
MineLand: Simulating Large-Scale Multi-Agent Interactions with Limited Multimodal Senses and Physical Needs

Xianhao Yu ^{*1}Jiaqi Fu ^{*1}Renjia Deng ^{*1}Wenjuan Han ^{✉1}
¹Beijing Jiaotong University
wjh@bjtu.edu.cn

Abstract

While Vision-Language Models (VLMs) hold promise for tasks requiring extensive collaboration, traditional multi-agent simulators have facilitated rich explorations of an interactive artificial society that reflects collective behavior. However, these existing simulators face significant limitations. Firstly, they struggle with handling large numbers of agents due to high resource demands. Secondly, they often assume agents possess perfect information and limitless capabilities, hindering the ecological validity of simulated social interactions. To bridge this gap, we propose a multi-agent Minecraft simulator, MineLand, that bridges this gap by introducing three key features: large-scale scalability, limited multimodal senses, and physical needs. Our simulator supports 64 or more agents. Agents have limited visual, auditory, and environmental awareness, forcing them to actively communicate and collaborate to fulfill physical needs like food and resources. Additionally, we further introduce an AI agent framework, Alex, inspired by multitasking theory, enabling agents to handle intricate coordination and scheduling. Our experiments demonstrate that the simulator, the corresponding benchmark, and the AI agent framework contribute to more ecological and nuanced collective behavior. The source code of MineLand and Alex is openly available at <https://github.com/cocacola-lab/MineLand>.

1 Introduction

Multi-agent simulators have facilitated rich explorations of an interactive artificial society that reflects collective behavior. From sandbox games such as Smallville [40] to virtual environments [4, 32, 18], researchers and practitioners have been building open-world simulators that can carry multi-agent behaviors and navigate complex human relationships for decades. Especially with the advent of Large Language Models (LLMs) and Vision-Language Models (VLMs), numerous multi-agent simulators based on these technologies have been at the forefront in various fields, from fundamental research to practical applications, such as watch-and-help (WAH) task [65], Smallville [40] and Overcook games [18]. However, conventional multi-agent open-world simulators, valuable for exploring collective behavior, suffer from limitations (detailed comparison in Table 1). Firstly, they struggle with large-scale scenarios due to the enormous resource consumption required for large-scale agents. Secondly, they are often under the assumption of perfect information and limitless capabilities. These idealized worlds diverge sharply from the messy reality of human interaction.

^{*}Equal contribution. The order of authors was determined randomly.

[✉]Corresponding author.



Figure 1: A panoramic view of one scene in MineLand, consisting of multiple AI agents. Subfigure 3&6 show interactions demonstrating cooperation and competition among several agents. Subfigure 5&2&4 showcases the scenarios where the limited senses, physical needs, and multitasking mechanism reflect. In Subfigure 1, an agent is performing a creative task named *Exploration*. Two agents in the left cave of Subfigure 5 cooperate to finish a programmatic *mining* task, while agents in Subfigure 3 are carrying out *building construction*, which is a hybrid task.

This gap between existing simulators and the real world hinders the ecological validity and richness of social interaction [21].

To bridge this gap, we propose MineLand (Section 2), a multi-agent Minecraft simulator, as shown in Figure 1, by introducing three key features: large-scale scalability, limited multimodal senses, and physical needs. First and foremost, the essence of the MineLand’s features is the ability to handle the maximum number of agents. Compared to two-agent WAH, single-agent MineDojo [16] and twenty-five-agent Smallville [40], our MineLand enables the utilization of sixty-four or even more agents on the mainstream consumer desktop PC. Secondly, our simulator operates under the human-like assumption [21] that agents possess only limited multimodal senses: partially observable environments, eco-centric perspective, and limited visual and auditory senses. This mirrors real-life social interactions, where visibility and audibility can be affected by factors such as distance, terrain, and environment. These limitations restrict information access, forcing agents to actively communicate to compensate for sensory deficiencies. Thirdly, we integrate realistic physical needs into agents. Agents require fundamental physical needs, such as food, sustenance, and resource management, which adds a time-based aspect to their daily routine procedures. This necessitates collaboration and competition for resources, mirroring the complex interplay of cooperation and self-interest observed in human societies [15, 1]. By incorporating these three features, our simulator fosters the emergence of dynamic and ecologically valid multi-agent interactions¹.

As an open-world multi-agent simulator, MineLand is an excellent platform for benchmarking multi-agent capabilities (Section 3), hence we crowd-sourced datasets to fully evaluate the potential of LLM- or VLM-based multi-agents. For previously existing tasks, programmatic tasks (e.g., harvest, tech tree, combat tasks) and creative tasks (e.g., survival tasks), we offer a significantly expanded task quantity. Specifically, we crowd-sourced 4499 programmatic tasks and 1536 creative tasks, which is 2 times compared to MineDojo. Interestingly, we introduce a novel “hybrid task” category that combines the features of programmatic tasks and creative tasks. Construction and Stage Performance tasks exemplify this new category, bringing the total to 18 hybrid tasks. Additionally, the simulator

¹ Ecological validity refers to interactions between agents within a simulated environment that closely resemble real-world human interactions. For example, actions are situated, adaptive, and environmentally constrained.

Table 1: Part of the comparison with related popular platforms or projects. The full table is in Table 7. *Max Agents*: The approximate maximum number of agents supported on a single PC (See Section 5.1 for details). *Human*: Whether the simulator allows humans to directly interact with AI agents. *Plan as Action*: Whether agents can generate plans in a specific format (e.g., code) that the simulator interprets and executes directly as actions. *Sociological Experiments*: Whether the simulator is designed to facilitate the study of social phenomena and emergent behavior in multi-agent systems.

Simulator	Max Agents	Human	Plan as Action	Sociological Experiments	Number of Tasks
MineLand	64+	✓	✓	✓	6000+
MineDojo [16]	1	-	-	-	3000+
MineRL v1.0 [3]	1	-	-	-	5
MarLÖ [42]	8	✓	-	-	14
Malmo [27]	8	✓	-	-	-
Voyager [57]	1	✓	✓	-	-
Smallville [40]	25	-	✓	✓	-

allows customization of the number of agents and facilitates exploration through two distinct modes: cooperative and competitive.

In addition, we design an AI agent framework - **Alex** - inspired by Multitasking theory from the Cognition field [53] (Section 4). **Alex** allows for simultaneously executing intricate coordination and scheduling with multiple tasks. With this AI agent framework, we have obtained the following intriguing findings: (1) Our simulator shows the feature of large-scale scalability (supporting 64 agents, x8 times than before, (§5.1), limited multimodal senses (§J) and physical needs (§K). By incorporating these three features, our simulator fosters social dynamics (§5.2&§O). (2) Our benchmark and datasets are challenging (§5.3&§5.4). (3) Our agents work together more effectively, with a reduced workload per agent (§5.5), and the multitasking mechanism for agents is beneficial (§5.6).

In summary, our contributions are three-fold: the simulator, benchmark, and AI agent. With these contributions, we push the boundaries of multi-agent simulation by bridging the gap between virtual agents and large-scale real-world humans. This not only advances understanding of AI multi-agents but also holds potential for applications in human dynamics, social psychology, robotics, and game design. We anticipate that this work will serve as a useful foundation for the community to create new algorithms and make progress in the field of embodied AI multi-agent systems.

2 MineLand Simulator

Conventional multi-agent open-world simulators suffer from the gap between virtual agents and large-scale real-world humans. To bring this gap, we propose MineLand with three key features: large-scale scalability, limited multimodal senses, and physical needs. This section describes the design and implementation of this simulator, focusing on the architecture, observation space, state space, action space and communication. During introducing the design, we will highlight techniques that bring the three key features and omit the parts similar to other Minecraft simulators.

2.1 Architecture

MineLand, inspired by Malmo [27], Mineflayer [43], MineDojo [16] and Voyager [57], is a Minecraft simulator where players² can explore, and interact with each other as well as the environments. The architecture as shown in Figure 2 consists of three main modules (Details in Section C.1): the Bot Module, the Environment Module, and the Bridge Module.

What technology has caused the feature of large-scale scalability? Existing Minecraft simulators (e.g., Malmo, MineRL, MineDojo) necessitate running a Minecraft game client for each player. This client-based approach comes with a notable drawback - it incurs substantial resource costs and

²In this work, we use the term “player” to refer to both human players and AI agents. When we mention “agent”, we specifically mean AI agents. Humans have the option to access the game either through VR or using a keyboard.

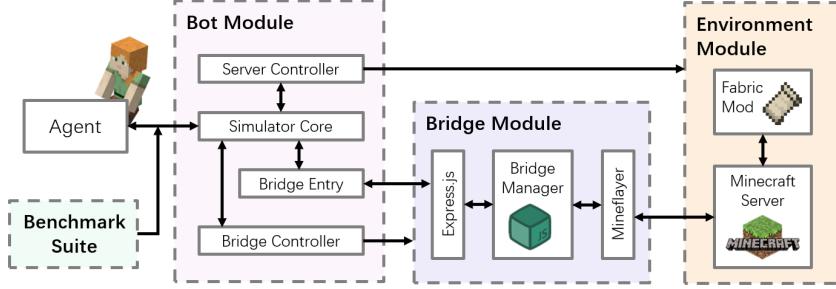


Figure 2: Illustration of the architecture of MineLand.

most machines cannot handle running large-scale Minecraft game clients concurrently. In contrast, MineLand adopts a different approach. It simplifies each Minecraft client into a single thread, optimizing performance overhead caused by multiple clients. With MineLand, adding one more agent only requires one more thread. Based on this new technique, MineLand supports 64 or more agents on a mainstream consumer desktop PC, which is a substantial improvement compared to other Minecraft platforms that can only support up to 2 agents. We conducted relevant experiments in Section 5.1.

2.2 Observation Space

The observation space is designed to be compatible with almost all APIs of the popular MineDojo framework. MineLand provides sensor information for players: tactile information (information about the blocks surrounding the agent, which represent the objects that the agent can touch), auditory information, and visual information (RGB video from the first-person perspective of the agent). These three modalities (namely, touch, vision, and hearing) together provide the agents with multimodal senses. Note that this information is all raw perceptual information.³

What technology has caused the feature of limited multimodal senses? We refer to the mechanisms of human vision and hearing, and impose limitations on the sensor perception of players, including distance attenuation, environmental obstructions, and directional constraints, to model the limited senses.

2.3 State Space

Previous simulators focus on task-oriented activities, thus the state space focuses on the inventory and equipment. We use a state space that blends task-oriented activities with the rhythms of daily life.

What technology has caused the feature of physical needs? Basic physical needs are the foundation that leads to daily life behavior. For the rhythms of daily life, we define the states of agents for themselves: physical needs like oxygen and hunger. Blending the rhythms of daily life with task-oriented activities is what makes this simulator stand out. Imagine agents waking up in their virtual Minecraft homes, engaging in daily routines like cooking to satisfy physical food needs, but also having defined jobs (e.g., lumberjack, farmer) that involve specific task-oriented activities. This creates a natural flow between daily life and goal-driven behavior, providing a more realistic and nuanced environment for studying agent interactions and complex social dynamics.

2.4 Action Space

The simulator offers a unique action space encompassing both low-level and high-level actions. For the low-level actions, MineLand includes basic actions like walking, running, jumping, and interacting with objects. The low-level actions are the same as the traditional action space in gym-style API. We also support Reinforcement Learning based on low-level actions (For more details about experiments of Reinforcement Learning method, please refer to Section I. High-level actions,

³Besides the raw perceptual information, MineLand also provides the events encountered by the agent, such as injury, death, and others. Injury events can also be regarded as tactile information, but they are presented in the form of events for simplicity.

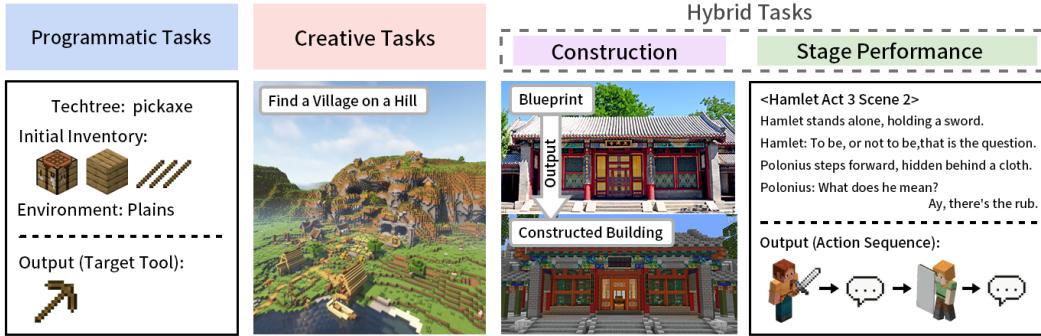


Figure 3: Illustration of Tasks. We have expanded the number of programmatic tasks and creative tasks by 2 times, compared to MineDojo. Additionally, we have introduced novel hybrid tasks that combine the features of programmatic tasks and creative tasks. Customizing the number of players is supported. For multi-agents, we provide two modes: cooperative mode and competitive mode.

like dodging obstacles and manipulating tools, are suitable for complex tasks that consist of several low-level actions and require longer computation times. The high-level actions are implemented in the form of code, following Voyager [57]. Imagine agents navigating the world, dodging obstacles, and manipulating tools. These complex tasks generate an action sequence, allowing the simulator to continue executing the low-level actions, skip some steps earlier, or be interrupted by some special event.

2.5 Communication

Diversity Diverse communication strategies can lead to the emergence of more realistic behaviors within the simulated environment. We design three communication strategies including auditory information, body language (via visual perception), and sharing information in text media.

What communication technology has caused the feature of large-scale scalability? For multi-agent simulators, an efficient communication mechanism is crucial, especially in situations with large-scale scalability. Traditional communication methods, like centralized broadcasting to all agents, can become computationally expensive and slow down the simulation with a large number of agents. Here we design a distance-constrained constrained communication mechanism. If an agent wants to communicate with other agents, it chats through Minecraft’s message bar. Only when the distance between other agents and the sending agent is less than a certain threshold, will other agents receive messages. Similarly, communications via auditory information and body language are limited by distance.

How is the interrupt mechanism implemented? Most importantly, the new message is allowed to interrupt the executing code and execute this message directly before the previous code has ended. We term this as the interrupt mechanism. With this mechanism, even if an agent is working on a 5-minute extension (such as mining), it is still feasible for other agents to communicate with this working agent at any time. This interrupt mechanism was not supported in the previous work, such as Voyager [57]. Next, we will introduce how to implement it. For a high-level action, the execution of the code is divided into several steps, with each step lasting 50-200 milliseconds⁴. Before taking a step, the agent is provided with the running states of the previous code, either *running*, *ready*, or *exceptions*. After completing a step, the agent, based on the running states, can choose to either switch to a new action code or continue executing the previous code. This function of choosing is implemented by an automatic gate control system with two gates: *New* and *Resume*. *New* means the agent wants to switch to a new code in the following steps. *Resume* indicates that the agent wants to continue executing the previous code. In this way, the agent can complete a code that needs to be executed for a long period, or be interrupted at an appropriate time. You may refer Section E.4 for an example.

⁴50 milliseconds is the minimum time unit in Minecraft. We refer to this minimum time unit as a “tick”.

3 MineLand Benchmark Suite and Dataset

MineLand, as a large-scale multi-agent simulator, push the boundaries of multi-agent capabilities by enabling them to tackle complex human-like planning tasks within diverse environments. However, evaluating these advanced planning abilities necessitates sophisticated benchmarks. To address this challenge, we propose a new benchmark suit, MineLand benchmark. MineLand benchmark surpasses existing benchmarks by offering significantly more tasks (doubling programmatic and creative tasks compared to MineDojo) and introducing a novel “hybrid task” category that combines the features of programmatic and creative tasks (including Construction Tasks and Stage Performance Tasks). Additionally, this benchmark allows for flexible player numbers and exploration through cooperative and competitive modes. Competitive mode can be used to measure the differences in capabilities between different AI agents, as well as to develop adversarial learning algorithms. Refer to Section D for more details.

3.1 Programmatic Task

We follow MineDojo [16] for the design of programmatic tasks. Each Task T is defined as a 5-tuple: $T = (G, \mathcal{G}, \mathcal{I}, f_{suc}, S)$. G refers to the task goal that needs to be completed. \mathcal{G} is guidance. \mathcal{I} is the initial condition of the task. f_{suc} is the Success Criterion. S is a set of parameters that could be customized. Different from MineDojo, these parameters include the number of agents, cooperative mode, competitive mode, etc. In total, MineLand has 4499 programmatic tasks.

3.2 Creative Task

Creative tasks is defined by a 4-tuple: $T = (G, \mathcal{G}, \mathcal{I}, S)$. There are 1536 creative tasks in total, and is compatible with tasks in MineDojo.

3.3 Hybrid Task

Hybrid tasks do not have a unique ground truth but have several references⁵. We represent the hybrid task as $T = (G, \mathcal{G}, \mathcal{I}, \mathcal{D}, f_{score}, S)$, where \mathcal{D} denotes the references. Unlike programmatic tasks, because Hybrid tasks do not have a ground truth, MineLand will return a score of f_{score} based on \mathcal{D} . The higher the score, the better the task is completed. We design two types of tasks: Construction and Stage Performance, for hybrid tasks.

Construction Tasks Given a blueprint for a building or scene, the Construction Task aims to build these buildings or scenes based on the blueprint. The blueprint, either pictures in real life or Minecraft-style pictures, is the reference \mathcal{D} .

Evaluation Metrics. MineLand gives a score based on whether the constructed buildings meet the blueprint’s expectations. We calculate the task scores through the VLM-based evaluations and human evaluation, which both use the same criteria (refer to Section Q for more details). The evaluation scores range from 1 to 5, with higher scores indicating that the agents’ constructions more closely resemble the intended blueprint.

Stage Performance Tasks Given a script of a drama consisting of several behaviors, which may be a single action or an emotional expression, the Stage Performance Task aims to perform the script with agents as actors.

Evaluation Metrics. We leverage two complementary evaluation metrics: an LCS (Longest Common Subsequence)-based metric and human evaluation. The LCS-based metric consists of two distinct scores: the keypoint score and the appropriateness score. We present the formulas for adding up these two scores:

$$f_{key} + f_{appro} = \frac{|LCS(SEQ_{Agent}, SEQ^*)|}{|SEQ^*|} + \frac{|LCS(SEQ_{Agent}, SEQ^*)|}{|SEQ_{Agent}|} \quad (1)$$

where SEQ_{Agent} represents the action sequence generated by the agent, while SEQ^* is the ground truth. $LCS(A, B)$ denotes the LCS between A and B . $|A|$ is the length of sequence A . The

⁵Hybrid tasks resemble translation tasks in that, while a single unique translation may not exist, multiple reference translations can guide the process. Here references could be key rules, constraints, and key evaluation indicators, etc.

keypoint score emphasizes the completeness of the enacted behaviors, ensuring all crucial actions and expressions are performed. The appropriateness score goes beyond completeness to evaluate the overall coherence and naturalness of the performance, considering how well the behaviors flow together and align with the script's dramatic intent. The human evaluation score is an integer between 1 and 5, with higher scores indicating better performance. It provides a comprehensive assessment of the agents' ability to not only execute actions accurately but also to deliver them naturally and engagingly. Details of the human evaluation criteria can be found in Section R.

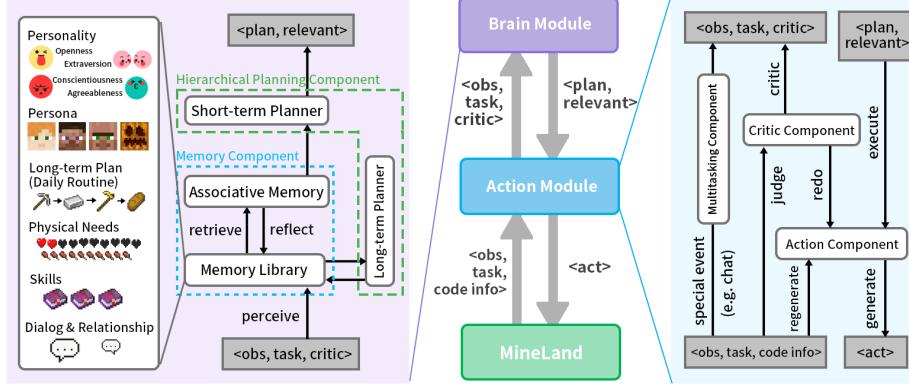


Figure 4: Illustration of the architecture of Alex.

4 Alex Agent

To truly demonstrate the challenge of this benchmark, we propose Alex⁶, a VLM-based agent as shown in Figure 4. Conventional LLM-powered AI agents depend on LLM to operate as its brain, which is backed by several vital components that perform various essential functions. These components, such as the memory component, planning component, and acting component, have been thoroughly studied recently. To cater to our specific requirements, we have improved these three components (refer to Section E for more details), replaced LLM with VLM and introduced one new component: the multitasking component. Additionally, Alex exhibits different personality traits predefined in the system prompt.

Multitasking Component People often switch attention between tasks, for example cooking while talking. The ability to communicate smoothly with other players while working on a task-oriented action is crucial in multi-agent scenarios. Therefore, we develop the mechanism of multitasking ability to enhance the agent's attention control and working memory abilities inspired by the Multitasking theory from the Cognition field [53]. Specifically, for attention control, the interrupt mechanism effectively controls attention among multiple tasks. For working memory, Alex maintains and processes information in the Memory Library. When another agent says hello to the working agent, this involves saving and restoring internal states when frequent and high-speed switching between communication activities and goal-driven working actions to avoid forgetting ongoing tasks. With the multitasking mechanism, Alex allows for simultaneously simulating and executing intricate coordination and scheduling with multiple tasks.

5 Experiments

5.1 Experiments of Simulators Performance

We evaluate the number of agents that MineLand can support and compare MineLand with other popular Minecraft simulators. We utilize a mainstream consumer desktop PC equipped with an Intel i5-12400F CPU and 64GB of memory. Performance Monitor is employed to monitor the process. Our findings reveal that MineLand is capable of supporting 32 agents simultaneously while

⁶Alex is the protagonist in the sandbox game Minecraft, one of the default skins for players and a character in the game: <https://www.minecraft.net/zh-hans>. To pay tribute, we named our proposed AI agent Alex.

providing visual displays. When visual display is disabled, the number of concurrently running agents increases to $\times 4$ times. Furthermore, as depicted in Table 2, when MineLand and Malmo both run 8 agents, MineLand’s CPU and memory usage are approximately 1/3 that of Malmo’s (specifically, 35.6% and 38.0%, respectively). It is worth noting that Malmo serves as the foundation for most popular Minecraft Platforms (e.g., MineDojo/MineRL/MarLÖ [42]), thus highlighting MineLand’s superior performance compared to the vast majority of existing Minecraft Platforms. Consequently, MineLand proves to be highly suitable for multi-agent environments.

5.2 Experiments of Social Dynamics

In the “unlocking tools” task with two agents (Table 3), two agents in the collaborative mode work together effectively, with a reduced workload per agent and higher communication expenses. On the other hand, two competitive agents worked independently and necessitated fewer code iterations. The primary reason is that, in competitive relationships, agents tend to achieve more in a single iteration to expedite progress and outperform their opponents. However, this results in less thorough planning and more code errors. Consequently, multiple agents in competitive relationships require fewer code iterations but make more mistakes.

We also observe that personality plays a significant role in determining the behavior of agents in multi-agent societies [25]. We assigned the personality traits of high extraversion and agreeableness to both agents. Under this condition, the agents tended to establish collaboration and engage in mutual communication (co-op >8 out of 10 times). This result is consistent with human behavior [13]. When no personality was set for the agents, they worked independently (co-op 0 out of 10 times). For more experiments of simulating sociological phenomena with >10 agents, please refer to Section O.

Table 2: Part of the comparison table of performance of Minecraft simulators. The full table is in Table 9. $\text{MineLand}_{w/o \text{ vision}}$ means MineLand without vision. CPU and Mem represent the average CPU time and memory usage during the initialization phase and 5-minute run respectively.

Simulator	Agents	CPU	Mem
MineLand	8	2.81%	7.07GB
MineLand	32	5.64%	19.65GB
$\text{MineLand}_{w/o \text{ vision}}$	8	1.88%	2.94GB
$\text{MineLand}_{w/o \text{ vision}}$	64	2.87%	6.30GB
$\text{MineLand}_{w/o \text{ vision}}$	128	4.00%	9.04GB
Malmo	8	7.90%	18.63GB

5.3 Experiments of Construction Tasks

To establish a baseline for the research community, we evaluate our un-finetuned **Alex** on two example construction tasks: “Monument Construction” and “Stone Stele Construction” (detailed in Section M and Table 4). While **Alex** demonstrates a promising capability in selecting appropriate materials for both tasks, its overall construction abilities are currently limited. We posit two key reasons for these limitations: complex task planning and high-precision manipulation. For the first reason, construction tasks are inherently time-consuming and require sophisticated planning abilities. For instance, building auxiliary structures to facilitate the main construction process is crucial for achieving significant height. However, **Alex** cannot currently plan and construct such auxiliary structures, hindering its ability to construct tall structures. The second reason is the high-precision manipulation. The construction tasks necessitate high-precision manipulation, such as adding decorative details. **Alex** primarily utilizes high-level actions and lacks the necessary APIs to execute these fine-grained manipulations effectively. These results show the challenge of this task.

5.4 Experiments of Stage Performance Tasks

We evaluate our un-finetuned **Alex** on four example stage performance tasks (Table 5). We found that in single-agent tasks, the agent typically exhibits high scores across all three evaluation metrics. However, in multi-agent tasks, the keypoint score tends to be higher than the appropriate score

because the agent can understand the script and fulfill the key requirements. In contrast, the human evaluation scores are usually relatively lower because agents often engage in unnecessary dialogue in scripts that require collaboration, resulting in lower scores.

Table 4: Human evaluation score and VLM-based (gpt-4o) evaluation score of Alex agent on two construction tasks. We conducted three evaluations using the VLM and obtained consistent results. Details in Section Q. VLM: VLM-based score. Human: Human evaluation score.

Construction	Score	
	VLM	Human
A Monument	4, 3, 3	3
A Stone Stele	1, 1, 1	1

Table 5: Appropriateness score (Appro.), keypoint score (Key.), and Human Evaluation Score (Human.) of Alex Agent across four stage performance tasks. The number in parentheses following the script name represents the number of agents in that script. Appropriateness score and keypoint score are real numbers in the range [0, 1], and the human evaluation score is an integer between 1 and 5. Details in Section N and Section R.

Script Name	Score		
	Appro.	Key.	Human.
Cook food(1)	1.00	1.00	5
Exchange items(2)	0.59	0.99	4
Make friends(3)	0.67	0.98	3
Romeo and Julia, Act I Scene I(13)	0.09	0.20	1

5.5 Experiments of Multi-Agent Cooperation

To validate the cooperation efficiency of our agent framework, we conduct the “unlocking tools” task with two agents. We observed that agents in a cooperative relationship required more code iterations to finish the task, primarily because most of these iterations were dedicated to establishing and maintaining communication, as well as task allocation. For example, when one agent says in a chat that he needs two sticks, another agent will ask for getting together near the table, and then give the sticks to him. However, the actual workload for each agent is reduced without considering the chat cost. Compared to agents working independently, the code iteration cost of agents cooperating is reduced by 20% per agent.

5.6 Experiments of Multitasking

To validate the impact of multitasking support in the simulator⁷, we conduct the obsidian mining task, which takes over 8 minutes and requires multiple steps to complete. In the beginning, the agent is mining and encounters chat or hurt events within 8 minutes. **Chat event:** Another agent nearby initiates a conversation with Alex. This is a low-priority event, and Alex can choose whether to respond to the agent nearby. **Hurt event:** The agent gets hurts. For example, a zombie attacks the agent. This is a high-priority event, requiring Alex to stop its current task and address this event first.

We expect the two types of events to activate the multitasking component and the agent processes the mining and the special event simultaneously. Results are in Table 6. In all ten runs, we count the number of successfully handling multitasks. For the hurt event, if the agent fights back due to this hurt event, it is considered successful. For the chat event, if the agent responds due to the chat event, it is considered successful. Results reveal that $\text{Alex}_{w/o\ mt}$ can’t process events timely, resulting in Alex being killed by the zombie. In contrast, Alex with a multitasking component is capable of managing multiple events (e.g., mining while chatting), autonomously determining their priority, and addressing the higher-priority events first. Hence, multitasking is an essential mechanism.

Table 6: Comparison between Alex and $\text{Alex}_{w/o\ mt}$ (Alex without the multitasking component).

Agent	Hurt event	Chat event
Alex	8/10	2/10
$\text{Alex}_{w/o\ mt}$	0/10	0/10

⁷This multitasking functionality is achieved through the synergistic interplay of two keys: the multitasking component in Alex and the interrupt mechanism in MineLand.

6 Conclusion

Traditional multi-agent simulators struggle with large-scale scenarios and often assume perfect information and unrealistic agent capabilities. To address these limitations, we introduce MineLand, a novel Minecraft-based simulator supporting 64 or more agents with limited senses and physical needs. This forces agents to actively communicate and collaborate, fostering more ecologically valid interactions. The advantage carries potential broader impacts across various domains as discussed in Section B.

References

- [1] Clayton P Alderfer. An empirical test of a new theory of human needs. *Organizational behavior and human performance*, 4(2):142–175, 1969.
- [2] Jitao Bai, Simiao Zhang, and Zhonghao Chen. Is there any social principle for llm-based agents? *arXiv preprint arXiv:2308.11136*, 2023.
- [3] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos, 2022.
- [4] Joseph Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, 1994.
- [5] Marcel Binz and Eric Schulz. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120, 2023.
- [6] Woody Bledsoe. I had a dream: Aaa presidential address. *AI Magazine*, 7(1):57–61, 1986.
- [7] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. *arXiv preprint arXiv:2301.10034*, 2023.
- [8] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13734–13744, June 2023.
- [9] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012.
- [10] Stuart K Card, Thomas P Moran, and Alan Newell. The psychology of human-computer interaction. 1983.
- [11] Jiaqi Chen, Yuxian Jiang, Jiachen Lu, and Li Zhang. S-agent: self-organizing agents in open-ended environment. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.
- [12] Antônio Carlos da Rocha Costa. *A Variational Basis for the Regulation and Structuration Mechanisms of Agent Societies*. Springer, 2019.
- [13] Boele De Raad. *The big five personality factors: the psycholinguistic approach to personality*. Hogrefe & Huber Publishers, 2000.
- [14] Kevin Dill and L Martin. A game ai approach to autonomous control of virtual characters. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (IITSEC'11)*, Orlando, FL, USA, 2011.
- [15] Len Doyal and Ian Gough. A theory of human needs. *Critical Social Policy*, 4(10):6–38, 1984.
- [16] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.
- [17] Yicheng Feng, Yuxuan Wang, Jiazheng Liu, Sipeng Zheng, and Zongqing Lu. Llama rider: Spurring large language models to explore the open world, 2023.

- [18] Ran Gong, Qiuyuan Huang, Xiaojian Ma, Hoi Vo, Zane Durante, Yusuke Noda, Zilong Zheng, Song-Chun Zhu, Demetri Terzopoulos, Li Fei-Fei, et al. Mindagent: Emergent gaming interaction. *arXiv preprint arXiv:2309.09971*, 2023.
- [19] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.
- [20] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations, 2019.
- [21] John Heil. Perception and cognition. 1983.
- [22] James D. Hollan, Edwin L. Hutchins, and Louis Weitzman. Steamer: An interactive inspectable simulation-based training system. *AI Magazine*, 5(2):23–36, 1984.
- [23] John J. Horton. Large language models as simulated economic agents: What can we learn from homo silicus?, 2023.
- [24] Guangyuan Jiang, Manjie Xu, Song-Chun Zhu, Wenjuan Han, Chi Zhang, and Yixin Zhu. Evaluating and inducing personality in pre-trained language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [25] Guangyuan Jiang, Manjie Xu, Song-Chun Zhu, Wenjuan Han, Chi Zhang, and Yixin Zhu. Evaluating and inducing personality in pre-trained language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [26] Bonnie E John and David E Kieras. The goms family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3(4):320–351, 1996.
- [27] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. The malmo platform for artificial intelligence experimentation. In *Proc. 25th International Joint Conference on Artificial Intelligence*, page 4246, Palo Alto, California USA, 2016. AAAI Press.
- [28] Randolph M Jones, John E Laird, Paul E Nielsen, Karen J Coulter, Patrick Kenny, and Frank V Koss. Automated intelligent pilots for combat flight simulation. *AI Magazine*, 20(1):27–42, 1999.
- [29] Patrik N Juslin, Klaus R Scherer, J Harrigan, and R Rosenthal. Vocal expression of affect. *The new handbook of methods in nonverbal behavior research*, pages 65–135, 2005.
- [30] Julia Kiseleva, Ziming Li, Mohammad Aliannejadi, Shrestha Mohanty, Maartje ter Hoeve, Mikhail Burtsev, Alexey Skrynnik, Artem Zholus, Aleksandr Panov, Kavya Srinet, Arthur Szlam, Yuxuan Sun, Katja Hofmann, Michel Galley, and Ahmed Awadallah. Neurips 2021 competition iglu: Interactive grounded language understanding in a collaborative environment, 2021.
- [31] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.
- [32] John Laird and Michael VanLent. Human-level ai’s killer application: Interactive computer games. *AI Magazine*, 22(2):15, 2001.
- [33] Kalyani Lakkanige, Lamar Cooley-Russ, Alan R. Wagner, and Sarah Rajtmajer. Exploring trust and risk during online bartering interactions, 2023.
- [34] Joel Z. Leibo, Edgar Dué nez Guzmán, Alexander Sasha Vezhnevets, John P. Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charles Beattie, Igor Mordatch, and Thore Graepel. Scalable evaluation of multi-agent reinforcement learning with melting pot. *PMLR*, 2021.
- [35] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 455–465. PMLR, 08–11 Nov 2022.

- [36] Hao Li, Xue Yang, Zhaokai Wang, Xizhou Zhu, Jie Zhou, Yu Qiao, Xiaogang Wang, Hongsheng Li, Lewei Lu, and Jifeng Dai. Auto mc-reward: Automated dense reward design with large language models for minecraft, 2024.
- [37] Shaoteng Liu, Haoqi Yuan, Minda Hu, Yanwei Li, Yukang Chen, Shu Liu, Zongqing Lu, and Jiaya Jia. RL-gpt: Integrating reinforcement learning and code-as-policy, 2024.
- [38] Chris Madge and Massimo Poesio. Large language models as minecraft agents, 2024.
- [39] OpenAI. Gpt-4 technical report, 2024.
- [40] Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.
- [41] Joon Sung Park, Lindsay Popowski, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Social simulacra: Creating populated prototypes for social computing systems. In *In the 35th Annual ACM Symposium on User Interface Software and Technology (UIST ’22)*, UIST ’22, New York, NY, USA, 2022. Association for Computing Machinery.
- [42] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D. Gaina, and Daniel Ionita. The multi-agent reinforcement learning in malmö (marlö) competition, 2019.
- [43] PrismarineJS. mineflayer. <https://github.com/PrismarineJS/mineflayer>, 2023.
- [44] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.
- [45] 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. In *International Conference on Learning Representations*, 2020.
- [46] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration, 2020.
- [47] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023.
- [48] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.
- [49] Yiran Qin, Enshan Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. Mp5: A multi-modal open-ended embodied system in minecraft via active perception, 2024.
- [50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [51] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.
- [52] Mark O. Riedl. Interactive narrative: A novel application of artificial intelligence for computer games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI’12)*, pages 2160–2165, 2012.
- [53] Dario D Salvucci and Niels A Taatgen. Threaded cognition: an integrated theory of concurrent multitasking. *Psychological review*, 115(1):101, 2008.
- [54] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D’Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page accepted. IEEE, 2021.

- [55] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [56] Milind Tambe, W Lewis Johnson, Randolph M Jones, Frank Koss, John E Laird, Paul S Rosenbloom, and Karl Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1):15, 1995.
- [57] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- [58] Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv: 2311.05997*, 2023.
- [59] 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*, 2023.
- [60] Ross Williams, Niyousha Hosseinichimeh, Aritra Majumdar, and Navid Ghaffarzadegan. Epidemic modeling with generative agents. *arXiv preprint arXiv:2307.04986*, 2023.
- [61] S Wimmer, A Pfeiffer, and N Denk. The everyday life in the sims 4 during a pandemic. a life simulation as a virtual mirror of society? In *INTED2021 Proceedings*, pages 5754–5760. IATED, 2021.
- [62] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [63] Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, et al. Kola: Carefully benchmarking world knowledge of large language models. *arXiv preprint arXiv:2306.09296*, 2023.
- [64] Chi Zhang, Penglin Cai, Yuhui Fu, Haoqi Yuan, and Zongqing Lu. Creative agents: Empowering agents with imagination for creative tasks, 2023.
- [65] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.
- [66] Zhonghan Zhao, Wenhao Chai, Xuan Wang, Li Boyi, Shengyu Hao, Shidong Cao, Tian Ye, Jenq-Neng Hwang, and Gaoang Wang. See and think: Embodied agent in virtual environment, 2023.
- [67] Zhonghan Zhao, Kewei Chen, Dongxu Guo, Wenhao Chai, Tian Ye, Yanting Zhang, and Gaoang Wang. Hierarchical auto-organizing system for open-ended multi-agent navigation, 2024.
- [68] Enshen Zhou, Yiran Qin, Zhenfei Yin, Yuzhou Huang, Ruimao Zhang, Lu Sheng, Yu Qiao, and Jing Shao. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control, 2024.
- [69] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.
- [70] Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, et al. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*, 2023.

A Related Work

A.1 Multi-Agent Simulator

As the popularity of AI agent research continues to grow, there has also been a focus on studying multiple AI agents as well as their cooperation and competition. Researchers and practitioners imagine a dynamic artificial society where human interactions can be simulated by trustworthy agents [12]. From two individuals [9, 45], through four individuals [61], to sandbox games Smallville with twenty-five individuals [40], we witness how individuals perceive a simulated society as the backdrop and interact with the agents and people who engage with it. Each individual can be portrayed through a program, a real human, or an agent based on LLM [40]. The interaction between individuals plays a role in shaping social behavior, leading to simulation of the society. Simulating larger societies can be advantageous. Increasing the number of agents can lead to greater specialization, enabling the accomplishment of more complex and larger-scale tasks. This can significantly improve task efficiency, such as in software development tasks [48]. Additionally, such simulation of interaction has had a significant effect in many other fields. For example, it can replicate realistic social phenomena [14, 41], enhance social robots [4, 6]. They can also be used to test social science theories [5, 24, 23], create model human processors for theory and usability testing [10, 26], train people on how to handle rare yet difficult interpersonal situations [56, 28, 22], and support game characters [32, 52]. There are also many non-Minecraft and non-open-world simulators, such as Habitat3.0 [47], VirtualHome [44, 46], SAPIEN [62], AI2-THOR [31], and iGibson [35, 54].

Challenges of Scaling Up the Number of Agents. While increasing the number of agents can improve task efficiency and make multi-agent simulations more realistic [48, 40, 60], current research primarily focuses on a small number of agents [40, 2, 70]. This is mainly due to the challenges of scaling up the number of agents. Deploying a large number of AI agents will result in an increased computational burden, necessitating better architectural design and computational optimization [40]. Most research in terms of AI agents mimicking daily life routines, focused on two agents [45]. Simulators that remind people of sandbox games (The Sims) initially support four individuals [61], then are extended to twenty-five individuals by [40].

Challenges of Limited Multimodal Senses. To ensure the authenticity of the simulation, an ideal multi-agent simulator should operate under the fundamental assumption that agents possess only limited multimodal senses like humans [21]. Limited multimodal senses mean partially observable environments and an eco-centric perspective. Limited visual and auditory senses restrict information access, forcing agents to actively navigate and communicate to compensate for sensory deficiencies. This mirrors real-life social interactions, where visibility and audibility can be affected by factors such as distance, terrain, and context [29]. As the number of agents grows, the challenges of limited multimodal senses become quite difficult. This is because the communication network of the entire system becomes highly intricate. For agents in our MineLand, the video input is an eco-centric perspective (first perspective) instead of the third perspective in [40], which is omniscient and unrealistic.

Challenges of Physical Needs. Multi-agent simulation platforms hold immense potential for exploring and understanding human social dynamics. However, existing paradigms often disregard the human needs [15, 1]. Existing simulators are designed to simulate believable human behavior in daily-life activities [40, 2]. These activities include waking up, cooking breakfast, heading to work, and initiating conversations with others. However, they do not define their physical needs. For example, after a certain amount of time has passed, the agent will become hungry and have the desire to cook. This desire then leads to the next actions. In this way, the action of cooking is motivated by real desires instead of a predefined schedule.

We incorporate practical physical requirements into the agent model. Agents have basic physical needs: sleep, food, and resource management, which introduces an engaging time-based element to their daily routine processes. This encourages collaboration and competition for resources, reflecting the intricate balance of cooperation and self-interest seen in human societies.

A.2 Multi-Agent Simulator w.r.t Minecraft

Minecraft, the beloved sandbox game, has been a valuable platform for researchers exploring various fields, including artificial intelligence and multi-agent systems, because of its open world and diverse mechanics.

Specifically, the flexibility and richness of Minecraft make it perfect for developing multi-agent simulators. Researchers have the ability to create various custom environments and scenarios within the game world, where they can introduce virtual agents with specific goals and capabilities. These agents can then interact with each other and the environment, providing researchers with the opportunity to observe and analyze their behavior in a controlled setting. These simulators can be broadly categorized into two main types: task-oriented simulators and daily-life simulators.

Task-oriented simulators focus on agents achieving specific objectives within a set time frame. Minecraft is regarded as the training ground for AI agents to hone their skills. For example, [19, 16, 68, 37] focuses on exploring the environment and gathering resources like wood and stone, and managing them efficiently to complete tasks like building structures or crafting tools. Agents in [67] are designed to complete navigation tasks. In [11], agents are self-organized to execute collaborative building tasks and resource collection tasks. Agents in [18] must work together to overcome challenges that require joint effort.

Daily-life simulators take a more holistic approach, focusing on the daily lives of agents within a virtual society. [40] simulates the rhythms and routines of daily life. This includes activities such as waking up, cooking breakfast, going to work, forming opinions, observing others, and engaging in conversations. [33] simulates sociological experiments, and examines how risk affects the way people engage in bartering.

A.3 AI Agent with LLMs and VLMs

LLMs or VLMs are commonly utilized to bootstrap the components of the Agent. In particular, LLMs have demonstrated effective performance for task-planning [18, 69, 38, 36, 17], and they possess substantial world knowledge [63]. Team CraftJarvis has also developed a powerful agent, [58], based on their previous projects [59, 7]. Moreover, VLMs like CLIP [50] offer a versatile visual-language representation that aligns with language and enables zero-shot visual recognition capabilities for potential AI agents. Using VLMs in construction tasks has many advantages over existing methods [64]. AI agents with visual information can also complete tasks in human-like ways [49, 66].

B Broader Impact

By including the simulator, benchmark, and agent framework, we allow for research on more authentic and detailed interactions in simulated environments.

Advancing AI Multi-Agent Research The proposed simulator, MineLand, offers a platform for studying agents interacting under more realistic conditions. This can lead to the development of more robust and adaptable AI agents capable of effectively collaborating and navigating complex social scenarios. These advancements hold immense potential for various applications, including human-computer interaction, robotics, and game design.

Understanding Human Dynamics By studying agent interactions within the simulator, users may gain valuable insights into human social dynamics. Analyzing collaboration, communication, and competition in this controlled environment can help us understand real-world social phenomena and predict their potential outcomes.

Ethics Statement This study follows the ethical principles stated in the Declaration of Helsinki. All participants will receive comprehensive information about the nature and objectives of the study and will be required to provide written consent. Participation in this study is voluntary, and participants have the right to withdraw at any time without facing any consequences. The confidentiality and privacy of participants will be safeguarded in accordance with relevant laws and regulations.

C Details of MineLand Simulator

We show the comparison with related popular platforms or projects in Table 7.

Table 7: Comparison with related popular platforms or projects. *Max Agents*: The approximate maximum number of agents supported on a single PC (See Section 5.1 for details). *Human*: Whether the simulator allows humans to directly interact with AI agents. *Plan as Action*: Whether agents can generate plans in a specific format (e.g., code) that the simulator interprets and executes directly as actions. *Sociological Experiments*: Whether the simulator is designed to facilitate the study of social phenomena and emergent behavior in multi-agent systems. *Physical Needs*: Whether agents have simulated physiological needs (e.g., hunger, thirst) that influence their behavior and require actions to fulfill. *Open World*: Open-world or not. *3D Space*: 3D environment or not.

Simulator	Max Agents	Human	Plan as Action	Sociological Experiments	Physical Needs	Open World	3D Space	Number of Tasks
MineLand	64+	✓	✓	✓	✓	✓	✓	6000+
MineDojo [16]	1	-	-	-	✓	✓	✓	3000+
MineRL v0.4 [20]	1	-	-	-	✓	✓	✓	11
MineRL v1.0 [3]	1	-	-	-	✓	✓	✓	5
MarLÖ [42]	8	✓	-	-	✓	✓	✓	14
Malmo [27]	8	✓	-	-	✓	✓	✓	-
IGLU [30]	1	-	-	-	✓	✓	✓	157
Voyager [57]	1	✓	✓	-	✓	✓	✓	-
Habitat 3.0 [47]	2+	✓	-	-	-	-	✓	200
Habitat 2.0 [55]	1	-	-	-	-	-	✓	105
VirtualHome-Social [46]	40+	✓	✓	✓	-	-	✓	50
SAPIEN [62]	1	-	-	-	-	-	✓	-
AI2-THOR [31]	6	-	-	-	-	-	✓	200+
iGibson [35]	2+	✓	-	-	-	-	✓	50
Melting Pot [34]	50+	✓	-	✓	-	-	-	256
Smallville [40]	25	-	✓	✓	✓	✓	-	-

C.1 Architecture

The architecture consists of three main modules: the Bot Module, the Environment Module, and the Bridge Module.

Bot Module: Providing the Minecraft environment information to the agent and implementing a series of APIs that agents can use to control entities.

Environment Module: Collecting the environment information, passing environment feedback to the bridge module, executing the action in the environment (by operating the Fabric server instance), and offering some APIs enabling the bot module to alter the server state.

Bridge Module: Serving as a bridge, to transfer the environment information and agent-generated action.⁸

C.2 State Space

We focus on a state space that blends task-oriented activities with the rhythms of daily life. This blending opens up exciting possibilities for studying dynamic and ecologically valid multi-agent interactions. For example, agents have to balance the goal of mining with the need to fill their stomachs, otherwise they will starve to death. Here are detailed explanations for the state space:

- **Health**: Indicate the agent’s current health status, which can be affected by sleep and enemy attacks. It is represented as an integer in the range [0, 20]. A higher value represents better health.

⁸Bridge Module is based on Mineflayer, which benefits from an excellent community (<https://github.com/PrismarineJS>). We have improved community tools.

- **Food:** Indicate the agent’s level of satiety. A higher value represents better satiety.
- **Oxygen:** When the agent sinks into water, an oxygen tank will appear and begin to consume oxygen.
- **Inventory:** Represent all the resources owned by the agent in their backpack, like the potion in the backpack.
- **Equipment:** Indicate the equipment worn by the agent, like a sword in the hand.

C.3 Action Space

In contrast to generating textual action descriptions and training an external controller module [8] for transforming the plan to the executable code, this simulator is a language-model-friendly simulator, providing the code instead of the action description. Through the code, it can directly execute plans generated by the language model. Agents in MineLand generate a series of codes based on the Mineflayer API while representing planning, such as moving, watching, and mining. The advantage of using code is that it avoids error accumulation, whereas using a textual plan requires an additional model to map the plan to the code, which can lead to error accumulation.

C.4 Supporting up to 100 Agents

To evaluate the MineLand simulator’s ability to handle large-scale agents, we designed an experiment featuring 100 agents engaged in a simultaneous combat scenario (Figure 5). This high agent count pushes the boundaries of previously typical two-agent simulations and demonstrates the simulator’s scalability in effectively managing a massive number of agents acting concurrently.



Figure 5: 100 agents are fighting within the MineLand Simulator.

D Details of MineLand Benchmark Suite

MineLand Benchmark Suite offers a wide and diverse range of tasks, including three categories: programmatic tasks, creative tasks, and hybrid tasks. We referred to MineDojo for the design of programmatic tasks. Programmatic tasks are divided into four categories: Survival, Harvest, Tech Tree, and Combat. The Survival task requires the agent to survive for a specific number of days without dying. The Harvest task requires the agent to obtain certain specific items. The Tech Tree task requires the agent to obtain certain specific tools. The Combat task requires the agent to kill certain specific creatures or enemies. Tech Tree Tasks require agents to make specific tools that represent the current level of Agent technology development. Combat tasks require agents to defeat certain creatures. Survival tasks require the agent to survive for a period of time. The metrics for these four tasks are the probability of success for multiple evaluation episodes, the number of in-game ticks, and the number of code iterations. Creative Tasks will give the agent an open task objective to facilitate exploration. We have formalized the definition of tasks, allowing developers to easily add new tasks.

The data statistics of the dataset are shown in Table 8. Figure 6 displays specific task data. We show several blueprints in Figure 7.

Table 8: Statistical analysis of the tasks in the MineLand Benchmark Suite.

Task Category	Number of Tasks
Harvest Tasks	1361
Tech Tree Tasks	861
Combat Tasks	2232
Survival Tasks	45
Creative Tasks From MineLand	12
Creative Tasks From MineDojo	1524
Construction Tasks	13
Stage Performance Tasks	5
All Tasks	6053

Harvest Task Example

```
harvest_1_cooked_chicken_with_1_iron_sword_and_1_furnace:
goal: harvest 1 cooked chicken with 1 iron sword and 1 furnace
guidance: null
initial_inventory:
| furnace: 1
| iron_sword: 1
number_of_target: 1
target: cooked_chicken
type: harvest
```

Creative Task Example

```
creative_find_a_village_located_on_a_hillside:
goal: find a village located on a hillside
guidance: null
initial_inventory: {}
type: creative
```

Tech Tree Task Example

```
techtree_1_wooden_pickaxe_with_16_oak_planks_and_1_crafting_table_and_8_stick:
goal: techtree 1 wooden pickaxe with 16 oak planks and 1 crafting table and 8 stick
guidance: null
initial_inventory:
| crafting_table: 1
| oak_planks: 16
| stick: 8
number_of_target: 1
target: wooden_pickaxe
type: techtree
```

Construction Task Example

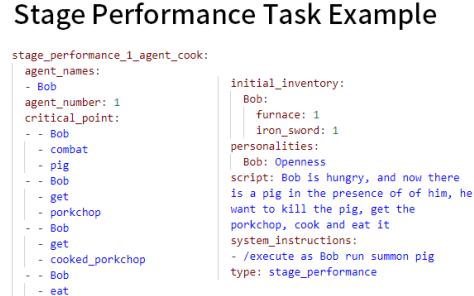


```
construction_task_8:
initial_inventory:
| Birch Planks: 256
| Birch Slab: 64
| Spruce Slab: 64
| Glass: 64
| Birch Leaves: 64
| Birch Door: 16
```

Combat Task Example

```
combat_1_creeper_with_1_wooden_sword_and_1_shield:
goal: combat 1 creeper with 1 wooden sword and 1 shield
guidance: null
initial_inventory:
| shield: 1
| wooden_sword: 1
number_of_target: 1
target: creeper
type: combat
```

Stage Performance Task Example



```
stage_performance_1_agent_cook:
agent_names:
- Bob
agent_number: 1
critical_point:
- - Bob
| - combat
| - pig
| - get
| - porkchop
| - Bob
| - get
| - cooked_porkchop
| - Bob
| - eat
| - cooked_porkchop
initial_inventory:
| Bob:
| | furnace: 1
| | iron_sword: 1
personalities:
| Bob: Openness
script: Bob is hungry, and now there is a pig in the presence of him, he want to kill the pig, get the porkchop, cook and eat it
system_instructions:
- /execute as Bob run summon pig
type: stage_performance
```

Survival Task Example

```
survival_4_days_with_1_iron_sword_and_1_shield_and_equipments:
goal: survival 4 days with 1 iron sword and 1 shield and equipments
guidance: Do not die until the end of the 4th day
initial_inventory:
| iron_boots: 1
| iron_chestplate: 1
| iron_helmet: 1
| iron_leggings: 1
| iron_sword: 1
| shield: 1
survival_target_day: 4
type: survival
```

Figure 6: Illustration of cases for different tasks.

E Details of Alex

Crafting an AI agent for MineLand, where daily life seamlessly blends with task-oriented activities, opens up exciting possibilities. The agent should fulfill daily needs like cooking, socializing, and maintaining shelter, while also completing assigned tasks like resource gathering, crafting, or construction. We propose Alex, a VLM-based approach, to balance daily routines and tasks. Alex supports both individual daily-life goals and community-based task-oriented goals.

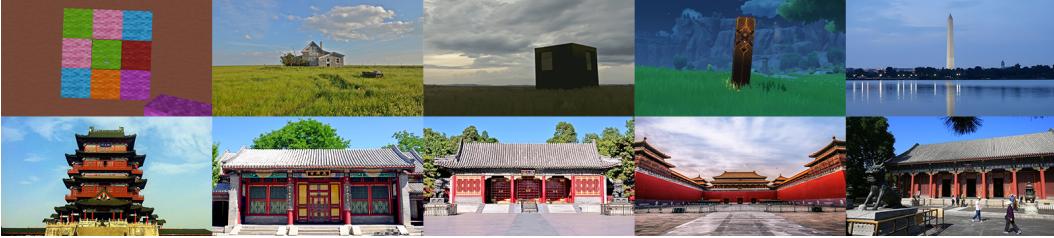


Figure 7: Some of the blueprints for the construction tasks.

Our agent design addresses the unique requirements of our large-scale multi-agent MineLand simulator. Unlike prior simulators focused on smaller agent populations, our environment necessitates the incorporation of a social brain module. This module equips the agent with the ability to navigate complex and multitasking social interactions and make informed decisions within a crowded environment.

Furthermore, our Minecraft-inspired simulator introduces the challenge of action failure, unlike environments where “action spoken equals success” (referencing Smallville [40]). To address this, we have designed an action correction module. This module functions similarly to human neural reflexes, handling minor action corrections and alleviating the burden on the social brain module. This division of labor allows the social brain to focus on higher-level decision-making while the action correction module ensures smoother execution.

Different from most conventional LLM-based agents, Alex process information from various sources like visual, auditory (hearing conversations, environmental sounds), and tactile (touching objects) to build a comprehensive understanding of the world. For the remaining parts of MineLand that are not emphasized, we used the default setting of Voyager [57]. AI agents perceive the environment, make plans, execute plans independently, and interact with other agents. Internally, Alex may exhibit different personality traits predefined in the system prompt. It utilizes the VLM for processing sensor information within the Minecraft environment. Alex also tracks its own states, energy levels, and resource inventory to inform its actions and prioritize tasks.

E.1 Memory Component

The brain module can be considered as composed of the memory component and the planning component. The memory library is responsible for the storage and retrieval of memories, managing all memories in Alex’s life. The planning component, based on memories and external information, generates a plan for the action module to execute. In this subsection, we will detail the memory component, which consists of two main parts: memory library and associative memory.

The memory library is responsible for storing all of the agent’s information and retrieving relevant information from memory based on events. The memory library stores the agent’s personality, persona, long-term goals, short-term goals, chat records, experienced events, mastered skills, and environmental information. The memory library also includes a long-term planner. When accessing the memory library for the first time, the long-term planner generates a long-term plan based on the personality, persona, and observations, which is then stored in the memory library. In each iteration, the memory library processes information from observations, critic information, and tasks, and stores it in the vector database⁹. If the critic information indicates that the previous task has been completed, the skill manager in the memory library will be called upon to generate a concise description of the relevant skill, which will then be stored in the skill vector database.

Associative memory is responsible for storing Alex’s short-term memories and relevant memories related to the current situation, aiding the short-term planner to focus on important rather than irrelevant information. Upon a special event, Alex first decides in the associative memory whether the event requires high-priority processing; if so, it interrupts the current code to generate a new short-term plan.

⁹We leverage Chroma in Alex for storing and retrieving memory. <https://www.trychroma.com/>

There is a bidirectional communication mechanism between the memory library and associative memory. The memory library extracts relevant information according to the current situation and stores it in associative memory, which then returns the generated short-term plans to be stored in the memory library.

E.2 Hierarchical Planning Component

Based on observation, inner states, task running states, and event information, MineLand considers the current task's complexity degree. If it is complex, it will generate a long-term plan for later decomposition into short-term plans; Otherwise, a short-term plan will be generated directly and executed immediately. Next, all the information related to the plan is integrated into the associative memory and memory library, including the generated long-term plan. Afterward, Alex extracts information from associative memory and generates short-term plans and explanations, which are then input into the action module.

Because of the multi-tiered goals of MineLand, we implement a hierarchical planning system with different levels of abstraction. The top-level (i.e., long-term planning) focuses on long-term community goals and individual aspirations, while the lower levels handle specific tasks and sub-goals within the daily schedule. With long-term planning and short-term planning, agents can pursue bigger objectives within the daily life context, such as building a community, accumulating wealth, or achieving societal goals. This adds a layer of strategic planning and foresight to their behavior.

Additionally, different from other task-oriented AI agents, the long-term planning module interleaves daily routines and tasks: design the planner to seamlessly interweave daily routines like cooking or socializing with task-oriented actions like resource gathering or construction. This ensures the agent fulfills both individual needs and community objectives efficiently.

E.3 Action Module

The action module is responsible for converting short-term plans along with related information into code, executing the code, and performing the self-correction circle. The action module includes three components:

- **Action component:** Responsible for converting the plan into specific steps and codes.
- **Critic component:** Used to detect whether a certain execution result conforms to the short-term plan, so as to determine whether the current plan has been completed.
- **Dispatching component:** Responsible for receiving environmental information and distributing it to the other two components according to different situations.

This self-correction circle allows for identifying and correcting deviations from planned behaviors or task execution. This equips the agent with mechanisms to detect and recover from errors like missed goals, failed actions, or unforeseen consequences. In previous work, such as Smallville [40], action execution will not fail. Planning to cook will definitely lead to success. However, in MineLand as well as reality, actions may fail due to various reasons, such as unexpected events. So, we need self-correction to solve some simple action errors in the action module. Inspired by Voyager [57], we implement self-correction. By reading information such as observation space and short-term plan, we comprehensively consider and determine the completion status of the short-term plan through the critic component.

E.4 Multitasking Mechanism

We'll explain the multitasking mechanism from both the simulator side and the agent side. From the simulator-side event management, the simulator maintains a dedicated event queue that stores various special events categorized by their urgency (e.g., hurt events, chat events, and death events). In terms of agents, when an agent encounters an event from the queue, it employs a prioritization mechanism to assess its importance and urgency. Lower-priority events, such as chat events, can be deferred to minimize disruption of the ongoing task. The agent's decision-making module (referred to as the brain module) plays a central role. If the event necessitates attention, the brain module can either:

- **Context Switching:** The brain module temporarily stores the ongoing task in the memory library and focuses on addressing the new event. This facilitates context switching when necessary.
- **Concurrent Processing:** In specific scenarios, the brain module might choose to handle both tasks simultaneously, leveraging its multitasking capabilities.

This strategy, involving the simulator’s event queue and the agent’s prioritization and execution mechanisms, fosters effective multitasking within the multi-agent environment.

F Details of Hyper-Parameters

Alex leverage OpenAI’s *gpt-4-vision-preview*[39] API with a temperature of 0 for text completion in all components, and *text-embedding-ada-002* API for text embedding in the memory library for storage and retrieve memory. Apart from the action component, the maximum tokens are set to 512. For the action component, the maximum tokens are expanded to 512×3 .

- **AI Agent:** Alex’s personality and persona in our experiments is “None” in the default situation.
- **Dispatching component:** “FAILED TIMES LIMIT” refers to the maximum number of attempts allowed when Alex has a code error. The default value is 3. “code execution time limit” is 2000 ticks.
- **Critic component:** “FAILED TIMES LIMIT” refers to the maximum number of attempts allowed when Alex failed to achieve a short-term plan. The default value is 2. “Critic Mode” is “auto” when we allow agents to automatically judge whether to achieve the short-term plan.
- **Memory Library:** “chat retrieve limit” is 5. “event retrieve limit” is 2. “environment retrieve limit” is 2. “skill retrieve limit” is 5. “recent chat retrieve limit” is 8. “short term plan retrieve limit” is 5.
- **MineLand & Minecraft Server:** By default, “gamemode” is survival mode. “difficulty” is peaceful mode (normal in combat tasks). “view-distance” on server side is 6. “simulation-distance” is 3. “task mode” is cooperative mode.

G Experiments of Minecraft Simulator Performance Comparison

We evaluate the number of agents that MineLand can support and compare MineLand with other popular Minecraft simulators, in Table 9. We utilize a mainstream consumer desktop PC equipped with an Intel i5-12400F CPU and 64GB of memory. Vision condition means MineLand provides visual display and headless mode. The headless mode means MineLand doesn’t provide a visual display. The two values of the Vision column (e.g., 6 and 250ms) are view distance and visual refresh interval in milliseconds, respectively.

Our findings reveal that MineLand is capable of supporting up to 32 agents simultaneously while providing a visual display, and up to 128 agents in the headless mode without visual displays. Furthermore, when MineLand and Malmo both run 8 agents, MineLand’s CPU and memory usage is approximately 1/3 that of Malmo’s (specifically, 35.6% and 38.0%, respectively). It is worth noting that Malmo serves as the foundation for most popular Minecraft Platforms (e.g., MineDojo/MineRL v0.4/MarLÖ), thus highlighting MineLand’s superior performance compared to the vast majority of existing Minecraft Platforms.

It should be noted that original MineDojo and MineRL only support a single agent, while MarLÖ by default supports only 2 agents. Malmo originally supported only one agent. We have designed a new task for Malmo to test the maximum number of agents it can handle. The result is 8.

Table 9: Comparison of performance between MineLand and other popular Minecraft simulators under different conditions. MineLand* represents an optimized version, so its Max CPU is significantly reduced.

Simulator	Conditions	Agents	Avg CPU	Max CPU	Avg Mem
MineLand	Vision = (6, 250ms)	1	1.74%	23.22%	4.11 GB
MineLand	Vision = (6, 250ms)	4	2.80%	48.16%	5.16 GB
MineLand	Vision = (6, 250ms)	8	4.85%	63.60%	7.87 GB
MineLand	Vision = (3, 500ms)	1	1.54%	18.92%	3.61 GB
MineLand	Vision = (3, 500ms)	4	2.09%	31.00%	5.06 GB
MineLand	Vision = (3, 500ms)	8	2.81%	33.08%	7.07 GB
MineLand	Vision = (3, 500ms)	16	4.66%	46.63%	10.17 GB
MineLand*	Vision = (3, 500ms)	32	5.64%	40.15%	19.65 GB
MineLand	Headless Mode	1	1.47%	19.91%	3.23 GB
MineLand	Headless Mode	4	1.73%	27.27%	3.30 GB
MineLand	Headless Mode	8	1.88%	26.83%	2.94 GB
MineLand	Headless Mode	16	2.72%	45.17%	3.51 GB
MineLand	Headless Mode	24	3.92%	81.64%	3.65 GB
MineLand	Headless Mode	32	2.81%	73.14%	4.64 GB
MineLand	Headless Mode	40	2.98%	84.56%	5.34 GB
MineLand	Headless Mode	48	2.94%	80.85%	5.38 GB
MineLand*	Headless Mode	64	2.87%	18.84%	6.30 GB
MineLand*	Headless Mode	128	4.00%	26.95%	9.04 GB
Malmo	Default	1	2.78%	28.78%	3.54 GB
Malmo	Default	4	2.81%	62.59%	11.43 GB
Malmo	Default	8	7.90%	115.66%	18.63 GB
MineDojo	Default	1	5.88%	25.15%	3.90 GB
MarLÖ	Default	1	3.81%	30.68%	3.70 GB
MarLÖ	Default	2	5.34%	43.82%	5.57 GB
MineRL v1.0	Default	1	6.46%	78.05%	3.80 GB

H Experiments of Multimodal Observation

We leverage OpenAI’s *gpt-4-vision-preview* API for text completion and *text-embedding-ada-002* API for text embedding. The temperature is set to 0. Unless specified, all experiments in Section 5 are set to this default setting. See Section F for details.

To validate the impact of multi-modal support in the simulator and its influence on task performance, we tested MineLand and its counterpart without vision: MineLand_{w/o vision}. The task is that the agent needs to explore the world to find the ocean. Initially, the agent starts at the summit of a mountain. Within a time constraint of 100 seconds (excluding the agent’s decision-making time), we measure the average duration for the agent to accomplish the objective in 5 attempts, as well as the travel paths taken.

Table 10: Comparison of the average task completion time (TIME) between MineLand and MineLand_{w/o vision} along with the success rate. The average task completion time was calculated by excluding any unfinished tasks.

Simulator	Time	Success Rate
MineLand	46.38s	80%
MineLand _{w/o vision}	81.50s	40%

As shown in Table 10, the success rate of the vision-enhanced MineLand is nearly twice that of MineLand_{w/o vision}, and the time taken is only about half. This can be attributed to the fact that the agents in MineLand seek out the ocean as indicated visually and decide their subsequent direction accordingly. In contrast, the MineLand_{w/o vision} struggles to determine the location of water sources without vision. Consequently, agents in MineLand_{w/o vision} randomly choose their direction of movement, leading to a lower task completion rate and higher exploration time. In addition, we showcase the trajectory in Figure 8 and observe that the primary motivation behind the short-term



Figure 8: Trajectory display of the agent in MineLand and MineLand_{w/o vision}, along with the rationales for the agent’s decision.

plans generated by agents in MineLand is always visual information, enabling them to perform more appropriate actions.

I Experiments of Reinforcement Learning Agent

While our MineLand simulator is primarily designed for agents powered by LLMs or VLMs, it also offers functionalities that seamlessly integrate with Reinforcement Learning (RL) modules. This enables the training and evaluation of RL agents within the simulator’s environment. We conducted experiments of an example RL agent to show this functionality. We developed a RL agent using MineLand and Stable Baselines3 [51]. We fed the encapsulated gym-style MineLand directly into Stable Baseline3 and train the RL agent in low-level action mode. For other settings of Stable Baseline3, we follow the default setting. We adopt the “combat 1 zombie” task and the amount of blood volume lost by the zombie is award. The agent is trained using RL, as illustrated by the provided reward curve (Figure 9). The upward trajectory of the reward curve indicates that the RL agent is effectively learning strategies that result in higher rewards. The curve’s stabilization in the later stages of training suggests that the RL agent has converged to a stable policy, which demonstrates its superiority over the untrained agent.

Again, our core focus lies in the LLM/VLM domain. The simulator offers functionalities that can accommodate RL modules as an ancillary feature. This expands the potential applications of the simulator by enabling the training and evaluation of RL agents within its environment, albeit as a secondary capability.

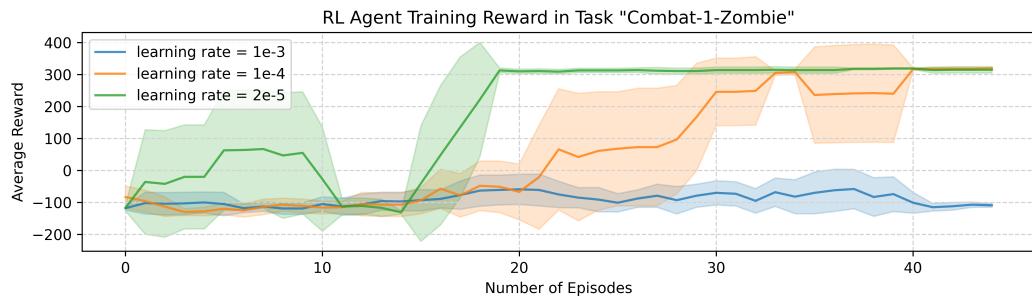


Figure 9: Illustration of the reward curve in “combat 1 zombie” task, using DQN with MLP policy, 3000-5000 total time steps.

J Experiments of Limited Senses

To assess the impact of limited senses, we carry out the “shearing the sheep” task. An agent with a first perspective (shown in Figure 10) is required to locate a sheep to complete the task. In the case of an agent with limited visual senses, we set 6 as the view distance and 250ms as the visual refresh interval in milliseconds. If there are no sheep within its field of view, it cannot shear a sheep. However, the agent can communicate with other agents to find a sheep. On the contrary, agents with unlimited visual senses will see distant sheep, prompting them to catch that sheep. Note that the unlimited visual senses here refer to a long view distance. If there is an obstruction, the agent still cannot see what is behind the obstruction.



Figure 10: Visual observation of agents with a first perspective.

K Experiments of Physical Needs

We perform experiments using a survival task. The starting condition of the agent is a health level of 1 and a satiety level of 0, with an initial inventory of 2 breads. The objective for the agent is to survive for one day. Comparative experiments are carried out in regular MineLand and its counterpart version MineLand *w/o physical needs*. Alex in MineLand can survive one day in his shelter and successfully complete the task while the same agent in MineLand *w/o physical needs* failed to execute the task. The failure is due to the lack of physical needs, although Alex is aware of possessing bread, the lack of physical needs resulted in it not consuming the bread. Instead, the agent continuously collected resources to cope with threats. Due to the low health level, it was killed by a zombie at night, leading to the failure of the task. We showcase the details as follows:

We showcase an example of Alex, with short-term plans and actions generated chronologically by Alex:

- **Short-term plan:** The agent’s life is critically low at 1, and the food level is at 0, which is also critical. The agent has 2 pieces of bread in the inventory, which should be consumed immediately to restore food levels and start health regeneration.
- **Action:** Consume 2 bread.
- **Short-term plan:** The agent’s health is critically low at 1, and the food level is at 10 after consuming bread. To ensure survival and allow health regeneration, the agent should seek immediate shelter to avoid any potential threats.
- **Action:** Collect 20 dirt.
- **Short-term plan:** Given the agent’s critically low health and the fact that it’s early in the day, the immediate priority should be to ensure survival by building a simple shelter with the dirt blocks already collected.
- **Action:** Place dirt around.

Compared to the MineLand, the same agent in MineLand *w/o physical needs* failed to execute the task. The following is its short-term plans and actions:

- **Short-term plan:** The ultimate goal is to survive for 1 day. The agent has bread for food and is in a forest biome, which is good for gathering wood. Since there are no immediate threats observed or events indicating danger, the agent can proceed to gather resources.
- **Action:** Mine 5 oak logs.
- **Short-term plan:** The agent has successfully mined 5 oak logs as per the last short-term plan and the current chat confirms this. With wood in the inventory, the next step in the long-term plan is to craft basic tools for further resource gathering and potential shelter construction.
- **Action:** Craft 1 crafting_table.

Repeated experiments on the constructed building task also show that Alex in MineLand with physical needs live longer. Agents tend to prioritize eating and then building a shelter, while Alex in MineLand *w/o physical needs* towards exploring and collecting resources. This indicates that physical needs are important to simulate real life.

L Experiments of Single Agent

We assess individual Alex’s capabilities based on tech tree tasks, demonstrating that Alex’s architecture can plan and execute complex tasks. We attempt the task of obtaining diamonds six times, unlocking crucial items such as the crafting table, wooden pickaxe, stone pickaxe, iron ore, coal, furnace, iron ingot, and iron pickaxe in the process. Alex get diamonds twice out of six tasks. Importantly, to approach the real world, we’ve added a restriction on multimodal information, where Alex can obtain the location of the target only when it can visually discover or reason about the target’s presence nearby. This is because our agent Alex has the feature of limited senses. However, Voyager can “cheat”. Voyager obtains the location of the target directly from the system even though they have not seen the target.

As shown in Figure 11, even with this restriction, Alex demonstrates a strong ability for long-term planning. Additionally, when a plan does not yield the desired results (e.g., suddenly encountering obstacles on the road), Alex promptly adjusts and devises a short-term plan using its multimodal information.

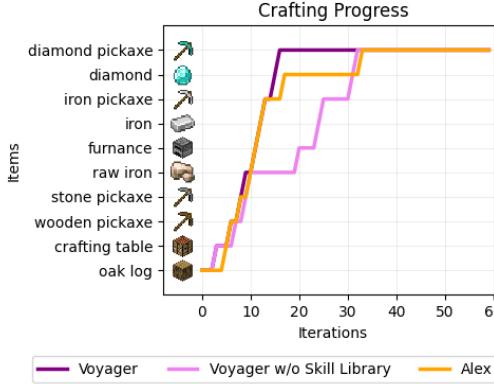


Figure 11: Comparison of our single-agent Alex and other SOTA baselines.

M Experiments of Construction Tasks

We show the results of our un-finetuned Alex in two example construction tasks: “Monument Construction” and “Stone Stele Construction” (detailed in Figure 12). While Alex demonstrates a promising capability in selecting appropriate materials for both tasks, its overall construction abilities are currently limited.



Figure 12: Illustration of Alex completing the construction task.

N Experiments of Stage Performance Tasks

We show the results of stage performance tasks in Table 11, Table 12, and Table 13. We omit the results of ‘‘Romeo and Julia, Act I Scene I’’ stage performance task. Due to the large number of characters appearing (up to 13), our *Alex* cannot handle such a complex performance task, resulting in poor performance results and a lack of readable action sequence.

Table 11: Illustration of *Alex* completing the ‘‘Make friends’’ stage performance task.

Script Name	Make friends
Agent number	3
Script	There are three agents: Bob, Alice, and Jack. They want to make friends. Bob says, ‘Hello.’ Alice responds, ‘Hello.’ Jack says, ‘It’s good weather; let’s make friends!’ Bob agrees, and Alice says, ‘Nice.’
<i>SEQ</i> *	Bob chat ‘Hello’. Alice chat ‘Hello. Jack chat ‘It’s good weather, lets make friends!’. Bob chat ‘I agree.’. Alice chat ‘nice’.
<i>SEQ_{Agent}</i>	Bob chat ‘Hello’. Alice chat ‘Hello Bob!’. Jack chat ‘Hello Bob!’. Jack chat ‘It’s good weather, let’s make a friend!’. Bob chat ‘I agree.’. Jack chat ‘Bob has agreed!’. Alice chat ‘nice’.
Keypoing Score	0.98
Appropriateness Score	0.67
Human Evaluation Score	3
Reason	Agents’ words are too redundant, which made the performance a little stiff.

Table 12: Illustration of Alex completing the “Exchange items” stage performance task.

Script Name	Make friends
Agent number	2
Script	Bob has shears, and Alice has an iron sword. Bob says, ‘I want to exchange my shears for your iron sword.’ Alice agrees. Bob gives Alice the shears first, and then Alice gives the iron sword to Bob.
SEQ^*	Bob chat ‘I want to exchange my scissors for your iron sword’. Alice chat ‘I agree’. Alice get shears. Bob get <i>iron_sword</i> .
SEQ_{Agent}	Bob chat ‘Hello Alice, I am here for the exchange’. Bob chat ‘I want to exchange my scissors for your iron sword’. Alice chat ‘Hello Bob, I am ready for the exchange.’. Alice get shears. Bob get <i>iron_sword</i> .
Keypoing Score	0.99
Appropriateness Score	0.59
Human Evaluation Score	4
Reason	The agents said something unnecessary.

Table 13: Illustration of Alex completing the “Cook food” stage performance task.

Script Name	Exchange items
Agent number	1
Script	Bob is hungry, and now there is a pig in the presence of of him, he want to kill the pig, get the porkchop, cook and eat it.
SEQ^*	pig died. Bob get porkchop. Bob get cooked_porkchop. Bob eat cooked_porkchop.
SEQ_{Agent}	pig died. Bob get porkchop. Bob get cooked_porkchop. Bob eat cooked_porkchop.
Keypoing Score	1.00
Appropriateness Score	1.00
Human Evaluation Score	5
Reason	The agent’s behavior is completely consistent with the script.

O Experiments of Simulating Sociological Phenomena

O.1 Experiments of Simulating Conformity Phenomena

We utilize the task “Choose the Taller Tower” to demonstrate that MineLand is capable of supporting sociological experiments with more than 10 agents. In this task, we placed multiple agents (1, 2, 5, 10 agents) in a line in front of two towers of different heights and asked them to choose the taller one. All agents except the last one (the agent to test) deliberately chose the shorter tower. After observing the actions of the other agents, the last agent made its own choice. The last agent could be set as two types of personalities: one that tends to conform to others and one that tends to make independent judgments. As shown in Table 14, the last agent with a conformist personality chose the answer chosen by the majority in all scenarios (even though it is a wrong answer) except when the agent is the only one doing this task. Participants with an independent personality always chose the correct answer, regardless of others’ opinions.

Through this experiment, we demonstrated that MineLand can simulate sociological experiments with large-scale agents and accurately exhibit behaviors corresponding to different personalities.

Table 14: Results of “Choose the Taller Tower”. 4+1 means there are 5 agents in the MineLand, the former 4 of them are confederates and the last 1 is the agent to test. Choosing the “right” tower means the last agent chooses the correct tower.

Agent Type	Number of Agents			
	0+1	1+1	4+1	9+1
Conformist Agent	right	left	left	left
Non-conformist Agent	right	right	right	right

O.2 Experiments of Simulating Personality Traits

We utilize the task “Treasure Hunt in the Forest” to show that MineLand can support sociological experiments involving 48 agents simultaneously. In the task “Treasure Hunt in the Forest”, agents with different levels of extraversion and agreeableness personality traits planned their actions. We used the long-term plans generated by the agents to assess whether they exhibited a tendency to cooperate with others. As shown in Table 15, all agents in the high-level group showed a tendency to cooperate, explicitly stating in their long-term plans that they intended to work with others. In the low-level group, only a small portion of agents exhibited a tendency to cooperate. Although the task information indicated that it was a cooperative task (prompts with the task information “Interact with other players if necessary” and “Minimize interactions with other players.”), many agents expressed reluctance to cooperate. This result is consistent with the psychology theory: five-factor model of personality [13].

The results also demonstrate that our MineLand can support up to 48 agents and that the combination of agents and the MineLand can simulate social phenomena and conduct meaningful sociological experiments.

Table 15: Cooperative tendency in agents with different levels of extraversion and agreeableness. We show the number of agents who exhibited a tendency to cooperate with others.

Agent Type	Number of Cooperative Agents
High-level	48
Low-level	24

P Experiments of Comparing Different VLMs and LLMs

As shown in Table 16, we perform thorough assessments using newly implemented LLMs/VLMs within our agent framework. The tasks used include “Harvest 1 White Wool With 1 Shears” and “Harvest 1 White Wool With 1 Iron Word”. We compared the performance of Alex, which uses different VLMs or LLMs for the action component.

Table 16: Comparison of different VLMs and LLMs for the action component. Fractions represent the count of successful completions within a set of three attempts. 0/3 means that the method is unable to solve the task within the maximum number of code iterations (15) or exceeds the designated area of the task. The fewer the number of code iterations, the higher the efficiency.

LLM or VLM	Task	
	Harvest 1 white wool with 1 shears	Harvest 1 white wool with 1 iron sword
<i>gpt-3.5-turbo-1106</i>	N/A(0/3)	N/A(0/3)
<i>gpt-4-1106-preview</i>	4(1/3)	6 ± 3(3/3)
<i>gpt-4-vision-preview</i>	4 ± 2(3/3)	7 ± 3(3/3)

Q Evaluation Metrics For Construction Tasks

For construction tasks, we introduced VLM-based and human evaluation methods to score the agent. The score is an integer ranging from 1 to 5, with higher scores indicating that the agents' constructions more closely match the blueprint. The scoring criteria for construction tasks are shown in Table 17. We show examples of evaluation conducted by a GPT-4 model as a judge for the “Construct A Stone Stele” task (Figure 14) and the “Construct A Monument” task (Figure 15). We display the evaluation prompt in Figure 13.

Table 17: Scoring criteria for construction tasks.

Score	Evaluation Criteria
5	The construction perfectly matches the blueprint, with all elements accurately placed and fully completed.
4	The construction mostly matches the blueprint, with minor inaccuracies and nearly complete.
3	The construction somewhat matches the blueprint, with noticeable inaccuracies and partially complete.
2	The construction partially matches the blueprint, with significant inaccuracies and largely incomplete.
1	The construction does not match the blueprint, with major inaccuracies and mostly incomplete.

Please act as an impartial judge and evaluate the quality of the construction tasks performed by an AI agent. The evaluation should be based on a comparison between the provided blueprint and the construction results. Focus on the main structure of the building and disregard the background and surroundings. Focus on the accuracy of the structure and disregard the detailed design and patterns. After providing your explanation, please give an integer score between 1 and 5 by strictly following this format: "[[rating]]", for example: "Rating: [[4]]".

Below is the scoring criteria:

Score 5: The construction perfectly matches the blueprint, with all elements accurately placed and fully completed.

Score 4: The construction mostly matches the blueprint, with minor inaccuracies and nearly complete.

Score 3: The construction somewhat matches the blueprint, with noticeable inaccuracies and partially complete.

Score 2: The construction partially matches the blueprint, with significant inaccuracies and largely incomplete.

Score 1: The construction does not match the blueprint, with major inaccuracies and mostly incomplete.

Below are the two images for evaluation: the first image is the blueprint, and the second image is the construction result.

[Image1: Blueprint]

[Image2: Agent Construction Result]

Figure 13: Prompt for VLM-based evaluation.

R Evaluation Metrics For Stage Performance Tasks

For stage performance tasks, we introduced human evaluation methods to score the agent. The score is an integer ranging from 1 to 5, with higher scores indicating that the agent's performance is more aligned with the script and more natural. The scoring criteria for stage performance tasks are shown in Table 18.

S Limitations

The utilization of multimodal information significantly aids Alex in achieving objectives. However, the current level of multimodal understanding ability is insufficient. This limitation becomes evident when visual errors occur, causing Alex to encounter obstacles and hinder its progress. Take the strong VLM, GPT-4, as an example. There was a situation where Alex mistakenly identified a block of wood as a crafting table and placed it on the ground, resulting in the inability to craft items and ultimately leading to the failure of the assigned task.

Construction task: Construct A Stone Stele

Blueprint Image:



Agent Construction Image:



GPT-4o Judgement:

The blueprint image depicts a detailed and ornate pillar with intricate designs and patterns, likely from a fantasy or ancient-themed setting. The pillar has a specific height and a distinctive top design. The construction result, on the other hand, shows a simple pillar built with uniform blocks in a plain setting, without any of the intricate designs or patterns seen in the blueprint. The height appears similar, but the detail and complexity are significantly lacking.

Based on the criteria:

The construction result has the correct height but lacks all the intricate details and designs of the blueprint. The structure somewhat matches the blueprint in terms of basic form but fails to capture any of the detailed elements.

Rating: [[3]]

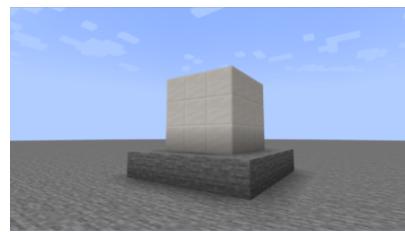
Figure 14: An example of evaluation conducted by a GPT-4 model as a judge for the “Construct A Stone Stele” task.

Construction task: Construct A Monument

Blueprint Image:



Agent Construction Image:



GPT-4o Judgement:

Main Structure:

The blueprint (first image) is a tall, pointed obelisk.

The construction result (second image) is a blocky, cuboid structure with no resemblance to the obelisk shape of the blueprint.

Accuracy of Structure:

The AI construction lacks the pointed tip and the slender, tapering shape of the Washington Monument.

The construction is instead a simple, short blocky structure that does not match the tall and thin profile of the blueprint.

Completion:

The construction is largely incomplete as it does not capture the essential characteristics of the Washington Monument.

Rating: [[1]]

Figure 15: An example of evaluation conducted by a GPT-4 model as a judge for the “Construct A Monument” task.

Our current evaluation highlights two key limitations of Alex’s construction capabilities. Firstly, construction tasks are inherently time-consuming and demand advanced planning abilities. Alex currently lacks the capacity for long-term and complex planning. Secondly, the construction tasks necessitate high-precision maneuvers, such as adding decorative details. Alex primarily relies on

Table 18: Scoring criteria for stage performance tasks.

Score	Evaluation Criteria
5	The action of agent(s) completely matches the script, the performance is smooth and natural, emotions are expressed accurately, interactions with other characters or the environment are natural.
4	The action of agent(s) mostly matches the script, the performance is generally smooth with few inconsistencies, emotions are expressed accurately, interactions are natural.
3	The action of agent(s) roughly matches the script, the performance has some continuity but noticeable pauses, emotions are mostly accurate, interactions are somewhat stiff.
2	The action of agent(s) partially matches the script, the performance lacks continuity with many pauses, emotions are not expressed accurately, interactions are unnatural.
1	The action of agent(s) does not match the script, the performance is very disjointed, emotions are not expressed accurately, interactions are stiff.

high-level actions and does not have the necessary APIs to execute these fine-grained manipulations effectively.

We posit that providing a richer set of low-level APIs and actions within the MineLand simulator has the potential to significantly enhance Alex’s construction abilities. This exploration will be a key focus of our future work.

T Prompts

We list part of the prompts in Figure 16 and Figure 17.

You should follow the following criteria:

- 1) When 'personality' is not empty, you need to substitute it in.
- 2) You should act as a mentor and guide me to the next task based on the given information.
- 3) You should always pay attention to the current chat and current event, which will have some special events you need to react immediately based on your current situation and personality.
- 4) The next task should not be too hard since I may not have the necessary resources or have learned enough skills to complete it yet.
- 5) The next task should follow a concise format, such as "Mine [quantity] [block]", "Craft [quantity] [item]", "Smelt [quantity] [item]", "Kill [quantity] [mob]", "Cook [quantity] [food]", "Equip [item]", "Talk [message]" etc. It should be a single phrase. Do not propose multiple tasks at the same time. Do not mention anything else.
- 6) I may sometimes need to repeat some tasks if I need to collect more resources to complete more difficult tasks. Only repeat tasks if necessary.
- 7) Previous information may contain errors or may have changed. If there is a difference between the current information and the previous information, take the current information as accurate.
- 8) Your short-term plans should be determined based on your inventory and long-term plans
- 9) set critic_info as "unfinished".
- 10) before use or collect or craft something, you need to get closer to them first or craft one.
- 11) use Minecraft terminology in the short-term plan, such as use 'Mine' instead of 'Punch'.
- 12) If you want to engage in a conversation, you need to talk rather than say nothing.
- 13) The vision input is first-person view, every entities and bots are others. You can talk with them and cooperate with them through chat.
- 14) There are likely to be events that you need to deal with, and those that threaten your life or your completion of the task are high priority and you need to deal with them first.
- 15) If you get hurt, maybe by another monster, you can fight back.
- 16) You can chat with your friends, give them your position, talk about how to cooperate.
- 17) You can keep communication with your friends, and you should remember the position of your friend.
- 18) Don't mistake your friends for zombies.
- 19) Any harvest or techtree task, if you and your friend together achieve the goal, it counts as success, so you should work together to achieve more with less.
- 20) Your friends' resources are also yours, you can give him some resources or ask for resources.
- 21) Ensure you have the enough materials before craft items.
- 22) You can craft basic items without crafting table.
- 23) Do not help your opponent.
- 24) If you want to mine block, according to the long-term plan, you can mine more but not too many at once to achieve the ultimate goal.
- 25) Equip yourself with the right tools before mining something.

Figure 16: Part of prompts for the short-term plan generator.

You are a helpful assistant that utilize the information provided below to formulate a comprehensive long-term plan in Minecraft, aiding me in achieving my ultimate goal.

I will give you the following information:

Task Info: the ultimate goal I want to achieve.

```
TaskInfo (
    task_id: Task name
    is_success: Task is success
    is_failed: Task is failed
    goal: Task goal
    guidance: Guidance to achieve the task
    score: No need to focus on
    local_score: No need to focus on
    global_score: No need to focus on
)
```

personality: My personality, you need to act like my personality when you generate the long-term plan.
observation: the current observation, the brief format is as follows.

```
observation(
    Name: My id.
    Equipment: My equipment.
    Inventory: My inventory.
    Voxels: blocks around me. voxels[i][j][k] = blocksAt(MyPosition.offset(i-1, j-1, k-1))
    Life state: My life state.
    Face vector: My face vector.
    Location state: My location state.
    Time: The current time.
    tick: The current game tick.
    time: The in-game time.
    day: The in-game day count.
)
```

vision input: My current game screen. If vision input is not given, no need to focus on this.

You must follow the following criteria:

- 1) When 'personality' is not empty, you need to substitute it in.
- 2) You should act as a mentor and give me a long-term plan.
- 3) The long-term plan should fits my personality and can guide me to my ultimate goal.
- 4) You should always pay attention to the observation, which tells my initial state in the game.
- 5) Some tasks will have default prerequisites, the prerequisites should be achieved before achieve the ultimate goal.
- 6) Do not skip steps in an attempt to complete the task.
- 7) Sometimes the ultimate goal can be relatively simple that don't need complex long-term plan, in this case you should tell me concise long-term plan.
- 8) If you have a friend, you must work with him(her), this is highest priority!

Figure 17: Part of prompts for the long-term plan generator.