MINDAGENT: EMERGENT GAMING INTERACTION

Ran Gong[1†*], **Qiuyuan Huang**[2‡*], **Xiaojian Ma**[1*], **Hoi Vo**[3], **Zane Durante**[4†], **Yusuke Noda**[3], **Zilong Zheng**[5], **Song-Chun Zhu**[1567], **Demetri Terzopoulos**[1], **Li Fei-Fei**[4], **Jianfeng Gao**[2]
[1]UCLA; [2]Microsoft Research, Redmond; [3]Xbox Team, Microsoft; [4]Stanford;[5]BIGAI; [6]PKU; [7]THU

Figure 1: The MindAgent system for gaming interactions. MindAgent enables complex task planning in a multi-agent system and provides a human-AI collaboration infrastructure across various domains.

ABSTRACT

Large Language Models (LLMs) can perform complex scheduling in a multi-agent system and can coordinate agents to complete sophisticated tasks that require extensive collaboration. However, despite the introduction of numerous gaming frameworks, the community lacks adequate benchmarks that support the implementation of a general multi-agent infrastructure encompassing collaboration between LLMs and human-NPCs. We propose a novel infrastructure—**MindAgent**—for evaluating planning and coordination capabilities in the context of gaming interaction. In particular, our infrastructure leverages an existing gaming framework to (i) require understanding of the coordinator for a multi-agent system, (ii) collaborate with human players via instructions, and (iii) enable in-context learning based on few-shot prompting with feedback. Furthermore, we introduce **CuisineWorld**, a new gaming scenario and its related benchmark that supervises multiple agents playing the game simultaneously and measures multi-agent collaboration efficiency. We have conducted comprehensive evaluations with a new auto-metric *collaboration score* **CoS** for assessing the collaboration efficiency. Finally, MindAgent can be deployed in real-world gaming scenarios in a customized VR version of CuisineWorld and adapted in the broader "Minecraft" gaming domain as showed in Figure 1. Our work involving LLMs within our new infrastructure for general-purpose scheduling and coordination can elucidate how such skills may be obtained by learning from large language corpora.

∗ Equal Contribution. ‡ Project Lead.
† Work done while Ran and Zane interning at Microsoft Research, Redmond.

1 INTRODUCTION

Large language Models (LLMs) have been driving the effort to develop general intelligent machines (Bubeck et al., 2023; Mirchandani et al., 2023). Although they are trained using large text corpora, their superior problem-solving capacity is not limited to canonical language processing domains. LLMs can potentially tackle complex tasks that were previously presumed exclusive to human experts or domain-specific algorithms. Recent research has shown the possibility of using LLMs to generate complex plans for robots and game AI (Liang et al., 2022; Wang et al., 2023b;a; Yao et al., 2023; Huang et al., 2023), marking an important milestone for LLMs as general-purpose intelligent agents. In this paper, we investigate the planning capacity of LLMs in the context of multi-agent systems (Stone & Veloso, 2000). Compared to planning for a single agent, which has been studied extensively (Wang et al., 2023b;a), multi-agent planning imposes much higher problem-solving complexity due to an action space that grows exponentially with respect to the number of agents. The planner must simultaneously control multiple agents, avoid possible conflicts, and coordinate agents into achieving a shared goal that requires potentially sophisticated collaboration. To understand to what extent LLMs can acquire multi-agent planning skills, we first develop a new benchmark, **CuisineWorld**, which is illustrated in Figure 1.

To incorporate agent AIs into video games, we design **MindAgent**, an infrastructure inspired by multi-agent task allocation optimization theories, to facilitate the multi-agent planning capabilities of LLMs. Our infrastructure enables LLMs to perform complex coordination and scheduling of multiple agents in order to achieve task completion. We conduct comprehensive evaluations with recently introduced LLMs, including GPT-4, Claude, and LLaMA, playing our CuisineWorld game within our MindAgent interactive multi-agent planning framework, leading to the following key observations: 1) **Zero shot multi-agent planning**: Powerful pretrained LLMs like GPT-4 are capable of scheduling multiple agents (ranging from 2 to 4) to complete dishes, even by collaborating with human players, by merely reading game instructions and recipes; 2) **Planning with advanced prompting**: We can significantly boost multi-agent planning performance by leveraging an emergent *in-context learning* ability (Brown et al., 2020; Wei et al., 2021) by adding only a few expert demonstrations (from different games) to the prompt, explaining the rationale of certain actions as in Chain-of-Thought prompting (Wei et al., 2022), and providing on-the-fly feedback to the LLMs during planning; and 3) **Generalization**: LLMs can potentially be generalist multi-agent planners as they are able to generalize in order to coordinate a growing number of agents and perform well in new game domains such as Minecraft.

The main contributions of our work are as follows:

- We develop a new gaming scenario and related benchmark based on a multi-agent virtual kitchen environment, CuisineWorld. It adopts a minimal text-based game format and supports planning tasks with various structures and challenges, making it an ideal test bed for the emergent multi-agent planning (i.e., scheduling and coordination) capacity of LLMs.

- We introduce MindAgent, an infrastructure for interactive multi-agent planning with LLMs. which demonstrates the in-context learning of the multi-agent planning capacity of LLMs and offers several prompting techniques to facilitate their planning ability, including providing few-shot demonstrations, planning rationals, and environmental feedback.

- We conduct extensive evaluations of our benchmark with multiple LLMs and prompting settings. Our experimental results validate its potential in helping develop generalist multi-agent planners.

- We deploy MindAgent in real-world gaming scenarios and demonstrate its ability to power human-AI interactions.

Compared to canonical domain-specific automated planning systems, although multi-agent planning with LLMs is more likely to be bottlenecked by high computational cost, context length limitations, non-optimal plans, *etc*., it can potentially improve planning performed by *in-context learning* from data without fine-tuning, seamlessly adapt to new planning problems across different domains, and offer a more flexible interface to human collaborators. Ultimately, our investigation into the leveraging of LLMs for general-purpose scheduling and coordination can elucidate how such skills may be acquired by learning from large text corpora, and is potentially instrumental to the future development of more effective LLM-based planners.

2 Related Work

Multi-Agent Coordination. The field of multi-agent collaboration boasts a comprehensive body of literature. Traditionally, such collaborations have been modeled using the MDP/POMDP frameworks (Lowe et al., 2017; Rashid et al., 2020; Jain et al., 2019). However, there has been a recent shift towards using LLMs for these collaborations. For instance, Zhang et al. (2023b) delved into how LLMs might communicate and cooperate in a watch-and-help (WAH) task. Meanwhile, Zhang et al. (2023a) investigated a two-agent collaboration game inspired by the simpler dynamics of the two-agent Overcooked-style game. Notably, their research mainly concentrated on the task success rate, with most studies typically anchored to a single task objective. By contrast, we emphasize the importance of collaboration efficiency in scenarios encompassing multiple task objectives. Further, our research uniquely focuses on evaluating the collaborative efficiency of two or more agents. Additionally, while other works such as that of Park et al. (2023) simulate each agent individually, we employ a centralized system. This approach not only significantly reduces the number of API calls but also reduces context length, making it more appropriate for use in gaming applications.

Planning With LLMs. A number of works leverage LLMs to perform task planning (Huang et al., 2022a; Wang et al., 2023a; Yao et al., 2023; Li et al., 2023), specifically the LLMs' WWW-scale domain knowledge and emergent zero-shot planning abilities to perform complex task planning and reasoning. Recent robotics research also leverages LLMs to perform task planning (Ahn et al., 2022; Huang et al., 2022b; Liang et al., 2022) by decomposing natural language instruction into a sequence of subtasks, either in the natural language form or in Python code , then using a low-level controller to execute these subtasks. Additionally, Huang et al. (2022b), Liang et al. (2022), and Wang et al. (2023b) also incorporate environmental feedback to improve task performance.

Benchmarks Using Games. Numerous games have been developed to study task planning (Baker et al., 2022; Carroll et al., 2019; Bakhtin et al., 2022), yet only a handful delve into multi-agent collaborations. Even within this limited subset, the focus predominantly remains on two-agent interactions where responsibilities are unevenly distributed between the agents (Wan et al., 2022; Puig et al., 2020)—it is common for one player to assume a dominant role while the other provides support. By contrast, our work assumes the equal apportion of responsibilities across agents, and we expand our investigation to encompass collaborations involving more than two agents, even including human players. While some previous studies have ventured into multi-task settings, none has delved into scenarios where agents must compete for resources to complete multiple distinct tasks with varied levels of difficulty within a single episode. Additionally, our work differs from that of Carroll et al. (2019) in that our game settings feature a diverse array of tools and task objectives, thereby generating an exponentially larger task space. A comparison between our work and other related studies can be found in the Appendix D.1.

3 The CuisineWorld Game

We introduce CuisineWorld as a novel and flexible game for multi-agent scheduling and coordination in a *virtual kitchen* environment. In this game, a multi-agent system must supervise multiple agents and coordinate them, with the goal of completing as many dish orders as possible. The game is equipped with a textual interface since our focus is on evaluating LLM-based planning agents. Our modularized design separates tasks and game engines, allowing inclusion of more tasks (dish types) and domains ("kitchen" implementation via text-based engine, Unity, Minecraft, *etc.*).

Tasks and Reward. A task in CuisineWorld is a dish order, ranging from the most basic `tunaSashimi`, which can be made by simply chopping raw tuna meat, to sophisticated dishes like `porkPasta` requiring various cooking tools. In a game episode with a maximum of T steps, in every *task interval* τ_{int}, a new task or dish order will be added to the active task list. A task will be regarded *completed* and be removed from the active task list when a suitable dish has been placed on the serving table. On the contrary, a task will be deemed to have *failed* and be removed from the list after its *lifetime* τ_{lft}, which depends on the complexity of the dish, is exceeded. Along with the tasks, the game provides rewards and penalties or feedback on certain occasions, *e.g.* when a task is just completed, when infeasible commands are dispatched, *etc*. We support five different actions 1) goto 2) get 3) put 4) activate 5) noop. The state space contains descriptions of the environment and agents. Due to space limitations, we refer the reader to additional details in Appendix C.

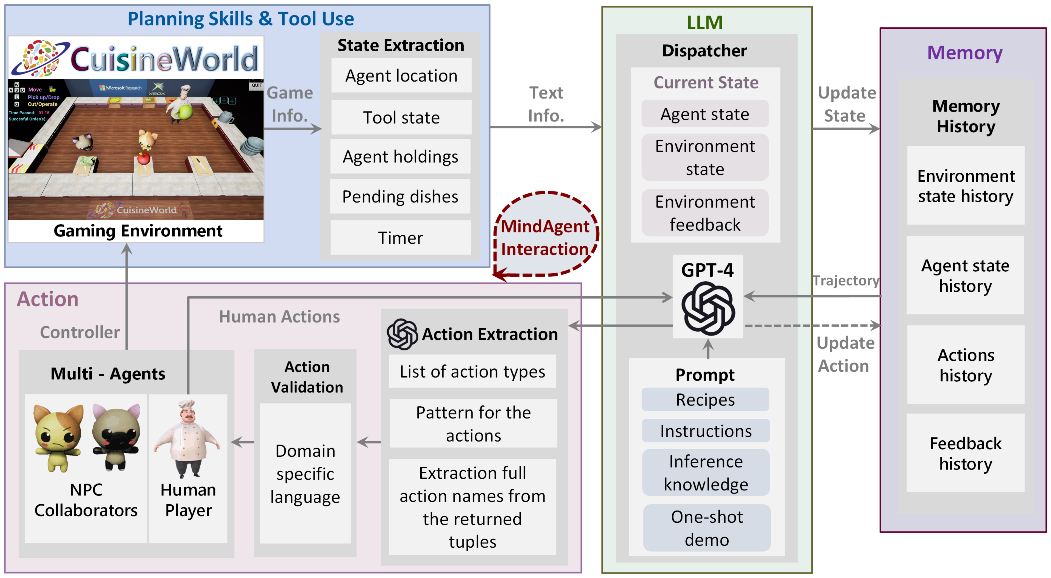


Figure 2: The MindAgent Infrastructure. **Planning Skill and Tool Use:** The game environment requires diverse planning skills and tool use to complete tasks. It generates relevant game information and converts the game data into a structured text format that the LLMs can process. **LLM:** The main workhorse of our infrastructure makes decisions, thus serving as a dispatcher for the multi-agent system. **Memory History:** A storage utility for relevant information. **Action Module:** Extracts actions from text inputs and convertd them into domain-specific language and validates DSLs so that they cause no errors during execution.

Collaboration Score (CoS). We need to evaluate to what extent the dispatcher (played by an LLM) can coordinate multiple agents to complete dish orders across a variety of scenarios. We are particularly interested in the question: Can the dispatcher continue to coordinate the agents into efficient collaborations with decreasing τ_{int}; *i.e.*, as more dish orders are flooding in? Our hypothesis is that an ideal dispatcher should be capable of coordinating the agents until there are way more tasks than the system can handle. Therefore, we introduce a *collaboration score* (**CoS**), defined as

$$\text{CoS} = \frac{1}{M}\sum_{i=1}^{M} \frac{\text{Number of completed tasks}\left[\tau_{\text{int},(i)}\right]}{\text{Number of completed tasks}\left[\tau_{\text{int},(i)}\right] + \text{Number of failed tasks}\left[\tau_{\text{int},(i)}\right]} \tag{1}$$

where M is the total number of τ_{int} intervals evaluated. Effectively, CoS is the average task completion rate across different τ_{int} conditions. In our default setting, we use $M = 5$. While the actual values of τ_{int} depend on the game level, we ensure that they span a wide range of difficulties including both relaxed and intense scenarios.

In summary, CuisineWorld is a game that emulates a virtual kitchen in which several robotic agents are commanded to use various cooking tools and ingredients to prepare as many dish orders as possible in a limited period of time. To necessitate collaboration, new orders will keep flooding in while the existing ones should be completed before their expiration times. Therefore, LLMs must properly coordinate the agents to maximize overall productivity. CuisineWorld offers game levels with a wide range of planning difficulty: dishes with different complexity (number of ingredients and tools involved), number of agents, order frequency and lifetime, etc., making it a useful test bed for LLM-based multi-agent planning.

4 The MindAgent Gaming AI Infrastructure

Our first foray into the challenging CuisineWorld benchmark is an interactive multi-agent planning framework with LLMs. It facilitates in-context learning and adopts a minimalist design for the purposes of demonstrating the emergent scheduling and coordination capacity while also bringing in exploratory prompting techniques that facilitate better planning and inform future approaches in this domain. Our MindAgent infrastructure comprises prompt, current state, and memory components, as shown in Figure 2 with details illustrated as follows:

Prompt incorporates four distinct sub-components: recipes, general instructions, inference knowledge, and a one-shot demo. **Recipes** outline hierarchical procedures for preparing various dishes at a given level. They specify the ingredients necessary for each intermediate or final product, the appropriate tools, and the expected post-cooking outcome. **Instructions** detail the foundational rules of CuisineWorld, delineating the array of actions agents can undertake within the game and enumerating the characteristics of every tool available in the current kitchen scenario. Moreover, they inform agents about the base ingredients retrievable from storage, as well as all potential intermediate products they can procure. Agents are also explicitly advised to remain cautious about feedback from the environment. **Inference Knowledge** encapsulates insights and helpful hints for the agent, which when utilized appropriately can guide agents to sidestep potential errors and improve their collaborative efficiency. **One-shot Demo** presents a step-by-step demonstration of the preparation of a distinct dish, different from other dishes at the current level, spanning several time steps, each of which is incorporated as part of the prompt. The demonstration illustrates the major procedures for cooking a dish in CuisineWorld, including obtaining ingredients, putting ingredients into different tools, transporting intermediate ingredients, and delivering the final dish to the serving table. More details of prompt please find in Appendix A.

Current State provides a snapshot of the prevailing observations from the environment. It encompasses information such as the locations of agents, the objects currently in the possession of agents, the tools that are accessible within the environment, the ingredients present within each tool, and the tools that are actively in use. Moreover, it includes optional feedback from the environment, triggered when agent actions violate the rules of the environment; for instance, when assigning two distinct actions to the same agent.

Memory archives the history of interaction with the environment. Specifically, it chronicles the state of the environment and the state of the agents at every time step.

In addition to the prompt modules, other modules are implemented to help interface between LLMs and CuisineWorld. **Action Extraction** employs a regular expression matching procedure to distill agent actions from the textual output of the LLMs. This module is indispensable because LLM output is not always clean, but may include information reflecting its internal thought processes or even issue apologies for prior missteps in reaction to environmental feedback. **Action Validation** utilizes a look-ahead checking mechanism. This module parses the proposed actions, assessing their feasibility. If an action is deemed unexecutable, an error message is returned.

4.1 Infrastructure Mechanisms

Assuming a multi-agent system with N agents, the system must complete a sequence of P different tasks. Each task has M_p different sub-tasks. Furthermore, the number and types of tasks are unknown at the beginning of the episode. The environment will sample a task for the agents to finish during a given interval. The agents must complete the designated task along with other tasks in the task queue. Additionally, each task has an expiration time, after which the task will be marked as a failure. The objective of the multi-agent system is to finish as many tasks as possible and fail as few tasks as possible within a given time frame.

Our objective is to find valid and optimal task planning, scheduling, and allocations. We define q_{pim} and c_{pim} as quality and cost, respectively, in the context of allocating agent i to work on sub-task m of task p in the episode. Then the combined utility for the sub-task is

$$u_{pim} = \begin{cases} q_{pim} - c_{pim}, & \text{if agent } i \text{ can execute sub-task } m \text{ of task } p \text{ in the episode} \\ -\infty & \text{otherwise.} \end{cases} \tag{2}$$

We define the assignment of sub-task m to agent i as

$$v_{pim} = \begin{cases} 1, & \text{agent } i \text{ is assigned to sub-task } m \text{ of task } p \text{ in the episode} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The goal is to maximize the utility of the episode subject to a time constraint. We define the execution time for task m by agent i for task p in the episode as τ_{pim}, and the maximum time allowed to execute the task as T_{max}, we express the task decomposition and assignment problem as

$$\arg\max_{v} \sum_{p=1}^{P} \sum_{i=1}^{N} \sum_{m=1}^{M_p} u_{pim} v_{pim}, \tag{4}$$

$$\text{subject to} \quad \begin{array}{rll} \sum_p \sum_i \sum_m \tau_{pim} v_{pim} & \leq T_{max} & \\ \sum_i v_{pim} & \leq 1 & \forall m \in M, \forall p \in P \\ v_{pim} & \in \{0,1\} & \forall i \in N, \forall m \in M, \forall p \in P. \end{array} \tag{5}$$

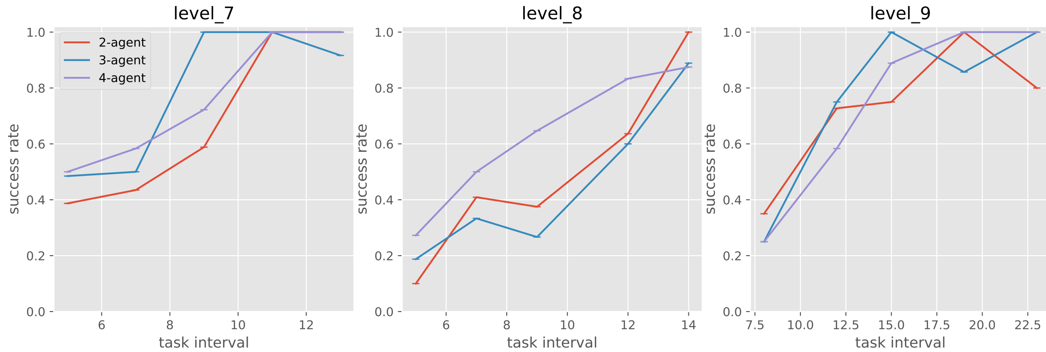


Figure 3: Collaboration efficiency curves on several levels. More level results can be found in the Figure 21 of Appendix D.

	very simple			simple			intermediate			advanced			Average
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
2 Agents	0.727	0.706	0.682	**0.687**	**0.664**	0.504	0.764	0.725	0.701	0.661	0.692	0.559	0.673
3 Agents	**0.781**	**0.778**	**0.780**	0.528	0.600	0.455	0.822	**0.771**	**0.815**	0.689	**0.733**	**0.570**	**0.694**
4 Agents	0.771	0.761	0.761	0.505	0.592	**0.626**	**0.848**	0.744	0.790	**0.692**	0.675	0.534	0.692

Table 1: Agent **CoS** performance scores on very simple, simple, intermediate, and advanced tasks for various numbers of agents.

Since this problem cannot be solved in polynomial time, we tackle it by leveraging LLMs.

Our prompt design choices try to help an LLM system solve Equation 4. In practice, we reformulate the equation with qualities or rewards expressed in natural language as environmental feedback. For example, when the agent successfully collects an item, the environment emits a signal "collect finish". When the dispatcher assigns a different task to the same agent, the environment emits a signal "agent IDs cannot be the same". As rewards are not immediately observable, we borrow spirits from temporal difference learning. State-action history is accumulated into the memory history. Due to context length limits, it is infeasible to fit the entire history into the context window. We select a fixed horizon history as part of the prompt. We further express the constraints of the system in natural language and repeat important constraints multiple times if necessary.

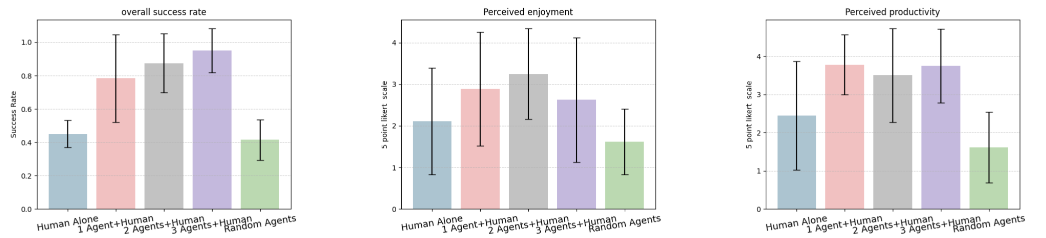
5 Experiments and Results

We have conducted extensive experiments in CuisineWorld. We first introduce the experiment settings and then present an analysis of our empirical results. We report LLM settings in Appendix B. Our experiments focused on addressing the following research questions:
Q1: How efficiently can the model dispatch multiple agents?
Q2: Can the model dispatch agents for dynamic, on-the-fly goals across different tasks?
Q3: How do various components of the input prompt influence the model's performance?
Q4: How do other LLMs perform compared to GPT-4?
Q5: To what extent can the existing methods collaborate with human users?
Q6: What is the human perception of collaborating with numerous intelligent agents?

5.1 Experimental Regimen I: LLMs Dispatch Multi-Agents (NPC)

Collaboration Efficiency (Q1, Q2). Figure 3 and Table 1 report the performance of our system under different settings. Please find the full results and visualizing figures in Appendix D .

Findings. As shown in Figure 3, more agents generally yield better collaboration efficiencies as is corroborated by computing **CoS** by levels: As shown in the tables, the **CoS** is the highest when there are two agents in two cases. The **CoS** is the highest when there are three agents in seven cases. The **CoS** is the highest when there are four agents in three cases. The results also confirm that more agents will lead to higher collaboration efficiencies. Thus, indicating that the LLM dispatcher can coordinate additional agents to execute tasks more efficiently. Second, we observe that the system performance degrades with more agents under less demanding conditions, indicating that the LLM dispatcher struggles with fewer tasks.



(a) **Collaboration score:** The collaboration score is higher if more agents are collaborating with human players, although the difference is insignificant.

(b) **Perceived enjoyment:** Humans enjoy the game more if they collaborate with the right number of agents.

(c) **Perceived productivity:** Players think collaborating with AI agents will improve productivity.

Figure 4: Human Evaluations. Full results can be found in Figure 27 of Appendix F.3.

5.2 Experimental Regimen II: Human and Multi-NPCs with LLMs

5.2.1 Human Data Collection

Measurement. In the background, we collect the numbers of failed and successful tasks during a participant's interaction with the game system. Additionally, we record the entire action history of players and intelligent agents. After each episode, the participants must complete a survey about their engagement with the system on a 5-point Likert chart. Our objective measure is intended to evaluate the human-AI teaming performance, and the subjective measure is designed to evaluate users' perceptions of the system. The human evaluation interface can be found in Appendix F.

5.2.2 Experimental Setting

We conducted a user study in our gaming environment that addresses **Q5** and **Q6**. The user study evaluates the LLM dispatcher's ability to collaborate with humans, where participants are collaborating with 1, 2, and 3 agents or working alone on the virtual cooking tasks. We consider the most general setting, where the LLM works on the unseen task, as Level_3.

5.2.3 Experiment Design

Hypotheses. The user study tests the following hypotheses:

- **H1: Task productivity**. Participants have higher productivity when collaborating with AI agents.
- **H2: Task productivity with more agents**. Participants have higher productivity when collaborating with more AI agents.
- **H3: Perception of the AI agents.** Participants have higher perceived task efficiency and more fun playing the game as a consequence of the collaboration.

Manipulated Variables. We use a within-subject design for our experiment. In particular, every user tries to finish the task solo or collaborates with different numbers of AI agents with varying degrees of competency. We randomize the order of the treatment to mitigate practice effects, fatigue effects, and carryover effects.

- **Single agent:** Participants work on the task by themselves.
- **LLM-powered multi-agent system:** Participants collaborate with the multi-agent AI system powered by an LLM.
- **Random agent:** Random agents execute random actions from a pool of valid actions. Participants collaborate with random AI agents.

We recruited 12 subjects for our study, including 2 females and 10 males. We used ANOVA to test the effects of different experimental conditions on collaboration performance and the subjective perception of the AI agents. Tukey HSD tests were conducted on all possible pairs of experimental conditions.

Findings. As showed in Figure 4 and Figure 27, we found significant effects on the team collaboration success rate $F(4, 55) = 28.11$, $p < 0.001$. Post-hoc comparisons using Tukey HSD tests revealed

2 agent	GPT-4 (full)	GPT-4 w/ only few-step	GPT-4 w/o inference knowledge	GPT-4 w/o feedback
CoS	0.764	0.710	0.714	0.311

Table 2: Additional ablation on Level 3

level_3	4agent using 4agent demo	4agent using 2agent demo	3agent using 3agent demo	3agent using 2agent demo
CoS	0.848	0.851	0.822	0.775

Table 3: Using different numbers of agents as one-shot demonstrations

that the team comprising the human player with LLM agents achieves a higher success rate than the human working alone ($p < 0.001$) across different numbers of agents, **thus confirming H1**. Although collaborating with more agents had a higher success rate, it was not significantly different from collaborating with one, two, or three agents ($p = 0.774$ and $p = 0.231$, respectively). We observed that human players have more fun playing the game when collaborating with LLM-powered AI agents than when playing alone ($p = 0.0126$). Players felt that collaboration with AI agents leads to higher productivity ($p = 0.0104$), **thus confirming H3**. Additionally, when playing with AI agents, human players take their actions based on other players' actions ($p = 0.00266$). Human players also found that AI agents are more predictable than random agents ($p < 0.001$). Further insights from player feedback highlighted an intriguing trade-off: while greater numbers of agents improved overall task success rates, this reduced the enjoyment of the game. Often, players felt sidelined and less involved. Thus, game developers should adjust AI performance to maintain player engagement and fun. As suggested by Yuan et al. (2022), aligning human values with AIs is a promising approach. The visualization figures of CuisineWorld showed in Appendix D.2.

6 Analysis and Emergent Gaming Abilities

6.1 Ablation Study for Multi-Agents

Study of the Prompt Components (Q3). In Table 2, we elucidate the performance of LLM dispatchers with certain components of the prompt omitted. Details about the prompt can be found in the appendices. Specifically, for these tests, we excluded individual components such as the inference knowledge, reduced the prompt example to a mere two steps instead of the complete demonstration, and evaluated the model without environmental feedback.

Findings. Table 2 indicates a significant drop in performance when environmental feedback is excluded, underscoring its pivotal role in the efficacy of the LLM dispatcher. Replaying action sequences reveals that, without feedback, the LLM dispatcher tends to repeat mistakes and gets stuck in specific states for prolonged durations. Another key takeaway is that a succinct two-step demonstration of input and output format can still achieve impressive performance for unseen tasks with dynamic objectives. Notably, in these two-step instances, there is no explicit guide to finishing any tasks, yet the model does not merely complete the task but continually performs additional tasks within the same episode. Furthermore, we observe that integrating human-crafted inference knowledge bolsters the performance of the LLM dispatcher. Lastly, even with few-shot demonstrations involving fewer agents, the LLM dispatcher retains satisfactory performance, as shown in Table 3.

Study of the Performance of other LLMs (Q4). To study how other LLMs perform on our tasks, we tested the collaboration performance of GPT-3.5, Claude-2, and LLaMA2, and Table 4 summarizes the results. For a fair comparison, all tests employed identical prompt inputs.

Findings. We observed that while other LLMs tend to underperform, models such as Claude-2 still manage to complete the task to a considerable extent.

6.2 Emergent Abilities

Across our experiments, our MindAgent framework exhibits the following emergent properties:

	2 agents				3 agents				4 agents			
	GPT-4	Claude-2	LLaMA2	ChatGPT	GPT-4	Claude-2	LLaMA2	ChatGPT	GPT-4	Claude-2	LLaMA2	ChatGPT
CoS	0.686	0.3125	0	0	0.822	0.372	0	0	0.848	0.473	0	0

Table 4: Performance of other LLMs on Level 3

(a) Multi-Agent (b) Human-NPC (c) VR

Figure 5: (a) Alex and Steve are collaborating to kill different animals. (b) A human player instructs the agents to perform certain actions. (c) A human player collaborating with agents in VR.

GPT-4 Minecraft	$\tau_{\text{int},(1)}$	$\tau_{\text{int},(2)}$	$\tau_{\text{int},(3)}$	$\tau_{\text{int},(4)}$	$\tau_{\text{int},(5)}$	**CoS**
Performance	0.195	0.381	0.704	0.792	0.833	0.581

Table 5: Performance of MindAgent framework in Minecraft

Emergent Collaboration Task Understanding. As shown in Table 2, especially in the few-step ablation entries, GPT-4 exhibits its proficiency even when not provided with a full demonstration of specific tasks. To clarify, a "full few-shot demo" typically refers to a comprehensive demonstration of a task, detailing each step and procedure involved. By contrast, we provide GPT-4 with only a partial demonstration or a glimpse of the task executing only two steps. Yet, despite this limited input, GPT-4's performance is remarkable. This underscores GPT-4's impressive **emergent zero-shot multi-agent planning** abilities. Beyond simply completing unseen tasks, GPT-4 also demonstrates adaptability by dynamically prioritizing multiple different tasks as they arise, emphasizing its **emergent multi-task, on-the-fly planning** skills.

Emergent Multi-agent Reasoning Abilities. Referring to Table 3, GPT-4 has the ability to deploy more agents based on demonstrations of fewer agents. For instance, it can effectively dispatch 4 agents having only seen demonstrations involving 2 agents. Moreover, the collaboration efficiency increases with an increasing number of agents, highlighting its **emergent collaboration** prowess.

7 Novel Game Adaptation

In line with our ongoing efforts to create collaborative, in-game, multi-agent systems, we ventured beyond CuisineWorld and integrated our infrastructure into Minecraft (Figure 5). In this adaptation, we designed several unique cooking tasks where two in-game agents, Alex and Steve, must cook various types of meat as shown in Appendix E.1. After cooking, they must deposit the meats into a chest. See Table 5 for the experimental results, and see Appendix E.2 for additional visualizations.

The empirical data that we collected from these game sessions provide us with compelling evidence that our multi-agent collaboration infrastructure has the robustness necessary to adapt it across multiple distinct games, paving the way for broader applications in the gaming industry. The action details of Minecraft please take the reference in Appendix E.3.

Going a step further, we integrated Microsoft's Azure speech-to-text API into the Minecraft environment, enabling human players to communicate and collaborate with in-game NPC agents by expressing their intents and desired goals to the NPCs through voice chat. The real-time vocal interaction enriches the gameplay experience, fostering a deeper level of immersion and synergy between human players and AI agents. In the case of the human player chatting with the multi-agent system, the prompt contains additional human instructions and human dialog history components.

8 Conclusion

We have introduced MindAgent, an infrastructure for multi-agent collaboration through LLMs across multiple gaming domains. We investigated its multi-agent planning capabilities, and we deployed our infrastructure into real-world video games that demonstrate its multi-agent and human-AI collaboration effectiveness. Additionally, we presented CuisineWorld, a text-based multi-agent collaboration benchmark that provides a new auto-metric Collaboration Score (CoS) to quantify collaboration efficiency. Beyond its practical applications, we anticipate that our work will guide the development of future gaming systems in which human-AI collaboration is seamless and intuitive. Furthermore, we are optimistic that our insights and findings will catalyze the design of games that are both technologically advanced and significantly more engaging and enjoyable for players.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022. 3

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022. 3

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022. 3, 22

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. 2

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019. 3, 22

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*, pp. 41–75. Springer, 2019. 22

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. *arXiv preprint arXiv:2306.06070*, 2023. 14

Xiaofeng Gao, Ran Gong, Yizhou Zhao, Shu Wang, Tianmin Shu, and Song-Chun Zhu. Joint mind modeling for explanation generation in complex human-robot collaborative tasks. In *2020 29th IEEE international conference on robot and human interactive communication (RO-MAN)*, pp. 1119–1126. IEEE, 2020. 23

Xiaofeng Gao, Qiaozi Gao, Ran Gong, Kaixiang Lin, Govind Thattai, and Gaurav S Sukhatme. Dialfred: Dialogue-enabled agents for embodied instruction following. *IEEE Robotics and Automation Letters*, 7(4):10049–10056, 2022. 22

Qiuyuan Huang, Jae Sung Park, Abhinav Gupta, Paul Bennett, Ran Gong, Subhojit Som, Baolin Peng, Owais Khan Mohammed, Chris Pal, Yejin Choi, et al. Ark: Augmented reality with knowledge interactive emergent ability. *arXiv preprint arXiv:2305.00970*, 2023. 2

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9118–9147. PMLR, 17–23 Jul 2022a. URL https://proceedings.mlr.press/v162/huang22a.html. 3

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022b. 3

Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6689–6699, 2019. 3

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*, 2023. 3

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022. 2, 3

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023. 14

Xinzhu Liu, Xinghang Li, Di Guo, Sinan Tan, Huaping Liu, and Fuchun Sun. Embodied multi-agent task planning from ambiguous instruction. *Proceedings of robotics: science and systems, New York City, NY, USA*, pp. 1–14, 2022. 22

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017. 3

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023. 2

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 2017–2025, 2022. 22

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023. 3, 22

Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration. *arXiv preprint arXiv:2010.09890*, 2020. 3, 22

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020. 3

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020. 22

Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8:345–383, 2000. 2

Alane Suhr, Claudia Yan, Charlotte Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. Executing instructions in situated collaborative interactions. *arXiv preprint arXiv:1910.03655*, 2019. 22

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. Learning to speak and act in a fantasy text adventure game. *arXiv preprint arXiv:1903.03094*, 2019. 22

Yanming Wan, Jiayuan Mao, and Josh Tenenbaum. Handmethat: Human-robot communication in physical and social environments. *Advances in Neural Information Processing Systems*, 35: 12014–12026, 2022. 3, 22

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a. 2, 3, 14

Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023b. 2, 3

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021. 2

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022. 2

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. 2, 3, 14

Luyao Yuan, Xiaofeng Gao, Zilong Zheng, Mark Edmonds, Ying Nian Wu, Federico Rossano, Hongjing Lu, Yixin Zhu, and Song-Chun Zhu. In situ bidirectional human-robot value alignment. *Science robotics*, 7(68):eabm4183, 2022. 8

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. Proagent: Building proactive cooperative ai with large language models. *arXiv preprint arXiv:2308.11339*, 2023a. 3

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*, 2023b. 3

Appendix for
MindAgent: Emergent Gaming Interaction

A PROMPT EXAMPLES

We provide some examples of prompts for CuisineWorld. Figure 6 shows an example of the system prompt info. Figure 7 shows an example of a partial demonstration.

```
The available actions are :
1) goto: goto a tool location
2) get: get some object from a tool
3) put: put some object into a tool
4) activate: activate the tool to cook all ingredients inside the tool into a different tools
5) noop: not performing any actions
Sometimes the system will give you error messages. Please consider these error messages when executing actions.
You need to specify action for all of the agents, **except human**. They all have different agent numbers. Do not assign actions to the same agent more than once.

When the tools reach its capacity, you need to take stuff out. Otherwise, you cannot put items inside.
When you are holding objects, you cannot get any more objects.
When you are holding objects, you cannot activate tools.
Afer you cooked a required dish, you need to put it into the servingtable.
You can only pick up objects from the tool location, if you are located at the tool location.
When you activate any tools, make sure all the items inside the tool are respecting the recipes. Otherwise, you will cook waste. Avoid waste at all cost.
*** You should mix salad in the mixer. To make salad you should chop veggies first. ***
*** If the tool is occupied, indicated by the occupy() predicate, you cannot get objects from it or put objects into it. ***
*** The food orders are keep coming. You should finish as many dishes as possible and finish every dish as soon as possible. Please deliver the order to the serveringtable when it is finished. ***
*** The dish will expire after the lifetime reaches 0 and it's not at the serveringtable. Please avoid this. *** Here are the recipes:

Cook porkMeatcake at:
 -- location: blender
 -- with ingredients:    pork,    flour,
Cook salmonSashimi at:
 -- location: chopboard
 -- with ingredients:    salmon,
Cook tunaSashimi at:
 -- location: chopboard
 -- with ingredients:    tuna,
Cook mixedSashimi at:
 -- location: mixer
 -- with ingredients:    salmonSashimi,    tunaSashimi,
The following objects are available:
 --1) salmonSashimi
 --2) tuna
 --3) mixedSashimi
 --4) tunaSashimi
 --5) porkMeatcake
 --6) salmon
 --7) flour
 --8) pork
The objecsts are cooked using tools or are just base ingredients.
Among them, the following are base ingredients:
 --1) tuna
 --2) salmon
 --3) flour
 --4) pork
You can only obtain base ingredients from the storage initially.
Additional rules:
You can place up to infinite item into the storage0
You can place up to infinite item into the storage0
You can place up to infinite item into the servingtable0
You can place up to infinite item into the servingtable0
You can place up to 1 item into the chopboard0
You can place up to 1 item into the chopboard0
You can place up to 1 item into the chopboard1
You can place up to 1 item into the chopboard1
You can place up to 5 item into the mixer0
You can place up to 5 item into the mixer0
You can place up to 5 item into the mixer1
You can place up to 5 item into the mixer1
** Only ** the following tools are available:
storage0, servingtable0, chopboard0, chopboard1, mixer0, mixer1, You cannot pick up these tools. You can only use those tools at the corresponding location.
```

Figure 6: The MindAgent system prompt example.

```
There are 2 agents available, so you can execute 2 actions at a time.
Goal: porkMeatcake
t=0
-state:
at(agent0, servingtable0)
at(agent1, servingtable0)
hold(agent0, None)
hold(agent1, None)
inside(storage0, None)
inside(blender0, None)
inside(chopboard0, None)
inside(servingtable0, None)

-action:
***
goto_agent0_storage0
goto_agent1_storage0
***

Goal: porkMeatcake
t=1
-state:
at(agent0, storage0)
at(agent1, storage0)
hold(agent0, None)
hold(agent1, None)
inside(storage0, None)
inside(blender0, None)
inside(chopboard0, None)
inside(servingtable0, None)

-action:
***
get_agent0_flour_storage0
get_agent1_pork_storage0
***

Goal: porkMeatcake
t=2
-state:
at(agent0, storage0)
at(agent1, storage0)
hold(agent0, flour)
hold(agent1, pork)
inside(storage0, None)
inside(blender0, None)
inside(chopboard0, None)
inside(chopboard1, None)
inside(pan0, None)
inside(servingtable0, None)

-action:
***
goto_agent0_blender0
goto_agent1_blender0
***
```

Figure 7: The MindAgent system partial one-shot demo example.

B LLM SETTINGS

We perform experiments on CuisineWorld through OpenAI APIs and Anthropic APIs. All GPT-4 experiments employ the gpt-4-0613 model, and all Chat-GPT experiments employ gpt-3.5-turbo-0613. For the Llama 2 experiments, we use the hugging face inference endpoints Llama-2-70b-chat-hf. We set the temperature for all experiments to 0.1 following Wang et al. (2023a). We report the average results over three episodes.

C CUISINEWORLD TASK DETAILS

C.1 CUISINEWORLD TASK DEFINITIONS

We follow prior work (Yao et al., 2023; Liu et al., 2023; Deng et al., 2023) to **interactively evaluate LLMs as planning agents**. Overall, the interactive evaluation can be formulated as a *Markov Decision Process* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G})$, with state space $\mathcal{S}$, action space $\mathcal{A}$ (effectively indicating all the possible schedules that can be made at a single time step), transition dynamics $\mathcal{T}$, reward function $\mathcal{R}$, and task instruction space $\mathcal{G}$. Note that, although there are multiple agents inside CuisineWorld that can be coordinated, asmentioned above, we adopt a centralized planning scheme and thereby formulate our game as a single-agent, fully-observable decision-making problem. An illustration of the state & action space and the possible tasks of our game can be found in Figure 1 of the paper.

State Space $\mathcal{S}$. In a CuisineWorld virtual kitchen, there are two types of entities: `location` and `agent`. For each entity, the game will provide a set of descriptions, and the aggregated descriptions of all entities will be the state returned by the game. A `location` can be *storage*, where one can obtain ingredients and dispense waste, a *serving table*, onto which one should put the completed, or a cooking tool; *e.g.*, *pan* or *blender*. We offer up to two descriptions for each location: `inside(location, items)`, indicating what items (some ingredients, completed dishes, *etc.*) are now inside the location, and `occupy(location)`, suggesting `location` is now being used and cannot be touched; *e.g.*, an activated blender. An `agent` is an entity that can be dispatched to complete the task, and we provide up to three descriptions for each agent: `at(location, agent)`, indicating that `agent` is now at `location`, `hold(agent, items)`, suggesting what items `agent` is holding, and `occupy(agent)`, implying `agent` is now operating a tool, *e.g.*, chopping some fruits, and will not respond to any dispatching command. The set of tool distributions can be found in Table 23.

Action Space $\mathcal{A}$. An action in CuisineWorld is a list of dispatching commands. Given N `agent` entities, a total of N commands must be generated. The agent provides the following commands (also tabulated in Table 13):

1. `goto(agent, location)`, to let `agent` move to `location`;
2. `get(agent, location, item)`, to let `agent` get a specific item from `location`;
3. `put(agent, location)`, to put whatever `agent` is holding into `location`;
4. `activate(agent, location)`, to let `agent` turn on `location` if it is a cooking tool, *e.g. blender*;
5. `noop(agent)`, to have `agent` perform no actions in this round of dispatching.

Note that, to avoid the possible confusion of multiple agents being dispatched to operate with the same `location`, the dispatcher also must properly order the dispatching commands as they will be executed sequentially.

C.2 IMPLEMENTING CUISINEWORLD

The implementation of CuisineWorld mostly follows the spirit of *Overcooked!*, a renowned video game. Therefore, we refer to many of its game mechanisms while simplifying some of them; *e.g.*, we skip low-level control and assume all `agent` entities have access to all `location` at any time. Specifically, we crawled the rules and recipes from the community-contributed wiki[1] of *Overcooked!* streamlined them, and made necessary modifications, ending up with the basic version

[1] https://steamcommunity.com/sharedfiles/filedetails/?id=1769729191

of CuisineWorld comprising **10** types of `location` (*serving table*, *storage*, and 8 different cooking tools), **27** types of ingredients, and **33** unique dishes. We grouped the dishes based on their difficulty (primarily based on the number of cooking tools involved) to design and implement **12** game levels, which are further categorized into 4 classes: *entry*, *simple*, *intermediate*, and *advanced*, with 3 levels each. Note that the recipes, dishes, and levels can be easily extended to incorporate more challenging tasks.

C.3 Task Graph Visualization

In CuisineWorld, we provide tasks of different complexities to holistically evaluate the multi-agent system's performance. Additionally, the environment is highly customizable and extendable. Users only need only modify the JSON files to add more tasks or modify existing tasks. In the following sebsections, we visualize different CuisineWorld task graphs.

C.3.1 Level 0 – Very Simple

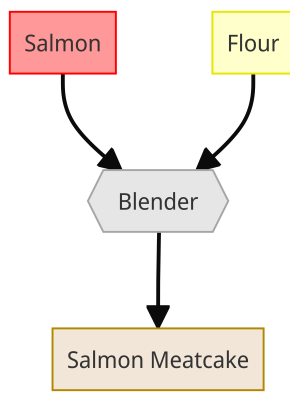
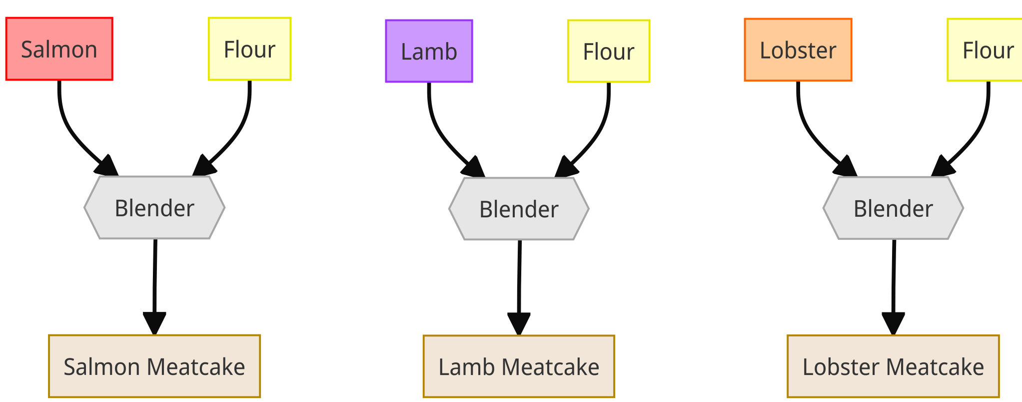


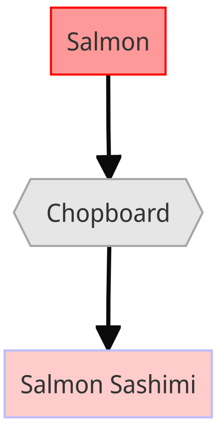
Figure 8: Salmon Meatcake

C.3.2 Level 1 – Very Simple

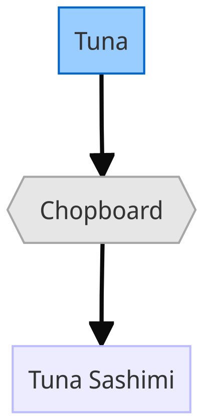


(a) Salmon Meatcake (b) Lamb Meatcake (c) Lobster Meatcake

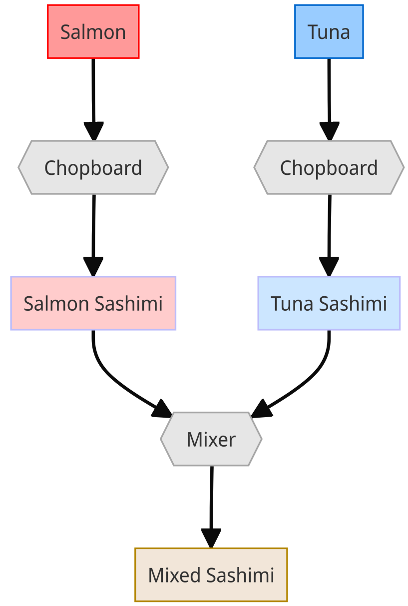
C.3.3 Level 2 – Simple



(a) Salmon Sashimi

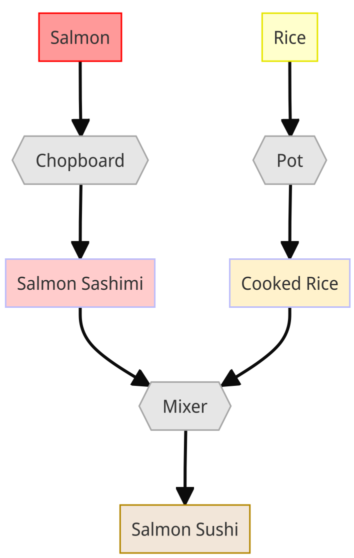


(b) Tuna Sashimi

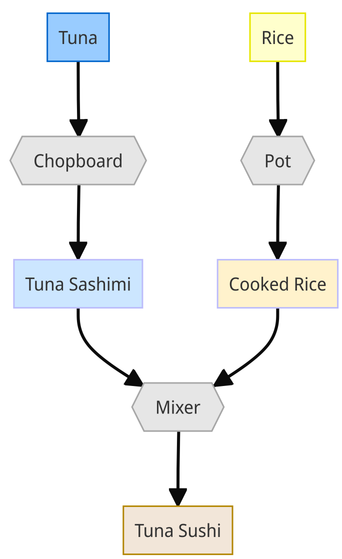


(c) Mixed Sashimi

C.3.4 Level 3 – Intermediate

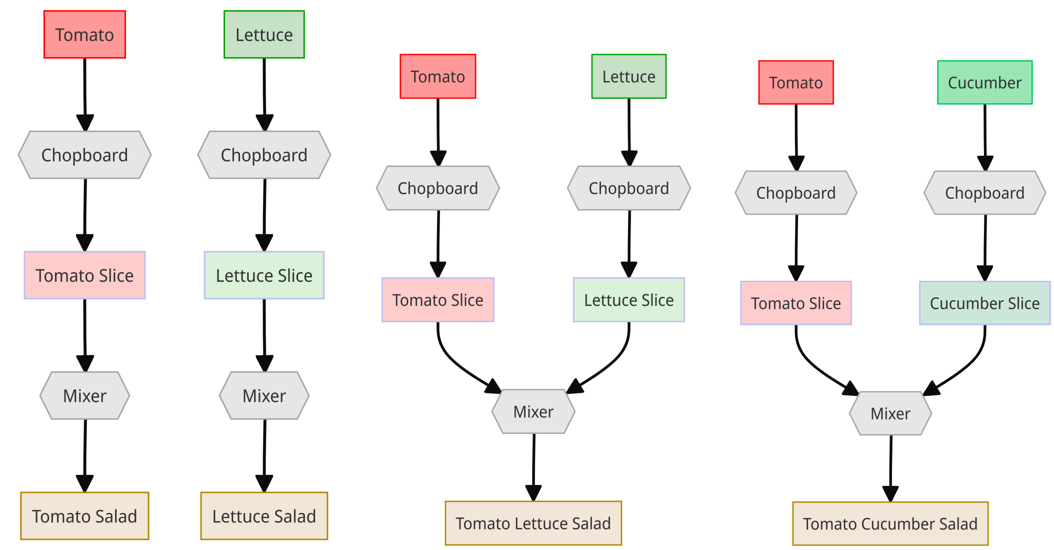


(a) Salmon Sushi



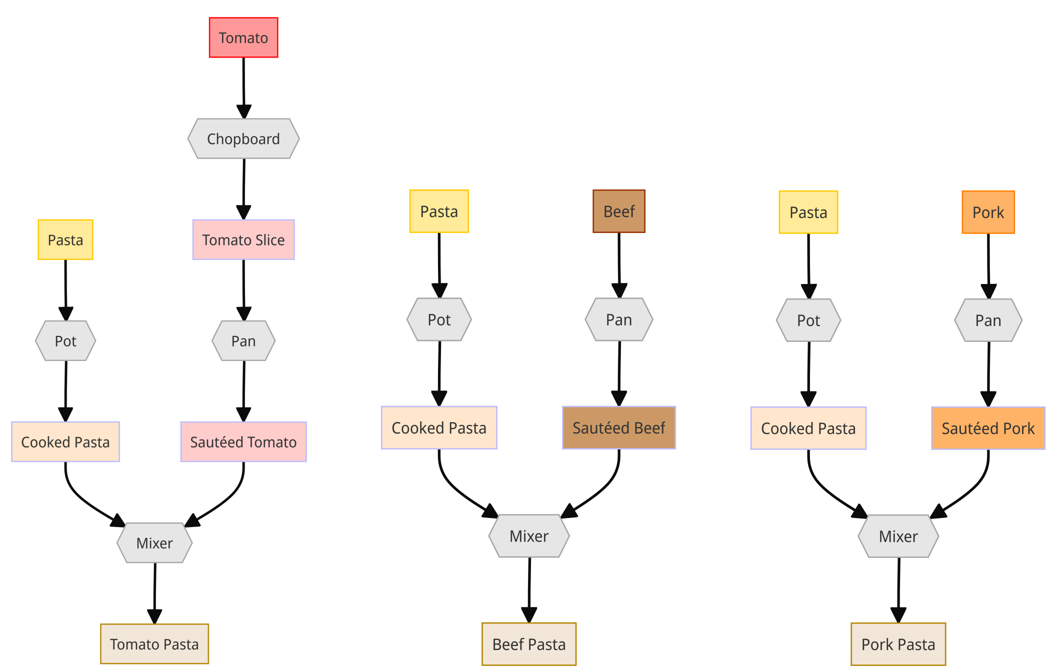
(b) Tuna Sushi

C.3.5 Level 4 – Simple



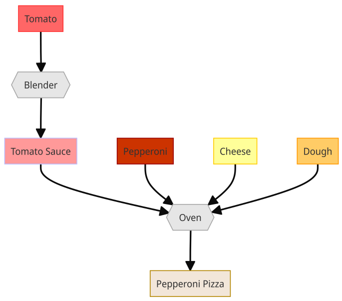
(a) Tomato Salad (b) Lettuce Salad (c) Tomato Lettuce Salad (d) Tomato Cucumber Salad

C.3.6 Level 5 – Advanced

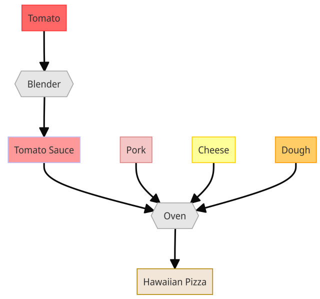


(a) Tomato Pasta (b) Beef Pasta (c) Pork Pasta

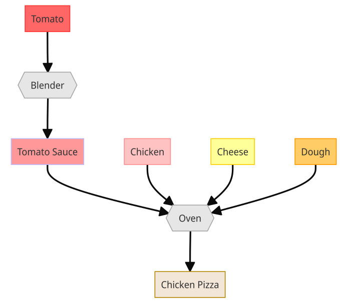
C.3.7 Level 6 – Unused



(a) Pepperoni Pizza

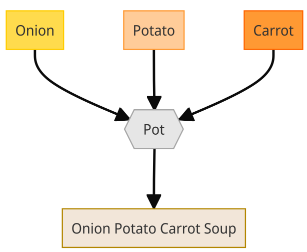


(b) Hawaiian Pizza

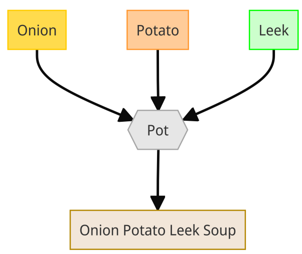


(c) Chicken Pizza

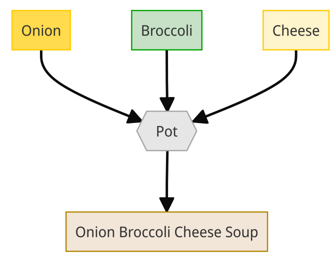
C.3.8 Level 7 – Very Simple



(a) Onion Potato Carrot Soup

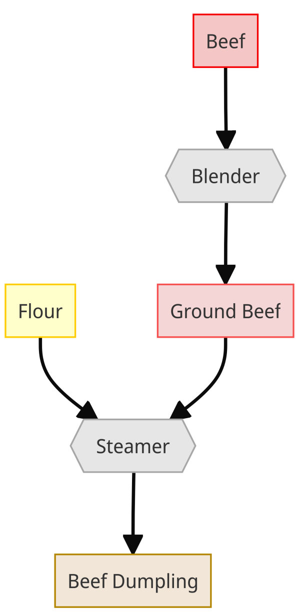


(b) Onion Potato Leek Soup

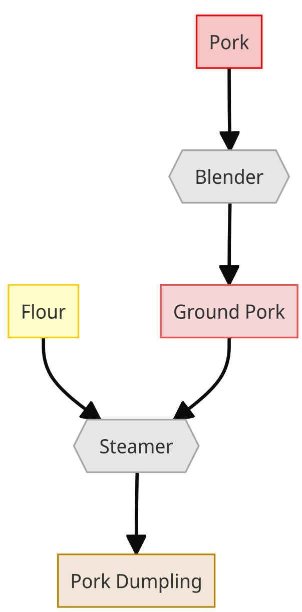


(c) Onion Broccoli Cheese Soup

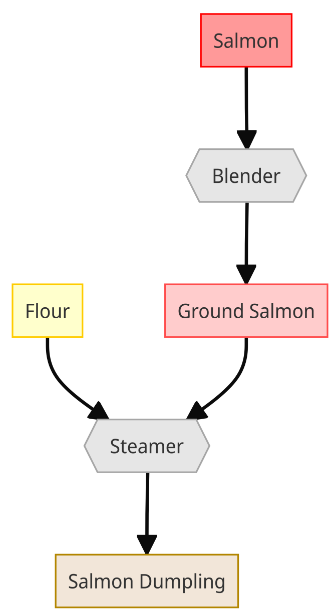
C.3.9 Level 8 – Simple



(a) Beef Dumpling

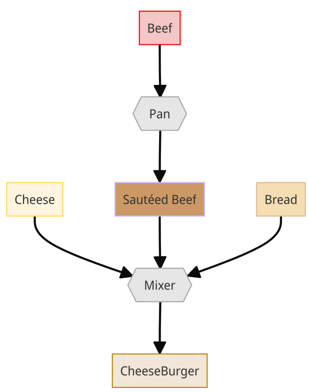


(b) Pork Dumpling

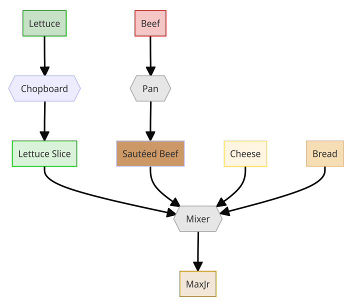


(c) Salmon Dumpling

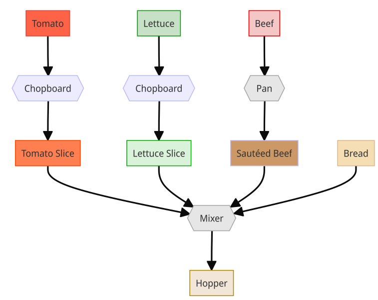
C.3.10 Level 9 – Intermediate



(a) Cheeseburger

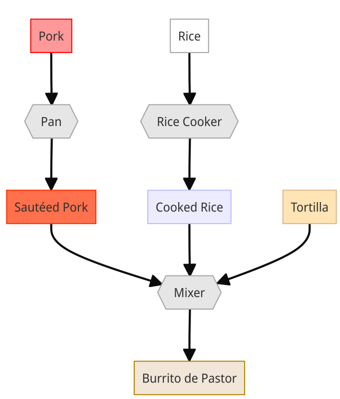


(b) MaxJr

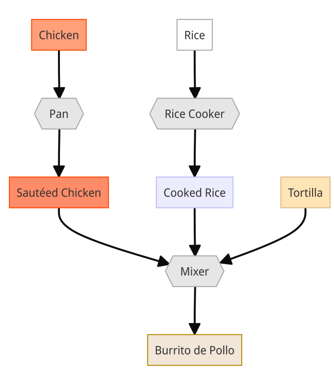


(c) Hopper

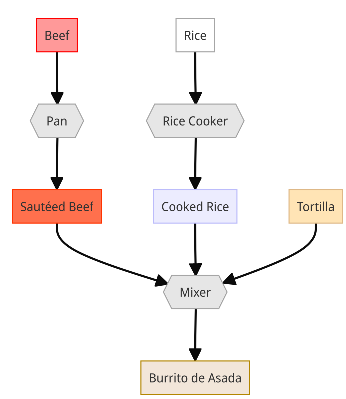
C.3.11 Level 10 – Intermediate



(a) Burrito de Pastor

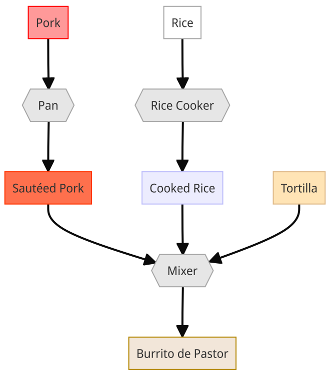


(b) Burrito de Pollo

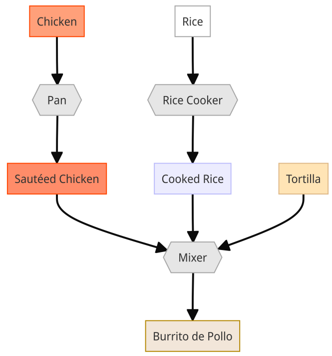


(c) Burrito de Asada

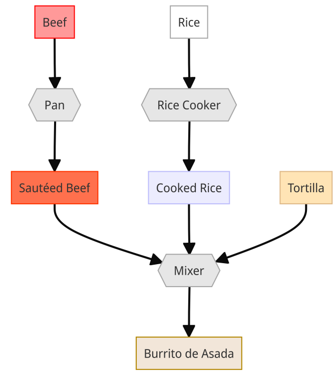
C.3.12 Level 11 – Advanced



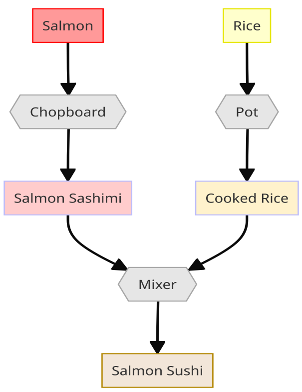
(a) Burrito de Pastor



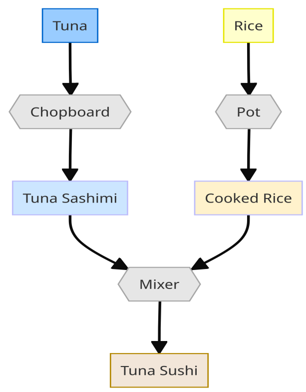
(b) Burrito de Pollo



(c) Burrito de Asada

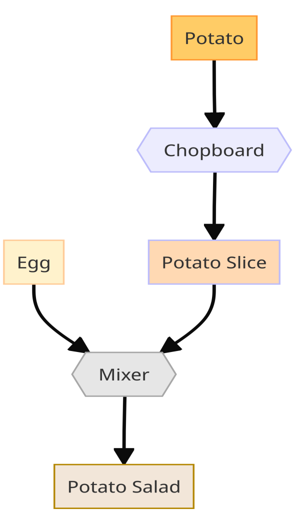


(d) Salmon Sushi

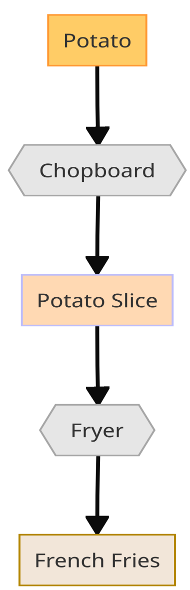


(e) Tuna Sushi

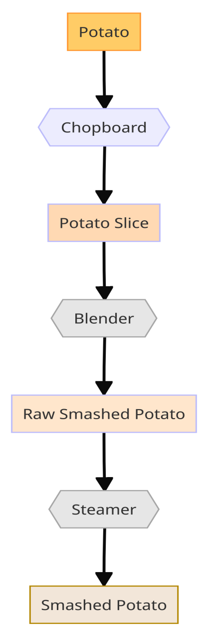
C.3.13 Level 12 – Advanced



(a) Potato Salad



(b) French Fries



(c) Smashed Potato

D ADDITIONAL RESULTS IN CUISINEWORLD

The following tables report additional performance results for several different numbers of agents and task complexity levels, performance of other LLMs, and additional ablation results.

2-agent	very simple			simple			intermediate			advanced			Avg.
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
GPT4 $\tau_{\text{int},(1)}$	18/54	18/56	12/31	14/34	12/30	3/30	10/26	7/20	7/23	6/23	6/21	10/36	0.318
GPT4 $\tau_{\text{int},(2)}$	18/31	17/34	10/23	13/26	12/22	9/22	10/17	8/11	6/12	5/13	4/14	8/21	0.486
GPT4 $\tau_{\text{int},(3)}$	18/25	19/25	10/17	16/18	11/18	6/16	11/13	6/8	7/10	8/10	9/9	8/17	0.709
GPT4 $\tau_{\text{int},(4)}$	18/18	18/19	12/12	11/14	11/12	7/11	12/12	8/8	9/9	6/7	8/9	11/12	**0.912**
GPT4 $\tau_{\text{int},(5)}$	18/18	17/17	12/12	11/13	11/13	9/9	11/11	4/5	7/7	8/8	8/8	9/12	**0.937**
CoS	0.727	0.706	0.682	**0.687**	**0.664**	0.504	0.764	0.725	0.701	0.661	0.692	0.559	0.673

Table 6: 2 agents performance on different tasks

3-agent	very simple			simple			intermediate			advanced			Average
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
GPT4 $\tau_{\text{int},(1)}$	21/55	24/55	16/33	17/33	9/28	6/32	12/25	5/20	8/21	7/22	7/22	9/26	**0.368**
GPT4 $\tau_{\text{int},(2)}$	20/31	25/33	11/22	4/24	13/24	7/21	14/20	9/12	9/13	7/14	8/14	10/23	0.549
GPT4 $\tau_{\text{int},(3)}$	22/25	21/26	17/17	11/20	9/17	4/15	13/14	8/8	12/12	7/7	9/10	10/16	**0.791**
GPT4 $\tau_{\text{int},(4)}$	22/22	20/21	14/14	9/13	7/10	6/10	10/10	6/7	10/10	5/8	7/8	11/13	0.846
GPT4 $\tau_{\text{int},(5)}$	20/20	15/16	11/12	10/14	10/11	8/9	12/12	6/6	8/8	5/5	8/8	6/10	0.914
CoS	**0.781**	**0.778**	**0.780**	0.528	0.600	0.455	0.822	**0.771**	**0.815**	0.689	**0.733**	**0.570**	**0.694**

Table 7: 3 agents performance on different tasks

4-agent	very simple			simple			intermediate			advanced			Average
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
GPT4 $\tau_{\text{int},(1)}$	22/54	18/55	17/34	13/34	8/28	9/33	16/27	5/20	8/23	5/22	8/22	8/35	0.349
GPT4 $\tau_{\text{int},(2)}$	24/32	21/33	14/24	14/25	12/24	11/22	16/19	7/12	9/15	7/14	6/12	12/23	**0.590**
GPT4 $\tau_{\text{int},(3)}$	23/25	23/26	13/18	11/19	10/17	11/17	15/17	8/9	11/11	7/8	10/11	9/17	0.785
GPT4 $\tau_{\text{int},(4)}$	22/22	21/22	14/14	7/15	10/13	10/12	12/13	9/9	10/10	6/7	8/8	9/13	0.875
GPT4 $\tau_{\text{int},(5)}$	14/18	20/20	14/14	7/13	9/11	7/8	12/12	5/5	7/7	6/6	3/5	7/10	0.859
CoS	0.771	0.761	0.761	0.505	0.592	**0.626**	**0.848**	0.744	0.790	**0.692**	0.675	0.534	0.692

Table 8: 4 agents performance on different tasks

	2 agent				3 agent				4 agent			
	GPT-4	Claude-2	LLaMA	ChatGPT	GPT-4	Claude-2	LLaMA	ChatGPT	GPT-4	Claude-2	LLaMA	ChatGPT
$\tau_{\text{int},(1)}$	10/26	3/24	0	0/24	12/25	5/26	0	0/24	16/27	9/25	0	0/24
$\tau_{\text{int},(2)}$	10/17	3/16	0	0/15	14/20	4/16	0	0/15	16/19	4/15	0	0/15
$\tau_{\text{int},(3)}$	11/18	3/12	0	0/12	13/14	3/12	0	0/12	15/17	4/12	0	0/12
$\tau_{\text{int},(4)}$	11/13	3/9	0	0/9	10/10	5/11	0	0/9	12/13	6/11	0	0/9
$\tau_{\text{int},(5)}$	11/11	4/6	0	0/6	12/12	5/7	0	0/6	12/12	6/7	0	0/6
CoS	0.686	0.3125	0	0	0.822	0.372	0	0	0.848	0.473	0	0

Table 9: Performance of other LLMs on Level 3

2 agent	GPT-4	GPT-4 w/ few-step	GPT-4 w/o inference knowledge	GPT-4 w/o feedback
$\tau_{\text{int},(1)}$	10/26	8/26	8/25	4/25
$\tau_{\text{int},(2)}$	10/17	11/19	9/17	4/17
$\tau_{\text{int},(3)}$	11/13	11/13	10/12	4/12
$\tau_{\text{int},(4)}$	12/12	9/11	8/9	1/9
$\tau_{\text{int},(5)}$	11/11	10/10	9/9	5/7
CoS	0.764	0.710	0.714	0.311

Table 10: Additional ablation results

level_3	4agent using 4agent demo	4agent using 2agent demo	3agent using 3agent demo	3agent using 2agent demo
GPT4 $\tau_{\text{int},(1)}$	16/27	14/27	12/25	11/25
GPT4 $\tau_{\text{int},(2)}$	16/19	16/20	14/20	11/19
GPT4 $\tau_{\text{int},(3)}$	15/17	15/16	13/14	12/14
GPT4 $\tau_{\text{int},(4)}$	12/13	13/13	10/10	12/12
GPT4 $\tau_{\text{int},(5)}$	12/12	12/12	12/12	11/11
CoS	0.848	0.851	0.822	0.775

Table 11: Using different numbers of agents demos

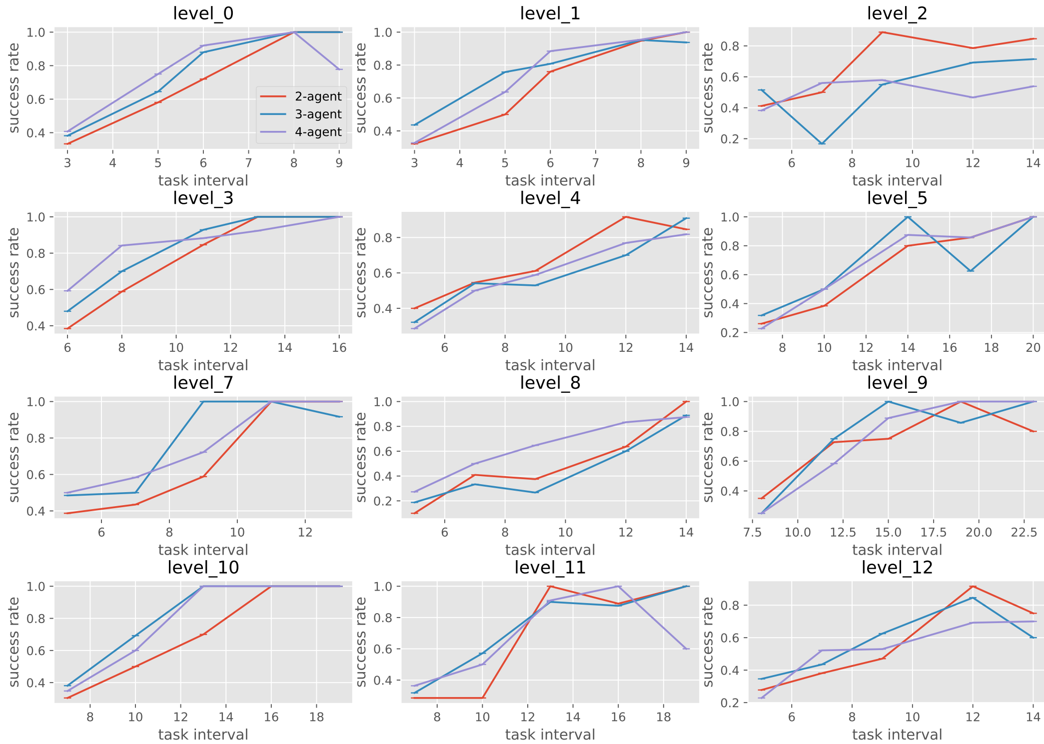


Figure 21: Collaboration results on different tasks

D.1 Comparison Between CuisineWorld and Related Benchmarks

Benchmark	Multi-task	Object Interaction	Tool Use	Maximum Agents	Collabo-ration	Human in-the-loop	Procedural Level Generation
ALFWorld (Shridhar et al., 2020)	✓	✓	✓	1	✗	✗	✗
WAH (Puig et al., 2020)	✓	✓	✗	2	✓	✓	✗
TextWorld (Côté et al., 2019)	✓	✓	✓	1	✗	✗	✓
Generative Agents (Park et al., 2023)	✓	✓	✓	25	✗	✗	✓
EMATP (Liu et al., 2022)	✓	✓	✓	2	✓	✗	✗
Overcooked-AI (Carroll et al., 2019)	✗	✓	✓	2	✓	✓	✗
HandMeThat (Wan et al., 2022)	✓	✓	✓	2	✓	✗	✗
DialFRED (Gao et al., 2022)	✓	✓	✓	2	✓*	✗	✗
TEACH (Padmakumar et al., 2022)	✓	✓	✓	2	✓*	✗	✗
CerealBar (Suhr et al., 2019)	✗	✗	✗	2	✓	✗	✗
LIGHT (Urbanek et al., 2019)	✓	✗	✗	1369	✗	✓	✓
Diplomacy (Bakhtin et al., 2022)	✗	✗	✗	7	✓	✓	✗
CuisineWorld (Ours)	✓	✓	✓	4+	✓	✓	✓

Table 12: Comparison between CuisineWorld and other related benchmarks. *: Notably, even though multiple agents can be present, the second agent is limited to communicating with the first agent. The second agent cannot interact with the environment in an active gaming capacity.

Table 12 compares CuisineWorld against related benchmarks along the following criteria:

- **Multi-task**: The benchmark contains multiple different tasks.
- **Object Interaction**: Agents must manipulate or engage with different items or environmental elements to achieve certain goals with irreversible actions.
- **Tool Use**: Completing tasks necessitates the use of specific tools by the agents.
- **Maximum Agents**: Denotes the upper limit of agents that can be present in any experiment.
- **Collaboration**: Many tasks mandate teamwork and collaboration between different agents.
- **Human in-the-loop**: The framework allows humans to join the game and collaborate actively with the agents.
- **Procedural Level Generation**: There is flexibility in adding new tasks, making the game dynamic and adaptable.

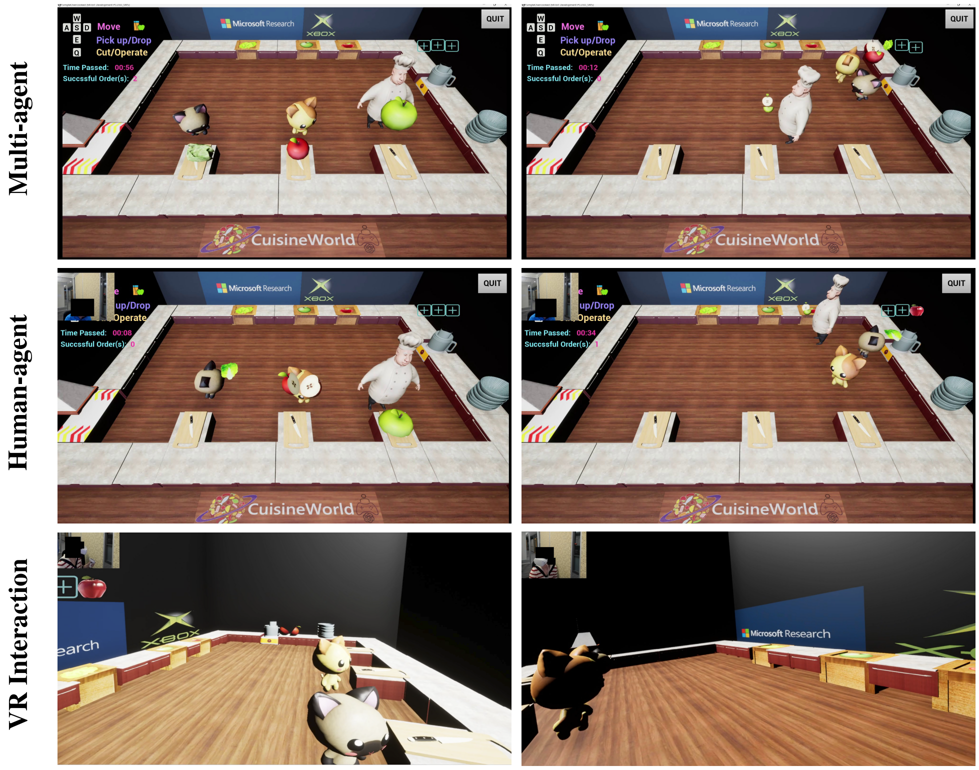


Figure 22: (Top) A multi-agent collaboration example in CuisineWorld; the three agents are preparing a mixed juice together. (Middle) A human player as the head chef instructing the agents to cook mixed juice. (Bottom) A human player collaborating with collaborative agents in VR.

D.2 Visualizing CuisineWorld

To implement CuisineWorld into a real-world game system, we built on top of Gao et al. (2020). In our game, as visually depicted in Figure 22, players are given the opportunity to engage in collaborative interactions with NPCs. In this game, the actions of human players can be obtained from an inverse dynamic model by checking pre-conditions and post-effects. This introduces a unique dynamic to the gameplay, enabling users to experience a more immersive cooperative environment. Additionally, the game's interface is versatile, providing players multiple ways to interact within the game world. They can either use a standard keyboard setup, which is conventional and likely familiar to most PC gamers, or immerse themselves even further using a Virtual Reality (VR) device. This VR functionality ensures a more tactile and realistic interaction, as players can physically move, gesture, and engage with the NPCs and other in-game elements in the 3D environment.

D.3 Additional CuisineWorld Details

Type	Arguments	Description
`goto`	`agent` `location`	Move `agent` to `location`
`get`	`agent` `location` `(item)`	`agent` obtain `item` from `location`
`put`	`agent` `location`	`agent` put everything it holds to `location`
`activate`	`agent` `location`	`agent` turn on `location`
`noop`	`agent`	not dispatching `agent`

Table 13: Action space in CuisineWorld.

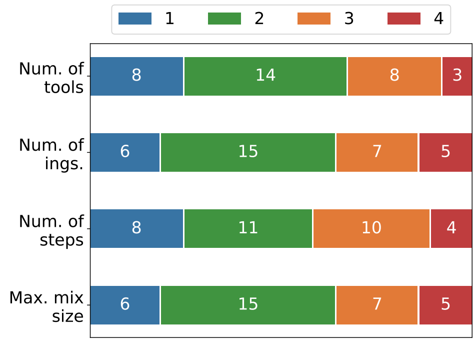
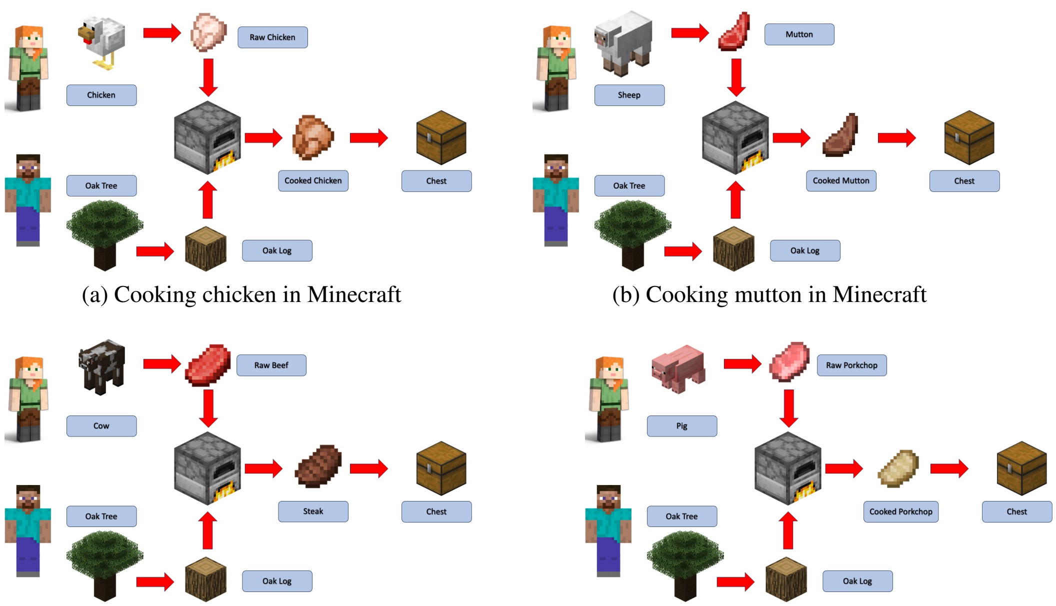


Figure 23: Dish distribution over the number of tools and ingredients (ings.) involved, cooking steps, and maximum mixture size as in the recipe.

E Minecraft

E.1 Task Graphs

In Figure 24 we visualize the task graphs for different tasks in Minecraft.



(a) Cooking chicken in Minecraft
(b) Cooking mutton in Minecraft
(c) Cooking steak in Minecraft
(d) Cooking porkchop in Minecraft

Figure 24: Task Visulization in Minecraft

E.2 Gameplay Visualization

We visualize Minecraft gameplay in Figure 25.

Figure 25: (Top) A multi-agent collaboration example in Minecraft. At left Alex and Steve are killing different animals and at right they are cooking meat in a furnace together. (Middle) A human player instructing the agents to perform certain actions. (Bottom) A human player collaborating with agents in VR.

E.3 Action Details for Mindecaft

We define the following actions for the multi-agent system in our Minecraft game: 1) `goto(agent, location)`; 2) `killMob(agent, mobType)`; 3) `mineBlock(agent, blockType)`; 4) `putFuelFurnace(agent, fuelType)`, to put the item from agent's inventory to the furnace's bottom slot. 5) `putItemFurnace(agent, itemType)`, to put the item from agent's inventory to the furnace's top slot; 6) `takeOutFurnace(agent)`, take out the cooked item from the furnace 7) `putInChest(agent, itemType)`.

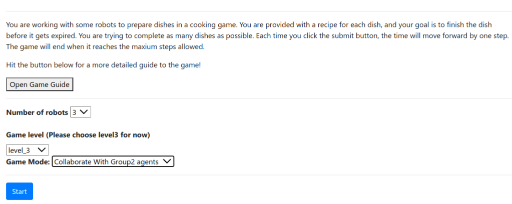
The state space in Minecraft contains the following: 1) nearby blocks for each agent, 2) nearby entities for each agent, 3) each agent's inventory, 4) items inside the furnace, 5) items inside the chest, and 6) the human player's inventory if a human player is involved.

To ensure reproducibility, we modify the game mechanism. A killed mob will respawn nearby, and a mined block will also respawn nearby.

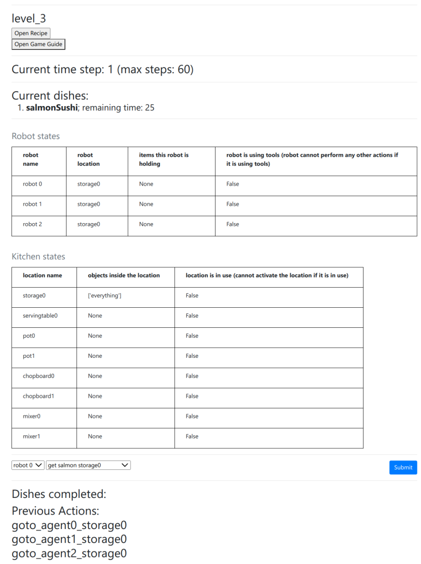
F Additional Information on Human Evaluation

F.1 Human Evaluation Interface

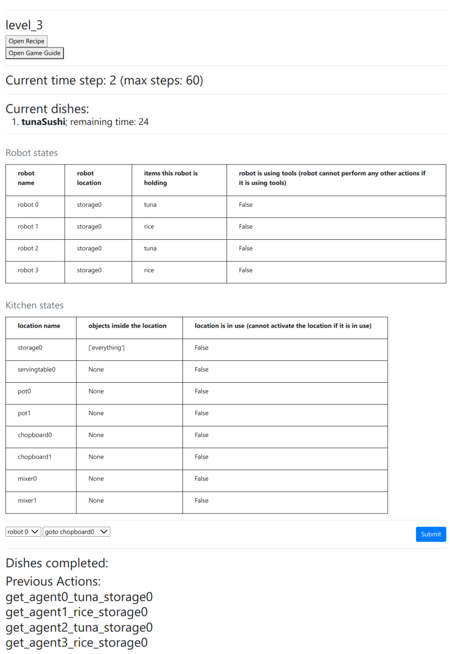
We use the human evaluation interface to test the human's perception of collaborative agents. This gives us a more controlled environment so users' perception of collaborative agents does not depend on their ability to control the keyboard and mouse, and their perception of collaborative agents does not depend on the latency and rate limits of GPT-4. Figure 26 shows the interface welcome screen, human evaluation examples, and examples of human instructions.



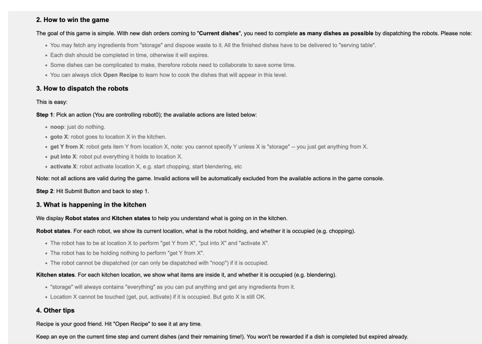
(a) Human evaluation interface welcome screen



(b) Human evaluation example



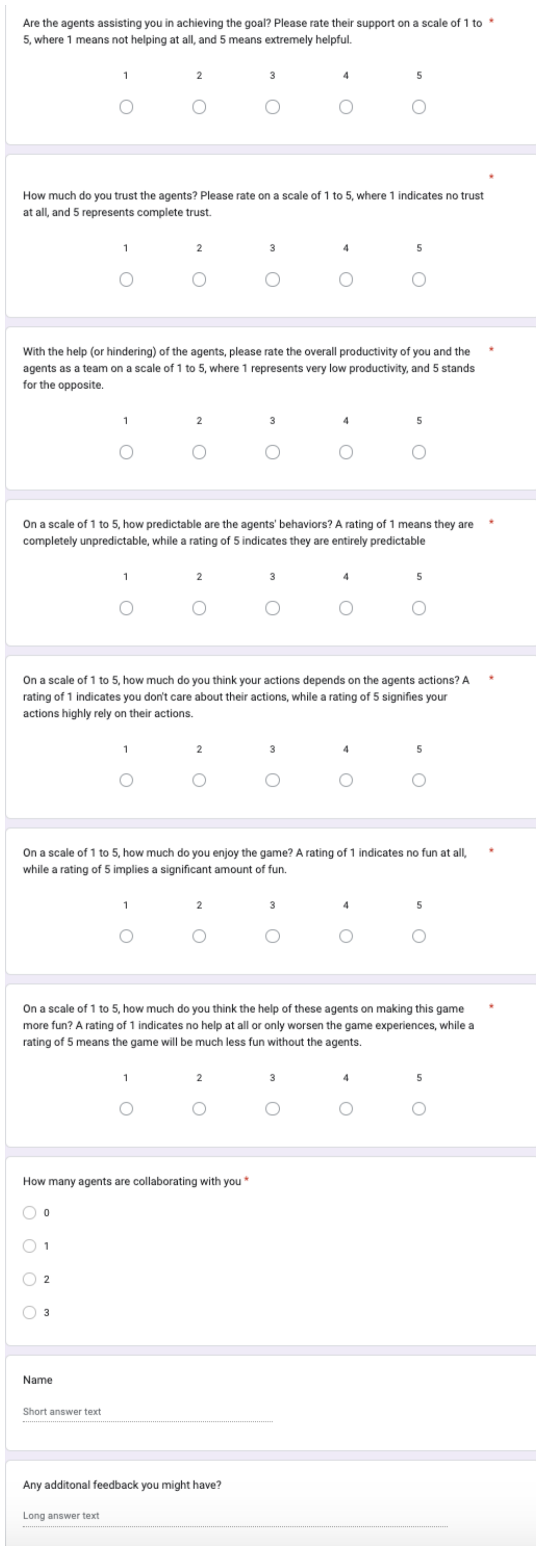
(c) Human evaluation example



(d) Human instructions

Figure 26: Human evaluation interface welcome screen (a), evaluation examples (b)–(c), and instructions to the human participants (d).

F.2 HUMAN EVALUATION QUESTIONNAIRE



Are the agents assisting you in achieving the goal? Please rate their support on a scale of 1 to 5, where 1 means not helping at all, and 5 means extremely helpful. *

1 2 3 4 5

How much do you trust the agents? Please rate on a scale of 1 to 5, where 1 indicates no trust at all, and 5 represents complete trust. *

1 2 3 4 5

With the help (or hindering) of the agents, please rate the overall productivity of you and the agents as a team on a scale of 1 to 5, where 1 represents very low productivity, and 5 stands for the opposite. *

1 2 3 4 5

On a scale of 1 to 5, how predictable are the agents' behaviors? A rating of 1 means they are completely unpredictable, while a rating of 5 indicates they are entirely predictable *

1 2 3 4 5

On a scale of 1 to 5, how much do you think your actions depends on the agents actions? A rating of 1 indicates you don't care about their actions, while a rating of 5 signifies your actions highly rely on their actions. *

1 2 3 4 5

On a scale of 1 to 5, how much do you enjoy the game? A rating of 1 indicates no fun at all, while a rating of 5 implies a significant amount of fun. *

1 2 3 4 5

On a scale of 1 to 5, how much do you think the help of these agents on making this game more fun? A rating of 1 indicates no help at all or only worsen the game experiences, while a rating of 5 means the game will be much less fun without the agents. *

1 2 3 4 5

How many agents are collaborating with you *

- 0
- 1
- 2
- 3

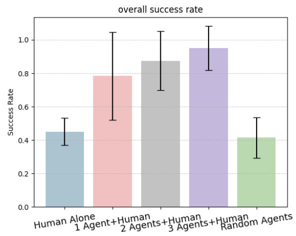
Name

Short answer text

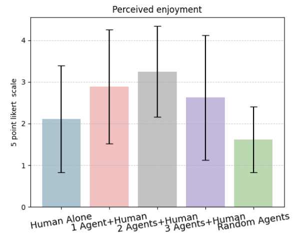
Any additonal feedback you might have?

Long answer text

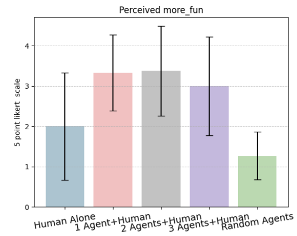
F.3 Additional Results on Human Evaluation



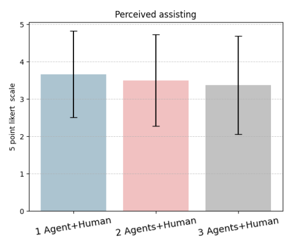
(a) **Collaboration score:** The collaboration score is higher if more agents are collaborating with human players, although the difference is not significant.



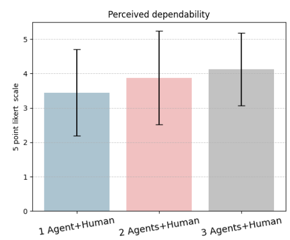
(b) **Perceived enjoyment:** Humans enjoy the game more if they collaborate with the right number of agents.



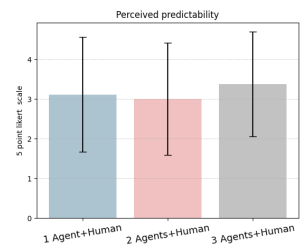
(c) **Perceived more fun:** Players enjoy the game more because of collaboration with competent agents.



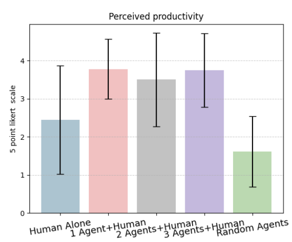
(d) **Perceived Assisting:** There is no significant difference in terms of human perceptions of helpfulness when collaborating with more agents, even though the task success rate is higher.



(e) **Perceived dependability:** When collaborating with more agents, players depend on the agents more.



(f) **Perceived Predictability:** There is no difference in terms of the predictability of agent behaviors when collaborating with more agents.



(g) **Perceived productivity:** Players think collaborating with AI agents will improve productivity.

(h) **Perceived Trust:** There is no difference in terms of trust when collaborating with more agents.

Figure 27: Full results of human evaluations