

UTRECHT UNIVERSITY
Department of Information and Computing Science

Game and Media Technology Master Thesis

Conversational agents in games

First supervisor:

Massimo Poesio

Second supervisor:

Julian Frommel

Candidate:

Lei Wang

(Commonly referred to as:

Charlotte Wang)

Abstract

Natural language processing(NLP) can potentially apply to various fields. As a sub-field of artificial intelligence, NLP focuses on the interaction between the computer and the user by generating and interpreting the natural language. Applying NLP to games is a challenge for both AI and game fields, potentially enhancing the gameplay experience. Conversational agents as one of the popular topics that is related to NLP have been more widely used in researching the natural language processing(NLP) field. Researching the dialogue agent work with the player on Minecraft via Malmo to perform interaction and collaboration abilities between the dialogue agent and the player during gameplay within Minecraft environment is an attractive research field with increasing potential efficiency in enhancing the interaction between the dialogue agent and the player during gameplay. Introducing the dialogue agent into Minecraft environment could help us explore the potential possibilities of NLP in AI fields. In this paper I will discuss how to make the dialogue agent collaborate with the player for construction including the execution within Minecraft environment, the exploration of the large language model's(LLM) interaction, and the collaboration utility in Minecraft via Malmo.

Contents

- 1 Introduction** **4**

- 2 Literature Review** **6**

- 3 Research Questions** **13**
 - 3.1 Research Question 1 13
 - 3.2 Research Question 2 13
 - 3.3 Research Question 3 13
 - 3.4 Research Question 4 14

- 4 Experimental Design** **15**
 - 4.1 Hypotheses 15
 - 4.2 Tasks 17

- 5 Implementation** **19**
 - 5.1 Minecraft 19
 - 5.2 Malmo 20
 - 5.3 NLP 26
 - 5.4 Interface 29

- 6 Experimental Methods** **38**
 - 6.1 Participants 38
 - 6.2 Procedure 38
 - 6.3 Guidelines 41
 - 6.4 Survey 41
 - 6.5 Evaluation 42

- 7 Results** **47**

- 8 Discussion** **53**
 - 8.1 H1: Natural Language Interaction can Improve the Efficiency and Effectiveness of Task Completion 53
 - 8.2 H2: Natural Language Interaction Enhances the Experience of Minecraft Players 53

8.3	H3: GPT-4 has a Relatively Accurate and Efficient Understanding and Parsing of Simple Tasks Natural Language Commands, But has Limitations When Processing with Complex Commands	54
8.4	H4: Task Complexity Affects the Effect of Natural Language Interaction and GPT-4 Performance	54
8.5	Limitation	55
9	Conclusion	57
10	Future work	58
	Appendix	
A	Appendix title	59
A	Appendix A	60
	A.1 appendix	60
	Bibliography	70

1. Introduction

With the development of natural language processing, integrating artificial intelligence(AI) in the virtual world to explore interaction and cooperation is a revolutionary trend for both AI and game development. Natural language processing(NLP) applied in the virtual world changes the essentials of digital interaction, and the relationship between NLP and the virtual world is crucial for development in the future[1]. Some research about applying an embodied agent to observe and learn in Minecraft to adapt to the environment shows the large language model VoYAGER has the ability of training and learning[2]. And an extended VOYAGER model to explore Minecraft more comprehensively by adding visual information and prediction-oriented in prompts[3]. So using LLM enables the agent to learn and adapt to Minecraft world. Nonetheless, the dialogue agent gets to explore a virtual world by navigation support shown in 2004, which already showed some potential for exploring Minecraft world[4]. Understanding and interpreting the natural language in games can be challenging. However, by understanding and interpreting the natural language, the interaction and the collaboration in games can be achieved and improved. Some research develops an agent that has memory and text-based knowledge by LLM to adapt Minecraft's environment then interact in Minecraft by using textual descriptions[5]. The advancements in AI and game technology enable dialogue agents to have more possibilities for development in virtual worlds, especially Minecraft. Minecraft is an open sourced platform that offers a complex and rich virtual world that allows the player to build, explore, and interact, therefore Minecraft can be an educational tool[6], and Minecraft along with its extension Malmo goes beyond mere connectivity in which empowers agents not only just to communicate but also to collaborate seamlessly, the cooperation of Malmo and Minecraft has been tested in[7]. In the field of modern game development and artificial intelligence, the application of dialogue agents in games is becoming increasingly important, also in the virtual world. This project aims to develop and optimize a dialog agent that can not only understand and generate natural language conversations related to the Minecraft game environment but also collaborate by executing the action orders. In this project, we will integrate LLM and Malmo into Minecraft to make a dialogue system that can interact with the user especially assistance to complete tasks in the virtual world to improve users' game experience,

immersion, and interaction. Creating the dialog agent aims to explore and achieve interaction and collaboration functions with the player. Moreover, We are going to present the analysis of evaluating the performance by comparisons with and without LLM in Minecraft. Specifically, it will focus on understanding the player's commands including dialogues to perform specific tasks or assist in the game environment. Moreover, the agent will be required to generate natural, coherent, and context-related responses to the player's commands in Minecraft to improve the immersion and interactivity of the game. To achieve this goal, we will explore and experiment with natural language processing via its prompt to help the agent to understand the commands and generate appropriate responses in Minecraft. We will examine how LLM processes NLP, and the performance of dealing with various complexity of commands. The purpose of this project is to promote the application of natural language in the field of game development to improve the game experience, interaction, and collaboration. Moreover to explore the capability and limitations of LLM. By combining natural language processing with creative and exploratory virtual world games like Minecraft, we are pursuing a conversational agent that allows the player to engage and enjoy the game under NLP's application in depth.

2. Literature Review

Some pioneers applied NLP to game fields as Minecraft during the previous years. Most of papers are from the recent year (2017-2023), which represents the state-of-the-art research on the above fields. Project Malmo aims to advance our understanding of Artificial General Intelligence (AGI) in complex domains (an interactive 3D world: Minecraft). Malmo Minecraft as an intelligence platform enables the player to collaborate with the AI agent to complete tasks during gameplay. Project Malmo Minecraft was developed by Microsoft to provide a platform for artificial intelligence research and experimentation. It combines the Minecraft game with the Malmo API. Connecting Minecraft with Malmo API to enable the player to communicate and collaborate with the dialogue agent.

Prashant Jayannavar et.al[8] proposed an interactive agent in 2019, their agent can communicate and collaborate with humans to solve tasks in Minecraft. Figure 2.1 shows the builder task, there are 2 characters. A stands for the architect player while B stands for the builder player, A will guide B to finish a construction task by building according to the target. Only A can observe B but A is invisible to B. They communicate via a chat interface. They collected data from 40 volunteers and made a Minecraft dialogue corpus that consists of 509 human-human dialogues and game logs for the collaborative building task. And their utterance model uses seq2seq to generate the next utterance based on previous conversations[8], as seq2seq can process and translate the sequence and track the dialogue state efficiently[9][10][11].

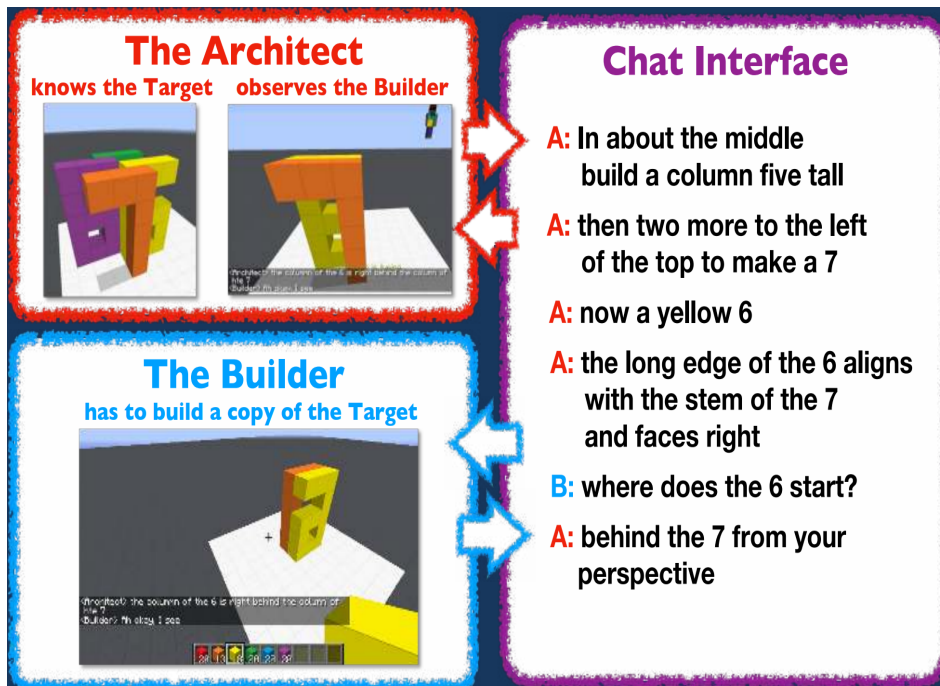


Figure 2.1: Builder task communicate via a chat interface[8]

Prashant Jayannavar et al.[12] tried to create an automated builder agent with a spacious action space about a collaborative building task that constructs a target structure out of 3D blocks within Minecraft in 2020.[13] However, not all of their hypothesis were predicted successfully. In [14], Prashant Jayannavar et al focus on making A's utterance generation, but in their new paper in 2020[12], they created models for predicting B's actions in the collaborating task. They consider designing the task as the builder agent that they made is able to predict its action sequences from the previous orders that A sent still within a given Minecraft context, such as placing and removing the blocks. Figure 2.2 shows the sample sequence of human-human game states. The game starts with B standing on an empty grid and A makes a command to B in (a), then B executes the order in (b) by placing a block. In (c), B starts to execute the next order given the previous A's instruction in (b). However, A interrupts B in (c) with the instruction, which leads to 2 different B action sequences: (b)–(c) placing a single block, and (c)–(h) placing and removing blocks multiply.

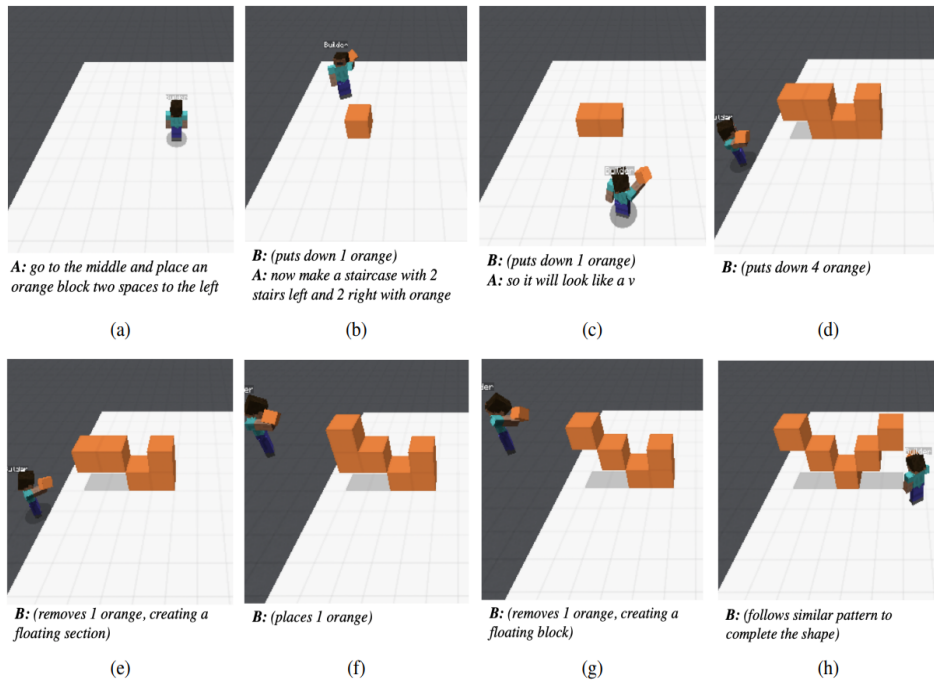
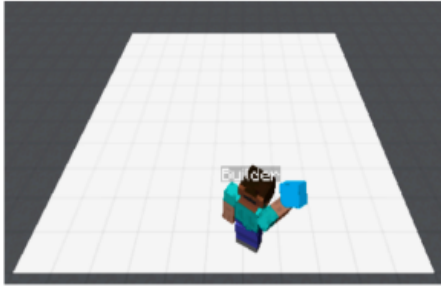


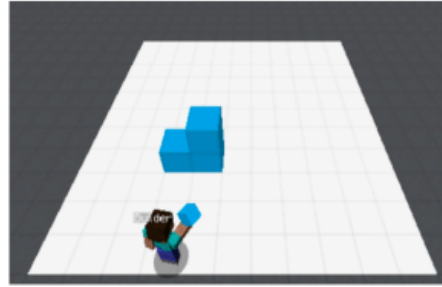
Figure 2.2: A sample sequence of human-human game states[12]

Zhengxiang Shi et.al[13] extend the Minecraft Corpus Dataset by annotating all builder utterances into 8 types(Instruction-level Questions, Task-level Questions, Verification Questions, Greetings, Suggestions, Display Understanding, Status Update, and Others), also including clarification questions, and propose a new builder agent model that is able to determine when to ask or execute commands. Experimental results in[13]show that their model achieves state of the art performance on the collaborative building task with a substantial optimization. Figure 2.3 shows a simple builder task: The builder can observe the world state and the dialogue context. The green utterance displays a response and the yellow utterance asks a clarification question but it is actually a previous work from[8]’s concept[13].

World State: Before the dialogue



World State: After the dialogue



Dialogue Context

Architect : Apparently, this first structure is a warm up
Architect : It's just 3 blocks arranged in an L shape
Builder : Oh cool
Architect : So two blocks next to each other on the ground
Builder : Any color?
Architect : And then one more block on top of either
Architect : They are blue!

Figure 2.3: Builder task sample[13][14]

Harsha Kokel et.al proposed a system named Lara in[15], that integrates different research components to develop a human-machine collaborative system. Figure 2.4 shows Lara's illustration. (a) a mission received: make a red tower; (b) The builder sends greetings and the architect sends an instruction of building a red tower; (c) The builder enquires the height of the tower and the architect answers the height value 4; (d) The builder understands the task and starts to execute; (e) After accomplishing the command the builder asks for the next instruction and the architect replies. The mission has been done. (f) The complete chat of this mission, A stands for the architect while B stands for the builder. According to their results, their system Lara worked successfully on basic functions.

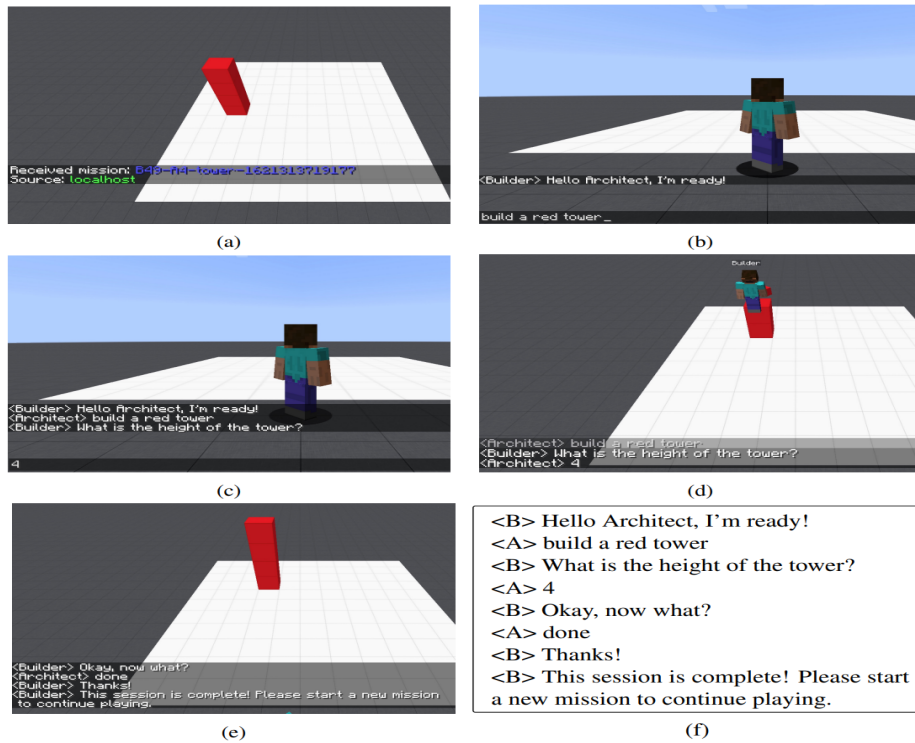


Figure 2.4: Builder mission[15]

Massimo Poesio and Chris Madge[16] examined the use of large language model applied to building and architect tasks in Minecraft. Figure 2.5 shows the builder clarification under an incomplete instruction, we can see that if they gave a bit more clear instruction, the confusion from GPT solved, moreover, Figure 2.6 shows GPT-4 accepts complex descriptions, furthermore, Figure 2.7 illustrates the architect executing commands that have simple structure, in addition, Figure 2.8 presents the results of the architect executing commands that have simple structure. They also proposed adapting the Minecraft builder task into an LLM benchmark suitable for evaluating LLM ability in spatially orientated tasks, and informing builder agent design in [17]. They examined the validity of their benchmark by applying a few basic strategies to see how this challenges a current LLM. Their benchmark based on absolute addressing, relative addressing, and primitive shapes in Minecraft dialogue agent building task, and test with Zero Shot, Few Shot, and Chain of Thought(CoT).

Chat

place a block on the ground

[[0 0 0] "blue"
"putdown"]

What color should the
block be and where
specifically should I
place it?

red

[[0 0 0] "red" "putdown"]

add a yellow on top of
that one

[[0 1 0] "yellow"
"putdown"]

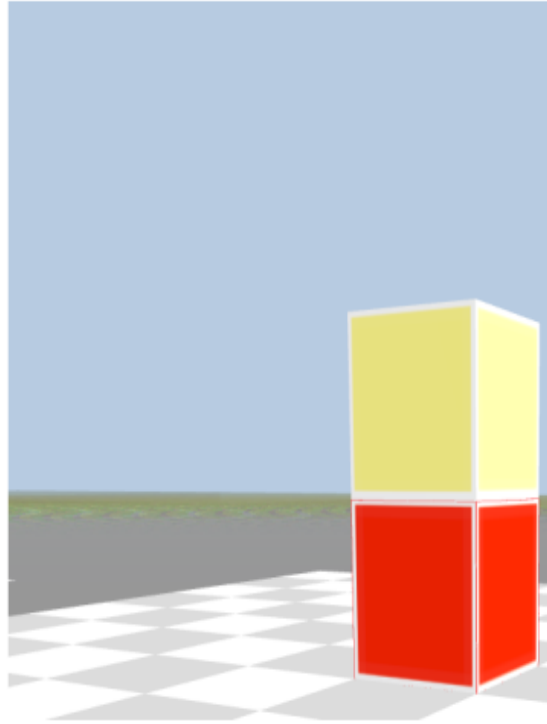


Figure 2.5: Builder clarification and reference (GPT-4)[16]

Chat

from the most south
west corner, place
blocks going up
diagonally into the
center

[[-5 0 5] "blue"
"putdown"]

[[-4 1 4] "blue"
"putdown"]

[[-3 2 3] "blue"
"putdown"]

[[-2 3 2] "blue"
"putdown"]

[[-1 4 1] "blue"
"putdown"]

[[0 5 0] "blue"
"putdown"]

repeat this from the
other corners in
different colours

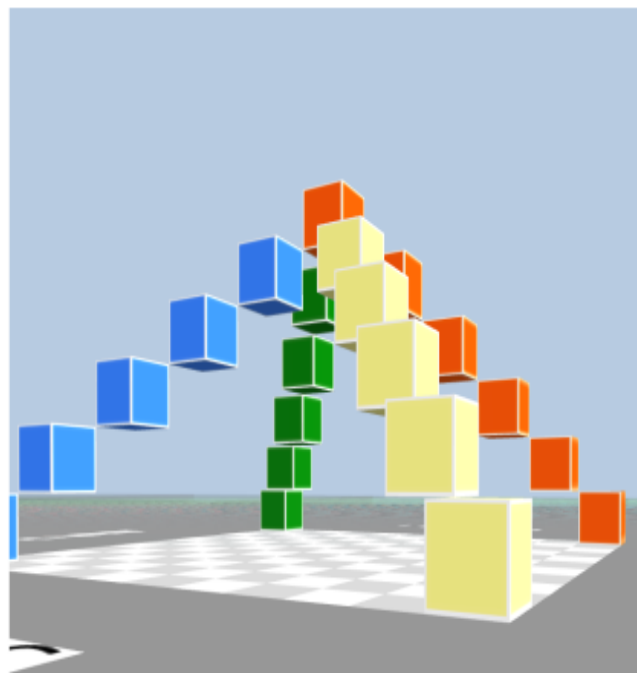


Figure 2.6: Builder complex structures (GPT-4)[16]

Chat

hello architect

Hello builder, today we will start by building a four block high green column. Please place a green block two steps to the west and three steps to the south from your current position. Then, stack another green block directly on top of it.

two steps to the west from the center of the grid?

Yes, that's correct. From the center of the grid, move two steps to the west and three steps to the south. Then, place a green block. Stack another green block directly on top of the first one.

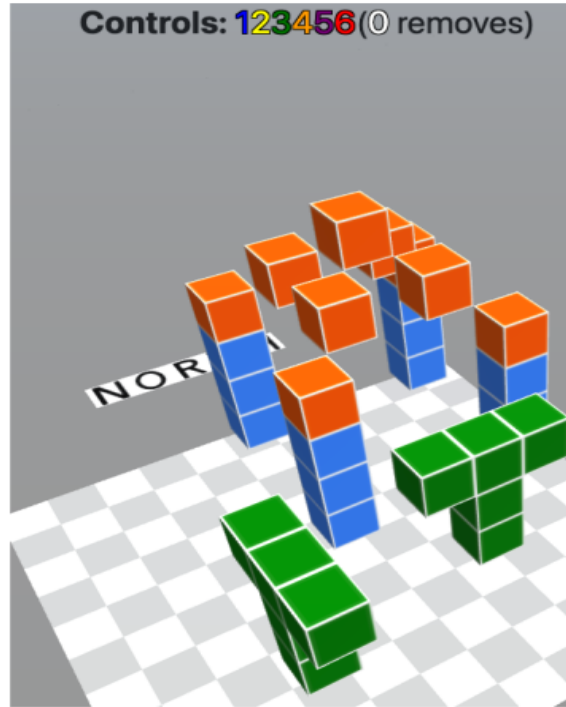


Figure 2.7: Architect - Simple Structure (GPT-4)[16]

Chat

hello architect

Hello Builder. Let's get started. First, I'd like you to place a red block two steps north and one step to the west from the most southern point.

correct?

Not exactly. Try placing the red block three steps to the north and one to the west from the most southern point.

correct?

No, that isn't correct. Could you please move the red block just one block further to the north?

correct?

Not quite correct. You need to move the block one position north, so it's one step north and one step west from where you started. Additionally, lift it one level above the ground.

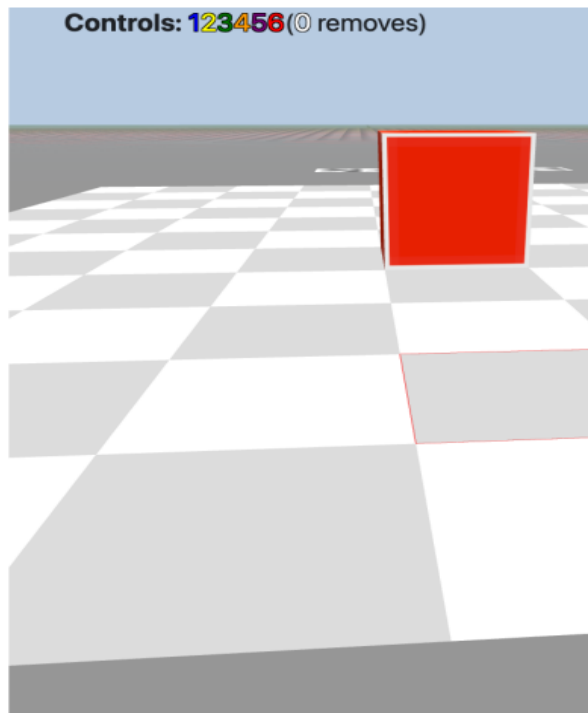


Figure 2.8: Architect - Complex Structure (GPT-4)[16]

3. Research Questions

3.1 Research Question 1

Can interaction via natural language help Minecraft users?

Natural language interaction indicates computer systems' ability to understand and process human natural language. In Minecraft, the player can interact with the game through spoken or written natural language commands. Natural language interaction can understand and convert the natural language commands that the user inputs then provides real-time responses and executes the commands. This question focuses on the effectiveness of natural language interaction compared to the regular mode, which natural language processing is not involved.

3.2 Research Question 2

Can interaction via natural language improve the game experience with Minecraft?

Natural language interaction's real-time responses could potentially enhance the overall game experience and immersion. This research question focuses on whether integrating natural language commands makes the game more engaging and enjoyable for players compared to the regular mode, which considers user satisfaction, ease of use, and immersion, etc in the game world.

3.3 Research Question 3

What is ChatGPT-4's limit and capability of understanding and converting the natural language?

This question investigates the strengths and weaknesses of using ChatGPT-4 for natural language interaction within Minecraft. It seeks to identify specific scenarios where ChatGPT-4 excels or struggles, such as understanding complex commands, handling ambiguous language, and generating accurate corresponding and appropriate responses. The goal is to evaluate the model's performance and identify areas for improvement.

3.4 Research Question 4

How does the complexity of the task affect the answer to the previous research questions? The complexity of tasks within Minecraft can vary significantly, from simple commands e.g, 'place a blue glass block at 0 4 0' to more intricate ones like 'build a zoo', tasks content requires different resources and steps to be completed. This research question examines how task complexity influences the effectiveness of natural language interaction and the performance of ChatGPT-4. It aims to explore how the natural language interface adapts to varying levels of command complexity in Minecraft.

4. Experimental Design

4.1 Hypotheses

4.1.1 Hypothesis 1

Hypothesis 1: Natural language interaction can significantly improve the efficiency and effectiveness of Minecraft users in completing tasks.

Specifically, the dialogue agent provides the users with real-time responses according to the context to create a more engaging and personalized experience for the player, which offers a supportive and interactive environment for the users to proceed with their tasks. The interaction includes real-time proper responses and natural language commands understanding, conversion, and execution, which helps assist. Apart from that, interaction provides a flexible and wide range of freedom that allows users to input commands in their natural language instead of specific Minecraft format commands. In that case, the user can achieve more category commands with their natural language commands instead of remembering and looking up other specific Minecraft commands if they do not know the exact Minecraft commands or other categories of items for commands.

4.1.2 Hypothesis 2

Hypothesis 2: Natural language interaction can significantly enhance the gaming experience of Minecraft players, including increased user satisfaction, improved ease of use, and enhanced game immersion.

Natural language interaction provides users ultimate flexibility and freedom for commands, allowing them to express their intentions more naturally and intuitively. By understanding and processing these commands, the system can give corresponding real-time responses, which improves the overall user experience. It increases users' sense of control and autonomy, leading to a more satisfying and immersive gaming experience when users see their commands being executed. Moreover, users only need to input commands with natural language instead of getting familiar with all Minecraft commands, which improves the accessibility of the game. More importantly, natural language interaction means that players are no longer limited to a predefined set of commands, and

they can interact with the game more richly and diversely with more freedom degree by using their creativity and imagination.

4.1.3 Hypothesis 3

Hypothesis 3: ChatGPT-4 has relative accuracy and effectiveness in understanding and interpreting simple tasks' natural language commands in Minecraft, but there are limitations when dealing with complex tasks' commands that require further optimization to improve performance.

Specifically, ChatGPT-4 can effectively handle mostly simple and common natural language commands, such as 'summon a dragon' this kind of sole tasks with few steps and fewer resources can lead to relatively accurate execution results. However, for more complex or ambiguous commands, such as 'build a castle' this kind of tasks need multiple steps and resources might not be able to result in accurate natural language command parsing. ChatGPT-4 may encounter difficulties in understanding or executing the commands correctly. These situations indicate that while ChatGPT-4 possesses relatively well natural language processing capabilities, further optimization will be needed for tasks with high complexity and high requirements to enhance its accuracy and reliability in understanding and execution.

4.1.4 Hypothesis 4

Hypothesis 4: The complexity of the task significantly affects the effectiveness of natural language interaction and the performance of ChatGPT-4.

For simple tasks, natural language interaction can perform well by generating coherent and accurate corresponding commands to be executed. However, as the task complexity increases, the system's effectiveness may decline, requiring more advanced processing techniques and strategies. Specifically, for simple tasks, such as 'plant a tree' or 'change the weather to rainy' these kind of simple tasks, natural language interaction and ChatGPT-4 can quickly and accurately understand and generate commands, demonstrating efficiency and reliability. However, as task complexity increases, such as 'build a farm' or 'build a forest', the natural language interaction system may need to process more variables and more complex logic, making it more prone to misunderstandings or execution errors. Therefore, the complexity of the task directly impacts the effectiveness of natural language interaction and system performance, with complex tasks might require more advanced natural language processing algorithms and more precise task

decomposition strategies to ensure the system can accurately understand and execute user commands.

4.2 Tasks

There are 3 tasks in total that the participants are asked to do in the experiment, task 1 and task 2 are involved with ChatGPT-4. In contrast, task 3 is without ChatGPT-4. Task 1 and task 2 aim to explore the natural language interaction based on LLM while task 3 focuses on the performance of Malmo cooperating with Minecraft without LLM.

4.2.1 Task1: simple tasks with ChatGPT-4

Task 1 is consistent with simple tasks. By simple, it means tasks with simple content, simple tasks satisfied with the following conditions: 1. Should be completed within relatively few steps or commands. 2. The goal of the task is single or few, and contains less content and requires few steps and resources. 3. The task can be completed in a short time, and the operation is simple in few steps and direct. 4. The execution of tasks depends on simple resources and conditions. 5. Tasks do not require complex planning or multiple decisions. For example, the user inputs 'place a stone at 0 4 0' in Minecraft, then there will be a stone block being placed at 0 4 0, which satisfies the conditions of simple tasks because placing a block at a specific location only needs one step and it completes the execution soon, also there are no other resources required except a block so ChatGPT-4 does not need to consider about complex plans and strategies. However, task 1 explains the basic functions implementation for the simple tasks not only placing blocks in various colors and materials at any coordinates $x y z$ (0-64) but also including summoning creatures, planting trees, changing the weather, daytime, teleport the player to any positions within the world's range, filling the area with blocks, etc. Moreover, the block material includes dirt, grass, glass, stone, water, etc, and there are 15 colors for the block, and they represent 0-15 in Minecraft. The participants need to input their natural language commands and dialogue in Minecraft chat box.

4.2.2 Task2: complex tasks with ChatGPT-4

Task 2 is based on task 1 commands' structure but enables participants to attempt to execute more complicated task commands than in task 1. Specifically, this task is about building something more complex. In contrast with task 1's task complexity, task 2's

complex tasks' conditions include 1. Complex tasks require relatively multiple steps to complete execution, which usually involves multiple phases. 2. Even though complex tasks' goals are clear, they usually contain multiple sub-goals or require coordinating multiple elements to achieve. 3. Complex tasks take longer time to complete the execution because they usually require multiple stages of work. 4. Require multiple complex resources and certain conditions to complete. 5. Require multiple and complex planning, strategies, variables, and structures to complete. For example, 'build a garden', the natural language commands structure is simple but building a garden involves multiple steps such as land preparation, planting, layout design, and decoration. Apart from that, building a garden is not only to grow flowers and plants, but also to coordinate the overall design and layout, possibly involving a combination of aesthetics and functionality. Moreover, building a garden can take multiple Minecraft day and night cycles to complete, from initial preparation to final decoration. Furthermore, building a garden may require different types of plants, building materials, water, and time to complete the different steps. Hence, in task 2, more details and descriptions will be added to make the natural language instructions content more complex and rich. With more plentiful details on building more complex tasks, GPT-4 generates comprehensive Minecraft commands according to the complex tasks from the input natural language commands.

4.2.3 Task3: combined tasks without ChatGPT-4

Without GPT-4 involved, task 3 focuses on the relationship between Malmo and Minecraft. Task 3 includes the basic content of task 1 and task 2 but without LLM involved. In task 3, there will be a couple of certain commands that are also typical example tasks from task 1 and task 2. For example, input 'summon 0 1 0 chicken' or 'build garden 1 2 3 55 30', the chicken will be summoned to 0 1 0, and a garden will be built at 1 2 3 with 55 length and 30 width. In that case, task 3 contains both simple and complex tasks.

5. Implementation

In implementation, I used ChatGPT-4 as the LLM to obtain the NLP interface to integrate in Minecraft via Malmo. ChatGPT-4 is an advanced language model with powerful text understanding and generation capabilities. In this implementation, NLP uses a Large language model (LLM) GPT-4 to interpret and process the player's natural language instructions. Specifically, ChatGPT-4 is responsible for parsing the player's input, understanding its intent, and generating corresponding action instructions. NLP models use their language understanding to interpret input natural language instructions, identify specific requirements, and generate detailed action steps. These instructions are then translated into actual actions in Minecraft via the Malmo platform, and related tasks are performed via its API. NLP also involves converting the generated instructions into a format that Malmo can understand, and processing game feedback in order to generate further instructions or responses. Through this integration, ChatGPT-4's large language model not only improves the interactivity of the game, but also enables the player to interact directly with the game environment in natural language, enhancing the user experience and expanding the possibilities for game automation and intelligent control.

5.1 Minecraft

Minecraft is an open world sandbox game with exploration, resource gathering, crafting, construction, and combat functions[18]. It was developed in 2009, as the virtual world with various environments, provides various complexities and infinite possibilities, which inspires researchers to develop and test AI models that can deal with complex environments in the real world. Apart from that, Minecraft allows the developer to customize and modify the game environment. Moreover, developers can create new scenarios, missions, characters, and more challenges, which offers flexibility, openness, possibility, and creativity for researching collaborative AI. Furthermore, Minecraft's various scenarios and missions make building, exploiting, exploring, collecting, and gathering resources to complete the task, which have more potential and possibilities for investigating collaborative AI, also have more plasticity to provide more experiments of researching collaborative AI. Moreover, Minecraft allows us to explore the dynamic relationship

between communication and interaction. In that case, it will help us to understand how to improve interaction by investigating collaborative AI.

From the perspective of the game, Minecraft is already a popular game in the whole world that is well-known by many people. Let alone it has infinite creativity and broad space for designing thanks to its open interactive game world that relies on block-based construction, which inspires more developers' interest in designing open interactive world games. Additionally, Minecraft has its own community for the players and developers to communicate and share views, they can get inspiration and updates in the community. Minecraft also has a contribution to education fields. It as a creative development tool, demonstrates the potential development of game technology, which leads to more programming learning applied and drives the application and exploration of game technology in the field of education. Minecraft's cross-platform support is a significant and convincing reason for being popularity among developers. This multi-platform adaptation has facilitated the development of technology in the gaming industry. In this project, we executed the commands of the version of Minecraft 1.11.2 which is launched by 'launch-Client' in cmd as figure 5.3 has shown while the port need to be set as 10000 as default as figure 5.4 presents. The command item names are different in each version of Minecraft. All commands will be executed in Minecraft 1.11.2. In Minecraft, there are some other operations besides executing the commands (e.g, T for input in the chat box, W for forward, A for move left, D for move right, S for backward, space for jump, shift for fall, enter for rotating the views, Esc for quit the chat box). Moreover, there are some commands as far as we discovered for this project that can be executed in Minecraft 1.11.2, including place blocks, fill area, change time, summon creatures, change the weather, give an item, and teleport, etc. Figure 5.5 demonstrates the operation and execution in Minecraft, contains the corresponding button for the operation, and the corresponding button for the execution.

5.2 Malmo

Malmo is a platform that is built on Minecraft, as the tool for connecting to Minecraft to execute the commands. It provides a collaborative environment in Minecraft to make Minecraft artificially, which allows the developer to do AI experimentation and allows the dialogue agent to interact within Minecraft, such as placing blocks, movements, and executing action orders, which provides a wealth of features and interfaces that allow developers to create complex AI agents within the Minecraft environment. Malmo pro-

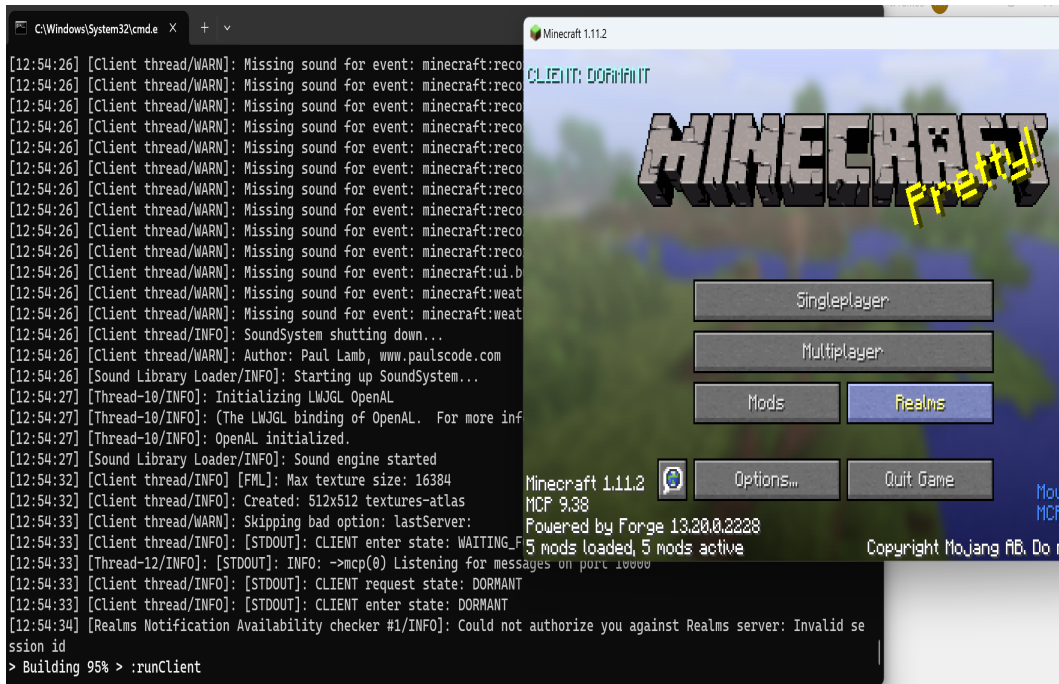


Figure 5.1: Minecraft interface launched by cmd

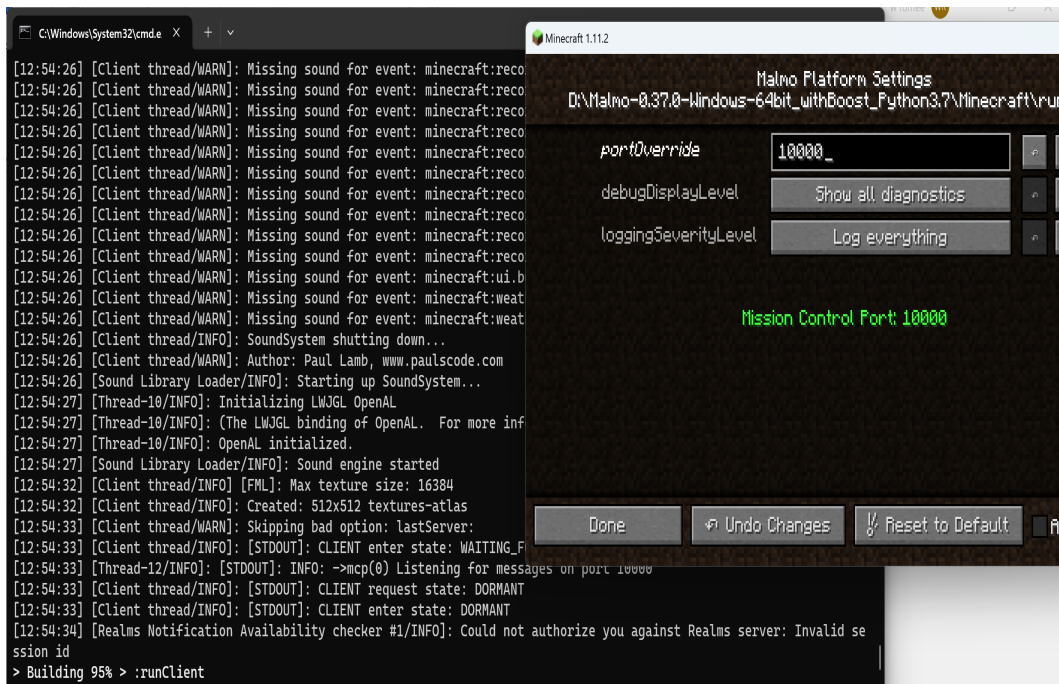


Figure 5.2: Port

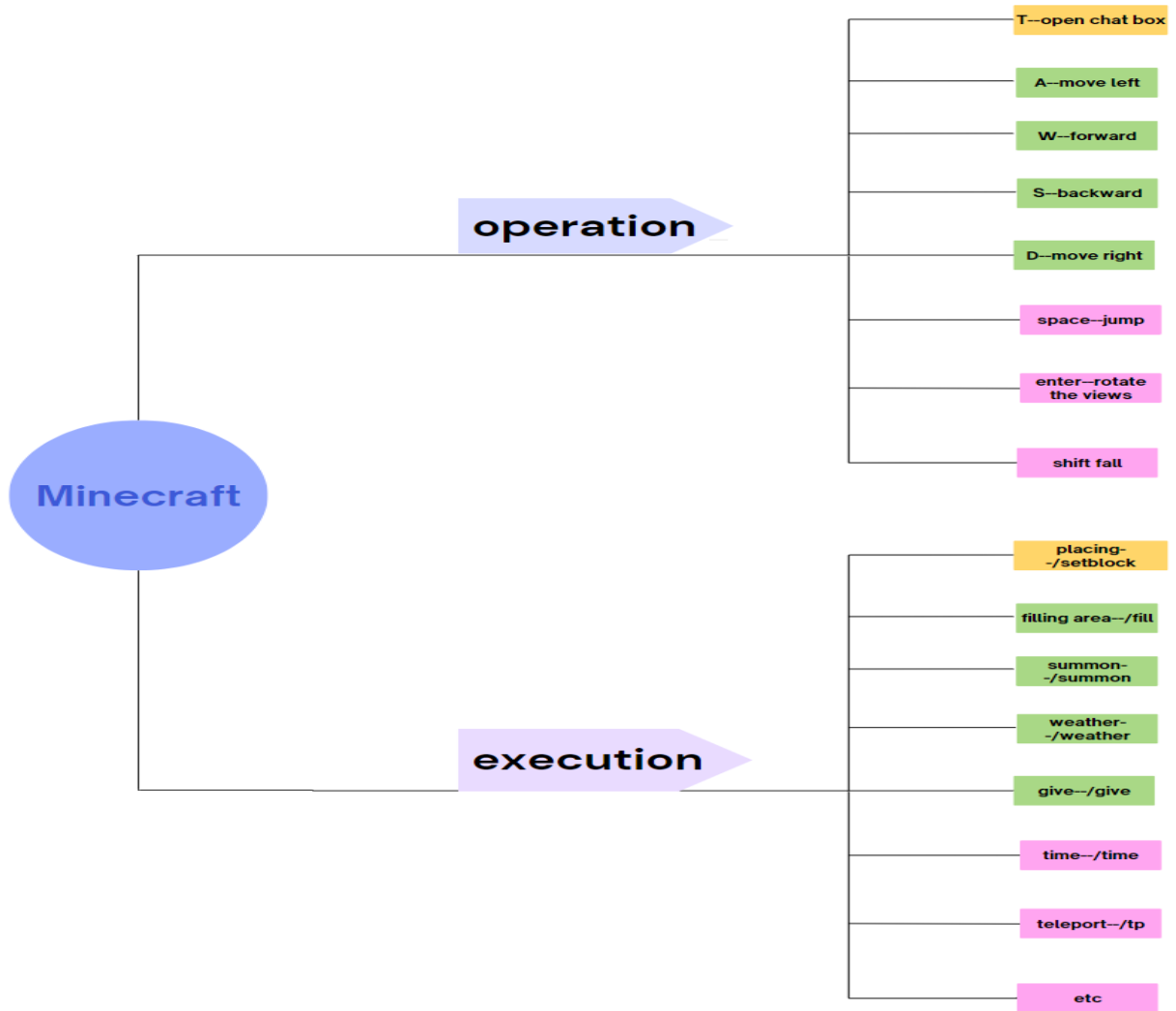


Figure 5.3: Minecraft operation and execution

vides the agent observations that include the environment, objects, and world state, etc. Additionally, the player can perform actions including creating the world, movement, and commands for inventory and building structures[18]. Specifically, Malmo allows the developer to create complex virtual environments, including various terrains, obstacles, and resources. This compatible development could assist the developer in exploring collaborative AI performances in a manifold and realistic environment. Furthermore, varied missions and challenges can be set up by Malmo, such as from simple navigation to complex collaboration. In addition, Malmo is flexible and extensible, so developers can customize the agent's dialogue, behavior, movements, response, planning, and strategy according to their needs because Malmo allows the agent to sense their surroundings by exploring, observing the world, and obtaining data about the environment. Generally, Malmo provides a flexible and compatible platform for developers to explore and experiment with different types of collaborative AI to adapt to diverse tasks and environments. Its flexibility, openness, and plasticity make it an ideal tool for doing collaborative AI research. In Malmo, AgentHost is an intermediary main class, essentially acting as a mediator enabling communication and interaction between the agent and the Minecraft. In this project, we used the version of Malmo 0.37, which requires the version of Python to be below 3.7 to import the library MalmoPython which contains Malmo classes and for now, Malmo 0.37 is the latest version. Specifically, Malmo will receive the Minecraft commands that are interpreted from natural language by LLM GPT-4, and then Malmo sends the commands to Minecraft and execute the commands in Minecraft via Malmo, so Malmo is the tool that connects to Minecraft and executes commands. Some Malmo classes are used in this project for designing and starting the mission, observing the current environment, sending commands to Minecraft, chatting in Minecraft, executing the commands in Minecraft, updating world state, etc. Figure 5.1 shows the classes from Malmo that are used in this project, which includes agenthost(sendCommand, startMission, and getWorldState), MissionRecordSpec, mission spec(XML), and WorldState(is_mission_running, number_of_observations_since_last_state, and observations).

5.2.1 AgentHost

AgentHost is one of the main classes that contains agent_host.sendCommand is used to send instructions to the Minecraft environment. These commands can control the actions of the agent, such as moves, jumps, attacks, etc., and can also be used to execute Minecraft commands, MalmoPython.AgentHost is used to control and manage the behavior of agents in the Malmo environment, sendCommand is for sending com-

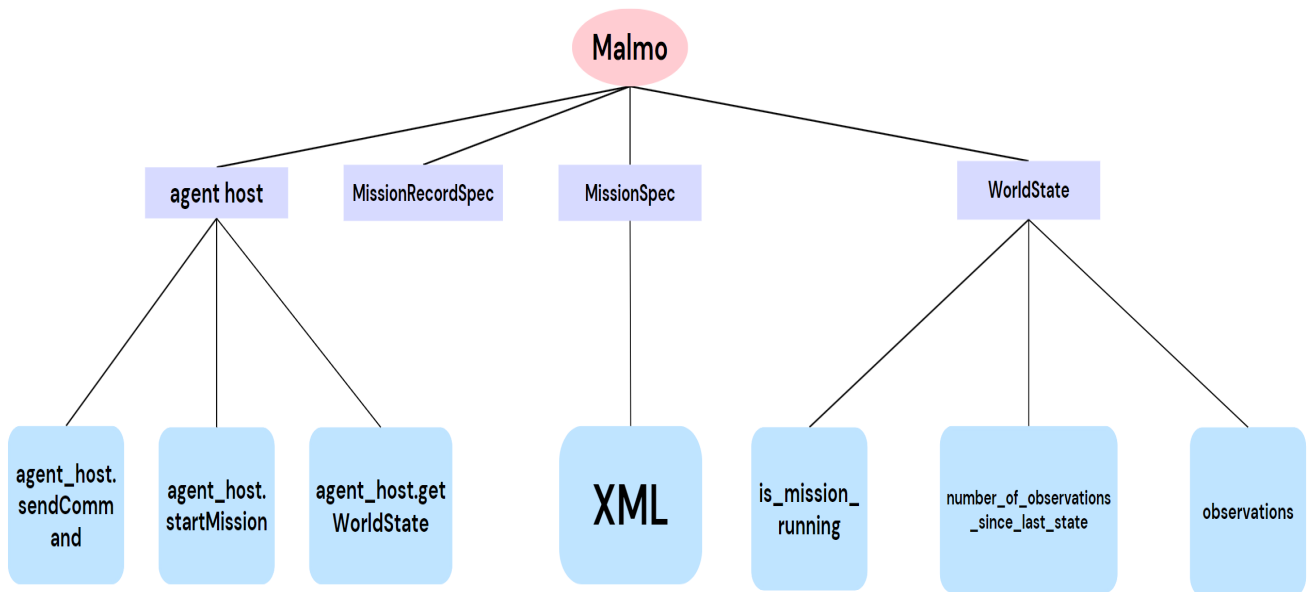


Figure 5.4: Malmo classes

mands and executing commands in Minecraft so that to interact and collaborate. `agent_host.startMission` is used to start a defined task, which sends the task to the Minecraft server and launches the task, `agent_host.getWorldState` is used to get the world state of the current task. The world state includes the observation data of the agent, whether the task has started, whether the task has ended.

5.2.2 MissionRecordSpec

`MalmoPython.MissionRecordSpec` is used to define how data is recorded during task execution. In this project, it is used to specify how to record the execution of Minecraft tasks. Specifically, `MissionRecordSpec` allows to save information about the status and events of task execution for analysis and review after the task is completed.

5.2.3 MissionSpec

`MalmoPython.MissionSpec` is used to define and configure Malmo tasks. The definition of a mission includes the way the world is generated, the task goal, the initial position and behavior of the agent, and so on. There is a XML file under `MissionSpec` class that is shown in Figure 5.2 that has the description for defining the basic framework and all the contents of the entire mission, providing the basic description of the mission, the server environment in which the mission runs, and the agent running in the mission, etc.

```
missionXML = '''<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Mission xmlns="http://ProjectMalmo.microsoft.com">
  <About>
    <Summary>Chat Interaction</Summary>
  </About>
  <ServerSection>
    <ServerHandlers>
      <FlatWorldGenerator generatorString="3;7,2;1;" />
      <ServerQuitWhenAnyAgentFinishes />
    </ServerHandlers>
  </ServerSection>
  <AgentSection mode="Creative">
    <Name>Builder</Name>
    <AgentStart>
      <Placement x="0.5" y="4" z="0.5" yaw="90" />
    </AgentStart>
    <AgentHandlers>
      <ChatCommands />
      <ObservationFromChat />
      <ObservationFromFullStats />
      <DiscreteMovementCommands />
      <MissionQuitCommands />
    </AgentHandlers>
  </AgentSection>
</Mission>'''
```

Figure 5.5: XML description

5.2.4 WorldState

WorldState is the class for representing the current state in Minecraft. In this project, there are three classes from WorldState that are used, `is_mission_running` specifies whether the mission is still running when the current world state was taken at that moment. Apart from that, `number_of_observations_since_last_state` contains the number of received observations since the current world state was taken from the last time. Additionally, `observations` contains the timestamped observations that are stored in the current world state.

5.3 NLP

Natural language processing(NLP) is for understanding and analyzing human languages while LLM is powerful tool that have been trained through massive amounts of data to understand and generate human language that is employed in NLP [19][20]. In this project, when the player inputs natural language in Minecraft, the natural language will be interpreted and translated to corresponding Minecraft commands and responds the related response in natural language if the player makes the dialogue in Minecraft and responds the context-related Minecraft commands if the user input natural language commands.

5.3.1 LLM and API

Considering the complexity of training, I used ChatGPT-4 as the large language model to parse and input natural language and generate the corresponding dialogue and Minecraft commands so that enables the AI agent to communicate with the player and execute the commands in this project. In [21], Zhiheng Xi et.al mentioned LLM can efficiently process user prompts and generate coherent, context-relevant responses and LLM is capable of processing multiple rounds of conversations, constantly building a conversation history and generating responses based on context, which is helpful for coherent dialogue and in-depth communication so that it can help the agent to understand the instructions. LLM has potential abilities on autonomy, reactivity, pro-activeness, and social ability. Hence it is suitable for being the core of an AI agent. In[21], Zhiheng Xi et. al introduced how LLM can help in understanding natural language to interact with the player, memory, planning, perceiving the environment, observation, navigation, and action.

Since in this project, I use chat GPT-4's API which requires the OPENAI API KEY from

Model	Accuracy %
IGLU Baseline	27.1
GPT-4	37.6
GPT-3.5	34.4
Llama2-70b-chat	23.9
Llama2-13b-chat	15.0
Vicuna-13b	6.3

Figure 5.6: Accuracy on IGLU NLP Evaluation[16]

the personal ChatGPT-4 account. Openai API Key is used for authorization requests to ensure that only authorized users can access OpenAI's API. This Key is the bridge that enables us connect to ChatGPT-4 to send request and get responses from GPT-4. I use GPT-4 because GPT-4 has the highest accuracy on processing natural language commands for building as figure tasks[16] compared with some other LLM as in Figure 5.1. Specifically, the API key allows us to access the powerful features of GPT-4 through a programmatic interface, enabling functionalities such as automated conversations and task instruction conversions. Note that it is crucial to ensure the security of the API key, as it is the sole credential for authorized access to the GPT-4 service. Anyone with this key can access and utilize the API resources associated with the account, so the OPENAI API KEY must be protected carefully to prevent from being leaked or misused. Furthermore, the API key information will be integrated in the API request headers. This ensures that each request is authenticated and properly processed, allowing commands and requests to be seamlessly transmitted to GPT-4 and accurate responses to be received.

5.3.2 Send Request To GPT

ChatGPT-4's corresponding model version is GPT-3.5-turbo, which has been clarified on the official ChatGPT website. GPT-3.5-turbo is an API-provided model that processes user requests and generates responses. The model version is mandatory to be consistent to the certain version of ChatGPT-4. Moreover, adherence to the corresponding model

version is often requires for compliance with licensing agreements and usage policies set forth by OpenAI. Ensuring that we use the correct version to help us stay compliant with these regulations, avoiding any potential legal or operational issues. In that case, we can keep the connection status persistently while connecting to GPT-4. The URL: <https://api.openai.com/v1/chat/completions> is the access point to the API and points to the OpenAI server. So in summery, openai API Key accesses GPT-3.5-turbo model through the URL and gets the generated content.

5.3.3 Prompt

The prompt is the input for sending to GPT-4, and instructs GPT-4 generates the appropriate response as the output, besides of generating responses, also includes answering questions, providing information these kind of interaction by understanding the context. In this project, appropriate responses include corresponding Minecraft commands. Specifically, prompt makes GPT-4 understand its ability to collaborate with malmo in Minecraft effectively. The content of the prompt clarified the responsibility of the role that GPT-4 will play. The content needs to be extremely clear and specified so that GPT-4 will understand the input natural language and convert it to Minecraft command form language and give responses. The prompt covers the specific commands and responding forms elaborately and is consistent with logical sentence structures. In the prompt, I specified that GPT-4 needs to generate commands that is compatible with the Minecraft 1.11.2 version. Prompt is the most significant part of the whole project, that all content has to be specified, and the sentence structure has to be very logic and coherent in prompt.

Minecraft Command Generation AI Instructions

```
You are an AI that can engage in general conversation and generate specific Minecraft 1.11.2 version compatible with /setblock and /give and /tp and /weather and /time and /summon and /fill commands based on user instructions and provides dynamic, real-time responses based on the commands executed and the current environment, please avoid generating <x> <y> <z>. For example, do not generate /setblock <x> <y> <z> stone. Please avoid generating commands with placeholder coordinates such as <x> <y> <z>. All coordinates' number have to be greater or equal to 0 instead of generating negative numbers (e.g., the number has to be
```

between 0-64). When the user's message is related to building structures (e.g., 'build a house', 'create a farm', 'build a pyramid', 'build a castle', 'build a fountain', 'build a forest', 'build a jungle', 'build a palace', 'build a igloo', etc), respond with the appropriate Minecraft command using Minecraft 1.11.2 syntax to create a simple, coherent structure using common blocks like planks, logs, glass, water, dirt, grass, leaves, stone, ice, snow, stairs, bricks, stone bricks, iron blocks, gold blocks, diamond blocks, and other materials, etc. Ensure the commands build a structurally sound and visually appealing structure with the right materials for each part. For example, a house should have walls made of planks or bricks, a roof made of stairs or slabs, windows made of glass, and a door. A pyramid should be made of sandstone or stone bricks. A fountain should include a stone or brick base and water blocks. Make sure to include commands for all necessary structural components.

Please avoid using placeholder coordinates like <x> <y> <z> and generate actual relative coordinates using ~ or absolute coordinates.

When the user's message is related to Minecraft (e.g., 'place a red wool block at 0 64 0'), respond with the appropriate Minecraft command using Minecraft 1.11.2 syntax and data values (e.g., '/setblock 0 64 0 wool 14'). Use numerical IDs for colors and adapt block names to Minecraft 1.11.2 standards (e.g., 'planks' for 'oak_planks').

For non-Minecraft related chat or ambiguous queries, provide a conversational response that encourages clarification or further interaction.

Please respond appropriately based on the context of the user's message, focusing on precise command generation for Minecraft tasks and engaging interaction for general conversation. Here is the user's instruction: '{prompt}'.

Please respond appropriately based on the context of the user's message, focusing on precise command generation for Minecraft tasks and engaging interaction for general conversation.

5.4 Interface

There are the interfaces, one with natural language involved, the other one without it involved. Algorithm 1 shows the pseudocode of the natural language processing inter-

face, and describes the flow of generating and executing Minecraft commands on the Malmo platform using GPT. Apart from that, figure 5.7 also shows the workflow for each main step and their corresponding malmo classes elaborately. After getting openai api key from GPT-4's account, the main flow focuses on define, start, and record the mission via `MissionSpec(XML)`, `startMission`, `MissionRecordSpec` respectively, `ObservationFromChat` from XML description obtains the input natural language instructions from Minecraft chat box, then extracts instructions via `getWorldState`, parse instructions via observations, send instructions to GPT-4 via URL: HTTP and get responses(generates corresponding Minecraft commands) after processing, send generated Minecraft commands to Minecraft and execute via `sendCommand()`. Specifically, users input commands through Minecraft's chat box, these natural language commands will be sent and translated into GPT-4's prompts, through multiple rounds of interaction with GPT-4, then generates corresponding commands for execution in Minecraft. Then the generated commands are validated and post-processed to ensure compatibility with Minecraft versions. Finally, these commands will be executed in Minecraft and continue to process and execute new commands until the end of the mission. The whole process, through interaction with the GPT-4 and command verification, ensures that the commands executed in Minecraft are valid and applicable.

Algorithm 2 presents the one only involves malmo and minecraft while figure 5.8 also shows the workflow for each main step and their corresponding malmo classes in details. Firstly, it initializes a Malmo agent (`AgentHost`) and defines an XML file for describing the framework of the task, setting the basic environment and initial conditions for the task. As the same Malmo functions in Figure 5.7, Malmo sends commands to Minecraft and execute also via `sendCommand`. Once the mission is launched, series of functions are defined for placing blocks, filling areas, summoning entities, etc in the Minecraft world. Users can perform these functions by entering commands from the command line. For example, users can use commands to place blocks, build pyramids, castles, houses, gardens, and other architectural structures. I set the commands including `place x y z block_type data_value`, `'fill x1 y1 z1 x2 y2 z2 block_type data_value'`, `'summon x y z entity_type'`, `'daytime time_value'`, `'tree x y z sapling_type'`, `'build pond x y z length width depth'`, `'build castle x y z length width height block_type data_value'`, `'build house x y z length width height block_type data_value'`, `'build garden x y z length width'`, `'build ladder x y z height'`, `'place torch x y z'`, `'teleport x y z'`, and `'quit'`. The program parses the instructions entered by the user and calls the corresponding function to perform the task. If the user enters quit, the program ends the task and outputs the message "Mission

Algorithm 1 GPT Interaction and Command Execution in Malmo

```

1: procedure SENDREQUESTTOGPT(prompt, openaikey, retries)
2:   Initialize url, headers, data with openaikey
3:   for attempt in 1 to retries do
4:     Send POST request
5:     if response status code is 200 then
6:       return message content
7:     else
8:       Wait
9:     end if
10:  end for
11:  return None
12: end procedure
13: procedure ITERATIVEPROMPTING(agent_host, prompt, openaikey, max_attempts)
14:  for attempt in 1 to max_attempts do
15:    response ← SENDREQUESTTOGPT(prompt, key, 3)
16:    if ISCOMMANDVALID(response) then
17:      return response
18:    else
19:      Update prompt
20:    end if
21:  end for
22:  return None
23: end procedure
24: procedure EXECUTEMINECRAFTCOMMAND(agent_host, commands)
25:  for command in commands do
26:    if command is valid then
27:      Execute command
28:      Wait 2 seconds
29:    end if
30:  end for
31: end procedure
32: procedure MAINLOOP(agent_host, openaikey)
33:  while mission is running do
34:    if new observations then
35:      for chat_msg in observations do
36:        prompt ← chat_msg
37:        response ← ITERATIVEPROMPTING(agent_host, prompt, openaikey, 5)
38:        if response then
39:          commands ← POSTPROCESSCOMMANDS(response)
40:          EXECUTEMINECRAFTCOMMAND(agent_host, commands)
41:        end if
42:      end for
43:    end if
44:  end while
45: end procedure

```

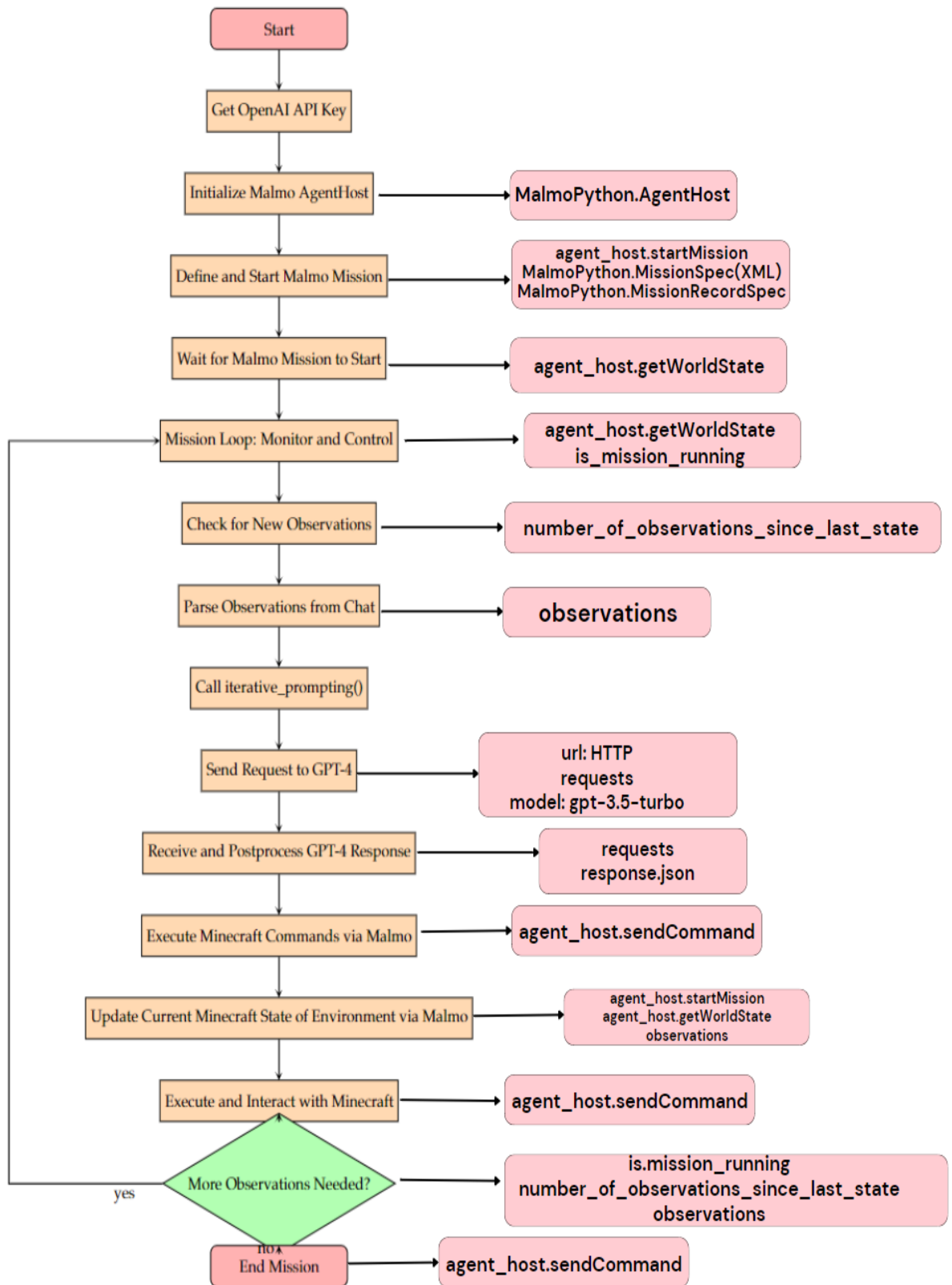


Figure 5.7: Used Malmo classes in the interface with natural language

ended", indicating that the task has ended.

Algorithm 2 Malmo Building Mission

```

Initialize AgentHost
2: Define mission XML
   Start mission with AgentHost
4: Wait for mission to start
   while mission not started do
6:   Wait 0.1 seconds
   end while
8: Mission started
   procedure PLACEBLOCKS(agent_host, x, y, z, block_type, data_value)
10:   Execute chat command to place blocks
      Wait 0.1 seconds
12: end procedure
   while True do
14:   Read user input
      Parse input into arguments
16:   if command is valid then
      Execute corresponding procedure
18:   else
      Print invalid command message
20:   if user input is "quit" then
      Break loop
22:
      End the mission
24:   Print "Mission ended."

```

Figure 5.9 and Figure 5.10 show the conversation in task 1 and task 2's interface.

Figure 5.11 demonstrates that simple task commands in Dutch was being understood and interpreted.

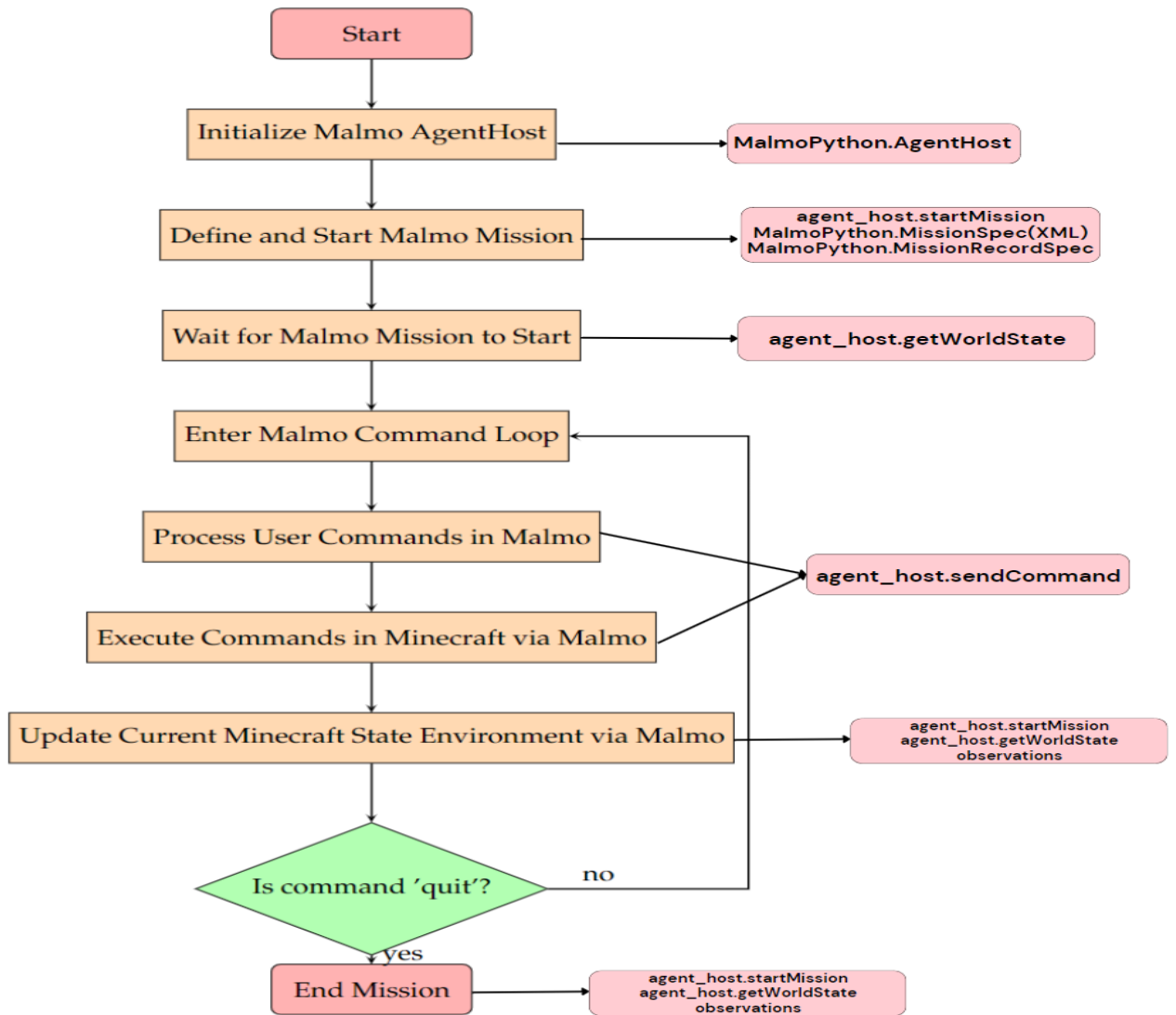


Figure 5.8: Used Malmo classes in the interface without natural language



Figure 5.9: Conversation 1 in Minecraft

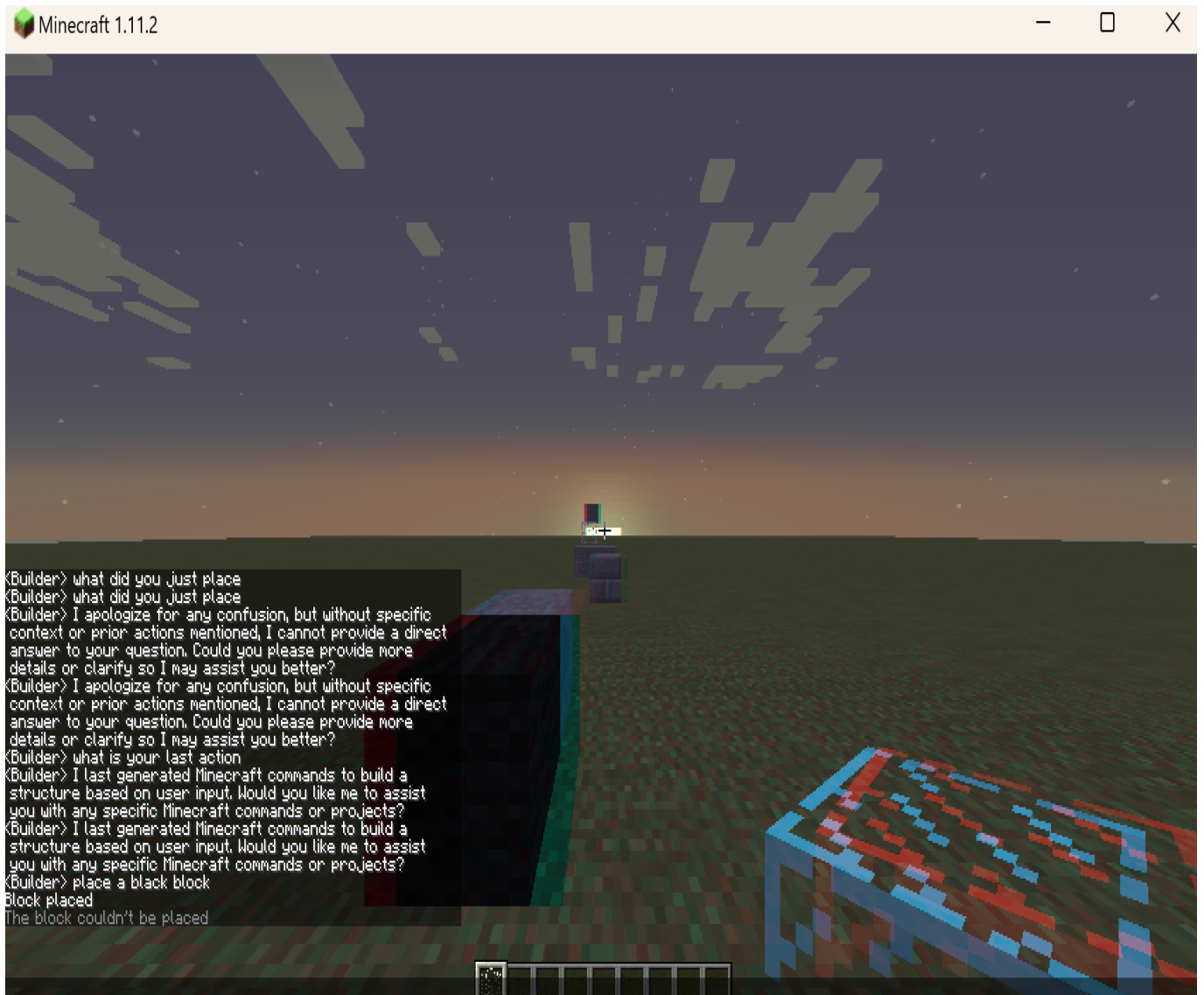


Figure 5.10: Conversation 2 in Minecraft

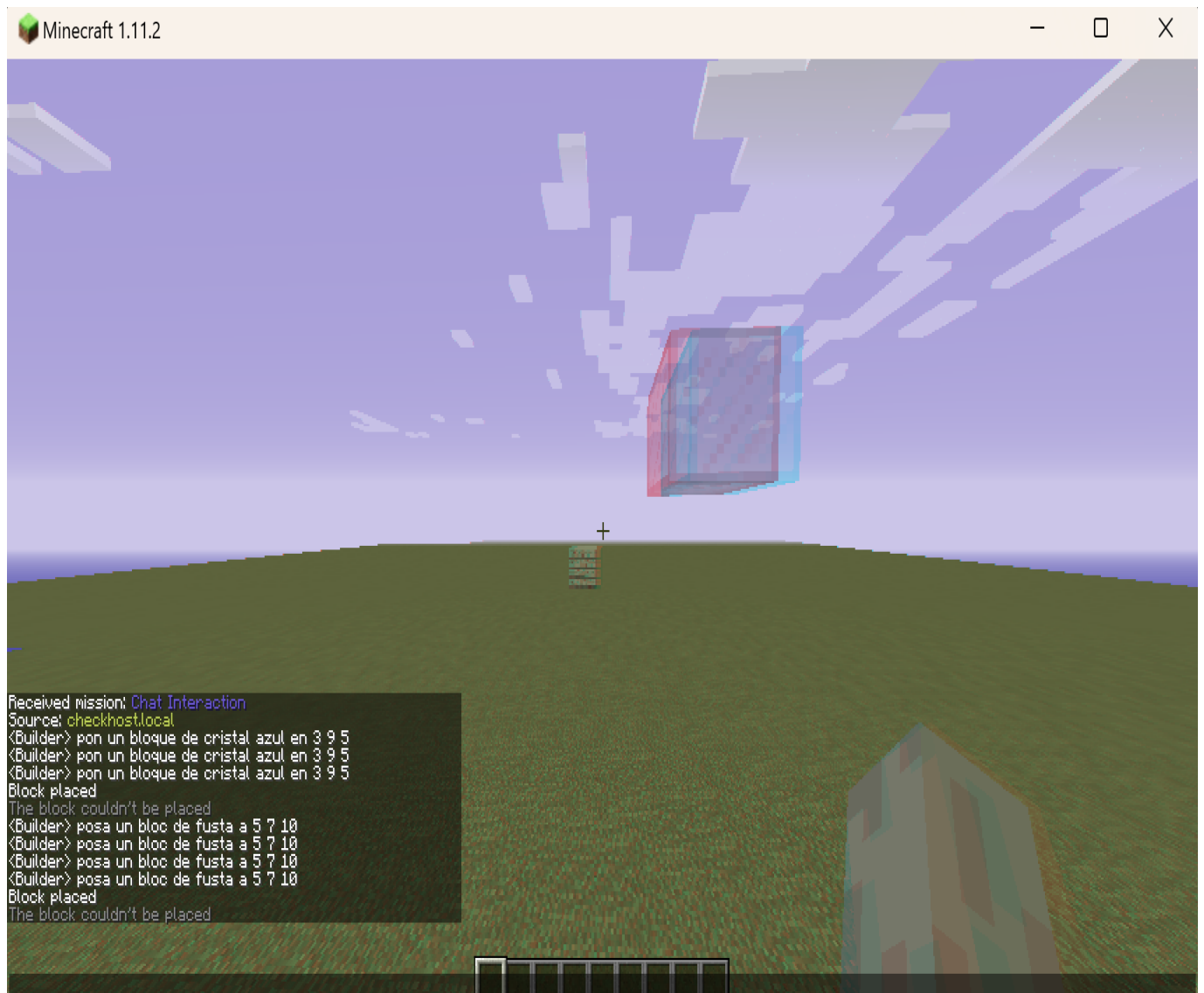


Figure 5.11: Simple task commands in Dutch

6. Experimental Methods

6.1 Participants

There are 20 participants and they will be divided into 2 groups equally, with 10 participants in each group.

6.2 Procedure

The experiments are offline in person, which means participants doing experiments in person on my laptop. Before the participants start experimenting, they will be required to read the guidelines first and they are allowed to read the guidelines anytime they want during the whole experiment. The guidelines contain elaborate content that might make the participants impatient to read thoroughly. Hence I will make a short verbal guideline before they start experimenting. Apart from that, they can ask me anything about this experiment during the gameplay but I will only answer their questions and will not give any suggestions. There are 2 main reasons for the procedure being set up this way.

1. The participants' experience should be considered because some participants might never played Minecraft before, some might played Minecraft a long time ago so they are not that familiar with operating Minecraft. Hence, the participants might not be able to get familiar with operating in Minecraft and need me to explain elaborately in person.
2. The project can not be exported because the project is based on Minecraft launched from cmd, which requires of tons configurations and installations, In addition, in task 3, participants need to enter specific commands in PyCharm to execute the program. The complexity and environmental dependencies of these operations make it impractical to package or export the project. In order to save participants time and effort, and to avoid distributing the program code to prevent code leakage or abuse, the experiment was arranged to be conducted in person. This ensures me to guide all configurations, installations, and operation steps are guided in person, and the participants can conduct the experiment directly in the field, avoiding complicated setup and possible technical problems. The participants were asked to finish the survey as soon as they finished the experiment to ensure the time duration would not affect the participants' experience and

memory.

For group 1, the task order during the experiment is task 1, task 2, and task 3, which means group 1 is required to do task 1 first, then task 2, and then task 3 the last. For group 2, the task order during the experiment is task 3, task 1, task 2, that group 2 is required to do task 3 first, then task 2, and then task 1 the last. I will specify to the participants about the task order they need to follow before the experiment after they read the guidelines, then repeat and emphasize before they start filling out the survey.

For the procedure of each task, the user is required to complete at least 5 different commands in each task generally. For the details of the procedure, firstly, task 1 and task 2 share the same platform that both input natural language commands in Minecraft chat box in the same Minecraft window. Figure 6.1 shows the workflow of task 1 that the user starts with inputting natural language commands to execute simple tasks in Minecraft, then Malmo parses commands after receiving them from Minecraft then sends parsed commands to GPT-4, and GPT-4 interprets commands and generates corresponding Minecraft commands, then Malmo sends commands to Minecraft and execute after receiving while keeps monitoring to update current states. Apart from that, Figure 6.2 illustrates the workflow on the same Minecraft interface after task 1 is finished but the user proceeds to input natural language commands to execute complex tasks. We encourage the participants to be as creative and innovative as possible in inputting the natural language commands in task 2 so that we can explore more details on natural language processing by accessing various forms of NLP. In that case, the participants are allowed to input any form of natural language.

Task 3's commands need to be input in PyCharm instead of Minecraft because task 3 does not involve GPT-4, Figure 6.3 presents task 3's workflow. IN specific, during the experiment, I ran the task 3 program on the PyCharm console. While the program is running, users can enter a specific Minecraft command from PyCharm's console once a time. The program performs corresponding operations according to these input commands. Specifically, when I launch the code, the PyCharm console displays a prompt where the user can enter Minecraft commands. These commands are passed to my code, which processes them according to predefined logic and performs the appropriate actions in the Minecraft environment. Each time the participant enters a certain Minecraft command and presses enter, the program processes and executes the command as required by task 3. This task aims to see the performances of Malmo cooperates in Minecraft.

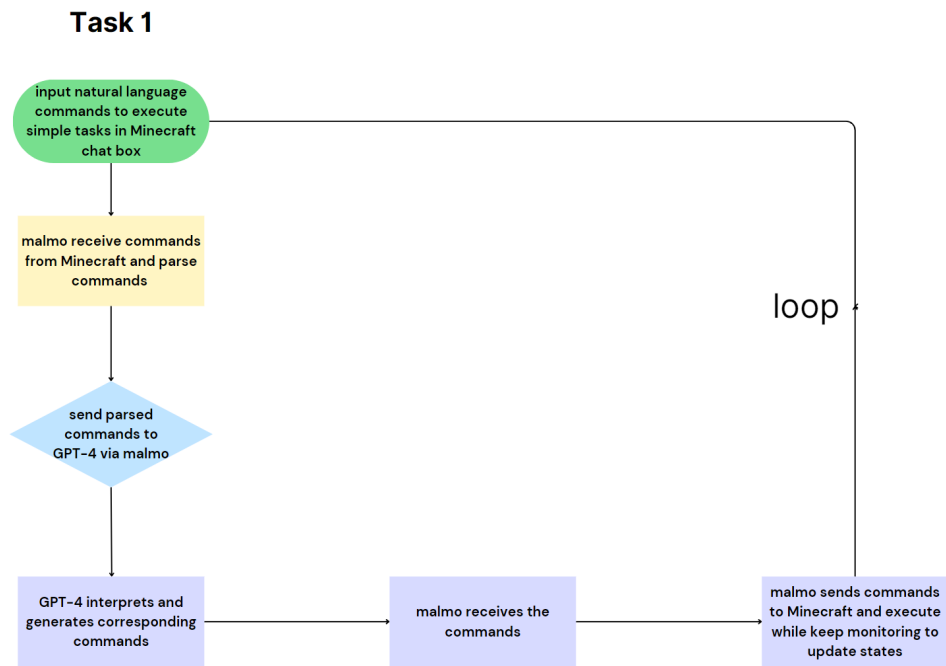


Figure 6.1: Task 1 workflow

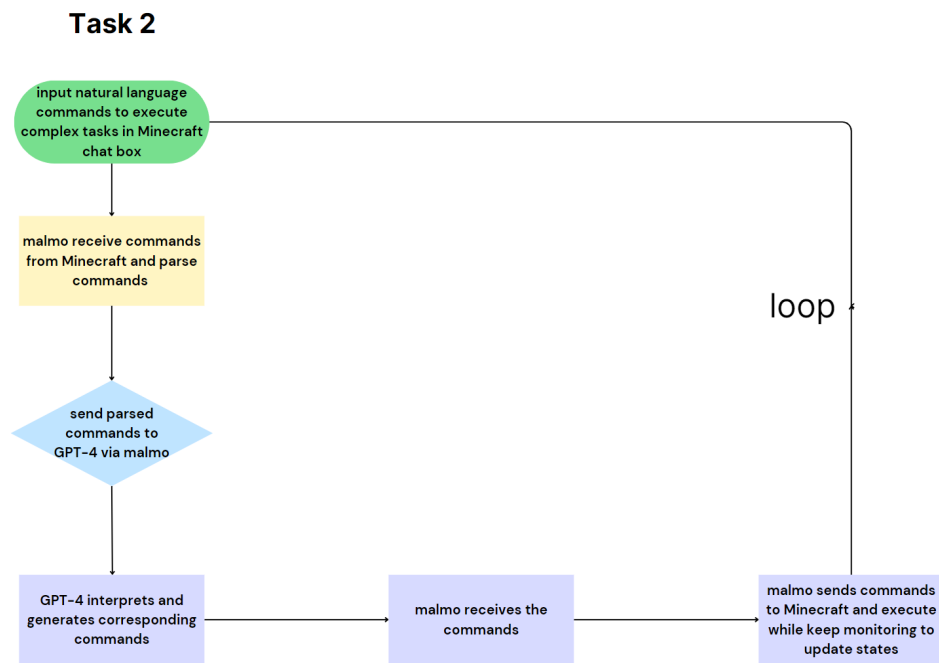


Figure 6.2: Task 2 workflow

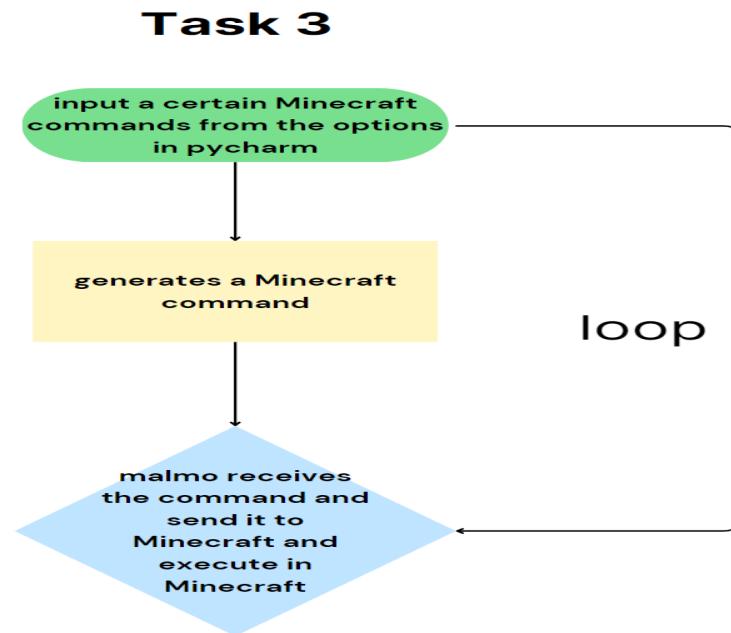


Figure 6.3: Task 3 workflow

6.3 Guidelines

The guidelines contain the content of 3 tasks and the instructions for operation in Minecraft with images.

The guidelines contain 2 sections: 1. The content of 3 tasks experiments. 2. The instruction on how to operate during the whole experiment in Minecraft. For experienced and inexperienced Minecraft participants, there is an amount of content in the guidelines especially when Minecraft is combined with a dialogue agent. The experienced Minecraft participants need to get familiar with the combination part that involves LLM. The inexperienced Minecraft participants need to get familiar with both Minecraft and the combination part. Each step is attached with figures to make the guidelines as elaborate as they can be. The guidelines also include lists for block types, colors, orientation, and tree types, etc for the participants to take as references.

6.4 Survey

I set up the questionnaire to collect data on Qualtrics. There are 11 questions in total and are divided into 4 blocks. The first block is for investigating participants' backgrounds in playing games and Minecraft due to the impacts of participants' games and Minecraft experience and contains 3 open questions. The second block contains 3 choice questions,

which compare among 3 tasks on the metric of ease of interaction, understood and execute, and effectiveness. The third block also contains 2 choice questions, and this block is for the game emotions during gameplay within 3 tasks, including happy and frustrated. The fourth block is for the conclusion to see the participants' willingness to play games with the dialogue agent mod in the future, which contains 1 choice question about game experience, and 2 open questions.

This survey focuses on getting to know the participants' video game and Minecraft backgrounds. Especially, the questions help to get to distinguish experienced and inexperienced Minecraft players. Their experience will affect the game experience. The survey will not record any personal information from the participants, it will be anonymous without any videos, images, eye tracking, and audio involved.

6.5 Evaluation

We aim to evaluate by computing the weighted ranking, average weighted ranking, mean, and std to see 3 tasks' performance. Apart from that, comparing 3 tasks' performance via pairwise and triplet to see whether the resulting data are significantly different. Friedman is used to evaluate 3 tasks' significance difference generally in triplets, then applied Wilcoxon to measure pairwise significance difference among 3 tasks so that the results will be more comprehensive.

6.5.1 Weighted average ranking

We use weighted for evaluating the performance of 3 tasks on ease of interaction, understood and execute, and effectiveness. This method combines information about the participants' rankings and takes into account the impact of different rankings. The weighted average ranking method takes the ranking of each task into account in the final score to reflect the actual impact of the ranking flexibly[22]. The task ranked 1 had the lowest weight because it was considered the best performer, while the task ranked 3 had the highest weight because it was the worst performer. This weighting can effectively reflect the actual ranking of the task in the minds of the participants. In addition, by applying the weighted values of the rankings (1, 2, 3) to the ranking number for each task, we were able to consider the evaluations of all participants comprehensively and balance the evaluations of different participants. The weight impact of each ranking is set according to the order in which it is ranked, so as to fairly reflect the opinions of all participants. Also, using a weighted average ranking allows us to witness a more intuitive comparison of

performance across tasks. Weighted ranking makes the ranking differences in tasks more obvious and easy to understand and explain by assisting us to identify the best task in terms of overall performance so that enables us to fully and fairly assess the performance of each task. This method not only takes into account the weight of each task on different rankings, but also reflects the participants' overall evaluation of the task, thus providing us with a more accurate and practical assessment result. Specifically on calculation, we first calculate the weight of each section, then calculate the overall average score to see their performance generally. We calculated the weight for ranking 3 tasks on the ease of interaction, understood and execute, and effectiveness to reveal the performance of different tasks in different dimensions, and multiple dimensions can be integrated to get the overall performance. From the weighted average ranking results, we can check the performances of 3 tasks' ranking generally. Then we use the overall average weighted score to see 3 tasks' performances on the ease of interaction, understood and executed, and effectiveness generally. [22]

$$\text{Weighted Rank} = \frac{1 \times \text{Number of Rank 1} + 2 \times \text{Number of Rank 2} + 3 \times \text{Number of Rank 3}}{\text{Total Number of Participants}} \quad [22] \quad (6.1)$$

6.5.2 Friedman

Friedman test is for comparing matched groups[23]. We use Friedman to illustrate the comparison of 3 tasks performance significance differences as triplets so that we can compare 3 tasks' performance together generally.

$$\chi_F^2 = \frac{12}{nk(k+1)} \left[\sum_{j=1}^k R_j^2 \right] - 3n(k+1) \quad [23] \quad (6.2)$$

where[23]:

- n is the number of subjects.
- k is the number of conditions.
- R_j is the sum of ranks for the j -th condition, which is calculated by ranking each task's performance across different metrics, then summing these ranks for each metric.

- The constant 12 is a scaling factor that adjusts the statistic to follow a chi-squared distribution under the null hypothesis.
- $\sum_{j=1}^k R_j^2$ is the sum of the squared ranks for each metric.
- $\frac{12}{nk(k+1)} \left[\sum_{j=1}^k R_j^2 \right]$ adjusts the squared sums of ranks to account for the number of tasks and metrics.
- $3n(k+1)$ is a correction factor that adjusts for the expected value of the test statistic under the null hypothesis.

6.5.3 Wilcoxon

Wilcoxon is for comparing two related samples, matched samples, or to conduct a paired difference test on repeated measurements from a single sample to determine whether there is a significant difference in their population mean ranks[24]. We use Wilcoxon to compare the pairwise significance differences among 3 tasks, which are task 1 vs task2, task 1 vs task3, and task 2 vs task3. Significant differences indicate real differences between tasks on some scoring dimensions. This can help us identify tasks that are performing well and those that need improvement.

$$W = \sum_{i=1}^n (\text{sign}(d_i) \cdot R_i) \quad [25][24] \quad (6.3)$$

where[25][24]:

- d_i is the difference between paired samples.
- R_i is the rank of the absolute value of the differences.
- $\text{sign}(d_i)$ is the sign of the difference d_i , which is +1 if d_i is positive, -1 if d_i is negative, and 0 if d_i is zero.
- n is the number of non-zero differences.
- The absolute differences $|d_i|$ are ranked, ignoring their signs, from the smallest to the largest. Tied ranks are assigned the average rank.

This test is used to determine whether there is a statistically significant difference between the pairwise tasks to validate the hypotheses.

6.5.4 Mean and STD

Some research already used mean and std to evaluate ChatGPT-4's performance[26]. We use mean for evaluating 3 tasks' performances on happy, frustrated, and experience generally, and std for analyzing the concentration and dispersion of the scores to see how stable the tasks' performances are from the participants' feedback. For presenting mean and std, we use a box plot to visualize the data.

The mean is calculated by summing up all the elements in the dataset of game feelings and experience scores and then dividing by the total number of elements.[27]

$$\text{Mean}(\mu) = \frac{1}{N} \sum_{i=1}^N x_i [27] \quad (6.4)$$

where[27]:

- μ represents the mean (average) value of the dataset(scoring).
- N is the total number of elements in the dataset(10 each group).
- x_i represents each individual element in the dataset, where i ranges from 1 to N .
- $\sum_{i=1}^N x_i$ is the summation of all elements in the dataset from $i = 1$ to $i = N$.

The standard deviation measures the amount of variation or dispersion in a dataset. It is calculated by taking the square root of the average of the squared differences between each element and the mean.[28]

$$\text{Standard Deviation}(\sigma) = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} [28] \quad (6.5)$$

where[28]:

- σ represents the standard deviation of the dataset.
- N is the total number of elements in the dataset.
- x_i represents each individual element in the dataset, where i ranges from 1 to N .
- μ is the mean (average) value of the dataset.
- $(x_i - \mu)$ represents the difference between each element and the mean.

- $(x_i - \mu)^2$ is the square of the difference between each element and the mean.
- $\sum_{i=1}^N (x_i - \mu)^2$ is the summation of the squared differences for all elements in the dataset from $i = 1$ to $i = N$.
- $\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$ is the average of these squared differences.
- $\sqrt{\cdot}$ denotes the square root of the average squared differences.

7. Results

The results are demonstrated in table 7.1 and table 7.2, table 7.1 illustrates the results for ranking among 3 tasks on ease of interaction, understood and executed, and effectiveness. Table 7.2 presents the results for scoring among 3 tasks on happy, frustrated, and experience.

Table 7.1: Results for ranking

task	rank 1st	rank 2nd	rank 3rd
ease task 1	9	10	1
ease task 2	5	9	6
ease task 3	6	1	13
understood task 1	8	8	4
understood task 2	4	6	10
understood task 3	8	6	6
effectiveness task 1	7	9	4
effectiveness task 2	5	7	8
effectiveness task 3	8	4	8

WEIGHTED

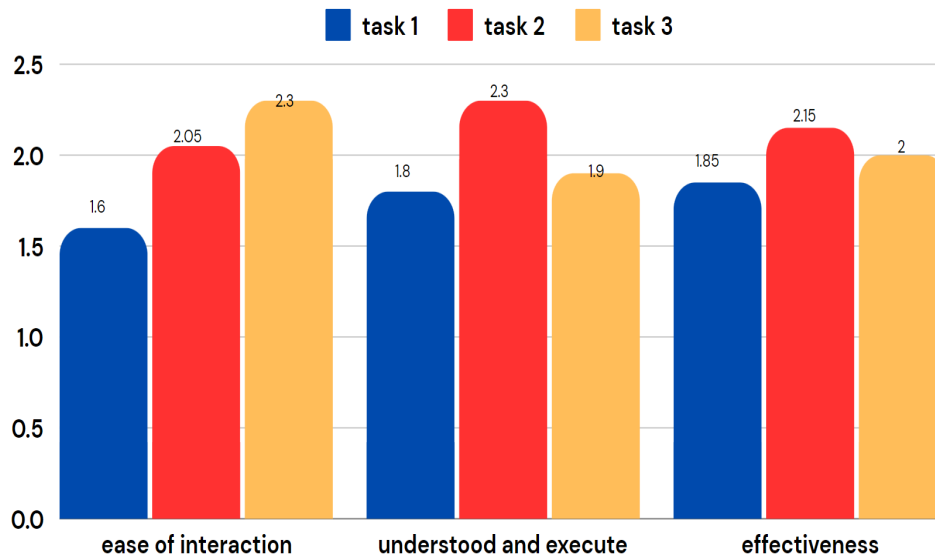


Figure 7.1: Tasks ranks for each metric

Table 7.2: Results for Scoring

Game Feelings	Scores																			
Frustrated_Task 1	3	1	5	1	3	4	9	3	6	9	3	4	0	2	7	4	5	1	0	0
Frustrated_Task 2	6	2	0	4	5	7	9	2	7	5	1	6	9	6	9	6	7	7	8	5
Frustrated_Task 3	1	2	1	3	1	1	3	5	8	10	0	5	8	4	5	10	7	3	0	10
Happy_Task 1	6	8	8	7	2	5	9	8	7	3	8	9	9	10	8	6	3	8	8	3
Happy_Task 2	3	9	9	7	2	5	9	8	6	2	8	10	1	10	4	7	3	4	2	1
Happy_Task 3	6	7	10	8	2	5	2	5	3	5	8	8	10	10	6	3	8	9	9	0
Experience_Task 1	6	9	8	8	2	4	9	5	7	0	7	4	9	5	7	6	7	10	7	1
Experience_Task 2	4	10	9	7	1	3	7	8	6	1	7	6	3	5	3	7	3	10	0	0
Experience_Task 3	7	6	10	9	4	6	2	5	3	2	6	8	10	3	4	1	6	5	7	0

From the weight we can see that task 1 performs the best in ease of interaction, understood and execute, and effectiveness with the smallest values. Task 2 performs mediocre on ease of interaction, the worst in understood and execute, and effectiveness. Task 3 performs the worst on ease of interaction, mediocre on effectiveness, same as good as task 1's performance on understanding and executing. In the overall average score, according to

OVERALL AVERAGE SCORE

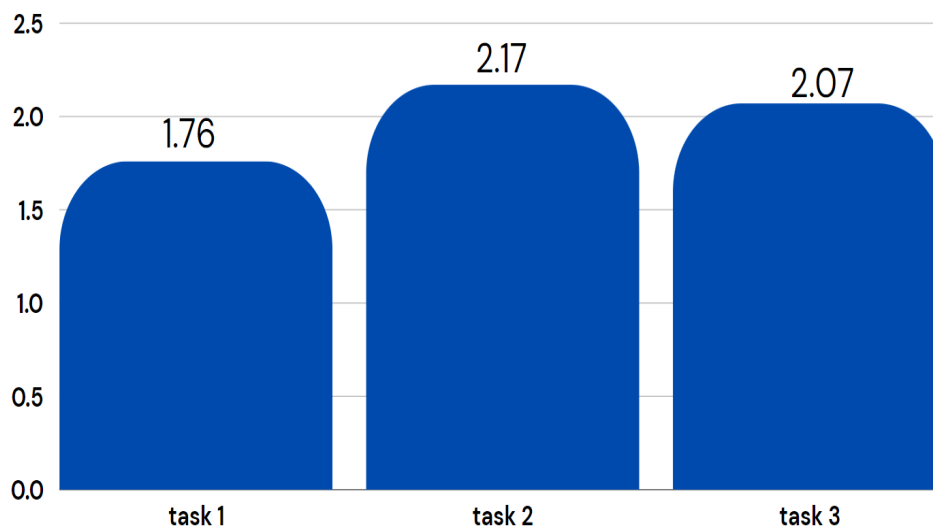


Figure 7.2: Overall average ranking score

the results, task 1 performs the best, and task 2 performs the worst. There is a significant difference between the 3 tasks on ease of interaction and frustration respectively, where the p values are 0.0003 and 0.027 respectively.

There are significant differences between task 1 and task 3(p-value=0.017), and a significant difference between task 2 and task 3(p-value=0.0003) on ease of interaction as Figure 7.4 shows. There is also a significant difference between task 1 and task 2 on happy (p-value=0.047), and a significant difference on frustrated between task 1 and task 2(p-value=0.02) that has shown in Figure 7.5.

According to the mean of 3 tasks on happy, frustrated, and experience, task 1 has the highest mean on happy and experience, and lowest mean on frustrated, which means the participants are satisfied with task 1 the most. Task 2 illustrates the lowest mean of happy and experience, and the highest mean of frustrated.

Happy: If we combine mean with std, we can see that in happy, task 1's std (2.36) is the lowest which shows most of the participants rated it close, and the rating agreement was high. In contrast, task 2 demonstrates the highest std (3.15), presents the scores of the participants vary greatly and the scores are scattered. Apart from that, task 3 has a relatively higher mean (6.2) indicating a higher overall score while its std is moderate (3.00), indicating that there is a certain consistency but a certain dispersion of the scores.

Frustrated: When we come to frustrated, both task 1 and task 2 have the low std indicating that there is a certain consistency, but task 1's mean is lower than task 2, so task 1

FRIEDMAN TEST P-VALUE

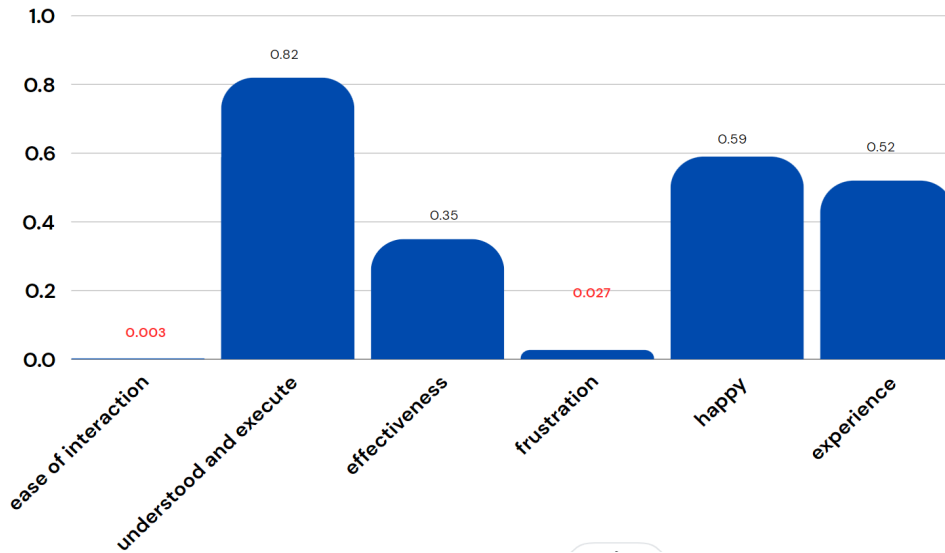


Figure 7.3: Friedman p-value

WILCOXON P-VALUE

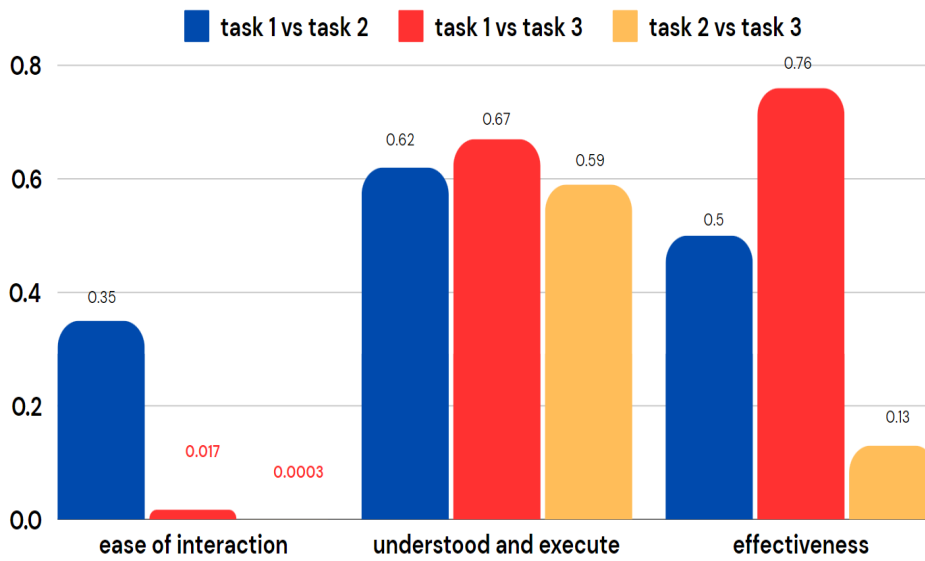


Figure 7.4: Wilcoxon p-value 1

WILCOXON P-VALUE

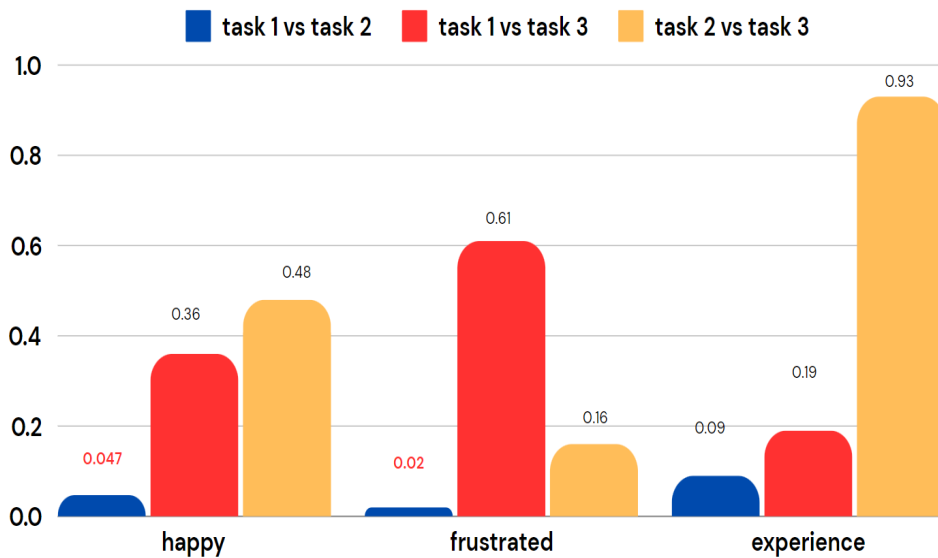


Figure 7.5: Wilcoxon p-value 2

performs better, and task 3's std (3.44) is the highest shows the scores of the participants vary greatly and the scores are scattered.

Experience: Additionally, task 1 still has the lowest std in experience, 2.58, which displays that most of the participants rated it close, and the rating agreement was high whereas task 2's std (3.10) is the highest indicating the scores of the participants vary greatly and the scores are scattered. Task 3's std is moderate (2.86), indicating that there is a certain consistency but a certain dispersion of the scores.

Here is the proportion of 20 participants' answers to "Do they prefer to interact with the dialogue agent in the future". 90% participants prefer yes and maybe to interact with the dialogue agent in the future, only 10% participants choose no.

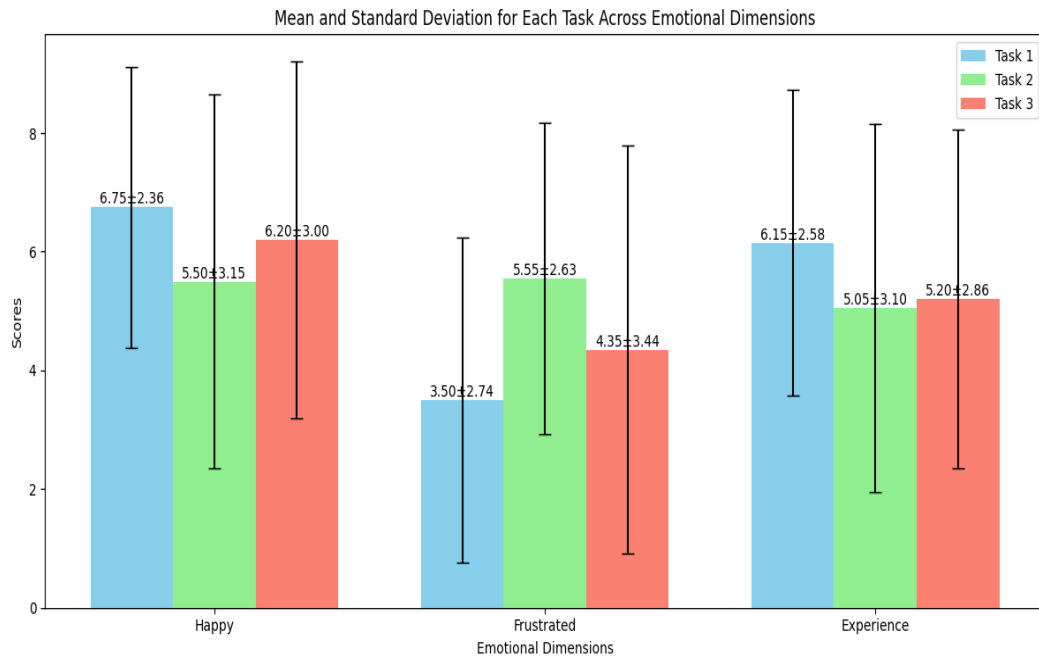


Figure 7.6: Mean and STD

PREFER TO INTERACT WITH THE DIALOGUE AGENT

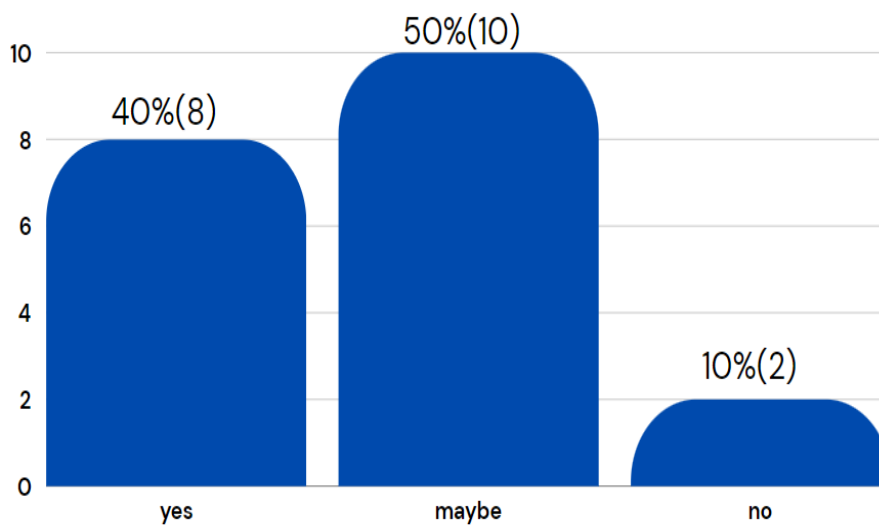


Figure 7.7: Participants who prefer to use the dialogue agent in Minecraft in the future

8. Discussion

We first compared the performance of 3 tasks as an entity, then we compared the performance between task 1 vs task 2, task 1 vs task 3, and task 2 vs task 3 in pairs. According to the results, task 1 performed the best on all metrics, including ease of interaction, understood and execute, effectiveness, happy, frustrated, and experience, Which partially validated our hypotheses.

8.1 H1: Natural Language Interaction can Improve the Efficiency and Effectiveness of Task Completion

According to the results of Figure 7.1 and Figure 7.4, task 1 performed the best on the ease of interaction, understood and execute, and effectiveness, in particular showing significant differences between task 1 and task 3 (p-value =0.017) and task 2 and task 3 (p-value =0.0003) on the ease of interaction, and that corresponds with the weighted results on task 1 and task 2 in ease of interaction, that task 1 and task 2 performed better than task 3, which means task 1 and task 2 as a natural language interaction, is easier to interact to complete tasks. In addition, although no significant differences were seen in understood and execute and effectiveness indicators, Task 1 still outperformed the other tasks on these indicators, especially the lowest weighted on effectiveness (Figure 7.1). These results support hypothesis 1 that natural language interaction does improve efficiency and effectiveness in Minecraft but only on simple task completion.

8.2 H2: Natural Language Interaction Enhances the Experience of Minecraft Players

Task 1 and Task 2 also showed significant differences in happy(p-value =0.047) and frustrated (P-value =0.02) as in Figure 7.5, indicating that natural language interaction can enhance player satisfaction and reduce frustration when completing simple tasks. In addition, Task 1 performed best across all metrics, with the highest scores on happy and experience and the lowest scores on frustrated (see Figure 7.6), which validates hypoth-

esis 2 that natural language interaction enhances the overall gaming experience and satisfaction of Minecraft players of task completion and making users more engaged and satisfied during gameplay but only on completing simple tasks.

8.3 H3: GPT-4 has a Relatively Accurate and Efficient Understanding and Parsing of Simple Tasks Natural Language Commands, But has Limitations When Processing with Complex Commands

Task 1 performed the best on all metrics, while task 2 performed relatively poorly, especially in ease of interaction, where the significant difference between task 1 and task 3 ($p = 0.017$) and task 2 and task 3 ($p = 0.0003$) show that GPT-4 performs well when processing with simple tasks' natural language commands. However, in the face of complex tasks' commands (such as the construction of complex buildings), the effect is significantly insufficient. This is reflected in the highest mean of frustration in task 2 (5.55) and the lowest mean of happy and experience (Figure 7.6), reflecting the limitations of GPT-4 in parsing and executing complex instructions, which validates hypothesis 3 that GPT-4 is relatively accurate and efficient when processing simple tasks' natural language instructions, but has significant limitations when processing complex tasks' instructions.

8.4 H4: Task Complexity Affects the Effect of Natural Language Interaction and GPT-4 Performance

The influence of task complexity is fully reflected in the experimental results. In particular, on ease of interaction, there are significant differences between task 1 and task 3 ($p = 0.017$) and task 2 and task 3 ($p = 0.0003$), indicating that task complexity does affect the effect of natural language interaction. In addition, the significant differences between task 1 and task 2 in happy (p -value $= 0.047$) and frustrated (P -value $= 0.002$) further validate the significant impact of task complexity on user experience and satisfaction. Since the complex instructions of task 2 lead to greater frustration and lower performance than task 1, hypothesis 4 is validated that task complexity does affect the effect of natural language interaction and GPT-4's performance.

Additionally, from Figure 5.9, Figure 5.10 we can see that GPT-4 can process and han-

dle relatively simple conversations in Minecraft. Figure 5.11 presents that GPT can understand and parse simple Dutch commands and commands that involve simple or few Dutch words in Minecraft.

8.5 Limitation

The versions of Minecraft and Malmo have some limitations since we can not test with the higher version of Malmo on the higher version of Minecraft for now. However, in the current stage, the versions of Minecraft and Malmo are the state-of-the-art we can use.

8.5.1 Unclear Coordinates

In task 1 and task 2, the whole building task is based on the player's position as the original principle spot hence once the player moves while the building task is proceeding, the block's position will follow the player's movement while building and misplaced. And the coordinates are usually unclear in all 3 tasks' interfaces even though I have set up the original spot in XML description, there are no clear grids in Minecraft interface to show clear coordinates. Hence making GPT understand and interpret the coordinates in Minecraft would be a challenge in the future, and the unclear coordinates impact route planning and the setup in specifying the prompt.

8.5.2 Minecraft 1.11.2 Version Command

In Minecraft 1.11.2, it has its own command names for certain versions. However, even though I specified to chat GPT-4 in the prompt clearly that the commands it generated have to be aligned with Minecraft 1.11.2 version command names. It still gives some command names that are not compatible with the Minecraft 1.11.2 version. For example, when the player inputs to build a castle, GPT-4 responds to commands that are compatible with the higher version of Minecraft. During the experiments, when the participants attempted to build some complex buildings, there was some reported information that said 'there is no such name called Minecraft:xxxxxxx', which means even though I specified compatibility in the prompt, and post processed the generated responses commands item names by replacing with the pre-fined item names in case GPT-4 still generates wrong Minecraft version item names. But GPT-4 still generated some Minecraft commands that are not compatible in Minecraft 1.11.2. I assume it still needs to be more

specified in the prompt in the future.

8.5.3 Respond Suggestions

During gameplay in task 1 and task 2, sometimes GPT-4 responds to suggestions instead of converting corresponding Minecraft commands to Malmo to execute in Minecraft. Even though I have already specified in the prompt clearly that as an assistant GPT-4 needs to generate corresponding and proper Minecraft commands instead of giving suggestions, but corresponding suggestions still happened every now and then.

8.5.4 Iterative Prompting

I used iterative prompting to gradually refine the generated Minecraft commands by interacting with the GPT-4 model in multiple iterations to perform better in the current environment. However, iterative prompting still did not adjust the generated Minecraft commands very decently.

8.5.5 Lag

This project connects to GPT-4's API, it lags very often during the gameplay in Minecraft. Sometimes the player has to input multiple times to get responses from GPT-4 due to the lag. To some extent, it affects the player's immersion and experience during gameplay. However, I assume the lag issue is due to ChatGPT itself. therefore hope the lag problem can be solved with GPT's update in the future.

9. Conclusion

Hypothesis 1 and hypothesis 2 are partially validated that the natural language interface can enhance the effectiveness and experience in Minecraft to the users but only in completing simple tasks. In addition, hypothesis 3 and hypothesis 4 have been validated that GPT-4 does have relative accuracy and effectiveness in simple tasks, but does not perform as well on complex tasks as it does on simple instructions (hypothesis 3), suggesting that task complexity does have a significant impact on the effect of natural language interactions and GPT-4's performance (hypothesis 4). These findings provide important guidance and insights for the future application of natural language processing techniques in Minecraft or other similar platforms. Furthermore, there are still some limitations and challenges that need to be considered and worked on in the future to improve GPT-4's performance in my project.

10. Future work

Due to limited time, the complex tasks did not perform well and still there are some limitations. However, in the future, if we have more time we will work on adjusting the prompt's sentence structures and content in the prompt via multiple attempts to improve the accuracy and effectiveness of enabling LLM such as GPT-4 to interpret complex commands based on the prompt. That might be also works on solving the first 4 limitations, because unclear coordinates, incompatible Minecraft item names might relate to the prompt. So besides well processing the complexity, GPT-4 generating accurate proper corresponding Minecraft commands, providing current state and previous actions' observation in Minecraft world by adjusting iterative prompting will also be considered as future work to enhance the interaction and collaboration in Minecraft.

Appendices

A. Appendix title

A. Appendix A

A.1 appendix

Guidelines:

Guidelines:

Welcome to this experiment. There are 3 tasks. You are required to give at least 5 different commands in each task.

Content:

Task1(simple tasks with GPT-4): placing blocks at any random X, Y, Z coordinates you want with any color and material (eg. dirt, stone, glass, grass, etc) blocks, filling area at any random X1, Y1, Z1, X2, Y2, Z2 coordinates with any color and material blocks or water or planks, summon creatures (eg. Chicken, wolf, pig, dog, duck, cat, skeleton, villager, cow, spider, sheep, dragon, cat), plant trees, set up the day time, teleport the user to any random X, Y, Z coordinates, setting up the weather, etc

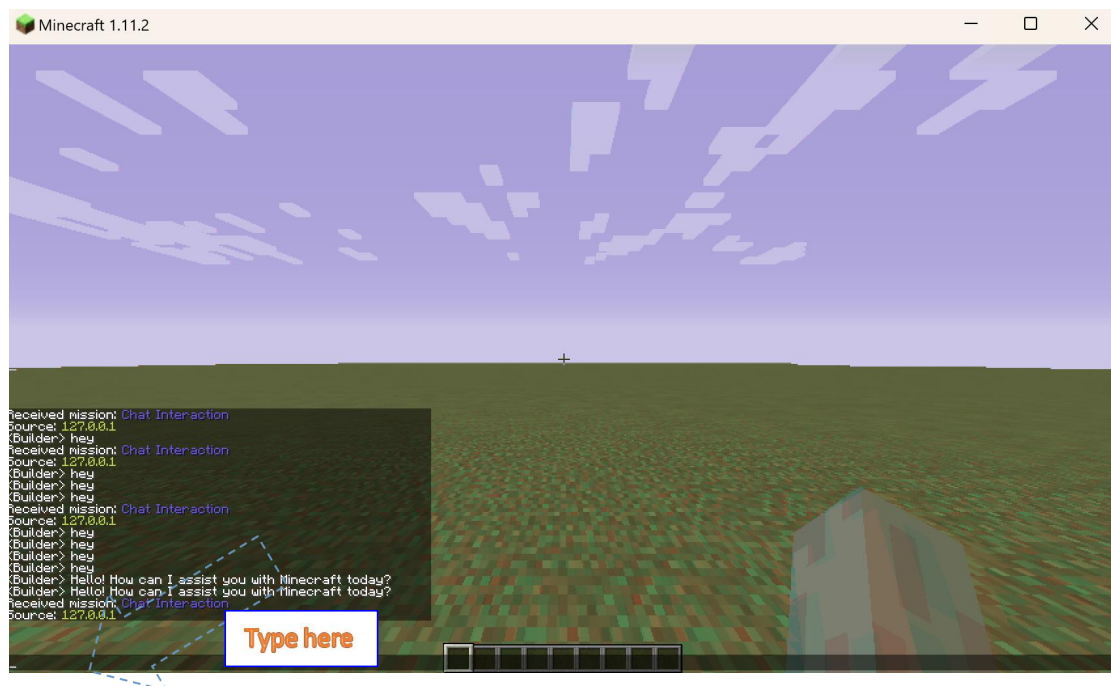


Figure: input interface

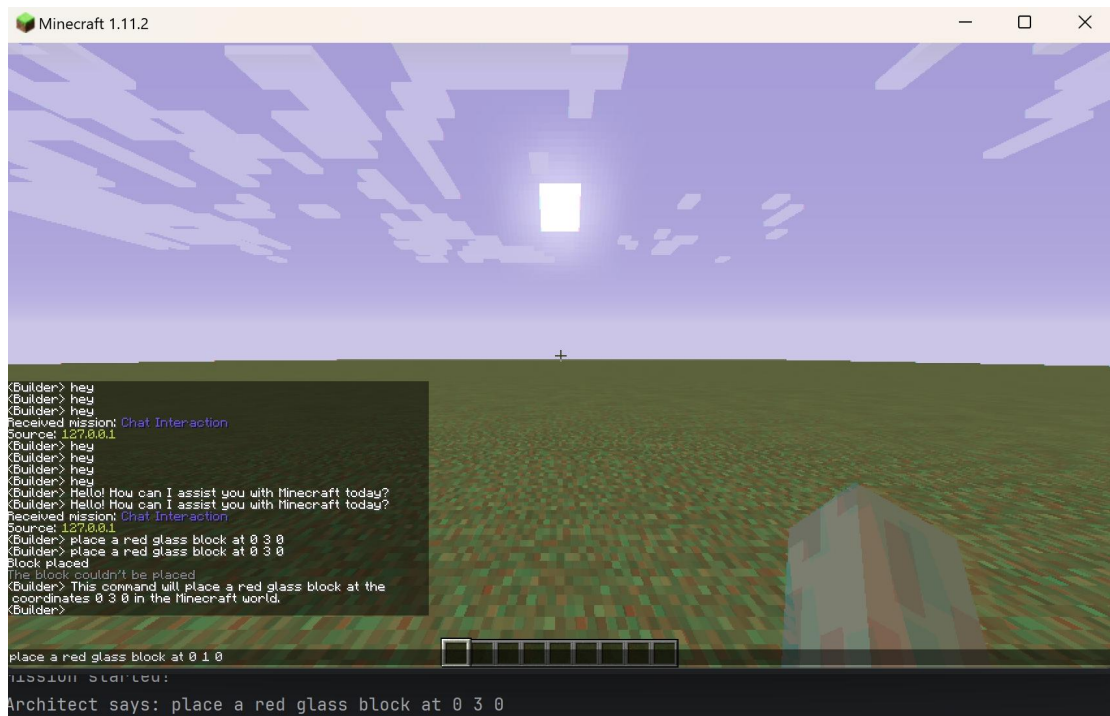


Figure: example

Task2 (complex tasks with GPT-4): creating/building something complex assembly (eg. a lake, pool, garden, stable, tower, house, castle, pyramid,etc). Please be as creative and innovative as you can when inputting the natural language commands in task2.

Task3(combined tasks without GPT-4): task 3 content combined simple and complex tasks

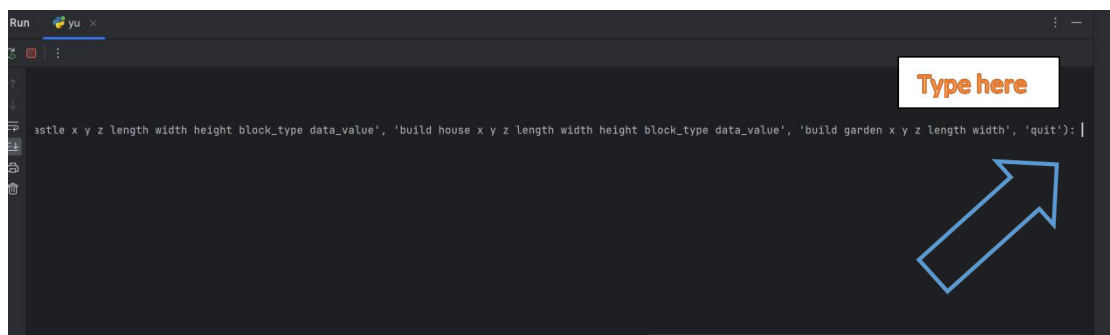


Figure: input interface

Instruction:

In general, for get better performance, all values you input must between 0-100. W: forward / A: left/ S: backward / D: right / Space: jump / Esc: cancel chat / Enter: move views /shift: fall/ left mouse: break blocks/ Up button: the previous commands in chat box

1. Task 1 & 2: Input the natural language commands/ instructions in minecraft chat box by pressing button 'T' (it might be lagging so you have to keep sending once a time until your command shows in pycharm controller) when you do task 1 & task 2. Please hold still while your build.

2. Task 3: Input the commands according to the commands options in pycharm interface once a time, when you finish typing, press 'Enter'. (e.g., 'place x y z block_type data_value', 'fill x1 y1 z1 x2 y2 z2 block_type data_value', 'summon x y z entity_type', 'daytime time_value', 'tree x y z sapling_type', 'build pond x y z length width depth', 'build castle x y z length width height block_type data_value', 'build house x y z length width height block_type data_value', 'build garden x y z length width', 'quit')

For blocks' data value: "white": 0, "orange": 1, "magenta": 2, "light_blue": 3, "yellow": 4, "lime": 5, "pink": 6, "gray": 7, "light_gray": 8, "cyan": 9, "purple": 10, "blue": 11, "brown": 12, "green": 13, "red": 14, "black": 15

Sapling type: 0-5

Oak Sapling: 0

Spruce Sapling: 1

Birch Sapling: 2

Jungle Sapling: 3

Acacia Sapling: 4

Dark Oak Sapling: 5

For stairs: the data_value is used to indicate both the direction and whether the stairs are inverted:

Direction

0 - Facing North (negative Z-axis)

1 - Facing South (positive Z-axis)

2 - Facing West (negative X-axis)

3 - Facing East (positive X-axis)

Inversion State

Adding 4 to the base values above indicates the stairs are inverted, for example:

4 - Inverted and facing North

5 - Inverted and facing South

6 - Inverted and facing West

7 - Inverted and facing East

For log: log Type and Orientation:

Basic Values - Default Upright Placement

0: Oak log

These values represent the logs placed in their default, upright position.

Extended Values for Orientation

Along the X-axis :

4: Oak log placed horizontally along the X-axis.

Along the Z-axis:

8: Oak log placed horizontally along the Z-axis.

All Bark Variant:

12: Oak log variant with bark on all sides.

Please note that I'll do the verbal guidelines before the experiments as well. You can check this guidelines anytime you want during the experiment or ask me.

Thanks for participating.

Q7. How often do you play games?

Q2. Are you experienced with Minecraft? If you are, how long have you been playing Minecraft?

Q3. How often do you usually play Minecraft?

Q4. Please rank the 3 tasks in terms of the ease of interaction (1 is the easiest, 3 is the highest)

	1	2	3
task 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Q5. Please rank the 3 tasks in terms of how well the agent understood and executed the instructions (1 is the best, 3 is the worst)

	1	2	3
task 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Q6. Please rank the 3 tasks in terms of the effectiveness of collaboration (1 is the most effective, 3 is the least effective)

	1	2	3
task 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Q7. I feel frustrated during game play in terms of these 3 tasks

	negative										positive
	0	1	2	3	4	5	6	7	8	9	10
task 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure A.1: Survey 1

Survey:

Q8. I feel happy during game play in terms of these 3 tasks

	negative										positive
	0	1	2	3	4	5	6	7	8	9	10
task 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Q9. How would you evaluate the game experience in terms of these 3 tasks during game play?

	negative										positive
	0	1	2	3	4	5	6	7	8	9	10
task 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
task 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Q10. Pros and cons about task 1, task 2, and task 3.

Q11. In the future, would you prefer to interact with the dialogue agent or not?

- Yes
- Maybe
- No

Figure A.2: Survey 2

Bibliography

- [1] R. I. Sumon, S. M. I. Uddin, S. Akter, M. A. I. Mozumder, M. O. Khan, and H.-C. Kim, "Natural language processing influence on digital socialization and linguistic interactions in the integration of the metaverse in regular social life," *Electronics*, vol. 13, no. 7, p. 1331, 2024.
- [2] G. Wang, Y. Xie, Y. Jiang, *et al.*, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.
- [3] W. Haijima, K. Nakakubo, M. Suzuki, and Y. Matsuo, "The embodied world model based on llm with visual information and prediction-oriented prompts," *arXiv preprint arXiv:2406.00765*, 2024.
- [4] A. Nijholt, J. Zwiers, and B. van Dijk, "Maps, agents and dialogue for exploring a virtual world," *Web Computing. International Institute of Informatics and Systemics (IIIS)*, 2004.
- [5] X. Zhu, Y. Chen, H. Tian, *et al.*, "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.
- [6] K. Krappala, L. Kemppinen, and E. Kemppinen, "Achievers, explorers, wanderers, and intellectuals: Educational interaction in a minecraft open-world action-adventure game," *Computers and Education Open*, vol. 6, p. 100 172, 2024.
- [7] Y. Xiong, H. Chen, M. Zhao, and B. An, "Hogriider: Champion agent of microsoft malmo collaborative ai challenge," in *Proceedings of the AAI conference on artificial intelligence*, vol. 32, 2018.
- [8] A. Narayan-Chen, P. Jayannavar, and J. Hockenmaier, "Collaborative dialogue in Minecraft," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5405–5415. [Online]. Available: <https://www.aclweb.org/anthology/P19-1537>.
- [9] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, "Sequence-to-sequence models can directly translate foreign speech," *arXiv preprint arXiv:1703.08581*, 2017.
- [10] Y. Feng, Y. Wang, and H. Li, "A sequence-to-sequence approach to dialogue state tracking," *arXiv preprint arXiv:2011.09553*, 2020.
- [11] M. Eric and C. D. Manning, "A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue," *arXiv preprint arXiv:1701.04024*, 2017.
- [12] P. Jayannavar, A. Narayan-Chen, and J. Hockenmaier, "Learning to execute instructions in a minecraft dialogue," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 2589–2602.
- [13] Z. Shi, Y. Feng, and A. Lipani, "Learning to execute actions or ask clarification questions," *arXiv preprint arXiv:2204.08373*, 2022.
- [14] A. Narayan-Chen, P. Jayannavar, and J. Hockenmaier, "Collaborative dialogue in minecraft," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5405–5415.

- [15] R. Islam, W. EDU, J. Bonn, *et al.*, "Lara–human-guided collaborative problem solver: Effective integration of learning, reasoning and communication,"
- [16] C. Madge and M. Poesio, "Large language models as minecraft agents," *arXiv preprint arXiv:2402.08392*, 2024.
- [17] C. Madge and M. Poesio, "A llm benchmark based on the minecraft builder dialog agent task," *arXiv preprint arXiv:2407.12734*, 2024.
- [18] S. Singh, "Minecraft as a platform for project-based learning in ai," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 504–13 505.
- [19] D. Gala and A. N. Makaryus, "The utility of language models in cardiology: A narrative review of the benefits and concerns of chatgpt-4," *International Journal of Environmental Research and Public Health*, vol. 20, no. 15, p. 6438, 2023.
- [20] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011.
- [21] Z. Xi, W. Chen, X. Guo, *et al.*, "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023.
- [22] A. F. Siegel, *Practical business statistics*. Academic Press, 2016.
- [23] D. G. Pereira, A. Afonso, and F. M. Medeiros, "Overview of friedman’s test and post-hoc analysis," *Communications in Statistics-Simulation and Computation*, vol. 44, no. 10, pp. 2636–2653, 2015.
- [24] B. Rosner, R. J. Glynn, and M.-L. T. Lee, "The wilcoxon signed rank test for paired comparisons of clustered data," *Biometrics*, vol. 62, no. 1, pp. 185–192, 2006.
- [25] S. Deng, J. Qin, *et al.*, "Graduation planning analysis using wilcoxon rank sum test," *Academic Journal of Humanities & Social Sciences*, vol. 5, no. 8, pp. 84–89, 2022.
- [26] J. K. M. Ali, M. A. A. Shamsan, T. A. Hezam, and A. A. Mohammed, "Impact of chatgpt on learning motivation: Teachers and students’ voices," *Journal of English Studies in Arabia Felix*, vol. 2, no. 1, pp. 41–49, 2023.
- [27] M. N. Martinez and M. J. Bartholomew, "What does it “mean”? a review of interpreting and calculating different types of means and standard deviations," *Pharmaceutics*, vol. 9, no. 2, p. 14, 2017.
- [28] G. W. Brown, "Standard deviation, standard error: Which ‘standard’ should we use?" *American journal of diseases of children*, vol. 136, no. 10, pp. 937–941, 1982.