BattleAgentBench: A Benchmark for Evaluating Cooperation and Competition Capabilities of Language Models in Multi-Agent Systems

Anonymous ACL submission

Abstract

Large Language Models (LLMs) are becoming increasingly powerful and capable of handling complex tasks, e.g., building single agents and multi-agent systems. Compared to single agents, multi-agent systems have higher requirements for the collaboration capabilities of language models. Many benchmarks are proposed to evaluate cooperation and competition abilities of LLMs. However, existing works have overlooked scenarios where cooperation and competition coexist. Additionally, real-world environments require agents to have precise spatial perception abilities, which many existing studies have overlooked. To address these two problems, we propose a benchmark, called BattleAgentBench, which defines seven sub-stages of three varying difficulty levels and conducts a fine-grained evaluation of language models in terms of single-agent scenario navigation capabilities, paired-agent task execution abilities, and multi-agent collaboration and competition capabilities. Experimental results indicate that API-based models perform excellently on simple tasks but open-source small models struggle with simple tasks. Regarding difficult tasks that require collaborative and competitive abilities, although API-based models have demonstrated some collaborative capabilities, there is still enormous room for improvement. The code for BattleAgentBench is available at https://anonymous.4open.science/r/BattleAgentBench-256D

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Team et al., 2023; GLM et al., 2024) have showcased remarkable capabilities in handling intricate tasks (Bi et al., 2023; Jumper et al., 2021; Singhal et al., 2023; Zhang et al., 2024a,b). Therefore, researchers build language model-based agents in various scenarios, including embodied robots in the physical world (Huang et al., 2023; O'Neill et al., 2024; Hu et al., 2024d) and agents in the virtual world. Virtual agents can be roughly divided into intelligent assistants (Gur et al., 2023; Xu et al., 2024b; Hu et al., 2024c) and game agents. Game agents aim to evaluate language models' capabilities in planning, decision-making, and other aspects through game environments (Hu et al., 2024a; Xu et al., 2024a). According to the main characteristics of the games, game environments can be categorized into six categories, including adventure, communication, competition, cooperation, simulation, and exploration (Hu et al., 2024a).

Level	Level 1		Level 2		Level 3		
Ability/Stage	1	2	3	4	5	6	7
Rule Understanding	✓	✓	✓	✓	✓	✓	✓
Spatial Perception	✓	✓	✓	✓	✓	✓	✓
Competition				✓	✓	✓	✓
Static Cooperation			✓		✓		✓
Dynamic Cooperation						✓	✓

Table 1: Evaluation abilities at each stage of each level of the BattleAgentBench.

To gauge cooperation and competition aptitude of LLMs, various benchmarks have been introduced. Several benchmarks center on the competition game, such as StarCraft II (Ma et al., 2023; Shao et al., 2024) and PokéLLMon (Hu et al., 2024b). Many studies also evaluate competitive abilities in board and card games (Chen et al., 2024b), such as Chess (Feng et al., 2024), and Texas Hold'em (Huang et al., 2024; Gupta, 2023; Zhang et al., 2024c; Guo et al., 2023). While MindAgent (Gong et al., 2023), ProAgent (Zhang et al., 2023a), and S-Agents (Chen et al., 2024a) focus on Overcooked (Carroll et al., 2019) and Minecraft (BAAI, 2023; Feng et al., 2023) in the context of agent cooperation.

Exploring and evaluating agent capabilities in virtual games is driven by the expectation that language model based agents will transition from the virtual to the real world. However, existing works

have overlooked scenarios where cooperation and competition coexist. This is common in real life, such as in various team sports competitions like football, basketball, volleyball, and so on. For instance, consider the case of a badminton double challenge, wherein cooperation is imperative within an internal team while competitive dynamics unfold between two opposing teams. Additionally, real-world environments require agents to have precise spatial perception abilities, which many existing studies have overlooked.

In light of these challenges, this paper introduces `BattleAgentBench`, a fine-grained benchmark to evaluate the collaborative and competitive capabilities of LLMs in multi-agent systems. Specifically, as shown in Table 1, the `BattleAgentBench` framework encompasses three different difficulty levels and seven stages. At Level 1 (the single-agent level), we design two simple stages to assess a single agent's abilities of rule understanding and spatial perception. At Level 2 (the double-agent level), we use two stages with two agents to evaluate agents' cooperative and competitive task execution abilities. At Level 3 (the multi-agent level), we design three stages with different cooperative and competitive relationships between agents to assess agents' static cooperation competition, dynamic cooperation competition, and hybrid cooperation competition.

In addition, we have assessed 11 different LLMs using `BattleAgentBench`, including both four API-based and seven open-source models. We found that even in very simple stages, the performance of open-source small models was particularly poor, while API-based models performed well. However, in difficult tasks, although API-based models are still better than open-source small models, the gap between them has narrowed, and API-based models still have significant room for improvement.

In summary, the contributions of this work are:

- We introduce the `BattleAgentBench`, a fine-grained multi-agent cooperation and competition evaluation benchmark, which includes three levels ranging from single agent to multiple agents.
- In the `BattleAgentBench` framework, we design seven stages from easy to difficult, capable of comprehensively assessing the basic abilities of single agents, cooperative abilities, and competitive abilities of multi-agents.
- We perform a thorough evaluation of 11 different LLMs using the `BattleAgentBench`, including the leading top four API-based LLMs and seven open-source models. Although API-based models perform well on simple stages, there is still significant room for improvement on complex stages.

2 Preliminary

3 The `BattleAgentBench`

To evaluate the cooperation and competition capabilities of large language models in a fine-grained manner, we design a new benchmark: `BattleAgentBench`. As shown in Figure 1, the `BattleAgentBench` consists of two main parts: stage design and agent interaction. In stage design, we design **seven stages of three different difficulty levels**. In agent interaction, we implement interactions between agents and servers to support evaluation in the above stages. Next, we introduce the overall evaluation framework and describe each stage in detail.

3.1 Composition of `BattleAgentBench`

We evaluate LLMs as agents in a turn-based interactive game, Battle City, where the agent takes corresponding actions based on the game state at each turn.

1. Stage Design. We design three different difficulty levels: Foundational Agent Capabilities (Level 1), Paired Agent Interaction (Level 2), and Multi-Agent Dynamics (Level 3), for a fine-grained evaluation of the agents' collaborative and competitive abilities. At Level 1 (the single-agent level), we mainly evaluate a single agent's basic abilities, for example, whether can understand the game's basic rules. Regarding Level 2 (the double-agent level), we assess two agents' collaboration capabilities and competitive abilities in two stages. At Level 3 (the multi-agent level), we evaluate both the collaboration and competitive abilities of multiple agents in more complex scenarios. Each stage corresponds to a specific stage setting, defining the unique game environment for that stage, such as the number of players, the number of bases, the win/lose logic of the game, etc.

2. Agent Interaction.

Game Server. At the start of the game, the game server loads the specific environment according to the stage setting to be evaluated. During the game,

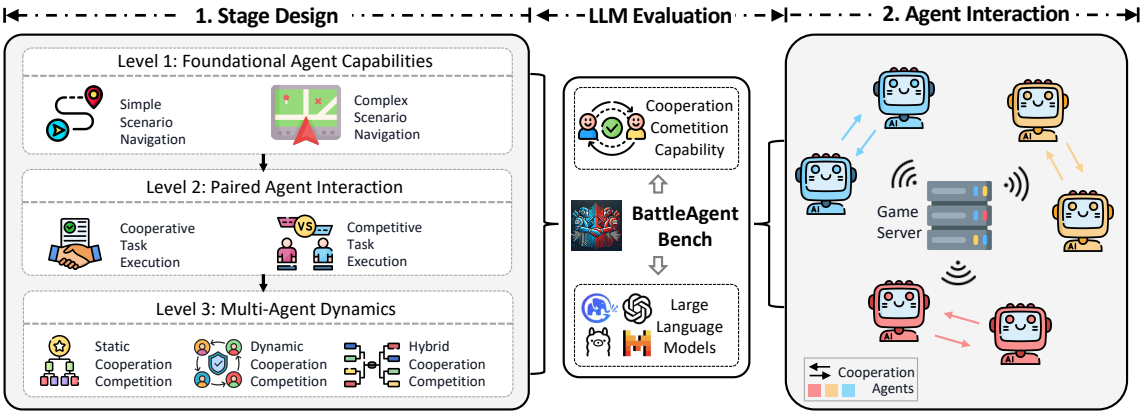
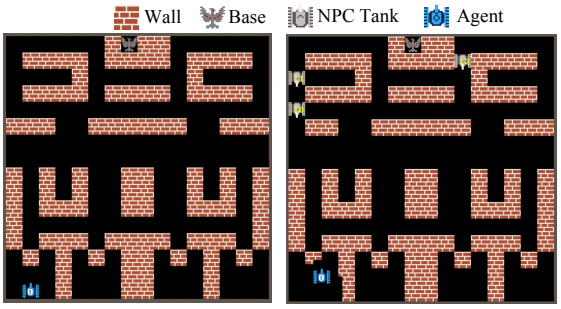


Figure 1: Overall evaluation framework of the `BattleAgentBench`.



(a) Level 1 - Stage 1 (b) Level 1 - Stage 2

Figure 2: Level 1: Stage 1 and Stage 2. The agent's goal in both stages is to reach the base location.

the game server is responsible for sending observation information to agents, receiving actions from agents, updating the status of agents and environment, calculating agents' rewards, and determining whether the game has ended.

LLMs as Game Agents. Each agent represents a player, and its core function is the decision-making function, which determines the next action based on the received observation data. We use LLMs to implement each action function. The game observation data is converted into text format based on predefined templates. Then, we prompt observation text to LLMs to obtain the output action.

Cooperation between LLM Agents. In addition to the interaction between agents and the server, we have also implemented a message communication interface between agents to support the evaluation of agents' static cooperation ability and dynamic cooperation ability.

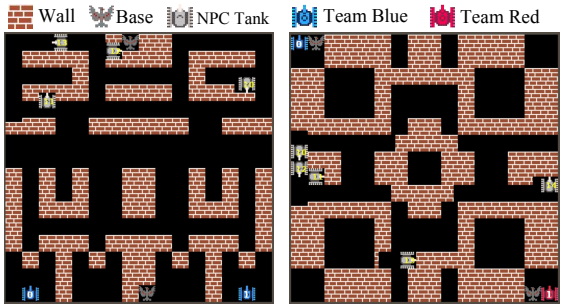
3.2 Level 1: Foundational Agent Capabilities

Compared to directly evaluating the collaboration abilities of multiple agents in complex collaborative and competitive scenarios, we believe that it is necessary to first assess the task completion capabilities of individual agents within the environment. This approach allows us to decouple the assessment of the agents' basic capabilities from their collaborative abilities to some extent. In complex environments, if an agent performs poorly, it becomes difficult to differentiate whether the subpar performance is due to the agent's basic capabilities or its collaborative abilities. Thus, we designed Level 1 (the single-agent level) as shown in Figure 2 to test basic capabilities .

Stage 1: Simple Scenario Navigation. In this stage, we test the agent's understanding of the game rules and the spatial perception ability regarding simple scenario navigation. **a) Agent Setting:** There is only one agent and one base. **b) Agent Goal:** We start with a simple goal, which is to reach the base location as quickly as possible. To achieve the goal, the agent first needs to understand the rules of the game with regard to these four elements. In addition to understanding these rules, the agent must also have good two-dimensional spatial perception skills to move correctly toward the target.

Stage 2: Complex Scenario Navigation. We assess the ability to handle dynamic obstacles. **a) Agent Setting** and **b) Agent Goal** are consistent with Stage 1. The agent needs to eliminate the tanks that pose a threat to itself while approaching the base. **c) Change:** Compared to Stage 1, this stage introduces interfering Non-Player Character (NPC) tanks. These NPC tanks will move and shoot randomly. This requires the agent to have the ability to handle dynamic obstacles.

In the next five stages, we also keep the basic game



(a) Level 2 - Stage 3 (b) Level 2 - Stage 4

Figure 3: Stages of Level 2 (double-agent level). In Stage 3 and Stage 4, the two agents have a cooperative relationship and a competitive relationship respectively.

environment unchanged, primarily increasing the number of agents from different factions and setting more complex objectives.

3.3 Level 2: Paired Agent Interaction

Compared to Level 1 (single-agent level), Level 2 (double-agent level) introduces another agent, allowing for further evaluation of the agent's cooperation and competition abilities. Moreover, compared to a multi-agent level, the two-agent environment can assess the agent's cooperation and competition abilities with the minimum number of agents, minimizing the influence of the number of agents on the results. The collaborative relationships among multiple agents are more complex, so we designed this level to gradually increase the difficulty of assessing collaborative and competitive capabilities, as shown in Figure 3.

Stage 3: Cooperative Task Execution. In this stage, we evaluate the cooperative task execution ability of agents. **a) Agent Setting:** two agents belong to the same faction. There are two bases: one is the own base, and the other is the enemy base. **b) Agent Goal:** The goal of the two agents is to eliminate the enemy base while protecting their own base from being attacked. **c) Change:** The two agents have a cooperative relationship and need to work together to accomplish this objective.

To facilitate cooperation between the agents, we have introduced a communication interface in the game that allows the two agents to send collaboration messages to each other in natural language. In each action step, an agent can autonomously decide whether to send a collaboration message and propose a specific cooperation plan. The other agent can then reply to the collaboration message, indicating whether it accepts or rejects the proposed cooperation plan.

Stage 4: Competitive Task Execution. In this stage, we assess the competitive task execution abilities of agents. **a) Agent Setting:** There are two agents and two bases. The two agents belong to different factions. **b) Agent Goal:** their objective is to destroy the opposing faction's base while protecting their own base from being destroyed. **c) Change:** The two agents have a competitive relationship and need to defeat their opponents.

3.4 Level 3: Multi-Agent Dynamics

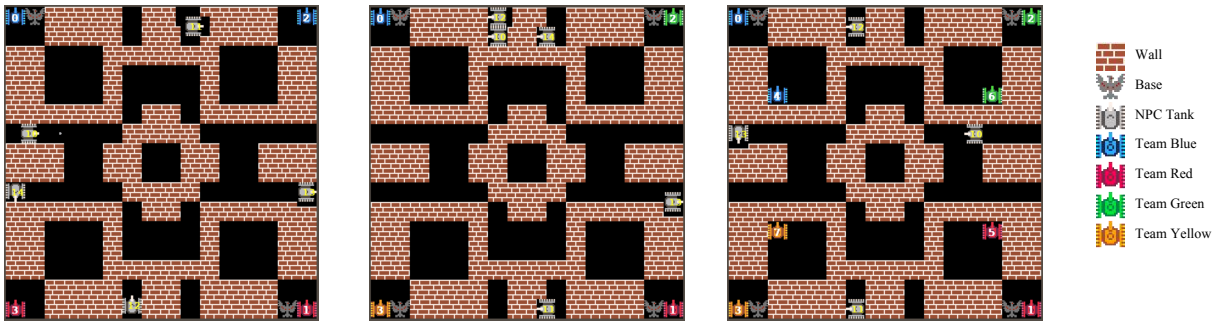
Compared to Level 2 (double-agent level), we further increase the number of agents and consider more complex cooperative and competition scenarios in Figure 4, such as managing dynamic cooperative relationships and facing multiple competition factions.

Stage 5: Static Cooperation Competition. **a) Agent Setting:** There are four agents and two bases. Four agents are divided into two teams: the Red Team and the Blue Team, with two agents on each team. **b) Agent Goal:** The goal is to destroy the opposing faction's base while protecting their own base from being destroyed. **c) Change:** In this stage, the agents have both cooperative and competitive relationships.

We restrict cooperation to only teammates within the same team, meaning that the agents of the Red Team can cooperate with each other, and the agents of the Blue Team can cooperate with each other. Since the cooperation partners for each agent are fixed and unchanging, we refer to this as static cooperative competition.

Stage 6: Dynamic Cooperation Competition. **a) Agent Setting:** There are four agents and four bases. Four agents belong to different teams: the Red Team, Blue Team, Green Team, and Yellow Team. **b) Agent Goal:** Each team can only win by defeating the other three teams. **c) Change:** This stage introduces a dynamic cooperative relationship.

During the confrontation, in pursuit of ultimate victory, each agent can freely engage in temporary cooperation with one of the other three teams. For example, two teams may choose to collaborate to eliminate a common enemy before turning against each other. Since the cooperation relationships between agents are dynamically changing, we refer to this as dynamic cooperative competition.



(a) Level 3 - Stage 5 (b) Level 3 - Stage 6 (c) Level 3 - Stage 7

Figure 4: Stages of Level 3 (multi-agent level). In Stage 5, the agents within the team have a cooperative relationship, while the agents between teams have a competitive relationship. In Stage 6, the agents between teams have a competitive relationship, while allowing for cooperative relationships between teams. In Stage 7, the relationship within the team is cooperative, the relationship between teams is competitive, and cooperation between teams is also allowed.

Stage 7: Hybrid Cooperation Competition. This stage includes dynamic and static. **a) Agent Setting:** There are eight agents and four bases. We divide eight agents into four teams, the Red Team, Blue Team, Green Team, and Yellow Team, each team consisting of two agents. **b) Agent Goal:** Each team can only win by defeating the other three teams. **c) Change:** This level has both static cooperative relationships and dynamic cooperative relationships.

During the confrontation, the cooperation among teammates remains constant, and the ultimate goal is also aligned: to eliminate the other teams. At the same time, agents can temporarily collaborate with an agent from another team. Because the ultimate goals of the other teams differ, this cooperation relationship is temporary. Therefore, we refer to this scenario as mixed cooperative competition.

4 Evaluation of `BattleAgentBench`

4.1 Evaluation Setup

Metrics. In Stage 1 and Stage 2, we primarily use the **Forward Distance** (F Dis) as the evaluation metric, since the goal of these two stages is to reach the target base. The distance the agent moves toward the base is a suitable metric. Specifically, given the tank's initial position p_s, the base's position p_{target}, and the tank's position at the end of the game p_e, we can calculate the tank's forward distance as follows.

$$\text{F Dis} = \text{L1}(p_s - p_{target}) - \text{L1}(p_e - p_{target}), \quad (1)$$

where L1 denotes L1 distance. We also introduced two auxiliary metrics: **Format Accuracy** (F Acc) and **Movement Accuracy** (M Acc). We predefined the output format for instructions and used regular expressions to match LLM's output. The format accuracy measures the LLM's ability to output in the specified format.

$$\text{F Acc} = N_{format}/N_{total}, \quad (2)$$

where N_{format} represents the number of turns where the model output is correctly formatted, while N_{total} represents the total number of turns. Since the target position is known, we can determine whether the agent's movement direction is correct at each step, so the movement accuracy is calculated as follows:

$$\text{M Acc} = N_{correct}/N_{format}, \quad (3)$$

where $N_{correct}$ denotes the number of turns with correct movement direction.

In the subsequent five stages, we use **Score** as the evaluation metric. The agents will earn a certain score for successfully attacking bases, other agents, and NPC tanks. The sum of the three types of scores will serve as the final score. Specifically, the agent's health is 5, and the NPC tank's health is 1. Each attack on the agent and tank obtains 1 score, while each attack on the base obtains 5 scores.

It should be noted that these 5 stages have multiple agents. Each agent needs to connect to a language model. To facilitate the evaluation of different language models, we have adopted the concept of primary and secondary agents. The primary agent is the first agent or the first team of agents, while the remaining agents are secondary agents. Each stage can connect up to two types of

Models	Stage 1			Stage 2			Avg. Dis
	F Dis	F Acc	M Acc	F Dis	F Acc	M Acc	
API-based models							
claude3.5-sonnet-0620	**13.7**	**1.00**	0.97	**12.0**	**1.00**	**0.90**	**12.8**
gpt-4o-mini	12.3	**1.00**	**0.98**	10.7	**1.00**	0.86	11.5
gpt-3.5-turbo-0125	10.7	**1.00**	0.83	5.7	**1.00**	0.65	8.2
glm-4-flash	4.3	**1.00**	0.60	5.0	0.92	0.68	4.7
open-source models							
internlm2.5-7b-chat	**6.7**	0.99	**0.63**	4.7	0.99	0.63	**5.7**
mistral-7b-instruct	3.3	0.98	0.58	**5.8**	0.98	**0.65**	4.6
glm-4-9b-chat	3.8	**1.00**	0.61	4.2	**1.00**	0.63	4.0
qwen2-7b-Instruct	3.7	0.98	0.60	2.1	0.99	0.56	2.9
yi-1.5-9b-chat-16k	1.9	**1.00**	0.54	1.9	**1.00**	0.55	1.9
gemma2-9b-it	2.8	0.98	0.60	0.7	0.99	0.55	1.8
llama3-8b-instruct	-1.4	0.99	0.41	-1.1	0.99	0.45	-1.3
random	1.0	-	0.49	1.4	-	0.52	1.2

Table 2: Evaluation results of Level 1. **Bold** denotes the best result on API-based models and open-source models.

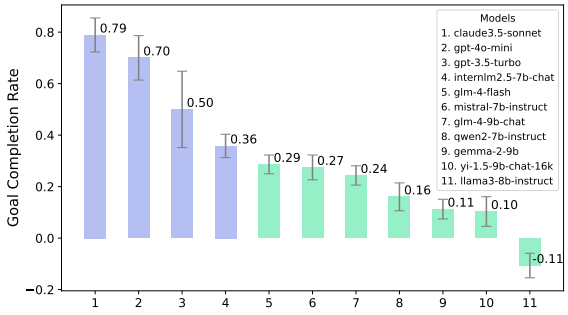


Figure 5: Goal completion rate of different models.

language models, with the primary agent connecting to the language model being evaluated, and the secondary agents connecting to another reference language model. Here, we have used yi-1.5-9b as the reference language model. By keeping the secondary agents connected to the reference language model unchanged, we can fairly connect and evaluate different language models in the primary agent. For example, in Level 2, there is only one team of 2 agents, so both agents are primary agents, connecting to the language model being evaluated, and their scores are summed for the final score.

Baselines. We evaluate two types of models: API-based models and open-source models. The API-based models include Claude3.5[1], GPT-4o-mini (Achiam et al., 2023), GPT-3.5 (Achiam et al., 2023), and GLM-4 (GLM et al., 2024). The open-source models include Llama3-8b, Mistral-7b (Jiang et al., 2023), Gemma2-9b (Team et al., 2024), GLM-4-9b (GLM et al., 2024), Yi-1.5-9b (Young et al., 2024), Internlm2.5-7b (Team, 2023), and Qwen2-7b (Yang et al., 2024).

4.2 Overall Experimental Results

Evaluation on Level 1. API-based models have stronger spatial perception capabilities than open-source small models, they can effectively move towards the target to complete navigation tasks. As shown in Table 2, API-based models have a significant advantage over open-source small models in terms of *forward distance*, especially claude3.5-sonnet and gpt-4o-mini, which can approach the base well. In contrast, open-source models make less progress. Based on the forward distance and total distance to the target, we can calculate the *goal completion rate*. The results are shown in Figure 5. The results show that claude3.5-sonnet and gpt-4o-mini can effectively complete navigation tasks. The *goal completion rates* of open-source models are all below 50%. In addition, open-source small models have lower *movement accuracy* while API-based models have higher *movement accuracy*. This indicates that open-source small models have poor two-dimensional spatial perception, with many movements being incorrect. This explains why these models make less progress.

Both types of models have high format accuracy, indicating that they can both follow the given action format well. The evaluation results of these two stages indicate that API-based models can understand the game environment and have a good spatial perception, thus effectively completing objectives. In contrast, open-source models are generally unable to effectively accomplish the goals.

Evaluation on Level 2. claude3.5-sonnet and gpt-4o-mini have better collaborative capabilities and adversarial capabilities. From Table 3, it can be seen that in collaborative and adversarial environments, among all models, only claude3.5-sonnet and gpt-4o-mini achieved relatively high scores, while other models scored comparatively lower. As the difficulty increases, only claude3.5-sonnet maintained a consistent movement accuracy, while the movement accuracy of gpt-4o-mini, gpt-3.5, and glm-4-flash all showed a decline. The movement accuracy of open-source models remains as poor as in Level 1. Due to the increased complexity of instruction formats, all models experienced some decline in format accuracy. However, API-based models showed a smaller decrease in format accuracy. These results demonstrate that claude3.5-sonnet has strong environmental understanding, collaboration, and adversarial capabilities, while gpt-4o-mini achieve the second-best performance in these aspects. Other models are unable

[1] https://www.anthropic.com/news/claude-3-5-sonnet

Models	Stage 4			Stage 5			Avg. Score
	Score	F Acc	M Acc	Score	F Acc	M Acc	
API-based models							
claude3.5-sonnet-0620	**5.3**	**1.00**	**0.96**	**4.3**	**1.00**	**0.90**	**4.8**
gpt-4o-mini	3.3	0.96	0.69	1.3	0.90	0.65	2.3
gpt-3.5-turbo-0125	0.3	0.97	0.63	0.3	0.88	0.63	0.3
glm-4-flash	0.3	0.92	0.63	0.7	0.98	0.55	0.5
open-source models							
internlm2.5-7b-chat	**1.8**	0.73	0.54	0.2	0.84	0.57	**1.0**
yi-1.5-9b-chat-16k	1.4	0.72	0.54	**0.4**	**0.96**	0.46	0.9
llama3-8b-instruct	0.4	0.81	0.34	**0.4**	0.63	0.57	0.4
glm-4-9b-chat	0.4	0.76	0.44	**0.4**	0.80	**0.64**	0.4
qwen2-7b-Instruct	0.4	0.65	**0.58**	0.2	0.88	0.56	0.3
gemma2-9b-it	0.4	**0.82**	0.53	0.0	0.10	0.42	0.2
mistral-7b-instruct	0.3	0.55	0.56	0.0	0.15	0.61	0.1
random	0.2	-	0.48	0.0	-	0.49	0.1

Table 3: Evaluation results of Level 2 of the `BattleAgentBench`. **Bold** denotes the best result on API-based models and open-source models.

Models	Stage 5			Stage 6			Stage 7			Avg. Score
	Score	F Acc	M Acc	Score	F Acc	M Acc	Score	F Acc	M Acc	
API-based models										
claude3.5-sonnet-0620	**6.3**	**1.00**	**0.91**	**7.0**	**1.00**	**0.98**	**8.3**	**1.00**	**0.92**	**7.2**
gpt-4o-mini	5.7	0.95	0.71	4.7	0.96	0.77	2.0	0.89	0.64	4.1
gpt-3.5-turbo-0125	1.0	0.97	0.72	1.3	0.88	0.70	2.3	0.98	0.63	1.6
glm-4-flash	0.7	0.95	0.54	2.0	0.95	0.64	0.3	0.98	0.91	1.0
open-source models										
internlm2.5-7b-chat	0.6	0.76	0.48	**2.2**	0.67	0.42	**3.2**	0.66	0.55	**2.0**
yi-1.5-9b-chat-16k	1.0	**0.89**	**0.56**	1.8	0.80	0.49	2.2	0.77	0.52	1.7
gemma2-9b-it	**1.8**	0.83	0.39	0.6	**0.87**	0.47	1.8	**0.86**	0.57	1.4
llama3-8b-instruct	1.0	0.85	0.47	0.8	0.85	0.55	1.2	0.83	**0.73**	1.0
glm-4-9b-chat	1.0	0.82	0.48	0.6	0.80	**0.63**	1.0	0.83	0.72	0.9
qwen2-7b-Instruct	0.0	0.69	0.52	1.4	0.63	0.47	1.2	0.65	0.64	0.9
mistral-7b-instruct	0.2	0.49	0.38	0.0	0.54	0.43	0.2	0.47	0.49	0.1
random	0.2	-	0.50	0.4	-	0.49	0.6	-	0.52	0.4

Table 4: Evaluation results of Level 3 of the `BattleAgentBench`. **Bold** denotes the best result on API-based models and open-source models.

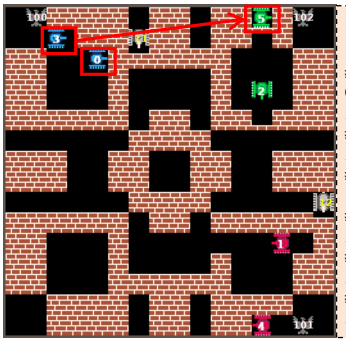
to cope with complex environments effectively.

Evaluation on Level 3. Only claude3.5-sonnet and gpt-4o-mini achieve effective collaboration in adversarial environments, while collaboration between other models is ineffective or even harmful. As shown in Table 4, we can see that in more complex environments, claude3.5-sonnet and gpt-4o-mini still obtain relatively high scores. Other models scored comparatively lower. Notably, only claude3.5-sonnet maintains a high movement accuracy. gpt-4o-mini achieved performance second only to claude3.5-sonnet across all metrics. Compared to these two models, other models have lower scores, movement accuracy, and format accuracy. This indicates that these two models have stronger capabilities in dealing with complex environments. The other models perform poorly in lower-level tasks and continue to underperform in higher-level tasks.

Evaluation results of Level 3 reveal the differences in the comprehensive capabilities of different models in complex environments, including environmental understanding, cooperation, competition abilities, etc. To assess the effectiveness of models' collaborative abilities, we conduct further ablation experiments that remove the cooperation interface between agents in the Level 3 evaluation. This means that while the team relationships between agents remained unchanged, no agent could cooperate with any other. The ablation results are shown in Table 5.

Models	Stage 5			Stage 6			Stage 7			Avg. Score	Δ Score
	Score	F Acc	M Acc	Score	F Acc	M Acc	Score	F Acc	M Acc		
API-based models											
claude3.5-sonnet-0620	**4.3**	**1.00**	**0.87**	**3.3**	**1.00**	**0.87**	**7.0**	**1.00**	**0.89**	**4.9**	-2.3
gpt-4o-mini	**4.3**	0.92	0.74	3.0	0.90	0.73	4.0	0.90	0.76	3.8	-0.3
gpt-3.5-turbo-0125	1.3	0.96	0.63	2.3	0.93	0.59	2.7	0.96	0.56	2.1	0.6
glm-4-flash	1.0	**1.00**	0.48	2.0	**1.00**	0.55	6.0	**1.00**	0.61	3.0	2.0
open-source models											
yi-1.5-9b-chat-16k	1.8	**0.94**	0.56	1.6	**0.94**	0.51	5.6	0.89	0.50	**3.0**	1.3
gemma2-9b-it	**2.0**	**0.94**	**0.60**	2.0	0.93	0.55	4.0	**0.92**	**0.62**	2.7	1.3
glm-4-9b-chat	0.6	0.79	0.45	1.0	0.79	0.58	**6.0**	0.82	0.57	2.5	1.7
internlm2.5-7b-chat	0.6	0.78	0.45	**3.0**	0.75	0.46	3.8	0.77	0.49	2.5	0.5
qwen2-7b-instruct	0.0	0.81	0.50	1.0	0.85	0.57	3.6	0.80	0.55	1.5	0.7
mistral-7b-instruct	0.0	0.26	0.46	0.2	0.17	0.35	2.3	0.23	0.24	0.8	0.7
llama3-8b-instruct	0.0	0.64	0.53	1.2	0.62	**0.69**	0.8	0.55	0.55	0.7	-0.3

Table 5: Ablation Evaluation results of Level 3 of BattleAgentBench. **Bold** denotes the best result on API-based models and open-source models. Δ Score represents Avg. Score in Table 5 subtracts Avg. Score in Table 4.



Input

#Game State Data:
Own id: **3**.
...
#Task Description:
...
#Game Instructions:
...
#Operation Options:
...
#Cooperation Options:
...
#Last Round Operation:
...

Output

#Thought process:

Attack target: Continue attacking tank 5. Although the previous round's attack didn't cause damage, this might be because we're not close enough yet. Tank 5 is located at (384, 0), the closest enemy advanced tank to our base, posing a serious threat.

Attack plan: I'm currently at (64, 32), facing right. To approach tank 5, I need to continue moving right. After moving, if within range, I'll shoot immediately.

Cooperation plan: Since both my teammate 0 and I are advanced tanks, we can continue to cooperate. Although our previous cooperation was aimed at tank 1, **considering tank 5's greater threat, we should adjust our cooperation target.**

#Attack operation:
Target 5: #Move_right#

#Cooperation operation:
#Request_coop# 0: Please adjust cooperation target, assist in attacking enemy tank 5 located at (384, 0), it poses the greatest threat to our base.

Figure 6: Cooperation example of claude3.5-sonnet to finish decision-making process in Stage 7.

Comparing Tables 4 and Table 5, we observe that after removing the collaboration interface, only the scores of claude3.5-sonnet and gpt-4o-mini decrease, while the scores of other models increase. This indicates that only claude3.5-sonnet and gpt-4o-mini's cooperation is effective, allowing them to achieve higher scores through cooperation. Conversely, collaboration among other models lowers their scores, indicating that cooperation between these models is ineffective or even harmful. **Although claude3.5-sonnet and gpt-4o-mini improve their scores through cooperation, we also notice that the improvement from cooperation is relatively small.** Through analysis of the models' battle records, we observe that the models currently exhibit some basic cooperative strategies. We will illustrate this finding through a case study.

Case Study of Cooperation between Agents. Let's illustrate this with an example in Figure 6. This example is claude3.5-sonnet's decision-making process in a certain round of Stage 7. In terms of attack operations, claude3.5-sonnet first analyzed the current game situation to identify the biggest threat and moved towards it to prepare for subsequent attacks. In terms of collaborative operations, given the enormous threat posed by this attack target, claude3.5-sonnet timely initiated a collaboration request to teammates to prioritize eliminating this enemy, indicating collaborative abilities of claude3.5-sonnet.

5 Conclusion

This work introduces the BattleAgentBench, a benchmark for LLM-based multi-agent collaboration and competition evaluation, which defines seven sub-stages of three varying difficulty levels and conducts a fine-grained evaluation of language models in terms of single-agent scenario navigation capabilities, paired-agent task execution abilities, and multi-agent collaboration and competition capabilities. We perform a thorough evaluation of 11 LLMs. Although API-based models have demonstrated some collaborative capabilities, there is still enormous room for improvement.

6 Limitations

Currently, the game design mechanisms are relatively simple. In the future, we will increase the game difficulty by adding impenetrable walls and partial observation information to further evaluate the cooperation and competition abilities of language models. The current evaluation uses the standard COT prompts. In the future, we will try better prompting strategies and also consider optimizing cooperation mechanisms to improve agent performance in cooperative and competitive environments.

References

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

Saaket Agashe, Yue Fan, and Xin Eric Wang. 2023. Evaluating multi-agent coordination abilities in large language models. *arXiv preprint arXiv:2310.03903*.

Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning*, pages 337–371. PMLR.

PKU BAAI. 2023. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. 2023. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619:1–6.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32.

Jiaqi Chen, Yuxian Jiang, Jiachen Lu, and Li Zhang. 2024a. S-agents: self-organizing agents in open-ended environment. *arXiv preprint arXiv:2402.04578*.

Junzhe Chen, Xuming Hu, Shuodi Liu, Shiyu Huang, Wei-Wei Tu, Zhaofeng He, and Lijie Wen. 2024b. Llmarena: Assessing capabilities of large language models in dynamic multi-agent environments. *arXiv preprint arXiv:2402.16499*.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2024. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36.

Yicheng Feng, Yuxuan Wang, Jiazheng Liu, Sipeng Zheng, and Zongqing Lu. 2023. Llama rider: Spurring large language models to explore the open world. *arXiv preprint arXiv:2310.08922*.

Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Ran Gong, Qiuyuan Huang, Xiaojian Ma, Hoi Vo, Zane Durante, Yusuke Noda, Zilong Zheng, Song-Chun Zhu, Demetri Terzopoulos, Li Fei-Fei, et al. 2023. Mindagent: Emergent gaming interaction. *arXiv preprint arXiv:2309.09971*.

Jiaxian Guo, Bo Yang, Paul Yoo, Bill Yuchen Lin, Yusuke Iwasawa, and Yutaka Matsuo. 2023. Suspicion-agent: Playing imperfect information games with theory of mind aware gpt-4. *arXiv preprint arXiv:2309.17277*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Akshat Gupta. 2023. Are chatgpt and gpt-4 good poker players?–a pre-flop analysis. *arXiv preprint arXiv:2308.12466*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Sihao Hu, Tiansheng Huang, Fatih Ilhan, Selim Tekin, Gaowen Liu, Ramana Kompella, and Ling Liu. 2024a. A survey on large language model-based game agents. *arXiv preprint arXiv:2404.02039*.

Sihao Hu, Tiansheng Huang, and Ling Liu. 2024b. Pok\'ellmon: A human-parity agent for pok\'emon battles with large language models. *arXiv preprint arXiv:2402.01118*.

Siyuan Hu, Mingyu Ouyang, Difei Gao, and Mike Zheng Shou. 2024c. The dawn of gui agent: A preliminary case study with claude 3.5 computer use. *arXiv preprint arXiv:2411.10323*.

Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. 2024d. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

Wenyue Hua, Lizhou Fan, Lingyao Li, Kai Mei, Jianchao Ji, Yingqiang Ge, Libby Hemphill, and Yongfeng Zhang. 2023. War and peace (waragent): Large language model-based multi-agent simulation of world wars. *arXiv preprint arXiv:2311.17227*.

Chenghao Huang, Yanbo Cao, Yinlong Wen, Tao Zhou, and Yanru Zhang. 2024. Pokergpt: An end-to-end lightweight solver for multi-player texas hold'em via large language model. *arXiv preprint arXiv:2401.06781*.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. 2023. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning*, pages 540–562. PMLR.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Mingyu Jin, Beichen Wang, Zhaoqian Xue, Suiyuan Zhu, Wenyue Hua, Hua Tang, Kai Mei, Mengnan Du, and Yongfeng Zhang. 2024. What if llms have different world views: Simulating alien civilizations with llm-based agents. *arXiv preprint arXiv:2402.13184*.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. 2021. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589.

Zhao Kaiya, Michelangelo Naim, Jovana Kondic, Manuel Cortes, Jiaxin Ge, Shuying Luo, Guangyu Robert Yang, and Andrew Ahn. 2023. Lyfe agents: Generative agents for low-cost real-time social interactions. *arXiv preprint arXiv:2310.02172*.

Nian Li, Chen Gao, Yong Li, and Qingmin Liao. 2023. Large language model-empowered agents for simulating macroeconomic activities. *Available at SSRN 4606937*.

Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. 2023. Avalonbench: Evaluating llms playing the game of avalon. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Weiyu Ma, Qirui Mi, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. 2023. Large language models play starcraft ii: Benchmarks and a chain of summarization approach. *arXiv preprint arXiv:2312.11865*.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299. IEEE.

Gabriel Mukobi, Hannah Erlebach, Niklas Lauffer, Lewis Hammond, Alan Chan, and Jesse Clifton. 2023. Welfare diplomacy: Benchmarking language model cooperation. *arXiv preprint arXiv:2310.08901*.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. 2024. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE.

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

Chen Qian and Xin Cong. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6:3.

Xiao Shao, Weifu Jiang, Fei Zuo, and Mengqing Liu. 2024. Swarmbrain: Embodied agent for real-time strategy game starcraft ii via large language models. *arXiv preprint arXiv:2401.17749*.

Zijing Shi, Meng Fang, Shunfeng Zheng, Shilong Deng, Ling Chen, and Yali Du. 2023. Cooperation on the fly: Exploring language agents for ad hoc teamwork in the avalon game. *arXiv preprint arXiv:2312.17515*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

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

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models

based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Shenzhi Wang, Chang Liu, Zilong Zheng, Siyuan Qi, Shuo Chen, Qisen Yang, Andrew Zhao, Chaofei Wang, Shiji Song, and Gao Huang. 2023. Avalon's game of thoughts: Battle against deception through recursive contemplation. *arXiv preprint arXiv:2310.01320*.

Martin Weiss, Nasim Rahaman, Manuel Wuthrich, Yoshua Bengio, Li Erran Li, Bernhard Schölkopf, and Christopher Pal. 2023. Rethinking the buyer's inspection paradox in information markets with language agents.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Shuang Wu, Liwen Zhu, Tao Yang, Shiwei Xu, Qiang Fu, Yang Wei, and Haobo Fu. 2024. Enhance reasoning for large language models in the game werewolf. *arXiv preprint arXiv:2402.02330*.

Xinrun Xu, Yuxin Wang, Chaoyi Xu, Ziluo Ding, Jiechuan Jiang, Zhiming Ding, and Börje F Karlsson. 2024a. A survey on game playing agents and large models: Methods, applications, and challenges. *arXiv preprint arXiv:2403.10249*.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. 2024b. Androidlab: Training and systematic benchmarking of android autonomous agents. *arXiv preprint arXiv:2410.24024*.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023. Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2024c. Language agents with reinforcement learning for strategic play in the werewolf game. In *Forty-first International Conference on Machine Learning*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. 2023a. Proagent: Building proactive cooperative ai with large language models. *CoRR*.

Dan Zhang, Ziniu Hu, Sining Zhoubian, Zhengxiao Du, Kaiyu Yang, Zihan Wang, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Sciglm: Training scientific language models with self-reflective instruction annotation and tuning. *arXiv preprint arXiv:2401.07950*.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024b. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. 2023b. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. 2023c. Exploring collaboration mechanisms for llm agents: A social psychology view. *arXiv preprint arXiv:2310.02124*.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. 2024c. Agent-pro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*.

Zhonghan Zhao, Kewei Chen, Dongxu Guo, Wenhao Chai, Tian Ye, Yanting Zhang, and Gaoang Wang. 2024. Hierarchical auto-organizing system for open-ended multi-agent navigation. *arXiv preprint arXiv:2403.08282*.

7 Preliminary

We evaluate LLMs as agents in a turn-based interactive game, Battle City, where the agent takes corresponding actions based on the game state at each turn. Thus, the entire interaction process between the agent and the game can be viewed as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{I}, \mathcal{O})$, which contains state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, reward function $\mathcal{R}$, game instruction $\mathcal{I}$, and observation space $\mathcal{O}$. The game server is responsible for state transitions and providing rewards. The agent needs to make an action a based on instruction i, observation o, and state s. We apply LLMs to implement the agent's decision-making function:

$$P(a_t|i, o_t, s_t) = \text{LLM}(i, o_t, s_t) \tag{4}$$

8 Basic Rules of BattleAgent

Specifically, the game contains four elements: the map, walls, bases, and tank as shown in Figure 2. Details and rules of this game are as follows:

- The game map size is 512x512, with (0, 0) representing the top-left corner and (512, 512) representing the bottom-right corner.
- The map contains tanks, bases, and walls; the tank size and the base size are 32x32, and the wall size is 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and other tanks can obstruct the movement of the tank.
- The vertical movement range of the tank is from 0 to 512, and the horizontal movement range is 0∼512.
- Tanks have four orientations: up, down, left, and right, and shooting can destroy the tank or wall in front of them in the current direction.
- When the tank's front faces the map boundary, it cannot continue to move forward.
- If there is a wall in front of the tank, it needs to shoot to eliminate the wall before it can continue to move forward.

9 Related Works

LLM-based Multi-Agents. Large Language Models (LLMs) are becoming increasingly powerful and capable of handling complex tasks. Hence, LLM-based multi-agent systems have rapidly developed and achieved considerable progress in complex problem-solving and world simulation (Wu et al., 2023; Guo et al., 2024). The primary advantage of employing Large Language Model Multi-Agent (LLM-MA) systems for problem-solving is to leverage the diverse and specialized expertise of individual agents. By working together, these agents can tackle intricate challenges across various domains, such as software development (Qian and Cong, 2023; Hong et al., 2023) and embodied AI (Mandi et al., 2024; Zhang et al., 2023b). In these systems, these agents typically collaborate to accomplish shared goals. Unlike problem-solving systems, world simulation systems aim to reflecting the complexity and variety of real-world interactions and they usually involve diverse methods of agent communication. Research on world simulation is rapidly developing in many fields, such as social sciences (Park et al., 2023; Kaiya et al., 2023; Gao et al., 2023), gaming (Xu et al., 2023; Light et al., 2023), psychology (Aher et al., 2023; Zhang et al., 2023c), and economics (Li et al., 2023; Weiss et al., 2023). Although the communication patterns between agents are typically cooperative or competitive, the knowledge and capabilities required for agents to achieve effective cooperation or competition vary significantly across different fields. In this paper, we mainly focus on multi-agent cooperation and competition in games.

According to the main characteristics of the games, game environments can be categorized into six categories, including adventure, communication, competition, cooperation, simulation, and exploration (Hu et al., 2024a). We focus on games related to competition and cooperation.

Competition Games. In competitive games, players need to compete with each other to achieve ultimate victory. StarCraft II (Vinyals et al., 2017) and Pokémon Battles are two typical battle games. CoS (Ma et al., 2023) and SwarmBrain (Shao et al., 2024) evaluate LLMs in the strategy game StarCraft II (Vinyals et al., 2017) while PokéLLMon (Hu et al., 2024b) evaluate LLMs in Pokémon. The games we use also belong to tactical battle games. However, we have significant differences from these two games. In these two games, although each team has multiple units, in reality, each team is controlled by only one agent, which is equivalent to a 1v1 setup, and therefore, there is no collaboration between agents in these games. In our game, there can be multiple teams, and each unit of each team is controlled by an agent, which is equivalent to an NvN setup. There are permanent cooperative relationships between agents of the same team. Therefore, we can simultaneously evaluate cooperative and competitive abilities of LLMs.

In addition, many studies also evaluate competitive abilities in board and card games (Chen et al., 2024b), such as Chess (Feng et al., 2024), Texas Hold'em (Huang et al., 2024; Gupta, 2023), and Leduc Hold'em (Zhang et al., 2024c; Guo et al., 2023). In these games, players often have only a competitive relationship. Because the victory in these games typically belongs to only one player. Even in some team-based card games, players within the same team cannot have explicit communication.

Cooperation Games. In cooperative games, players need to work together to finish common goals. Overcooked (Carroll et al., 2019) is commonly used by many works, such as MindAgent (Gong et al., 2023), ProAgent (Zhang et al., 2023a), and LLM-Co (Agashe et al., 2023). Additionally, some works set cooperative objectives in the exploration game Minecraft to evaluate the cooperative capabilities of LLMs, such as S-Agents (Chen et al., 2024a) and HAC (Zhao et al., 2024). However, these works often only involve collaboration without including competition.

Communication Games. Communication games involve multiple players taking turns to engage in various interactions such as communication, negotiation, and even deception. Avalone and Werewolf are commonly used. Specifically, AvaloneBench (Light et al., 2023), CodeAct (Shi et al., 2023), and Recon (Wang et al., 2023) evaluate on Avalone while Xu et al. (2023), Xu et al. (2024c), and Thinker (Wu et al., 2024) evaluate on Werewolf. In addition, Welfare diplomacy (Mukobi et al., 2023), WarAgent (Hua et al., 2023), and CosmoAgent (Jin et al., 2024) evaluate on diplomacy-like games. The core mechanism of these games is communication through dialogue between agents, focusing on evaluating large models' mastery of various dialogue strategies, such as deception strategies and negotiation strategies. Compared to these works, the game we use is more oriented towards tactical battle games, where communication between agents is primarily used to convey cooperative information. Additionally, our games place more emphasis on precise spatial perception capabilities.

10 Evaluation Details

Stage	Turns	Agents	Teams	NPC	Goal
1	60	1	1	0	Navigation
2	60	1	1	10	Navigation
3	80	2	1	10	Cooperative Task
4	80	2	2	10	Competitive Task
5	80	4	2	10	Static Cooperation Competition
6	80	4	4	10	Dynamic Cooperation Competition
7	80	6	3	10	Hybrid Cooperation Competition

Table 6: Stage Settings

10.1 Evaluation Setup

We design 7 stages from easy to difficult for `BattleAgentBench`. Table 6 shows the detailed settings for each stage. The setting of the number of turns takes into account the cost factor of calling LLM. Increasing the number of turns may improve the scores of all models. However, the cost of calling the API will also increase accordingly. Therefore, we have set a relatively reasonable turn number for the unified evaluation of all models.

When evaluating each stage, we run multiple times and calculate the average score of the model to assess each model more accurately. For each run, we generate a random map by randomly sampling the agent's position within a small range. Specifically, for API-based models, we run each stage 3 times; for open-source models, we run each stage 5 times.

10.2 Prompt Template

We use predefined prompt templates to convert game observation data into text prompts, in order to facilitate calling LLM to provide decision results. Figures 1-7 are the prompt templates for each stage. Figure 8-14 shows the Chinese versions of the templates for each stage. Each template consists of 5 parts: game state data, goals, game instructions, operation options, and output format. The main difference is that the goals vary for different stages.

Figure 1: Prompt Template of Level 1 - Stage 1

```
<game>
Current round number:
%d

Our tank position information (ID, horizontal position, vertical position, direction, health):
%s

Base position information (ID, horizontal position, vertical position):
%s

Map information in front of the tank:
%s

Previous round operation information:
%s
</game>

You are an assistant for a tank battle game, helping users control tanks to achieve victory.
Your ultimate goal is to reach the base on the map in the shortest time possible.

#Game instructions:
- The game map size is 512x512, (0,0) represents the top-left corner, (512,512) represents the bottom-right corner.
- In coordinates (x,y), x represents the horizontal position, y represents the vertical position. Moving left decreases x, moving right increases x, moving up decreases y, moving down increases y.
- The map contains tanks and walls. Tanks are 32x32 in size, walls are 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and tanks will block tank movement.
- The vertical movement range for tanks is 0-512, the horizontal movement range is 0-512.
- Tanks have 4 directions: up, down, left, and right. Shooting can destroy tanks or walls in front of the current direction.
- When a tank faces the map boundary, it cannot move forward.
- When there's a wall in front of a tank, it needs to shoot to remove the wall before moving forward.

The current game state is given above. Please provide the next action for the tank based on the game state.
You can execute the following defined operations to control the tank.

#Operation options:
- #Move_up#: Move upward
- #Move_down#: Move downward
- #Move_left#: Move left
- #Move_right#: Move right
- #Shoot#: Shoot

#Note
- You can only output one control operation each time.

```

```
Your output should follow this format:
#Thought process:
- Movement plan: {Based on your position, base position, formulate a movement and shooting plan, and decide the next operation}
#Operation: {Specific operation command}
```

Figure 2: Prompt Template of Level 1 - Stage 2

```
<game>
Current round number:
%d

Our tank position information (ID, horizontal position, vertical position, direction, health):
%s

Base position information (ID, horizontal position, vertical position):
%s

Enemy tank position information (ID, horizontal position, vertical position, direction, health):
%s

Map information in front of the tank:
%s

Previous round operation information:
%s
</game>

You are an assistant for a tank battle game, helping users control tanks to achieve victory.
Your ultimate goal is to reach the base on the map in the shortest time possible. During movement, you can eliminate enemy tanks that threaten your safety.

#Game instructions:
- The game map size is 512x512, (0,0) represents the top-left corner, (512,512) represents the bottom-right corner.
- In coordinates (x,y), x represents the horizontal position, y represents the vertical position. Moving left decreases x, moving right increases x, moving up decreases y, moving down increases y.
- The map contains tanks and walls. Tanks are 32x32 in size, walls are 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and tanks will block tank movement.
- The vertical movement range for tanks is 0-512, the horizontal movement range is 0-512.
- Tanks have 4 directions: up, down, left, and right. Shooting can destroy tanks or walls in front of the current direction.
- When a tank faces the map boundary, it cannot move forward.
- When there's a wall in front of a tank, it needs to shoot to remove the wall before moving forward.

The current game state is given above. Please provide the next action for the tank based on the game state.
You can execute the following defined operations to control the tank.

```

```
#Operation options:
- #Move_up#: Move upward
- #Move_down#: Move downward
- #Move_left#: Move left
- #Move_right#: Move right
- #Shoot#: Shoot

#Note
- You can only output one control operation each time.

Your output should follow this format:
#Thought process:
- Movement plan: {Based on your position, base position, and enemy tank positions, formulate a movement and shooting plan, and decide the next operation}
#Operation: {Specific operation command}
```

Figure 3: Prompt Template of Level 2 - Stage 3

```
<game>
Current round number:
%d

Own tank position information (ID, horizontal position, vertical position, direction, health):
%s

Teammate tank position information (ID, horizontal position, vertical position, direction, health):
%s

Our base position information (ID, horizontal position, vertical position):
%s

Enemy base position information (ID, horizontal position, vertical position):
%s

Enemy tank position information (ID, horizontal position, vertical position, direction, health):
%s

Our tanks' attack target information from the last round (our ID, enemy ID):
%s

Historical cooperation attack information:
%s

Map information around the tank:
%s
</game>

You are an assistant for a tank battle game, helping users control tanks to achieve victory.
Your ultimate goal is to destroy the enemy base while protecting our base from being destroyed by the enemy. Destroying enemy tanks also provides rewards. To achieve the ultimate goal, you can cooperate with your teammate.

```

```
#Game instructions:
- The game map size is 512x512, (0,0) represents the top-left corner, (512,512) represents the bottom-right corner.
- In coordinates (x,y), x represents the horizontal position, y represents the vertical position. Moving left decreases x, moving right increases x, moving up decreases y, moving down increases y.
- The map contains tanks and walls. Tanks are 32x32 in size, walls are 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and tanks will block tank movement.
- The vertical movement range for tanks is 0-512, the horizontal movement range is 0-512.
- Tanks have 4 orientations: up, down, left, and right. Shooting can destroy tanks or walls in front of the current direction.
- When a tank faces the map boundary, it cannot move forward.
- When there's a wall in front of a tank, it needs to shoot to remove the wall before moving forward.

The current game state is given above. Please provide the next action for the tank based on the game state.
You can execute the following defined operations to control the tank. You can also choose cooperation options to decide whether to cooperate with teammates.

#Operation options:
- #Move_up#: Move upward
- #Move_down#: Move downward
- #Move_left#: Move left
- #Move_right#: Move right
- #Shoot#: Shoot

#Cooperation options:
- #Request_coop# {Teammate tank ID x}: {Message content}: Send a cooperation message to the tank with ID x
- #Keep_coop#: Maintain cooperation
- #Stop_coop#: Terminate cooperation
- #No_coop#: No cooperation needed

#Note
- When blocked by an enemy tank, shoot immediately to eliminate the enemy.
- For attack effectiveness, don't frequently change attack targets without new emergencies, as this will cause many ineffective movements.
- You can only output one control operation and one cooperation operation each time.

#Last round operation:
- Operation: %s
- Operation feedback: %s

Your output should follow this format:
#Thought process:
- Attack target: {Reason for choosing an attack target, can continue attacking the current target, or choose a new target based on the game state}
- Attack plan: {Based on your position and the attack target's position, make a movement and shooting plan, and decide the next operation}
```

```
- Cooperation plan: {Based on your position, teammate's position, and attack target's position, decide on a cooperation plan. You can maintain the previous cooperation plan, initiate a new cooperation request, or terminate the previous cooperation plan}
#Attack operation: Target {Enemy tank ID}: {Specific operation command}
#Cooperation operation: {Specific cooperation command}
```

Figure 4: Prompt Template of Level 2 - Stage 4

```
<game>
Current round number:
%d

Our tank position information (ID, horizontal position, vertical position, direction, health):
%s

Our base position information (ID, horizontal position, vertical position):
%s

Enemy base position information (ID, horizontal position, vertical position):
%s

Enemy tank position information (ID, horizontal position, vertical position, direction, health):
%s

Map information around the tank:
%s
</game>

You are an assistant for a tank battle game, helping users control tanks to achieve victory.
Your ultimate goal is to destroy the enemy base while protecting our base from being destroyed by the enemy. Destroying enemy tanks also provides rewards.

#Game instructions:
- The game map size is 512x512, (0,0) represents the top-left corner, (512,512) represents the bottom-right corner.
- In coordinates (x,y), x represents the horizontal position, y represents the vertical position. Moving left decreases x, moving right increases x, moving up decreases y, moving down increases y.
- The map contains tanks and walls. Tanks are 32x32 in size, walls are 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and tanks will block tank movement.
- The vertical movement range for tanks is 0-512, the horizontal movement range is 0-512.
- Tanks have 4 orientations: up, down, left, and right. Shooting can destroy tanks or walls in front of the current direction.
- When a tank faces the map boundary, it cannot move forward.
- When there's a wall in front of a tank, it needs to shoot to remove the wall before moving forward.

The current game state is given above. Please provide the next action for the tank based on the game state.
You can execute the following defined operations to control the tank.

```

```
#Operation options:
- #Move_up#: Move upward
- #Move_down#: Move downward
- #Move_left#: Move left
- #Move_right#: Move right
- #Shoot#: Shoot

#Note
- When blocked by an enemy tank, shoot immediately to eliminate the enemy.
- For attack effectiveness, don't frequently change attack targets without new emergencies, as this will cause many ineffective movements.
- You can only output one control operation each time.

#Last round operation:
- Operation: %s
- Operation feedback: %s

Your output should follow this format:
#Thought process:
- Attack target: {Reason for choosing an attack target, can continue attacking the current target, or choose a new target based on the game state}
- Attack plan: {Based on your position and the attack target's position, make a movement and shooting plan, and decide the next operation}
#Operation: Target {Enemy tank ID}: {Specific operation command}
```

Figure 5: Prompt Template of Level 3 - Stage 5

```
<game>
Current round number:
%d

Own tank position information (ID, horizontal position, vertical position, direction, health):
%s

Teammate tank position information (ID, horizontal position, vertical position, direction, health):
%s

Our base position information (ID, horizontal position, vertical position):
%s

Enemy base position information (ID, horizontal position, vertical position):
%s

Enemy tank position information (ID, horizontal position, vertical position, direction, health):
%s

Our tanks' attack target information from the last round (our ID, enemy ID):
%s

Historical cooperation attack information:
%s

```

```
Map information around the tank:
%s
</game>

You are an assistant for a tank battle game, helping users control tanks to achieve victory.
Your ultimate goal is to destroy the enemy base while protecting our base from being destroyed by the enemy. Destroying enemy tanks also provides rewards. To achieve the ultimate goal, you can cooperate with your teammate.

#Game instructions:
- The game map size is 512x512, (0,0) represents the top-left corner, (512,512) represents the bottom-right corner.
- In coordinates (x,y), x represents the horizontal position, y represents the vertical position. Moving left decreases x, moving right increases x, moving up decreases y, moving down increases y.
- The map contains tanks and walls. Tanks are 32x32 in size, walls are 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and tanks will block tank movement.
- The vertical movement range for tanks is 0-512, the horizontal movement range is 0-512.
- Tanks have 4 orientations: up, down, left, and right. Shooting can destroy tanks or walls in front of the current direction.
- When a tank faces the map boundary, it cannot move forward.
- When there's a wall in front of a tank, it needs to shoot to remove the wall before moving forward.

The current game state is given above. Please provide the next action for the tank based on the game state.
You can execute the following defined operations to control the tank. You can also choose cooperation options to decide whether to cooperate with teammates.

#Operation options:
- #Move_up#: Move upward
- #Move_down#: Move downward
- #Move_left#: Move left
- #Move_right#: Move right
- #Shoot#: Shoot

#Cooperation options:
- #Request_coop# {Teammate tank ID x}: {Message content}: Send a cooperation message to the tank with ID x
- #Keep_coop#: Maintain cooperation
- #Stop_coop#: Terminate cooperation
- #No_coop#: No cooperation needed

#Note
- When blocked by an enemy tank, shoot immediately to eliminate the enemy.
- For attack effectiveness, don't frequently change attack targets without new emergencies, as this will cause many ineffective movements.
- You can only output one control operation and one cooperation operation each time.

#Last round operation:
- Operation: %s
- Operation feedback: %s
```

```

Your output should follow this format:
#Thought process:
- Attack target: {Reason for choosing an attack target, can continue attacking the current target, or choose a new target based on the game state}
- Attack plan: {Based on your position and the attack target's position, make a movement and shooting plan, and decide the next operation}
- Cooperation plan: {Based on your position, teammate's position, and attack target's position, decide on a cooperation plan. You can maintain the previous cooperation plan, initiate a new cooperation request, or terminate the previous cooperation plan}
#Attack operation: Target {Enemy tank ID}: {Specific operation command}
#Cooperation operation: {Specific cooperation command}
```

Figure 6: Prompt Template of Level 3 - Stage 6

```
<game>
Current round number:
%d

Our tank position information (ID, horizontal position, vertical position, direction, health, type):
%s

Our base position information (ID, horizontal position, vertical position):
%s

Enemy base position information (ID, horizontal position, vertical position):
%s

Enemy tank position information (ID, horizontal position, vertical position, direction, health, type):
%s

Our tanks' attack target information from the last round (our ID, enemy ID):
%s

Historical cooperation attack information:
%s

Map information around the tank:
%s
</game>

You are an assistant for a tank battle game, helping users control tanks to achieve victory.
Your ultimate goal is to destroy the enemy base while protecting our base from being destroyed by the enemy. Destroying enemy tanks also provides rewards. To achieve the ultimate goal, you can cooperate with an enemy to eliminate other enemies.

#Game instructions:
- The game map size is 512x512, (0,0) represents the top-left corner, (512,512) represents the bottom-right corner.
```

```
- In coordinates (x,y), x represents the horizontal position, y represents the vertical position. Moving left decreases x, moving right increases x, moving up decreases y, moving down increases y.
- The map contains tanks and walls. Tanks are 32x32 in size, walls are 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and tanks will block tank movement.
- The vertical movement range for tanks is 0-512, the horizontal movement range is 0-512.
- Tanks have 4 orientations: up, down, left, and right. Shooting can destroy tanks or walls in front of the current direction.
- When a tank faces the map boundary, it cannot move forward.
- When there's a wall in front of a tank, it needs to shoot to remove the wall before moving forward.
- Tanks have two types: normal and advanced. Only advanced tanks have cooperation capabilities.

The current game state is given above. Please provide the next action for the tank based on the game state.
You can execute the following defined operations to control the tank. You can also choose cooperation options to decide whether to cooperate with teammates.

#Operation options:
- #Move_up#: Move upward
- #Move_down#: Move downward
- #Move_left#: Move left
- #Move_right#: Move right
- #Shoot#: Shoot

#Cooperation options:
- #Request_coop# {Tank ID x}: {Message content}: Send a cooperation message to the tank with ID x
- #Keep_coop#: Maintain cooperation
- #Stop_coop#: Terminate cooperation
- #No_coop#: No cooperation needed

#Note
- When blocked by an enemy tank, shoot immediately to eliminate the enemy.
- For attack effectiveness, don't frequently change attack targets without new emergencies, as this will cause many ineffective movements.
- You can only output one control operation and one cooperation operation each time.

#Last round operation:
- Operation: %s
- Operation feedback: %s

Your output should follow this format:
#Thought process:
- Attack target: {Reason for choosing an attack target, can continue attacking the current target, or choose a new target based on the game state}
- Attack plan: {Based on your position and the attack target's position, make a movement and shooting plan, and decide the next operation}
- Cooperation plan: {Based on your position and the attack target's position, decide on a cooperation plan. You can maintain the previous round's cooperation plan, initiate a new cooperation request, or terminate the previous round's cooperation plan}
#Attack operation: Target {Enemy tank ID}: {Specific operation command}
#Cooperation operation: {Specific cooperation command}
```

Figure 7: Prompt Template of Level 3 - Stage 7

```
<game>
Current round number:
%d

Own tank position information (ID, horizontal position, vertical position, direction, health, type):
%s

Teammate tank position information (ID, horizontal position, vertical position, direction, health, type):
%s

Our base position information (ID, horizontal position, vertical position):
%s

Enemy base position information (ID, horizontal position, vertical position):
%s

Enemy tank position information (ID, horizontal position, vertical position, direction, health, type):
%s

Our tanks' attack target information from the last round (our ID, enemy ID):
%s

Historical cooperation attack information:
%s

Map information around the tank:
%s
</game>

You are an assistant for a tank battle game, helping users control tanks to achieve victory.
Your ultimate goal is to destroy the enemy base while protecting our base from being destroyed by the enemy. Destroying enemy tanks also provides rewards. To achieve the ultimate goal, you can cooperate with your teammates, or temporarily cooperate with an enemy to eliminate other enemies.

#Game instructions:
- The game map size is 512x512, (0,0) represents the top-left corner, (512,512) represents the bottom-right corner.
- In coordinates (x,y), x represents the horizontal position, y represents the vertical position. Moving left decreases x, moving right increases x, moving up decreases y, moving down increases y.
- The map contains tanks and walls. Tanks are 32x32 in size, walls are 8x8.
- Tanks can move in four directions: up, down, left, and right. Walls and tanks will block tank movement.
- The vertical movement range for tanks is 0-512, the horizontal movement range is 0-512.
- Tanks have 4 orientations: up, down, left, and right. Shooting can destroy tanks or walls in front of the current direction.
```

```
- When a tank faces the map boundary, it cannot move forward.
- When there's a wall in front of a tank, it needs to shoot to remove the wall before moving forward.
- Tanks have two types: normal and advanced. Only advanced tanks have cooperation capabilities.

The current game state is given above. Please provide the next action for the tank based on the game state.
You can execute the following defined operations to control the tank. You can also choose cooperation options to decide whether to cooperate with teammates.

#Operation options:
- #Move_up#: Move upward
- #Move_down#: Move downward
- #Move_left#: Move left
- #Move_right#: Move right
- #Shoot#: Shoot

#Cooperation options:
- #Request_coop# {Tank ID x}: {Message content}: Send a cooperation message to the tank with ID x
- #Keep_coop#: Maintain cooperation
- #Stop_coop#: Terminate cooperation
- #No_coop#: No cooperation needed

#Note
- When blocked by an enemy tank, shoot immediately to eliminate the enemy.
- For attack effectiveness, don't frequently change attack targets without new emergencies, as this will cause many ineffective movements.
- You can only output one control operation and one cooperation operation each time.

#Last round operation:
- Operation: %s
- Operation feedback: %s

Your output should follow this format:
#Thought process:
- Attack target: {Reason for choosing an attack target, can continue attacking the current target, or choose a new target based on the game state}
- Attack plan: {Based on your position and the attack target's position, make a movement and shooting plan, and decide the next operation}
- Cooperation plan: {Based on your position and the attack target's position, decide on a cooperation plan. You can maintain the previous round's cooperation plan, initiate a new cooperation request, or terminate the previous round's cooperation plan}
#Attack operation: Target {Enemy tank ID}: {Specific operation command}
#Cooperation operation: {Specific cooperation command}
```

Figure 8: Prompt Template of Level 1 - Stage 1 in Chinese

```
<game>
当前会合数:
%d

我方坦克位置信息（编号，水平位置，垂直位置，朝向，血量）:
```

```
%s

基地位置信息（编号，水平位置，垂直位置）：
%s

坦克前方地图信息:
%s

上一会合操作信息:

</game>

你是一个坦克对战游戏的助手，可以帮助用户在游戏中控制坦克取得胜利。
你的最终目标是以最短时间到达地图中的基地。

#游戏说明:
- 游戏地图大小为512x512，(0，0)表示左上角，(512，512)表示右下角。
- 坐标(x，y)中x表示水平位置，y表示垂直位置，向左移动x减小，向右移动x增大，向上移动y减小，向下移动y增大。
- 地图中有坦克及wall，坦克大小为32x32，wall大小为8x8。
- 坦克可以向上、下、左、右四个方向移动，wall和坦克会阻挡坦克的移动。
- 坦克上下的移动范围为0-512，左右的移动范围0-512。
- 坦克有4个朝向，上、下、左、右，射击可以销毁当前方向前方的坦克或wall。
- 坦克前方为地图边界时，无法继续向前移动。
- 坦克前方有wall时，需要射击消除wall之后才能继续向前移动。

上面给出了当前游戏的状态，请根据游戏状态，给出坦克的下一步操作。
你可以执行以下定义的操作来控制坦克。

#操作选项:
- #Move_up#: 向上移动
- #Move_down#: 向下移动
- #Move_left#: 向左移动
- #Move_right#: 向右移动
- #Shoot#: 射击

#注意
- 你每次只能输出一个操作。

你的输出应该符合以下格式:
#思考过程:
- 移动计划: {根据自己的位置和基地位置，制订移动和射击计划，并决定下一步的操作}
#操作: {具体操作指令}
```

Figure 9: Prompt Template of Level 1 - Stage 2 in Chinese

```
<game>
当前会合数:
%d

我方坦克位置信息（编号，水平位置，垂直位置，朝向，血量）：
%s
```

```

基地位置信息（编号，水平位置，垂直位置）：
%s

敌方坦克位置信息（编号，水平位置，垂直位置，朝向，血量）：
%s

坦克前方地图信息:
%s

上一会合操作信息:
%s
</game>

你是一个坦克对战游戏的助手，可以帮助用户在游戏中控制坦克取得胜利。
你的最终目标是以最短时间到达地图中的基地。移动过程中你可以消灭影响你安全的敌方坦克。

#游戏说明:
- 游戏地图大小为512x512，(0，0)表示左上角，（512，512）表示右下角。
- 坐标（x，y）中x表示水平位置，y表示垂直位置，向左移动x减小，向右移动x增大，向上移动y减小，向下移动y增大。
- 地图中有坦克及wall，坦克大小为32x32，wall大小为8x8。
- 坦克可以向上、下、左、右四个方向移动，wall和坦克会阻挡坦克的移动。
- 坦克上下的移动范围为0-512，左右的移动范围0-512。
- 坦克有4个朝向，上、下、左、右，射击可以销毁当前方向前方的坦克或wall。
- 坦克前方为地图边界时，无法继续向前移动。
- 坦克前方有wall时，需要射击消除wall之后才能继续向前移动。

上面给出了当前游戏的状态，请根据游戏状态，给出坦克的下一步操作。
你可以执行以下定义的操作来控制坦克。

#操作选项:
- #Move_up#: 向上移动
- #Move_down#: 向下移动
- #Move_left#: 向左移动
- #Move_right#: 向右移动
- #Shoot#: 射击

#注意
- 你每次只能输出一个操作。

你的输出应该符合以下格式:
#思考过程:
- 移动计划: {根据自己的位置和基地位置，以及敌方坦克的位置，制订移动和射击计划，并决定下一步的操作}
#操作: {具体操作指令}
```

Figure 10: Prompt Template of Level 2 - Stage 3 in Chinese

```
<game>
当前回合数:
%d

```

```
自己坦克位置信息（编号，水平位置，垂直位置，朝向，血量）：
%s

队友坦克位置信息（编号，水平位置，垂直位置，朝向，血量）：
%s

我方基地位置信息（编号，水平位置，垂直位置）：
%s

敌方基地位置信息（编号，水平位置，垂直位置）：
%s

敌方坦克位置信息（编号，水平位置，垂直位置，朝向，血量）：
%s

我方坦克上一回合攻击目标信息（我方编号，敌方编号）：
%s

历史协作攻击信息:
%s

坦克周围地图信息:
%s
</game>

你是一个坦克对战游戏的助手，可以帮助用户在游戏中控制坦克取得胜利。
你的最终目标是消灭敌方基地，并保护我方基地不被敌人消灭。消灭敌方坦克也有奖励。你可以与你的队友协作完成目标。

#游戏说明:
- 游戏地图大小为512x512，(0，0)表示左上角，（512，512）表示右下角。
- 坐标（x，y）中x表示水平位置，y表示垂直位置，向左移动x减小，向右移动x增大，向上移动y减小，向下移动y增大。
- 地图中有坦克及wall，坦克大小为32x32，wall大小为8x8。
- 坦克可以向上、下、左、右四个方向移动，wall和坦克会阻挡坦克的移动。
- 坦克上下的移动范围为0-512，左右的移动范围0-512。
- 坦克有4个朝向，上、下、左、右，射击可以销毁当前方向前方的坦克或wall。
- 坦克前方为地图边界时，无法继续向前移动。
- 坦克前方有wall时，需要射击消除wall之后才能继续向前移动。

上面给出了当前游戏的状态，请根据游戏状态，给出坦克的下一步操作。
你可以执行以下定义的操作来控制坦克。并可以选择协作选项决定是否与队友协作攻击。

#操作选项:
- #Move_up#: 向上移动
- #Move_down#: 向下移动
- #Move_left#: 向左移动
- #Move_right#: 向右移动
- #Shoot#: 射击

#协作选项:
- #Request_coop# {队友坦克编号x}: {消息内容}: 向编号为x的坦克发送协作消息
- #Keep_coop#: 保持协作
```

```
- #Stop_coop#: 终止协作
- #No_coop#: 无需协作

#注意
- 当阻挡自己的为敌方坦克时，立即射击消灭敌方。
- 为了攻击的有效性，没有新的突发情况，不要频繁地更换攻击目标，这样会造成很多无效移动。
- 你每次只能输出一个控制操作和一个协作操作。

#上一回合操作:
- 操作: %s
- 操作反馈: %s

你的输出应该符合以下格式:
#思考过程:
- 攻击目标: {选择某个攻击目标的原因，可以继续攻击当前目标，或者根据游戏状态选择新的攻击目标}
- 攻击计划: {根据自己的位置和攻击目标的位置，制订移动和射击计划，并决定下一步的操作}
- 协作计划: {根据自己的位置、队友的位置和攻击目标的位置，决定协作计划，可以保持上一会合的协作计划，也可发起新的协作请求，也可终止上一会合协作计划}
#攻击操作: Target {敌方坦克编号}: {具体操作指令}
#协作操作: {具体协作指令}
```

Figure 11: Prompt Template of Level 2 - Stage 4 in Chinese

```
<game>
当前回合数:
%d

我方坦克位置信息（编号，水平位置，垂直位置，朝向，血量）:
%s

我方基地位置信息（编号，水平位置，垂直位置）:
%s

敌方基地位置信息（编号，水平位置，垂直位置）:
%s

敌方坦克位置信息（编号，水平位置，垂直位置，朝向，血量）:
%s

坦克周围地图信息:
%s
</game>

你是一个坦克对战游戏的助手，可以帮助用户在游戏中控制坦克取得胜利。
你的最终目标是消灭敌方基地，并保护我方基地不被敌人消灭。消灭敌方坦克也有奖励。

#游戏说明:
- 游戏地图大小为512x512，(0，0)表示左上角，（512，512）表示右下角。
- 坐标（x，y）中x表示水平位置，y表示垂直位置，向左移动x减小，向右移动x增大，向上移动y减小，向下移动y增大。
- 地图中有坦克及wall，坦克大小为32x32，wall大小为8x8。
- 坦克可以向上、下、左、右四个方向移动，wall和坦克会阻挡坦克的移动。
```

```
- 坦克上下的移动范围为0-512，左右的移动范围0-512。
- 坦克有4个朝向，上、下、左、右，射击可以销毁当前方向前方的坦克或wall。
- 坦克前方为地图边界时，无法继续向前移动。
- 坦克前方有wall时，需要射击消除wall之后才能继续向前移动。

上面给出了当前游戏的状态，请根据游戏状态，给出坦克的下一步操作。
你可以执行以下定义的操作来控制坦克。

#操作选项:
- #Move_up#: 向上移动
- #Move_down#: 向下移动
- #Move_left#: 向左移动
- #Move_right#: 向右移动
- #Shoot#: 射击

#注意
- 当阻挡自己的为敌方坦克时，立即射击消灭敌方。
- 为了攻击的有效性，没有新的突发情况，不要频繁地更换攻击目标，这样会造成很多无效移动。
- 你每次只能输出一个操作。

#上一回合操作:
- 操作: %s
- 操作反馈: %s

你的输出应该符合以下格式:
#思考过程:
- 攻击目标: {选择某个攻击目标的原因，可以继续攻击当前目标，或者根据游戏状态选择新的攻击目标}
- 攻击计划: {根据自己的位置和攻击目标的位置，制订移动和射击计划，并决定下一步的操作}
#操作: Target {敌方坦克编号}: {具体操作指令}
```

Figure 12: Prompt Template of Level 3 - Stage 5 in Chinese

```
<game>
当前回合数:
%d

自己坦克位置信息（编号，水平位置，垂直位置，朝向，血量）:
%s

队友坦克位置信息（编号，水平位置，垂直位置，朝向，血量）:
%s

我方基地位置信息（编号，水平位置，垂直位置）:
%s

敌方基地位置信息（编号，水平位置，垂直位置）:
%s

敌方坦克位置信息（编号，水平位置，垂直位置，朝向，血量）:
%s

我方坦克上一回合攻击目标信息（我方编号，敌方编号）:
```

```
%s

历史协作攻击信息:
%s

坦克周围地图信息:
%s
</game>

你是一个坦克对战游戏的助手，可以帮助用户在游戏中控制坦克取得胜利。
你的最终目标是消灭敌方基地，并保护我方基地不被敌人消灭。消灭敌方坦克也有奖励。你可以与你的队友协作完成目标。

#游戏说明:
- 游戏地图大小为512x512，(0，0)表示左上角，（512，512）表示右下角。
- 坐标（x，y）中x表示水平位置，y表示垂直位置，向左移动x减小，向右移动x增大，向上移动y减小，向下移动y增大。
- 地图中有坦克及wall，坦克大小为32x32，wall大小为8x8。
- 坦克可以向上、下、左、右四个方向移动，wall和坦克会阻挡坦克的移动。
- 坦克上下的移动范围为0-512，左右的移动范围0-512。
- 坦克有4个朝向，上、下、左、右，射击可以销毁当前方向前方的坦克或wall。
- 坦克前方为地图边界时，无法继续向前移动。
- 坦克前方有wall时，需要射击消除wall之后才能继续向前移动。

上面给出了当前游戏的状态，请根据游戏状态，给出坦克的下一步操作。
你可以执行以下定义的操作来控制坦克。并可以选择协作选项决定是否与队友协作攻击。

#操作选项:
- #Move_up#: 向上移动
- #Move_down#: 向下移动
- #Move_left#: 向左移动
- #Move_right#: 向右移动
- #Shoot#: 射击

#协作选项:
- #Request_coop# {队友坦克编号x}: {消息内容}: 向编号为x的坦克发送协作消息
- #Keep_coop#: 保持协作
- #Stop_coop#: 终止协作
- #No_coop#: 无需协作

#注意
- 当阻挡自己的为敌方坦克时，立即射击消灭敌方。
- 为了攻击的有效性，没有新的突发情况，不要频繁地更换攻击目标，这样会造成很多无效移动。
- 你每次只能输出一个控制操作和一个协作操作。

#上一回合操作:
- 操作: %s
- 操作反馈: %s

你的输出应该符合以下格式:
#思考过程:
- 攻击目标: {选择某个攻击目标的原因，可以继续攻击当前目标，或者根据游戏状态选择新的攻击目标}
- 攻击计划: {根据自己的位置和攻击目标的位置，制订移动和射击计划，并决定下一步的操作}
```

```
- 协作计划：{根据自己的位置、队友的位置和攻击目标的位置，决定协作计划，可以保持上一会合的协作计划，也可发起新的协作请求，也可终止上一会合协作计划}
#攻击操作：Target {敌方坦克编号}：{具体操作指令}
#协作操作：{具体协作指令}
```

Figure 13: Prompt Template of Level 3 - Stage 6 in Chinese

```
<game>
当前回合数：
%d

我方坦克位置信息（编号，水平位置，垂直位置，朝向，血量，类型）：
%s

我方基地位置信息（编号，水平位置，垂直位置）：
%s

敌方基地位置信息（编号，水平位置，垂直位置）：
%s

敌方坦克位置信息（编号，水平位置，垂直位置，朝向，血量，类型）：
%s

我方坦克上一回合攻击目标信息（我方编号，敌方编号）：
%s

历史协作攻击信息：
%s

坦克周围地图信息：
%s
</game>

你是一个坦克对战游戏的助手，可以帮助用户在游戏中控制坦克取得胜利。
你的最终目标是消灭敌方基地，并保护我方基地不被敌人消灭。消灭敌方坦克也有奖励。为了完成最终目标，你可以与某个敌人协作消灭其它敌人。

#游戏说明：
- 游戏地图大小为512x512，(0，0)表示左上角，（512，512）表示右下角。
- 坐标（x，y）中x表示水平位置，y表示垂直位置，向左移动x减小，向右移动x增大，向上移动y减小，向下移动y增大。
- 地图中有坦克及wall，坦克大小为32x32，wall大小为8x8。
- 坦克可以向上、下、左、右四个方向移动，wall和坦克会阻挡坦克的移动。
- 坦克上下的移动范围为0-512，左右的移动范围0-512。
- 坦克有4个朝向，上、下、左、右，射击可以销毁当前方向前方的坦克或wall。
- 坦克前方为地图边界时，无法继续向前移动。
- 坦克前方有wall时，需要射击消除wall之后才能继续向前移动。
- 坦克有普通和高级两种类型，只有高级坦克具有协作能力。

上面给出了当前游戏的状态，请根据游戏状态，给出坦克的下一步操作。
你可以执行以下定义的操作来控制坦克。并可以选择协作选项决定是否与队友协作攻击。

```

```
#操作选项:
- #Move_up#: 向上移动
- #Move_down#: 向下移动
- #Move_left#: 向左移动
- #Move_right#: 向右移动
- #Shoot#: 射击

#协作选项:
- #Request_coop# {坦克编号x}: {消息内容}: 向编号为x的坦克发送协作消息
- #Keep_coop#: 保持协作
- #Stop_coop#: 终止协作
- #No_coop#: 无需协作

#注意
- 当阻挡自己的为敌方坦克时，立即射击消灭敌方。
- 为了攻击的有效性，没有新的突发情况，不要频繁地更换攻击目标，这样会造成很多无效移动。
- 你每次只能输出一个控制操作和一个协作操作。

#上一回合操作:
- 操作: %s
- 操作反馈: %s

你的输出应该符合以下格式:
#思考过程:
- 攻击目标: {选择某个攻击目标的原因，可以继续攻击当前目标，或者根据游戏状态选择新的攻击目标}
- 攻击计划: {根据自己的位置和攻击目标的位置，制订移动和射击计划，并决定下一步的操作}
- 协作计划: {根据自己的位置、攻击目标的位置，决定协作计划，可以保持上一会合的协作计划，也可发起新
↪ 的协作请求，也可终止上一会合协作计划}
#攻击操作: Target {敌方坦克编号}: {具体操作指令}
#协作操作: {具体协作指令}
```

Figure 14: Prompt Template of Level 3 - Stage 7 in Chinese

```
<game>
当前回合数:
%d

自己坦克位置信息（编号，水平位置，垂直位置，朝向，血量，类型）:
%s

队友坦克位置信息（编号，水平位置，垂直位置，朝向，血量，类型）:
%s

我方基地位置信息（编号，水平位置，垂直位置）:
%s

敌方基地位置信息（编号，水平位置，垂直位置）:
%s

敌方坦克位置信息（编号，水平位置，垂直位置，朝向，血量，类型）:
%s

```

```
我方坦克上一回合攻击目标信息（我方编号，敌方编号）：
%s

历史协作攻击信息:
%s

坦克周围地图信息:
%s
</game>

你是一个坦克对战游戏的助手，可以帮助用户在游戏中控制坦克取得胜利。
你的最终目标是消灭敌方基地，并保护我方基地不被敌人消灭。消灭敌方坦克也有奖励。为了完成最终目标，你可以与你的队友协作，也可以暂时与某个敌人协作消灭其它敌人。

#游戏说明:
- 游戏地图大小为512x512，(0，0)表示左上角，（512，512）表示右下角。
- 坐标（x，y）中x表示水平位置，y表示垂直位置，向左移动x减小，向右移动x增大，向上移动y减小，向下移动y增大。
- 地图中有坦克及wall，坦克大小为32x32，wall大小为8x8。
- 坦克可以向上、下、左、右四个方向移动，wall和坦克会阻挡坦克的移动。
- 坦克上下的移动范围为0-512，左右的移动范围0-512。
- 坦克有4个朝向，上、下、左、右，射击可以销毁当前方向前方的坦克或wall。
- 坦克前方为地图边界时，无法继续向前移动。
- 坦克前方有wall时，需要射击消除wall之后才能继续向前移动。
- 坦克有普通和高级两种类型，只有高级坦克具有协作能力。

上面给出了当前游戏的状态，请根据游戏状态，给出坦克的下一步操作。
你可以执行以下定义的操作来控制坦克。并可以选择协作选项决定是否与队友协作攻击。

#操作选项:
- #Move_up#: 向上移动
- #Move_down#: 向下移动
- #Move_left#: 向左移动
- #Move_right#: 向右移动
- #Shoot#: 射击

#协作选项:
- #Request_coop# {坦克编号x}: {消息内容}: 向编号为x的坦克发送协作消息
- #Keep_coop#: 保持协作
- #Stop_coop#: 终止协作
- #No_coop#: 无需协作

#注意
- 当阻挡自己的为敌方坦克时，立即射击消灭敌方。
- 为了攻击的有效性，没有新的突发情况，不要频繁地更换攻击目标，这样会造成很多无效移动。
- 你每次只能输出一个控制操作和一个协作操作。

#上一回合操作:
- 操作: %s
- 操作反馈: %s

你的输出应该符合以下格式:
#思考过程:
```

```
- 攻击目标：{选择某个攻击目标的原因，可以继续攻击当前目标，或者根据游戏状态选择新的攻击目标}
- 攻击计划：{根据自己的位置和攻击目标的位置，制订移动和射击计划，并决定下一步的操作}
- 协作计划：{根据自己的位置、攻击目标的位置，决定协作计划，可以保持上一会合的协作计划，也可发起新的协作请求，也可终止上一会合协作计划}
#攻击操作：Target {敌方坦克编号}：{具体操作指令}
#协作操作：{具体协作指令}
```