturally and coherently:

- **Previous Step Summary**. A concise reflection on the preceding moment, capturing what just happened and why it leads to a new line of thought.
- **Current Situation Analysis**. An immediate interpretation of the current scene, focusing only on essential elements such as UI hints, mission description, mechanisms, NPCs, or enemies.
- **Next Move Planning**. The player's adjusted plan and next move, expressed naturally in their inner voice.

This design aimed to make each annotation feel like an authentic inner monologue, reflecting the player's short-term reasoning and decisions rather than external narration or mechanical labeling. During the actual annotation, a one-hour gameplay video was segmented into continuous **10-second clips**, with one image sampled every **200 ms** (50 frames per segment). Each segment included both visual data and raw action logs. To align reasoning consistency with human intuition and reduce noise, we established several annotation principles:

- Avoid mechanical or overly literal descriptions; the inner voice should sound natural and intention-oriented.
- Minimize redundant keyframes and omit unnecessary UI-related frames that do not contribute to reasoning.
- Avoid dense, high-frequency annotations within short intervals. A well-formed thought should provide guidance for the next 5–20 seconds of gameplay, with a minimum interval of more than one second between annotations.
- Pay attention to the timing of annotation. Each thought should precede the player's corresponding action, rather than being recorded after the key input occurs.
- Ignore random or insignificant player actions, such as switching characters while running—and focus only on events that meaningfully affect gameplay progression.
- During combat, do not annotate every skill release or character switch. Only mark thoughts related to significant combat mechanics; in most cases, battles can be simplified to "eliminate the enemies."
- Always refer to the main character as the Traveler, rather than the player's chosen name.

By following these rules, each annotation becomes a coherent chain of thoughts, where every keyframe captures not just what the player sees, but what they decide, allowing the reasoning data to connect smoothly with the instruct model and maintain a faithful representation of real-time player cognition.

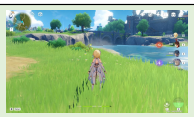
D Benchmark Introduction

In this section, we present additional examples from our task benchmark. The benchmark consists of 141 tasks grouped into four categories: *Collection*, *Combat*, *NPC Interaction*, and *Puzzle*. Owing to the large number of tasks, we showcase only a representative subset, categorized by difficulty and visually distinguished using background colors:

- **Simple Tasks**: Highlighted with a **light green** background.
- **Unseen Tasks**: Highlighted with a **light blue** background.
- **Hard Tasks**: Highlighted with a **light red** background.

D.1 Collection

The *Collection* category includes a total of 62 tasks: 21 simple, 31 unseen, and 10 hard. Tasks are presented with their starting point and corresponding instruction.

	靠近并拾取前方的红色日落果 Approach and collect the red Sunset Berry ahead		靠近并采摘左前方的黄色树莓 Approach and collect the yellow Raspberry in front and to the left
	采集嘟嘟莲 Collect Calla Lily		采集小灯草 Collect Small Lamp Grass
	击打铁矿石，拾取掉落的铁矿石 Hit the Iron Chunk and collect the dropped Iron Chunk		切换角色为凯亚，对前方的树木使用普通攻击以获得掉落的木材 Switch character to Kaeya and use a Normal Attack on the tree ahead to obtain the dropped wood
	采集霓裳花 Collect Silk Flower		采摘前方的紫色植物鸣草 Collect the Naku Weed ahead.
	拾取前方山顶的白色植物清心 Collect the white plant, the Qingxin, on the mountaintop ahead		靠近并拾取石柱上方的紫色雷神瞳 Approach and collect the purple Electroculus above the stone pillar
	采集甜甜花 Collect the Sweet Flower		采集薄荷 Collect Mint
	靠近并拾取前方的蓝色风神瞳 Approach and collect the blue Anemoculus ahead		收集前方飘在空中的风神瞳 Collect the Anemoculus floating in the air ahead
	爬上右侧的石柱，到达最高处后收集位于左侧空中的蓝色风神瞳 Climb the stone pillar on the right and, once you reach the top, collect the blue Anemoculus floating in the air on the left		切换角色为凯亚，不断释放 E 技能冻结水面，以收集前方浮在水面上的风神瞳 Switch to Kaeya, continuously use his Elemental Skill (E Skill) to freeze the water surface, and collect the Anemoculus floating ahead

D.2 Combat

The *Combat* category includes a total of 21 tasks: 8 simple, 7 unseen, and 6 hard. All tasks share the same instruction: *Defeat the monsters ahead and open the chest.*

消灭前方怪物并打开宝箱

Defeat the monsters ahead and open the chest

D.3 NPC Interaction

The *NPC Interaction* category includes a total of 21 tasks: 7 simple, 8 unseen, and 6 hard.

 与 NPC 喜儿对话 Talk to NPC Xi'er	 与 NPC 石榴对话 Talk to NPC Shiliu
 与 NPC 布兰琪对话 Talk to NPC Blanche	 与 NPC 芙萝拉对话 Talk to NPC Flora
 与 NPC 瓦格纳对话 Talk to NPC Wagner	 和前方绿色衣服 NPC 对话 Talk to the NPC in green clothes in front of you
 与 NPC 快刀陈对话 Talk to NPC Chen the Sharp	 与 NPC 东升对话 Talk to NPC Dongsheng
 与 NPC 凯瑟琳对话 Talk to NPC Katheryne	 与右方的 NPC 小仓澪对话 Talk to the NPC Ogura Mio on the right
 与 NPC 柚子对话 Talk to NPC Yuzu	 与 NPC 汤雯对话 Talk to NPC Tang Wen
 与 NPC 蒂玛乌斯对话 Talk to NPC Timaeus	 与 NPC 舒茨对话 Talk to NPC Schulz
 与 NPC 葛瑞丝对话 Talk to NPC Grace	 与 NPC 萨义德对话 Talk to NPC Sayid
 与 NPC 门罗对话 Talk to NPC Monroe	 与 NPC 苏西对话 Talk to NPC Susie

D.4 Puzzle

The *Puzzle* category includes a total of 23 tasks: 9 simple and 14 hard.

	打开前方被荆棘包裹的宝箱 Open the chest wrapped in thorns ahead.		击碎前方的巨石，开启宝箱 Break the boulder ahead and open the chest.
	使用对应的元素激活元素方碑 Activate the Elemental Monument using the corresponding element.		击碎前方的巨石，开启宝箱 Break the boulder ahead and open the chest.
	激活元素方碑 Activate the Elemental Monument.		点燃火把打开宝箱 Light the torches to open the chest.
	使用风元素激活风车花，沿着激活的风场飞行，开启宝箱 Use Anemo to activate the Pinwheel, fly along the activated Wind Current, and open the chest.		完成前方的限时挑战，消灭所有飞在空中的风史莱姆 Complete the Time Trial Challenge ahead: Defeat all airborne Anemo Slimes.
	收集风种子以激发风场，进入风障内，开启宝箱 Collect the three Wind Anemograna to activate a Wind Current, then enter the Wind Barrier to open the chest.		完成前方的限时挑战，在限时时间内引爆 4 个炸药桶 Complete the Time Trial Challenge ahead: Detonate 4 Explosive Barrels within the time limit.
	完成前方的限时挑战，在限时时间内开启宝箱 Complete the Time Trial Challenge ahead: Open the chest within the time limit.		完成前方的限时挑战，在限时时间内消灭所有敌人 Complete the Time Trial Challenge ahead: Defeat all opponents within the time limit.
	使用对应的元素激活元素方碑 Activate the Elemental Monument using the corresponding element.		击破漂浮的风史莱姆后，开启宝箱 After defeating the floating Anemo Slime, open the chest.

E Prompt

In this section, we present the system prompts used by Lumine for instruction-following and reasoning tasks. The original prompts are written in Chinese; we also provide their English translations for reference.

Chinese System Prompt for Instruction Following

```
你是一名经验丰富的原神PC端玩家，精通键盘鼠标操作。请根据当前游戏画面，规划接下来 200ms 的操作，由 6 步组成，每步间隔 33ms。每步动作在执行时刻起持续 33ms，直至下一步开始。

**输出格式**
<|action_start|>X Y Z ; k1 k2 k3 ; k4 k5 ; k6 ; k7 ; k8 ; k9 k10<|action_end|>

**说明**
1. **鼠标位移**：首先给出相对位移X,Y（X>0 右移，Y>0 下移）以及滚轮量Z（Z>0 上滚）。
2. **按键序列**：随后列出 6 组按键；同组内用空格分隔，不同组用分号分隔。
- 每组最多 4 个按键。
- 若某组无按键，留空但保留 ';'。
3. 只输出符合上述格式的纯字符串，不换行、不加引号。

**按键命名规范**
- 数字键 '1-9'：使用小写英文，如 'one' 表示键盘上的 '1'。
- 功能键 'F1-F12'：使用首字母大写英文单词，如 'One' 表示 'F1'，'Two' 表示 'F2'，依此类推。
- 其他按键（如字母、Shift、Tab、Space 等）：统一使用首字母大写的真实键盘名称，如 'A'、'D'、'Shift'、'Space' 等。

你当前要完成的任务是：<instruction>
```

English System Prompt for Instruction Following

```
You are an experienced Genshin Impact PC player, proficient in keyboard and mouse operations.
Based on the current game screen, 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 ; k8 ; k9 k10<|action_end|>

**Explanation**
1. **Mouse Movement**: First, specify the relative displacement X, Y (X>0 means move right, Y>0
means move down) and scroll amount Z (Z>0 means scroll up).
2. **Key Sequence**: Then list 6 groups of keys; within each group, keys are separated by spaces,
and groups are separated by semicolons.
   - Each group can contain up to 4 keys.
   - If a group has no keys, leave it empty but keep the ';'.
3. Only output a plain string that conforms to the above format - no line breaks and no quotation
marks.

**Key Naming Rules**
- Number keys '1-9': use lowercase English words, e.g., 'one' represents the '1' key on the
keyboard.
- Function keys 'F1-F12': use capitalized English words, e.g., 'One'
represents 'F1', 'Two' represents 'F2', and so on.
- Other keys (letters, Shift, Tab, Space, etc.): use the real keyboard name with an initial
capital letter, e.g., 'A', 'D', 'Shift', 'Space', etc.

Your current task is: <instruction>
```

Chinese System Prompt for Reasoning

你是一名资深原神PC玩家，熟练使用键盘和鼠标进行高水平操作。你熟知游戏机制与战斗节奏，能够从实时画面中迅速提取关键信息，在关键时刻进行思考并做出精准决策。请根据当前画面，规划接下来 200ms 的操作，由 6 步组成，每步间隔 33ms。每步动作在执行时刻起持续 33ms，直至下一步开始。若当前情境延续前一分析策略，可直接输出动作；仅当局势发生明显变化、先前分析失效或出现新目标时，需进行必要的思考并输出思考内容。

输出格式
- **仅动作（常用）**
  <|action_start|>X Y Z ; k1 k2 k3 ; k4 k5 ; k6 ; k7 ; k8 ; k9 k10<|action_end|>
- **思考 + 动作（必要时）**
  <|thought_start|>思考内容<|thought_end|><|action_start|>X Y Z ; k1 k2 k3 ; k4 k5 ; k6 ; k7 ; k8 ; k9 k10<|action_end|>

说明
1. **鼠标位移**：首先给出相对位移X,Y（X>0 右移，Y>0 下移）以及滚轮量Z（Z>0 上滚）。
2. **按键序列**：随后列出 6 组按键；同组内用空格分隔，不同组用分号分隔。
   - 每组最多 4 个按键。
   - 若某组无按键，留空但保留 ';'。
3. 只输出符合上述格式的纯字符串，不换行、不加引号。

按键命名规范
- 数字键 '1-9'：使用小写英文，如 'one' 表示键盘上的 '1'。
- 功能键 'F1-F12'：使用首字母大写英文单词，如 'One' 表示 'F1'，'Two' 表示 'F2'，依此类推。
- 其他按键（如字母、Shift、Tab、Space 等）：统一使用首字母大写的真实键盘名称，如 'A'、'D'、'Shift'、'Space' 等。

当前目标
<cur_thought>

English System Prompt for Reasoning

```
You are a veteran Genshin Impact PC player, highly skilled in using the keyboard and mouse for
advanced gameplay. You are familiar with the game's mechanics and combat rhythm, capable of
quickly extracting key information from real-time visuals, thinking critically in crucial moments,
and making precise decisions. Based on the current game screen, 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. If the current situation continues from the previous analytical
strategy, directly output the actions; only when the situation changes significantly, previous
analysis becomes invalid, or new targets appear, you should provide necessary reasoning and output
your thought process.

**Output Format**
- **Action Only (commonly used)**
  <|action_start|>X Y Z ; k1 k2 k3 ; k4 k5 ; k6 ; k7 ; k8 ; k9 k10<|action_end|>
- **Thought + Action (when necessary)**
  <|thought_start|>Thought content<|thought_end|><|action_start|>X Y Z ; k1 k2 k3 ; k4 k5 ; k6 ;
  k7 ; k8 ; k9 k10<|action_end|>

**Explanation**
1. **Mouse Movement**: First, specify the relative displacement X, Y (X>0 means move right, Y>0
means move down) and scroll amount Z (Z>0 means scroll up).
2. **Key Sequence**: Then list 6 groups of keys; within each group, keys are separated by spaces,
and groups are separated by semicolons.
   - Each group can contain up to 4 keys.
   - If a group has no keys, leave it empty but keep the ';'.
3. Only output a plain string conforming to the above format - no line breaks, no quotation marks.

**Key Naming Rules**
- Number keys '1-9': use lowercase English words, e.g., 'one' represents the '1' key on the
keyboard.
- Function keys 'F1-F12': use capitalized English words, e.g., 'One' represents 'F1', 'Two'
represents 'F2', and so on.
- Other keys (letters, Shift, Tab, Space, etc.): use the real keyboard name with an initial
capital letter, e.g., 'A', 'D', 'Shift', 'Space', etc.

**Current Objective**
<cur_thought>
```