Can VLMs Play Action Role-Playing Games? Take Black Myth Wukong as a Study Case

Peng Chen[1*] **Pi Bu**[1*] **Jun Song**[1†] **Yuan Gao**[1] **Bo Zheng**[1]

[1]Alibaba Group

Abstract

Recently, large language model (LLM)-based agents have made significant advances across various fields. One of the most popular research areas involves applying these agents to video games. Traditionally, these methods have relied on game APIs to access in-game environmental and action data. However, this approach is limited by the availability of APIs and does not reflect how humans play games. With the advent of vision language models (VLMs), agents now have enhanced visual understanding capabilities, enabling them to interact with games using only visual inputs. Despite these advances, current approaches still face challenges in action-oriented tasks, particularly in action role-playing games (ARPGs), where reinforcement learning methods are prevalent but suffer from poor generalization and require extensive training. To address these limitations, we select an ARPG, "*Black Myth: Wukong*", as a research platform to explore the capability boundaries of existing VLMs in scenarios requiring visual-only input and complex action output. We define 12 tasks within the game, with 75% focusing on combat, and incorporate several state-of-the-art VLMs into this benchmark. Additionally, we will release a human operation dataset containing recorded gameplay videos and operation logs, including mouse and keyboard actions. Moreover, we propose a novel **VARP** (**V**ision **A**ction **R**ole-**P**laying) agent framework, consisting of an action planning system and a human-guided trajectory system. Our framework demonstrates the ability to perform basic tasks and succeed in 90% of easy and medium-level combat scenarios. This research aims to provide new insights and directions for applying multimodal agents in complex action game environments. The code and datasets will be made available at `https://varp-agent.github.io/`.

1 Introduction

In recent years, LLM-based agents have achieved significant breakthroughs across various fields [5, 9, 10, 25, 34], particularly with the integration of tools and memory modules[35], as seen in AutoGPT and Reflection [30, 20]. Among these, applying LLM-based agents in video games has become one of the most popular areas of research.[18, 16, 13, 27, 26] These methods input information from video games into LLMs, which then undergo complex reasoning and integration through agent frameworks, ultimately producing keyboard and mouse commands that can directly interact with the game to complete tasks. Previous works have mostly focused on accessing video game APIs to read in-game environmental and action information. For instance, the framework proposed by Wang et al.[24] has been successfully applied in the game Minecraft. Agents can achieve autonomous mining, building, and attacking enemies in the game. However, this approach does not align with how humans play games, and most games do not offer open APIs, which limits the

[*]Equal Contribution.

[†]Corresponding Author.

Table 1: Comparison of several existing agents. Among them, "API" refers to the model's use of video game APIs to access in-game environmental and action information, whereas "Screen" indicates that visual understanding is derived solely from game screenshots.

Agents	**Agent Type**	**Game**	**Game Type**	**Environment**
Reflexion [20]	LLM-based	ALFWorld	Text-based Adventure	API
ReAct [31]	LLM-based	ALFWorld	Text-based Adventure	API
Voyager [24]	LLM-based	Minecraft	Sandbox	API
CreativeAgent [33]	VLM-based	Minecraft	Sandbox	API and Screen
Cradle [22]	VLM-based	RDR2	AAA Action Adventure	Screen
DQN [1]	RL-based	Sekiro	AAA Action Role-Playing	Screen
Other Project [23, 7]	RL-based	BMW	AAA Action Role-Playing	Screen
VARP (Ours)	VLM-based	BMW	AAA Action Role-Playing	Screen

widespread application of this method. Recently, the emergence of vision language models (VLMs) like GPT-4o has further enhanced the visual understanding capabilities of these agents, showcasing broader potential in mobile apps and games. For example, the Cradle framework [22] has been implemented in Red Dead Redemption 2 (RDR2). It directly uses game screenshots from RDR2 as input, rather than using an API to read game memory information. However, Cradle relies heavily on text-based guiding information in the game screenshots to create new skills. For tasks or games with weak textual guidance, such as some action role-playing games(ARPG), Cradle is unable to leverage the effective performance of VLMs. For ARPGs, many researchers employ reinforcement learning methods, where penalties and rewards are predefined for specific tasks. After extensive training periods and numerous iterations, the trained agents can complete the given specific tasks. However, RL-based agents can only accomplish tasks within the environment they were trained in and find it challenging to transfer to other tasks. ARPGs contain a large number of specialized tasks, which pose a significant challenge for RL-based agents with poor generalization capabilities. We conducted a comparison of some representative methods in Tab. 1.

Thus, most of the existing research focuses on relatively simplified settings. This simplification arises primarily from two significant challenges: 1) **Immediate visual input.** Since environmental data is not always accessible through game APIs, learning from visual input becomes a more straightforward strategy, especially in AAA games (characterized by A lot of time, A lot of resources, A lot of money), where understanding the immediate visual input is crucial. 2) **Action-oriented tasks.** Action games are immensely popular among players; however, in this domain, RL-based agents still dominate, which require extensive training time and have poor generalization ability. For VLM-based agents, the game interfaces of ARPGs provide very few textual hints, and most of the actions need to be learned through experience and self-innovation. As a result, previous agents have found it challenging to extract effective guidance information from visual inputs.

In this paper, we will select the "*Black Myth: Wukong*," abbreviated as BMW, an AAA ARPG, as our research platform for extensive experimentation. We are dedicated to establishing a VLM-based agent framework to thoroughly investigate the capability boundaries of existing models (e.g. GPT-4o, Gemini) in scenarios requiring visual-only input and complex action output. Among them, visual-only input refers to the model making decisions solely by understanding and analyzing the game screenshot, while complex action output necessitates the model to perform intricate and continuous actions, such as precise operations in combat scenarios.

To achieve this goal, we define 12 tasks in the game "*Black Myth: Wukong*," with 75% of these tasks being combat-related. Several state-of-the-art VLM models, including GPT-4o, will be incorporated into this benckmark to comprehensively explore their performance boundaries. Subsequently, to advance the development of VLM-based agents in AAA action games, we will open-source a human operation dataset, which includes records of mouse and keyboard commands as well as gameplay recordings. Lastly, we innovatively propose a VARP(Vision Action Role-Playing) agent framework, consisting of an action planning system and a human-guided trajectory system. Specifically, the action planning system is responsible for generating action combos that are suitable for combat scenarios, while the human-guided trajectory system learns from human data via retrieval. Through extensive evaluations, our proposed framework demonstrates the capability to accomplish basic tasks such as picking up items and opening treasure chests, while also succeeding in 90% of esay and medium battles. We hope this research will provide new insights and directions for the application of multi-modal agents in complex action game environments. The main contributions of this paper are summarized as follows:

- **Benchmark**. We define 12 tasks based on the game "*Black Myth: Wukong*," with 75% of these tasks focused on combat. Several state-of-the-art VLM models, including GPT-4o, will be incorporated into this benckmark to thoroughly explore their capability boundaries.
- **Dataset**. We release a dataset containing recorded gameplay videos along with relevant operation logs, including mouse movements, clicks, and keyboard actions, which includes 1000 records.
- **Framework**. We propose a VARP agent framework, which comprises an action planning system and a human-guided trajectory system. Through these systems, the agent can execute complex action combos and learn from human operation.

2 Methodology

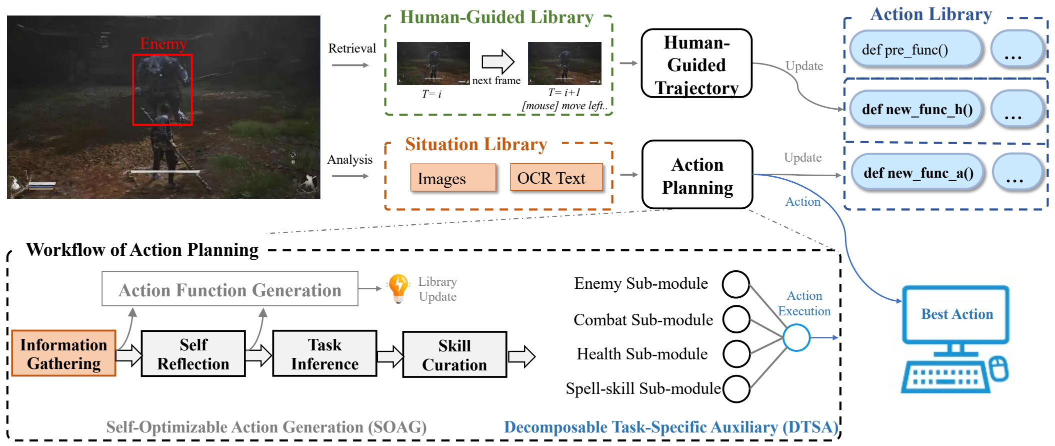


Figure 1: **Pipeline of VARP.** We propose a novel framework named the VARP agent, which directly takes game screenshots as input and generates keyboard and mouse operations to play the ARPG.

2.1 Overview

We propose a novel framework named the VARP agent, which directly takes game screenshots as input. Through inference by a group of Vision-Language Models (VLMs), it ultimately generates actions in the form of Python code, which can directly operate the game character. Each action is a sequence that consists of various combinations of atomic commands. These atomic commands include light attack, dodge, heavy attack, restore health, and others. Meanwhile, the VARP agent maintains three libraries: a situation library, an action library, and a human-guided library. These libraries can be retrieved and updated to store intensive knowledge for self-learning and human guidance. Overall, the VARP agent is divided into two systems: the action planning system and the human-guided trajectory system, as shown in Fig. 1. In the action library, "def new_func_a()" represents the new action generated by the action planning system, while "def new_func_h()" represents the new action generated by the human-guided trajectory system. "def pre_func()" represents the predefined actions. The following sections will elaborate on each system in detail.

2.2 Action Planning System

The action planning system is primarily used for action reasoning and generation. This system utilizes a situation library and an updatable action library as knowledge retrieval bases. Guided by input game screenshots, the system employs a group of VLMs to select or generate actions appropriate for the current situation. The generated situations and actions are stored or updated in the two libraries. Additionally, We propose decomposable task-specific auxiliary modules to break down large tasks into smaller subtasks, which are then distributed across multiple VLMs to reduce the occurrence of model forgetting and hallucinations. We also introduce a self-optimizable action generation module to encourage VLMs to generate new actions specific to some hard tasks, thereby completing complex tasks more efficiently and with higher quality.

2.2.1 Basic VLMs Group

Inspired by Cradle [22], our main pipeline continues to adopt the five basic modules from Cradle, with some of these basic modules calling the VLM for reasoning, forming a basic group of VLMs. During initialization, we manually predefined some actions and placed them into the action library as the prior knowledge. Each action is a Python function with detailed textual annotations, and we computed the embeddings of these annotations for storage. **Information Gathering** is responsible for gathering information from sampled game screenshots, including textual and visual information related to situations and actions. The textual information primarily includes text guides, text labels, and notifications; the visual information mainly covers environmental positions, character actions, and interface icons. The former is assisted by OCR tools for text recognition, while the latter uses the object detection tools for visual localization. **Self Reflection** takes a few game screenshots from the last video in the situation library as input to assess whether the last executed action successfully produced the correct effect and whether the current task has been completed. If execution fails, the module needs to provide a reason for the failure to guide the next step in action generation. **Task Inference** infers the current task to be executed based on the results of previous modules, and generates the task description. **Skill Curation** calculates the similarity between the task description's embedding and the embeddings of the textual annotations in the action library to find some matching actions, which form the candidate action set. **Decision Making** utilizes the Chain of Thought (CoT) [28] approach to reason through and deeply analyze multiple sequential questions (such as whether to enable combat mode, restore health, or select from available spell skills, etc.). Finally, the module infers the most suitable action from the candidate action set, executes the Python code, and operates the keyboard and mouse to control the player character to complete the corresponding task. These five basic modules will record each intermediate product into the situation library.

2.2.2 Self-Optimizable Action Generation Module

The basic VLMs group can only acquire actions from a predefined action library or from game screenshots with clear textual prompts. For certain tasks in ARPGs that have weak textual guidance, such as real-time combat, this method is unable to learn new actions. Therefore, we propose a self-optimizable action generation module (SOAG) that allow the VARP agent to summarize the enemy's actions during combat, thereby optimizing existing actions and generating new ones to counter enemy attacks. The new actions are combinations of the two atomic commands: dodging and light strikes. The optimization goal is to maximize the evasion of enemy attacks and the ability to strike the enemy while minimizing the player character's health loss.

Specifically, in SOAG, we introduce a component responsible for action function generation. This component takes the information gathering and self reflection results, along with the current and last game screenshots, as input. It analyzes the characteristics of the enemy under the current task, such as name, appearance, weapon, etc. Most importantly, it needs to analyze the enemy's current and previous actions. For example, for the hard enemy named Bullguard, its attack actions can be roughly categorized as: "charging forward with the axe", and "chopping the axe downwards three times consecutively", etc. Therefore, this component needs to inference new actions for dodging and counterattacking based on the current enemy actions. For instance, for "charging forward with the axe," the new action should be to dodge once and then attack continuously; for "chopping the axe downwards three times consecutively," the new action should be to dodge three more times before counterattacking. The generated new actions are permutations of the atomic operations "dodge" and "light attack." The generated actions are stored in the action library with detailed textual annotations.

2.2.3 Decomposable Task-Specific Auxiliary Modules

In ARPGs, especially in BMW game, the VLM's inference involves a large number of tokens, including multiple images and long texts. The attention mechanism used by VLMs allocates attention to all tokens in long texts. As the input length increases, the attention distribution becomes increasingly diluted. In the basic VLMs group, due to the excessive number of input tokens for each module, the model may fail to effectively focus on key information, leading to errors such as forgetting and hallucination. This issue is particularly evident in the decision-making module, where the VLM frequently makes mistakes when answering multiple questions.

To address this problem, we decomposed the basic modules and added multiple parallel auxiliary sub-modules for specific tasks, which are then integrated by the VLM. The structure is similar to an

Table 2: Task definitions in *Black Myth: Wukong* (BMW), where "*" indicates the challenging task.

Task ID	Task Name	Description	Diffuculty
1	Guidance	Defeat Erlang, the Sacred Divinty	Easy
2	Combat 1	Defeat WolfScout	Easy
3	Gather	Gather	Easy
4	Combat 2	Defeat WolfStalwart	Easy
5	Combat 3	Defeat WolfSwornsword	Easy
6	Open	Open	Easy
7	Combat 4	Defeat WolfSoldier	Easy
8	Combat 5	Defeat Croaky	Easy
9	Combat 6	Defeat Crow Diviner	Middle
10	Combat 7	Defeat Bullguard	Hard
*11	Combat 8	Defeat Wandering Wight	Very Hard
*12	Move	Autonomous Navigation	Very Hard

MLP. Specifically, as shown in the workflow of action planning in Fig. 1, we decomposed the original decision-making module that handled multiple tasks into 5 sub-modules. **1) Enemy Sub-module** is used to analyze the enemy's status (such as its health, position, etc.) and action description, which assists the agent in obtaining detailed information of the enemy. **2) Combat Sub-module** determines which combat method to use, including light attack or heavy attack, by observing the heavy-attack status in the bottom right corner of the game screen. **3) Health Sub-module** is responsible for constantly monitoring the player's health bar. If the health is consumed excessively, it assists the agent by prioritizing the action of recovering health. **4) Spell-skill Sub-module** monitors the status of the player's spell skills while simultaneously analyzing the situation in the combat state to determine the appropriate time to use available spell skills. **5) The integration sub-module** is responsible for integrating the outputs of all sub-modules and reasoning to determine the best action from the candidate action set for the current specific task. The decomposable task-specific auxiliary modules decompose long tokens and focus on each individual question, significantly improving the accuracy of the decision-making module.

2.3 Human-Guided Trajectory System

Human actions are seen as valuable data, implicitly rich in knowledge of the physics and game world, which can lead to advanced action combinations for very complex tasks, such as way-finding tasks and high-difficulty combat tasks. To learn the human experience from this implicit data, we first collected a human dataset and then used it to improve the performance of our VARP agent. The collection process of human operation data and dataset analysis will be detailed in Sec. 3.1, which consists of mouse keyboard logs, and recording game screenshots. In this section, we focus on how to use it to implement a human-guided trajectory system. In this section, we refer to our annotated dataset as the human-guided library. It is a collection of pairs consisting of game screenshots and human operations, with each pair having a unique timestamp.

For very hard tasks in the game, we first take a screenshot of the current game interface. Based on this game screenshot, we query the human-guided library for the screenshot with the highest similarity. We then input this screenshot along with the subsequent n-frame screenshots and their corresponding operations into the human-guided trajectory system. This system will utilize a VLM to analyze and summarize the input images and operations, ultimately outputting a new human-guided action, which is then stored in the action library for the action planning system to choose and execute.

3 Experiments

3.1 Dataset Collection

We collected a human operation dataset that includes mouse and keyboard logs, as well as recordings of game screenshots. Specifically, we recruited 200 volunteers to play the BMW game and record their operations, with approximately 70% of them experiencing this game for the first time. To ensure the dataset's quality, we eliminated invalid data from volunteers who did not complete the tasks. Over the course of two weeks, we ultimately gathered a total of 1,000 valid data entries. Specifically, over 90% of Task 11 (i.e. Defeat Wandering Wight) was discarded, indicating that it is challenging for players to defeat the boss in a single attempt. Moreover, we observed that

(a) Task 1

(b) Task 2

(c) Task 3

(d) Task 4

(e) Task 5

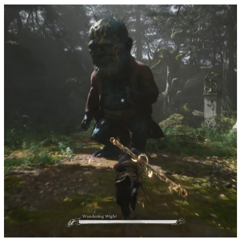
(f) Task 6

(g) Task 7

(h) Task 8

(i) Task 9

(j) Task 10


(k) Task 11

(l) Task 12

Figure 2: Image examples of defined tasks.

volunteers exhibited redundant actions during the annotation process, such as excessive mouse clicks and scrolling. Therefore, some volunteers will be asked to play the game again to identify the optimal actions, and this refined data will be labeled as "clean" in our released dataset. Please refer to the supplementary material for more details about our dataset.

3.2 Benchmark and Task Definition

To investigate the capabilities of existing VLMs in playing action games, we define 10 basic tasks and 2 challenging tasks aligned with the game's narrative, with 75% of these tasks occurring in combat scenes. As illustrated in Tab. 2 and Fig. 2, all tasks are concentrated in the first chapter of the game, due to the limited understanding and reasoning abilities of VLMs. In terms of benchmarking, we allow the agent to test each task 5 times and calculate the success rate for each task. For combat tasks, a task is deemed successful if the player's character defeats the enemy, while a task is considered a failure if the player's character is defeated and killed by the enemy. We have manually assessed the difficulty of 12 tasks, categorizing them as easy, medium, hard, and very hard. Due to the absence of maps and guidance, and the presence of numerous "invisible walls" in the BMW game, we classify task 12, autonomous navigation (i.e., moving from the spawn point to the Bullguard's location within five minutes), as a very hard task. This is a challenging task even for human novices. We utilize the success rates from this benchmark to evaluate the performance of the VARP agent and various VLMs.

3.3 Implementation Details

All evaluations are performed on a machine equipped with an NVIDIA RTX 4090 GPU running the Windows operating system. We use three of the most popular VLMs to drive our agent: GPT-4o-2024-05-13[15], Claude 3.5 Sonnet[2], and Gemini 1.5 pro[8]. We also utilize OpenAI's text-embedding-ada-002[21] model to generate embeddings for each action. The size of game interface for the BMW game is set to 1920×1080. During the inference of VLMs, we pause the game using the photo mode. We employ Grounding DINO[14] for object detection of people and objects in game screenshots to assist the VLMs in better extracting useful information.

3.4 Performance Evaluation

To evaluate the performance of the VARP agent without human guidance, we conducted experiments on our proposed benchmark while disabling the human-guided trajectory system of the VARP agent. In this performance evaluation, we only tested the benchmark and compared the VARP agent with human novice players.

We calculated the success rates of the VARP agent driven by different VLMs and human novice players when completing each task. As shown in Fig. 3, both the VARP agent and human novice players achieved high success rates on tasks 1 to 8, reaching nearly 100% on most tasks. In task 9, the VARP agent's average success rate was 40%, which also confirms its "middle" difficulty. The enemy in task 10 is the first boss-level monster that the player encounters in the game. For human novice players, the success rate for this task was 15.63%, while the VARP agent's average success rate was 20%. Task 11 is classified as "very hard," so the success rates for both human novices and the VARP agent were very low. Specifically, the VARP agent is limited by the reasoning speed of VLMs, making it unable to input every game frame in real-time and only able to input keyframes at second-level intervals. In ARPGs, this can easily result in missing critical information about enemy attacks. Therefore, task 11 is particularly challenging for the agent. In terms of autonomous navigation, humans can easily find the final boss enemy of the level within five minutes, but for VLMs, this is an almost impossible task. Without human guidance, the success rate is 0%. Since the game provides no guidance or hints for navigation tasks and contains many "invisible walls," VLMs lack the ability to perceive the correct path in the 3D scene without human assistance.

In summary, the VARP agent's performance on tasks 1 through 11 is already close to that of novice human players. However, in terms of 3D scene perception and prior knowledge, the VARP agent is still far inferior to humans.

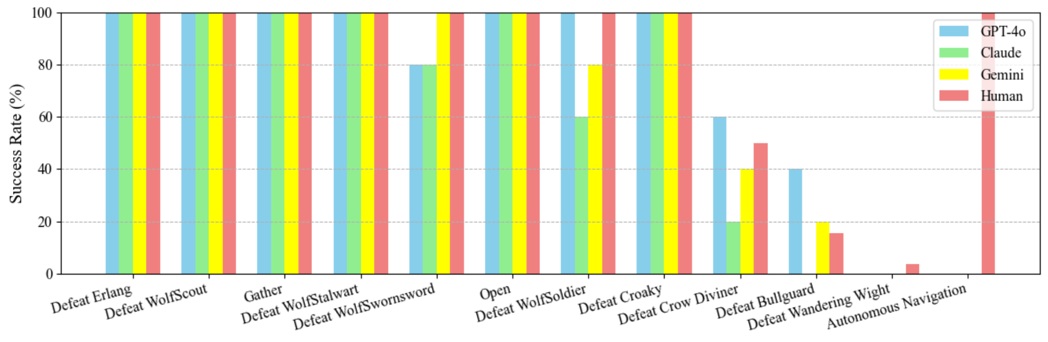


Figure 3: Evaluation Results on various VLMs and human.

3.5 Ablation Study

To evaluate the effectiveness of the self-optimizable action generation module(SOAG) and the decomposable task-specific auxiliary module(DTSA) in the action planning system, we conducted experiments with each of these modules removed separately, calculating the success rate on the benchmark. The VLM used in this part of the experiment is GPT-4o-2024-05-13. As shown in Fig. 4, without SOAG, the agent's performance significantly declines in the middle and hard tasks. This is because the enemies in these tasks have high health points, resulting in prolonged battles. The function of SOAG is to continuously learn the enemies' attack patterns, aiding players in dodging and counterattacking. Therefore, in long-duration tasks like middle and hard tasks, the effectiveness of SOAG becomes more apparent. On the other hand, DTSA aims to decompose large tasks into smaller ones, focusing more on precision. This approach helps prevent global errors caused by local issues such as the forgetting and hallucination of the VLM. Hence, without DTSA, the agent tends to fail in some easy tasks.

3.6 Case Studies

For the VARP agent, the newly generated actions originate from two sources: one is the human-guided trajectory system, and the other is the SOAG in the action planning system. Next, we will examine some cases of the newly generated actions.

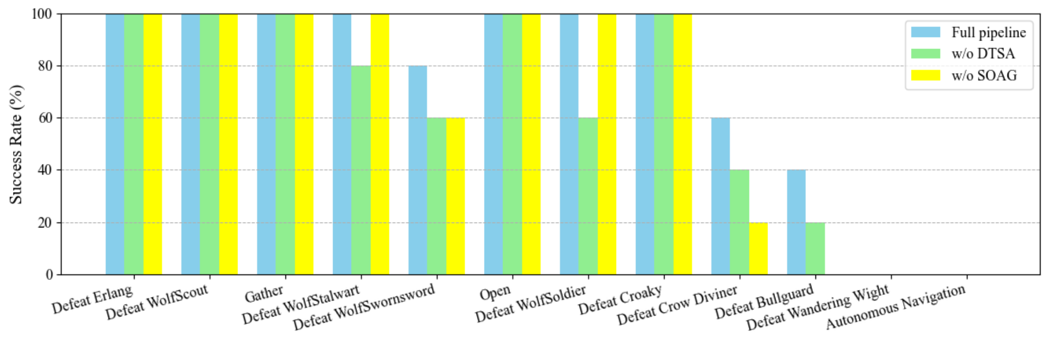
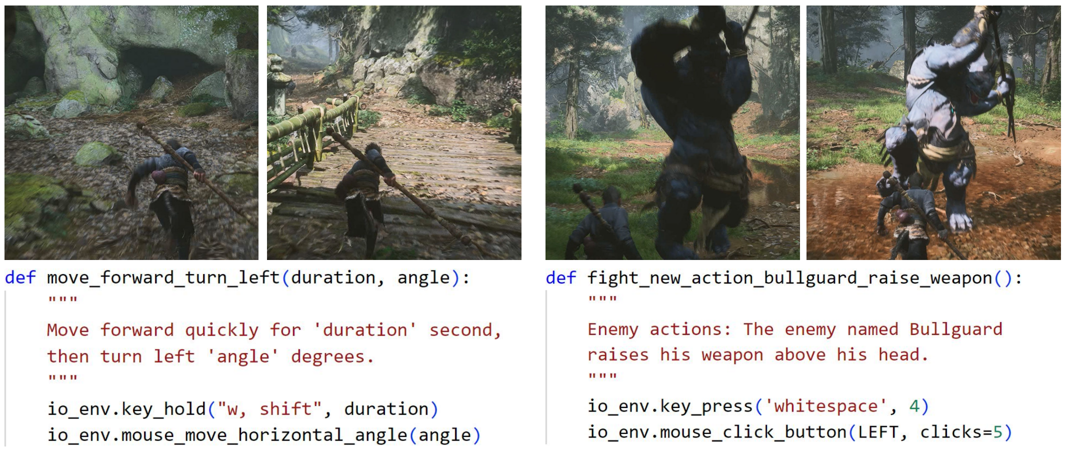


Figure 4: Ablation Study.

To validate the effectiveness of the human-guided trajectory system, we introduced human guidance and conducted a case study on task 12, a task of very hard difficulty. The objective was for the VARP agent to control the player character to move from the “Earth Temple” spawn point to the location of the Bullguard enemy within 5 minutes. GPT-4o was chosen as the VLM. The experimental results showed a success rate of 40%. This indicates that human guidance can significantly enhance the decision-making accuracy of the agent. Fig. 5 shows the new action responsible for pathfinding generated by the human-guided trajectory system during the execution of task 12. Additionally, Fig. 5 depicts the new action generated by SOAG in response to the enemy, Bullguard, during task 10. The enemy’s current action indicates an impending attack: “swinging the axe downwards three times consecutively.” Therefore, the new action should be to dodge consecutively more than three times before counterattacking.



```python
def move_forward_turn_left(duration, angle):
    """
    Move forward quickly for 'duration' second,
    then turn left 'angle' degrees.
    """
    io_env.key_hold("w, shift", duration)
    io_env.mouse_move_horizontal_angle(angle)
```

```python
def fight_new_action_bullguard_raise_weapon():
    """
    Enemy actions: The enemy named Bullguard
    raises his weapon above his head.
    """
    io_env.key_press('whitespace', 4)
    io_env.mouse_click_button(LEFT, clicks=5)
```

Figure 5: Case studies of new actions generated by human guidance and SOAG.

4 Conclusion

In this study, we have explored the boundaries of current Vision Language Models (VLMs) in the context of complex action role-playing games (ARPGs) using “*Black Myth: Wukong*” as our experimental platform. Our proposed framework, VARP, introduces a novel approach to game interaction by leveraging visual-only inputs for action planning in ARPG environments. The VARP framework demonstrates its potential by achieving an 90% success rate in basic and moderate combat scenarios, suggesting that VLMs can be effectively utilized in tasks traditionally dominated by reinforcement learning. Our proposed benchmark can effectively evaluate the performance of visual-only agents in the BMW game. Additionally, the human operation dataset we provide offers a valuable resource for future research, enabling the study of human-like gameplay and action decision-making in visually complex environments. Our findings underscore the promise of multimodal agents in enhancing generalization and performance in action-oriented tasks within video games. Moving forward, the insights gained from this research could pave the way for more sophisticated agent designs that can handle a broader range of challenges in ARPGs and beyond.

References

[1] analoganddigital. Dqn_play_sekiro. `https://github.com/analoganddigital/DQN_play_sekiro`, 2021. Accessed: 2024-09-17.

[2] Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024.

[3] 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.

[4] 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.

[5] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023.

[6] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] fange. `https://www.bilibili.com/video/BV1n74y1m7U3/`, 2024.

[8] Google. Gemini, 2024.

[9] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis, 2024.

[10] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models, 2024.

[11] 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.

[12] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4501–4510, 2020.

[13] Junkai Li, Siyu Wang, Meng Zhang, Weitao Li, Yunghwei Lai, Xinhui Kang, Weizhi Ma, and Yang Liu. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint arXiv:2405.02957*, 2024.

[14] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2024.

[15] OpenAI. Gpt-4o. `https://www.openai.com/`, 2024.

[16] 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.

[17] Siyuan Qi, Shuo Chen, Yexin Li, Xiangyu Kong, Junqi Wang, Bangcheng Yang, Pring Wong, Yifan Zhong, Xiaoyuan Zhang, Zhaowei Zhang, et al. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. *arXiv preprint arXiv:2401.10568*, 2024.

[18] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, 2024.

[19] 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.

[20] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

[21] Claude 3.5 Sonnet. New and improved embedding model, 2022.

[22] Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering foundation agents towards general computer control, 2024.

[23] turingscat. `https://www.bilibili.com/video/BV1qE421c7mU/`, 2024.

[24] 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, 2023.

[25] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception, 2024.

[26] 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, 2024.

[27] Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models, 2023.

[28] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[29] 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.

[30] Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*, 2023.

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[32] Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. Fine-tuning large vision-language models as decision-making agents via reinforcement learning, 2024.

[33] Chi Zhang, Penglin Cai, Yuhui Fu, Haoqi Yuan, and Zongqing Lu. Creative agents: Empowering agents with imagination for creative tasks, 2023.

[34] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users, 2023.

[35] Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents. 2024.

A Appendix / supplemental material

A.1 Overview

A.2 Clarifications and Limitations

Thank you for reading our paper. We would like to begin by clarifying and explaining some potential concerns that may arise due to the extreme popularity of the BMW game. We want to emphasize that our framework has broad applicability and will subsequently be generalized to include more games and other scenarios, not just limited to the BMW game. In this paper, we explored the potential of using VLMs to execute action combos in game tasks, particularly focusing on how it achieves victory against medium-powered monsters by leveraging the advantages of both action planning and visual trajectory modules. Additionally, we provided a human operation dataset, which presents possibilities for integrating technologies such as multi-modal retrieval-augmented generation, imitation learning, and reinforcement learning.

We must also candidly acknowledge some limitations in our research, specifically: 1) Task Definitions: As LLM- and VLM-based agents are still evolving, the current task definitions are somewhat simplistic. 2) Game Scenarios: Our research has only been tested within the BMW game and has not yet been extended to other scenarios. 3) Dataset Size: We have a limited amount of data, and in the future work, we plan to recruit more volunteers to collect higher quality data to enhance the depth of our research. 4) Model Capabilities: As shown in the performance evaluation section, there is still room for improvement in existing VLMs, including speed and accuracy. Therefore, it would be interesting to train an ARPG-specific VLM, such as VideoGameBunny[*].

Finally, we sincerely welcome your new ideas and feedback regarding this work, or even contribute your game records. Please feel free to reach out to us, and we look forward to exploring together, ultimately making VLMs play games as well as humans.

A.3 Related Work

A.3.1 LLM and VLM-Driven Agents

In recent years, various intelligent agents driven by large language models (LLM) and multimodal language models (VLM) have gradually come to the forefront, demonstrating immense potential in multitasking and autonomous learning. For LLM, Reflexion [20] enhances the decision-making ability of language agents through a framework of linguistic feedback and self-reflection, allowing agents to autonomously reflect in the face of feedback signals and maintain contextual memory during task execution and decision-making processes. ReAct [31], on the other hand, emphasizes real-time information retrieval and strategy adjustment. By combining reasoning with action and interacting with external knowledge sources (such as the Wikipedia API), it adds dynamic information retrieval capabilities, providing greater interpretability and controllability. Voyager [24] can explore and learn skills in an unsupervised manner within the Minecraft environment, continuously exploring the world, acquiring diverse skills, and making new discoveries without human intervention through a combination of automated courses, skill libraries, and iterative prompting mechanisms.

For VLM, CreativeAgent [33] focuses on creative tasks, employing multimodal generation to achieve the construction of complex structures. Its combination of an imagination module and controller enables efficient planning and execution based on free-form language instructions and the generated

[*]https://videogamebunny.github.io/

task details. Cradle [22] takes video images displayed on a screen as input, extracting text and visual information to make decisions through a workflow of "reflecting on the past, summarizing the present, and planning for the future," outputting control signals for keyboard and mouse interaction, allowing AI agents to interact with software like humans without relying on any internal APIs.

A.3.2 RL-based Agents in ARPGs

Reinforcement learning (RL) has shown significant improvements in video games[32, 3, 12, 4, 6, 19, 11, 17, 29] especially action role-playing games (ARPG). DQN-play-sekiro [1] employs the deep Q-network (DQN) algorithm to train AI to automate gameplay in "Sekiro: Shadows Die Twice." This project observes the game visuals and makes decisions based on the current state, gradually mastering the game strategy to defeat boss-level enemies. Additionally, AI achieves interactive learning based on reinforcement learning in "*Black Myth: Wukong*" by recognizing game images and scripting simulated keyboard input signals[23, 7]. This method uses successful dodging as positive feedback while being attacked by monsters as negative feedback, prompting the AI to optimize its decision-making process.

These creative works not only showcase the potential of AI in complex gaming environments but also provide effective means for game testing and automated gameplay. However, agents trained solely using RL methods can only be applied to a limited range of specific tasks. For new tasks, the agent needs to be retrained. Therefore, agents based on this method have poor generalization capabilities.

A.4 Additional Dataset Collection

We collected a total of approximately 1,000 valid data samples, each representing a video segment of a human completing a task along with the corresponding mouse and keyboard operation records. Among these, 4.0% represent task 1, 12.5% represent task 2, and so on. The specific information is shown in Fig. 6.

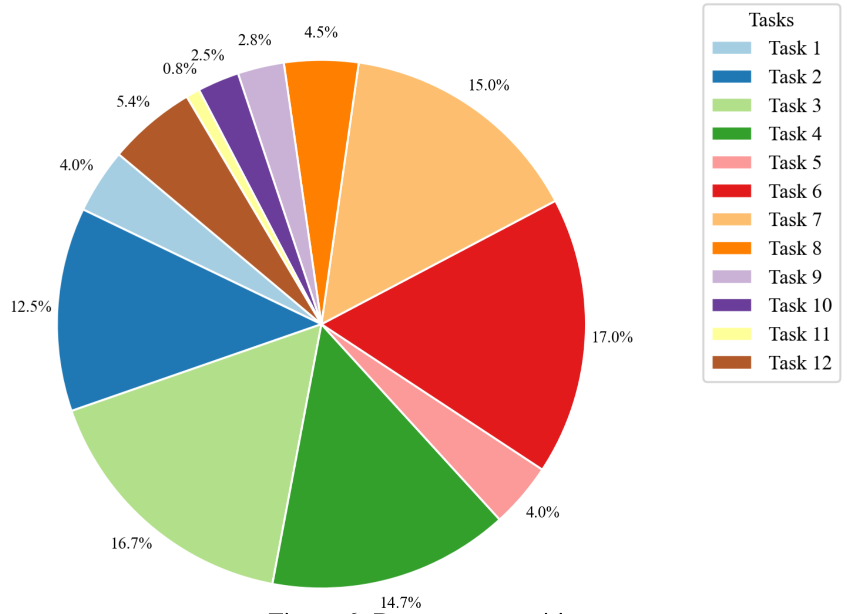


Figure 6: Dataset composition.

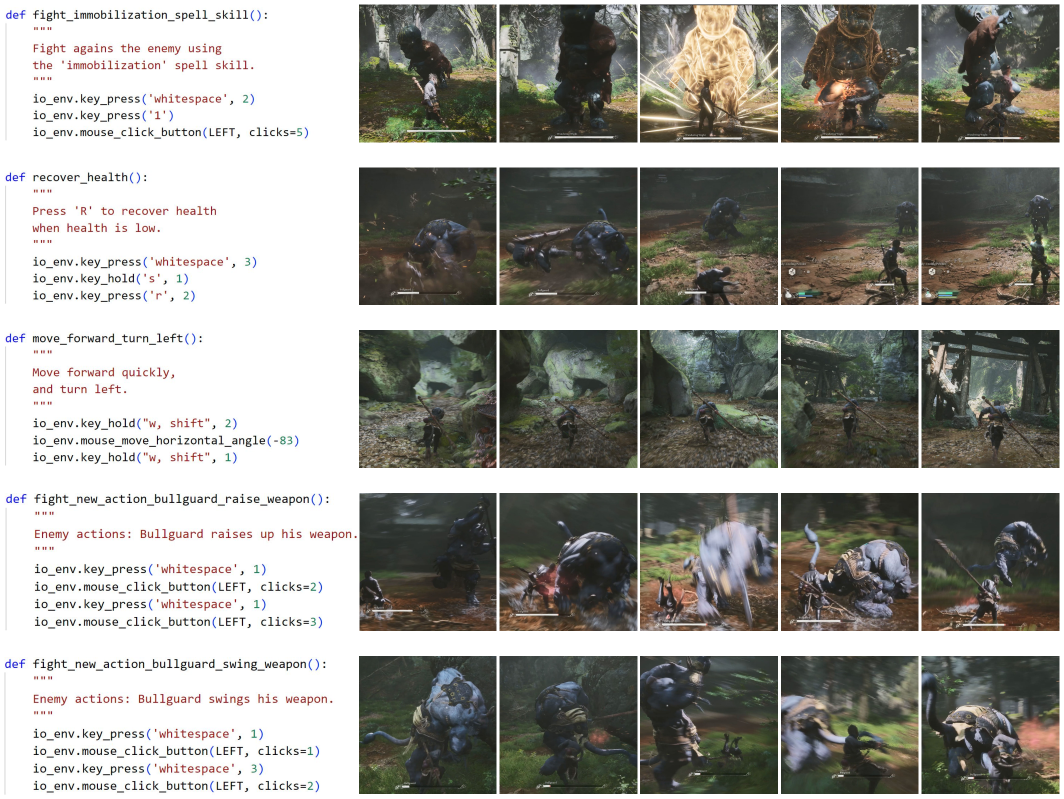
A.5 Additional Performance Evaluation

Based on Sec. 3.4 of the main text, we also recorded the average time and average inference count for the VARP agent without human guidance to complete each task. Each inference count represents the generation of an executable action, with each combat action containing an average of 8.6 atomic operations. Additionally, we recorded the number of atomic operations performed by human for each task. By dividing this number by 8.6, we estimated the inference count of human in combat tasks. As shown in Tab. 3, compared to humans, the VARP agent has a much lower inference count in task 1, task 9, and task 10. This indicates that humans tend to perform a large number of redundant operations when completing more difficult or time-consuming tasks in ARPGs, which is not conducive to task completion. In contrast, the actions generated by the VARP agent in these tasks are relatively more refined and concise.

Table 3: Additional evaluation results of the average time (in minutes) and average inference count.

Task ID	GPT-4o (time)	Claude (time)	Gemini (time)	GPT-4o (count)	Claude (count)	Gemini (count)	Human(count)
1	16.09	19.12	17.14	71.6	88	77	**98.7**
2	0.53	0.65	0.63	3.8	5	4.4	2.3
3	0.18	0.23	0.18	1.4	1.6	1.4	1.7
4	0.57	0.68	0.64	4.6	5.2	4.6	3.0
5	0.69	0.77	0.68	5.5	6.25	5.4	3.5
6	0.27	0.25	0.11	2	1.8	1.2	1.3
7	0.81	0.78	0.69	5.4	5.7	5.75	3.0
8	0.41	0.42	0.38	3.8	3.2	2.8	2.6
9	1.24	1.19	1.19	8.3	9	8.5	**16.7**
10	2.20	-	2.06	13.5	-	13	**36.6**

A.6 More Case Studies



```python
def fight_immobilization_spell_skill():
    """
    Fight agains the enemy using
    the 'immobilization' spell skill.
    """
    io_env.key_press('whitespace', 2)
    io_env.key_press('1')
    io_env.mouse_click_button(LEFT, clicks=5)
```

```python
def recover_health():
    """
    Press 'R' to recover health
    when health is low.
    """
    io_env.key_press('whitespace', 3)
    io_env.key_hold('s', 1)
    io_env.key_press('r', 2)
```

```python
def move_forward_turn_left():
    """
    Move forward quickly,
    and turn left.
    """
    io_env.key_hold("w, shift", 2)
    io_env.mouse_move_horizontal_angle(-83)
    io_env.key_hold("w, shift", 1)
```

```python
def fight_new_action_bullguard_raise_weapon():
    """
    Enemy actions: Bullguard raises up his weapon.
    """
    io_env.key_press('whitespace', 1)
    io_env.mouse_click_button(LEFT, clicks=2)
    io_env.key_press('whitespace', 1)
    io_env.mouse_click_button(LEFT, clicks=3)
```

```python
def fight_new_action_bullguard_swing_weapon():
    """
    Enemy actions: Bullguard swings his weapon.
    """
    io_env.key_press('whitespace', 1)
    io_env.mouse_click_button(LEFT, clicks=1)
    io_env.key_press('whitespace', 3)
    io_env.mouse_click_button(LEFT, clicks=2)
```

Figure 7: More case studies of actions and corresponding game screenshots.

In this section, we will showcase some predefined actions and more newly generated actions with the corresponding game screenshots. The VLM is GPT-4o.

As shown in Fig.7, the actions in the first and second rows are predefined functions. The VARP agent automatically detects whether to use these actions based on the input visual information. For example, if the immobilization spell skill can be used, the agent executes the "fight_immobilization_spell_skill" action. Similarly, if the player's health is low, it uses the "recover_health" action.

The actions in the third row are generated by the human-guided trajectory system. Human prior knowledge can effectively guide the agent to improve efficiency in navigation tasks.

The new actions in the fourth and fifth rows are summarized by SOAG after each combat interaction between the player character and the enemy and stored in the action library. These actions are specific to particular enemies and their attack patterns. For instance, in the fourth row, when the agent observes that an enemy named Bullguard is raising up his weapon, it indicates that the enemy is about to perform the action "chopping the axe downwards three times consecutively." The agent can then find a specific counter-action in the action library. At the beginning of the combat, this "fight_new_action_bullguard_raise_weapon" action is defined as dodging four times consecutively, followed by attacking five times, as shown in Fig.5. As the combat progresses, this action is optimized to counterattack during the intervals between dodges, significantly increasing the success rate and efficiency in defeating the enemy, as illustrated in the fourth row of Fig.7. This demonstrates that SOAG can continuously optimize the actions it generated.

A.7 Ethical Consideration

Our method can automatically play ARPGs, which may lead to game cheating and false advertising. This can have a significant negative impact on society. Therefore, it is crucial to consider methods that can reliably distinguish between genuine and forged content. We strongly condemn the unauthorized and malicious use of this technology and emphasize the need to consider ethical issues when using our method.

A.8 Demo Video

We have provided a detailed demo video to demonstrate the effectiveness of our VARP agent. Please refer to `https://varp-agent.github.io/`.