

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}

Abstract

Conventional multi-agent simulators often assume perfect information and limitless capabilities, hindering the ecological validity of social interactions. We propose a multi-agent Minecraft simulator, MineLand, that bridges this gap by introducing limited multimodal senses and physical needs. Our simulator supports up to 48 agents with limited visual, auditory, and environmental awareness, forcing them to actively communicate and collaborate to fulfill physical needs like food and resources. This fosters dynamic and valid multi-agent interactions. 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 (Park et al., 2023) to virtual environments (Bates, 1994; Laird & VanLent, 2001; Gong et al., 2023), 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 (Zhang et al., 2023), Smallville (Park et al., 2023) and Overcook games (Gong et al.,

2023). However, conventional multi-agent simulators are often under the assumption of perfect information and limitless capabilities. These idealized worlds diverge sharply from the messy reality of human interaction. This gap between simulated agents and real-world humans hinders the ecological validity and richness of social interaction within these platforms (Heil, 1983).

We propose MineLand, a multi-agent Minecraft simulator as shown in Figure 1, that bridges this gap by introducing limited multimodal senses and physical needs as primary drivers of agent behavior and interaction. 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 (Fan et al., 2022) and twenty-five-agent Smallville (Park et al., 2023), our MineLand enables the utilization of forty-eight agents in the mainstream consumer desktop PC (§5.1). Secondly, our simulator operates under the fundamental assumption (Heil, 1983) 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 context. 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 (Doyal & Gough, 1984; Alderfer, 1969). By incorporating these three features, our simulator fosters the emergence of dynamic and ecologically valid multi-agent interactions¹.

MineLand supports a wide range of task structures and difficulties, covering harvest, tech tree, combat, survival, creative, construction, and stage performance tasks. Cus-

^{*}Equal contribution ¹Beijing Jiaotong University, Beijing, China. Correspondence to: Wenjuan Han <wjhan@bjtu.edu.cn>.

¹Ecological 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.



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 multi-tasking 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.

tomizing the number of players is supported. We also offer two task modes: cooperative mode and competitive mode, in addition to the default free mode. This makes it an excellent platform for benchmarking emergent multi-agent capabilities. With its flexibility, we design different scenes, scenarios, and settings, including crowd-sourcing 4499 programmatic tasks and 1536 creative tasks (2 times compared to MineDojo), to fully understand the potential of LLM- or VLM-based AI agents. Additionally, we have introduced 18 hybrid tasks that combine the features of programmatic tasks and creative tasks.

To integrate agents into this simulator, we design an AI agent framework - *Alex* - inspired by Multitasking theory from Cognition field (Salvucci & Taatgen, 2008). *Alex* allows for simultaneously simulating and executing intricate coordination and scheduling with multiple tasks. With this interactive agent framework, we have obtained the following intriguing findings: (1) Multimodal information enabling agents to perform more appropriate actions (§5.2); (2) Multitasking mechanism allows for processing multiple tasks by autonomously determining their priority (§5.3); (3) Limited senses force agents to actively communicate to compensate for sensory deficiencies (§5.4); (4) Physical needs mirrors real life; agents with physical needs survive longer (§5.5); (5) Our agents work together more effectively, with a reduced workload per agent (§5.6).

With contributions in the simulator (§2), benchmark (§3) and AI agent (§4), we push the boundaries of multi-agent simulation by bridging the gap between virtual agents and 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

MineLand, inspired by Malmo (Johnson et al., 2016) and MineDojo (Fan et al., 2022), is a Minecraft simulator where players² can explore, and interact with each other as well as the environments. This section dissects the design and implementation of our simulator, focusing on the architecture, observation space, state space, action space and communication.

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.

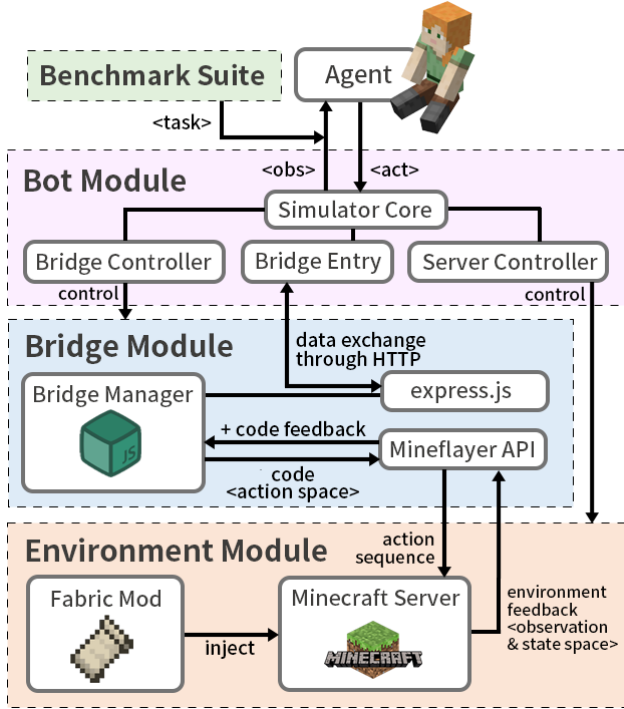


Figure 2. Illustration of the architecture of MineLand.

2.1. Architecture

MineLand enables a large number of players, with a better architectural design and a mechanism to facilitate unlimited large-scale interactions among them. To accomplish this, we suggest an architecture to expand Mineflayer (PrismarineJS, 2023) as shown in Figure 2. Based on this new architecture, MineLand supports up to 48 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. The architecture consists of three main modules: the bot module (implemented in Python), the environment module (implemented in Java), and the bridge module (implemented in JavaScript).

- **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.³

³Bridge Module is based on Mineflayer, which bene-

With these three modules, we construct the base of this simulator and offer many advantages. Most importantly, it is capable of supporting dozens of players entering the game simultaneously. It differs from previous Minecraft simulators (e.g., Malmö, MineRL, MineDojo), all of which necessitate running a Minecraft game client for each player. This client-based approach comes with a notable drawback - it incurs substantial resource costs and 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, allowing support for up to 48 agents simultaneously accessing Minecraft. Additionally, with the introduction of the Mineflayer, a new mode of execution - code, is offered. In contrast to generating textual action descriptions and training an external controller module (Cai et al., 2023) for transforming the plan to the executable code, this simulator is the first language-model-friendly simulator, providing the code instead of the high-level action description. Through the code, it can directly execute plans generated by the language model. Beyond its language-model-friendly nature, MineLand’s three features, namely large-scale multi-agents, limited multimodal senses, and physical needs, make it ideal for developing multi-agent simulators. 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.

2.2. Observation Space

Our observation space is designed to be compatible with almost all APIs of the popular MineDojo framework. We provide information related to touch, vision, and sound: 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 together provide the agents with multimodal senses. Note that this information is all raw perceptual information.⁴ Additionally, we detail the mechanisms and limitations implemented for vision and hearing, including distance attenuation, environmental obstructions, and directional constraints, to model the limited senses.

fits from an excellent community (<https://github.com/PrismarineJS>). We have improved community tools.

⁴Besides 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.

2.3. State Space

Previous simulators focus on task-oriented activities, thus containing state space of inventory and equipment. We design a fascinating state space that blends task-oriented activities with the rhythms of daily life. For the rhythms of daily life, we define the states of agents themselves: physical needs like sleep and hunger. Basic physical needs are the foundation that leads to daily behavior. Here are detailed explanations:

- **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.
- **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.

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 and socializing, 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 control, MineLand includes basic actions like walking, running, jumping, and interacting with objects. We also incorporate specific task-related actions like mining resources, chopping wood, or crafting tools. Inspired by Wang et al. (2023), these low-level actions are implemented in the form of code. High-level actions provide encapsulated behaviors that may consist of several or hundreds of low-level actions. High-level actions are suitable for complex tasks that require longer computation times. Imagine agents navigating the world, dodging obstacles, and manipulating tools. These complex tasks generate an action sequence (i.e., code), allowing the simulator to continue executing the action sequence, skip some steps earlier, or be interrupted by some special event.

The execution of the code is divided into several steps, with each step lasting 50-200 milliseconds⁵. 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.

2.5. Communication

Agents in MineLand are allowed to communicate with each other, including auditory information, body language (via visual perception), and sharing information in text media. If an agent wants to communicate with other agents, it can directly chat through Minecraft’s message bar, and MineLand will detect it. Only when the distance between other agents and the sending agent is less than a certain threshold, will other agents receive messages. Most importantly, the new message is allowed to interrupt the executing code and execute this message directly before the previous code has ended. This means that 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. This communication mechanism was not supported in the previous work.

3. MineLand Benchmark Suite and Dataset

Benchmark Suite offers a wide and diverse range of tasks. These tasks are divided into three categories: programmatic tasks, creative tasks, and hybrid tasks. For each task, customizing the number of players is supported. Competitive mode can be used to measure the differences in capabilities between different AI agents, as well as to develop adversarial learning algorithms. Next, we will provide a detailed introduction to each type of task. Refer to Appendix B for more details.

3.1. Programmatic Task

This type of task has a clear success criterion so that MineLand directly determines whether the task is completed based on the environmental feedback. We referred to MineDojo for the design of programmatic tasks. In total, MineLand has 4499 programmatic tasks.

Each Task T is defined as a 5-tuple: $T = (G, \mathcal{G}, \mathcal{I}, f_{suc}, S)$.

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

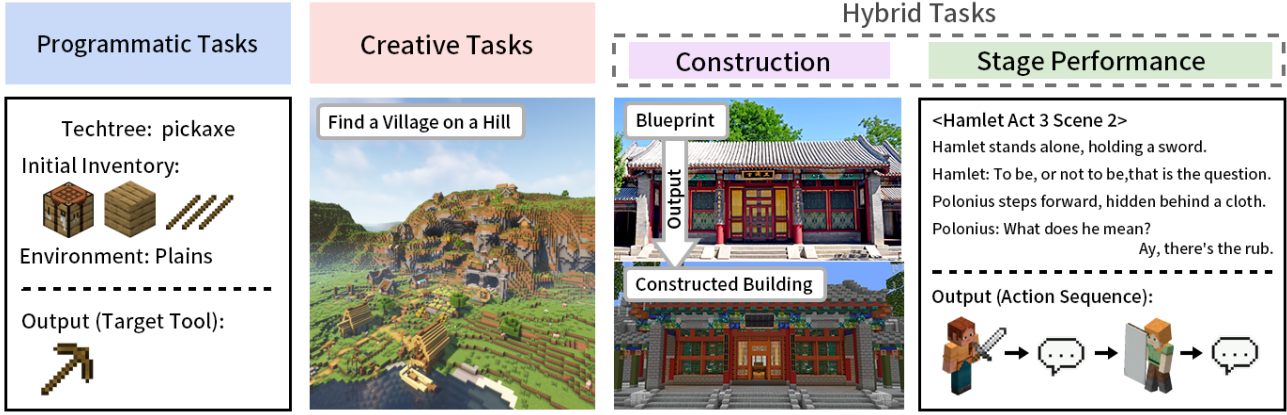


Figure 3. Illustration of Task. We have expanded the number of programmatic tasks and creative tasks by 2 times, compared to MineDojo). Additionally, we have introduced hybrid tasks that combine the features of programmatic tasks and creative tasks. Customizing the number of players is supported. For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

G refers to the task goal that needs to be completed, which will be presented in the form of a short sentence, for example: “Harvest 1 cobblestone with 1 wooden pickaxe”. $\mathcal{G}$ is guidance, which refers to the steps taken to achieve the task goal. These steps are automatically generated by OpenAI’s *gpt-4-1106-preview* based on the goal. It is presented in the form of several sentences. $\mathcal{I}$ is the initial condition of the task, for example, initial inventory, spawn position, and nearby enemies. f_{suc} is the Success Criterion used to determine whether a task is successful. S is a set of parameters set by developers to adjust task settings. These parameters include the number of agents, cooperative mode, competitive mode, etc.

3.2. Creative Task

Compared with programmatic tasks, there is no clear success criterion for creative tasks. Creative tasks is defined by a 4-tuple: $T = (G, \mathcal{G}, \mathcal{I}, S)$. MineLand’s creative tasks include all of the tasks in MineDojo. Besides, we construct 12 new tasks. There are 1536 creative tasks in total.

3.3. Hybrid Task

Hybrid task combines the characteristics of programmatic tasks and creative tasks. These tasks do not have a unique ground truth but have some references. 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. $\mathcal{D}$ can be some key rules, constraints, or key evaluation indicators.

Hybrid Tasks have two types of tasks: Construction Tasks and Stage Performance Tasks. Construction Task aims to provide a blueprint for a building or scene, and agents

are expected to build these buildings or scenes based on the blueprint. The blueprint is the reference $\mathcal{D}$, and MineLand gives a score based on whether the constructed buildings meet the blueprint’s expectations. Blueprints can be pictures of buildings and scenes in real life, or Minecraft-style pictures. Moreover, as blueprints may be partial or 2D, it is meaningless to directly compare 3D buildings with 2D drawings. We use ORB (Ruble et al., 2011) to calculate the similarity between the pictures of the constructed buildings and the pictures of the blueprint. ORB f_{score} is calculated by the formula of $f_{score} \in [0, 1]$ represents the similarity between the constructed building picture and the blueprint.

Stage Performance Tasks aim to provide a script of a drama consisting of several behaviors, which may be a single action or an emotional expression. Agents need to perform the script. MineLand provides a comprehensive score for agents based on the accuracy of the Agent’s actions in terms of time, order, and correlation. The metrics of Stage Performance Tasks are as follows. Seq_{agent} represents the action sequence generated by the agent, while Seq^* is the ground truth. Let n and m be the lengths of Seq^* and Seq_{agent} respectively. And let LCS be the length of the Longest Common Subsequence of Seq^* and Seq_{agent} . After that, the final score is calculated as $f_{score} = \frac{LCS}{n} + \frac{LCS}{m}$.

4. Alex Agent

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, craft-

ing, or construction. We propose Alex⁶, a VLM-based approach as shown in Figure 4, to balance daily routines and tasks. Alex supports both individual daily-life goals and community-based task-oriented goals.

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 Appendix C for more details) and introduced one new component: the multi-tasking component. Additionally, Alex exhibits different personality traits predefined in the system prompt.

Multitasking Component During multitasking, 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 mechanism of multi-tasking ability to enhance the agent’s attention control and working memory abilities inspired by Multitasking theory from Cognition field (Salvucci & Taatgen, 2008). With the multi-tasking mechanism, Alex allows for simultaneously simulating and executing intricate coordination and scheduling with multiple tasks.

For attention control, Alex 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 agent, this involves saving and restoring internal states when switching between communication activities and goal-driven actions to avoid disrupting progress or forgetting ongoing 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 up to 16 agents simultaneously while providing a visual display. When visual display is disabled, the number of concurrently running agents increases to 48. Furthermore, as depicted in Table 1, when MineLand and Malmo both run 8 agents, MineLand’s CPU and memory usage are

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.

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 (Perez-Liebana et al., 2019)), 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.

Table 1. Part of comparison table of performance of Minecraft simulators. The full table is in Table A.1. MineLand_{headless} means MineLand without vision. CPU and MEM both means the average of CPU time and memory usage during initialization phase and five-minute run.

SIMULATOR	AGENTS	CPU	MEM
MINELAND	8	2.81%	7.07GB
MINELAND _{w/o vision}	8	1.88%	2.94GB
MINELAND _{w/o vision}	48	2.94%	5.38GB
MALMO	8	7.90%	18.63GB

5.2. 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. All experiments. Unless specified, all experiments in Section 5 are set to this default setting. See Appendix D 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 2. 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 2, 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} strug-

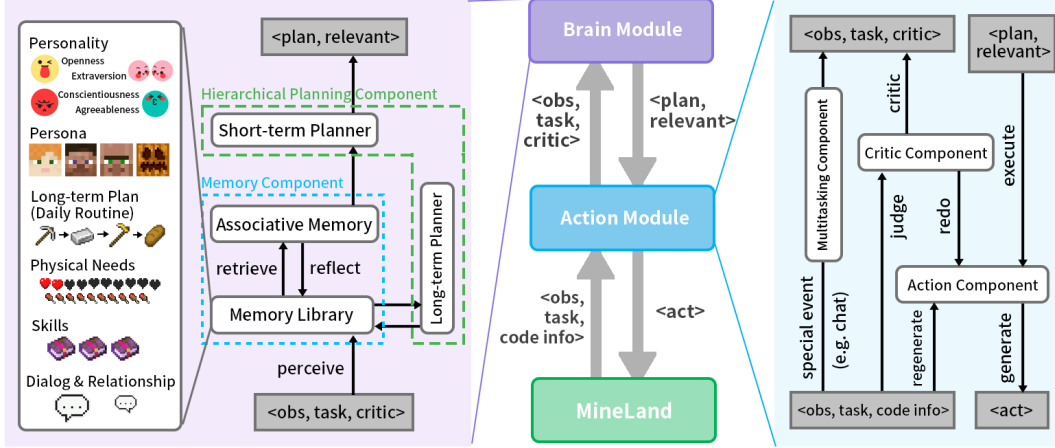


Figure 4. Illustration of the architecture of Alex.


 Figure 5. Trajectory display of the agent in MineLand and MineLand_{w/o vision}, along with the rationales for the agent’s decision.

gels 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 5 and observe that the primary motivation behind the short-term plans generated by agents in MineLand is always visual information, enabling them to perform more appropriate actions.

5.3. Experiments of Multitasking

To validate the impact of multitasking support in the simulator, we conduct the obsidian mining task, which takes over 8 minutes and requires multiple steps to complete. During

these steps, two types of events activate the multi-tasking component and interrupt the mining process:

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 other agent.

Hurt event: The agent gets hurt. 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.

The results reveal that Alex_{w/o multitasking} can’t process events timely, but counterattack zombies after mining, resulting in Alex being killed by the zombie. In contrast, Alex with a multitasking component is capable of managing multiple events (mining while hearing warnings from peers about zombies), autonomously determining their priority, and addressing the higher-priority events first. Hence, multitasking is an essential mechanism.

Table 3. Comparison between Alex and Alex_{w/o multitasking}. In all ten runs, we count the number of successfully handling multitasks. Handling multitasks refers to the agent successfully completing multiple tasks by appropriately prioritizing them.

AGENT	HURT EVENT	CHAT EVENT
ALEX	9/10	2/10
ALEX _{w/o multitasking}	0/10	0/10

5.4. Experiments of Limited Senses

To assess the impact of limited senses, we carry out the “shearing the sheep” task. An agent is required to locate a sheep to complete the task. In the case of an agent with limited visual senses, if there are no sheep within its field of view, it cannot shear a sheep. Therefore, the agent communicates with other agents to find the whereabouts of a sheep. On the contrary, an agent with unlimited visual senses is cognizant of the presence of a sheep in another location,

prompting it to explore the surroundings to locate one.

5.5. 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* without 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 may refer to Appendix G for more details of short-term plans and actions generated chronologically in MineLand *w/o physical needs* and MineLand.

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 without physical needs lean towards exploring and collecting resources. This indicates that physical needs’ importance to simulate real life.

5.6. 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.

5.7. 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 atop Voyager, where Alex can “cheat” (i.e., obtain the location of the target) only when it can visually discover or reason about the target’s presence nearby.

As shown in Appendix F, 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.

5.8. Social Dynamics

In the “unlocking tools” task with two agents, two agents 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 for this is that, in adversarial 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 adversarial 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. We assigned the personality trait of high openness to both agents. Under this condition, the agents tended to establish collaboration and engage in mutual communication. When no personality was set for the agents, they tended to work independently.

Table 4. The number of code iterations needed per agent to unlock tools made of various materials is determined under three conditions. These conditions include a single agent, two agents in a cooperative relationship, and a competitive relationship. Each experiment is repeated three times, and the success rate is 100%.

RELATIONSHIP	WOODEN	STONE	IRON
SINGLE AGENT	7±2	10±3	25±7
COOPERATIVE	13±5	20±7	49±10
COMPETITIVE	6±2	10±3	27±10

6. Conclusion

Traditional multi-agent simulators have facilitated rich explorations of collective behavior, often under the assumption of perfect information and limitless capabilities. We propose a multi-agent simulator that bridges this gap through three aspects: large-scale agents, limited multimodal senses, and physical needs. The advantage carries potential broader impacts across various domains as discussed in Appendix A.

References

- Alderfer, C. P. An empirical test of a new theory of human needs. *Organizational behavior and human performance*, 4(2):142–175, 1969.
- Bai, J., Zhang, S., and Chen, Z. Is there any social principle for llm-based agents? *arXiv preprint arXiv:2308.11136*, 2023.
- Bates, J. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, 1994. doi: 10.1145/176789.176803.
- Binz, M. and Schulz, E. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120, 2023.
- Bledsoe, W. I had a dream: Aai presidential address. *AI Magazine*, 7(1):57–61, 1986.
- Cai, S., Wang, Z., Ma, X., Liu, A., and Liang, Y. 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)*, pp. 13734–13744, June 2023.
- Cao, Y., Yu, W., Ren, W., and Chen, G. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012.
- Card, S. K., Moran, T. P., and Newell, A. The psychology of human-computer interaction. 1983.
- da Rocha Costa, A. C. *A Variational Basis for the Regulation and Structuration Mechanisms of Agent Societies*. Springer, 2019.
- Dill, K. and Martin, L. 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.
- Doyal, L. and Gough, I. A theory of human needs. *Critical Social Policy*, 4(10):6–38, 1984.
- Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=rc8o_j8I8PX.
- Gong, R., Huang, Q., Ma, X., Vo, H., Durante, Z., Noda, Y., Zheng, Z., Zhu, S.-C., Terzopoulos, D., Fei-Fei, L., et al. Mindagent: Emergent gaming interaction. *arXiv preprint arXiv:2309.09971*, 2023.
- Guss, W. H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., and Salakhutdinov, R. Minerl: A large-scale dataset of minecraft demonstrations.
- Heil, J. Perception and cognition. 1983.
- Hollan, J. D., Hutchins, E. L., and Weitzman, L. Steamer: An interactive inspectable simulation-based training system. *AI Magazine*, 5(2):23–36, 1984.
- Horton, J. J. Large language models as simulated economic agents: What can we learn from homo silicus?, 2023.
- Jiang, G., Xu, M., Zhu, S.-C., Han, W., Zhang, C., and Zhu, Y. Evaluating and inducing personality in pre-trained language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=I9xE1JsJfx>.
- John, B. E. and Kieras, D. E. The goms family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3(4):320–351, 1996.
- Johnson, M., Hofmann, K., Hutton, T., and Bignell, D. The malmo platform for artificial intelligence experimentation. In *Proc. 25th International Joint Conference on Artificial Intelligence*, pp. 4246, Palo Alto, California USA, 2016. AAAI Press. URL <https://github.com/Microsoft/malmo>.
- Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., and Koss, F. V. Automated intelligent pilots for combat flight simulation. *AI Magazine*, 20(1):27–42, 1999.
- Juslin, P. N., Scherer, K. R., Harrigan, J., and Rosenthal, R. Vocal expression of affect. *The new handbook of methods in nonverbal behavior research*, pp. 65–135, 2005.
- Laird, J. and VanLent, M. Human-level ai’s killer application: Interactive computer games. *AI Magazine*, 22(2):15, 2001. doi: 10.1609/aimag.v22i2.1558.
- Park, J. S., Popowski, L., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. 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. ISBN 9781450393201. doi: 10.1145/3526113.3545616. URL <https://doi.org/10.1145/3526113.3545616>.
- Park, J. S., O’Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.

- Perez-Liebana, D., Hofmann, K., Mohanty, S. P., Kuno, N., Kramer, A., Devlin, S., Gaina, R. D., and Ionita, D. The multi-agent reinforcement learning in malmö (marl ) competition, 2019.
- PrismarineJS. mineflayer. <https://github.com/PrismarineJS/mineflayer>, 2023.
- Puig, X., Shu, T., Li, S., Wang, Z., Liao, Y.-H., Tenenbaum, J. B., Fidler, S., and Torralba, A. Watch-and-help: A challenge for social perception and human-ai collaboration. In *International Conference on Learning Representations*, 2020.
- Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z., and Sun, M. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.
- Riedl, M. O. Interactive narrative: A novel application of artificial intelligence for computer games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI’12)*, pp. 2160–2165, 2012.
- Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.
- Salvucci, D. D. and Taatgen, N. A. Threaded cognition: an integrated theory of concurrent multitasking. *Psychological review*, 115(1):101, 2008.
- Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., and Schwamb, K. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1):15, 1995.
- Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- Williams, R., Hosseinichimeh, N., Majumdar, A., and Ghafarzadegan, N. Epidemic modeling with generative agents. *arXiv preprint arXiv:2307.04986*, 2023.
- Wimmer, S., Pfeiffer, A., and Denk, N. The everyday life in the sims 4 during a pandemic. a life simulation as a virtual mirror of society? In *INTED2021 Proceedings*, pp. 5754–5760. IATED, 2021.
- Yu, J., Wang, X., Tu, S., Cao, S., Zhang-Li, D., Lv, X., Peng, H., Yao, Z., Zhang, X., Li, H., et al. Kola: Carefully benchmarking world knowledge of large language models. *arXiv preprint arXiv:2306.09296*, 2023.
- Zhang, H., Du, W., Shan, J., Zhou, Q., Du, Y., Tenenbaum, J., Shu, T., and Gan, C. Building cooperative embodied agents modularly with large language models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.
- Zhuge, M., Liu, H., Faccio, F., Ashley, D. R., Csord s, R., Gopalakrishnan, A., Hamdi, A., Hammoud, H. A. A. K., Herrmann, V., Irie, K., et al. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*, 2023.

A. 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.

B. 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 B.1. Figure 6 displays specific task data.

Table B.1. Statistical analysis of the tasks in 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

C. Details of Alex

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 Wang et al. (2023). AI agents perceive the environment, make plans, execute plans independently, and interact with other agents. Internally,

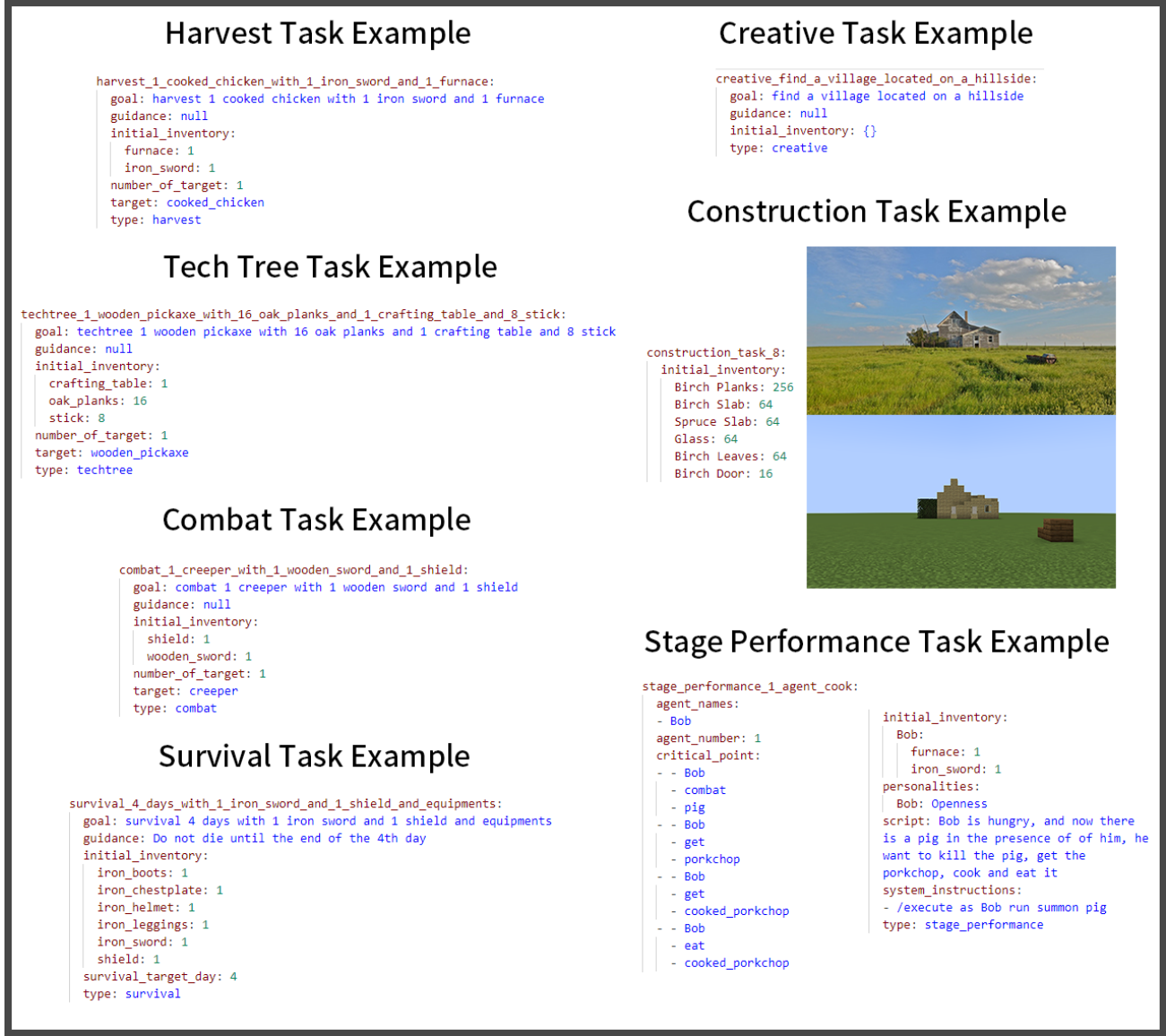


Figure 6. Illustration of example for different tasks.

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.

C.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 database⁷. 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.

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.

C.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.

C.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 Park et al. (2023), 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, 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.

⁷We leverage Chroma in Alex for storage and retrieve memory. <https://www.trychroma.com/>

C.4. Algorithm

Algorithm 1 demonstrates Alex’s core algorithm, which corresponds to the process in Figure 4.

Algorithm 1 Core algorithm of Alex.

```

Input: observation space obs, task info task, code info code
state, description  $\leftarrow$  Distributor(obs, task, code)
if state = None then
    return None
end if
if state = Critic then
    state, description, critic  $\leftarrow$  Critic Component(obs)
end if
if state = Action then
    Perceive(obs, critic)
end if
if state = Brain then
    Perceive(obs, critic, code)
    Generate Long-term Plan(obs, task)
    retrieved  $\leftarrow$  Retrieve(obs)
    Plan(obs, task, retrieved)
    state  $\leftarrow$  Action
end if
act  $\leftarrow$  Execute(obs, description, code, critic, short-term plan, relevant info)
return act

```

D. Details of Hyper-Parameters

Alex leverage OpenAI’s *gpt-4-vision-preview* API 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 5. “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” for leveraging AI to 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.

E. Minecraft Simulator Performance Comparison

We evaluate the number of agents that MineLand can support and compare MineLand with other popular Minecraft simulators, in Table A.1. 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 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 16 agents simultaneously while providing a visual display while it supports 48 agents without a visual display. Furthermore, 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 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, so it supports 8 agents.

Table A.1. Comparison between MineLand and other popular Minecraft simulators under different conditions.

SIMULATOR	CONDITIONS	NUMBER OF AGENTS	AVG CPU TIME	MAX CPU TIME	AVG MEM USAGE
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	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
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

F. Experiments of Single Agent

As shown in Figure 7, even with this restriction, Alex demonstrates a strong ability for long-term planning.

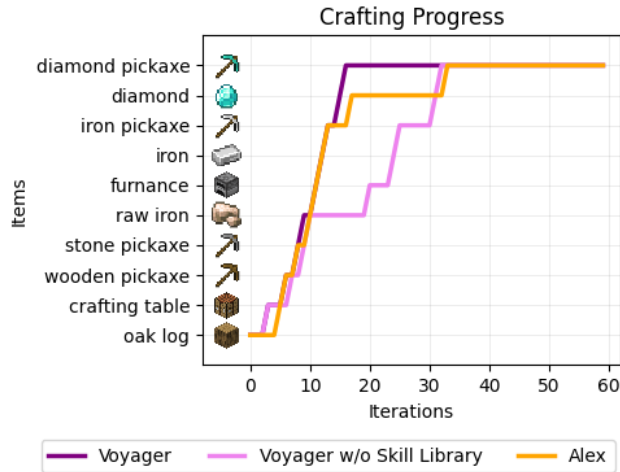


Figure 7. Comparison of our single-agent Alex and other SOTA baselines.

G. 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 bread. 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* without physical needs.

Alex in MineLand can survive one day in his shelter and successfully completed the task. 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.

From the above plans and actions, we observe that, 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.

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 without physical needs lean towards exploring and collecting resources. This indicates that physical needs’ importance to simulate real life.

H. Experiments of Comparing VLMs and LLMs

As shown in Table A.2, 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 the gpt-4-1106-preview API for the complete action component, and Alex, which uses different VLMs or LLMs for the action component.

Table A.2. 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.

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 $\pm$ 3(3/3)
<i>gpt-4-vision-preview</i>	4 $\pm$ 2(3/3)	7 $\pm$ 3(3/3)

I. 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 hindering 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.

J. Related Work

J.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 (da Rocha Costa, 2019). From two individuals (Cao et al., 2012; Puig et al., 2020), through four individuals (Wimmer et al., 2021), to sandbox games Smallville with twenty-five individuals (Park et al., 2023), 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 (Park et al., 2023). 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 (Qian et al., 2023). Additionally, such simulation of interaction has had a significant effect in many other fields. For example, it can replicate realistic social phenomena (Dill & Martin, 2011; Park et al., 2022), enhance social robots (Bates, 1994; Bledsoe, 1986). They can also be used to test social science theories (Binz & Schulz, 2023; Jiang et al., 2023; Horton, 2023), create model human processors for theory and usability testing (Card et al., 1983; John & Kieras, 1996), train people on how to handle rare yet difficult interpersonal situations (Tambe et al., 1995; Jones et al., 1999; Hollan et al., 1984), and support game characters (Laird & VanLent, 2001; Riedl, 2012).

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 (Qian et al., 2023; Park et al., 2023; Williams et al., 2023), current research primarily focuses on a small number of agents (Park et al., 2023; Bai et al., 2023; Zhuge et al., 2023). 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 (Park et al., 2023). Most research in terms of AI agents mimicking daily life routines, focused on two agents (Puig et al., 2020). Simulators that remind people of sandbox games (The Sims) initially support four individuals (Wimmer et al., 2021), then are extended to twenty-five individuals by Park et al. (2023).

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 (Heil, 1983). 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 (Juslin et al., 2005). 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 (Park et al., 2023), 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 (Doyal & Gough, 1984; Alderfer, 1969). Existing simulators are designed to simulate believable human behavior in daily-life activities Park et al. (2023); Bai et al. (2023). 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.

J.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, Guss et al.; Fan et al. (2022) 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 Gong et al. (2023) 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. Park et al. (2023) 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.

J.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 (Gong et al., 2023), and they possess substantial world knowledge (Yu et al., 2023). Moreover, VLMs like CLIP (Radford et al., 2021) offer a versatile visual-language representation that aligns with language and enables zero-shot visual recognition capabilities for potential AI agents.

K. Prompts

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 you 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 is 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 8. Part of 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 9. Part of long-term plan generator