Thesis proposal: Minecraft Social Agents

PIC2 - Master in Computer Science and Engineering
Instituto Superior Técnico, Universidade de Lisboa

Rafael Alexandre Milheiro Lourenço — 92543*
rafael.milheiro.lourenco@tecnico.ulisboa.pt

Advisor: Rui Prada

Abstract Within this thesis proposal, I conducted an analysis of existing research on socially aware interactions in gaming. Some of the approaches presented are broad in scope, with only a select few focusing on a specific game. I have elected to develop a tailored approach for creating social agents in Minecraft, as this game offers unique possibilities for social interactions that cannot be found elsewhere. Attempting to create a generalized solution that caters to Minecraft as well would diminish the potential of social agents within the game.

Given the above considerations, I propose the development of a Minecraft Plugin that can be effortlessly installed on any Bukkit server without the need for Mods or intricate configurations, allows the easy deployment of agents, and allows the agents to act autonomously within a society. The proposed Plugin would rely on an existing Plugin as a dependency, which permits the creation of player-looking Non-Player Characters (NPCs) and grants complete control over their actions. ChatGPT would be used to choose agents' actions due to its massive training data that includes written social interactions.

Keywords — Social, Interaction, Game, Minecraft, Society, Artificial Intelligence, Large Language Models, ChatGPT

*I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa (https://nape.tecnico.ulisboa.pt/en/apoio-ao-estudante/documentos-importantes/regulamentos-da-universidade-de-lisboa/).

Contents

1 Introduction

The incorporation of social behavior in non-player characters (NPCs) is an area of significant interest and importance in gaming. While games aim to provide immersive experiences through storytelling, and realistic reactions from in-game elements like mud footprints or a character's physical exhaustion, believable NPCs are also crucial for players to feel like active participants in the virtual world.

The inclusion of social agents offers the possibility of creating new forms of entertainment solely based on them. For instance, a game where the objective is to fix the relationships of two NPCs solely through dialogue.

Social agents can also add a new dimension to open-world games, making them more exciting and immersive. Besides exploring and discovering new territories, players can interact with and learn from virtual civilizations that respond differently to various situations. Understanding how these civilizations respond to threats, such as invasions of zombies, offers a new level of entertainment and intrigue.

Despite the significance and potential of social agents in gaming, creating NPCs that exhibit social behavior similar to humans remains a challenge. This is due to the complex and nuanced nature of social behavior, which requires advanced AI and machine learning techniques to implement. Video game consoles and personal computers can only do so much. Running large Neural Networks on these devices that mimic human behavior is not a reasonable expectation.

The proposed solution involves utilizing ChatGPT to simulate coherent and context-aware social behavior. Through accessing Open AI's API, the approach aims to leverage ChatGPT's natural language processing capabilities to determine the appropriate in-game actions for specific situations. This process involves querying ChatGPT for a human-like response, which is then converted into relevant in-game actions.

1.1 Work Objectives

- Develop a simple and efficient method for deploying agents in Minecraft,
- Define and implement a diverse range of actions that agents can perform,
- Establish a mechanism for converting an agent's context into a textual format that can be processed by ChatGPT,
- Determine the most appropriate message format for ChatGPT, with a focus on ease of extraction of actions indicated by the response,
- Translate ChatGPT's responses into executable agent actions,
- Deploy agents into a Minecraft world that can interact with each other as a society by creating relations, engaging in dialog, scheduling and participating in events, distributing tasks, etc.

1.2 Envisaged Contributions

This thesis aims to make the following contributions:

- Demonstrate the potential of applying ChatGPT to enhance the social experience of games, thereby increasing immersion for players,
- Enhance the sense of vitality in Minecraft's gameplay by incorporating dynamic and contextually appropriate social interactions through the utilization of ChatGPT,
- Streamline the deployment of agents within the Minecraft environment, thereby providing a more accessible and efficient framework for agent integration.

By achieving these contributions, this research endeavors to advance the field of game development by leveraging ChatGPT's capabilities to create more engaging and immersive social interactions in games, particularly within the Minecraft ecosystem. Additionally, the proposed framework aims to facilitate the deployment and integration of intelligent agents, fostering a more dynamic and interactive gameplay experience.

2 Background

2.1 What is Minecraft?

Minecraft [1], developed by Mojang Studios [2] in 2009, is a renowned sandbox video game that has achieved unparalleled success, boasting a staggering 238 million copies sold worldwide [3]. With its open-ended nature, Minecraft offers players the freedom to create their own objectives, whether it involves exploration, mining, constructing cities, battling monsters, or engaging with non-player character (NPC) villagers, among countless other possibilities.

Beyond its widespread popularity as a source of entertainment, Minecraft has transcended its gaming realm and found utility in various research endeavors, particularly in the field of Artificial Intelligence. The game's expansive virtual world and versatile mechanics have made it a valuable tool for scientific investigations and experimentation, facilitating advancements in AI and related domains.

Figure 1: Minecraft cover art

2.2 Minecraft Editions

Minecraft was originally developed in Java as a PC game by Mojang Studios [2], and the Java Edition continues to receive updates every year, making it the most played edition of Minecraft. Due to the game's popularity and growth, other editions [4] were created to spread the game to other platforms, such as Xbox, with better performance due to being written in C++.

For this proposal, I will focus on three editions of Minecraft, the Java Edition, Bedrock Edition, and Education Edition since they are the most relevant.

2.2.1 Java Edition

Minecraft Java Edition [5] was the first edition of the game to be developed. It was initially released in 2009 as a single-player game and has since then received regular updates from Mojang Studios. Java Edition was not launched in any game sale platforms, as it had its own website where it could be purchased. When multiplayer was introduced, the only way to play with friends was to download the server code and run it on a machine that allows port-forwarding so that other player connections may exchange packets with it.

Since Minecraft servers are run by the players and not the company, players have full control over them. Very early on, libraries were created that allowed everyone to easily create a Plugin [6] that modifies how Minecraft servers behave and could manipulate the packets being sent to Minecraft clients. Bukkit [7] was one of the Plugin libraries created and its forks are still used.

Plugins are limited in their capabilities. If a player wants their server to have completely new blocks or monsters, there's no packet manipulation that allows Minecraft clients to recognize the custom blocks and entities.

There are ways to apply custom texture pack tricks that make players think they're interacting with a brand new entity or seeing a brand new block, but in reality, they're just seeing a retextured existing entity or block.

Minecraft Mods [8] solve the limitations of Plugins, at the expense of forcing players to have to install said mods locally so that the Minecraft client may interpret the new entities, blocks, etc. The Forge Library [9] was one of the first libraries that allowed Minecraft mod development. It allows full control over Minecraft, from changing its very code (for example, to optimize rendering), to adding new blocks, machines, light rendering, sound effects, etc.

Due to its open-source-like nature and active plugin and mod development community, Java Edition has become the most popular version of Minecraft, with a large number of mods and plugins available for download. However, Java Edition's reliance on Java as a programming language means that it is less optimized for performance than other editions of the game. Additionally, the requirement for players to install mods locally means that mods cannot be used in multiplayer unless all players have the same mods installed.

2.2.2 Bedrock Edition

The Bedrock Edition of Minecraft [10] was developed with the aim of expanding the game's reach to more platforms such as Xbox, Mobile, PlayStation, Nintendo Switch, and other devices. It also sought to enhance the game's performance, given that Java is generally slower for game development, while C++ provides a more efficient alternative.

In line with the creation of Bukkit for Java Edition, the Minecraft community developed LiteLoaderBDS [11] as an unofficial plugin loader that offers basic API support for the Bedrock Dedicated Server, with an extensive API. Mods in Bedrock are referred to as "Add-ons" but are less flexible than those in Java Edition, as Mojang's mod compatibility implementation limits creators to "only modify features that Mojang explicitly allows and exposes" [13].

It is worth noting that most of the other Minecraft Editions are based on the Bedrock Edition.

2.2.3 Education Edition

"Minecraft Education is based on Bedrock Edition [10] and contains features that make Minecraft more accessible and effective in a classroom setting" [14]. For example, there's a block called "Element Constructor" which can be used to create elements by adjusting the number of protons, neutrons, and electrons.

While it is not possible to add plugins or mods, this edition offers new features that allow for some customization.

3 Related Work

3.1 Deploying Agents in Minecraft

Before delving into Social Behaviour, it is imperative to determine the optimal method for deploying manipulable agents within Minecraft. Given the game's popularity and adaptability, there are numerous approaches available, each with its unique advantages and drawbacks. Some methods may be more accessible to programmers, while others may be easier for casual players to install and experiment with the programmed agents.

3.1.1 Minecraft Education Edition [14]

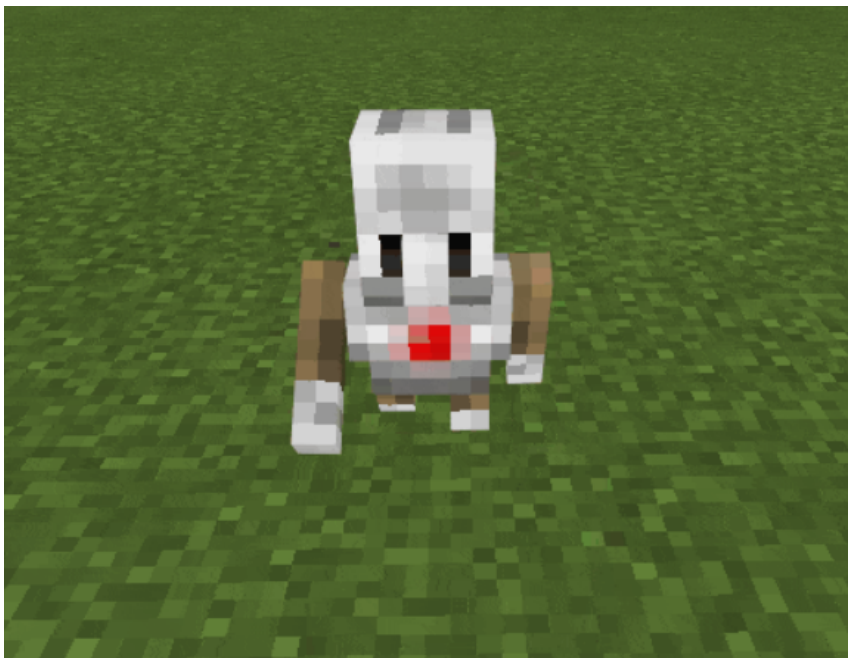

Figure 2: Education Edition's Agent

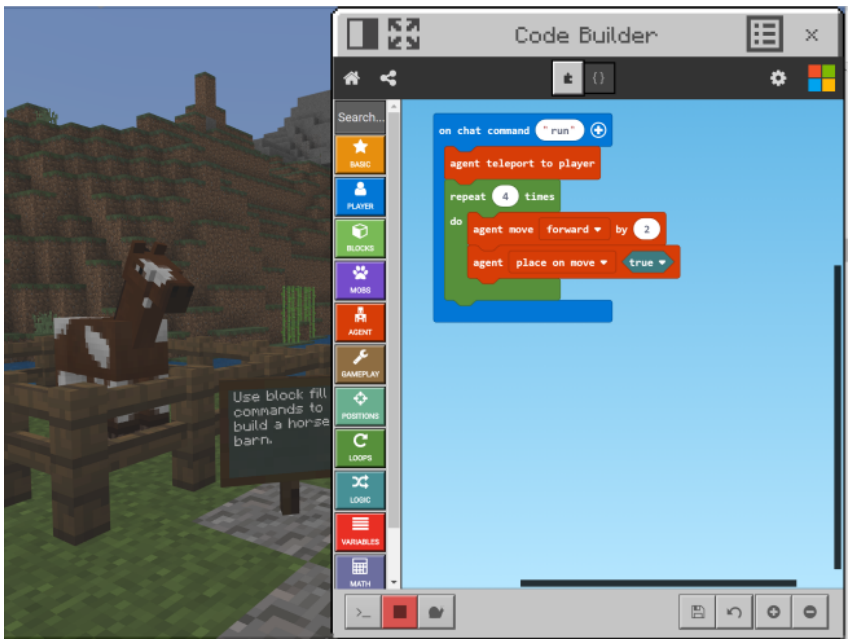

Figure 3: Education Edition's Visual Programming Interface, taken from Minecraft Education Edition Website

Minecraft Education Edition is a version of Minecraft intended to be used for Education. It also provides a visual code editor for programming a small entity in the game, known as an "agent". However, this solution is not suitable for the proposed project, as it is not universally accessible since Education Edition is played by so few people, and the programming interface is limited in its capabilities for handling complex tasks.

3.1.2 Modded Minecraft

Mods are limitless in the extensions that they can do to Minecraft. One of the things they can do is enable the addition of any type of customizable entity, including entities that resemble players. There are two mods that allow this: MineColonies [16] and MinecraftComesAlive [17].

When it comes to Minecraft mods, they exert their modifications on both the client and server sides of the game. Leveraging mods provides a convenient avenue for deploying agents, as it grants full control over the game's code. However, it is worth noting that the mod's installation process and its potential re-installation (due to updates) itself can present challenges and inconveniences for players seeking to incorporate these mods.

MineColonies [16] , an old mod that remains compatible with newer versions of Minecraft, endeavors to simulate a civilization within the Minecraft universe. Upon spawning into the world, players can initiate their own civilization, assuming the role of its leader, and even encounter other civilizations that, regrettably, lack a leader and remain stagnant in their development unless assisted by the player as an ally.

Figure 4: An example of a MineColonies civilization guided by a player

The NPCs within MineColonies lack true autonomy. To engage in actions, they must be assigned specific jobs by their leader, which is the player. Reproduction among NPCs occurs randomly and is contingent upon the availability of sufficient housing within the civilization to accommodate newborns. Some jobs grant NPCs a degree of autonomy, such as Soldiers who either shadow the leader and provide protection if so commanded or patrol the civilization, vigilantly seeking out threats. Conversely, certain jobs exhibit much less autonomy. For instance, builders (figure 5) will solely construct structures designated by the leader and exclusively at the designated locations.

Figure 5: MineColonies NPC with the Builder job

Social interactions within the mod are limited in scope. NPC-to-NPC interactions are predominantly confined to random reproduction events, contingent upon suitable housing availability. Player-to-NPC interactions possess a slightly broader range, as players issue commands to NPCs, although the NPCs' behavior remains unaltered regardless of whether their leader's disposition is benevolent or malevolent.

MinecraftComesAlive [17] (MCA) introduces a distinct approach to NPCs, wherein social interactions play a prominent role. NPCs within MCA possess the ability to engage in actions such as marriage, reproduction, and forming families, with players themselves being able to initiate familial connections. Importantly, MCA NPCs exhibit a degree of autonomy even in the absence of player interaction. They engage in activities such as farming, construction, procreation, and interactions with other NPCs. The NPCs perceive the player as one of their own, rather than as a leader, although players have the potential to assume leadership roles within the mod.

In comparison to MineColonies, MCA NPCs feature substantially richer social interactions, as they possess individual opinions that influence their choices. For instance, in order to pursue marriage with an NPC, players must first engage in flirtatious behavior, such as telling pre-defined jokes (which may or may not resonate with the NPC's sense of humor) or presenting gifts (which may be met with varying degrees of appreciation based on the NPC's personal preferences).

Figure 6: NPCs in MinecraftComesAlive

Of all the NPCs presented in this thesis thus far, those within MinecraftComesAlive most closely mirror the appearance of the Minecraft player, thereby creating a more immersive experience.

One notable advantage of this mod is its status as an open-source project. Consequently, forking from the mod represents a viable prospect for this thesis.

3.1.3 Minecraft Plugins

Through the utilization of plugins, it becomes possible to inject code into the server environment, thereby simulating player-like entities by dispatching packets to real players that emulate the actions of fictional players. Consequently, the Minecraft client of each player renders these fabricated entities and their corresponding actions, operating under the assumption that the packets originate from genuine player interactions.

The Citizens plugin [18] stands as a widely recognized tool for incorporating player-like NPCs into Minecraft, enabling some degree of behavior customization through packet manipulation. However, it is important to note that this plugin does not encompass social behavior. Primarily employed by server owners, it finds application in populating server lobbies with interactive characters that engage players in conversation when they approach and execute predetermined commands upon player interaction. Moreover, the plugin's open-source nature permits developers to expand its Application Programming Interface (API), thereby facilitating the creation of more engaging and intricate interactions.

Figure 7: Player configuring an NPC using the Citizens plugin

The Sentinels plugin [19] builds upon the open-source features of Citizens, providing enhanced functionality that empowers NPCs with configurable combat abilities, such as patrolling. It is worth noting that the Sentinels plugin does not extend the Citizens plugin with social behavior and interactions, as it focuses solely on combat. Similar to Citizens, the Sentinels plugin is also open-source.

The EntityUtils plugin [20], currently under development by a work colleague at EVNT Games [21], provides a streamlined solution for deploying player-like NPCs in Minecraft. As part of my work related to Minecraft Plugin and Mod development at EVNT Games, I have had the opportunity to contribute to this plugin and have gained direct access to it. This means that in the event of encountering any bugs or issues, I can address them personally, without being solely dependent on the plugin's creators. The plugin simplifies the process of incorporating player-looking NPCs into our projects, offering greater convenience and ease of use compared to alternative solutions like the Citizens plugin (for example, Citizens does not offer a way to make an NPC pretend to break a block). If needed, we can optimize and fine-tune the plugin's functionality to align with our specific requirements, resulting in a more tailored and efficient experience for our Minecraft Plugin development endeavors.

Server Networks extensively leverage and harness the power of plugins to achieve a myriad of objectives, ranging from the creation of captivating mini-games to the implementation of cheat detection mechanisms and the integration of novel features that seem like they're pushing the boundaries of plugin development possibility. Server Networks represent a highly popular avenue for players to engage in collaborative Minecraft gameplay, as they simply need to launch Minecraft, locate the IP address of a desired network, and effortlessly join without the need for additional installations. This convenience accounts for the considerable disparity in player numbers between those who frequent Modded Minecraft servers and those who opt for Plugin-based Minecraft servers.

One preeminent example of a Minecraft Server Network is Hypixel [22], which presently boasts more than 38,000 consecutive online players. Hypixel's success ultimately led to the formation of Hypixel Studios [23] as a dedicated entity following the server network's meteoric rise. The network's accomplishments have been formally recognized through the acquisition of four Guinness World Records, as documented in "It's Official - Hypixel holds 4 Guinness World Records". Their resounding triumph stems from their ingenious utilization of plugins. It is crucial to reiterate that plugins cannot introduce new content to the game, as their operations are confined to the server realm rather than the Minecraft clients. However, Hypixel's exceptional creativity within these limitations has cemented its reputation. An illustrative instance pertinent to this thesis is their renowned mini-game titled "Hide and Seek". In this captivating mini-game, players are divided into two teams: hiders and seekers. The game unfolds within a map teeming with NPCs that emulate authentic player actions. Hiders must assimilate among the NPCs and complete assigned tasks to secure victory, while seekers endeavor to locate and eliminate all hiders before they accomplish their objectives.

Figure 8: Screenshot from Hypixel's "Hide and Seek" mini-game featuring a seeker player who has designated one NPC as "suspicious" and another as "checked"

3.1.4 Minecraft Bedrock Edition

The Minecraft Bedrock Edition is a version of Minecraft written in C++ instead of Java. It offers a straightforward method for creating Non-Player Characters (NPCs) that resemble players [15]. However, their actions are severely restricted, as they are only capable of executing commands and displaying dialogue.

Figure 9: Built-in NPCs in Minecraft Bedrock Edition

Alternatively, it is possible to create NPCs in a manner akin to the Java Edition, utilizing Bedrock's equivalent counterparts of Mods and Plugins. Nonetheless, it is worth noting that such an undertaking proves considerably more challenging in the Bedrock Edition due to the more limited extent of plugin and mod creation capabilities, as well as the comparatively smaller community support it enjoys in contrast to its Java counterpart.

3.1.5 Packet Sending and Receiving

An alternative approach involves the development of a program capable of sending and receiving packets to a Minecraft server while impersonating a player. However, this option presents a significantly higher level of complexity. The program must be able to format packets in a manner that Minecraft can interpret, as well as interpret packets received from the server. Additionally, this approach requires a second non-Minecraft program to be running, adding an extra layer of intricacy that may discourage some users from exploring it.

The primary advantage of this approach lies in the flexibility it offers regarding the choice of programming languages and frameworks. Developers can select languages independent of Minecraft's supported languages (Java, C++). For example, if developers wish to deploy agents whose actions are determined by machine learning techniques, they may opt for Python and leverage its extensive AI resources. In this scenario, the program would receive packets from a Minecraft server, interpret them, choose actions, and send corresponding packets back to the server.

MineFlayer [24] is a JavaScript API that simplifies the deployment of NPCs into the game by utilizing packets, as mentioned earlier. Its powerful API enables developers to easily issue complex commands to NPCs, such as building structures or attacking entities. However, one notable drawback is its usage complexity. To employ the agents, players need to install the necessary JavaScript dependencies and run the deployment script, making it less accessible for non-developers.

SocialCraft , [44] the predecessor to this thesis, pursued the same objective of deploying socially intelligent agents into Minecraft. It utilized MineFlayer to execute agent actions. In SocialCraft, agents possessed autonomy and would programmatically determine their actions based on their assigned job. Additionally, they exhibited social characteristics, including traits like "agreeableness" and a "friendship" rating, influencing their social interactions, such as engaging in dialog, as depicted in Figure 10.

Figure 10: Interaction between two SocialCraft agents through in-game chat.

The implementation of SocialCraft involved the utilization of Docker, where each agent operated within its dedicated Docker instance in its own process, sending and receiving packets. However, this introduced additional complexity to the deployment process, as Docker is not particularly user-friendly, even for developers like myself who encountered challenges while working with it.

Project Malmo [45], developed by Microsoft, represents a sophisticated AI experimentation platform built on top of Minecraft, designed to facilitate fundamental research in artificial intelligence. Agents within Malmo run on their own process and receive input in the form of screen pixels from the game, along with optional additional data such as block positions and types in their vicinity. The pixel information is provided to the agents through a custom Mod developed by Microsoft, which establishes communication with the running agent process. "Minecraft needs to create windows and render to them with OpenGL, so the machine you do this from must have a desktop environment." [26], therefore, machines used for this purpose must have a desktop environment, rendering Malmo unsuitable for servers running on headless machines due to the fact that wanting to run agent processes on the machine wouldn't work, which limits its usability for the majority of server owners who rely on dedicated rented machines.

Project Malmo enables the creation of objectives for agents to accomplish, and during training mode, agents leverage deep learning methods to improve their performance in achieving these objectives.

3.1.6 Discussion

The Education Edition's NPCs, with their robot-like appearance, fail to replicate the immersive experience of interacting with actual players. Additionally, their behavior programming is confined to a drag-and-drop interface, which offers limited flexibility compared to traditional programming languages. Project Malmo, while capable of teaching agents to accomplish objectives using deep learning techniques, is not suitable for my thesis due to the inherent challenge of teaching social behavior through such methods. Although Malmo could be valuable for executing agent actions like mining in caves, its limitation of being unable to deploy agents on servers without a desktop environment renders it impractical for widespread usage.

While SocialCraft demonstrated some success, its intricate setup process hinders its accessibility to the majority of users.

The Modded Minecraft approach initially appears to be a logical choice, given its expansive possibilities and existing groundwork to build upon. However, the additional burden placed on players to install mods presents a drawback, leading me to favor the plugin alternative.

MinecraftComesAlive offers the potential as a suitable starting point for a forked project. However, as I will explain later in this paper, my implementation of social interactions will not be algorithmic. Thus, a significant restructuring of the action selection process would be necessary, diminishing the value of reusing the project. As such, the primary benefit would be limited to agent deployment, making it less compelling.

Although EntityUtils is still a work in progress, it presents a convenient option for me as I have editing access to its code, allowing for seamless implementation of agent actions, such as mining. This aligns with the goals of my thesis while simultaneously contributing to EVNT Games. Consequently, EntityUtils surpasses the alternative of forking from the Citizens plugin. Both plugins facilitate easy agent deployment, but Citizens' agent actions offer limited utility, as they lack essential player activities like wood chopping and farming.

While plugins impose more limitations compared to mods, these constraints do not impede the objectives of my thesis. Plugins enable the creation of NPCs and the manipulation of their actions and behaviors, without needing to introduce new NPC types, items, or effects. Therefore, the potential offered by plugins is more than sufficient, reducing the advantage of mods. Furthermore, the installation process of plugins does not burden players directly but rather places the responsibility on server owners.

In conclusion, my chosen approach involves developing a single Minecraft Java Plugin, named MineSocieties. The limitations imposed by plugins do not hinder the project's objectives, and they are the easiest Minecraft modifications to install. Additionally, I possess several years of experience in plugin development, eliminating the need to invest time in learning a new system. I will leverage EntityUtils as a dependency to streamline agent deployment and agent action execution, aligning with the requirements of my thesis and benefiting both my research and EVNT Games.

3.2 Social Behaviour

The aim of this thesis is to develop Minecraft agents that exhibit social behavior comparable to that of humans. The concept of artificial social behavior extends beyond Minecraft and has been extensively studied in diverse settings. Consequently, there exist several research papers exploring artificial social behavior in various contexts, as well as numerous games and studies that incorporate social behavior.

Historically, classical approaches to artificial social behavior have employed algorithmic techniques. These methods dictate the agent's responses based on predefined rules, such as reacting in a specific way if the agent is angry and if a player performs a particular action. However, a more recent trend has emerged, favoring the utilization of Large Language Models (LLMs). LLMs possess the distinct advantage of being trained on vast quantities of data, including written social interactions. This enables them to simulate human-like responses by leveraging contextual factors such as mood, prior conversations, and more.

3.2.1 FAtiMA Toolkit [43]

The FAtiMA (Fearnot's Agent that is Multimodal and Adaptive) Toolkit is a comprehensive collection of tools and assets designed to facilitate the creation of characters with social and emotional intelligence. It has been developed by "GAIPS – Group of Artificial Intelligence for People and Society" [28], which is a "research group on agents and synthetic characters at INESC-ID" [28] [29]. Building upon the advancements made in the development of the FAtiMA agent architecture, the toolkit offers a range of functionalities to enhance character behavior.

"FAtiMA Toolkit is a collection of tools/assets designed for the creation of characters with social and emotional intelligence. The project is the result of the work that was developed under an EU-funded project named RAGE and an FCT funded project named Slice. As implied by its name, the toolkit is a continuation of the work done in developing the FAtiMA agent architecture." [30]

One notable component of the FAtiMA Toolkit is the "FAtiMA-Toolkit Authoring Tool," [31] which provides users with a user-friendly interface for creating scenarios and defining emotional characteristics for characters. This tool enables users to customize the responses of characters based on specific player interactions, taking into account factors such as the character's mood. Figure 11 illustrates an example of the FAtiMA-Toolkit Authoring Tool.

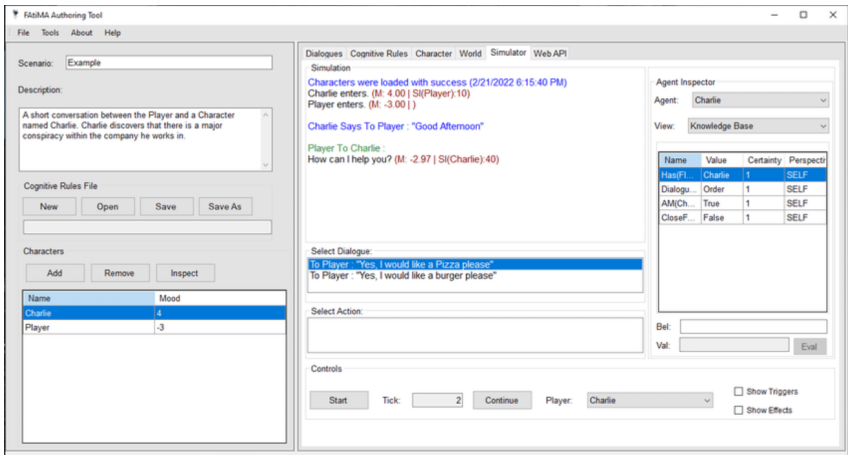

Figure 11: FAtiMA-Toolkit Authoring Tool example

With the FAtiMA Toolkit and its Authoring Tool, creators gain access to a comprehensive set of resources to infuse their NPCs with nuanced social and emotional behaviors. This results in a more immersive and engaging gameplay experience for players. Leveraging logic inference, the toolkit enables characters to dynamically adapt their responses and actions based on various situational factors, thereby enhancing the depth and realism of their interactions within the environment.

Once the character and scenario data (possible player input, NPC responses, etc) are specified, they can be saved into files. For a game to interpret these files, it must use the FAtiMA-Engine as an external library in its development engines such as Unreal Engine [32] and Unity [33].

3.2.2 Space Modules Inc. [34]

Space Modules Inc. is an engaging game developed by Playgen, an esteemed game development company [35]. In this unique game, players assume the role of help desk employees working for a prominent spaceship part manufacturer. Their primary responsibility is to assist customers in resolving their problems by employing effective questioning techniques to gather crucial information and identify appropriate solutions. The success of players hinges on their ability to maintain a friendly demeanor and skillfully manage customer emotions to ensure that interactions do not end abruptly.

Figure 12 provides a glimpse into the gameplay experience of Space Modules Inc., showcasing the intricacies of customer interactions within the game.

Figure 12: Space Module Inc. example

To facilitate the creation of realistic and engaging social interactions, the developers utilized the FAtiMA framework within the Unity [33] game engine. FAtiMA's powerful capabilities enabled the design of dynamic player responses and the realistic modeling of customer reactions based on those responses. This integration of FAtiMA in Unity demonstrates the vast potential of the framework in constructing lifelike dialogues that simulate social interactions.

Space Modules Inc. serves as a compelling testament to the effectiveness of FAtiMA in creating rich and immersive social experiences within gaming environments. By leveraging the power of FAtiMA, game developers can infuse their creations with sophisticated social interaction mechanics, enhancing player engagement and immersion.

3.2.3 Large Language Models [36]

Large language models (LLMs) serve a multitude of purposes, encompassing tasks ranging from assisting humans in problem-solving to generating text. Due to their extensive training on vast amounts of human-written text, LLMs have acquired a deep understanding of social interactions documented in written form. Consequently, certain LLMs demonstrate a remarkable ability to predict human behavior in response to social stimuli, taking into account the individual's emotional state. This observation led to the hypothesis that LLMs could effectively simulate the social behavior of NPCs. To investigate this hypothesis, I conducted an experiment using one of the LLMs.

However, it is important to note that LLMs are still relatively new and undergoing active experimentation.

Consequently, the availability of LLMs to the general public is limited, and some models have complex setups or rely on outdated Python libraries, making them challenging to utilize. For instance, the YaLM-100B language model demands substantial disk space and GPU memory, rendering it impractical for typical PC users due to its resource-intensive requirements.

The current landscape of LLMs for public access remains somewhat restricted, necessitating careful consideration and selection of appropriate models for research and practical applications involving social behavior simulations. Nonetheless, as LLM technology continues to advance, we can anticipate more accessible and user-friendly models that enable wider exploration and utilization of their capabilities.

ChatGPT [38] is a prominent example of a LLM developed by OpenAI [37]. It has been trained using the Reinforcement Learning from Human Feedback (RLHF) approach. The model itself is not open-source. However, OpenAI provides a convenient web interface for interacting with ChatGPT, accessible through their website (https://chat.openai.com/), and offers libraries in multiple programming languages (https://platform.openai.com/docs/libraries) that allow developers to integrate ChatGPT into their applications. It's worth noting that an API key is required to make queries using these libraries.

To assess the viability of using ChatGPT for simulating human behavior in NPCs, I conducted a test within ChatGPT's web interface, as depicted in Figure 13.

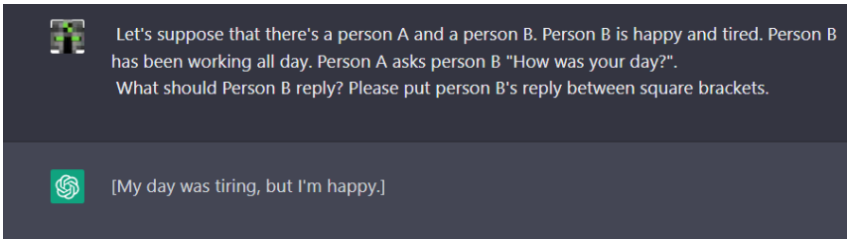

Figure 13: Test of human-like dialog with ChatGPT

Additional tests were conducted under different circumstances, consistently yielding realistic and context-aware responses. However, due to the page limit of this thesis, further examples will not be presented.

This test demonstrates a significant advantage of ChatGPT over FAtiMA: fully dynamic dialog. While FAtiMA is restricted to predefined dialog choices for players and pre-programmed responses from NPCs, ChatGPT allows unrestricted player input and highly dynamic NPC replies.

Furthermore, to align with my broader goals beyond dialog generation, I expanded the prompts, as illustrated in Figure 14.

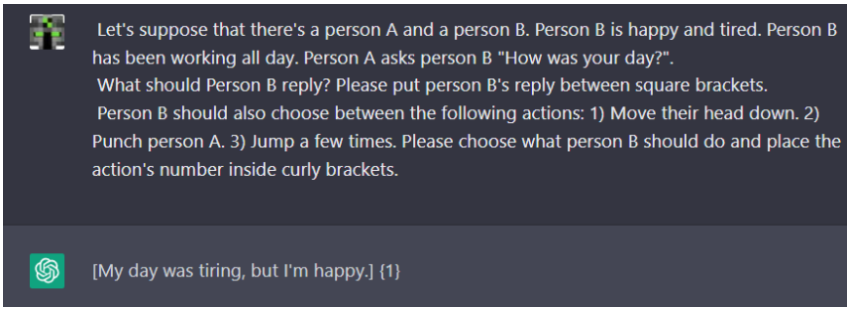

Figure 14: Test of dialog with action choice using ChatGPT

Notably, ChatGPT not only generates consistent and realistic replies but also selects the most appropriate actions based on the contextual situation. This capability indicates immense potential for integrating ChatGPT into NPC behavior systems.

It is worth mentioning that the tests conducted involved ChatGPT-3, OpenAI's ChatGPT model available to the public. ChatGPT-4 [39], its successor, has shown significant improvements in understanding context and generating even better responses. However, access to ChatGPT-4 requires the API key holder to have a monthly subscription, although OpenAI has plans to make it available for free to the public.

3.2.4 Generative Agents: Interactive Simulacra of Human Behavior [42]

The utilization of ChatGPT for choosing NPC dialog and actions is already being employed in simple games, as demonstrated in this video. It showcases interactions between NPCs in a 2D world, where NPCs attend a birthday party organized through word-of-mouth invitations, highlighting ChatGPT's ability to handle context effectively. This further validates the hypothesis and underscores the potential of using ChatGPT for simulating social behavior in NPCs.

Figure 15: The paper's first figure

Researchers from the United States conducted this project utilizing ChatGPT to determine the dialog and actions of agents by formulating the agent's context as a prompt and querying ChatGPT for guidance. To maximize the information richness of the ChatGPT prompts, the researchers incorporated the agent's memory, which was initially established through a natural language paragraph explicitly describing the agent, and the agent's reflections, representing a more abstract form of memory, encompassed one or more inferred observations with the inference process also facilitated by ChatGPT.

```
[Agent's Summary Description]
It is February 13, 2023, 4:56 pm.
John Lin's status: John is back home early from
work.
Observation: John saw Eddy taking a short walk
around his workplace.
Summary of relevant context from John's memory:
Eddy Lin is John's Lin's son. Eddy Lin has been
working on a music composition for his class. Eddy
Lin likes to walk around the garden when he is
thinking about or listening to music.
Should John react to the observation, and if so,
what would be an appropriate reaction?
```

Figure 16: Example of a prompt given to ChatGPT

This project closely aligns with the objectives of my thesis, particularly in terms of achieving socially adept agents. I aspire for my agents to exhibit behavior akin to the ones showcased in this project. Although the project's source code is not publicly available, I intend to draw inspiration from their architectural explanations and provided examples to inform the development of my own agent system.

3.2.5 Discussion

While FAtiMA is a valuable tool for creating social interactions, its scope is limited primarily to conversations. In the case of MineSocieties, NPCs are expected to engage in a broader range of activities, including gift-giving and forming relationships. Consequently, if FAtiMA were employed, it would solely serve the dialog aspect of MineSocieties. Moreover, utilizing FAtiMA would hinder the development of MineSocieties as a Java-based Minecraft mod or plugin since FAtiMA lacks Java compatibility. On the other hand, ChatGPT-3 exhibits significant promise by not only generating dialog but also selecting contextually appropriate actions. By utilizing the ChatGPT API, I can request ChatGPT to suggest suitable actions based on a given set of options and the NPC's context written in natural language. This approach resolves the challenge of ensuring NPCs respond in a manner consistent with their respective contexts. If feasible, I aim to employ ChatGPT-4 or even ChatGPT-5, provided it becomes available in time for the thesis submission.

Consequently, I have opted to leverage ChatGPT to simulate social behavior by utilizing its capabilities to generate dialog and choose actions from a predefined list. During the implementation phase, I intend to draw inspiration from the "Generative Agents: Interactive Simulacra of Human Behavior" project due to its successful attainment of compelling NPC social behavior and its innovative ideas.

4 Solution

4.1 Solution Overview

The proposed solution is a Minecraft plugin that operates on a Minecraft server. My approach involves leveraging ChatGPT as the decision-making engine for all Agents. To accomplish this, the contextual information in which an Agent finds itself must be translated into natural language text. The text must be formatted in a manner that conveys the necessary information to ChatGPT, which will in turn provide a response that can be translated into an in-game action. EntityUtils will be used to deploy agents and execute the decided actions. A lot of actions, such as going mining, have not been implemented, so I'll have to search for existing algorithms or come up with them myself to make sure that NPCs have a good variety of actions to choose from.

4.2 Device Arquitecture

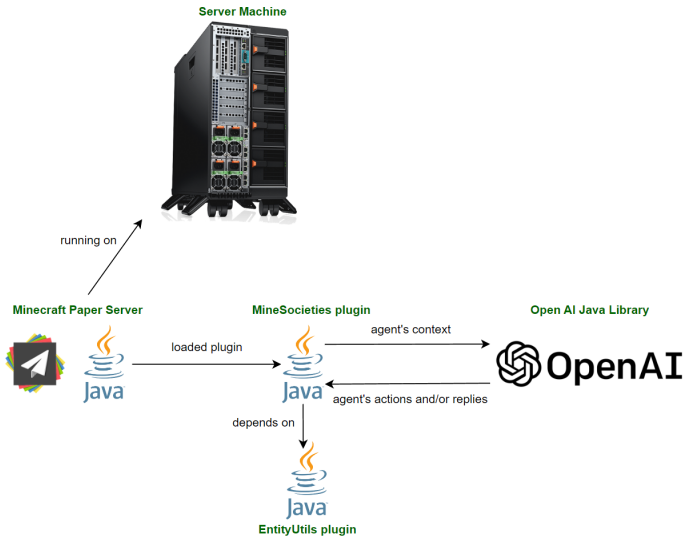

Figure 17: Solution Overview

The proposed architecture is illustrated in Figure 17. The server machine can be a dedicated machine or a regular computer with internet access.

The solution plugin, named MineSocieties, operates on the machine using the Paper [40] Minecraft server plugin API. Paper acts as both a library for plugin development and a means of optimizing base Minecraft code. MineSocieties is loaded by Paper from a specific folder, as mandated by Paper's plugin loading protocol.

MineSocieties is dependent on both Paper and OpenAI's Java API. Communication with ChatGPT is handled entirely by the OpenAI API, which abstracts away any networking-related implementation details.

4.3 MineSocieties implementation

4.3.1 Conversion of an Agent's context into natural language

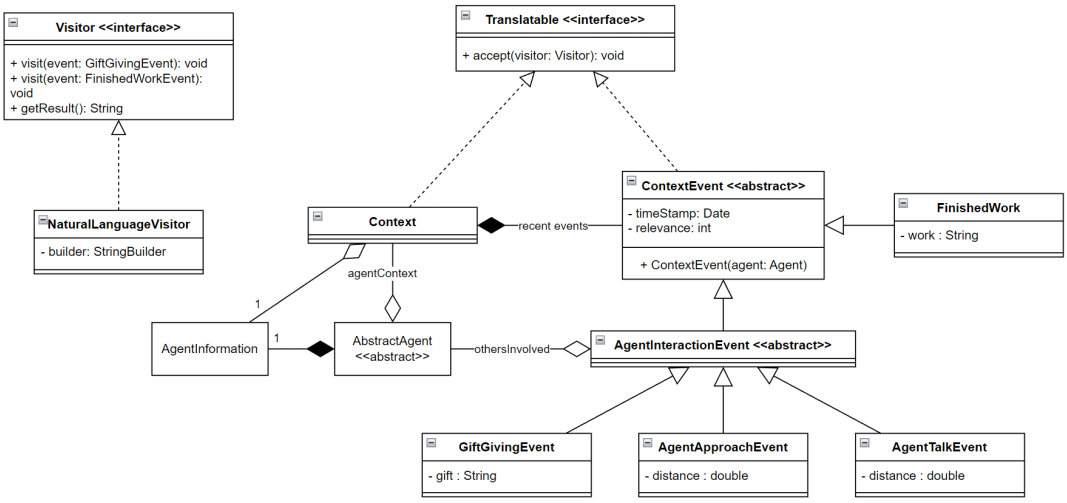

Figure 18: Translation UML

Figure 18 depicts the Unified Modeling Language (UML) diagram of the classes responsible for storing the Agent's context and translating it into natural language.

MineSocieties will incorporate an internal representation of an Agent's context, through the Context class, which collects ContextEvent. There are two example event classes provided here. For instance, if an Agent has finished work, a FinishedWork instance is saved in its Context class alongside the timestamp at which this event took place. Another instance is the GiftGivingEvent, which gets stored when this Agent gives a gift to another Agent, and the other Agent could store a similarly implemented GiftReceivingEvent.

A possible optimization is to distinguish Agent-bound events from Global events. For example, if it starts raining, it's unnecessary to create as many rain events as there are agents.

The Visitor Design Pattern can be employed since there might be more reasons why one would like to inquire about an Agent's recent events. Currently, the only concrete Visitor is the NaturalLanguageVisitor, which is capable of converting concrete events into textual descriptions. For instance, if this visitor were to visit an instance of FinishedWork, it could generate a description such as "Matthew has just finished work."

In addition, a "relevance" attribute is added to the ContextEvent class to represent the impact of events. For instance, if today was a very harsh workday for Matthew, the FinishedWork's relevance value would be quite high, allowing for richer decision-making on ChatGPT's end.

Finally, the agent's information, such as their age, memory, and mood, is also included.

4.3.2 Non-social Behaviour

In order to create engaging gameplay, Agents in Minecraft must do more than just socialize. They should also engage in activities that are relevant to the game, such as harvesting resources and building structures. While ChatGPT is adept at determining what an Agent should do, it is not well-suited for determining how to accomplish these tasks. Therefore, an alternative approach is necessary.

There exists a multitude of existing implementations where Agents perform actions using an algorithmic approach, with minimal use of Artificial Intelligence. In line with this, my approach will also rely heavily on algorithmic implementations for actions such as "Find a tree", "Go home", and "Fight a zombie", instead of employing Machine Learning techniques that would allow Agents to learn by themselves how to execute said actions. Not only does this approach prove to be less complex, but it also aligns with the thesis' primary focus on Social interactions, rather than visually appealing fighters, farmers, and lumberjacks.

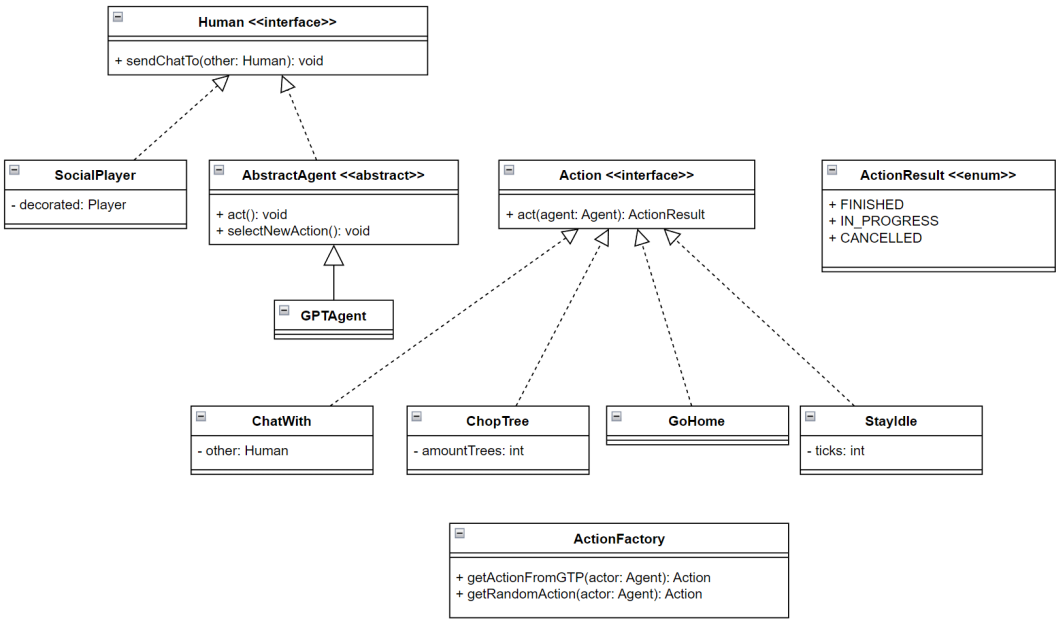

Figure 19: Actions UML

Figure 19 depicts the UML architecture for actions. At each game tick (every 0.05 seconds), all agents are given the opportunity to act, with action selection only occurring when an action has been completed, canceled, or when an event occurs. This design helps to keep ChatGPT from becoming overburdened with new action requests.

The SocialPlayer class, which employs the Decorator Design Pattern to decorate Minecraft players, is also included to represent real players in the game world. Finally, the ActionFactory serves as the primary interface between ChatGPT and the game world, providing the AI with an Agent's context and translating the responses into actionable commands for the Agents to execute.

While many more actions are possible, the example actions shown in the UML diagram provide a useful overview of the design.

4.3.3 Deploying agents

MineSocieties is designed to offer easy installation and user-friendly interactions for all server users. To achieve this goal, the process of deploying Agents must be intuitive and accessible to all.

The primary method for deploying Agents will be through a user-friendly graphical user interface (GUI), allowing players to select an Agent's traits, such as name, age, and other personality attributes, and deploy them into their Minecraft world.

For developers using MineSocieties as a dependency for their projects, deploying agents is done by calling the AgentFactory's deploy method.

To ensure that all existing Agents persist when the Minecraft server stops and resumes later, a Java Library called GSON [41] will be utilized. GSON allows for the easy serialization/deserialization of Java objects, including an Agent's context and traits, into the human-friendly JSON storage language. This will also enable server administrators to add, modify, or delete Agents by manipulating their corresponding JSON files.

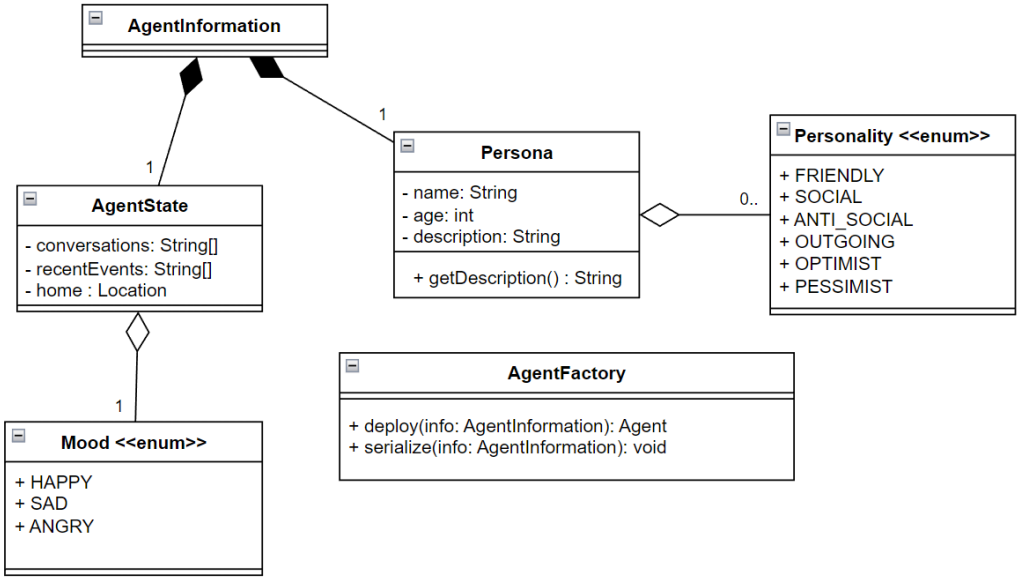

Figure 20: Agent Information

Figure 20 shows how Agent information can be represented. It can be created and modified both by editing the respective Agent's JSON files or by interacting with the in-game GUI.

Agents should hold information that is relevant for decision making, since the contents of recent conversations, events that took place, the Agent's personality and mood, and possibly more.

5 Evaluation

The success of the plugin will be assessed based on the following criteria:

- Ease of deployment and configuration: Evaluating the simplicity of setting up and modifying agents within the game,
- Creation of agent definition files: Assessing the ease of creating JSON files that define agent characteristics, enabling deployment upon server restarts,
- Developer API usability: Determining the ease with which other developers can utilize the plugin's API to create their own agents,
- Variation in agent behavior: Analyzing whether agents demonstrate diverse and non-monotonous actions,
- Human-like reactions: Evaluating the extent to which agent responses resemble those of real humans,
- Societal behavior of agent groups: Assessing whether groups of agents exhibit collective behavior akin to a society.

To conduct the evaluation, volunteers will be invited to participate in specific tests and provide feedback through customized Google Forms. For example, to assess the ease of deploying and configuring agents in-game, participants will be given a scenario where they read the deployment documentation and attempt to create agents with predefined characteristics. They will join a pre-created server, execute necessary commands, and interact with relevant GUIs. The Google Form responses will include questions such as: "How clear was the documentation?" and "Did you find the configuration GUI intuitive?" Another evaluation approach involves logging all agent decisions and engaging psychologist volunteers to review the logs. These psychologists will assess the human-likeness of agent reactions and provide feedback through a dedicated Google Form.

The combination of responses from all the Google Forms will contribute to the measurement of the following dimensions:

- Player experience: Evaluating the overall satisfaction and user-friendliness of the plugin from the players' perspective,
- Social believability: Assessing the extent to which the agents' social behavior is perceived as realistic and believable,
- Configurability: Assessing the level of customization and flexibility available in defining agent characteristics and behavior,
- Performance: Evaluating the impact of the plugin on the server's performance, including resource consumption and responsiveness,
- Scalability: Analyzing the plugin's ability to handle a large number of agents simultaneously without significant degradation in performance,
- Stability: Assessing the reliability and robustness of the plugin in terms of avoiding crashes, errors, or unintended behaviors.

6 Work Schedule

The planned schedule for the project is outlined as follows:

- Creation of the base MineSocieties plugin and setup of a Paper server.
  - Duration: 3rd June 2023 - 4th June 2023
- Integration of EntityUtils and development of a simple deployment command.

  - Duration: 5th June 2023 - 8th June 2023
- Creation of abstract and concrete representations of agents and actions.
  - Duration: 9th June 2023 - 21st July 2023
- Development of a representation of the agent's context.
  - Duration: 21st July 2023 - 29th July 2023
- Integration of OpenAI's Java library and implementation of an efficient querying mechanism for ChatGPT.
  - Duration: 29th July 2023 - 1st August 2023
- Iteration and refinement of interactions with ChatGPT, including translation of agent context into appropriate prompts, interpretation of responses for generating actions and dialog, and potential prompts for obtaining agent reflections (similarly to what Generative Agents 3.2.4 does) and daily summaries.
  - Duration: 2nd August 2023 - 25th August 2023
- Evaluating the project.
  - Duration: 1st September 2023 - 30th September 2023

Please note that the schedule may be subject to adjustments and refinements as the project progresses. Regular monitoring and coordination of tasks will be essential to ensure timely completion and successful execution.

Bibliography

[1] Minecraft, https://www.minecraft.net/. Accessed 26 May 2023.

[2] Mojang Studios, https://en.wikipedia.org/wiki/Mojang_Studios. Accessed 10 May 2023.

[3] List of best-selling video games, https://en.wikipedia.org/wiki/List_of_best-selling_video_games. Accessed 10 May 2023.

[4] Minecraft Editions, https://help.minecraft.net/hc/en-us/articles/360034753992. Accessed 10 May 2023.

[5] Minecraft Java Edition, https://minecraft.fandom.com/wiki/Java_Edition. Accessed 10 May 2023.

[6] Minecraft Plugins, https://minecraft-archive.fandom.com/wiki/Plugins. Accessed 10 May 2023.

[7] Minecraft Bukkit Library, https://bukkit.fandom.com/wiki/Main_Page. Accessed 10 May 2023.

[8] Minecraft Mods, https://minecraft.fandom.com/wiki/Mods. Accessed 10 May 2023.

[9] Minecraft Forge Mod Library, https://minecraftuniverse.fandom.com/wiki/Minecraft_Forge. Accessed 10 May 2023.

[10] Minecraft Bedrock Edition, https://minecraft.fandom.com/wiki/Bedrock_Edition. Accessed 10 May 2023.

[11] LiteLoaderBDS, https://github.com/LiteLDev/LiteLoaderBDS. Accessed 10 May 2023.

[12] Minecraft Bedrock Dedicated servers, https://minecraft.fandom.com/wiki/Bedrock_Dedicated_Server. Accessed 10 May 2023.

[13] Minecraft Modding, https://en.wikipedia.org/wiki/Minecraft_modding. Accessed 10 May 2023.

[14] Minecraft Education Edition, https://minecraft.fandom.com/wiki/Minecraft_Education. Accessed 10 May 2023.

[15] Minecraft Bedrock built-in NPCs, https://learn.microsoft.com/en-us/minecraft/creator/documents/createnpcs. Accessed 22 May 2023.

[16] MineColonies mod for Minecraft. CurseForge, https://www.curseforge.com/minecraft/mc-mods/minecolonies. Accessed 11 Apr. 2023.

[17] MinecraftComesAlive Reborn mod for Minecraft. CurseForge, https://www.curseforge.com/minecraft/mc-mods/minecraft-comes-alive-reborn. Accessed 11 Apr. 2023.

[18] Citizens plugin for Minecraft. SpigotMC, https://www.spigotmc.org/resources/citizens.13811/. Accessed 23 Apr. 2023.

[19] Sentinels plugin for Minecraft. SpigotMC, https://www.spigotmc.org/resources/sentinel.22017/. Accessed 23 May. 2023.

[20] EntityUtils plugin for Minecraft. GitHub, https://github.com/Yoursole1/EntityUtils. Accessed 24 May. 2023.

[21] The EVNT company. LinkedIn, https://www.linkedin.com/company/evntgames. Accessed 24 May. 2023.

[22] Hypixel server network. Hypixel, https://hypixel.net/. Accessed 23 May. 2023.

[23] Hypixel Studios. https://hypixelstudios.com/. Accessed 23 May. 2023.

[24] MineFlayer. https://github.com/PrismarineJS/mineflayer. Accessed 24 May. 2023.

[25] Project Malmo. Microsoft, https://www.microsoft.com/en-us/research/project/project-malmo/. Accessed 24 May. 2023.

[26] Project Malmo. GitHub, https://github.com/Microsoft/malmo. Accessed 24 May. 2023.

[27] FAtiMA Toolkit, https://fatima-toolkit.eu/. Accessed 25 May. 2023.

[28] FAtiMA Toolkit about us. FAtiMA, https://fatima-toolkit.eu/home/about-us/. Accessed 26 May. 2023.

[29] INESC-ID, https://www.inesc-id.pt/. Accessed 26 May. 2023.

[30] FAtiMA Toolkit. GitHub, https://github.com/GAIPS/FAtiMA-Toolkit. Accessed 25 May. 2023.

[31] FAtiMA Toolkit getting started tutorial. FAtiMA, https://fatima-toolkit.eu/home/get-started/. Accessed 25 May. 2023.

[32] Unreal Engine, https://www.unrealengine.com/. Accessed 25 May. 2023.

[33] Unity, https://unity.com/. Accessed 25 May. 2023.

[34] Space Modules Inc. Game Components, https://www.gamecomponents.eu/content/351. Accessed 26 May. 2023.

[35] Playgen, https://playgen.com/. Accessed 26 May. 2023.

[36] List of large language models. Wikipedia, https://en.wikipedia.org/wiki/Large_language_model#List_of_large_language_models. Accessed 26 May. 2023.

[37] Open AI, https://openai.com/. Accessed 26 May. 2023.

[38] ChatGPT. Open AI, https://openai.com/blog/chatgpt. Accessed 26 May. 2023.

[39] ChatGPT-4. Open AI, https://openai.com/product/gpt-4. Accessed 26 May. 2023.

[40] PaperMC, a Plugin API. PaperMC, https://papermc.io/. Accessed 17 Apr. 2023

[41] GSON, a Java JSON serialization/deserialization Library. GitHub, https://github.com/google/gson. Accessed 19 Apr. 2023

[42] S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," arXiv preprint arXiv:2304.03442, 2023.

[43] Mascarenhas, M. Guimarães, R. Prada, P. A. Santos, J. a. Dias, and A. Paiva, "Fatima toolkit: Toward an accessible tool for the development of socio-emotional agents," ACM Trans. Interact. Intell. Syst., vol. 12, no. 1, mar 2022. [Online]. Available: https://doi.org/10.1145/3510822

[44] C. Marques, "Social agents in minecraft," Av. Rovisco Pais 1049-001 Lisboa, 2022.

[45] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The malmo platform for artificial intelligence experimentation." in Ijcai, 2016, pp. 4246–4247.